OPTIMIZATION OF CHEMICAL PLANT SIMULATION USING DOUBLE COLLOCATION

Josef Ladislaus Illes

Doctor of Philosophy

The University of Aston in Birmingham

April 1993

Thesis Summary

The University of Aston in Birmingham

OPTIMIZATION OF CHEMICAL PLANT SIMULATION USING DOUBLE COLLOCATION

Josef Ladislaus Illes

Doctor of Philosophy

1993

A method has been constructed for the solution of a wide range of chemical plant simulation models including differential equations and optimization.

Double orthogonal collocation on finite elements is applied to convert the model into an *NLP* problem that is solved either by the *VF13AD* package based on successive quadratic programming, or by the *GRG2* package, based on the generalized reduced gradient method. This approach is termed simultaneous optimization and solution strategy. The objective functional can contain integral terms. The state and control variables can have time delays. Equalities and inequalities containing state and control variables can be included into the model as well as algebraic equations and inequalities. The maximum number of independent variables is 2. Problems containing 3 independent variables can be transformed into problems having 2 independent variables using finite differencing. The maximum number of *NLP* variables and constraints is 1500. The method is also suitable for solving ordinary and partial differential equations. The state functions are approximated by a linear combination of Lagrange interpolation polynomials. The control function can either be approximated by a linear combination of Lagrange interpolation polynomials or by a piecewise constant function over finite elements. The number of internal collocation points can vary by finite elements. The residual error is evaluated at arbitrarily chosen equidistant grid-points, thus enabling the user to check the accuracy of the solution between collocation points, where the solution is exact. The solution functions can be tabulated.

There is an option to use control vector parameterization to solve optimization problems containing initial value ordinary differential equations. When there are many differential equations or the upper integration limit should be selected optimally then this approach should be used. The portability of the package has been addressed converting the package from *VAX FORTRAN 77* into *IBM PC FORTRAN 77* and into *SUN SPARC 2000 FORTRAN 77.*

Computer runs have shown that the method can reproduce optimization problems published in the literature. The *GRG2* and the *VF13AD* packages, integrated into the optimization package, proved to be robust and reliable.

The package contains an executive module, a module performing control vector parameterization and 2 nonlinear problem solver modules, *GRG2* and *VF13AD*. There is a stand-alone module that converts the differential-algebraic optimization problem into a nonlinear programming problem.

key words: differential-algebraic, optimal control, partial differential equations

Dedication

To the memory of my parents, Helen Preisinger and József Illés, and of my friend, Illés Drabik, chemical engineer, and of Margit Farkas, 1st year student in 1956 at Veszprém University.

Acknowledgements

I wish to express my heartfelt thanks to :

A woman secretary at the Personnel Department in the Ministry of Heavy Industry in Budapest for signing, in spite of her superiors' opposition for reasons of my involvement in the 1956 uprising, which also made it next to impossible for me to get a proper job in chemical engineering, a paper, thus making it possible for me to study applied mathematics part-time;

an official of the Personnel Department in the Ministry of Heavy Industry in Budapest for getting me my first job as a chemical engineer at Köbányai Fütöerömüvek;

Sándor Ferenczi, chemical engineer at Kelenföldi Höerömü and my former chief, a lecturer at Budapesti Müszaki Egyetem, Kémia Technológia Tanszék, Dr. Pál Benedek, my former Professor at Veszprém University and later my chief at MAVEMI, Dr. Tibor Hoffman, my former chief at MAVEMI, for starting my career in computing;

Alan Jeffries, my former manager at SIA Ltd, for getting me my first job in computing in Britain; Peter King, Professor of Computer Science at Birkbeck College, for accepting me for a part-time M.Sc. course in computer science;

Dr. John Fletcher, my supervisor, for his direction, for his constant availability and for his contribution to this research;

Dr. William R. Paterson, my external examiner, for his time, interest, valuable suggestions and for reviewing this work;

Professor Lasdon and Mr. Marlow for sending me nonlinear programming codes;

Dr. Sarma, Dr. Stadtherr and Dr. Masi for sending me useful material;

Lajos Dienes, chemical engineer and my friend, for having aroused my interest in the optimization of chemical processes through our joint investigation into optimizing Fuller earth manufacturing at the Nagytétény chemical plant of Budapesti Vegyimüvek;

my wife, Irene, for her love, understanding, encouragement, patience and total support throughout all my part-time studies in England;

my parents, Helen Preisinger and József Illés, for their total support, understanding, encouragement and patience throughout my many years of school in Hungary and for being there when I needed them and for their love.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Optimization

Optimization is a procedure for problem solving, where the problem can be stated as the minimization or maximization of given economic or technological measures. The measures to be minimized or maximized must each be put into mathematical form, termed an objective function or objective functional. These objective functions or functionals form part of the mathematical model of the problem. Objective functions or objective functionals are also known as performance index. In an objective functional at least one variable is dependent on time or a spatial independent variable. This implies that there is at least one differential equation in the model, and the problem is described as a differential-algebraic optimization problem.

An optimization problem is formulated as a mathematical model consisting of an objective function or objective functional and constraints. The model is a set of algebraic and/or differential equations and inequalities which approximates the true behaviour of the system. A system is a part of the real world that is being studied. In this work a system is normally a chemical process. The model can contain algebraic equations and inequalities, equations and inequalities containing variables dependent on one or two independent variables, ordinary differential equations, partial differential equations having 2 independent variables, an objective functional that may contain integral terms. The work involved is the application of optimization to the solution of chemical engineering problems.

This thesis extends, and was inspired by, the current methods capable of solving differential-algebraic optimization problems described by ordinary differential equations and algebraic equations and inequalities [72-76] to a method capable of solving problems described by first and second order partial differential equations and inequalities containing no more than 2 independent variables and algebraic equations and inequalities. Although emphasis was put on solving differential-algebraic optimization problems, mathematical programming problems also can be solved by the package.

1.2 Models in Chemical Engineering

The following table presents some typical models in chemical engineering [85].

Table 1.1 Models in Chemical Engineering

EQUATION TYPE	MATHEMATICAL MODEL
Algebraic equations	Equilibrium conditions of a chemical reactor. Steady-state of a CSTR. Distillation column.
Ordinary differential equations with initial conditions	Batch reactor. Transient of a CSTR. Steady state of a plug-flow reactor.
Ordinary differential equations with boundary conditions	Steady state of a gas-liquid bubble-column reactor. Steady state of a tubular reactor with axial dispersion. Steady state diffusion with reaction in a catalyst particle.
Partial differential equations in two independent variables	Transient of a tubular reactor with axial dispersion. Steady state of a heterogenous tubular reactor or of a homogeneous one with radial and axial dispersion. Dynamics of an adsorption bed.
First-order partial differential equations	mass-balance catalyst decline Heat-balance in a plug-flow tubular reactor
Hyperbolic differential equation	Wave equation $$\frac{\partial^2 c}{\partial t^2} - a^2 \frac{\partial^2 c}{\partial x^2} = 0$$

Parabolic differential equation	Convective time-space diffusion $$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2}$$
Elliptic differential equation	$$\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} = f$$ If f = 0 flow of a non-viscous fluid in a pipe If f ≠ 0 flow of a viscous fluid in a pipe Steady-state heat flow Steady-state diffusion
Partial differential equations in three independent variables	Transient of a tubular reactor with radial and axial dispersion. Transient of the thermal field of a heated bidimensional body.

Important areas of optimization in chemical engineering are flowsheet optimization and optimal control. When the plant already exists, the optimization can be carried out without knowing or making use of the mathematical model of the plant, using the output variables of the plant to evaluate the objective function and adjusting the control variables according to a certain optimization algorithm. This approach is known in the literature as EVOP (evolutionary optimization). When the plant already exists and the model of the plant exist, the model of the plant can be optimized. When the plant does not exist, to perform the optimization, the model of the plant is needed. Flowsheet optimization is a steady-state optimization, the model contains only algebraic constraints, the problem is a mathematical programming problem. In case of optimal control, the model contains differential equations. It can also contain differential inequalities. The analytical solution of the equations that constitute the model is usually obtainable only for very simplified models. Therefore the optimization problem has to be solved numerically for each particular case. The objective functionals can be classified into 2 classes, according to whether at least one control(optimizing) variable is dependent upon the independent variable. A control variable is a variable or a function of the independent variables that should be chosen optimally to maximize or minimize the objective functional. A state variable is a variable or a function of the independent variables that cannot be chosen freely and the value of which is determined by the equations and control variables. If they are functions of independent variables, their derivatives occur in the model. When at least one control variable is dependent upon the independent variable, the optimization is carried out in the infinite dimensional function space. The theory for the solution of such problems is found in the field of variational calculus.

1.3 Methods of Solution

This research is concerned with optimization in the finite dimension Euclidean space, because optimization in the infinite dimension function space cannot be fully automated; if to solve optimization problems having partial differential equations in the model control vector iteration is employed, then to obtain the *Euler-Lagrange* equations explicitly, analytical differentiation of the integrand of the objective functional and the state equations is required and the model can have only simple bounds on the control variables, but cannot have general equality or inequality constraints. This work is aimed at widening the class of problems that can be solved as well as at providing an alternative method to solve differential-algebraic optimization problems containing partial differential equations. In order to carry out the optimization in the finite dimension *Euclidean* space, some kind of transformation has to be applied to the model. Three kinds of transformations will be applied to convert a variational problem into an optimization problem in the finite dimension Euclidean space. The transformations described in (a) and (b) result in a sequential optimization strategy, where one performs an unconstrained optimization, while for each function evaluation the differential-algebraic constrains have to be solved. The transformation described in (c) results in a nonlinear programming (NLP) problem, and this approach is termed a simultaneous simulation-optimization strategy.

a) all control variables are regarded as constant throughout the integration. For example, if optimal temperature progression to maximize the yield of a batch reactor is to be found, the best isothermal yield will be found instead.

b) control vector parameterization is used, e.g. instead of T(t), say, $T(t)=a_0+a_1e^{-a_2t}$ or $T(t)=a_0+a_1t+a_2t^2$ is used and a_0, a_1, a_2 are chosen optimally. Here the infinite dimensional problem has been reduced into a three dimensional one.

c) orthogonal collocation on finite elements is used. The approximating function is a piecewise continuous polynomial joined at knot points. The polynomials are usually Lagrange or Hermite interpolation polynomials [86]. The collocation(interpolation) points are chosen to be the zeros of a Legendre polynomial. The coefficients of the interpolation polynomials are values of the continuous dependent variables (state or control variables) and will be the unknowns of the resulting system of algebraic equations to be satisfied at the collocation points. The control variables can also be approximated by piecewise constant functions defined over the integration domain. Both the coefficients of the interpolation polynomials (values of the dependent state or control variables) and the piecewise constant values of the control variables will be variables of the NLP problem. Thus orthogonal collocation on finite elements transforms the objective functional into an objective function and the differential-algebraic constraints into constraints of a NLP problem.

The following combinations of objective functions and model equations can occur:

Table 1.2 Objective Functions versus Constraints

	objective function	objective functional
algebraic constraints	feasible	not feasible
differential-algebraic constraints	unlikely	feasible

The objective function - algebraic constraints combination is a mathematical programming problem. The other combinations are differential-algebraic optimization problems.

There are certain special features encountered in chemical processes:

The variables are mostly continuous and in most cases both the objective function and constraints are nonlinear with a high degree of nonlinearity. There are a large number of equality constraints. The inequality constraints are usually of the simple bound type. The models have relatively few degrees of freedom. The majority of models can only be described by differential-algebraic equations. Recycling occurs very frequently, often feeding unconsumed reactant back to the reactor. An algebraic equation is nonlinear if the coefficient of at least one variable is not constant. A differential equation is nonlinear, if the coefficient of at least one derivative is function of derivatives. Degree of freedom is equal to n_v - n_e, where n_v is the number of variables and n_e is the number of equations. Chemical process models can also be distinguished according to whether the model is deterministic or stochastic. A stochastic model contains variables subject to random disturbances and measurement errors. Optimization can be carried out on-line, when there is interaction between the plant and the optimization package or off-line.

The work in this thesis is concerned with deterministic, off-line, continuous variable optimization problems. Only real-valued functions and variables are considered. The area of optimization tackled is optimization of problems described by differential-algebraic equations. Optimization theory and algorithms will only be described as they pertain to the class of optimization problems to be tackled and to optimization algorithms judged to be most successful for chemical process optimization in the literature. Techniques currently exist to solve optimization problems containing *PDE*s with 2 independent variables. The package implemented in this work can solve optimization problems containing *PDE*s with 2 independent variables, differential inequalities, algebraic equations and inequalities.

1.4 Thesis Outline

Chapter 3 describes the theoretical foundations of control vector parameterization, of orthogonal collocation on finite elements, of generalized reduced gradient method and of successive quadratic programming. The control vector parameterization, although well established, was included, because for a certain class of problems, i.e., when there are many initial value ordinary differential equations, it is more advantageous to use control vector parameterization, than orthogonal collocation in conjunction with nonlinear programming, and if the upper integration limit is an optimizer variable, then control vector parameterization is the only method to solve the optimization problem.

Chapter 4 describes how the model is transformed into a nonlinear programming problem using orthogonal collocation on finite elements. The class of problems that can be solved is defined. Then the discretization of differential equations is described, how in the model independent variables, state variables, their derivatives, control variables, auxiliary variables and input variables are transformed resulting in nonlinear algebraic equations. Then time delay handling and parameter estimation and irregular domain handling is described. Expressions, initial and boundary conditions, objective functionals and constraints are also transformed into algebraic equations and inequalities.

Chapter 5 describes the definition of the state and control functions in terms of the *NLP* optimal solution and how the residual error is evaluated.

Chapter 6 describes the implementation of the optimization package.

Chapter 7 deals with the analysis of 8 example problem runs including 2 differential-algebraic optimization problems, 2 dynamic simulation problems, a time delay problem, a parameter estimation problem, a nonlinear programming problem and a *PDE* with 3 independent variables. Except for the nonlinear programming problem, they all were solved both by the generalized reduced gradient method and by successive quadratic programming. The nonlinear programming problem was solved only be the generalized reduced gradient method. The parameter estimation problem was also solved by control vector parameterization. Example 8 was solved only using successive quadratic programming, because it ran faster on this problem than *GRG2*.

Chapter 8 contains discussion, conclusions and recommendations for future work.

The appendices contain the user guide and the program documentation to facilitate maintenance and further enhancement of the nonlinear programming problem generating program.

Chapter 2

Literature Review

2.1 Introduction

A number of important process optimization problems are described by models containing ordinary or partial differential equations. These cover such applications as optimal control, optimization of batch or continuous chemical processes, parameter estimation of dynamic systems, etc. The optimization of dynamic systems can be formulated as a differential-algebraic optimization problem.

2.2 Classification of Problems

Differential-algebraic optimization problems (*DAOPs*) can be classified by whether they contain ordinary differential equations (*ODEs*) or partial differential equations (*PDEs*). *DAOPs* described by *ODEs* are referred to as lumped parameter systems. *DAOPs* described by *PDEs* are referred to as distributed parameter systems. Systems can be classified also by whether or not there is time-delay in any state or control variable. *DAOPs* having more than one state variable are termed multivariable systems. Most of the systems are multivariable. *DAOPs* can also be classified as follows:

1. Linear systems, where the model is linear.
2. Linear-quadratic systems, where the objective functional is quadratic and the state of the system can be represented by a matrix equation. If every matrix in the model is constant, it is called an autonomous system.
3. Nonlinear systems, where either the objective functional or the constraints are nonlinear.

These 3 classes may be further subdivided by the order of the differential equations and by the type of boundary conditions. In order to review the literature, it is useful to classify the solution methods which have been used. This will be done separately for lumped parameter and distributed parameter systems.

2.3 Classification of Solution Methods:

2.3.1 Lumped Parameter Systems

1. Generation and Solution of a linear-quadratic model, if necessary linearizing a nonlinear model.
2. Generation of approximate solution to a problem formulated using dynamic

programming.

3. Discretization of the integration domain by finite differences to create an *NLP* problem.
4. The extension of the ideas of *Liapunov* stability.
5. Dynamic Matrix Control.
6. Solution of the necessary conditions for optimality for the control vector.

   The resulting problem can be solved as follows:

   (a) using on-off control

   (b) control vector iteration

      (i) using gradient method

      (ii) using conjugate gradient method

      (iii) using quasi-Newton method

   (c) quasilinearization

   (d) using the method of weighted residuals to discretize the problem
7. Control vector parameterization
8. Orthogonal collocation coupled with quasi-Newton methods
9. Approximation of the *DAOP* by algebraic functions to form a nonlinear problem.

The methods 1-2 and 4-6 are all problem-specific, each new problem must be solved individually, the methods cannot be generalized or automated.

2.3.2 Distributed Parameter Systems:

10. Solution of the necessary conditions for optimality
11. Using the method of weighted residuals to convert the problem into a lumped parameter system. Then the resulting problem can be solved by one of the solution methods applicable to lumped parameter systems.
12. Using double orthogonal collocation on finite elements to convert the problem into a nonlinear programming problem, then solve it either by the generalized reduced gradient method or by successive quadratic programming.

The first step of method 11, lumping, can be automated, but the second step can be fully automated only if the resulting lumped parameter system is solved by methods 3 or 7-9. Optimal control problems can be solved in such a way that the control contains feedback information, i.e., state or output variables may appear in the control law. In a closed-loop control scheme there is feedback information in the control law, in an open-loop control scheme there is no feedback information in the control law.

2.4 Literature Survey

A vast body of literature is devoted to addressing the solution of differential-algebraic optimization problems. One of the first approaches to the solution of nonlinear optimal control problems was to use the properties of the linear-quadratic problem

$$\min_{u} \left[\, I = G(x(\theta)) + \int_{0}^{\theta} [x^{T}Qx + u^{T}Ru]\,]dt \right. \tag{1}$$

subject to

$$\frac{dx}{dt} = Ax + Bu \qquad x(0) = x_0 \tag{2}$$

where (2) approximates the optimal control of the more general nonlinear system

$$\frac{dx}{dt} = f(x,u,t) \qquad x(0) = x_0 \tag{3}$$

This was done by Pearson [1], Burghart [2], and Weber and Lapidus [3], reviewed in [48], by rewriting equation (3) in the form

$$\frac{dx}{dt} = A(x,t)x + B(x,t)u \tag{4}$$

and solving the linear-quadratic problem at several different values of **x**. According to Hicks and Ray [48], Burghart [2] was able to entirely pre-compute the suboptimal feedback gain by expanding the gain in a Taylor series about some reference value of **x**. To find the suboptimal solution, many computer runs had to be done with a range of reference values of **x** and for different numbers of terms used in the Taylor series expansion.

Another approach, as suggested by Durbeck [4], Garrard et al. [5], and Bukreev [6], and reviewed in [48], is to generate approximate solutions to the Hamilton-Jacobi-Bellman equations of dynamic programming to produce a suboptimal feedback control. In dynamic programming, optimization is carried out in a stagewise manner starting from the *n*-th stage and progressing backward. According to Hicks and Ray [48], these methods generally require a great deal of computation, and as shown in comparisons made by Garrard et al. [5] and Burghart [2], give somewhat worse performance than the approximation to the linear-quadratic problem discussed above.

A method suggested by Tabak and Kuo [63] is to discretize the optimal control problem by discretizing the integration domain. The derivatives are approximated by finite differences. The values of the continuous functions at grid points become variables in the resulting nonlinear programming problem.

An extension of the ideas of Liapunov Stability to generate near-optimal control policies was suggested by Koepecke and Lapidus [8], reviewed by Hicks and Ray [48]. In this case the control problem is to find the discrete control signals, $\mathbf{u}(0)$, $\mathbf{u}(1)$,..., which will move the dynamic system from the initial state $\mathbf{x}(0)$ to the origin in some optimum way. "Optimum way" is defined as a fast response with a minimum overshoot. The overshoot requirements lead to stability considerations, and hence it is considered natural to define "optimum way" around stability. The approach is to establish the control criteria on the basis of Liapunov's direct method which is a technique for the stability analysis of linear and nonlinear processes. The method is a mathematical abstraction of the concept that if the total free energy of a system is decreasing then the system must be stable. This may be implemented in the present case by finding a positive scalar function $V(\mathbf{x}) = V(x_1,x_2,\ldots,x_n)$ called the *Liapunov* function which has the property that the difference $\Delta V(\mathbf{x})$ is minimized over each sampling period by a suitable choice of the $u(k)$. This corresponds to the analogous case of a continuous system of causing the derivative of $V(\mathbf{x})$ with respect of time to be negative. The optimum control vector is that $\mathbf{u}(k)$ which makes $\Delta V(k)$ the most negative. The method of *Liapunov* is stable, simple, practical and therefore most attractive for establishing a control criterion. $V(\mathbf{x})$ is said to be positive definite with the following properties:

(a) $V(\mathbf{x})$ is continuous together with its first partial derivatives in a certain open region Ω about the origin.

(b) $V(\mathbf{0}) = 0$

(c) $V(\mathbf{x}(t))$ is positive in Ω. The origin is a local minimum of V. Along the path g of

$$\dot{x}(t) = X(x(t)) \tag{5}$$

we have

$$\dot{V} = X \cdot \nabla V \tag{6}$$

If in addition $\dot{V} \leq 0 \; in \; \Omega$, V is called a *Liapunov* function. This technique assumes an objective of the form

$$I = \int_0^\theta V(x)dt \tag{7}$$

where $V(\mathbf{x})$ can be considered a *Liapunov* function. Then the controls $\mathbf{u}$(t) are selected so that $\frac{dV(x)}{dt}$ is made as negative as possible at each point in time. The technique has been considered by Hicks and Ray [48] to produce quite good sub-optimal feedback controls [7,8]. Paradis and Perlmutter [9] are reported by Hicks and Ray [48] applying this technique to the feedback control of a tubular reactor modelled as a distributed parameter problem with reasonably good success.

Universal Dynamic Matrix Control is used by Morshedi [10], reviewed by Renfro [72]. *Universal Dynamic Matrix Control (UDMC)* differs from a conventional optimal control, where the control horizon is equal to the integration horizon, in that in *UDMC* the control moves are made only in the control horizon $[t_0,t_c]$, and not in the steady-state horizon $[t_0,t_f]$.

A much used approach to optimal control is to use the necessary conditions of Pontryagin's maximum principle, i.e., to solve the Euler-Lagrange equations and to maximize the Hamiltonian. Although *Pontryagin's Maximum Principle* is applicable for both lumped and distributed parameter systems, for simplicity, a brief outline of the method is given below for lumped parameter systems, based on [23].

Theorem. (*Weak Maximum Principle*).

In order for a control $\bar{u}(t)$, $u_* \leq \bar{u}(t) \leq u^*$ to be optimal in the sense that it maximizes the objective I in Equation (9) while satisfying the system Eqs. (8), it is necessary that Eq. (11) be satisfied for the unconstrained portion of the control trajectory and H defined by (12) be maximized along constrained portions of the control trajectory. Thus given

$$\frac{dx}{dt} = f(x,u),\ x(0) = x_0 \tag{8}$$

and

$$I[u(t)] = G(x(t_f)) + \int_0^{t_f} F(x,u)dt \tag{9}$$

the necessary conditions for $\bar{u}(t)$ to maximize

$$I[u(t)] \tag{10}$$

is that

$$\frac{\partial H}{\partial u} = 0 \tag{11}$$

on the unconstrained portion of the path and

$$H = F(x,u) + \lambda^T f(x,u) \tag{12}$$

be at the maximum on the constrained portion of the path. Here H is the *Hamiltonian* defined by Eq. (12), and λ is the time dependent *Lagrange* multiplier, which is defined by

$$\frac{d\lambda}{dt} = -\frac{\partial H}{\partial x} \tag{13}$$

and

$$\lambda_i(t_f) = \frac{\partial G}{\partial x_i} \tag{14}$$

for those state variables unspecified at t_f. Eqs (13-14) are termed *adjoint* or *Euler-Lagrange* equations.

One method to solve the above problem used quasilinearization [135], reviewed by Ray and Szekely [23]. According to Ray and Szekely [23], quasilinearization is a direct substitution approach to solve the necessary conditions for optimality equations.

The equation (11) is solved for *u(t)* explicitly. Then the solution is substituted into the state equations (8) and in the adjoint equation (13). The resulting equations are linearized about some reference trajectory **x**,λ. Then they are solved repeatedly with a better estimate of **x**,λ until the solution of the linearized equations converge to the solution of the nonlinear equations. This approach may fail to converge unless a reasonably good initial guess is available; however, according to Ray and Szekely [23], it has been used successfully on a number of practical problems [135].

Miele [12] proposed gradient algorithms for the *DAOP* problem, reviewed by Biegler and Grossmann [94]. From the first order optimality conditions, a linear two-point boundary value problem is formulated and solved at each iteration. Before beginning the next iteration, however, a different linear two point problem is formulated and solved successively to restore the nonlinear constraints to feasibility. According to Biegler and Grossmann [94], these methods as well as control vector iteration methods can be computational expensive for large problems.

The most common method to the above problem is control vector iteration. Here, an initial control profile is assumed, the state equations are integrated forward and the adjoint equations are integrated backward. At the end of this pass, a new trajectory is calculated that minimizes the Hamiltonian. The control function is updated using the gradient method in [13-32,36,40,41], referenced in reviews by Ray [22], Ray and Szekely [23], and Jones and Finch [36]. The control function is updated using the conjugate gradient method in [33-35,37-39], reviewed by Ray and Szekely [23], Jones and Finch [36] and Renfro [72].

Quasi-Newton algorithms are based on a static optimization algorithm and approximate a second-order operator using first-order gradient information only, thus seeking the accelerated convergence associated with second-order methods. This type of '*variable metric*' algorithm has been adopted to optimal control (Tripathi and Nanendra[42], Lasdon[43]), reviewed by Jones and Finch [36]. According to Biegler and Grossmann [94], the main drawback of control vector iteration is that repeated solutions of state and adjoint equations are required.

The first-order necessary conditions of the maximum principle are used to produce a two-point boundary-value problem. Then the method of weighted residuals is used to discretize the problem and convert it to a system of nonlinear equations in

[46,47,57,58]. The system of nonlinear equations was solved by Newton method in [46,47]. [47] was reviewed by Biegler and Grossmann in [94]. According to Oh and Luus [46], the method is computationally efficient and yields results close to the optimum with a small number of collocation points. The method was used to determine the optimal control of a distributed parameter system, a model of a nuclear reactor. Computational results demonstrated the validity of the method. [57,58] was reviewed by Hicks and Ray in [48]. According to Hicks and Ray [48], this approximation method works well for simple problems [57], or for linear ones [58], but it has certain drawbacks for complex non-linear problems. One of the drawbacks is that the equation (11) has to be solved for **u** which is not always possible for highly non-linear or multivariable problems.

Methods based on the necessary conditions of optimality require iterative integration of the original and adjoint equations (two-point boundary value problem). Also, these methods cannot handle general inequality constraints. An additional difficulty is the fact that one must be able to analytically differentiate the integrand of the objective functional and the state equations in order to obtain the *Euler-Lagrange* equations explicitly. This can be very tedious for some chemical engineering problems, where complex empirical functions of the state variables are imbedded in the process model.

An alternative approach to solve differential-algebraic optimization problems is termed control vector parameterization, where the control function is expressed as a linear combination of trial functions and the linear combination coefficients and the coefficients of the trial functions are the optimizing variables. Control vector parameterization as well as control vector iteration are termed as sequential optimization methods. Control vector parameterization is used in [48-56,59-62]. These were reviewed in [23], [48] and [72].

According to Ray and Szekely [23], Sage [54] gives several examples in which partial differential equations have been discretized in the spatial direction and control vector parameterization was applied to the resulting set of ordinary differential equations. According to Ray and Szekely [23], Zahrednik and Lynn [52] and Bosarge [53] suggest a method, whereby the necessary conditions for optimality is applied to the model, then all the variables in the resulting equations are expanded in trial functions using control vector parameterization, then the method of weighted residuals is used to evaluate the coefficients; however, according to Ray and Szekely [23], there had been little computational experience on nonlinear problems.

Control vector parameterization also requires iterative integration of the original set of differential equations. Also, this method cannot handle algebraic equations and general inequality constraints. The principal disadvantage of the parameterization methods is that the functional form of the optimal control must be specified in advance. This requires much more physical insight than is needed by other methods, like control vector iteration. In the absence of the physical feeling for the general shape of the optimal control, a very general functional form must be used and the optimization performed with respect to a large number of coefficients.

Mentioned by Renfro [72], Hertzberg and Asbjornsen [64] were the first to introduce the idea of using orthogonal collocation coupled with a *quasi-Newton* method to perform simultaneous parameter estimation and integration in nonlinear dynamic systems. Differential equations and other equations containing state variables were replaced by an approximating set of algebraic equations, and the optimization was performed in the subspace of parameters. This method proved to be superior in computational efficiency to other existing parameter estimation algorithms.

Tsang et al. [65] and Luus [66] solved optimal control problems as a nonlinear programming problem using algebraic approximation, reviewed by Biegler and Grossmann [94].

Linear-quadratic feedback problem is converted using algebraic approximation into a quadratic programming problem in [67-69], reviewed by Biegler and Grossmann [94].

Orthogonal Collocation was first applied to the solution of chemical engineering problems by Villadsen and Sorensen [90].

A linear parabolic *PDE* is discretized in [70] using both global orthogonal collocation at Legendre roots and Lagrange interpolation polynomials, reviewed by Cuthrell [73]. The resulting lumped system is then integrated repeatedly with the variables found by direct search optimization. According to Cuthrell [73], the disadvantage of this approach to solve differential-algebraic optimization problems is that it is expensive as far as computer run time is concerned. In my opinion, its advantage is the low dimensionality of the resulting direct search optimization problem.

Lynn and Zahrednik used *Galerkin's* method to discretize a linear partial differential equation *(PDE)* [71], reviewed by Cuthrell [73]. Then the resulting linear-quadratic control problem containing *ODEs* is solved with the method used in [46,47] and mentioned in [94]. *Pontryagin's maximum principle* was applied to redefine the problem in terms of the necessary conditions for optimality. Then orthogonal collocation was applied to transform the *ODE*s into a system of nonlinear algebraic equations, which then was solved by *Newton*' s method.

Paterson and Cresswell made the first step in the direction of Orthogonal Collocation on Finite Elements [136], what was extended by Carey and Finlayson into Orthogonal Collocation on Finite Elements [89].

Orthogonal collocation on finite elements is applied to differential-algebraic optimization problems containing *ODEs* in [72-76]. The resulting *NLP* problem is then solved with Successive Quadratic Programming.

There currently exist methods that use orthogonal collocation on finite elements that convert the differential-algebraic optimization problem containing *ODE*s onto a *NLP* problem, then solve the *NLP* problem using successive quadratic programming. Control vector iteration and control vector parameterization can be applied to solve distributed parameter systems not containing algebraic equations and general inequalities. The method referenced in [73] and used by Lynn and Zahrednik [71] can solve distributed parameter systems, but cannot be completely automated, it requires analytical differentiation of the integrand of the objective functional and the state equations in order to obtain the *Euler-Lagrange* equations explicitly.

This work uses double orthogonal collocation on finite elements to convert the differential-algebraic optimization problem into a *NLP* problem that is solved either by generalized reduced gradient method or by successive quadratic programming. This approach was taken because it can be fully automated from model to the optimal solution without manual intervention and it does not depend on the assumptions in the variational conditions since it does not use the necessary conditions for optimality. It can be considered an extension of the approaches described in [72-76] to *PDE*s.

Chapter 3
Theory

3.1 Introduction

In this work control vector parameterization and double orthogonal collocation on finite elements have been implemented to solve differential-algebraic optimization problems. Double collocation converts a *PDE* into a number of algebraic equations. Collocation converts a *PDE* into a number of *ODEs* or an *ODE* into a number of algebraic equations. This chapter presents the theory necessary for this work.

Control vector parameterization represents the control functions of a problem as linear combinations of suitable trial functions. The linear combination coefficients and the coefficients of the trial functions are the optimization variables. It is a sequential optimization method, whereby the optimum is found by an unconstrained optimization method, and each time the optimization algorithm requires the value of the objective function, the original set of differential equations has to be solved.

Orthogonal collocation on finite elements discretizes the differential equations at chosen collocation points into a set of algebraic equations by approximating the unknown functions to be found by a linear combination of *Lagrange* interpolation polynomials. The derivatives of the functions are approximated by a similar linear combination of derivatives of the *Lagrange* interpolation polynomials. The resulting *NLP* can be solved by any suitable *NLP* solver. In this work either the generalized reduced gradient method or successive quadratic programming are used.

The generalized reduced gradient method is a feasible path method. It solves the nonlinear optimization problem in the subspace of the *n-m* nonbasic variables, where *n* is the number of variables and *m* is the number of constraints. Its advantage is low dimensionality, its disadvantage is that it requires that the constraints be satisfied at each iteration.

The successive quadratic programming method is an infeasible path method. It does not require that the constraints be satisfied at each iteration. It constructs a quadratic programming subproblem by constructing a quadratic objective function and linearizing the constraints and solves the subproblem repeatedly. These topics will now be covered in more detail.

3.2 Control Vector Parameterization

3.2.1 Lumped Parameter Systems

In a lumped parameter system with time variation, the properties are spatially uniform throughout the system, thus allowing the system to be represented by a set of ordinary differential equations.

The control functions *$u_i(t)$* are expressed as linear combination of trial functions *$f_{ij}(t)$*. The functions *$u_i(t)$* and *$f_{ij}(t)$* are chosen by the user, using any insights into the nature of the problem, e.g.,

$$\begin{aligned} u_i(t) &= a_{i1} + a_{i2}t + a_{i3}t^2 \\ f_{i1}(t) &= 1 \\ f_{i2}(t) &= t \\ f_{i3}(t) &= t^2 \end{aligned} \tag{1}$$

and covering different possible solution behaviours.

$$u_i(t) = \sum_{j=1}^{m} a_{ij} f_{ij}(c_{ij_1},\ldots,c_{ij_{n_{ij}}},t) \qquad i=1,\ldots,p \tag{2}$$

Unconstrained optimization techniques are used to determine the optimal set of coefficients a_{ij} and c_{ij}. Another approach is to generate u_i in a feedback form, i.e., expand it as a linear combination of trial functions of the state variables

$$u_i(t) = \sum_{j=1}^{m} b_{ij} f_{ij}(x_1,x_2,\ldots x_n,c_{ij_1},\ldots,c_{ij_{n_{ij}}},t) \qquad i=1,\ldots,p \tag{3}$$

and determine the optimal coefficients b_{ij} and c_{ij}. If the control policy is purely on-off, a control of the form

$$u_i(t) = u_{i*} + (u_i^* - u_{i*})\left[\sum_{j=1}^{m} (-1)^{j+1}\, H(t-a_{ij})\right] \tag{4}$$

where *$H(t)$* is the Heaviside function

$$H(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{5}$$

could be used.

3.2.2 Distributed Parameter Systems

Distributed parameter systems are represented by partial differential equations. If the independent variables are x,y, then

$$u_i(x,y) = \sum_{j=1}^{m} a_{ij} f_{ij}(x,y,c_{ij_1},\ldots,c_{ij_{n_{ij}}}) \quad i=1,\ldots,p_1 \tag{6}$$

$$v_i(x) = \sum_{j=1}^{k} b_{ij} g_{ij}(x,c_{ij_1},\ldots,c_{ij_{n_{ij}}}) \quad i=1,\ldots,p_2 \tag{7}$$

$$w_i(y) = \sum_{j=1}^{l} c_{ij} h_{ij}(y,c_{ij_1},\ldots,c_{ij_{n_{ij}}}) \quad i=1,\ldots,p_3 \tag{8}$$

assuming we have p_1 control variables u_i depending on both independent variables, p_2 control variables v_i depending on the independent variable x, and p_3 control variables w_i depending on the independent variable y.

3.2.3 Algorithm to solve an optimization problem using control vector parameterization

(1) Guess the coefficients

(2) Numerically integrate the state equations

(3) Compute the objective functional

(4) Compute new values of coefficients using an unconstrained optimization procedure

(5) Repeat (2)-(4) until the optimal set of coefficients is found

3.3 Orthogonal Collocation on Finite Elements

In the method of weighted residuals (*MWR*), each unknown function $c(x,y)$ is approximated by

$$\hat{c}(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \phi_i(x) \phi_j(y) \tag{9}$$

Substitution of $\hat{c}$ and its derivatives into the *PDE* results in a residual, R. We require that the integral of the residual be zero:

$$\int\limits_x\int\limits_y R(x,y)w_{ij}(x,y)dxdy = 0 \quad i=1,2,...,N \; j=1,2,...M \tag{10}$$

This equation is the general equation describing the *MWR*, and a multiplicity of schemes arise out of this expression through the definition of the weighting functions w_{ij}. In the collocation method the weighting functions w_{ij} are all chosen to be the *Dirac delta*, that is,

$$w_{ij} = \delta(x-x_i,y-y_j) \tag{11}$$

The *Dirac delta* function has the important property that

$$\int\limits_x\int\limits_y a(x,y)\delta(x-x_i,y-y_j)dxdy \equiv a|_{x_i y_j} \tag{12}$$

Thus the orthogonality requirement for the collocation method

$$\int\limits_x\int\limits_y R(x,y)\delta_{ij}(x,y)dxdy = 0, \quad i=1,2,...,N \; j=1,2...,M \tag{13}$$

are a mathematical statement of the requirement that the residual vanish at each collocation point (x_i,y_i)

Orthogonal collocation and orthogonal collocation on finite elements are methods to discretize differential equations. The unknown function, its first and second derivatives are approximated by a linear combination of *Lagrange* interpolation polynomials and their derivatives. The linear combination coefficients are the values of the unknown function at collocation points. The *Lagrange* polynomials are defined at collocation points within one or more finite elements. A separate set of *Lagrange* polynomials is used in each finite element. The collocation points are chosen within the finite element. In this work they are chosen to be the roots of the *Legendre* polynomials in the interval 0 to 1 [79-80]. It is required that the residual, i.e. the differential equation after substituting its approximation for the unknown function and for its derivatives be zero at the collocation points. If there are N_x finite elements along the x-interval, N_y finite elements along the y-interval, N internal collocation points in each finite element in x-interval, M internal collocation points in y-interval, a *PDE* with 2 independent variables will give rise to $(N_x(N+1)+1)(N_y(M+1)+1)$ algebraic equations and a state variable dependent on 2 independent variables will give rise to the same number of *NLP* variables. The derivatives of the *Lagrange* polynomials will be constants in the equations. The values of the unknown functions at collocation points will become *NLP* variables. The differential equations are collocated at those collocation points at the boundary of the integration domain, where there are no initial or boundary conditions.

Orthogonal Collocation on Finite Elements applied to an *ODE*:

State variables and control variables are replaced by a linear combination of *Lagrange* interpolation polynomials. The derivatives of the state variables are replaced by a linear combination of the derivatives of *Lagrange* interpolation polynomials. The differential equations are evaluated at collocation points where there are no initial or boundary conditions. In each finite element k the equation is to be satisfied at collocation points t_i, i = 1,...,N+2 excluding any boundary condition.

$$f(c,\frac{dc}{dt},\frac{d^2c}{dt^2},w,t)\Big|_{t=t_i} = 0 \qquad k = 1,...,N_x \tag{14}$$

where w is the control and c is the state variable,

N_x = number of finite elements,

N = number of internal collocation points.

The state variable $c(t)$ is approximated by

$$c^k(t) = \sum_{n=1}^{N+2} c_n^k l_n(u) \quad u=\frac{(t-t_k)}{\Delta t_k} \tag{15}$$

where

$$\Delta t_k = t_{k+1} - t_k \tag{16}$$

and

$$l_n(u) = \prod_{j=1, j\neq n}^{N+2} \frac{u-u_j}{u_n-u_j} \tag{17}$$

is the *n-th Lagrange* interpolation polynomial, for which

$$l_n(u_i) = \begin{bmatrix} 0 & for\ i \neq n \\ 1 & for\ i = n \end{bmatrix} \tag{18}$$

giving

$$c^k(t_i) = \sum_{n=1}^{N+2} c_n^k l_n(u_i) \tag{19}$$

This reduces to $c^k(t_i) = c_i^k$ because of (18). c_i^k is the value of *c(t)* at the collocation point u_i in the *k-th* finite element.

The control variable *w(t)* is similarly approximated by

$$w^k(t) = \sum_{n=2}^{N+1} w_n^k l_n(u) \tag{20}$$

at only the N internal collocation points [79-80] giving

$$w^k(t_i) = \sum_{n=2}^{N+1} w_n^k l_n(u_i) \qquad for\ 2 \leq i \leq N+1 \tag{21}$$

and $w^k(t_i) = w_i^k$ as before because of (18).
At the end points of finite elements :

$$w^k(t_i) = \sum_{n=2}^{N+1} w_n^k l_n(u_i) \qquad for\ i = 1\ and\ N+2 \tag{22}$$

The derivatives of the variable *c(t)* are approximated by

$$\frac{dc^k(t_i)}{dt} = \frac{1}{\Delta t_k} \sum_{n=1}^{N+2} c_n^k \frac{dl_n(u_i)}{du} \tag{23}$$

and

$$\frac{d^2c^k(t_i)}{dt^2} = \frac{1}{\Delta t_k^2}\sum_{n=1}^{N+2} c_n^k \frac{d^2 l_n(u_i)}{du^2} \tag{24}$$

Theorem [59]

There exists a unique polynomial $p_n(x)$ of degree n which assumes prescribed values at $n + 1$ distinct real numbers $x_0 < x_1 < ... < x_n$.

Proof.

Let

$$p_n(x) = \sum_{i=0}^{n} c(x_i) l_i(x) \tag{25}$$

From (17) and (18) can be seen that $p_n(x)$ is a polynomial of degree n and goes through the points $(x_0,c(x_0)),...,(x_n,c(x_n))$. In order to prove uniqueness, let us suppose that $q(x)$ is also a polynomial of degree n interpolating $c(x)$ at $x_0 < x_1 <,..., < x_n$. Let

$$r(x) = p_n(x) - q(x). \tag{26}$$

Then $r(x)$ is a polynomial of degree n with $n + 1$ zeros at $x_0,x_1,...,x_n$. It follows from the fundamental theorem of algebra that $r(x)$ is identically equal to zero (i.e., the only polynomial of degree n with more than n roots is the zero polynomial. Thus $p_n(x)$ is unique.

Orthogonal Collocation on Finite Elements applied to *PDEs* with 2 independent variables:

State variables and control variables dependent on 2 independent variables are replaced by a linear combination of two sets of *Lagrange* interpolation polynomials. The derivatives of the state variables are replaced by a linear combination of the derivatives of *Lagrange* interpolation polynomials. The differential equation

$$f\left((c,\frac{\partial c}{\partial x},\frac{\partial^2 c}{\partial x^2},\frac{\partial c}{\partial y},\frac{\partial^2 c}{\partial y^2},\frac{\partial^2 c}{\partial x\partial y},w,x,y\right) = 0 \tag{27}$$

must be satisfied at collocation points in each finite element *(kl)*, k=1,...,N_x, l=1,...,N_y, at the collocation points (x_i,y_j) where i = 1,...,N+2, j = 1,..., M+2 excluding any boundary condition where w is the control and c is the state variable. The state variable $c(x,y)$ is approximated by

$$c^{kl}(x,y) = \sum_{n=1}^{N+2}\sum_{m=1}^{M+2} c_{nm}^{kl} l_n(u) l_m(v) \quad u = \frac{(x-x_k)}{\Delta x_k}, \; v = \frac{y-y_l}{\Delta y_l} \tag{28}$$

$$c^{kl}(x_i,y_j) = \sum_{n=1}^{N+2}\sum_{m=1}^{M+2} c_{nm}^{kl} l_n(u_i) l_m(v_j) \tag{29}$$

and $c^{kl}(x_i,y_j) = c_{ij}^{kl}$ because of (18). Here c_{ij}^{kl} is the value of c at the collocation points (u_i,v_j) in the *(kl)-th* element

The control variable $w(x,y)$ is similarly approximated by

$$w^{kl}(x,y) = \sum_{n=2}^{N+1}\sum_{m=2}^{M+1} w_{nm}^{kl} l_n(u) l_m(v) \tag{30}$$

$$w^{kl}(x_i,y_j) = \sum_{n=2}^{N+1}\sum_{m=2}^{M+1} w_{nm}^{kl} l_n(u_i) l_m(v_j) \quad \begin{matrix} \textit{for } 2 \le i \le N+1 \\ \textit{and } 2 \le j \le M+1 \end{matrix} \tag{31}$$

and $w^{kl}(x_i,y_j) = w_{ij}^{kl}$ because of (18).

$$w^{kl}(x_i,y_j) = \sum_{n=2}^{N+1}\sum_{m=2}^{M+1} w_{nm}^{kl} l_n(u_i) l_m(v_j) \tag{32}$$

at the corners of the 2-dimensional finite elements.

$$w^{kl}(x_i,y_j) = \sum_{n=2}^{N+1} w_{nj}^{kl} l_n(u_i) \quad \begin{matrix} \textit{for } i=1 \textit{ and } N+2 \\ 2 \le j \le M+1 \end{matrix} \tag{33}$$

$$w^{kl}(x_i,y_j) = \sum_{m=2}^{M+1} w_{im}^{kl} l_m(v_j) \quad \begin{matrix} \textit{for } 2 \le i \le N+1 \\ j=1 \textit{ and } M+2 \end{matrix} \tag{34}$$

The derivatives of the state variable $c(x,y)$ are approximated by

$$\frac{\partial c^{kl}(x_i,y_j)}{\partial x} = \frac{1}{\Delta x_k}\sum_{n=1}^{N+2}\frac{dl_n(u_i)}{du}\left[\sum_{m=1}^{M+2} c_{nm}^{kl} l_m(v_j)\right] = \frac{1}{\Delta x_k}\sum_{n=1}^{N+2}\frac{dl_n(u_i)}{du}c_{nj}^{kl} \tag{35}$$

$$\frac{\partial^2 c^{kl}(x_i,y_j)}{\partial x^2} = \frac{1}{\Delta x_k^2}\sum_{n=1}^{N+2}\frac{d^2 l_n(u_i)}{du^2}\left[\sum_{m=1}^{M+2} c_{nm}^{kl} l_m(v_j)\right] = \frac{1}{\Delta x_k^2}\sum_{n=1}^{N+2}\frac{d^2 l_n(u_i)}{du^2}c_{nj}^{kl} \tag{36}$$

$$\frac{\partial c^{kl}(x_i,y_j)}{\partial y} = \frac{1}{\Delta y_l}\sum_{m=1}^{M+2}\frac{dl_m(v_j)}{dv}\left[\sum_{n=1}^{N+2} c_{nm}^{kl} l_n(u_i)\right] = \frac{1}{\Delta y_l}\sum_{m=1}^{M+2}\frac{dl_m(v_j)}{dv}c_{im}^{kl} \tag{37}$$

$$\frac{\partial^2 c^{kl}(x_i,y_j)}{\partial y^2} = \frac{1}{\Delta y_l^2}\sum_{m=1}^{M+2}\frac{d^2 l_m(v_j)}{dv^2}\left[\sum_{n=1}^{N+2} c_{nm}^{kl} l_n(u_i)\right] = \frac{1}{\Delta y_l^2}\sum_{m=1}^{M+2}\frac{d^2 l_m(v_j)}{dv^2}c_{im}^{kl} \tag{38}$$

$$\frac{\partial^2 c^{kl}(x_i,y_j)}{\partial x\partial y} = \frac{1}{\Delta x_k \Delta y_l}\sum_{n=1}^{N+2}\sum_{m=1}^{M+2}\frac{dl_n(u_i)}{du}\frac{dl_m(v_j)}{dv} \tag{39}$$

Theorem.

There exists a unique polynomial of degree n in x and of degree m in y solving the interpolation problem

$$p_{nm}(x_i,y_j) = c(x_i,y_j) \quad 0 \le i \le n;\ 0 \le j \le m \tag{40}$$

on a rectangular grid.

Proof.

Eqs. (28-29) establish the existence of such polynomials. In order to prove uniqueness, suppose

$$q_{nm}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}x^i y^j \tag{41}$$

is a polynomial of degree n in x and of degree m in y also solving (40). Equ. (28) in terms of this theorem is

$$p_{nm}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} l_i(x)l_j(y)c_{ij} \tag{42}$$

where $c_{ij} = c(x_i,y_j)$. From the assumption that p_{nm} and q_{nm} solve the interpolation problem (40), it follows that

$$p_{nm}(x_p,y_q) = q_{nm}(x_p,y_q) = c(x_p,y_q) \tag{43}$$

From (43) the following equation holds

$$\sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}x_p^i y_q^j = c_{pq} \tag{44}$$

The coefficient of $x^p y^q$ in Equ. (41) is a_{pq}. The coefficient of $x^p y^q$ in Equ. (42) is given by

$$\sum_{i=0}^{n}\sum_{j=0}^{m} \frac{1}{\prod\limits_{k=0,k\neq i}^{n}\prod\limits_{l=0,l\neq j}^{m}(x_i-x_k)(y_j-y_l)}\left[\sum_{1}^{s}\left((-1)^{n+m-p-q}\prod_{k\neq i,k\in\alpha} x_k \prod_{l\neq j,l\in\beta} y_l\right)\right]c_{ij} \tag{45}$$

where α is a selection of n-p elements out of $x_0, x_1, ..., x_{i-1}, ..., x_{i+1}, ..., x_n$ and β is a selection of m-q elements out of $y_0, y_1, ..., y_{j-1}, ..., y_{j+1}, ..., y_m$ without regard to order. $s = C_{n-p}^{n}C_{m-q}^{m}$, where C_{n-p}^{n} is the number of ways of selecting n-p objects out of n objects without regard to order and C_{m-q}^{m} is the number of ways of selecting m-q objects out of m objects without regard to order. a_{pq}'s are uniquely determined by the corresponding coefficients in Equ. (42) shown in (45). Therefore a_{pq}'s are unique. Since the coefficients of q_{nm} are unique, they must be equal to the corresponding coefficients shown in (45), demonstrating that $q_{nm}(x,y)$ is $p_{nm}(x,y)$ in rearranged from, i.e.,

$$q_{nm}(x,y) \equiv p_{nm}(x,y). \tag{46}$$

This proves that p_{nm} is unique.

Example. If $n = 2$, $m = 2$, $p = 1$, $i = 2$, $q = 1$, $j = 2$, then Equ. (45) becomes

$$\frac{1}{(x_2-x_0)(x_2-x_1)(y_2-y_0)(y_2-y_1)}\left[\sum_1^4\left((-1)^2\prod_{k\neq i,k\in\alpha}x_k\prod_{l\neq j,l\in\beta}y_l\right)\right]c_{22} = \qquad (47)$$
$$(\,x_0y_0 + x_0y_1 + x_1y_0 + x_1y_1\,)\;c_{22}$$

where

$$C_{n-p}^{n} = C_p^{n} = C_1^{2} = \binom{2}{1} = 2;\; C_{m-q}^{m} = C_q^{m} = C_1^{2} = \binom{2}{1} = 2 \qquad (48)$$

For the first combination $\alpha = x_0$, for the second combination $\alpha = x_1$, for the first combination $\beta = y_0$, for the second combination $\beta = y_1$.

The state variables are defined at the internal collocation points of finite elements as well as at the boundaries of finite elements and at the boundaries of the integration domain. The control variables are only defined at the internal collocation points of finite elements. The control variables are extrapolated at the boundaries of finite elements and at boundaries of the integration domain. In order to make the number of collocation points equal to the number of algebraic equations, the differential equations are collocated at finite element boundaries and at those boundary points where there are no boundary conditions. Although the state functions are approximated by different polynomials in different finite elements, the state function is a continuous function over the integration domain, because the value of the state function at finite element boundary is a linear combination coefficient in the polynomial approximation and an *NLP* variable, and because of property (18) the approximating polynomial reduces to this variable. The control functions are not continuous over the integration domain, because the value of the control function at boundary is not a linear combination coefficient of the polynomial approximation and not an *NLP* variable, but extrapolated by a linear combination of *Lagrange* polynomials defined over internal collocation points. This would result in the control function having different values at boundary for different finite elements sharing the boundary. In order to make the control function a one-valued function, it is approximated in only 1 finite element at boundary. An arbitrary choice has been made that boundary points belong to finite elements on the left hand side or on the lower side of finite elements.

3.4 Constrained Optimization

The *DAOP* problem has now been formulated a as large *NLP* problem where there are many equality constraints, and there may also be inequality constraints. It is therefore necessary to solve the optimization to solve the original problem. The conditions for a solution of such problems consist of following 6 conditions now developed [116].

Definition. A point $\mathbf{x}^*$ satisfying the constraint $\mathbf{h}(\mathbf{x}^*)=\mathbf{0}$ is said to be a regular point of the constraint if the gradient vectors $\nabla h_1(\mathbf{x}^*),\nabla h_2(\mathbf{x}^*),\ldots,\nabla h_m(\mathbf{x}^*)$ are linearly independent.

First-Order Necessary Conditions (Equality Constraints)

Let $\mathbf{x}^*$ be a local extremum point of f subject to the constraints $\mathbf{h}(\mathbf{x})=\mathbf{0}$.

Assume further that $\mathbf{x}^*$ is a regular point of these constraints. Then there is a $\lambda \in E^m$
such that

$$\begin{aligned} \nabla f(x^*) + \lambda^T \nabla \boldsymbol{h}(x^*) &= \mathbf{0} \\ \boldsymbol{h}(x^*) &= \mathbf{0} \end{aligned} \tag{49}$$

The necessary conditions can be expressed in the form

$$\begin{aligned} \nabla_x l(x^*,\lambda) &= \mathbf{0} \\ \nabla_\lambda l(x^*,\lambda) &= \mathbf{0} \end{aligned} \tag{50}$$

where

$$l(x,\lambda) = f(x) + \lambda^T \boldsymbol{h}(x) \tag{51}$$

is the *Lagrangian.*

Second-Order Necessary Conditions (Equality Constraints)

Suppose that $\mathbf{x}^*$ is a local minimum of f subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ and that $\mathbf{x}^*$ is a regular point of these constraints. Then there is a $\lambda \in E^m$ such that

$$\nabla f(x^*) + \lambda^T \nabla \boldsymbol{h}(x^*) = \mathbf{0} \tag{52}$$

or

$$\frac{\partial f(x^*)}{\partial x_1} + \lambda_1 \frac{\partial h_1(x^*)}{\partial x_1} + \lambda_2 \frac{\partial h_2(x^*)}{\partial x_1} + \cdots + \lambda_m \frac{\partial h_m(x^*)}{\partial x_1} = 0$$

$$\vdots$$

$$\frac{\partial f(x^*)}{\partial x_n} + \lambda_1 \frac{\partial h_1(x^*)}{\partial x_n} + \lambda_2 \frac{\partial h_2(x^*)}{\partial x_n} + \cdots + \lambda_m \frac{\partial h_m(x^*)}{\partial x_n} = 0 \tag{53}$$

$$h_1(x^*) = 0$$

$$\vdots$$

$$h_m(x^*) = 0$$

If the tangent plane $M = \{\mathbf{y}:\nabla\mathbf{h}(\mathbf{x}^*)\mathbf{y} = \mathbf{0}\}$ is denoted by M, then the matrix

$$L(x^*) = F(x^*) + \lambda^T H(x^*) \tag{54}$$

is positive semidefinite on M, that is, $\mathbf{y}^T\mathbf{L}(\mathbf{x}^*)\mathbf{y} \geq \mathbf{0}$ for all $\mathbf{y} \in M$.

$\mathbf{L} = \mathbf{F} + \lambda^T\mathbf{H}$ is the matrix of second partial derivatives, with respect to $\mathbf{x}$, of the Lagrangian $l = \mathrm{f} + \lambda^T\mathbf{h}$.

$\mathbf{F}$ is the Hessian of f and $\lambda^T H(x^*) = \sum_{i=1}^{m} \lambda_i H_i(x^*)$ where $\mathbf{H}_i(\mathbf{x}^*)$ is the Hessian of $\mathrm{h}_i(\mathbf{x}^*)$.

Second-Order Sufficiency Conditions (Equality Constraints)

Suppose there is a point $\mathbf{x}^*$ satisfying $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$ and a $\lambda \in E^m$ such that

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) = \mathbf{0} \tag{55}$$

Suppose also that the matrix

$$L(x^*) = F(x^*) + \lambda^T H(x^*) \tag{56}$$

is positive definite on

$M = \{\mathbf{y}:\nabla\mathbf{h}(\mathbf{x}^*)\mathbf{y} = \mathbf{0}\}$, that is, for $\mathbf{y} \in M$, $\mathbf{y} \neq \mathbf{0}$ there holds $\mathbf{y}^T\mathbf{L}(\mathbf{x}^*)\mathbf{y} > \mathbf{0}$. Then $\mathbf{x}^*$ is a local minimum of f subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$.

Let $\mathbf{x}^*$ be a point satisfying the constraints

$$h(x^*) = \mathbf{0}\ ,\ g(x^*) \leq \mathbf{0} \tag{57}$$

and let J be the set of indices j for which $g_j(x^*) = 0$. Then $\mathbf{x}^*$ is said to be a regular point of the constraints (57) if the gradient vectors $\nabla h_i(x^*)$, $\nabla g_j(x^*)$, $1 \leq i \leq m$, $j \in J$ are linearly independent.

Kuhn-Tucker Conditions (First-Order Necessary Conditions - Inequality Constraints).

Let $\mathbf{x}^*$ be a relative minimum point for the problem

$$\begin{aligned} &\min f(x) \\ &\textit{subject to } \boldsymbol{h}(x) = \mathbf{0},\ \boldsymbol{g}(x) \le \mathbf{0}, \end{aligned} \tag{58}$$

and suppose $\mathbf{x}^*$ is a regular point for the constraints. Then there is a vector $\lambda \in E^m$ and a vector $\boldsymbol{\mu} \in E^p$ with $\boldsymbol{\mu} \geq \mathbf{0}$ such that

$$\nabla f(x^*) + \lambda^T \nabla \boldsymbol{h}(x^*) + \mu^T \nabla \boldsymbol{g}(x^*) = \mathbf{0} \tag{59}$$

$$\mu^T \boldsymbol{g}(x^*) = \mathbf{0} \tag{60}$$

Since $\boldsymbol{\mu} \geq \mathbf{0}$ and $\mathbf{g(x)} \leq \mathbf{0}$, (60) is equivalent to the statement that a component of $\boldsymbol{\mu}$ may be nonzero only if the corresponding constraint is active.

Second-Order Necessary Conditions (Inequality Constraints)

Suppose the functions $\mathbf{f,g,h} \in C^2$ and that $\mathbf{x}^*$ is a regular point of the constraints (57). If $\mathbf{x}^*$ is a relative minimum point for the problem

$$\begin{aligned} &\min f(x) \\ &s.t.\ \boldsymbol{h}(x) = \mathbf{0} \\ &\quad\ \boldsymbol{g}(x) \le \mathbf{0} \end{aligned} \tag{61}$$

then there is a $\lambda \in E^m$, $\mu \in E^p$, $\mu \geq \mathbf{0}$ such that (59) and (60) hold and such that

$$\boldsymbol{L}(x^*) = \boldsymbol{F}(x^*) + \lambda^T \boldsymbol{H}(x^*) + \mu^T \boldsymbol{G}(x^*) \tag{62}$$

is positive semidefinite on the tangent subspace of the active constraints at $\mathbf{x}^*$ (Eqn. 65).

$\mathbf{L} = \mathbf{F} + \lambda^T\mathbf{H} + \mu^T\mathbf{G}$ is a matrix of second partial derivatives, with respect to $\mathbf{x}$, of the Lagrangian $l = \mathrm{f} + \lambda^T\mathbf{h} + \mu^T\mathrm{g}$. $\mathbf{F}$ is the Hessian of f, $\lambda^T \boldsymbol{H}(x^*) = \sum_{i=1}^{m} \lambda_i \boldsymbol{H}_i(X)$ where $\mathbf{H}_i(\mathbf{x}^*)$ is the Hessian of $\mathrm{h}_i(\mathbf{x}^*)$ and $\mu^T \boldsymbol{G}(x^*) = \sum_{i=1}^{p} \mu_i \boldsymbol{G}_i(x^*)$ where $\mathbf{G}_i(\mathbf{x}^*)$ is the Hessian of $\mathrm{g}_i(\mathbf{x}^*)$.

Second-Order Sufficiency Conditions (Inequality Constraints)

Let $\mathbf{f,g,h}$ be $\in C^2$. Suppose there exist $\lambda \in E^m$, $\mu \in E^p$, such that

$$\mu \geq 0$$

$$\mu^T g(x^*) = 0 \tag{63}$$

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) = 0,$$

and the Hessian matrix

$$L(x^*) = F(x^*) + \lambda^T H(x^*) + \mu^T G(x^*) \tag{64}$$

is positive definite on the subspace

$$M = \{ y : \nabla h(x^*) y = 0,\ \nabla g_j(x^*) y = 0 \ \textit{for all } j \in J \} \tag{65}$$

where

$$J = \{ j : g_j(x^*) = 0,\ \mu_j > 0 \} \tag{66}$$

Then $\mathbf{x}^*$ satisfying (57) is a strict relative minimum of problem (61).

3.5 Primal Methods [124].

Introduction. A primal method of solution is a method that searches through the feasible region for the optimal solution. Each point in the process is feasible and the value of the objective function constantly decreases. For a problem with n variables and having m equalities only, primal methods work in the feasible space, which has dimension n-m. This reduction in dimensionality is a clear advantage of the primal methods. Another advantage of this method is that since each point generated in the search procedure is feasible, if the process is terminated before reaching the solution, the terminating point is feasible. The major disadvantages of primal methods are, that they require a phase I procedure to obtain an initial feasible point, and they are all plagued, particularly for problems with nonlinear constraints, with computational difficulties arising from the necessity to remain within the feasible region as the method progresses.

3.5.1 The Generalized Reduced Gradient Method.

The choice of this method is based on [127-129], where the generalized reduced gradient method was evaluated as one of the best methods to solve nonlinear programming problems. For simplicity let us initially consider the nonlinear equality-constraint form of the *NLP*, namely

$$\begin{aligned} &\min f(x) \\ &\text{subject to } h_i(x) = 0, \; i=1,2,...m \end{aligned} \tag{67}$$

Implicit Variable Elimination. Suppose $\mathbf{x}^{(1)}$ is a point satisfying the constraints of the equality-constrained problem (67), and suppose a linear approximation to the problem at the point $\mathbf{x}^{(1)}$ is constructed. The result is

$$\tilde{h}_i(x;x^{(1)}) \equiv h_i(x^{(1)}) + \nabla h_i(x^{(1)})(x-x^{(1)}) \qquad i=1,...,m \tag{68}$$

where ∇h_i is a row vector.

Suppose these linearized equations are used to predict the location of another feasible point, that is, a point at which

$$\tilde{h}_i(x;x^{(1)}) = 0 \qquad i=1,2,...,m \tag{69}$$

Then, since $h_i(x^{(1)}) = 0$, $i = 1,2,...,m$, that point must satisfy the system of linear equations

$$\nabla h_i(x^{(1)})(x-x^{(1)}) = 0 \qquad i=1,2,...,m \tag{70}$$

In general, $m < n$; consequently, this system of equations will have more unknowns than equations and cannot be solved to yield a unique solution. It can be solved for m of the n variables in terms or the other n-m. The first m variables are labelled $\hat{x}$ (basic) and the remaining variables are labelled $\bar{x}$ (nonbasic). Corresponding to this partition of the **x**'s, the row vectors ∇h_i are partitioned into $\nabla \hat{h}_i$ and $\nabla \bar{h}_i$ and these subvectors are accumulated into two matrices **B** and **C**. The matrix **B** will consist of elements

$$B = \begin{bmatrix} \nabla \hat{h}_1 \\ \nabla \hat{h}_2 \\ \vdots \\ \nabla \hat{h}_m \end{bmatrix} \tag{71}$$

and the matrix **C** of the elements

$$C = \begin{bmatrix} \nabla \bar{h}_1 \\ \nabla \bar{h}_2 \\ \vdots \\ \nabla \bar{h}_m \end{bmatrix} \tag{72}$$

The system of Eqs. (70) can be written using the matrix notation introduced above as

$$B(\hat{x} - \hat{x}^{(1)}) + C(\bar{x} - \bar{x}^{(1)}) = 0 \tag{73}$$

Assuming that the square $m \times m$ matrix **B** has nonzero determinant, this set of equations can be solved in terms of the $\bar{x}$ for the $\hat{x}$ variables. This is equivalent to

$$\hat{x} - \hat{x}^{(1)} = - B^{-1}C(\bar{x} - \bar{x}^{(1)}) \tag{74}$$

To a first-order approximation, all points in the vicinity of $\mathbf{x}^{(1)}$ satisfying the constraints $h_i(\mathbf{x}) = 0$ will be given by the matrix equation (74). For any choice of the nonbasic variables $\bar{x}$, the matrix equation will calculate values of the basic variables $\hat{x}$ that will satisfy the linear approximations to the original constraints. The linearization has made it possible to solve the constraints for *m* variables even though this could not be accomplished directly with the constraints themselves.

Eq. (74) can be used to eliminate variables $\hat{x}$ from the objective function f(**x**). Thus,

$$\tilde{f}(\hat{x} \,;\, \bar{x}) \approx f(\hat{x}^{(1)} - B^{-1}C(\bar{x} - \bar{x}^{(1)}), \bar{x}) \tag{75}$$

and *f* is reduced to a function involving only the *n-m* nonbasic variables $\bar{x}$. Since *f* is an unconstrained function of the nonbasic variables $\bar{x}$, the necessary conditions for $\mathbf{x}^{(1)}$ to be a local minimum of *f* are that the gradient of *f* with respect to $\bar{x}$ be zero.

Using the chain rule, since $\tilde{f}(\bar{x}) = f(\hat{x}(\bar{x}), \bar{x})$, then

$$\frac{\partial \tilde{f}}{\partial \bar{x}} = \frac{\partial f}{\partial \bar{x}} + \frac{\partial f}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial \bar{x}} \tag{76}$$

Since, from Eq. (74),

$$\frac{\partial \hat{x}}{\partial \bar{x}} = -\boldsymbol{B}^{-1}\boldsymbol{C} \tag{77}$$

and if we write $\nabla\bar{f} = (\partial f \,/\, \partial\bar{x})$ *and* $\nabla\tilde{f} = (\partial\tilde{f} \,/\, \partial\bar{x})$ *and* $\nabla\hat{f} = (\partial f \,/\, \partial\hat{x})$, then it follows that

$$\nabla\tilde{f}(\boldsymbol{x}^{(1)}) = \nabla\bar{f}(\boldsymbol{x}^{(1)}) - \nabla\hat{f}(\boldsymbol{x}^{(1)})\boldsymbol{B}^{-1}\boldsymbol{C} \tag{78}$$

The first-order necessary conditions thus become

$$\nabla\bar{f}(\boldsymbol{x}^{(1)}) - \nabla\hat{f}(\boldsymbol{x}^{(1)})\boldsymbol{B}^{-1}\boldsymbol{C} = \boldsymbol{0} \tag{79}$$

The vector $\nabla\tilde{f}$ defined by (78) is called the reduced gradient (or constrained derivative) of the equality-constrained problems. The first-order necessary conditions for the local minimum expressed in terms of the reduced gradient reduces simply to the statement that

$$\nabla\tilde{f}(\boldsymbol{x}^{(1)}) = \boldsymbol{0} \tag{80}$$

It is going to be shown [124] that the reduced gradient optimality criterion is equivalent to the *Lagrangian* optimality criterion. Thus, zeros of the reduced gradient are *Lagrangian* stationary points. To verify this, the *Lagrangian* necessary conditions must be considered for the equality-constrained problem. These are

$$\nabla f(\boldsymbol{x}^{*}) + (\lambda^{*})^{T}\nabla \boldsymbol{h}(\boldsymbol{x}^{*}) = \boldsymbol{0} \tag{81}$$

where $\nabla\mathbf{h}$ is the matrix of constraint gradients. If the definition of independent and dependent variables $\bar{x}$ and $\hat{x}$ is introduced, then the above system of equations can be written as

$$\begin{aligned} \nabla\hat{f}(\boldsymbol{x}^{*}) + (\lambda^{*})^{T}\boldsymbol{B} &= \boldsymbol{0} \\ \nabla\bar{f}(\boldsymbol{x}^{*}) + (\lambda^{*})^{T}\boldsymbol{C} &= \boldsymbol{0} \end{aligned} \tag{82}$$

The first of these equations can be solved for λ^{*} to yield

$$\lambda^* = -(\nabla\hat{f}(x^*)B^{-1})^T \tag{83}$$

which when substituted into the second equation yields

$$\nabla\bar{f}(x^*) - \nabla\hat{f}(x^*)B^{-1}C = 0 \tag{84}$$

This is precisely the reduced gradient condition derived earlier. By using linear approximations to the equality constraints two things have been accomplished:

1. A set of linear equations has been obtained to estimate the values of $\hat{x}$ corresponding to any perturbation of the nonbasic variables $\bar{x}$ about the point $\mathbf{x}^{(1)}$ such that the resulting point is feasible for the first-order approximation to the nonlinear constraints.

2. A direct first-order necessary condition for the equality-constrained problem has been derived that implicitly accounts for the equality constraints.

The Basic GRG Algorithm. In the previous section, linearization of the constraints was used to express the objective function as an unconstrained function of $n - m$ independent variables. As a consequence of this construction, the problem can be treated as an unconstrained optimization problem - at least for small excursions about the base point for the linearization - and presumably it can be solved using any of the variety of unconstrained gradient methods. Thus the following algorithm suggests itself.

At iteration k, suppose the feasible point $\mathbf{x}^{(k)}$ is available along with the partition $x = (\hat{x}, \bar{x})$ which has associated with it a constraint gradient submatrix $\mathbf{B}$ with nonzero determinant.

Step 1. Calculate the reduced gradient

$$\nabla\tilde{f} = \nabla\bar{f}(x^{(k)}) - \nabla\hat{f}(x^{(k)})B^{-1}C \tag{85}$$

Step 2. If $\|\nabla\tilde{f}\| \leq \varepsilon$ stop. Otherwise, set

$$\bar{d} = (-\nabla\tilde{f})^T \tag{86}$$

$$\hat{d} = -B^{-1}C\bar{d} \tag{87}$$

$$d = (\hat{d},\bar{d})^T \tag{88}$$

Step 3. Minimize $f(x^{(k)} + \alpha d)$ with respect to the scalar parameter α. Let $\alpha^{(k)}$ be the optimizing α, set

$$x^{k+1} = x^{(k)} + \alpha^{(k)}d \tag{89}$$

and go to step 1.

This prototype algorithm will now be refined.

It is easy to verify that regardless of the degree of nonlinearity, the direction **d** generated in the fashion is a descent direction. From a first-order Taylor's expansion of Eq. (75), we have that

$$f(x) - f(x^{(k)}) \approx \tilde{f}(x) - \tilde{f}(x^{(k)}) = \nabla\tilde{f}(x^{(k)})(\bar{x} - \bar{x}^{(k)}) = \alpha\nabla\tilde{f}(x^{(k)})\bar{d} \tag{90}$$

Thus, if $\bar{d}$ is chosen as

$$\bar{d} = -\nabla\tilde{f}(x^{(k)}) \tag{91}$$

then it follows that

$$f(x) - f(x^{(k)}) \approx \alpha(\nabla\tilde{f})^T(-\nabla\tilde{f}) = -\alpha\|\nabla\tilde{f}\|^2 \tag{92}$$

The right-hand side of this expression is less than zero for all positive values of α. Consequently, for all positive α's small enough that the linear approximations used are sufficiently accurate, it follows that

$$f(x) - f(x^{(k)}) < 0 \tag{93}$$

and consequently **d** is a descent direction.

While it is thus evident that step 2 produces a descent direction, it is not at all clear that the points generated in following that direction will be feasible. In fact, for nonlinear constraints, **d** will not be a feasible direction. Rather, because it is constructed using a linearization of the equality constraints, it is almost certain to lead to points away from the constraints. It has been shown that the vector $\bar{d}$ is a descent direction in the space of nonbasic variables $\bar{x}$ but that the composite direction vector

$$d = \begin{pmatrix} \hat{d} \\ \bar{d} \end{pmatrix} \tag{94}$$

where $\hat{d}$, calculated via the linear equation

$$\hat{d} = -B^{-1}C\bar{d} \tag{95}$$

yields infeasible points. Since $\bar{d}$ has desirable properties but the addition of $\hat{d}$ causes undesirable consequences, it is necessary that the computation of $\hat{d}$ should be revised. $\bar{d}$ is calculated using the linearization, but then, rather than calculating $\hat{d}$ as in step 2 and minimizing *f* along the line fixed by **d**, $\bar{d}$ is projected onto the constraint surface and *f* is minimized along the resulting curve. This means that for every value of α that is selected as a trial, the constraint equation will have to be solved for the values of the dependent variables $\hat{x}$ that will cause the resulting point to be feasible.

Thus step 3 of the prototype algorithm must be replaced by an iterative procedure that will, for a given value of α, iterate on the constraints to calculate a value of $\hat{x}$ that satisfies the constraints. This can be accomplished by using, for instance, Newton's method to solve the set of equations

$$h_i(\bar{x}^{(k)} + \alpha\bar{d}, \hat{x}) = 0 \quad i=1,2,\dots,m \tag{96}$$

Assuming that Newton's method will converge to an $\hat{x}$ that satisfies the constraints, then f(**x**) can be evaluated at that point to determine if an improvement over $\mathbf{x}^{(k)}$ has been achieved. If no improvement has been achieved, then α must be reduced using some search logic, and the Newton iteration must be repeated. If *f*(**x**) is improved, then the algorithm can either continue with a new value of α until no further improvement in *f*(**x**) can be obtained, or else, as is more typically done, the current point can be retained and a new direction vector calculated.

It is possible that for the α selected, the Newton iterations may fail to find an $\hat{x}$ that satisfies the constraints. This indicates that α was chosen too large and should be reduced. If this occurs, then α must be reduced and another cycle of Newton iterations initiated.

Extensions of the Basic Method. The preceding development of the basic *GRG* algorithm has been restricted to equality constraints and has resulted in essentially a gradient method operating within the reduced space of the nonbasic variables. In this section extensions of the basic algorithm to accommodate the general *NLP* problem with both upper and lower variable bounds as well as the inequality constraints will be considered:

$$\begin{aligned} &Min\ f(\boldsymbol{x}) \\ subject\ to\ a_j \le\ & g_j(\boldsymbol{x}) \le b_j \quad j=1,2,...,p \\ & h_k(\boldsymbol{x}) = 0 \quad k=1,2,...,m \\ x_i^{(L)} \le\ & x_i \le x_i^{(U)} \quad i=1,2,...,n \end{aligned} \tag{97}$$

Treatment of bounds: Variable upper and lower bounds can be accommodated either explicitly by treating each bound as an inequality constraint or implicitly by accounting for the bounds within the appropriate steps of the algorithm. The latter approach is clearly preferable, since it results in a smaller matrix **B** that must be inverted. To incorporate such constraints into the algorithm in an implicit fashion, three changes are necessary:

1. A check must be made to ensure that only variables that are not on or very near their bounds are labelled as basic variables. This check is necessary to ensure that some free adjustment of the basic variables can always be undertaken.

This can be accomplished by simply ordering the variables according to distance from their nearest bound. Thus, at the current feasible point $\mathbf{x}^{(k)}$, let

$$z_i^{(k)} = \min\{(x_i^{(U)} - x_i^{(k)}),(x_i^{(k)} - x_i^{(L)})\} \tag{98}$$

The quantities $z_i^{(k)}$ can then be ordered by decreasing magnitude, and the variables associated with the first m can be selected as dependent variables. Of course, this partition of the variables must also result in a nonsingular **B**. Therefore, one or more of the first m variables may have to be rejected in favour of variables farther down the order before a nonsingular **B** results.

2. The direction vector $\bar{d}$ is modified to ensure that the bounds on the independent variables will not be violated if movement is undertaken in the $\bar{d}$ direction. This is

accomplished by setting

$$\bar{d}_i = \begin{cases} 0 & \text{if } \bar{x}_i = \bar{x}_i^{(U)} \text{ and } (\nabla f)_i < 0 \\ 0 & \text{if } \bar{x}_i = \bar{x}_i^{(L)} \text{ and } (\nabla f)_i > 0 \\ -(\nabla f)_i & \text{otherwise} \end{cases} \tag{99}$$

3. Checks must be inserted in step 3 of the basic *GRG* algorithm to ensure that the bounds are not exceeded either during the search on α or during the Newton iterations. The *GRG2* package [91] treats variable bounds separately.

Treatment of Inequalities:

General inequality constraints can be handled within the *GRG* framework either by explicitly writing these constraints as equalities using slack variables or by implicitly using the concept of active constraint set. In the former case, inequalities of the form

$$a_j \leq g_j(x) \leq b_j \tag{100}$$

are converted to equalities

$$h_{m+j}(x) = g_j(x) - x_{n+j} = 0 \tag{101}$$

by the introduction of slack variables x_{n+j}, together with the bounds

$$a_j \leq x_{n+j} \leq b_j \tag{102}$$

This approach is attractive because it allows all constraints to be treated in a uniform fashion without further restructuring of the basic algorithm. The disadvantage is that **B** must include a row for each $g_j(x)$ even if the inequalities are not binding. In addition, the problem dimensionality is increased.

If an active constraint strategy is employed, then at each iteration point $x^{(k)}$ the active inequalities must be identified and their linearizations added to those of the equalities to define B and C. The logic of step 3 of the basic algorithm must be modified to include test for the violation of the previously inactive constraint. If at target point $x^{(k)}+\alpha d$ some previously inactive constraint is violated, then the step size α must be adjusted and the Newton calculations repeated until a point satisfying both the active and inactive constraints is found. At the resulting points, any tight but previously inactive constraint must be added to the active constraint set, and any active constraint now not binding must be deleted from the active set. These considerations will clearly complicate the step 3 computations and will result in the need to continually readjust the size of $\boldsymbol{B}$ and $\boldsymbol{C}$ and update $\boldsymbol{B}^{-1}$.

The direction vector $\bar{d}$ can be updated using conjugate gradient or quasi-Newton methods. Provided that the variable partition is not altered, the use of such updates will substantially enhance the convergence rate. Whenever a change is made in the partition, the updating formula must be restarted, since the nature of the search subspace is altered. Thus, the modifications of direction vectors is most effective in the final stages of the iterations when no further significant partitioning changes are encountered. The *GRG2* package [91] was implemented in this work based on its good performance [127-129]. It uses slack variables to convert inequality constraints into equality constraints and uses active set strategy, i.e., constructs the basis from the Jacobian of the binding constraints only and adding new constraints to the active set if they become binding.

3.6 Lagrangian Method/Successive Quadratic Programming

General. This method is based on directly solving the *Lagrange* first-order necessary conditions. The method works in the *(n+m)* dimensional space.

Direct Quadratic Approximation. Given $\boldsymbol{x}^0$, an initial solution estimate, and a suitable method for solving *QP* subproblems. a plausible solution strategy would consist of the following steps:

Step 1. Formulate the *QP* problem, replacing $f(\boldsymbol{x})$ by its quadratic approximation and the constraints by their linear approximation:

$$\begin{aligned} \min \quad & \nabla f(\boldsymbol{x}^{(k)})^T d + \frac{1}{2} d^T \nabla^2 f(\boldsymbol{x}^{(k)}) d \\ \textit{subject to} \quad & h_i(\boldsymbol{x}^{(k)}) + \nabla h_i(\boldsymbol{x}^{(k)}) d = 0 \quad i=1,2,\ldots,m \\ & g_j(\boldsymbol{x}^{(k)}) + \nabla g_j(\boldsymbol{x}^{(k)})^T d \geq 0 \quad j=1,2,\ldots,p \end{aligned} \tag{103}$$

Step 2. Solve the *QP* problem, and set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}$.

Step 3. Check for convergence. If not converged, repeat from step 1.

Quadratic Approximation of the Lagrangian Functions. It is motivated by the observation that it is desirable to incorporate into the subproblem definition not only the curvature of the objective function but also that of the constraints. However it is preferable to deal with linearly constrained rather than quadratically constrained subproblems. This can be accomplished by making use of the *Lagrangian* function. The equality-constrained problem is first considered. The extension to inequality constraints will follow in a straightforward fashion.

For the problem

$$\begin{aligned} &\min f(\boldsymbol{x}) \\ &\text{subject to} \quad \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \end{aligned} \tag{104}$$

the necessary conditions for a point $\mathbf{x}^*$ to be a local minimum are that there exist a multiplier λ^* such that

$$\nabla_x l(\boldsymbol{x}^*,\lambda^*) = \nabla f^* + (\lambda^*)^T \nabla \boldsymbol{h}^* = \mathbf{0} \textit{ and } \boldsymbol{h}(\boldsymbol{x}^*) = \mathbf{0} \tag{105}$$

Sufficient conditions for x^* to be a local minimum are that conditions (105) hold and that the *Hessian* of the *Lagrangian* function,

$$\nabla_x^2 l(\boldsymbol{x}^*,\lambda^*) = \nabla^2 f^* + (\lambda^*)^T \nabla^2 \boldsymbol{h}^* \tag{106}$$

satisfy

$$\boldsymbol{d}^T \nabla_x^2 l \boldsymbol{d} > 0 \textit{ for all } \boldsymbol{d} \textit{ such that } (\nabla \boldsymbol{h}^*)^T \boldsymbol{d} = \mathbf{0} \tag{107}$$

Given some point $(\bar{x},\bar{\lambda})$, let us construct the following subproblem expressed in terms of the variables $\boldsymbol{d}$:

$$\min \ \nabla f(\bar{x}) \boldsymbol{d} + \frac{1}{2} \boldsymbol{d} \nabla_x^2 l(\bar{x},\bar{\lambda}) \boldsymbol{d} \tag{108}$$

$$\textit{subject to} \quad \boldsymbol{h}(\bar{x}) + \nabla \boldsymbol{h}(\bar{x})^T \boldsymbol{d} = \mathbf{0} \tag{109}$$

If $\boldsymbol{d}^* = \boldsymbol{0}$ is the solution to the problem consisting of (108) and (109), then $\bar{x}$ must satisfy the necessary conditions for a local minimum of the original problem. First note that if $\mathbf{d}^* = \mathbf{0}$ solves the subproblem, then (109) must be satisfied. Then

from (109) it follows that $h(\bar{x}) = 0$; in other words, $\bar{x}$ is a feasible point in the original problem. Next, since $d^* = 0$ is a solution to the problem consisting of (108) and (109), there must exist some λ^* such that the subproblem functions satisfy the *Lagrangian* necessary conditions at $\mathbf{d}^* = \mathbf{0}$. Thus, since the gradient of the objective function (108) with respect to d at $\mathbf{d}^* = \mathbf{0}$ is $\nabla f(\bar{x})$ and that of (109) is $\nabla h(\bar{x})$, it follows from the first-order necessary conditions for $\mathbf{d}^*$ to be a local minimum of (108)
that

$$\nabla_d \left[\nabla f(\bar{x})d + \frac{1}{2}\, d\nabla_x^2 l((\bar{x},\bar{\lambda})d\right]_{d^*=0} + (\lambda^*)^T\, \nabla_d \left[h(\bar{x}) + \nabla h(\bar{x})^T d\right]_{d^*=0} = \tag{110}$$

$$\nabla f(\bar{x}) + (\lambda^*)^T \nabla h(\bar{x}) = \mathbf{0} \tag{111}$$

Then λ^* will serve as Lagrange multiplier for the original problem, and thus $\bar{x}$ satisfies the necessary conditions for a local minimum.

If the subproblem is reformulated with $(\bar{x},\lambda^*)$ instead of $(\bar{x},\bar{\lambda})$, (111) still holds so that $\mathbf{d} = \mathbf{0}$ satisfies the necessary conditions. Suppose the subproblem also satisfies the second-order sufficient conditions at $\mathbf{d} = \mathbf{0}$ with λ^*.

The Lagrangian of the problem consisting of (108) and (109) is

$$\nabla f(\bar{x})d + \frac{1}{2}\, d\ \nabla_x^2 l(\bar{x},\lambda)d + (\lambda)^T[h(\bar{x}) + \nabla h(\bar{x})^T d]) \tag{112}$$

The second order sufficiency conditions state that the second derivative of the Lagrangian (112) must be positive definite for all d such that

$$\nabla_d \left[h(\bar{x}) + \nabla h(\bar{x})^T d\ \right]^T d = \mathbf{0} \tag{113}$$

Note that the second derivative with respect of d of (112) is $\nabla_x^2 l(\bar{x},\lambda^*)$ and (113) reduces to $\nabla h(\bar{x})^T d = \mathbf{0}$. Then, it must be true that

$$d^T \nabla_x^2 l(\bar{x},\lambda^*)d > 0 \textit{ for all } d \textit{ such that } \nabla h(x)^T d = \mathbf{0} \tag{114}$$

Therefore, the pair $(\bar{x},\lambda^*)$ satisfies the sufficient conditions for a local minimum of the original problem.

This demonstration indicates that the subproblem consisting of (108) and (109) has the following very interesting features:

1. If no further corrections can be found, that is, $\mathbf{d} = \mathbf{0}$, then the local minimum of the original problem will have been obtained.

2. The Lagrange multipliers of the subproblem can be used conveniently as estimates of the multipliers used to formulate the next subproblem.

3. For points sufficiently close to the solution of the original problem the quadratic objective function is likely to be positive definite, and thus the solution of the *QP* subproblem will be well behaved.

By making use of the sufficient conditions stated for both equality and inequality constraints, it is easy to arrive at a *QP* subproblem formulation for the general case involving m equality and p inequality constraints. If we let

$$l(x,\lambda,\mu) = f(x) + \sum \lambda_i h_i(x) - \sum \mu_j g_j(x) \tag{115}$$

then at some point $(\bar{x},\bar{\lambda},\bar{\mu})$ the subproblem becomes

$$\begin{aligned} \min \quad & q(d;\bar{x}) \equiv \nabla f(\bar{x})^T d + \frac{1}{2} d^T \nabla_x l(\bar{x},\bar{\lambda},\bar{\mu}) d \\ \textit{subject to} \quad & \tilde{h}_i(d;\bar{x}) \equiv h_i(\bar{x}) + \nabla h_i(\bar{x})^T d = 0 \quad i=1,2,\ldots,m \\ & \tilde{g}_j(d;\bar{x}) \equiv g_j(\bar{x}) + \nabla g_j(\bar{x})^T d \geq 0 \quad j=1,2,\ldots,p \end{aligned} \tag{116}$$

The algorithm retains the basic steps outlined for the direct *QP* case. Namely, given an initial estimate $\mathbf{x}^0$ as well as λ^0 and μ^0, the subproblem [Eq. (116)] is formulated and solved; set $x^{(k+1)} = x^{(k)} + d$; check for convergence; and repeat, using the next estimates of λ and μ the corresponding multipliers obtained at the solution of the subproblem.

Constrained Variable Metric Method. Given initial estimates of $\mathbf{x}^0,\lambda^0,\mu^0$, and a symmetric positive definite matrix $\mathbf{H}^0$,

Step 1. Solve the problem

$$\begin{aligned} & \min \quad \nabla f(x^{(k)})^T d + \frac{1}{2} d^T H^{(k)} d \\ \text{subject to} \quad & h_i(x^{(k)}) + \nabla h_i(x^{(k)}) d = 0 \quad i=1,2,\ldots,m \\ & g_j(x^{(k)}) + \nabla g_j(x^{(k)}) d \geq 0 \quad j=1,2,\ldots,p \end{aligned} \tag{117}$$

Step 2. Select the step size α along $d^{(k)}$, and set

$$x^{(k+1)} = x^{(k)} + \alpha d^{(k)} \tag{118}$$

Step 3. Check for convergence.

Step 4. Update $H^{(k)}$, using the gradient difference

$$\nabla_x l(x^{(k+1)},\lambda^{(k+1)},\mu^{(k+1)}) - \nabla_x l(x^{(k)},\lambda^{(k+1)},\mu^{(k+1)}) \tag{119}$$

in such a way that $H^{(k+1)}$ remains positive definite.

In order to ensure convergence, it is necessary to find a suitable merit(penalty) function - a merit function that is compatible with the direction-finding algorithm in the sense that it decreases along the direction generated. It is a somewhat arbitrary function, defined for the sole purpose of guiding and measuring the progress of an algorithm. It is defined so that it is minimized at the solution to the original problem, and under appropriate circumstances it will serve as a descent function for an algorithm, decreasing in value at each step. The quadratic programming subproblem solver VE17AD of the VF13AD package [92] uses an active set strategy, i.e., a subset of the constraints that are satisfied as equalities by the current estimate of the solution vector is considered at each iteration.

3.7 Discussion

Although control vector parameterization can be used to solve distributed parameter systems, in this work only an initial value 4th order Runge-Kutta ODE solver has been implemented. In order to make the package more robust and to enable it to solve more problems in case one of the solver fails to find the optimal solution of a problem, two nonlinear programming problem solvers have been included into the optimization package. The GRG2 and VF13AD packages were chosen as two very reliable packages, each representing the currently known 2 best nonlinear programming methods available [127-129]. Chemical engineering problems have states and controls that represent quantities like temperature or concentration. In this work orthogonal collocation on finite elements has been chosen as a method of weighted residuals , because the linear combination coefficients of the Lagrange interpolation polynomials are the values of the functions to be approximated at the collocation points, so that the coefficients are physically meaningful quantities. This becomes useful when providing variable bounds, initializing state or control functions or interpreting state or control functions. Also the formulas are easily programmable. The property (18) of the Lagrange interpolation polynomials is particularly advantageous in making programming of the formulas simpler.

3.8 Conclusions

In this work, control vector parameterization can only be used for models containing initial value ordinary differential equations. If there are only such differential equations and their number is at least 3 and there are no algebraic constraints in the model, then control vector parameterization is the preferred approach for solving an optimization problem. If the upper limit of the integration is an optimization variable, then control vector parameterization is the only possible approach to employ.

Although in the literature, successive quadratic programming, an infeasible path method, is highly praised as the method used to solve DAOP problems arising in chemical engineering [94], the generalized reduced gradient method, a feasible path method, seems more suitable to solve models containing logarithmic and exponential functions, square roots and fractions because problems are less likely to occur in the GRG2 package due to negative or too large arguments in functions or zero denominators.

Orthogonal collocation of finite elements combined with nonlinear programming is an approach that can be used to solve the most general differential-algebraic optimization problems. The only disadvantage of the method is the dimensionality that precludes its current use for differential-algebraic optimization problems containing *PDEs* having 4 independent variables.

Chapter 4

Generating the Nonlinear Programming Problem

4.1 Introduction

Orthogonal collocation on finite elements is used in this work to discretize the differential equations and the initial or boundary conditions and to transform the objective functional into an objective function. The differential equations and the initial or boundary conditions will give rise to algebraic equations after the unknown functions and their derivatives have been replaced by a linear combination of *Lagrange* interpolation polynomials and their derivatives. This will enable any correctly posed problem within the limitations of the next section to be described as a NLP problem. This description can be developed automatically using the computer program to be described. The program provides an input language for the description of user's problems in the form of a model. Some parts of the program may have to be written by the user to describe particular requirements.

4.2 Problem Definition

Let us consider the following problem:

$$\begin{aligned}
&\min_{u_1,u_2(x),u_3(y),u_4(x,y)} \quad G_1(x,y,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y)))\big|_{x=x_f,y=y_f} \\
&+ \int_0^{x_f} G_2(x,c_1,c_2(x),c_4(x,y_f),p_1,p_2(x),u_1,u_2(x))dx + \int_0^{y_f} G_3(y,c_1,c_3(y),c_4(x_f,y),p_1,p_3(y),u_1,u_3(y))dy \\
&+ \int_0^{x_f}\int_0^{y_f} G_4(x,y,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y))dxdy
\end{aligned} \qquad (1)$$

subject to $g(x,y,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y)) \leq 0$

$h(x,y,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y)) = 0$

$f(x,y,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),$
$u_1,u_2(x),u_3(y),u_4(x,y),dc_2(x)/dx,d^2c_2(x)/dx^2,$
$dc_3(y)/dy,d^2c_3(y)/dy^2,\partial c_4(x,y)/\partial x,\partial^2 c_4(x,y)/\partial x^2,$
$\partial c_4(x,y)/\partial y,\partial^2 c_4(x,y)/\partial y^2,\partial^2 c_4(x,y)/\partial x\partial y) = 0 \quad x \in \Omega 1,\ y \in \Omega 2,\ (x,y) \in \Omega 3$

$c_2 = \bar{c}_2 \ at\ x \in \partial\ \Omega 1$

$c_3 = \bar{c}_3 \ at\ y \in \partial\ \Omega 2$

$c_4 = \bar{c}_4 \ at\ (x,y) \in \partial \Omega 3$

$u_1^L \leq u_1 \leq u_1^U$

$u_2^L(x) \leq u_2(x) \leq u_2^U(x)$

$u_3^L(y) \leq u_3(y) \leq u_3^U(y)$

$u_4^L(x,y) \leq u_4(x,y) \leq u_4^U(x,y)$

where x,y = independent variables

$\mathbf{c}_1$ = state variable vector

$\mathbf{c}_2(x),\mathbf{c}_3(y),\mathbf{c}_4(x,y)$ = state function vectors

$\mathbf{u}_1$ = control variable vector

$\mathbf{u}_2(x),\mathbf{u}_3(y),\mathbf{u}_4(x,y)$ = control function vectors

$\mathbf{p}_1$ = parameter vector

$\mathbf{p}_2(x),\mathbf{p}_3(y),\mathbf{p}_4(x,y)$ = parameter function vectors

$\mathbf{u}_1^L,\mathbf{u}_1^U,\mathbf{u}_2^L(x),\mathbf{u}_2^U(x),\mathbf{u}_3^L(y),\mathbf{u}_3^U(y),\mathbf{u}_4^L(x,y),\mathbf{u}_4^U(x,y)$ = control bounds

$\mathbf{g},\mathbf{h}$ = design constraint vectors

$\mathbf{f}$ = vector of *ODE*s and/or *PDE*s

$\Omega 1$ = integration domain of x

$\Omega 2$ = integration domain of y

$\Omega 3$ = integration domain of (x,y) which should be rectangular

$\partial\Omega 1$ = boundary of $\Omega 1$

$\partial\Omega 2$ = boundary of $\Omega 2$

$\partial\Omega 3$ = boundary of $\Omega 3$

4.3 Domain Discretization

The rectangular grid *G* is defined by the parameters

LB_1,UB_1 : left and right endpoints of x-interval

LB_2,UB_2 : bottom and top endpoints of y-interval

N_x	: number of finite elements along x-interval
N_y	: number of finite elements along y-interval
N_i	: number of internal collocation points in the i-th finite element along x-interval
M_j	: number of internal collocation points in the j-th finite element along y-interval

The collocation points are chosen the roots of Legendre polynomials in the interval [0,1]. Each finite element is mapped in the interval [0,1]. The domain is discretized into rectangular elements with the boundaries

$$\begin{aligned} x &= x_1, x_2, \ldots, x_k, \ldots, x_{N_x+1} \\ y &= y_1, y_2, \ldots, y_p, \ldots, y_{N_y+1} \end{aligned} \tag{2}$$

There are $N_x N_y$ rectangular elements and

$$\left[\left(\sum_{i=1}^{N_x} N_i\right) + N_x + 1\right]\left[\left(\sum_{j=1}^{N_y} M_j\right) + N_y + 1\right] \tag{3}$$

grid points.

The grid points along x-interval are $root1_{ncol_1, N_x}$ and along y-interval $root2_{ncol_2, N_y}$ where

$$ncol_1 = \max_{1 \le i \le N_x} N_i + 2 \tag{4}$$

and

$$ncol_2 = \max_{1 \le i \le N_y} M_i + 2 \tag{5}$$

4.4 Discretization of Differential Equations

A *PDE* problem will be assumed. For each grid point in the rectangular integration domain, where there are no initial or boundary conditions, an algebraic equation is generated through scanning the *PDE* and replacing variables and derivatives. *ODE*s are handled in the same way, only the discretization is done along x-interval or y-interval depending on whether the variables in the *ODE* are dependent on the first or the second independent variable, and not over a rectangular integration domain. Every state and control variable maps into a range of *NLP* variables. The ranges are defined as follows.

The start index of a range, referred to as a base, is 1 for the first variable, range(first variable) + 1 for the second variable, etc. The range for a state variable *c(x)* is

$$\left[\sum_{p=1}^{N_x} N_p\right] + N_x + 1 \tag{6}$$

The range of a state variable *c(y)* is

$$\left[\sum_{p=1}^{N_y} M_p\right] + N_y + 1 \tag{7}$$

The range of a state variable *c(x,y)* is

$$\left\{\left[\sum_{p=1}^{N_x} N_p\right] + N_x + 1\right\}\left\{\left[\sum_{p=1}^{N_y} M_p\right] + N_y + 1\right\} \tag{8}$$

The range of a control variable *w(x)* is

$$\sum_{p=1}^{N_x} N_p \tag{9}$$

The range of a control variable *w(y)* is

$$\sum_{p=1}^{N_y} M_p \tag{10}$$

The range of a control variable *w(x,y* is

$$\left[\sum_{p=1}^{N_x} N_p\right]\left[\sum_{p=1}^{N_y} M_p\right] \tag{11}$$

The range of a control variable which is dependent on the first independent variable and is a piecewise constant function along the x-interval is N_x.

The range of a control variable which is dependent on the second independent variable and is a piecewise constant function along the y-interval is N_y.

The range of a control variable $w(x,y)$, when $w(x,y)$ is a piecewise constant function over the x-interval is

$$N_x \left[\sum_{p=1}^{N_y} M_p \right] \tag{12}$$

The range of a control variable $w(x,y)$, when $w(x,y)$ is a piecewise constant function over the y-interval is

$$N_y \left[\sum_{p=1}^{N_x} N_p \right] \tag{13}$$

The range of a control variable $w(x,y)$, when $w(x,y)$ is a piecewise constant function over 2-dimensional finite elements is $(N_x)(N_y)$. The range of a control variable $w(x), w(y)$ or $w(x,y)$, constant over the integration domain is 1.

4.4.1 Replacing Independent Variables

In the polynomial approximation the independent variables x and y are mapped into the interval [0,1] using the transformation

$$u = \frac{x - x_k}{\Delta x_k} \quad and \quad v = \frac{y - y_l}{\Delta y_l} \tag{14}$$

When an independent variable appears in the model, when an equation is collocated at (u_i,v_j), the inverse transformation has to be performed to compute the correct value of the independent variable. Therefore the independent variable x in the kl-th element at the collocation point (u_i,v_j), is replaced by $u_i\Delta x_k + x_k$, where Δx_k is the length of the k-th finite element along the x-interval, x_k is the left endpoint of the k-th finite element along the x-interval, and y is replaced by $v_j\Delta y_l+y_l$, where Δy_l is the length of the l-th finite element along the y-interval and y_l is the bottom endpoint of the l-th finite element along the y-interval.

4.4.2 Replacing State Variables.

a) A state variable $c(x,y)$, dependent on both independent variables in the kl-th element at the collocation point (u_i,v_j), is replaced by the NLP variable x_{ind} where

$$ind = base - 1 + I_1 n + I_2 + (i-1)n + j \tag{15}$$

where

$$I_1 = \left[\sum_{p=1}^{k-1} N_p\right] + k - 1,\ I_2 = \left[\sum_{p=1}^{l-1} M_p\right] + l - 1,\ n = \left[\sum_{p=1}^{N_y} M_p\right] + N_y + 1 \tag{16}$$

and *base* is the start index of the range of indices corresponding to the state variables.

b) a state variable $c(x)$, dependent on the first independent variable in the kl-th element at the collocation point (u_i,v_j), is replaced by the NLP variable x_{ind} where

$$ind = base - 1 + \left[\sum_{p=1}^{k-1} N_p\right] + k + i - 1 \tag{17}$$

c) a state variable $c(y)$, dependent on the second independent variable in the kl-th element at the collocation point (u_i,v_j), is replaced by the NLP variable x_{ind} where

$$ind = base - 1 + \left[\sum_{p=1}^{l-1} M_p\right] + l + j - 1 \tag{18}$$

4.4.3 Replacing Control Variables

a) a control variable $w(x)$, dependent on the first independent variable in the kl-th element at the collocation point (u_i,v_j), if $1 < i < N_k + 2$, is replaced by the NLP variable x_{ind}, where

$$ind = base - 1 + \left[\sum_{p=1}^{k-1} N_p\right] + i - 1 \tag{19}$$

If $i = 1$ or $i = N_k + 2$ then $w(x)$ is replaced by

$$\sum_{n=2}^{N_k+1} x_{ind}\, l_n(u_i) \text{ where } ind = base + \left[\sum_{p=1}^{k-1} N_p\right] - 2 + n \tag{20}$$

where $u_i = 0$ if $i = 1$ and $u_i = 1$ if $i = N_k + 2$

b) a control variable $w(y)$, dependent on the second independent variable in the *kl*-th element at the collocation point (u_i,v_j), if $1 < j < M_l + 2$, is replaced by the *NLP* variable x_{ind}, where

$$ind = base - 1 + \left[\sum_{p=1}^{l-1} M_p\right] + j - 1 \tag{21}$$

If $j = 1$ or $j = M_l + 2$ then $w(y)$ is replaced by

$$\sum_{n=2}^{M_l+1} x_{ind}\, l_n(v_j) \text{ where } ind = base + \left[\sum_{p=1}^{l-1} M_p\right] - 2 + n \tag{22}$$

where $v_j = 0$ if $j = 1$ and $v_j = 1$ if $j = M_l + 2$

c) A control variable $w(x,y)$, dependent on both independent variables in the *kl*-th element at the collocation point (u_i,v_j), when $1 < i < N_k + 2$ and $1 < j < M_l + 2$, is replaced by the *NLP* variable x_{ind} where

$$ind = base - 1 + I_1 n + I_2 + (i-2)n + j - 1 \tag{23}$$

where

$$I_1 = \sum_{p=1}^{k-1} N_p \,,\; I_2 = \sum_{p=1}^{l-1} M_p \,,\; n = \sum_{p=1}^{N_y} M_p \tag{24}$$

and *base* is the start index of the range of indices corresponding to the control variable.

If $i = 1$ or $i = N_k + 2$ and $1 < j < M_l + 1$ then $w(x,y)$ is replaced by

$$\sum_{p=2}^{N_k+1} x_{ind}\, l_p(u_i) \tag{25}$$

where I_1, I_2 and n are the same as above, $u_i = 0$ if $i = 1$ and $u_i = 1$ if $i = N_k + 2$ and

$$ind = base - 1 + I_1 n + I_2 + (p-2)n + j - 1 \tag{26}$$

If $j = 1$ or $j = M_l + 2$ and $1 < i < N_k + 1$ then $w(x,y)$ is replaced by

$$\sum_{s=2}^{M_l+1} x_{ind}\, l_s(v_j) \tag{27}$$

where I_1, I_2 and n is the same as above, $v_j = 0$ if $j = 1$ and $v_j = 1$ if $j = M_l + 2$ and

$$ind = base - 1 + I_1 n + I_2 + (i-2)n + s - 1 \tag{28}$$

If $i = 1$ or $i = N_k + 2$ and $j = 1$ or $j = M_l + 2$ then $w(x,y)$ is replaced by

$$\sum_{p=2}^{N_k+1} \sum_{s=2}^{M_l+1} x_{ind}\, l_p(u_i) l_s(v_j) \tag{29}$$

where I_1, I_2 and n is the same as above, $u_i = 0$ if $i = 1$ and $u_i = 1$ if $i = N_k + 2$, $v_j =$ 0 if $j = 1$ and $v_j = 1$ if $j = M_l + 2$ and

$$ind = base - 1 + I_1 n + I_2 + (p-2)n + s - 1 \tag{30}$$

d) a control variable $w(x)$, dependent on the first independent variable and a piecewise constant function along the x-interval in the kl-th element at collocation point (u_i,v_j), is replaced by the *NLP* variable x_{ind}, where $ind = base - 1 + k$

e) a control variable $w(y)$, dependent on the second independent variable and a piecewise constant function along the y-interval in the kl-th element at collocation point (u_i,v_j), is replaced by the *NLP* variable x_{ind}, where $ind = base - 1 + l$

f) a control variable $w(x,y)$, dependent on both independent variables and a piecewise constant function along the x-interval in the kl-th element at collocation point (u_i,v_j), and $1 < j < M_l + 2$, is replaced by the *NLP* variable x_{ind}, where

$$ind = base - 1 + (k-1)n + I_2 + j - 1 \tag{31}$$

where

$$I_2 = \sum_{p=1}^{l-1} M_p, \; n = \sum_{p=1}^{N_y} M_p \tag{32}$$

If $j = 1$ or $j = M_l + 2$ then $w(x,y)$ is replaced by

$$\sum_{s=2}^{M_l+1} x_{ind}\, l_s(v_j) \tag{33}$$

where

$$ind = base - 1 + (k-1)n + I_2 + s - 1 \tag{34}$$

I_2 and n are the same as above, $v_j = 0$ if $j = 1$ and $v_j = 1$ if $j = M_l + 2$

g) a control variable $w(x,y)$, dependent on both independent variables and a piecewise constant function along the y-interval in the kl-th element at collocation point (u_i,v_j), and $1 < i < N_k + 2$, is replaced by the *NLP* variable x_{ind}, where

$$ind = base - 1 + \left[\sum_{p=1}^{k-1} N_p\right] N_y + l + (i-2)N_y \tag{35}$$

If $i = 1$ or $i = N_k + 2$ then $w(x,y)$ is replaced by

$$\sum_{n=2}^{N_k+1} x_{ind}\, l_n(u_i) \tag{36}$$

where

$$ind = base - 1 + \left[\sum_{p=1}^{k-1} N_p\right] N_y + l + (n-2)N_y \tag{37}$$

where

$u_i = 0$ if $i = 1$ and $u_i = 1$ if $i = N_k + 2$

h) if a control variable $w(x,y)$, dependent on both independent variables and a piecewise constant function over the 2-dimensional finite elements, in the kl-th element, at collocation point (u_i,v_j), is replaced by x_{ind}, where

$$ind = (k-1)N_y + l \tag{38}$$

If a control variable is constant over the integration domain, in the kl-th element, at collocation point (u_i,v_j), is replaced by x_{base}

4.4.4 Replacing Derivatives of the State Variables

a) $dc(x)/dx$ in the kl-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta x_k}\sum_{p=1}^{N_k+2} x_{ind}\frac{dl_p(u_i)}{du} \tag{39}$$

where

Δx_k = the length of the k-th finite element along the x-interval and

$$ind = base + \left[\sum_{p=1}^{k-1} N_p\right] + k - 2 + p \tag{40}$$

b) $\partial c(x,y)/\partial x$ in the kl-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta x_k}\sum_{p=1}^{N_k+2} x_{ind}\frac{dl_p(u_i)}{du} \tag{41}$$

where

$$ind = base - 1 + I_1 n + I_2 + (p-1)n + j \tag{42}$$

and I_1, I_2 and n are the same as in 4.4.2 (a).

c) $d^2c(x)/dx^2$ in the kl-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta x_k^2}\sum_{p=1}^{N_k+2} x_{ind}\frac{d^2 l_p(u_i)}{du^2} \tag{43}$$

where ind is the same as in 4.4.4 (a).

d) $\partial^2 c(x,y)/\partial x^2$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta x_k^2} \sum_{p=1}^{N_k+2} x_{ind} \frac{d^2 l_p(u_i)}{du^2} \tag{44}$$

where *ind* is the same as in 4.4.2 (a)

e) $dc(y)/dy$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta y_l} \sum_{s=1}^{M_l+2} x_{ind} \frac{dl_s(v_j)}{dv} \tag{45}$$

where

Δy_l = the length of the *l*-th finite element along the *y*-interval and

$$ind = base + \left[\sum_{p=1}^{l-1} M_p\right] + l - 2 + s \tag{46}$$

f) $\partial c(x,y)/\partial y$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta y_l} \sum_{s=1}^{M_l+2} x_{ind} \frac{dl_s(v_j)}{dv} \tag{47}$$

where

$$ind = base - 1 + I_1 n + I_2 + (i-1)n + s \tag{48}$$

and I_1, I_2 and n are the same as in 4.4.2 (a).

g) $d^2c(y)/dy^2$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta y_l^2} \sum_{s=1}^{M_l+2} x_{ind} \frac{d^2 l_s(v_j)}{dv^2} \tag{49}$$

where *ind* is the same as in 4.4.4 (e).

h) $\partial^2 c(x,y)/\partial y^2$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta y_l^2} \sum_{s=1}^{M_l+2} x_{ind} \frac{d^2 l_s(v_j)}{dv^2} \tag{50}$$

where *ind* is the same as in 4.4.2 (a)

i) $\partial^2 c(x,y)/\partial x \partial y$ in the *kl*-th element at collocation point (u_i,v_j) is replaced by

$$\frac{1}{\Delta x_k \Delta y_l} \sum_{p=1}^{N_k+2} \sum_{s=1}^{M_l+2} x_{ind} \frac{dl_p(u_i)}{du} \frac{dl_s(v_j)}{dv} \tag{51}$$

where

$$ind = base - 1 + I_1 n + I_2 + (p-1)n + s \tag{52}$$

and I_1, I_2 and n are the same as in 4.4.2 (a).

4.4.5 Replacing Auxiliary Variables

The model can contain expressions like $K = K_0 A e^{-E/RT}$, where K is termed a parameter or auxiliary variable. Wherever K appears in the constraints or in the differential equations, an index is calculated and appended to the parameter. The dependence of K on the independent variables is defined by the dependence of the state or control variables on the independent variable. For example, if in the above expression T is dependent on x, then K is considered to be dependent on x. When a differential equation is discretized over the integration domain in the finite elements (kl) and at collocation points (u_i,v_j), the indices are calculated as follows:

a) $p(x)$ is replaced by p_n, where n is incremented when k or i is incremented, i.e., the discretization moves to a new collocation point along the x-interval.

b) $p(y)$ is replaced by p_n, where n is incremented when l or j is incremented, i.e., the discretization moves to a new collocation point along the y-interval.

c) $p(x,y)$ is replaced by p_n, where n is incremented when k, i, l or j is incremented, i.e., the discretization moves to a new collocation point.

d) $p(k)$, a parameter dependent on the first independent variable and a piecewise constant function of finite elements along the x-interval, is replaced by p_k.

e) $p(l)$, a parameter dependent on the second independent variable and a piecewise constant function of finite elements along the y-interval, is replaced by p_l.

f) $p(k,y)$, a parameter dependent on both independent variables and a piecewise constant function of finite elements along the x-interval, is replaced by p_n, where n is incremented when k,l or j is incremented, i.e., the discretization moves to a new finite element along the x-interval and to a new collocation point along the y-interval.

g) $p(x,l)$, a parameter dependent on both independent variables and a piecewise constant function of finite elements along the y-interval, is replaced by p_n, where n is incremented when k,l or i is incremented, i.e., the discretization moves to a new finite element along the y-interval and to a new collocation point along the x-interval.

h) $p(k,l)$, a parameter dependent on both independent variables and a piecewise constant function over the 2-dimensional finite elements (kl), is replaced by p_n, where $n = (k-1)N_y + l$.

4.4.6 Replacing Input Variables

Disturbances and input variables that come from another process units and thus are not available for manipulation, are functions of time or constants and must be specified in advance before the problem can be solved. They have to be specified in a program module *USERINT.FOR* by the user and referenced in the model. For example, if *FUNCTION FLOWRATE(T)* is specified in *USERINT.FOR*, it has to be referenced in the model as *FLOWRATE[T]*, where *T* is the independent variable time. Examples of this are discussed later.

4.4.7 Time Delay

Time delays arise in a wide range of applications, including paper making, chemical reactors, and distillation. Time delay may occur when a state, control or output variable is measured at time t, showing the value of the variable at $t - \tau$.

Time delay can arise due to transport delay in a pipe, and due to delay caused by chemical analysis of state, control or output variables. Constant or time-varying time-delays can be handled. For a constant time delay τ, the value and the initial function giving the value of the state, control or output variable at $-\tau \leq t \leq 0$ must be given. If both the time delay and the initial function are constant, the time delay is specified in the model, as $c(t-\tau;\theta,x)$, where c is a variable depending on independent variables t and x, τ is the constant time delay, and θ is the value of the initial function. In the case that either the time delay or the initial function is a function of time, the time delay must be specified in the model as *c(t-TIMEDELAY(I)*, where *I* is an integer, which is used in a computed *GOTO* statement in the *FUNCTION VARTIMEDELAY* in module *OCFEINT.FOR*. This must be written by the user, an example is given later.

4.4.8 Parameter Estimation

The package can be used to estimate unknown parameters in a lumped or distributed parameter model. The least-square estimates for the parameters will be computed.

For lumped parameter systems, the measurements have to be given in the main data file which is *OCFEINP.DAT* in the form of a least-square function, such as

$$\sum_{t\in\{t_1,\dots,t_j,\dots,t_m\}} \sum_{k=1}^{n} [a_k(t) - a_k(t_j)]^2 \tag{53}$$

where measurements of the variables $a_1,\dots a_n$ were taken at $t = t_1,\dots,t_m$ and the variables $a_1,\dots,a_n$ and parameters $k_1,\dots,k_p$ appear in the model.

For distributed parameter systems, the measurements have to be given in the data file *OCFEINP.DAT* in the form of a least-square function, such as

$$\sum_{x\in\{x_1,\dots,x_i,\dots,x_{m_1}\}} \sum_{y\in\{y_1,\dots,y_j,\dots,y_{m_2}\}} \sum_{k=1}^{n} [a_k(x,y) - a_k(x_i,y_j)]^2 \tag{54}$$

where measurements of the variables $a_1,\dots a_n$ were taken at

$x = x_1,\dots,x_{m_1}$ *and* $y = y_1,\dots,y_{m_2}$ and the variables $a_1,\dots,a_n$ and parameters $k_1,\dots,k_p$ appear in the model.

4.4.9 Irregular Domain

An irregular domain has to be embedded in a rectangular domain, and in the *FUNCTION WITHIN* in the module *OCFEINT.FOR* the interior of the irregular domain has to be specified. If we denote the irregular domain as D, the rectangular domain as R, $D \subset R$, and the differential equations are collocated at collocation points $(x,y) \in D$, and the boundary condition should hold in $(x,y) \in R \setminus D$.

4.5 Discretization of Expressions

Expressions like $k = k_0 e^{\frac{-E}{RT}}$ are discretized like differential equations, the only difference is that the discretized expressions do not become constraints in the *NLP* problem.

4.6 Discretization of Initial or Boundary Conditions

Initial or boundary conditions are discretized like differential equations, the condition is discretized at grid points where the condition holds.

4.7 Transformation of the Objective Functional

4.7.1 Transformation of an Objective Functional not Containing Integral Terms.

An objective functional that does not contain integral terms is discretized like differential equations, but usually there is only 1 collocation point, at $x = x_f$, $y = y_f$ where x_f is the end point of the x integration interval and y_f is the end point of the y integration interval.

4.7.2 Transformation of the Objective Functional Containing Integral Terms

An integral term $\int_{x_0}^{x_f} F(c,u,x)dx$ in the objective functional is transformed as follows:

Let

$$I = \int_{x_0}^{x_f} F(c,u,x)dx \tag{55}$$

Both sides are differentiated with respect to x.

$$\begin{aligned} \frac{dI}{dx} &= F(c,u,x) \quad x \in [x_0,x_f], \\ I(x_0) &= 0 \end{aligned} \tag{56}$$

The resulting differential equation is discretized in the usual manner. $I(x_f)$ replaces the integral term in the objective functional.

An integral term $\int_{y_0}^{y_f} F(c,u,y)dy$ is transformed in a similar fashion.

An integral term $\int_{x_0}^{x_f}\int_{y_0}^{y_f} F(c,u,x,y)dxdy$ is replaced in the objective functional by $I(x_f,y_f)$. The resulting differential equation

$$\frac{\partial^2 I}{\partial x \partial y} = F(c,u,x,y) \quad (x,y) \in [x_0,x_f]\times[y_0,y_f],\ I(x_0,y_0) = 0 \tag{57}$$

is discretized over the rectangular integration domain.

4.7.3 Transformation of the Least-square Objective Functional of Lumped Parameter Systems.

The objective functional is in the form of

$$\sum_{t\in\{t_1,\dots,t_i,\dots t_m\}} \sum_{s=1}^{n} [a_s(t)-a_s(t_i)]^2 \tag{58}$$

where measurements of variables $a_1,\dots,a_n$ were taken at $t = t_1,\dots,t_m$ and the variables $a_1,\dots,a_n$ and the parameters $k_1,\dots,k_p$ appear in the model, t_i is mapped into the interval [0,1] as

$$u_i = \frac{(t_i - t_k)}{\Delta t_k} \tag{59}$$

and a_s(t) is replaced by

$$\sum_{p=1}^{N_k+2} l_p(u_i)a_{sp}^k \tag{60}$$

where a_{sp}^k is the value of $a_s^k(t)$ at collocation point u_p in the k-th finite element. Thus we discretize the least-square objective functional in such a way that in

$\sum_{s=1}^{n} [a_s(t) - a_s(t_i)]^2$ $a_s(t)$ is replaced by its approximation and evaluated at $t = t_1,\dots,t_m$.

4.7.4 Transformation of the Least-square Objective Functional of Distributed Parameter Systems.

The objective functional is in the form of

$$\sum_{x\in\{x_1,\dots,x_i,\dots,x_{m_1}\}} \sum_{y\in\{y_1,\dots,y_j,\dots,y_{m_2}\}} \sum_{s=1}^{n} [a_s(x,y)-a_s(x_i,y_j)]^2 \tag{61}$$

where measurements of variables $a_1,\dots,a_n$ were taken at $x = x_1,\dots,x_{m_1}$ and $y = y_1,\dots,y_{m_2}$ and the variables $a_1,\dots,a_n$ and parameters $k_1,\dots,k_p$ appear in the model and x_i, y_j is mapped into the interval [0,1]x[0,1] as

$$u_i = \frac{(x_i - x_k)}{\Delta x_k} \qquad v_j = \frac{(y_j - y_l)}{\Delta y_l} \tag{62}$$

and a_s(x,y) is replaced by

$$\sum_{p=1}^{N_k+2} \sum_{q=1}^{M_l+2} l_p(u_i) l_q(v_j) a_{spq}^{kl} \tag{63}$$

where a_{spq}^{kl} is the value of $a_s^{kl}(x,y)$ at collocation point (u_p,v_q) in the *kl*-th finite element. Thus we discretize the least-square objective functional in such a way that in

$\sum_{s=1}^{n} [a_s(x,y) - a_s(x_i,y_j)]^2$ $a_s(x,y)$ is replaced by its approximation and evaluated

at $x = x_1,\dots,x_{m_1}$ *and* $y = y_1,\dots,y_{m_2}$.

4.8 Discretization of Constraints

The discretization of constraints is performed in the same manner as the discretization of differential equations, with the difference that we collocate at the boundary, too, because there are no initial or boundary conditions.

4.9 Completion of Equation Sets

In order to ensure that the resulting *NLP* problem is well posed, the number of *NLP* variables derived from the state variables and the number of algebraic equations derived from the discretization of differential equations and initial or boundary conditions must be the same.

There are 2 ways to create additional algebraic equations:

1. At those boundary points where there are no initial or boundary conditions, the state

function is approximated by a linear combination of *Lagrange* interpolation polynomials defined at other points in the finite element, excluding integration domain boundary points. At finite element boundaries the continuity of derivatives can be enforced.

2. The collocation method can be applied at boundaries of finite elements and at those integration domain boundary points where there are no initial or boundary conditions.

Computer runs have shown that the second approach resulted in smaller residual errors. An arbitrary decision had to be taken to which finite elements the points lying on a finite element boundary belong. A point at finite element boundary belongs to the finite element to the left or below of the point.

4.10 Conclusion

The user problem defined in (1) has been formulated as a *NLP* problem which can be solved by one of the methods *GRG2* or *SQP* already described.

Chapter 5
Interpolation and Error Evaluation

5.1 Interpolation

5.1.1 Introduction

The solution of the nonlinear programming problem yields the values of the state variables and of the control variables that are defined as continuous functions at the roots of the *Legendre* polynomials in each finite element. While this provides an efficient solution, it gives the values of state and control functions at discrete collocation points. The approximating polynomials are defined as continuous functions over each finite element. Although the user can construct the continuous approximating functions using information from the model and from the output file *OCFEOUT.DAT*, it was felt that the user should be given an easy way to obtain the values of the state and control functions at chosen equidistant points over the integration domain. This also facilitates data generation for creating 2-dimensional graphs or 3-dimensional surfaces.

5.1.2 Implementation

A program is generated that tabulates the approximated values of the state and control variables for a given Δx and Δy. The approximate values of the state and control variables are calculated as follows:

For a given (x^*,y^*), find finite element (kl), in which (x^*,y^*) lies. Then

$$u^* = \frac{(x^*-x_k)}{\Delta x_k}, \quad v^* = \frac{(y^*-y_l)}{\Delta y_l} \tag{1}$$

is calculated.

Find the index set $I = \{i_1,\ldots,i_n\}$ into which the state or control variable maps when the differential-algebraic optimization problem is converted into a *NLP* problem. If we denote the *n-th* state or control variables as $c_n(x)$, $c_n(y)$, $c_n(x,y)$, $w_n(x)$, $w_n(y)$, $w_n(x,y)$, $w_n(k)$, $w_n(l)$, $w_n(k,y)$, $w_n(x,l)$, $w_n(k,l)$, where $w_n(k)$ denotes a control variable dependent on the first independent variable and a piecewise constant function along the x-interval, $w_n(l)$ denotes a control variable dependent on the second independent variable and a piecewise constant function along the y-interval, $w_n(k,y)$ denotes a control variable dependent on both independent variable and a piecewise constant function along the x-interval, $w_n(x,l)$ denotes a control variable dependent on both independent variables and a piecewise constant function along the y-interval, $w_n(k,l)$ denotes a control variable dependent on both independent variables and a piecewise constant function over 2-dimensional finite elements, then

$$c_n^k(x^*) = \sum_{i=1}^{N_k+2} l_i(u^*)x_{ind} \tag{2}$$

where

$$ind = base(n) + \left[\sum_{p=1}^{k-1} N_p \right] + k + i - 2 \tag{3}$$

is the index of the *NLP* variable, into which the collocation point u_i and finite element k maps for the n-th variable.

$$c_n^l(y^*) = \sum_{j=1}^{M_l+2} l_j(v^*)x_{ind} \tag{4}$$

where

$$ind = base(n) + \left[\sum_{p=1}^{l-1} M_p \right] + l + j - 2 \tag{5}$$

is the index of the *NLP* variable, into which the collocation point v_j and finite element l maps for the n-th variable.

$$c_n^{kl}(x^*,y^*) = \sum_{i=1}^{N_k+2} \sum_{j=1}^{M_l+2} l_i(u^*)l_j(v^*)x_{ind} \tag{6}$$

where

$$ind = base(n) - 1 + I_1 I_3 + I_2 + (i-1)I_3 + j \tag{7}$$

where

$$I_1 = \left[\sum_{p=1}^{k-1} N_p\right] + k - 1,\ I_2 = \left[\sum_{p=1}^{l-1} M_p\right] + l - 1,\ I_3 = \left[\sum_{p=1}^{N_y} M_p\right] + N_y + 1 \tag{8}$$

and ind is the index of the *NLP* variable, into which the collocation point (u_i,v_j) and finite element (k,l) maps for the n-th variable.

$$w_n^k(x^*) = \sum_{i=2}^{N_k+1} l_i(u^*) x_{ind} \tag{9}$$

where

$$ind = base(n) + \left[\sum_{p=1}^{k-1} N_p\right] + i - 2 \tag{10}$$

$$w_n^l(y^*) = \sum_{j=2}^{M_l+1} l_j(v^*) x_{ind} \tag{11}$$

where

$$ind = base(n) + \left[\sum_{p=1}^{l-1} M_p\right] + j - 2 \tag{12}$$

$$w_n^{kl}(x^*,y^*) = \sum_{i=2}^{N_k+1} \sum_{j=2}^{M_l+1} l_i(u^*) l_j(v^*) x_{ind} \tag{13}$$

where

$$ind = base(n) + I_1 I_3 + I_2 + (i-2)I_3 + j - 2 \tag{14}$$

where

$$I_1 = \left[\sum_{p=1}^{k-1} N_p\right],\ I_2 = \left[\sum_{p=1}^{l-1} M_p\right],\ I_3 = \left[\sum_{p=1}^{N_y} M_p\right] \tag{15}$$

A control variable $w_n(k)$ is approximated by

$$w_n^k(x^*) = x_{ind} \tag{16}$$

where

$$ind = base(n) - 1 + k \tag{17}$$

is the index of the *NLP* variable, into which the finite element k maps for the n-th variable.

A control variable $w_n(l)$ is approximated by

$$w_n^l(y^*) = x_{ind} \tag{18}$$

where

$$ind + base(n) - 1 + l \tag{19}$$

is the index of the *NLP* variable, into which the finite element l maps for the n-th variable.

A control variable $w_n(k,y)$ is approximated by

$$w_n^{kl}(x^*,y^*) = \sum_{j=2}^{M_l+1} l_j(v^*)x_{ind} \tag{20}$$

where

$$ind = base(n) + (k-1)I_1 + I_2 + j - 2 \tag{21}$$

where

$$I_1 = \left[\sum_{p=1}^{N_y} M_p \right] I_2 = \left[\sum_{p=1}^{l-1} M_p \right] \tag{22}$$

and ind is the index of the *NLP* variable, into which the finite element (kl) and the collocation point v_j maps for the n-th variable.

A control variable $w_n(x,l)$ is approximated by

$$w_n^{kl}(x^*,y^*) = \sum_{i=2}^{N_t+1} l_i(u^*)x_{ind} \tag{23}$$

where

$$ind = base(n) - 1 + \left[\sum_{p=1}^{k-1} N_p\right] N_y + l + (i-2)N_y \tag{24}$$

is the index of the *NLP* variable, into which the finite element (kl) and the collocation point u_i maps for the n-th variable.

A control variable $w(k,l)$ is approximated by

$$w_n^{kl}(x^*,y^*) = x_{ind} \tag{25}$$

where

$$ind = (k-1)N_y + l \tag{26}$$

is the index of the *NLP* variable, into which the finite element (kl) maps for the n-th variable.

5.2 Error Evaluation

In general, it is impossible to ascertain when a given approximation accuracy has been reached since the exact solution is unknown [130]. The simplest approach is to compare results obtained with consecutive approximation order N and $N+1$. The error in the *N-th* order approximation will normally be much smaller than the difference between the *N-th* order and the *(N+1)th* order approximation.

The following example [130] shows that some a priori guidelines can be obtained for a particular problem. For the differential equation

$$\frac{d^2c}{dx^2} + \frac{1}{x}\frac{dc}{dx} = \Phi^2 c \qquad c(1) = 1 \qquad c'(0) = 0 \tag{1}$$

The maximum value of both sides is

$$\max\left(\frac{d^2c}{dx^2} + \frac{1}{x}\frac{dc}{dx}\right) = \max\ (\Phi^2 c)\ = \Phi^2 \tag{2}$$

If an approximate of the type

$$c_N = 1 + (1 - x^2)\sum_{i=1}^{N} a_i\, x^{2i-2} \tag{3}$$

is used and it is assumed that c_N is positive and has positive derivatives in [0,1], since the steepest possible function satisfying those condition being $c_N = x^{2N}$, then

$$\max\left(\frac{d^2c_N}{dx^2} + \frac{1}{x}\frac{dc_N}{dx}\right) = 4N^2 \tag{4}$$

i.e. the approximation order should at least satisfy

$$4N^2 > \Phi^2 \ or\ N > \frac{\Phi}{2} \tag{5}$$

The approximation error is = $\alpha\, h^s$ [86] where $h = max(h_x,h_y)$, where h_x = the size of a finite element in x-direction and h_y is the size of a finite element in y-direction, s is bounded between 3 and r+1, for polynomials of degree r. The exact solution has derivatives of at least degree s. The approximation error of an elliptic differential equation is [84]

$$C\left(\frac{1}{NP}\right)^{NP}\left(\frac{1}{NE}\right)^{\min(NP,k)} \tag{6}$$

where $N_x=N_y=NE$ and $N=M=NP$

NE is the number of finite elements and *NP* is the number of internal collocation points.

Parameter k characterizes the continuity of solution and C is independent of *NP* and *NE.*

Now the approximation error for *Lagrange* interpolation polynomials in the one-dimensional case will be derived [59].

Let $c \in C^{N_x+2}\,[a,b]$ and let $p(x)$ be the Lagrange polynomial of degree $N_x + 1$ interpolating $c(x)$ at $a = x_1 < x_2 < ... < x_{N_x+2} = b$. Let us define the function $g(x)$ by

$$g(x) = [p(x) - c(x)] - \frac{w(x)}{w(x')}\,[p(x') - c(x')]. \tag{7}$$

where $w(x) = (x - x_1)(x - x_2)...(x - x_{N_x+2})$. Since for $x = x_1,...,x_{N_x+2}$ $p(x) = c(x)$ and $g(x') = 0$, $g(x)$ has $N_x + 3$ zeros at $x_1,x_2,...,x_{N_x+2},x'$. Applying Rolle's theorem repeatedly $N_x + 2$ times to $g(x)$, it follows that $g^{(N_x+2)}(\xi) = 0$ for some $\xi \in [a,b]$. Since $p(x)$ is a polynomial of degree $N_x + 1$, $p^{(N_x+2)}(x) = 0$ and since $w(x)$ is a polynomial of degree $N_x + 2$ with a leading coefficient 1, $w^{(N_x+2)}(x) = (N_x+2)!$

$$g^{(N_x+2)}(\xi) = 0 = -\,c^{(N_x+2)}(\xi) - \frac{(N_x+2)!}{w(x')}\,[p(x') - c(x')], \tag{8}$$

Therefore

$$c(x') - p(x') = \frac{c^{(N_x+2)}(\xi)}{(N_x+2)!}\,w(x'). \tag{9}$$

If we let $h = max\ (\ x_{i+1} - x_i\ )$, $1 \le i \le N_x + 2$ and $x = \dfrac{x_{N_x+1} + x_{N_x+2}}{2}$ it follows that

$$|w(x)| = |(x-x_1)(x-x_2)...(x-x_{N_x+2})| \leq \frac{(N_x+1)!\ h^{N_x+2}}{4} \tag{10}$$

Thus

$$\|c-p\| \leq \frac{\|c^{N_x+2}\|\ h^{N_x+2}}{4(N_x+2)} \tag{11}$$

This result can be extended to the two-dimensional case as follows [59].

For each fixed x in [a,b] let $I_x c(x,y)$ be the Lagrange interpolate to $c(x,y)$ in the y direction; interpolating $c(x,y)$ at the knots π_y. Specifically,

$$I_x c(x,y) = \sum_{j=1}^{N_y+2} c(x,y_j) l_j(y). \tag{12}$$

For each fixed y in [c,d] let $I_y c(x,y)$ be the Lagrange interpolate to $c(x,y)$ in the x direction; interpolating $c(x,y)$ at the knots π_x. Then

$$I_y c(x,y) = \sum_{i=1}^{N_x+2} c(x_i,y) l_i(x). \tag{13}$$

Then

$$Ic(x,y) = I_x I_y c(x,y) = \sum_{j=1}^{N_y+2} \sum_{i=1}^{N_x+2} c(x_i,y_j) l_i(x) l_j(y). \tag{14}$$

is the Lagrange interpolate $p(x,y)$ to $c(x,y)$ on a rectangular grid. The operators I_x and I_y are commutative, giving

$$I_x I_y c = I_y I_x c = I_{xy} c = Ic = p \tag{15}$$

If $D_y^k c = \dfrac{\partial^k c}{\partial y^k}$ exists at a point (x,y) in R, it holds that

$$D_y^k I_y c(x,y) = D_y^k \sum_{i=1}^{N_x+2} c(x_i,y) l_i(x) = \sum_{i=1}^{N_x+2} l_i(x) \frac{\partial^k c(x_i,y)}{\partial y^k} = I_y D_y^k c(x,y) \tag{16}$$

and

$$D_x^k I_x c(x,y) = D_x^k \sum_{j=1}^{N_y+2} c(x,y_j) l_j(y) = \sum_{j=1}^{N_y+2} l_j(y) \, \frac{\partial^k c(x,y_j)}{\partial x^k} = I_x D_x^k c(x,y) \qquad (17)$$

when the partial derivatives exist.

Assume $c \in C^{N_x+2,N_y+2}[R]$. Since $I_x c(x,y)$ is the Lagrange interpolate to c in the y direction, applying (11)

$$\|c - I_x c\| \leq \frac{\|D_y^{N_y+2} c\| \; h^{N_y+2}}{4(N_y+2)} \qquad (18)$$

Now invoking the triangle inequality, with $I_{xy} = I_x I_y$

$$\|c - Ic\| \leq \|c - I_x c\| + \|I_x c - I_{xy} c\|. \qquad (19)$$

To estimate the second quantity on the right-hand side, from (15) we obtain

$$I_x c - I_{xy} c = (I_x c) - I_y(I_x c) \qquad (20)$$

$I_y(I_x c)$ is the Lagrange interpolate of degree $N_x + 1$ in the x direction to $I_x c(x,y)$ which interpolates $I_x c$ at the knots of π_x.

$$I_y(I_x c) = \sum_{i=1}^{N_x+2} l_i(x) \left[\sum_{j=1}^{N_y+2} c(x_i, y_j) l_j(y) \right] \qquad (21)$$

From (11) it follows that

$$\|I_x c - I_{xy} c\| \leq \frac{\|D_x^{N_x+2}(I_x c)\| \; h_x^{N_x+2}}{4(N_x+2)}, \qquad (22)$$

considering $I_x c$ as a function of x. From (17), $D_x^{N_x+2}(I_x c) = I_x(D_x^{N_x+2} c)$ is the Lagrange interpolate of degree $N_y + 1$ to $D_x^{N_x+2} c$ considered as a function of y with knots at π_y. Thus from (11)

$$\|D_x^{N_x+2} c - I_x \left[D_x^{N_x+2} c \right] \| \leq \frac{\|D_y^{N_y+2} D_x^{N_x+2} c\|}{4(N_y+2)} \; h_y^{N_y+2} \qquad (23)$$

so that

$$\|I_x \left[D_x^{N_x+2} c\right] \| = \|D_x^{N_x+2} \left[I_x c\right] \| \leq \|D_x^{N_x+2} c\| + \frac{\|D_y^{N_y+2} D_x^{N_x+2} c\|}{4(N_y+2)} h_y^{N_y+2} \tag{24}$$

From (18), (19), (22) and (24)

$$\|c - Ic\| \leq \frac{\|D_y^{N_y+2} c\|}{4(N_y+2)} h_y^{N_y+2} + \frac{1}{4(N_x+2)} \left[\|D_x^{N_x+2} c\| + \frac{\|D_y^{N_y+2} D_x^{N_x+2} c\|}{4(N_y+2)} h_y^{N_y+2} \right] h_x^{N_x+2} \tag{25}$$

Error estimates can also be derived for the derivatives of Lagrange polynomials.

Eqs. (11) and (25) give approximation error estimates interpolating c in the operator equation $Lc = d$, where L is the differential equation operator. We can estimate the approximation error to the solution of the differential equation as follows.
Let ε be the approximation error approximating c, ε_1 approximating $\frac{\partial c}{\partial x}$, ε_2 approximating $\frac{\partial c}{\partial y}$, ε_{11} approximating $\frac{\partial^2 c}{\partial x^2}$, ε_{22} approximating $\frac{\partial^2 c}{\partial y^2}$, ε_{12} approximating $\frac{\partial^2 c}{\partial x \partial y}$ then for a differential equation

$$\alpha c + \beta \frac{\partial c}{\partial x} + \gamma \frac{\partial c}{\partial y} + \delta \frac{\partial^2 c}{\partial x^2} + \xi \frac{\partial^2}{\partial y^2} + \zeta \frac{\partial^2 c}{\partial x \partial y} \tag{26}$$

the approximation error $\leq$

$$|\alpha|\varepsilon + |\beta|\varepsilon_1 + |\gamma|\varepsilon_2 + |\delta|\varepsilon_{11} + |\xi|\varepsilon_{22} + |\zeta|\varepsilon_{12} \tag{27}$$

There is also an iterate error to the solution of the algebraic equation superimposed to the approximation error as well as round-off error originating from the machine representation of the floating point numbers.

In this work the following error evaluation strategy has been used:
The principle term in the mean square approximation error is calculated as follows [105].
If $N_x = 1$ then

$$E_1 = \left(\frac{1}{NP}\right)^{NP} \tag{28}$$

otherwise

$$E_1 = \frac{\sum_{i=1}^{N_x} (\Delta x_i)^{N_i+2}}{N_x} \tag{29}$$

If $N_y = 1$ then

$$E_2 = \left(\frac{1}{NP}\right)^{NP} \tag{30}$$

otherwise

$$E_2 = \frac{\sum_{j=1}^{N_y} (\Delta y_j)^{M_j+2}}{N_y} \tag{31}$$

The approximation error =

$$\frac{E_1 + E_2}{2} \tag{32}$$

The residual error is calculated as follows.

Given Δx and Δy, the differential equations are discretized at the collocation points (x_0,y_0), $(x_0,y_0+\Delta y)$, ... etc. The discretization is similar to the described in chapter 4.4, with the difference that since the collocation points normally do not coincide with the roots of *Legendre* polynomials, the attribute of *Lagrange* polynomials, i.e.

$$l_n(u_i) = \begin{cases} 0 \; for \; i \neq n \\ 1 \; for \; i = n \end{cases} \tag{33}$$

cannot be utilized and therefore

$$\sum_{n=1}^{N+2} c_n^k l_n(u_i) \neq c_i^k\ ,\ \sum_{n=2}^{N+1} w_n^k l_n(u_i) \neq w_i^k,$$

$$\sum_{n=1}^{N+2}\sum_{m=1}^{M+2} c_{nm}^{kl} l_n(u_i) l_m(v_j) \neq c_{ij}^{kl}, \tag{34}$$

$$\sum_{n=2}^{N+1}\sum_{m=2}^{M+1} w_{nm}^{kl} l_n(u_i) l_m(v_j) \neq w_{ij}^{kl},$$

Therefore the discretized algebraic equations are more complex. Also, the collocation points *$(x_0+i\Delta x, y_0+j\Delta y)$* must be mapped into a finite element *(kl)* and into a collocation point *(u,v)* where

$$u = \frac{x-x_k}{x_{k+1}-x_k}\ ,\ v = \frac{y-y_l}{y_{l+1}-y_l} \tag{35}$$

before the differential equations are discretized.

Then after the *NLP* has been solved, the residuals at the grid points are evaluated with the optimal solution vector x.

Experience by running example problems has shown that the best strategy to define control functions as piecewise constant functions over finite elements and have 1 internal collocation points in each subinterval. Therefore the state functions will be approximated by a quadratic polynomial, and since we collocate at the boundaries of finite elements, the approximation will be exact at the corners and in the middle of the finite elements and the approximating function and the derivatives will not fluctuate between collocation points. Then, if the residual error at certain grid points *$(x_0+i\Delta x, y_0+j\Delta y)$* is unacceptably large, then we subdivide elements at these points and solve the new problem. Large error can occur if the exact optimal solution cannot be approximated by quadratic polynomials, then we need more finite elements to ensure that the 2-dimensional finite elements are sufficiently small so that in that finite element if the state function is approximated by a quadratic polynomial, the approximation error is sufficiently small.

Chapter 6
Implementation Strategy

6.1 Generating the Nonlinear Programming Problem

6.1.1 Introduction

The algorithms described in Chapter 4 have been implemented in a FORTRAN 77 program. The problem solution is carried out by a program called OPTIMIZER which must contain a program written to describe the user's problem. In some cases this will be written by the user, but more usually the problem is described in a data file and this is translated by another program, OCFE, which performs automatically the operations described in Chapter 4. Because of the large dimensionality of the resulting nonlinear programming problem, implementation has been limited to 2 independent variables.

6.1.2 Input File for OCFE

The user has to set up an input file, OCFEINP.DAT, describing the model to be solved. This file must contain character strings which describe the equations, together with numerical data. In this way the need for the user to write a program is avoided. Instead the equation is described in a natural way. The syntax of the input file is described in detail in Appendix A. The input file contains various parameters such as the numbers of equations of different types, the numbers and types of variables, initial or boundary conditions, the bounds of integration and the equations including the objective functional. Each variable and equation is classified with a type. Details are given in Appendix A. Also, the choice of GRG2 or SQP must be made by the user. If the variables are dependent on independent variables, they must be followed by their independent variables in parenthesis, e.g. c(x,y). The derivatives are indicated as shown in the following examples.

DC(X,Y)/DX

DC(X,Y)/DY

D2C(X,Y)/DX2

D2C(X,Y)/DY2

D2C(X,Y)/DXDY.

Algebraic variables must be denoted by *X* followed by their subscript, e.g. *X(1)*. Integrals are denoted by the word *INTEGRAL*. For a partial differential equation, boundary conditions, where present, have to be given, separately, on the sides and at the corners of the rectangular integration domain.

The following examples illustrate the difference between the mathematical model and the input file.

model	input
$\frac{dc_1}{dt} = k_1(T)c_1(t)^2$	DC1(T)/DT-K1(TEM)*C1(T)**2
$c_1(0) = 1$	C1(T)-1. AT T = 0
$\frac{\partial c_1(z,t)}{\partial t} - k_1(T)c_1^2(z,t)$	DC1(T,Z)/DT+K1(TEM)*C1(T,Z)**2
$c_1(z,t) = 1 \; for \; z = 0, \; for \; all \; t > 0$	C1(T,Z) - 1 AT Z = 0 C1(T,Z) - 1 AT Z = 0 AT T = 1
$\int_0^1 c(1,t)dt$	INTEGRAL C(1,T) DT

6.1.3 Output Files

The *OCFE* program generating the nonlinear programming problem creates 2 output files, OCFEOUT.DAT and OCFExIF.DAT, where x is either GRG2 or SQP. OCFEOUT.DAT contains the echoed input file, the values of the first and second derivatives by finite elements, by collocation points and by Lagrange polynomials. For example, if there are 2 independent variables and 5 finite elements along each independent variable and 1 internal collocation point in each finite element, then there will be output 180 entries. There will be 18 entries for each 1-dimensional finite element. The following table shows the first 18 entries. They are for the first finite element along the x-axis.

Since in each finite element the number of internal collocation points is 1, these entries will be repeated in the output matrix. OCFEOUT.DAT also contains variable names, their types and the NLP indices of the first NLP variable associated with them. It also contains the equations with their type numbers, the number of NLP variables associated with the state and control variables and the collocation points. The output

Table 6.1 *Derivatives of Lagrange Polynomials*

$$\begin{array}{ccc}
\left.\frac{dl_1}{dx}\right|_{x=0.0} & \left.\frac{dl_2}{dx}\right|_{x=0.0} & \left.\frac{dl_3}{dx}\right|_{x=0.0} \\
\left.\frac{dl_1}{dx}\right|_{x=0.5} & \left.\frac{dl_2}{dx}\right|_{x=0.5} & \left.\frac{dl_3}{dx}\right|_{x=0.5} \\
\left.\frac{dl_1}{dx}\right|_{x=1.0} & \left.\frac{dl_2}{dx}\right|_{x=1.0} & \left.\frac{dl_3}{dx}\right|_{x=1.0} \\
\left.\frac{d^2l_1}{dx^2}\right|_{x=0.0} & \left.\frac{d^2l_2}{dx^2}\right|_{x=0.0} & \left.\frac{d^2l_3}{dx^2}\right|_{x=0.0} \\
\left.\frac{d^2l_1}{dx^2}\right|_{x=0.5} & \left.\frac{d^2l_2}{dx^2}\right|_{x=0.5} & \left.\frac{d^2l_3}{dx^2}\right|_{x=0.5} \\
\left.\frac{d^2l_1}{dx^2}\right|_{x=1.0} & \left.\frac{d^2l_2}{dx^2}\right|_{x=1.0} & \left.\frac{d^2l_3}{dx^2}\right|_{x=1.0}
\end{array} \qquad (1)$$

OCFExIF.DAT, contains the number of NLP variables associated with the state and control variables, the number of NLP variables, NLP constraints, etc. Full description is given in Appendix A.

For GRG2 a file called GRG2INT.FOR contains a subroutine GCOMP for the objective function and constraint equations. For SQP the file SQOINT.FOR contains SQPCONSTRAINTS which has the same purpose.

6.1.4 Processing of the Input File

The following description is closely related to Chapter 4.

First the model is read in. Then a control module called INTERFACE calls the following subprograms.

SETVARRANGE is called to set the NLP range of the state and control variables. Then SETUPAB is called to set to set up the first and second derivatives of the Lagrange polynomials. Following that, if integral terms are present in the objective functional, PROCESSOF is called to transform them into a differential equation. Then if the user wishes to run GRG2, the generation of the subroutine GCOMP begins, otherwise the generation of SQPCONSTRAINTS takes place. Since every program module is written in FORTRAN 77, a program module breaks up lines generated by the program into a number of fixed format FORTRAN program lines, each starting at column 7 and ending at column 72. The generated algebraic equations are first put into a temporary file, then in a second pass from this file GCOMP or SQPCONSTRAINTS is generated.

First constants and algebraic equations are written into the file. Then constraints containing state or control variables dependent on 1 or 2 independent variables are processed. Then differential equations, expressions, boundary conditions are processed, including differential equations and initial or boundary conditions derived from integral terms. PROCESSEQUATION1 processes equations dependent on the first independent variable, PROCESSEQUATION2 processes equations dependent on the second independent variable and PROCESSEQUATION3 processes equations dependent on 2 independent variables. Objective functionals, boundary conditions and differential equations are processed in similar fashion, only the range of the collocation points is set accordingly. PROCESSEQUATION1B, PROCESSEQUATION2B and PROCESSEQUATION3B perform similar tasks, but the equations are collocated along a user-defined rectangular grid, rather than at collocation points and the values of the generated equations will be the residual errors at those grid points. The subprograms PROCESSEQUATION1, PROCESSEQUATION2 and PROCESSEQUATION3 process an equation as follows. For each finite element and for each collocation point within the finite element, they scan the equations and replace variables and their derivatives according to the algorithms described in Chapter 4. An equation is scanned for each collocation point. The various subprograms performing the transformation are described in Appendix B. Then an interpolation program called EVALFUNC.FOR described in Chapter 5 is generated. Then the residual error evaluation program EVALERR.FOR is generated. The OCFEOUT.DAT is written. Then a function called ERROR is generated that the user can add to the objective functional in order to bring down the residual error. Then the data file PLOTPREP.DAT is generated, that contains data used by PLOTPREP.FOR. PLOTPREP.FOR reads PLOTPREP.DAT and FUNCOUT.DAT and generates input data files for each state and control variable in the model for

GNUPLOT. As every other program module, this program is written in FORTRAN 77 and is available on the SUN SPARC 2000. Example of PLOTPREP.DAT is shown in Appendix A.

6.2 Optimization

6.2.1 Control Program

The control program called Dispatcher invokes program modules pertaining to control vector parameterization or to the GRG2 interface module SUBINT or to SQP according to the selection of the user. The modules are fully integrated. The control module reads the input files OCFExIF.DAT generated by OCFE where x is either GRG2 or SQP if a DAOP is solved by the collocation method in conjunction with GRG2 or SQP. In addition, the program reads OPTIMINP.DAT which supplies the parameters required for optimization. This must be written by the user. Details are given in Appendix A.

6.2.2 Control Vector Parameterization

Control vector parameterization (CVP) has been implemented using Hooke-Jeeves method and Broyden-Fletcher-Goldfarb-Shanno method [104] and the 4th order Runge-Kutta method. The parameterization can solve only models containing first-order initial-value ordinary differential equations. The model cannot contain algebraic equations or inequalities. The use of CVP is recommended only when there are more than 3 ordinary differential equations in the model or if the upper limit of the integration is an optimization variable. Also it can be used only for initial value *ODEs*. The user, however, has to set up the objective functional evaluation function F and the derivative evaluation function DX2 in USERSUBS.FOR.

6.2.3 Solution of the Nonlinear Programming Problem

The *NLP* problem is solved either by the package *GRG2*, that uses the generalized reduced gradient method, and was received from *Professor L. S. Lasdon* from the *University of Texas, Austin, Texas*, [91] or by *SQP*, that uses successive quadratic programming, and was received from the *Harwell Physical Laboratory* [92]. For *GRG2*, the problem is stated as

$$\min \quad g_k(x)$$

$$\text{subject to} \quad lb_i \le x_i \le ub_i \quad i=1,2,\dots,n \tag{2}$$

$$lb_{n+i} \le g_i(x) \le ub_{n+i} \quad i=1,2,\dots,m \quad i \ne k$$

For *SQP*, the problem is stated as

$$\min f(x)$$

$$\text{subject to } g_i(x) = 0, \quad i=1,2,\dots,m' \tag{3}$$

$$g_i(x) \ge 0, \quad i=m'+1,m'+2,\dots,m$$

The GRG2 and SQP packages have been chosen based on Schittkowski's study [127-129]. The study showed those packages as being among the most reliable packages currently available. GRG2 and SQP are based on 2 different state of art nonlinear programming methods, the generalized reduced gradient method and successive quadratic programming. Both GRG2 and SQP require initial values. For the SQP package, the calculation of the gradient vector of the objective function and of the Jacobian matrix of the constraints have been implemented. In both cases central differences are used. The arguments of the packages have been tuned. The GRG2 User Guide [91] recommends that variables should be scaled so that a unit change represents a small but significant change in that variable. In order to conform to that strategy, for the GRG2 variables are scaled into the interval [0,100]. The SQP User Guide [92] recommends that the values of the variables and the derivative vectors of the objective function and of the constraints all have magnitudes about unity. For the SQP the variables are scaled into the interval [0,1]. Computer run experience has shown that the *GRG2* and the *SQP* packages are capable of producing good results even without scaling. Because of the *VAX* 32-bit word size to store REAL*4 floating point numbers, double precision was used throughout. *NLP* parameters, like accuracy, stop criterion, iteration limit, etc. are tuned to suit most of the problems, but the user can override them if necessary. The packages stop when the Kuhn-Tucker conditions are satisfied [Chapter 3 (47-48)]. Dump and restart facilities have been implemented. Both packages have been implemented to handle up to 1500 variables and constraints and are used as subroutines integrated into the optimization package. The main difference between GRG2 and SQP is in that GRG2 treats individual bounds on variables separately, while SQP treats them as inequality constraints. Also, in GRG2 every variable and constraint must have a lower and an upper bound, in SQP the constraints are equations and greater than zero inequalities.

6.2.4 NLP Problem Solution

An NLP problem can be solved either by the GRG2 or by the SQP package. The user has to set up either the interface subroutine GCOMP for GRG2 [91] or SQPOF and SQPCONSTRAINTS for SQP [92]. GCOMP must be in the file grg2int.for and SQPOF and SQPCONSTRAINTS in the file sqpint.for. SQPCONSTRAINTS contain the constraints of the model, SQPOF evaluates the objective function. GCOMP contains both the constraints and the objective function.

6.2.5 DAOP Problem Solution

When GRG2 or SQP is used to solve a DAOP, then the program files GRG2INT.FOR and SQPINT.FOR are automatically generated by the program OCFE as already described. Arguments to the packages are generated using data from OPTIMINP.DAT set up by the user and from OCFExIF.DAT written by the OCFE package.

The objective functional is replaced by an objective function and the differential-algebraic constraints are replaced by algebraic equations.

Each state and control variable dependent on 1 or 2 independent variables is replaced by a number of NLP variables called the range. If GRG2 and SQP are used to solve NLP problems derived from a DAOP, then the initial values and bounds given to state and control variables are assigned to the range of NLP variables corresponding to them. If GRG2 is used to solve the NLP problem derived from a DAOP, then variable lower and upper bounds are assigned to the range of NLP variables corresponding to them. If GRG2 is used to solve the NLP problem derived from a DAOP and variable scaling is used, then the scaling bounds will become variable bounds.

6.2.6 Evaluation of Function and Error Values.

For each problem translated by OCFE, the programs EVALERR and EVALFUNC operate on the output files to generate data on a grid chosen by the user.

6.2.7 Conclusion

The programs implement the methods discussed in Chapters 4 and 5 and enable a user to solve modelling and optimization problems within the class described.

Chapter 7
Example Problems

Computer codes were written implementing the methods described in previous chapters. The method presented in this work has been tested on approximately 40 various test problems, of which 8 are included in this thesis. In this chapter, example problems are solved to illustrate the concepts presented in previous chapters and to validate the computer code implementations of these algorithms developed in this work for the solution of differential-algebraic optimization problems containing partial differential equations to be solved over a 2-dimensional rectangular domain. Known solutions to these problems exist, so the correctness of the solutions may be analyzed. Examples 1 and 2 have models containing first order linear partial differential equations. All first order partial differential equations are hyperbolic. Examples 3 and 4 have models containing parabolic differential equations. Examples 5 and 6 have models containing first order linear ordinary differential equations. Example 7 is a *NLP* problem. Example 8 is a partial differential equation with 3 independent variables that has been transformed manually into a *PDE* having 2 independent variables.

7.1 Differential-Algebraic Optimization Problems

Two test problems were solved using the proposed approach described in previous chapters. These problems were taken from the book *Advanced Process Control* by *Ray* [22] and from [96]. The solutions obtained are in the form of piecewise constant functions for the control variables, and polynomial approximations for the state variables.

7.1.1 Example 1 : Optimal Control of a Tubular Plug-Flow Heat Exchanger.

This example is concerned with the determination of near-optimal control policies for a tubular plug-flow heat exchanger with uniform wall flux forcing. (E.S. Parkin and R.L. Zahrednik, [96]). The dynamics of the uniform heat flux exchanger, assuming plug flow in the tube, constant physical properties, and no axial or radial diffusion of heat, can be represented in the following dimensionless form:

$$\frac{\partial x}{\partial t} = -\frac{\partial x}{\partial r} + u \qquad (1)$$

The state variable, $x(r,t)$ represents the deviation of the temperature profile from the final steady-state condition, and $u(t)$ represents the deviation of the wall flux from the final flux profile. The inlet temperature $T(0,t)$ is taken to be constant at $T_n(0)$, so that

$$x(0,t) = 0 \tag{2}$$

Initially the exchanger is assumed to be in a steady state corresponding to a steady state control of unity. Thus the differential equation can be integrated analytically to obtain the initial condition

$$x(0,r) = r \tag{3}$$

The exchanger is controllable and the optimal policy is chosen to minimize the following quadratic performance index:

$$p(u) = \frac{1}{2}\int_0^{t_f}\int_0^1 (\mu x^2 + u^2) dr dt \tag{4}$$

where μ is a non-negative scalar weight. In [96], the problem was solved as follows: *Pontryagin's* maximum principle was used to derive the necessary condition for optimality leading to the derivation of the *Hamiltonian* and of the adjoint equations.

Then the state and adjoint variables were approximated by linear combination of spatially dependent trial functions and time-dependent coefficients. Application of the method of weighted residuals reduced the problem to one involving ordinary differential equations in terms of the linear combination coefficients. Then the system of *ODEs* were solved. Here double orthogonal collocation on finite elements was applied to transform the model into an *NLP* problem. The temperature profile closely approximates the solution given in [96]. The problem parameters and problem statistics are shown in Tables 7.1-7.3. The knot points in t-interval are 0.2, 0.4, 0.6, 0.8. The knot points in r-interval are 0.2, 0.4, 0.6, 0.8. The integration domain was [0,1]x[0,1]. The residual error of the discretized differential equation was evaluated at (0.125,0.125), (0.125,0.25),...,(1,1). There were 25 2-dimensional finite elements. The discretized differential equations were evaluated both at internal collocation points and at the boundaries of the finite elements and at those boundary points of the integration domain where there are no initial/boundary conditions. The discretized differential equation in element (k,l) k = 1,2,...5 l = 1,2,...,5 and at collocation points i = 1,2,3 j = 1,2,3 is

$$\frac{1}{\Delta t_k}\sum_{n=1}^{3} x_{nj}^{kl}\,\frac{dl_n(t_i')}{dt'} + \frac{1}{\Delta r_l}\sum_{n=1}^{3} x_{in}^{kl}\frac{dl_n(r_j')}{dr'} - u(k) \qquad \begin{array}{l} k = 1,2,...,5 \; l = 1,2,...,5 \\ i = 2,3 \; j = 2,3 \end{array} \tag{5}$$

The differential equations were not collocated where initial/boundary conditions were given.

The discretized initial/boundary conditions are

$$\begin{array}{ll} x_{il}^{kl} = 0 & \begin{array}{l} k = 1 \; i = 1,2,3 \\ k = 2,...,5 \; i = 2,3 \end{array} \\ x_{1j}^{1l} = r & l = 1,2,...,5 \; j = 2,3 \end{array} \tag{6}$$

The integral part of the objective functional has been transformed into a differential equation

$$\frac{\partial^2 I}{\partial r \partial t} = \frac{\mu x^2 + u^2}{2} \qquad I(0,0) = 0 \tag{7}$$

It was discretized as follows
at k = 1,2,...,5 l = 1,2,...,5 i = 2,3 j = 2,3

$$\frac{1}{\Delta t_k \Delta r_l}\sum_{n=1}^{3}\sum_{m=1}^{3} \frac{dl_n(t_i')}{dt'}\,\frac{dl_m(r_j')}{dr'}\,I_{ij}^{kl} - \left[\frac{\mu(x_{ij}^{kl})^2 + (u^k)^2}{2}\right] = 0 \qquad I_{11}^{11} = 0 \tag{8}$$

In order to reduce the residual error, a weighted error term was added to the objective functional. The error term is the least-square residual error over the integration domain. The objective functional is

$$I_{33}^{55} + 0.1ERROR(x) \tag{9}$$

Table 7.1 shows the starting values and bounds for the state and control variables.

Table 7.1 Starting Values

Function/ Variable	lower bound	upper bound	starting point
x	0	1	0
u	-1	1	0
I	0	1	0

The state variables x and I are unbounded, the bounds are only used for scaling. The number of unknowns is illustrated for x in Figure 7.1.

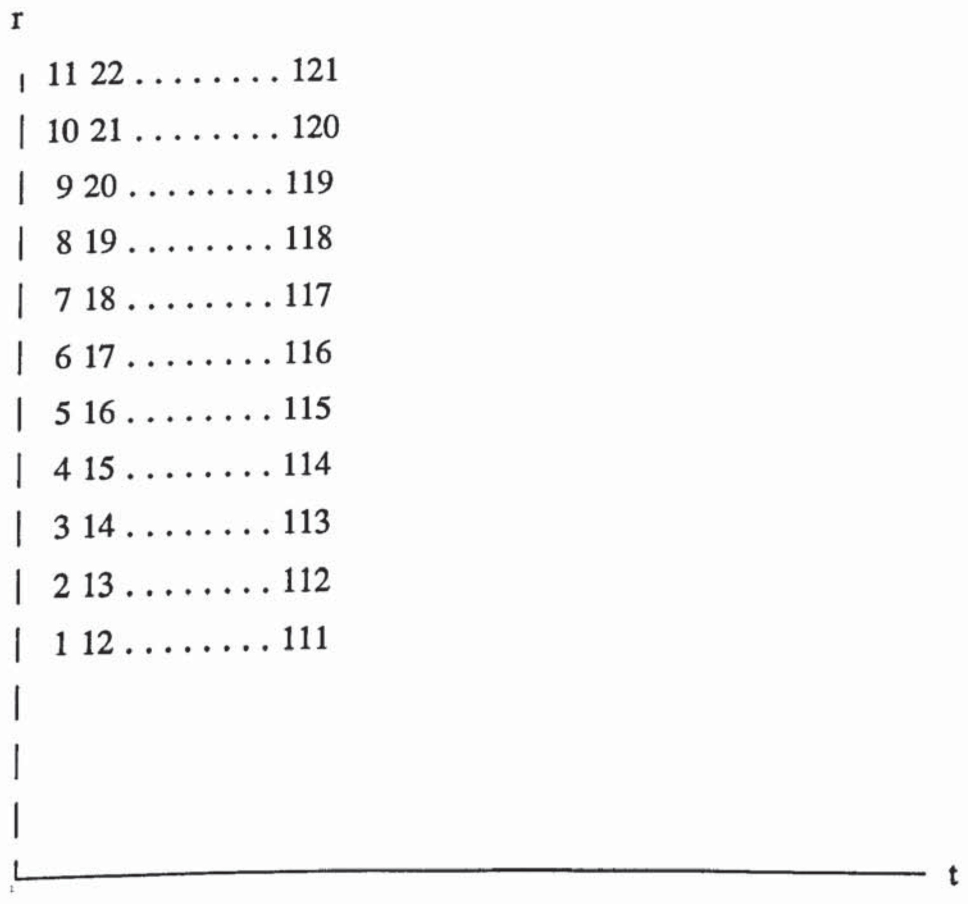

Fig. 7.1 NLP variables for x

Table 7.2 shows the number of *NLP* variables derived from the state and control variables.

Table 7.2 Relationship between Continuous and *NLP* Variables

State or control function	NLP dimension
x	121
u	5
I	121

The problem was run twice, once using *GRG2* and once using *VF13AD* on the *VAX*. Table 7.3 shows the parameters of the collocation method applied and the *NLP* solution statistics.

Table 7.3 Collocation Parameters and *NLP* Statistics

	GRG2	SQP
N	5	5
M	5	5
N_t	5	5
N_r	5	5
n	247	247
m	243	243
nfun	114	-
niter	11	11
acc	0.000001	0.0001
CPU	00:18:31	00:29:31

	GRG2	SQP
err	0.08	0.08
aveerr	0.01	0.01
OF	0.079	0.073

where		
N	=	number of internal collocation points in *t*-interval
M	=	number of internal collocation points in *r*-interval
N_t	=	number of finite elements in *t*-interval
N_r	=	number of finite elements in *r*-interval
n	=	number of *NLP* variables
m	=	number of *NLP* constraints
nfun	=	number of function evaluations
niter	=	number of *NLP* iterations
acc	=	accuracy of *NLP* solution
CPU	=	*CPU* time (hh:mm:ss)
err	=	the maximum point-wise residual error
aveerr	=	the average point-wise residual error

$$OF = \frac{1}{2}\int_0^1\int_0^1 (\mu x^2 + u^2)drdt + 0.1ERROR(x)$$

Table 7.4 compares the optimal values of the temperature at r = 1

Table 7.4 Comparison of Temperature Values

Time	GRG2	SQP	[96]
0	1	1	1
0.2	0.766	0.761	~0.76
0.4	0.549	0.543	~0.56

Time	GRG2	SQP	[96]
0.6	0.332	0.327	~0.34
0.8	0.134	0.130	~0.15
1	0.0053	0.0024	~0

Figure 7.2 shows the optimal values of x (GRG2 run).

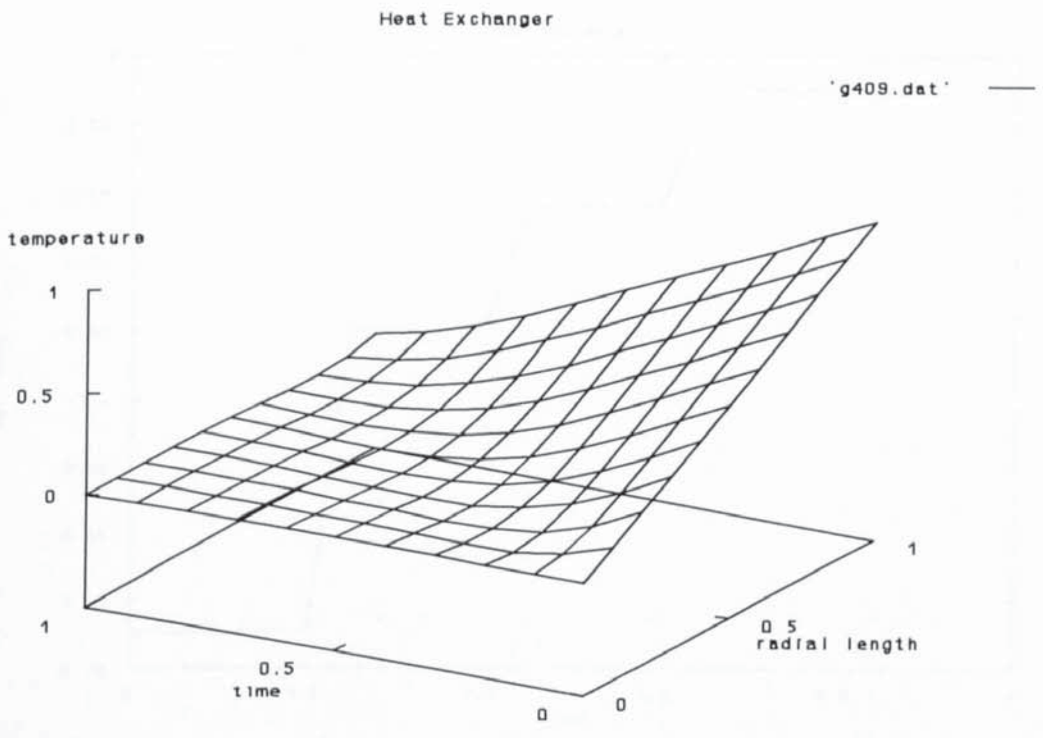

Fig. 7.2 temperature = f(time,radial length)

Figure 7.3 shows the optimal values of u (GRG2 run).

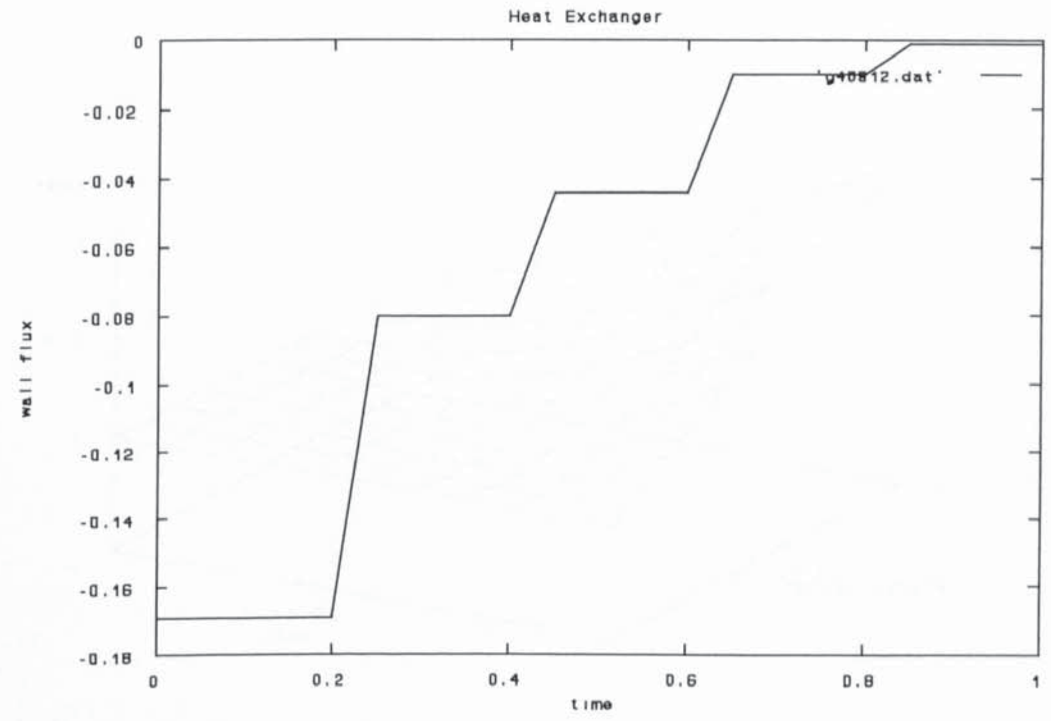

Fig. 7.3 wall flux = f(time)

Figure 7.4 shows the optimal values of *x* (*SQP* run).

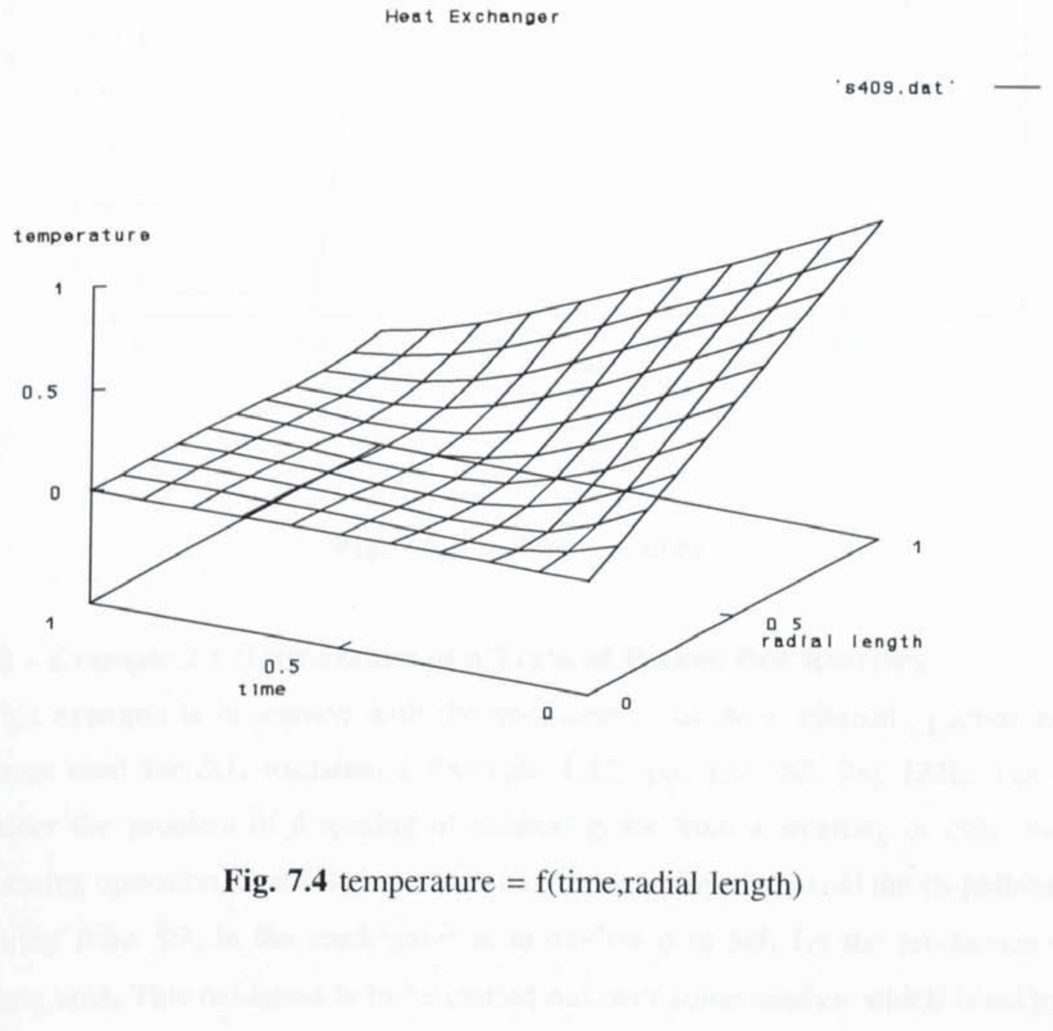

Fig. 7.4 temperature = f(time,radial length)

Figure 7.5 shows the optimal values of u (*SQP* run).

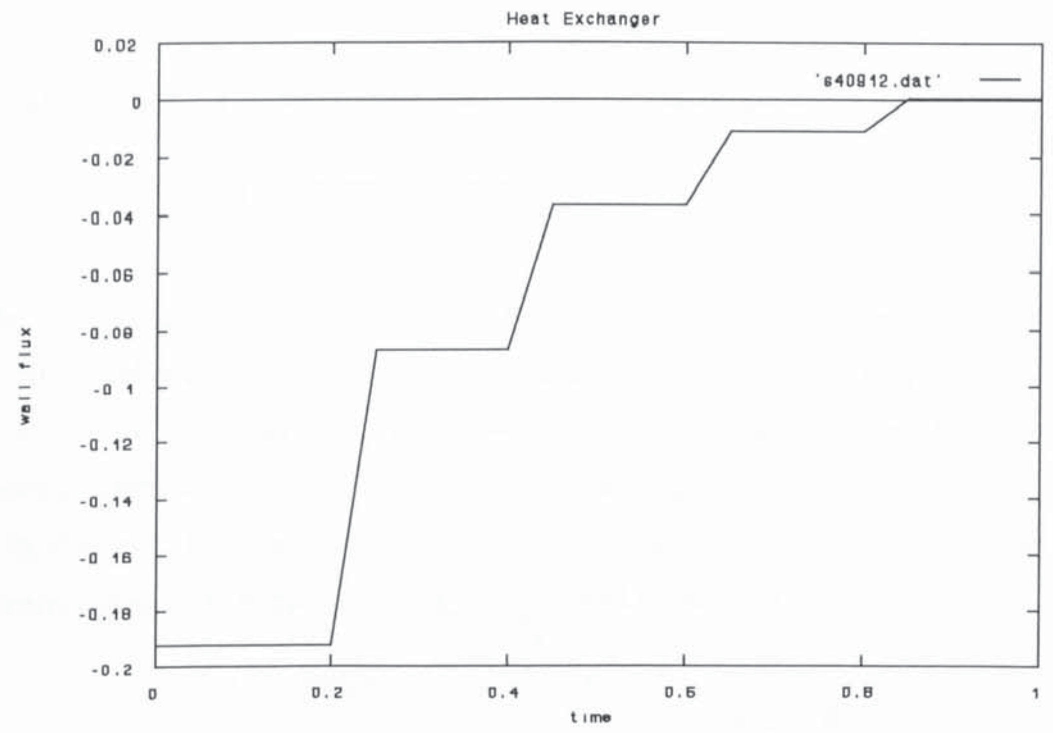

Fig. 7.5 wall flux = f(time)

7.1.2 - Example 2 : Optimization of a Train of Packed Bed Reactors

This example is concerned with the optimization of three adiabatic packed bed reactors used for SO_2 oxidation (Example 4.3.2, pp. 177-182, Ray [22]). Let us consider the problem of disposing of exhaust gases from a smelting or other ore-processing operation. One solution which has been employed to avoid the air pollution resulting from SO_2 in the stack gases is to oxidize it to SO_3 for the production of sulfuric acid. This oxidation is to be carried out over some catalyst which is subject to deactivation with time. Because the reaction is exothermic and is assumed to be reversible, a number of adiabatic stages are employed with interstage cooling. It is assumed that species A is the reactant and B is the oxidation product. Thus the first-order reaction

$$A \underset{k2}{\overset{k1}{\rightleftarrows}} B$$

is to be carried out in three adiabatic bed packed reactors.

$$\begin{array}{cccccc} \text{-------|} & \text{-------|} & \text{-------|} & \text{-------|} & \text{-------|} & \text{-------|} \\ \alpha_1^+ & \alpha_2^- & \alpha_2^+ & \alpha_3^- & \alpha_3^+ & \alpha_4^- \end{array}$$

The first reactor begins at α_1^+, ends at α_2^-, the second reactor begins at α_2^+, ends at α_3^-, the third reactor begins at α_3^+ and ends at α_4^-. For modelling purposes $\alpha_1 = \alpha_{1+,}$ $\alpha_2 = \alpha_2^- = \alpha_2^+$, $\alpha_3 = \alpha_3^- = \alpha_3^+$ and $\alpha_4 = \alpha_4^-$, neglecting place occupied by interstage cooling, where $\alpha_1 = 0$, $\alpha_2 = \frac{1}{3}$, $\alpha_3 = \frac{2}{3}$, and $\alpha_4 = 1$.

In the k-th bed the dimensionless reaction temperature τ_k is related to the dimensionless inlet temperature $u_k = \frac{RT_k}{E_k}$ and conversion by

$$\tau_k(z,t) = u_k(t) + J[x_1(z,t) - x_1(\alpha_k,t)] \quad \begin{array}{l} 0 \le t \le 1 \\ 0 \le z \le 1 \\ k = 1,2,3 \end{array} \tag{10}$$

where J is constant and $x_1(z,t)$ is the conversion.

The equation describing the conversion is

$$\frac{\partial x_1(z,t)}{\partial z} = x_2(z,t)[\beta_1 e^{-1/\tau_k}(1-x_1(z,t)) - \beta_2 e^{-p_1/\tau_k} x_1(z,t)] \quad \begin{array}{l} 0 \le t \le 1 \\ 0 \le z \le 1 \\ k = 1,2,3 \end{array} \tag{11}$$

$x_1(0,t) = 0$

The decay of catalyst is given by

$$\frac{\partial x_2(z,t)}{\partial t} = -\rho[x_2(z,t)^2 e^{-1/(p\tau_k)}] \quad \begin{array}{l} 0 \le t \le 1 \\ 0 \le z \le 1 \\ k = 1,2,3 \end{array} \tag{12}$$

$x_2(z,0) = 1$

We wish to maximize the accumulative conversion of A over a catalyst lifetime

$$I = \int_0^1 x_1(1,t)dt \tag{13}$$

We wish to control the interstage coolers (i.e. the inlet temperatures T_1 , T_2, T_3) so as to maximize the conversion of A over the catalyst lifetime.

The set of parameters are

$\beta_1 = 5.244\text{x}10^5$ $\beta_2 = 2.28\text{x}10^9$ $\rho = 1300$ $u_{k*} = 0.07$

$u_k^* = 0.08$ $p = 1.648$ $p_1 = 1.666$ $J = 0.005$

In Ray [22], this problem was solved by control vector iteration. Pontryagin's maximum principle was used to develop the derivation of necessary conditions for optimality leading to the derivation of the Hamiltonian and the adjoint equations. The computational procedure was as follows:

1. Guess $u_k(t)$, $0 \leq t \leq 1$, k=1,2,3.
2. Solve the state equations (the 2 partial difference equations) forward in z,t using the method of characteristics (or finite differences); compute the objective functional I.
3. Solve the adjoint equations backward in z,t.
4. Correct the control function $u_k(t)$ by

$$u_k^{new}(t) = u_k^{old}(t) + \varepsilon_0 \int_{\alpha_k}^{\alpha_{k+1}} \left(\frac{\partial H}{\partial u_k}\right) dz \tag{14}$$

where k = 1,2,3, $\alpha_1 = 0$, $\alpha_2 = \frac{1}{3}$, $\alpha_3 = \frac{2}{3}$, $\alpha_4 = 1$, and ε_0 is determined by a one-dimensional search.

5. Return to step 2 and iterate.

The optimal control was found in about 5 minutes of computing time (IBM 360/75).

In this work orthogonal collocation on finite elements was applied to transform the model into an NLP problem and run it using OCFE on VAX. The conversion and the control closely approximates the solution given in Ray [22]. The problem parameters and problem statistics are shown in Figure 7.6 and in Tables 7.5 - 7.7

The *SQP* model differs from the *GRG2* model, in that it contains 2 inequality constraints

$$\begin{aligned} 0.08 - u(z,t) &\geq 0 \\ u(z,t) - 0.07 &\geq 0 \end{aligned} \tag{15}$$

It is because GRG2 treats individual bounds separately while SQP treats individual bounds as inequality constraints.

The integration domain was [0,1]x[0,1].

The residual error of the discretized differential equations was evaluated at (z,t) = (0,0), (0,0.25), ..., (1,1).

The knot points in z-interval are 1/3, 2/3, corresponding to the 3 stages.

The knot points in t-interval are 0.2, 0.4, 0.6, 0.8.

There are 15 2-dimensional finite elements. The discretized differential equations were evaluated both at internal collocation points and at boundaries of finite elements and at the boundaries of integration domain, where there are no initial/boundary conditions. The discretized differential equations are in element (k,l) $k = 1,...1,3$ $l = 1,...,5$ and at collocation points $i = 1,..., 3$ $j = 1,..., 3$

$$\frac{1}{\Delta z_k} \sum_{n=1}^{3} x_{1nj}^{kl} \frac{dl_n(z_i')}{dz'} - x_{2ij}^{kl} \left[\beta_1 e^{-1/\tau_{ij}^{kl}}(1 - x_{1ij}^{kl}) - \beta_2 e^{-p_1/\tau_{ij}^{kl}} x_{1ij}^{kl} \right] \quad \begin{matrix} k = 1,2,3 \ l = 1 \\ i = 2,3 \ j = 1,2,3 \\ k = 1,2,3 \ l = 2,...,5 \\ i = 2,3 \ j = 2,3 \end{matrix} \tag{16}$$

$$\frac{1}{\Delta t_l} \sum_{n=1}^{3} x_{2in}^{kl} \frac{dl_n(t_j')}{dt'} + \rho (x_{2ij}^{kl})^2 e^{-1/(\rho \tau_{ij}^{kl})} \quad \begin{matrix} k=2,3 \ l=1,...,5 \\ i=2,3 \ j=2,3 \\ k=1 \ l=1,...,5 \ i=1,2,3 \ j=2,3 \end{matrix} \tag{17}$$

The differential equations are not collocated where initial/boundary conditions are given. The discretized initial/boundary conditions for x_1 and x_2 are

$$
\begin{array}{lll}
x_{11j}^{1l} = 0 & l=1 & j = 1,2,3 \\
 & l = 2,\ldots,5 & j = 2,3 \\
x_{2il}^{kl} = 1 & k = 2,3 & i = 2,3 \\
 & k = 1 & i = 1,2,3
\end{array}
\tag{18}
$$

The integral part of the objective functional has been transformed into the differential equation

$$
\begin{gathered}
\frac{dI}{dt} - x_1(1,t) = 0 \\
I(1,0) = 0
\end{gathered}
\tag{19}
$$

It was discretized as follows:

At k =3 l=1,...,5 i = 3 j = 2,3

$$
\begin{gathered}
\frac{1}{\Delta t_l} \sum_{n=1}^{3} I_{3n}^{3l} \frac{dl_n(t_j')}{dt'} - x_{13j}^{3l} = 0 \\
I_{31}^{31} = 0
\end{gathered}
\tag{20}
$$

In order to reduce the residual error, a weighted error term was added to the objective functional. The error term is the least-square residual error over the integration domain. Then the objective functional is

$$
I_{33}^{35} - 0.1ERROR(x) \tag{21}
$$

Table 7.5 shows the starting values and bounds for the state and control variables

Table 7.5 Starting Values

Functions/ Variables	lower bound	upper bound	starting point
x_1	0	1	0
x_2	0	1	1

Functions/ Variables	lower bound	upper bound	starting point
u	0.07	0.08	0.075
I	0	1	0

where x_1	=	the dimensionless conversion (state variable)
x_2	=	the catalyst activity (state variable)
u	=	the dimensionless inlet temperature (control variable)
I	=	variable derived from the integral term

The state variables x_1, x_2 and I are unbounded, the bounds only used for scaling. τ_1 and τ_2 are parameters. *z* and *t* are independent variables.

The numbering of unknowns is illustrated for x_1 in Figure 7.6.

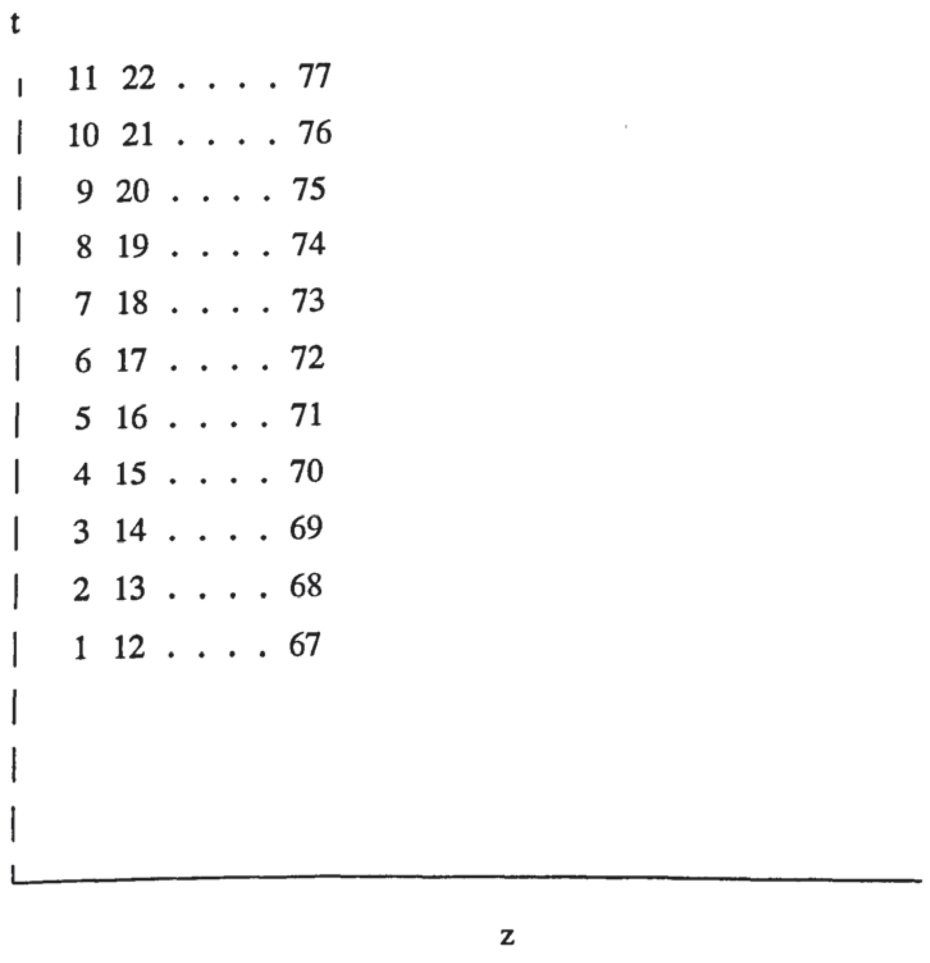

Fig. 7.6 NLP variables for x_1

The numbers represent *NLP* variable numbers at grid points.

Table 7.6 shows the number of *NLP* variables derived from the state and control variables.

Table 7.6 Relationship between Continuous and *NLP* Variables

State or control function	NLP dimension
x1	77
x2	77
u	15
I	11

The problem was run twice, once using *GRG2* and once using *SQP* on the *VAX*.

Table 7.7 shows the parameters of the collocation method applied and the *NLP* solution statistics.

Table 7.7 Parameters and NLP Statistics

	GRG2	SQP
N	3	3
M	5	5
N_z	3	3
N_t	5	5
n	180	180
m	166	196
nfun	733	-
niter	119	20

	GRG2	SQP
acc	0.000001	0.0001
CPU	00:37:26	00:22:45
error	0.05	0.06
aveerr	0.008	0.01
conv	0.488	0.491

where			
	N	=	number of internal collocation points in *z*-interval
	M	=	number of internal collocation points in *t*-interval
	N_z	=	number of finite elements in *z*-interval
	N_t	=	number of finite elements in *t*-interval
	n	=	number of *NLP* variables
	m	=	number of *NLP* constraints
	nfun	=	number of function evaluations
	niter	=	number of *NLP* iterations
	acc	=	accuracy of *NLP* solution
	CPU	=	*CPU* time (hh:mm:ss)
	err	=	the maximum point-wise residual error
	aveerr	=	the average point-wise residual error
	conv	=	cumulative conversion

Table 7.8 compares the optimal values of the conversion at t = 1

Table 7.8 Comparison of Conversion Values

z	GRG2	SQP	[22]

k=1
z=1/3
	0.28	0.28	~0.28

z	GRG2	SQP	[22]

k=2
z=2/3
	0.40	0.40	~0.40

k=3
z=1
	0.46	0.46	~0.465

OF	0.488	0.491	0.50

Figure 7.7 shows the optimal solution x_1 (GRG2 run).

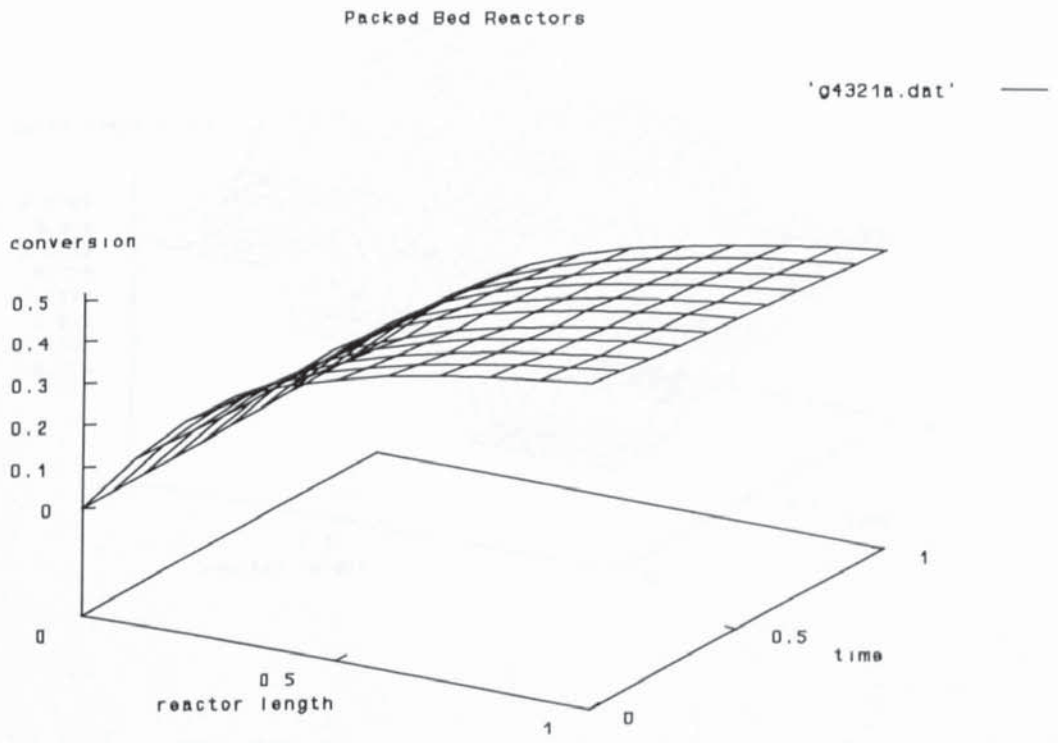

Fig. 7.7 conversion = f(reactor length,time)

Figure 7.8 shows the optimal values of *u* (GRG2 run).

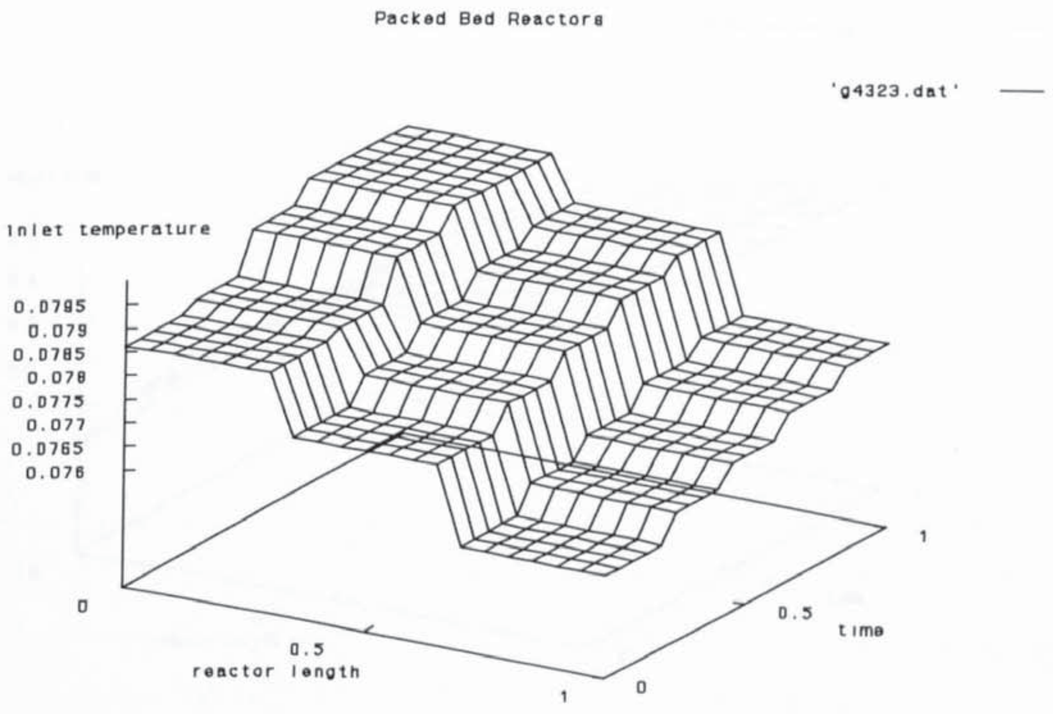

Fig. 7.8 inlet temperature = f(reactor length,time)

Figure 7.9 shows the optimal values of x_1 (SQP run).

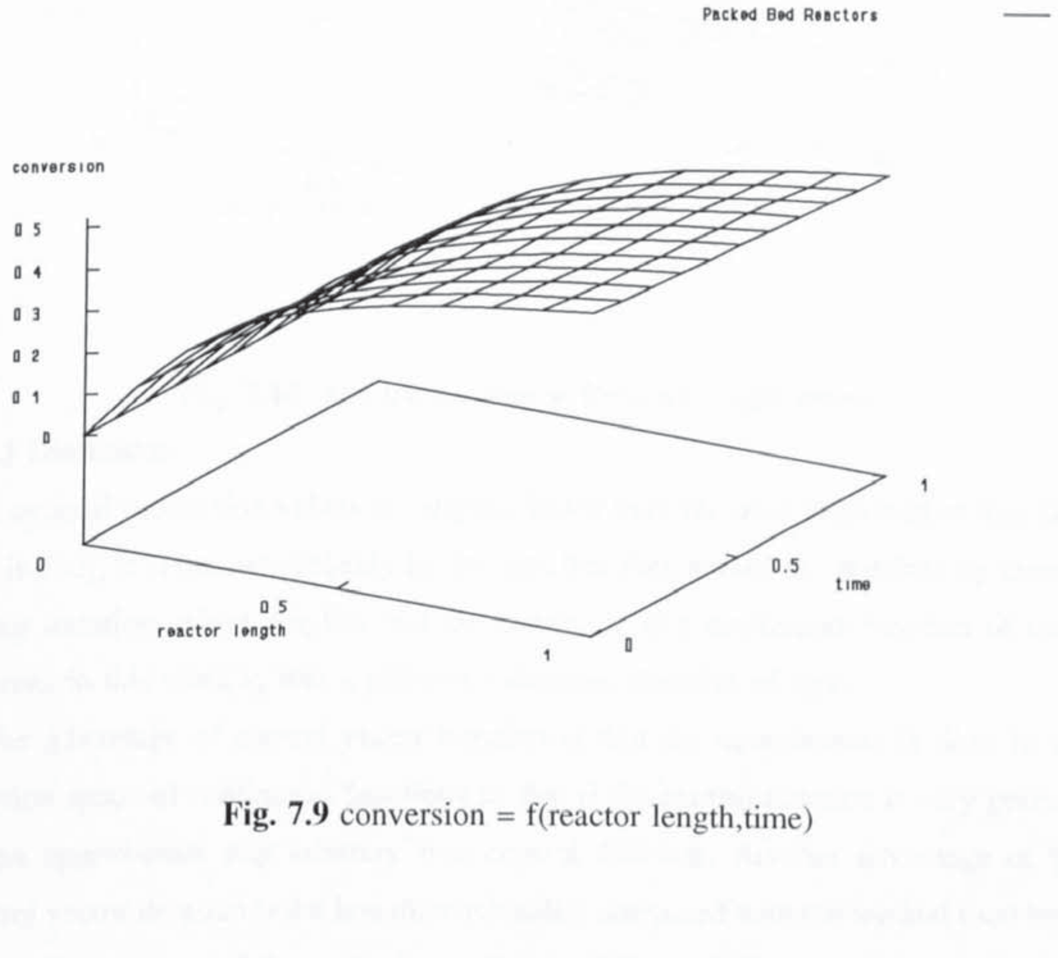

Fig. 7.9 conversion = f(reactor length,time)

Figure 7.10 shows the optimal values of u (SQP run).

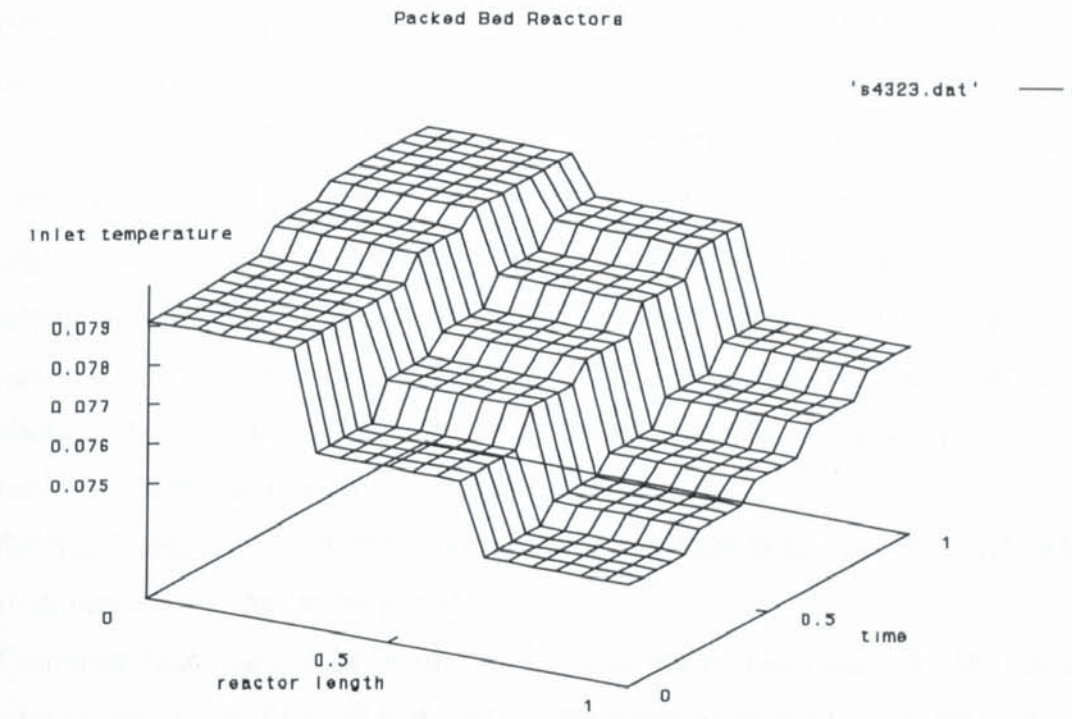

Fig. 7.10 inlet temperature = f(reactor length,time)

7.1.3 Discussion

The optimal conversion values are slightly lower than the ones published in Ray [22] and in [30], it is caused probably by the fact that they solved the problem by control vector iteration, which implies that the control u_k is a continuous function of time, whereas in this work u_k was a piecewise constant function of time.

The advantage of control vector iteration is that the optimization is done in the function space of continuous functions so that if the control function is very general, it can approximate any arbitrary true control function. Another advantage of the control vector iteration is the low dimensionality compared with the method used here.

The disadvantage of the method described in [22] and [96] is that it requires setting up manually the *Hamiltonian* and the adjoint equations, coding manually the differential equations and the adjoint equations and probably requires human effort along the various stages leading up to the solution. So it requires more time to solve the problem from start to finish. Another disadvantage of using the necessary conditions of optimality is that only simple control bounds can be handled.

The advantage of using double orthogonal collocation on finite elements over the method described in [22] and [96], that the transformation of the problem requires only setting up the model according to the syntax of the conversion package, from that point the solution is obtained automatically. In comparison with the method of using the equations derived from the necessary conditions for optimality, the advantage of using double orthogonal collocation on finite elements to discretize the differential-algebraic optimization problem into a *NLP* problem, that it can solve problems containing also algebraic constraints and constraints containing state and control variables. This advantage did not show up in this examples, because there were no constraints except for bounds on the control variable.

The disadvantage of the method described in this work is that an *NLP* problem of high dimensionality has to be solved.

Computer times cannot be compared, because the problem was run on different machines, and it was observed that the *CPU* time depend on the load on the computer system.

7.2 Dynamic Simulation Problems

7.2.1 Example 3 : Diffusion and Chemical Reaction in a Tubular Reactor with Non-Newtonian Laminar Flow

The dimensionless form of the steady-state continuity equation with first-order homogeneous chemical reaction and negligible axial diffusion of reactant is given by the differential equation

$$\kappa \left(\frac{\partial^2 C}{\partial \xi^2} + \frac{1}{\xi} \frac{\partial C}{\partial \xi} \right) - \beta \, [1 - \phi(\xi)] \, \frac{\partial C}{\partial \zeta} - C = 0$$

$$\textit{where } s = 0.4, \ \beta = \frac{3s+1}{s+1}, \ \kappa = 0.01 \textit{ and} \qquad (22)$$

$$\phi(\xi) = \xi^{\frac{s+1}{s}}$$

subject to the boundary conditions

$$C = 1 \ at \ \zeta = 0, \ 0 \le \xi \le 1$$

$$\frac{\partial C}{\partial \xi} = 0 \ at \ \xi = 0, \ \zeta > 0 \qquad (23)$$

$$\frac{\partial C}{\partial \xi} = 0 \ at \ \xi = 1, \ \zeta > 0$$

In [97], the problem was solved as follows:

By separation of the variables an analytical solution was found, containing a function that should satisfy an ordinary differential equation. *Galerkin's* method was used to express the eigenfunctions of the *ODE* in a finite set of trial functions. *Galerkin's* method reduced the problem of solving an ordinary differential equation to one of solving a matrix equation. It was solved to yield eigenvectors and eigenvalues. From the eigenvectors the eigenfunctions were determined. The solution of the *ODE* was then obtained as a linear combination of the eigenfunctions. Here double orthogonal collocation on finite elements was applied to transform the *PDE* into a system of algebraic equations. The solution obtained closely approximates the solution given by R. V. Homsy and R. D. Strohman in [97]. But the solution function $c(\zeta,\xi)$ has "humps" in the ζ (axial) direction. The derivatives in the ζ-direction are not monotonic, although *c(ζ,ξ)* is monotonic in both co-ordinate directions. In [136] there is reference to "hump" in measured radial temperature profiles. Also in order to reduce residual error, in this work the derivatives are not continuous at finite element boundaries. It probably can be ruled out, that the model is wrong. I also checked repeatedly every step leading to the creation of the surface plot starting with the input model to *OCFE*. The problem parameters and problem statistics are shown in Tables 7.9-7.11.

The knot points in ζ-interval are 1., 2., 3., 4., 5., 6. The knot points in ξ-interval are 0.1, 0.3, 0.6. The integration domain was [0,7]x[0,1]. The residual error of the discretized differential equation was evaluated at (0.9,0.125), (0.9,0.25),...,(6.3,0.875). There were 28 2-dimensional finite elements. The discretized differential equation was evaluated both at internal collocation points and at boundaries of finite elements and at those initial/boundary points of the integration domain, where there are no initial/boundary conditions. The discretized differential equation in element *(k,l)* k=1,2,...,7 l = 1,2,3,4 and at collocation points i = 1,2,3 j = 1,2,3 is

$$\begin{gathered}
\kappa \frac{1}{\Delta\xi_l^2} \sum_{n=1}^{3} C_{in}^{kl} \frac{d^2 l_n(\xi_j')}{d\xi'^2} + \frac{1}{\xi} \frac{1}{\Delta\xi_l} \sum_{n=1}^{3} C_{in}^{kl} \frac{dl_n(\xi_j')}{d\xi'} \\
- \beta[1 - \phi(\xi')] \frac{1}{\Delta\zeta_k} \sum_{n=1}^{3} C_{nj}^{kl} \frac{dl_n(\zeta_i')}{d\zeta'} - C_{ij}^{kl} \qquad (24) \\
k = 1,2,\ldots,7 \; l = 1,2,3 \; i = 2,3 \; j = 2,3 \\
k = 1,2,\ldots,7 \; l = 4 \; i = 2,3 \; j = 2
\end{gathered}$$

The differential equation was not collocated where initial/boundary conditions were given. The discretized initial/boundary conditions are

$$\begin{gathered}
c_{ij}^{kl} = 1 \; k = 1 \; l = 1 \; i = 1 \; j = 1,2,3 \\
c_{ij}^{kl} = 1 \; k = 1 \; l = 2,3,4 \; i = 1 \; j = 2,3 \qquad (25) \\
\frac{1}{\Delta\xi_l} \sum_{n=1}^{3} c_{in}^{kl} \frac{dl_n(\xi_j')}{d\xi'} \quad \begin{cases} k = 1,2,\ldots,7 \; l = 1 \; i = 2,3 \; j = 1 \\ k = 1,2,\ldots,7 \; l = 4 \; i = 2,3 \; j = 3 \end{cases}
\end{gathered}$$

Table 7.9 shows the starting values and bounds for the state variable

Table 7.9 Starting Values

Function/ variable	lower bound	upper bound	starting point
C	0	1	1

The state variable *C* is unbounded, the bounds are only used for scaling. The numbering of *C* is shown in Figure 7.11

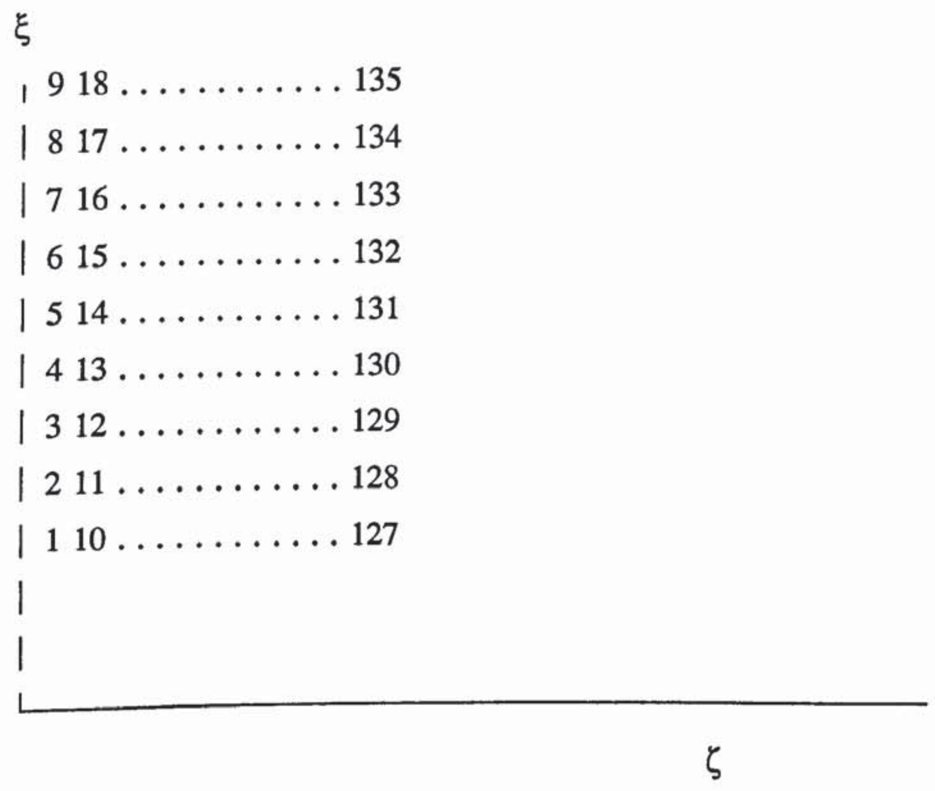

Fig. 7.11 NLP Variables for C

Table 7.10 shows the number of *NLP* variables derived from the state variable *C*.

Table 7.10 Relationship between Continuous and *NLP* Variables

Function/variable	NLP dimension
C	135

The problem was run twice, once using *GRG2* and once using *VF13AD* on the *VAX*. Table 7.11 shows the parameters of the collocation method applied and the *NLP* solution methods.

Table 7.11 Collocation Parameters and *NLP* Statistics

	GRG2	SQP
N	7	7
M	4	4
N_ζ	7	7
N_ξ	4	4
n	135	135
m	136	136
nfun	312	-
niter	90	1
acc	0.000001	0.0001
CPU	00:10:28	00:00:51
err	0.06	0.06
aveerr	0.006	0.006

where	N	=	number of internal collocation points in *z*-interval
	M	=	number of internal collocation points in *x*-interval
	N_ζ	=	number of finite elements in ζ-interval
	N_ξ	=	number of finite elements in ξ-interval
	n	=	number of *NLP* variables
	m	=	number of *NLP* constraints

nfun	=	number of function evaluations
niter	=	number of *NLP* iterations
acc	=	accuracy of the *NLP* solution
CPU	=	*CPU* time (hh:mm:ss)
err	=	the maximum point-wise residual error
aveerr	=	the average point-wise residual error

Table 7.12 compares the optimal values of concentrations for $\zeta = 5$ and $\kappa = 0.01$ and $s = 0.4$.

Table 7.12 Comparison of Concentration Values

radius	GRG2	SQP	[97]
0	0.04	0.04	~0.039
0.2	0.04	0.04	~0.038
0.4	0.03	0.03	~0.03
0.6	0.01	0.01	~0.015
0.8	0.006	0.006	~0.007
1	0.002	0.002	~0.003

Figure 7.12 shows the values of *C* (*GRG2* run).

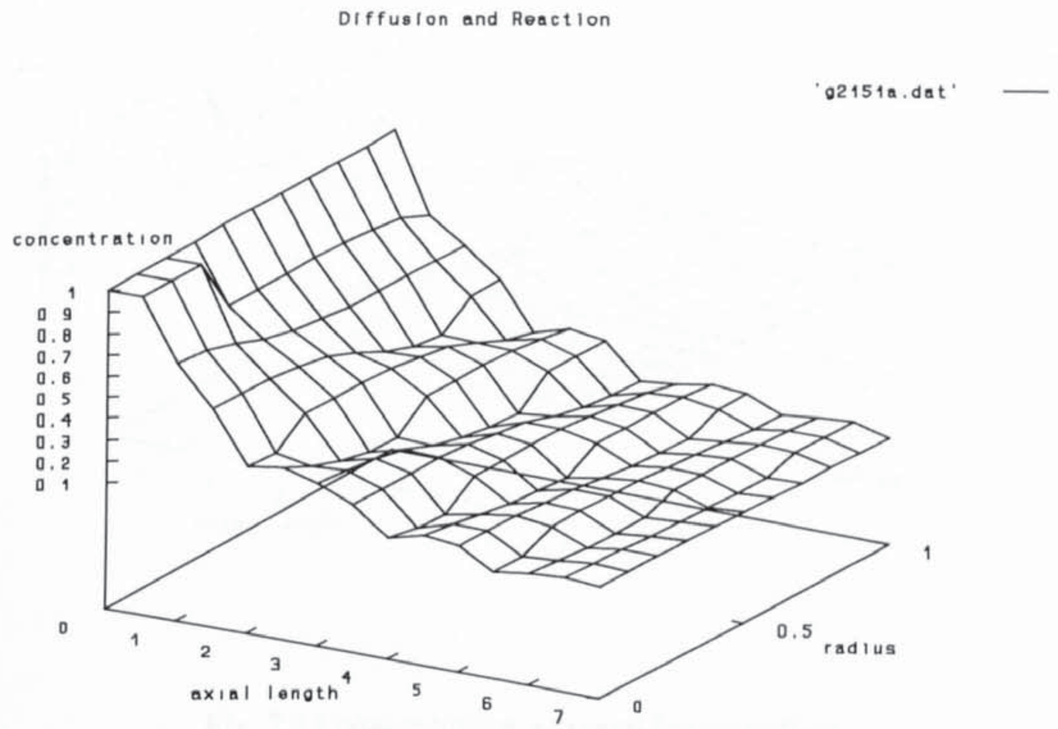

Fig. 7.12 concentration = f(axial length,radius)

Figure 7.13 shows the values of *C* (*SQP* run).

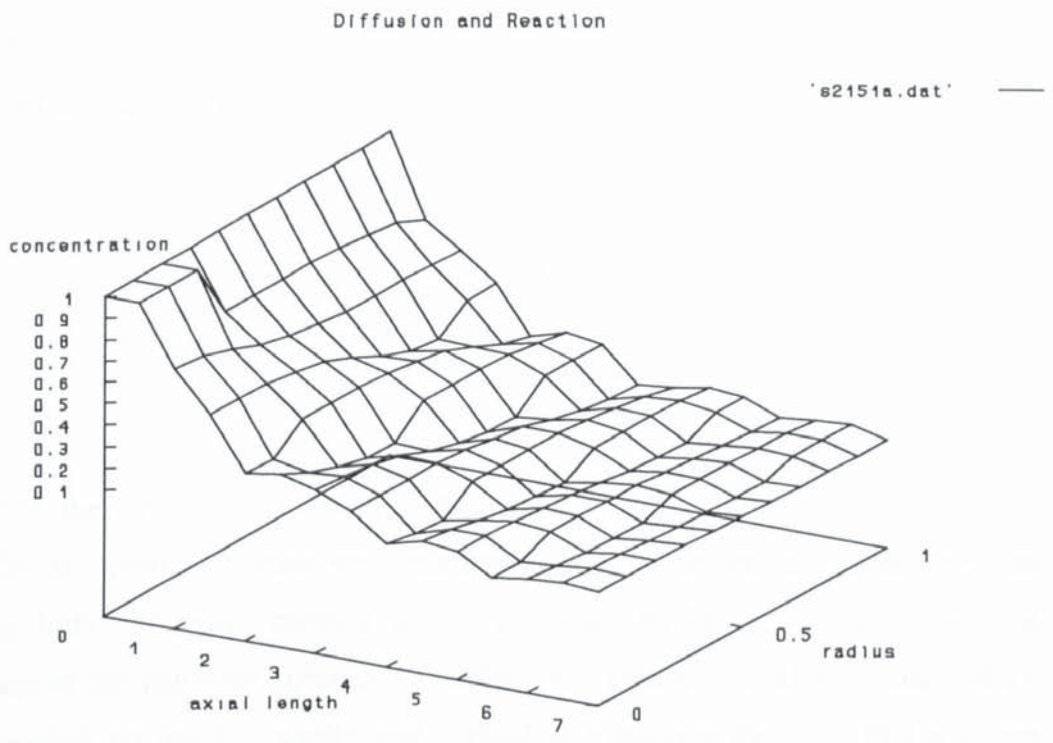

Fig. 7.13 concentration = f(axial length,radius)

The advantage of using double collocation over the method described in [97], that the transformation of the problem requires only setting up the model according to the syntax of the conversion package, from that point the solution is obtained completely automatically. Using the method described in [97] requires more manual procedures and requires more human effort along the various stages leading to the solution. So it requires more time to solve a problem from start to finish. The advantage of the method described in [97] over the method employed in this work that one does not have to solve a *NLP* problem of high dimensionality.

7.2.2 Example 4 : Diffusion and Reaction in Viscous-flow Tubular Reactor.

A differential volume of fluid within a tubular flow reactor will be considered. Steady-state, axial symmetry, and flow in the axial direction only are assumed. An irreversible first-order chemical reaction in dimensionless form becomes:

$$-(1 - U^2)\frac{\partial C}{\partial \lambda} + \alpha\left(\frac{\partial^2 C}{\partial U^2} + \frac{1}{U}\frac{\partial C}{\partial U}\right) - C = 0 \tag{26}$$

where $\alpha = 0.1$

subject to the boundary conditions

$$\begin{aligned} &C = 1 \; at \; \lambda = 0 \\ &\frac{\partial C}{\partial U} = 0 \; at \; U = 0 \\ &\frac{\partial C}{\partial U} = 0 \; at \; U = 1 \end{aligned} \tag{27}$$

In [98], the problem was solved as follows:

The differential equation was transformed into corresponding difference equation using finite difference method. C was expanded in *Taylor's* series. The resulting system of simultaneous difference equations was then solved. Here double orthogonal collocation on finite elements was applied to transform the *PDE* into a system of algebraic equations. The solution obtained shows some discrepancies as compared to the solution given by F. A. Cleland and R. H. Wilhelm in [98]. In [98] there is no boundary condition for $U = 0$. But the differential equation (26) is parabolic, and a parabolic differential equation requires 1 initial and 2 boundary conditions. In this work, differential equations are collocated at the boundaries of the integration domain, where there are no boundary conditions. But the differential equation (26) is not defined at $U = 0$, because there is division by U. In this example there was a boundary condition at $U = 0$, this fact might explain the discrepancy in the state function for $U = 0.8$ and 1. The problem parameters and problem statistics are shown in Tables 7.13-7.15.

The knot points in λ-interval are 0.1, 0.3, 0.6, 1., 1.5. The knot points in U-interval are 0.2, 0.4, 0.6, 0.8. The integration domain was [0,2]x[0,1]. The residual error of the discretized differential equation was evaluated at (0.25,0.125), (0.25,0.25),...,(2,0.875). There are 30 2-dimensional finite elements. The discretized differential equation was evaluated both at internal collocation points and at boundaries of finite elements at those initial/boundary points of the integration domain, where there are no initial/boundary conditions. The discretized differential equation in element (k,l) k=1,2,...,6 l = 1,2,...,5 and at collocation points i = 1,2,3 j = 1,2,3 is

$$-(1-U^2)\frac{1}{\Delta\lambda_k}\left[\sum_{n=1}^{3} C_{nj}^{kl}\frac{dl_n(\lambda_i')}{d\lambda'}\right]+\alpha$$

$$\left\{\frac{1}{\Delta U_l^2}\left[\sum_{n=1}^{3} C_{in}^{kl}\frac{dl_n^2(U_j')}{dU'^2}\right]+\frac{1}{U}\frac{1}{\Delta U_l}\left[\sum_{n=1}^{3} C_{in}^{kl}\frac{dl_n(U_j')}{dU'}\right]\right\}-C_{ij}^{kl} \qquad (28)$$

$$k=1,2,...,6 \; l=1,2,3,4 \; i=2,3 \; j=2,3$$

$$k=1,2,...,6 \; l=5 \; i=2,3 \; j=2$$

The differential equation was not collocated where initial/boundary conditions were given. The discretized initial/boundary conditions are

$$c_{ij}^{kl}=1 \; k=1 \; l=1 \; i=1 \; j=1,2,3$$

$$c_{ij}^{kl}=1 \; k=1 \; l=2,3,4,5 \; i=1 \; j=2,3 \qquad (29)$$

$$\frac{1}{\Delta U_l}\sum_{n=1}^{3} C_{in}^{kl}\frac{dl_n(U_j')}{dU'} \quad \begin{cases} k=1,...,6 \; l=1 \; i=2,3 \; j=1 \\ k=1,...,6 \; l=5 \; i=2,3 \; j=3 \end{cases}$$

Table 7.13 shows the starting values and bounds for the state variable

Table 7.13 Starting Values

Function/ variable	lower bound	upper bound	starting point
C	0	1	1

The state variable C is unbounded, the bounds are only used for scaling. The numbering of C is shown in Figure 7.14

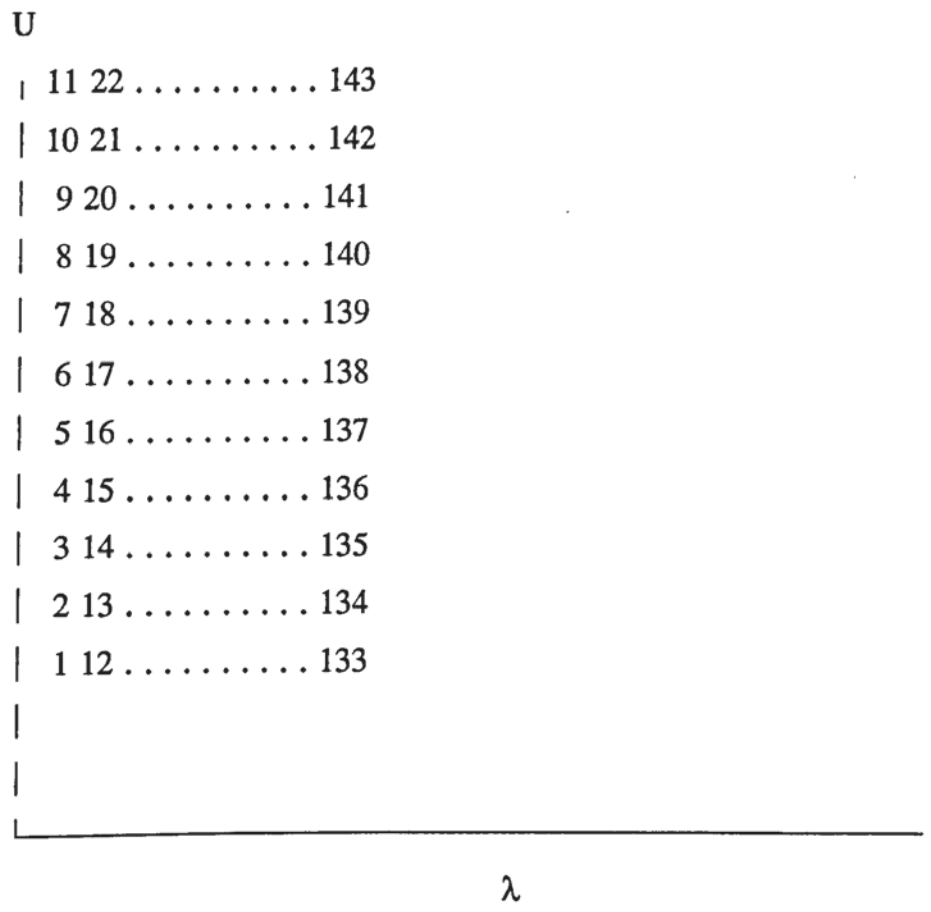

Fig. 7.14 NLP Variables for C

Table 7.14 shows the number of *NLP* variables derived from the state variable *C*.

Table 7.14 Relationship between Continuous and *NLP* Variables

Function/variable	NLP dimension
C	143

The problem was run twice, once using *GRG2* and once using *VF13AD* on the *VAX*. Table 7.15 shows the parameters of the collocation method applied and the *NLP* solution methods.

Table 7.15 Collocation Parameters and *NLP* Statistics

	GRG2	SQP

N	6	6
M	5	5
N_λ	6	6
N_U	5	5
n	143	143
m	144	144
nfun	322	-
niter	108	2
acc	0.000001	0.0001
CPU	00:14:03	00:01:49
err	0.1	0.1
aveerr	0.008	0.008

where N	=	number of internal collocation points in λ-interval
M	=	number of internal collocation points in U-interval
N_λ	=	number of finite elements in λ-interval
N_U	=	number of finite elements in U-interval
n	=	number of *NLP* variables
m	=	number of *NLP* constraints
nfun	=	number of function evaluations
niter	=	number of *NLP* iterations
acc	=	accuracy of the *NLP* solution
CPU	=	*CPU* time (hh:mm:ss)
err	=	the maximum point-wise residual error
aveerr	=	the average point-wise residual error

Table 7.16 compares the concentration values of the reactant at $\lambda = 0.5$

Table 7.16 Comparison of Concentration Values

U	GRG2	SQP	[98]
0	0.57	0.57	~0.58
0.2	0.55	0.55	~0.53
0.4	0.51	0.51	~0.47
0.6	0.39	0.39	~0.37
0.8	0.13	0.13	~0.30
1	0.12	0.12	~0.26

Figure 7.15 shows the values of *C* (*GRG2* run).

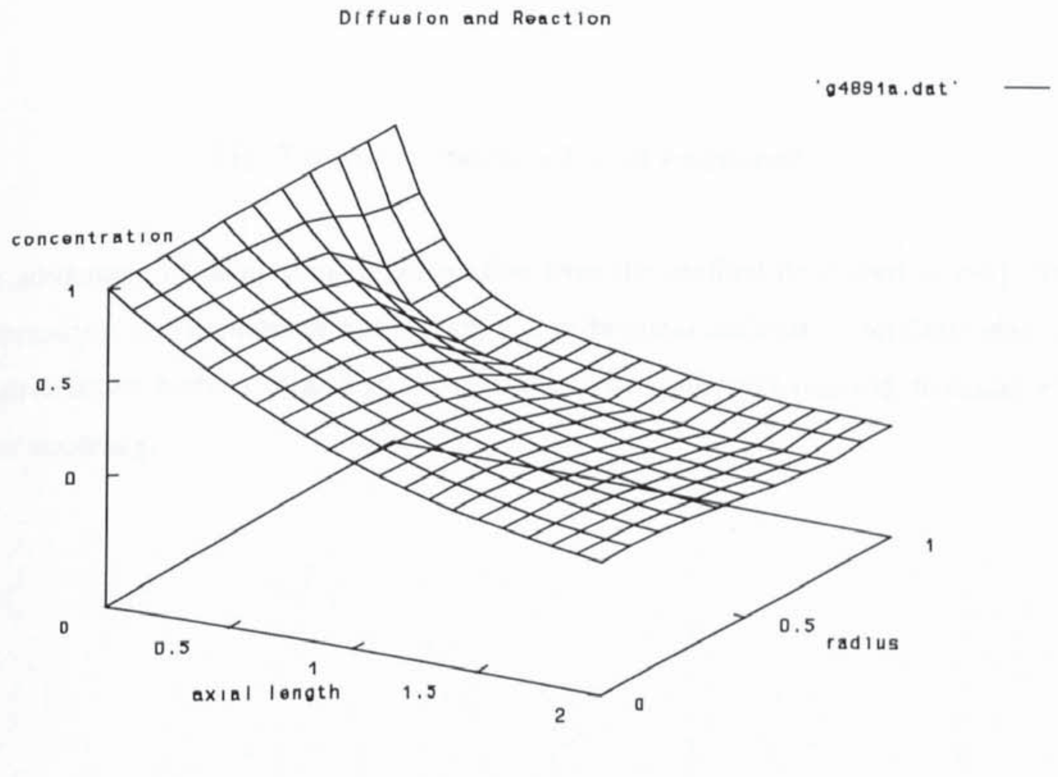

Fig. 7.15 concentration = f(axial length,radius)

Figure 7.16 shows the values of *C* (*SQP* run).

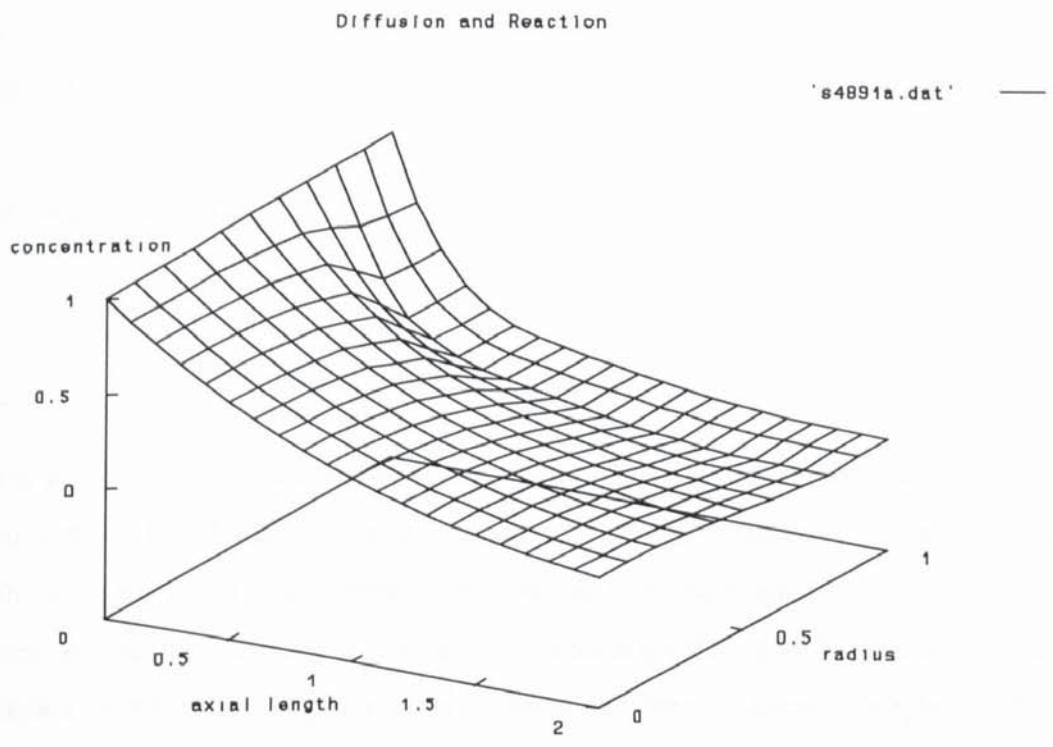

Fig. 7.16 concentration = f(axial length,radius)

The advantage of using double collocation over the method described in [98], finite differencing, that experience has shown, that orthogonal collocation on finite elements requires fewer number of grid points than the finite difference method, to achieve the same accuracy.

7.3 Example 5 : Time Delay

This example is concerned with optimal control of a time-delay system. *Pontryagin's* maximum principle is applied to derive necessary conditions of optimality that results in obtaining the *Hamiltonian* and the adjoint equations. Since the control vector is unconstrained, the minimum of *H*, the *Hamiltonian*, can be obtained from the stationary condition

$$\frac{\partial H}{\partial u} = 0 \tag{30}$$

where *u(t)* is the unconstrained control vector. The optimal control is obtained by solving the original differential equations, the adjoint equations and the stationary conditions. By using the stationary equations, *u* is eliminated from the state and adjoint equations yielding a two-point boundary-value problem. The unknown functions are expressed as polynomial expansions. The numerical example considered is a linear time-delay system, using algebraic manipulations, the boundary value problem was reduced to a system of linear algebraic equations, for which the solution can be obtained with no iteration. The linear time-delay system

$$\dot{x}(t) = x(t) + x(t-\theta) + u(t) \tag{31}$$

with the initial state profile

$$x(\tau) = 1 \qquad -\theta \le \tau \le 0 \tag{32}$$

is considered with the performance index

$$J = \int_0^2 (x^2 + u^2)\,dt \tag{33}$$

to be minimized.

The method developed in this work, orthogonal collocation on finite elements was applied to transform the optimal control problem into a *NLP* problem. The state profile closely approximates the solution given in [46]. The problem parameters and problem statistics are shown in Tables 7.17-7.18. The knot points are 0.2, 0.4, 0.6, 0.8, 1., 1.2, 1.4, 1.6, 1.8. The time delay is $\Theta = 1.0$. The integration domain is [0,2]. The residual error of the discretized differential equations was evaluated at 0.05, 0.1, ...,2. There are 10 finite elements. The discretized differential equation was evaluated both at the internal collocation points and at the boundaries of finite elements and at $t = 2$. It was not evaluated at $t = 0$, because initial condition is given at $t = 0$. The discretized differential equation in element *k*, k=1,2,...,10 and at collocation points i=1,2,3 is

$$\frac{1}{\Delta t_k} \sum_{n=1}^{3} x_n^k(t_i') \frac{dl_n(t_i')}{dt'} - x_i^k(t_i') - x_i^k(t_i' - \theta) - u^k(t_i') \qquad (34)$$

$$k = 1,2,...,10 \qquad i = 2,3$$

The discretized initial condition is

$$x_1^1 = 1 \qquad k = 1 \qquad i = 1 \qquad (35)$$

The objective functional has been transformed into a differential equation

$$\frac{dI}{dt} = x^2 + u^2 \qquad (36)$$

$$I(0) = 0$$

It was discretized as follows

at k = 1,2,...,10 i=2,3

$$\frac{1}{\Delta t_k} \left[\sum_{n=1}^{3} I_n^k \frac{dl_n(t_i')}{dt'} \right] - (x_i^k)^2 - (u^k)^2 = 0 \qquad (37)$$

$$I_1^1 = 0$$

The objective functional is

$$I_3^{10} \tag{38}$$

Table 7.17 shows the number of *NLP* variables derived from the state and control variables.

Table 7.17 Relationship between Continuous and *NLP* Variables

State or control function	NLP dimension
x	21
u	10
I	21

The problem was run twice, once using *GRG2* and once using *VF13AD* on the *VAX*. Table 7.18 shows the parameters of the collocation method applied and the *NLP* solution methods.

Table 7.18 Collocation Parameters and *NLP* Statistics

	GRG2	SQP
N	10	10
N_t	10	10
n	52	52
m	43	43
nfun	437	-
niter	44	76
acc	0.000001	0.0001
CPU	00:19:00	00:06:05
err	0.02	0.03

	GRG2	SQP
aveerr	0.002	0.001
OF	6.25	6.293

where N	=	number of internal collocation points in the integration domain
N_t	=	number of finite elements
n	=	number of *NLP* variables
m	=	number of *NLP* constraints
nfun	=	number of function evaluations
niter	=	number of *NLP* iterations
acc	=	accuracy of the *NLP* solution
CPU	=	*CPU* time (hh:mm:ss)
err	=	the maximum point-wise residual error
aveerr	=	the average point-wise residual error
OF	=	$\int_0^2 (x^2 + u^2)dt$

Table 7.19 compares the optimal values of the state variables.

Table 7.19 Comparison of State Variables

time	GRG2	SQP	[46]
0	1	1	1
0.4	0.44	0.40	~0.45
0.8	0.29	0.27	~0.30
1.2	0.39	0.30	~0.36
1.6	0.55	0.45	~0.53
2	0.89	0.73	~0.84

Figure 7.17 shows the optimal values of x (GRG2 run).

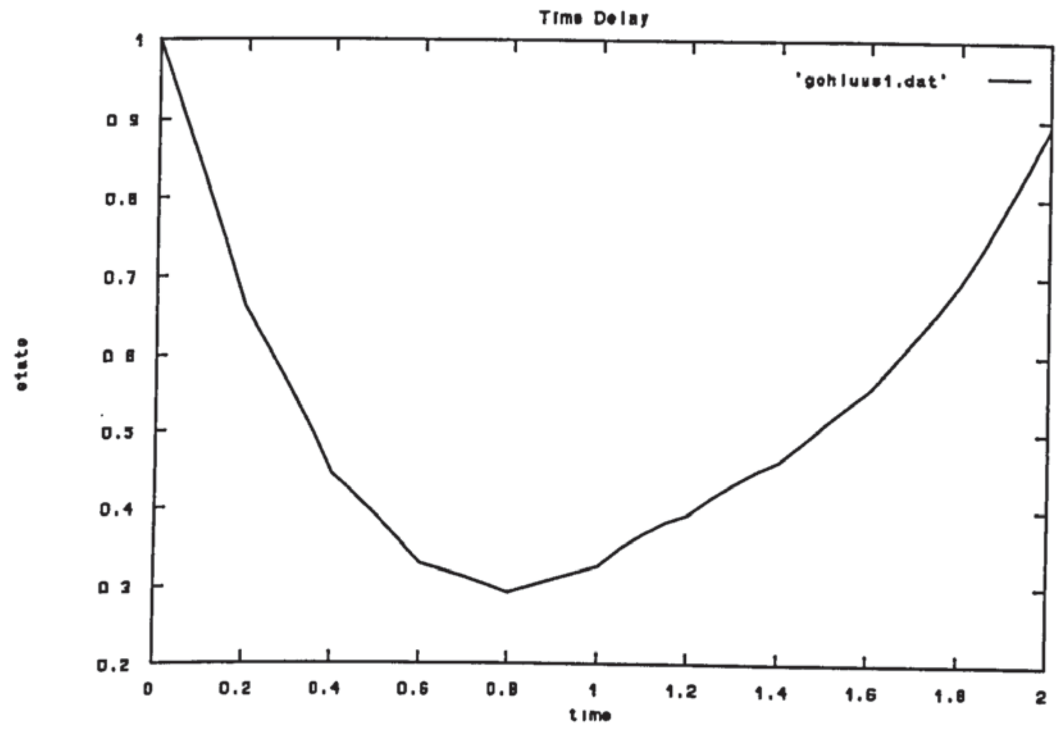

Fig. 7.17 state variable = f(time)

Figure 7.18 shows the optimal values of *u* (GRG2 run).

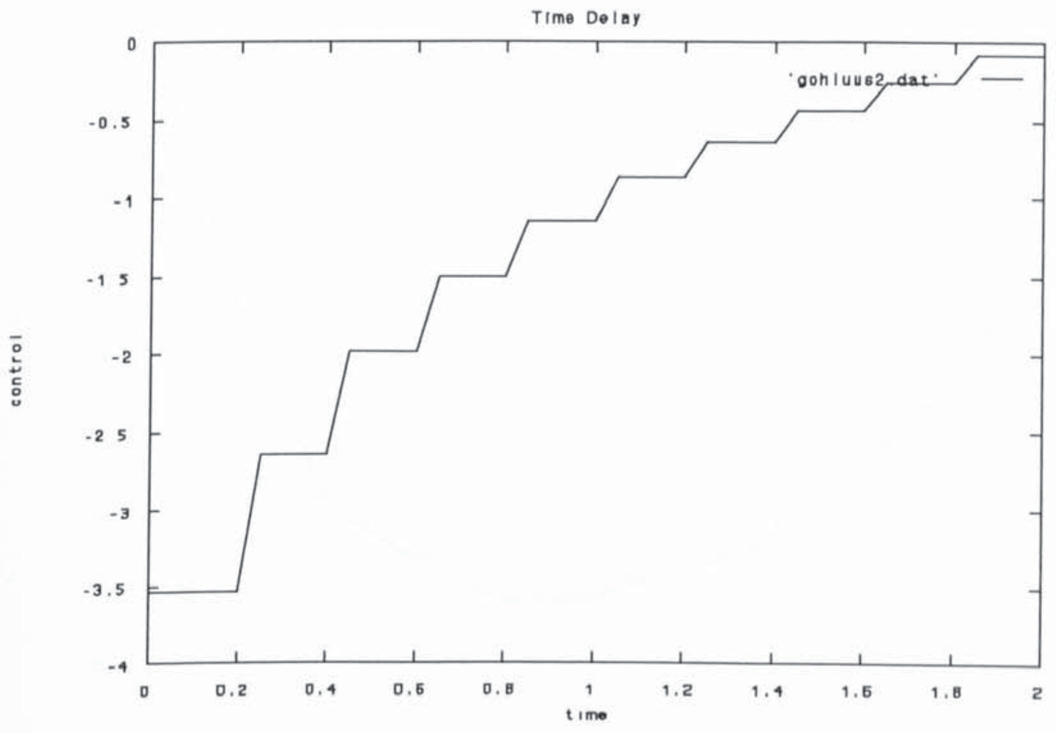

Fig. 7.18 control variable = f(time)

Figure 7.19 shows the optimal values of x (SQP run).

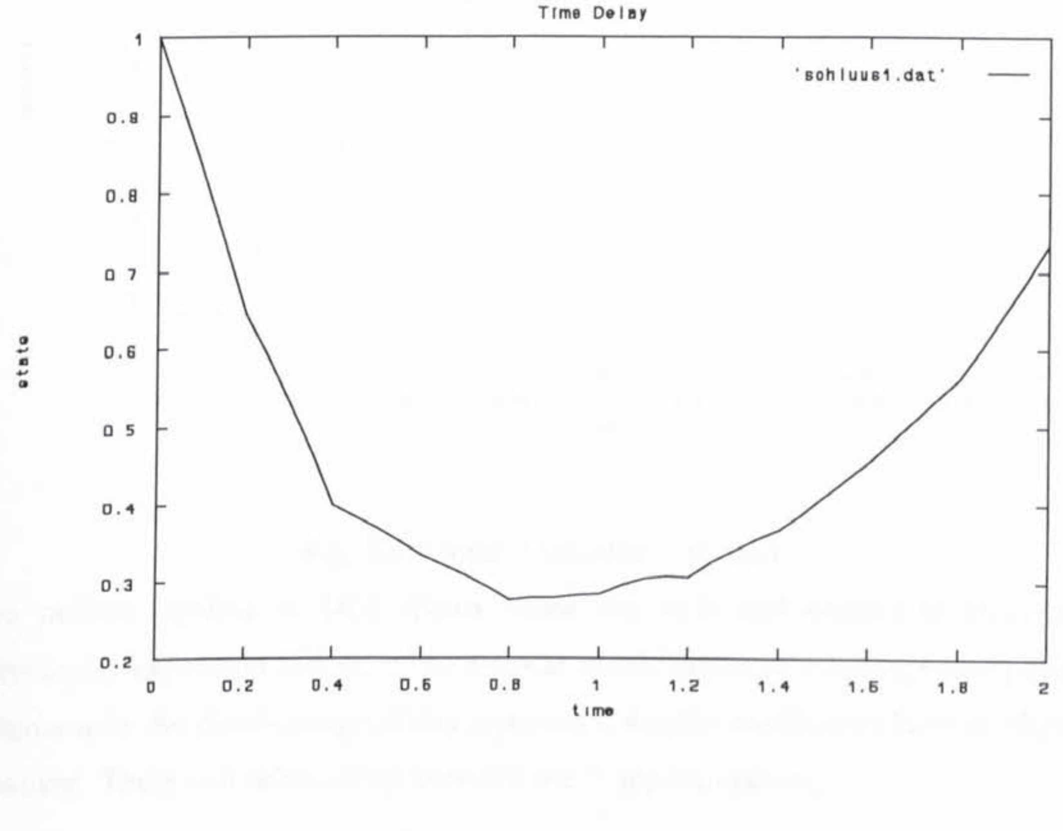

Fig. 7.19 state variable = f(time)

Figure 7.20 shows the optimal values of u (SQP run).

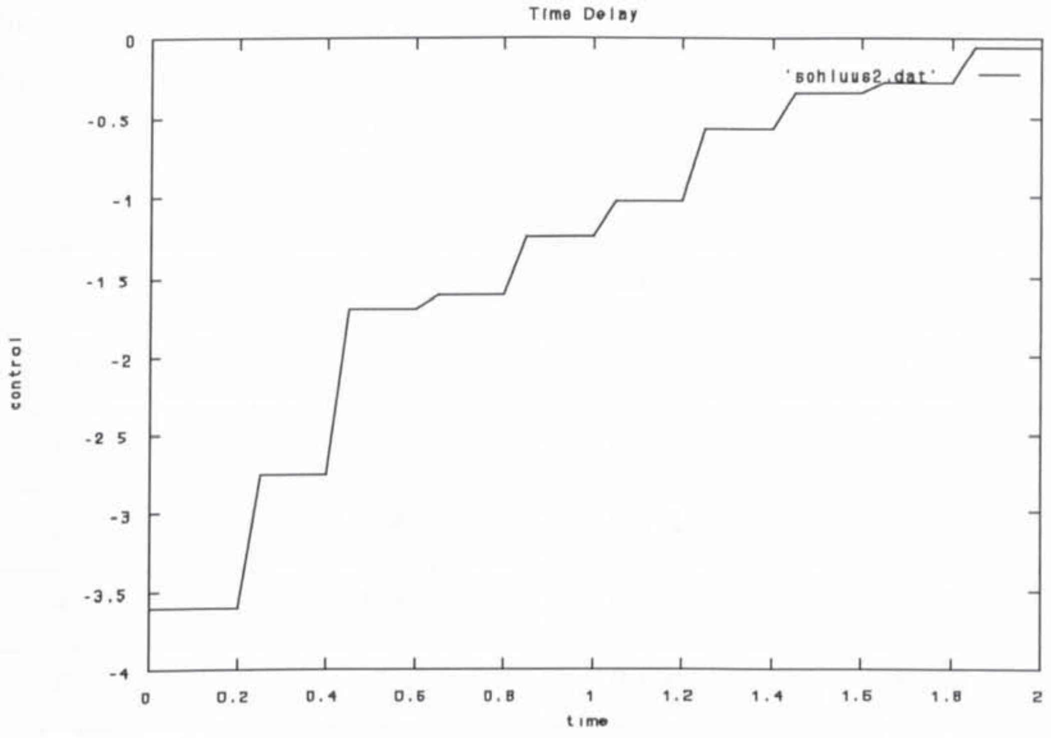

Fig. 7.20 control variable = f(time)

The method applied in [46] approximated the state and control function as a polynomial expansion instead of as a linear combination of *Lagrange* interpolation polynomials, the disadvantage of this approach is that the coefficients have no physical meaning. There is a relationship between the 2 approximations:

$$x(t) = \sum_{j=1}^{N+2} c_j t^{j-1} \equiv \sum_{j=1}^{N+2} x_j l_j(t) \tag{39}$$

The method can only be applied if the control vector is unconstrained. The approach developed in this work does not apply the theorems and results of control theory, because then it would be difficult to automate the computational procedure and it would not be so general-purpose, e.g., in [46] the fact that the control vector was unconstrained, was made use of. The work in this thesis does not make use of the linearity of systems, because there would be problems concerning automating the conversion of optimal control problems into *NLP* problems.

7.4 Example 6 : Parameter Estimation Example

Given experimental time-concentration data for species *A*,*B*,*C*, the task is to find the values for the rate constants k_1, k_2 and k_3, that best fit the experimental data. The data are presented in Table 7.20.

Table 7.20 Experimental Data

	concentration		
Time	A	B	C
0	0.75	0	0
2	0.266	0.412	0.0479
4	0.105	0.520	0.148
6	-	0.470	0.216
8	0.0095	0.420	0.329
10	0.00525	0.377	0.357

The mathematical model

$$\frac{dA}{dt} = -k_1 A$$

$$\frac{dB}{dt} = k_1 A + k_3 C - k_2 B \tag{40}$$

$$C = A_0 + B_0 + C_0 - A - B \text{ overall mass balance}$$

where A_0, B_0, C_0 are the initial values of *A*, *B*, *C*, respectively. This problem is shown in the Simusolv User Guide, [93], where it was solved by maximizing the maximum likelihood function which is a measure of how well the model fits the experimental data. In this work orthogonal collocation on finite elements was applied to transform the model into an *NLP* problem. The values of the rate constants and the functions *A*(*t*), *B*(*t*), *C*(*t*) closely approximate the experimental data as well as the solution given in the Simusolv manual. The least-square function

$$\sum_{t \in \{0,2,4,6,8,10\}}^{6} \left[(A(t)-A'(t))^2 + (B(t)-B'(t))^2 + (C(t)-C'(t))^2 \right] \qquad (41)$$

was minimized using *GRG2* and *VF13AD*. The problem parameters and problem statistics are shown in Tables 7.21-7.23. The knot points are 1,2,3,4,5,6,7,8,9,10,11. The integration domain is [0,12]. The residual error of the discretized differential equations was evaluated at [0.5,1.,1.5,...12.]. There were 12 finite elements. The discretized differential equations were evaluated at internal collocation points, at finite element boundaries and at t=12. They were not evaluated at t=0, where initial conditions are given. The discretized differential equations are in element k, k=1,2,...,12 and at collocation points i=1,2,3

$$\begin{aligned} &\frac{1}{\Delta t_k} \sum_{n=1}^{3} A_n^k \frac{dl_n(t_i')}{dt'} + k_1 A_i^k \quad \{ k = 1,2,...,12 \ i = 2,3 \\ &\frac{1}{\Delta t_k} \sum_{n=1}^{3} B_n^k \frac{dl_n(t_i')}{dt'} - k_1 A_i^k - k_3 C_i^k + k_2 B_i^k \quad \{ k = 1,2,...,12 \ i = 2,3 \end{aligned} \qquad (42)$$

The overall material balance was discretized in the element k, k=1,2,...,12 and at collocation points i=1,2,3

$$C_i^k - A_0 - B_0 - C_0 + A_i^k + B_i^k \quad \begin{matrix} k=1 & i=1,2,3 \\ k=2,...,12 & i=2,3 \end{matrix} \qquad (43)$$

The discretized initial conditions are

$$\begin{aligned} A_1^1 &= 0.75 \ \{ k=1 \ i=1 \\ B_1^1 &= 0 \quad \{ k=1 \ i=1 \\ C_1^1 &= 0 \quad \{ k=1 \ i=1 \end{aligned} \qquad (44)$$

The objective function was evaluated at

$$\begin{array}{lll} t{=}0 & k{=}1 & i{=}1 \\ t{=}2 & k{=}2 & i{=}3 \\ t{=}4 & k{=}4 & i{=}3 \\ t{=}6 & k{=}6 & i{=}3 \\ t{=}8 & k{=}8 & i{=}3 \\ t{=}10 & k{=}10 & i{=}3 \\ t{=}12 & k{=}12 & i{=}3 \end{array} \tag{45}$$

Table 7.21 shows the starting values and bounds for the state variables

Table 7.21 Starting Values

Function	lower bound	upper bound	starting point
A	0	1	0.75
B	0	1	0
C	0	1	0

The numbering system of unknowns is illustrated for A in Figure 7.21

```
   1  2 3  4  5 6  7  8  91011 12131415 16171819 202122 232425
—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x—x————————
t= 0    1    2     3   4    5    6    7    8    9   10   11   12
```

Fig. 7.21 NLP Variables for A

Table 7.22 shows the number of *NLP* variables derived from the state variables

Table 7.22 State Functions versus NLP Variables

State Function	NLP dimension
A	25
B	25

State Function	NLP dimension
C	25

The problem was run twice, once using *GRG2* and once using *VF13AD* on the *VAX*. Table 7.23 shows the parameters of the collocation method applied and the *NLP* solution methods.

Table 7.23 Collocation Parameters and *NLP* Statistics

	GRG2	SQP
N	12	12
N_t	12	12
n	75	75
m	76	79
nfun	404	-
niter	52	11
acc	0.000001	0.000001
CPU	00:01:29	00:01:07
err	0.02	0.02
aveerr	0.0005	0.0005
OF	0.001	0.001
k_1	0.506	0.506
k_2	0.102	0.102
k_3	0.033	0.033

where N	=	number of internal collocation points in the integration domain
N_t	=	number of finite elements
n	=	number of *NLP* variables
m	=	number of *NLP* constraints
nfun	=	number of function evaluations
niter	=	number of *NLP* iterations
acc	=	accuracy of the *NLP* solution
CPU	=	*CPU* time (hh:mm:ss)
err	=	the maximum point-wise residual error
aveerr	=	the average point-wise residual error
OF	=	the value of the least-square function
k_1	=	the rate constant of species A
k_2	=	the rate constant of species B
k_3	=	the rate constant of species C

The parameter estimation problem has been solved also using control vector parameterization in conjunction with the Hooke-Jeeves and BFGS method and with the 4th order Runge-Kutta method. The objective functional evaluation, the derivatives evaluation, and the mass-balance equation evaluation codes are shown in Appendix A.

Table 7.24 compares the values of the rate constants.

Table 7.24 Comparison Values of the Rate Constants

	GRG2	SQP	Hooke-Jeeves	BFGS	SIMU-SOLV
k_1	0.506	0.506	0.504	0.503	0.507
k_2	0.102	0.102	0.107	0.104	0.103
k_3	0.033	0.033	0.043	0.034	0.036

Table 7.25 compares the concentration values of A.

Table 7.25 Comparison of Concentration of A

time	GRG2	SQP	Hooke-Jeeves	BFGS	SIMU-SOLV
0	0.75	0.75	0.75	0.75	0.75
2	0.27	0.27	0.27	0.27	0.27
4	0.1	0.1	0.09	0.1	0.098
6	0.03	0.03	0.03	0.03	0.035
8	0.01	0.01	0.01	0.01	0.012
10	0.004	0.004	0.004	0.004	0.004
12	0.001	0.001	0.001	0.001	0

Table 7.26 compares the concentration values of B.

Table 7.26 Comparison of Concentration of B

time	GRG2	SQP	Hooke-Jeeves	BFGS	SIMU-SOLV
0	0	0	0	0	0
2	0.42	0.42	0.42	0.42	0.42
4	0.50	0.50	0.50	0.50	0.50
6	0.47	0.47	0.47	0.47	0.47
8	0.42	0.42	0.42	0.42	0.42

time	GRG2	SQP	Hooke-Jeeves	BFGS	SIMU-SOLV
10	0.37	0.37	0.37	0.37	0.37
12	0.33	0.33	0.33	0.33	0.33

Table 7.27 compares the concentration values of C.

Table 7.27 Comparison of Concentration of C

time	GRG2	SQP	Hooke-Jeeves	BFGS	SIMU-SOLV
0	0	0	0	0	0
2	0.05	0.05	0.05	0.05	0.05
4	0.14	0.14	0.14	0.14	0.14
6	0.23	0.23	0.23	0.23	0.23
8	0.30	0.30	0.31	0.30	0.30
10	0.36	0.36	0.36	0.36	0.36
12	0.41	0.41	0.41	0.41	0.38

Figure 7.22 shows the optimal values of *A* (GRG2 run).

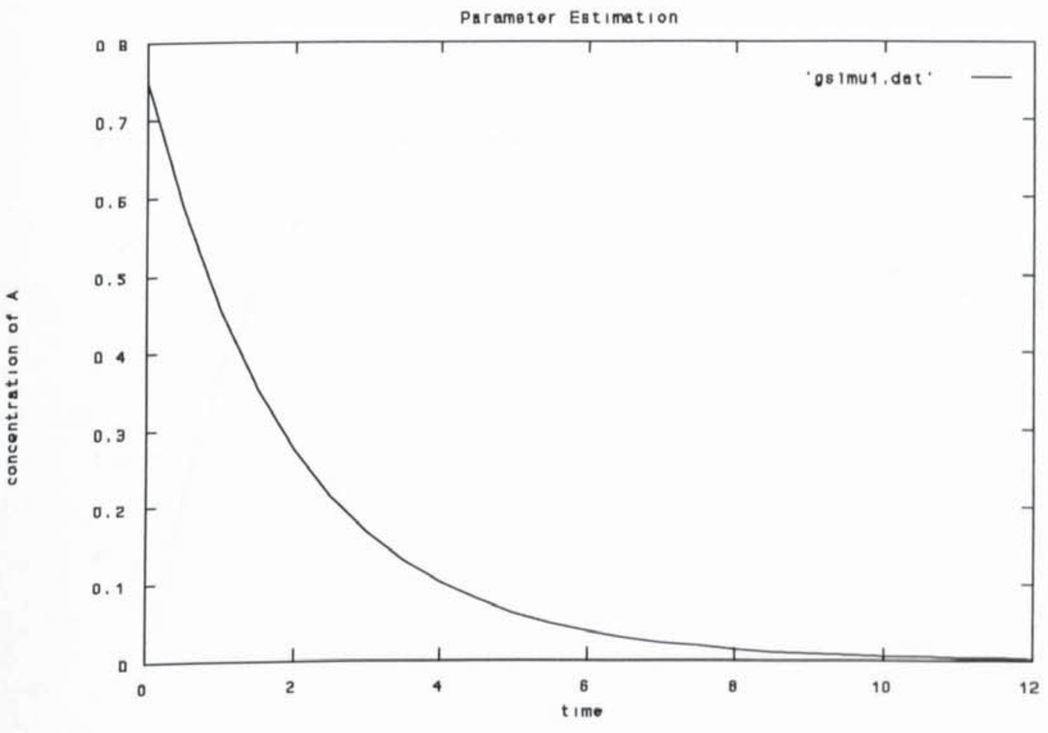

Fig. 7.22 concentration of A = f(time)

Figure 7.23 shows the optimal values of *B* (GRG2 run).

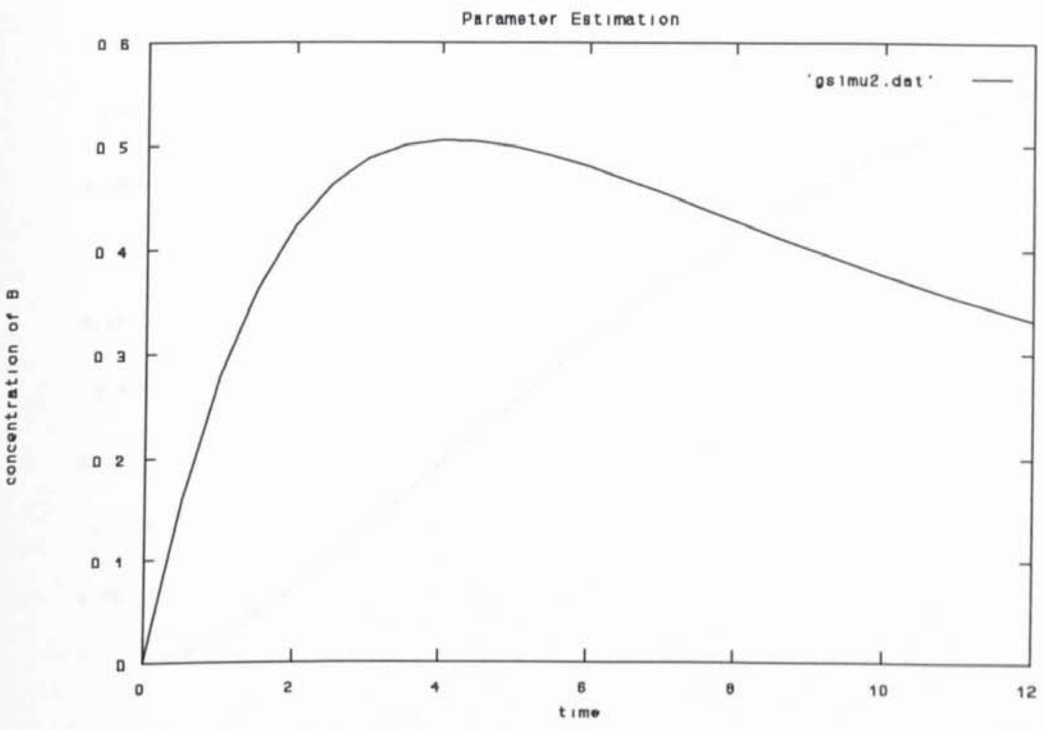

Fig. 7.23 concentration of B = f(time)

Figure 7.24 shows the optimal values of *C* (GRG2 run).

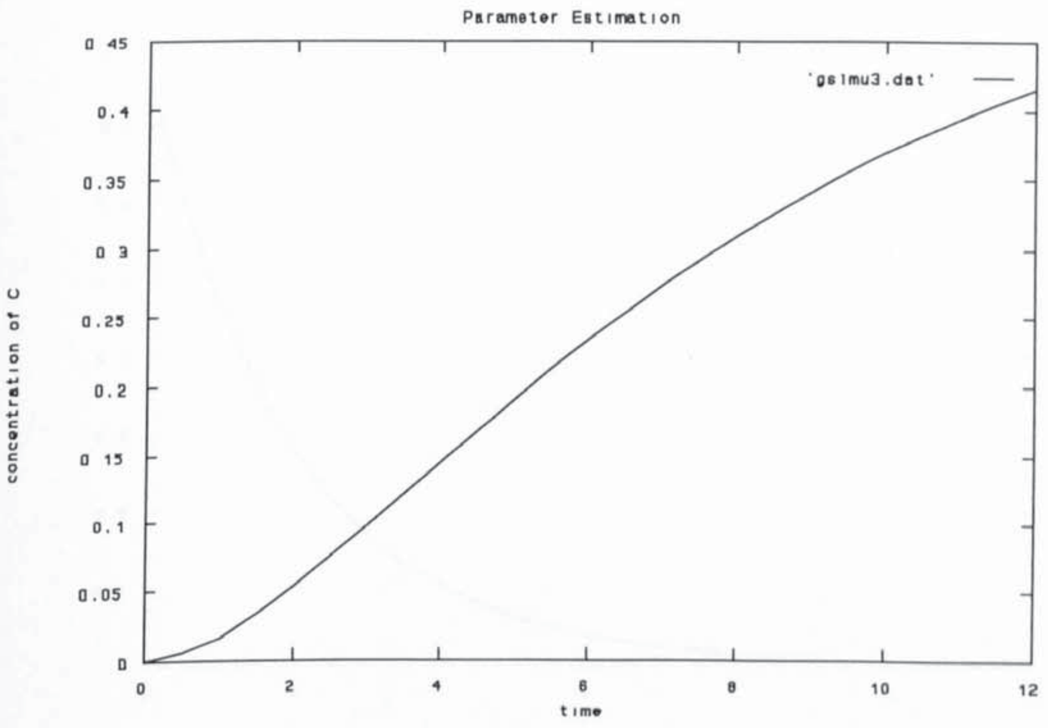

Fig. 7.24 concentration of C = f(time)

Figure 7.25 shows the optimal values of *A* (SQP run).

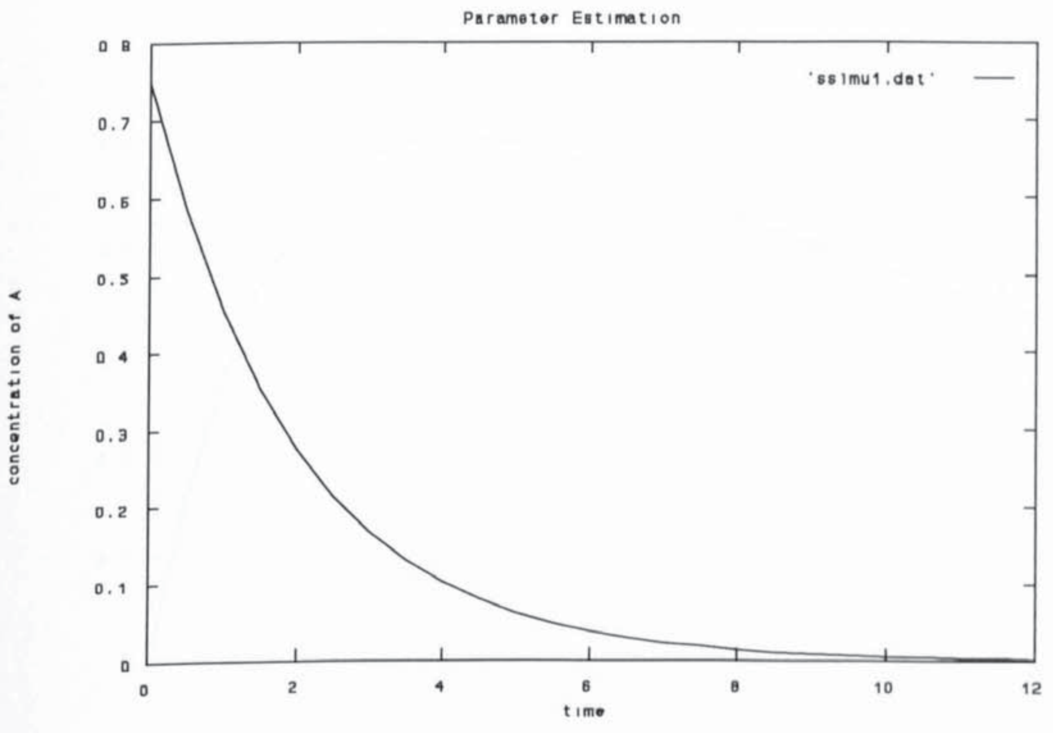

Fig. 7.25 concentration of A = f(time)

Figure 7.26 shows the optimal values of *B* (SQP run).

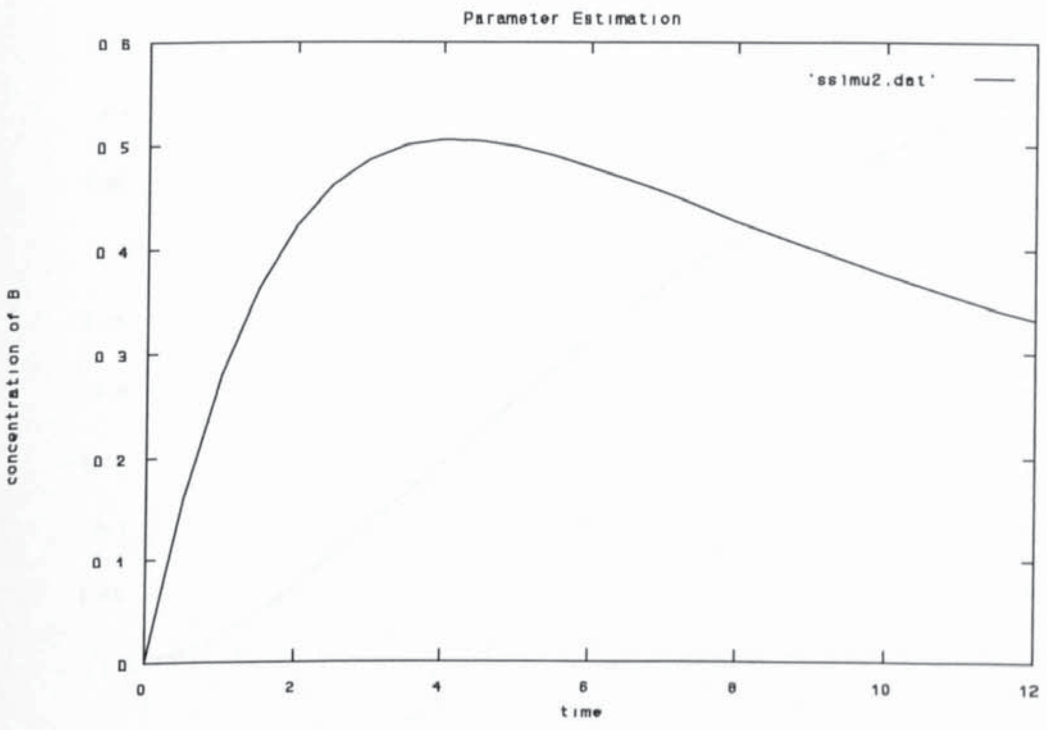

Fig. 7.26 concentration of B = f(time)

Figure 7.27 shows the optimal values of *C* (SQP run).

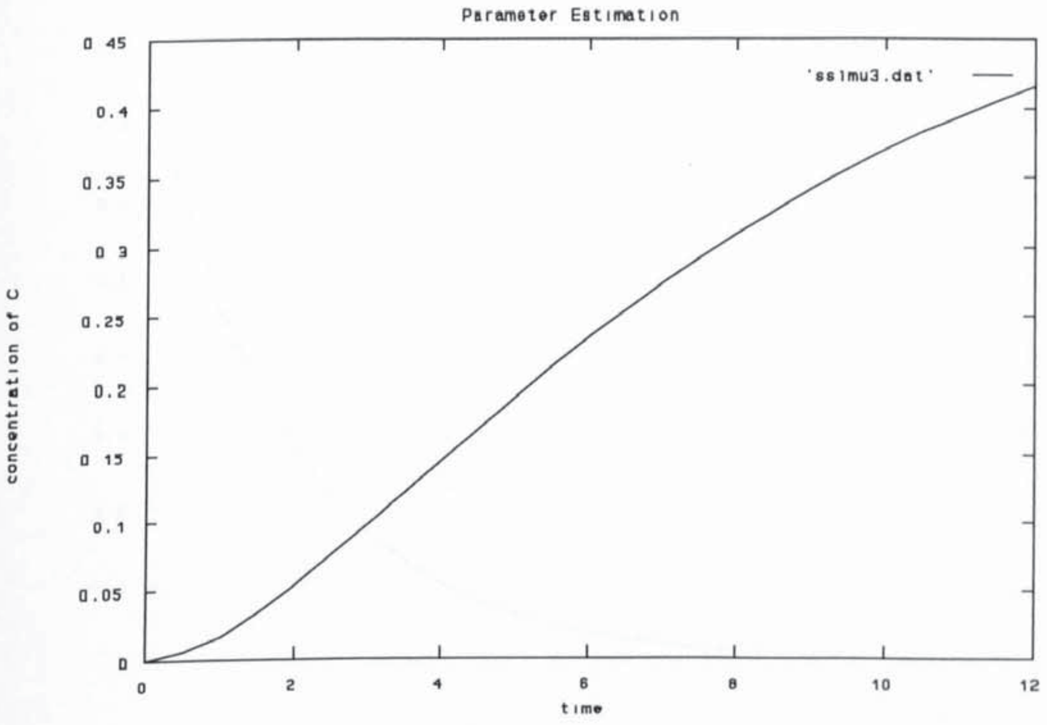

Fig. 7.27 concentration of C = f(time)

Figure 7.28 shows the optimal values of *A* (CVP HOOKE-JEEVES run).

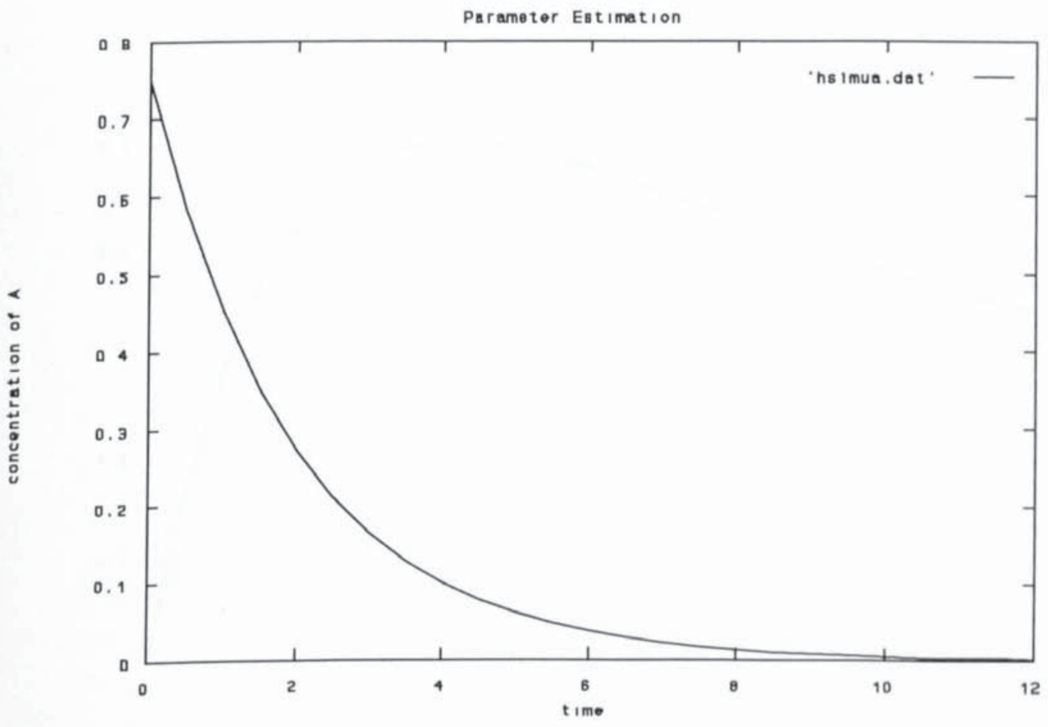

Fig. 7.28 concentration of A = f(time)

Figure 7.29 shows the optimal values of *B* (CVP HOOKE-JEEVES run).

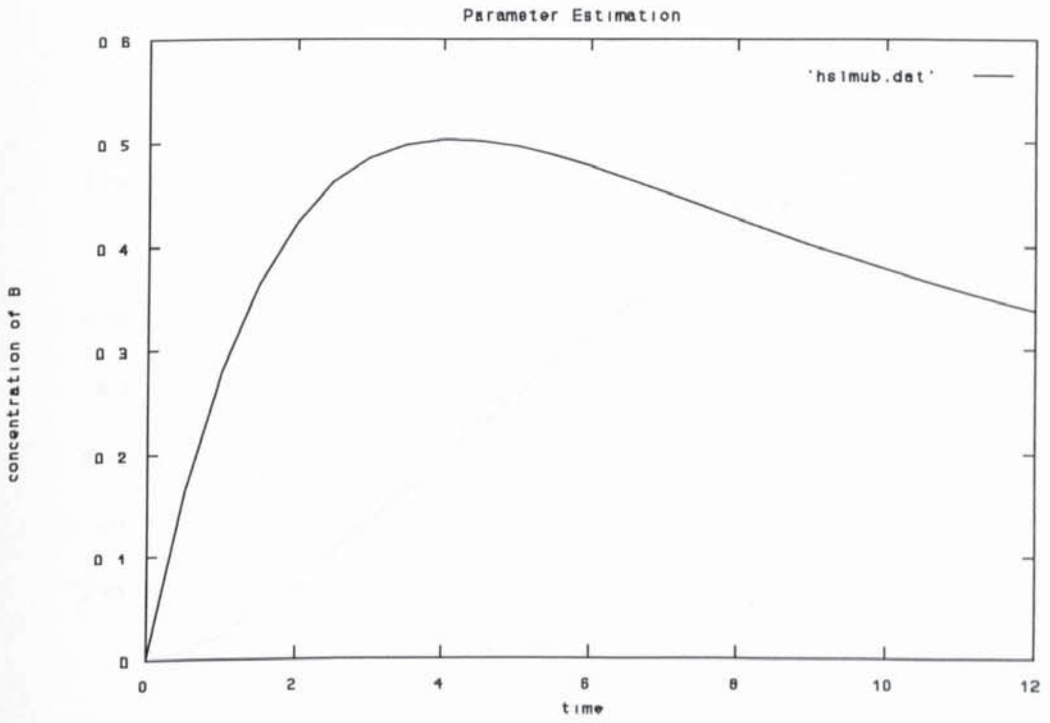

Fig. 7.29 concentration of B = f(time)

Figure 7.30 shows the optimal values of *C* (CVP HOOKE-JEEVES run).

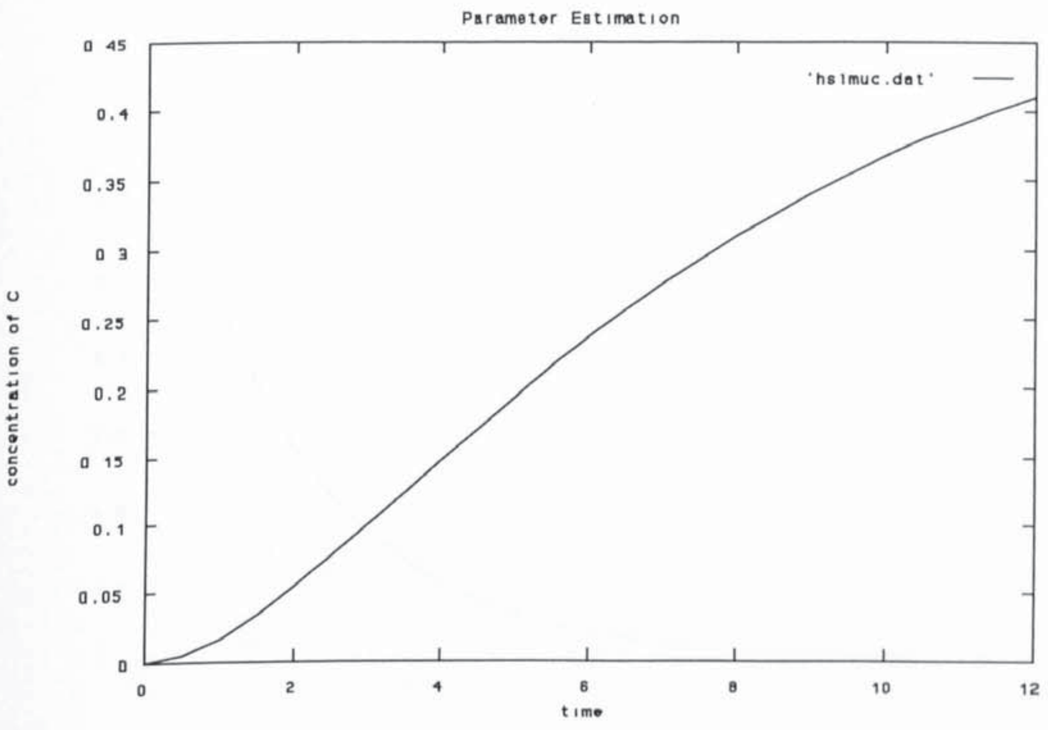

Fig. 7.30 concentration of C = f(time)

Figure 7.31 shows the optimal values of *A* (CVP BFGS run).

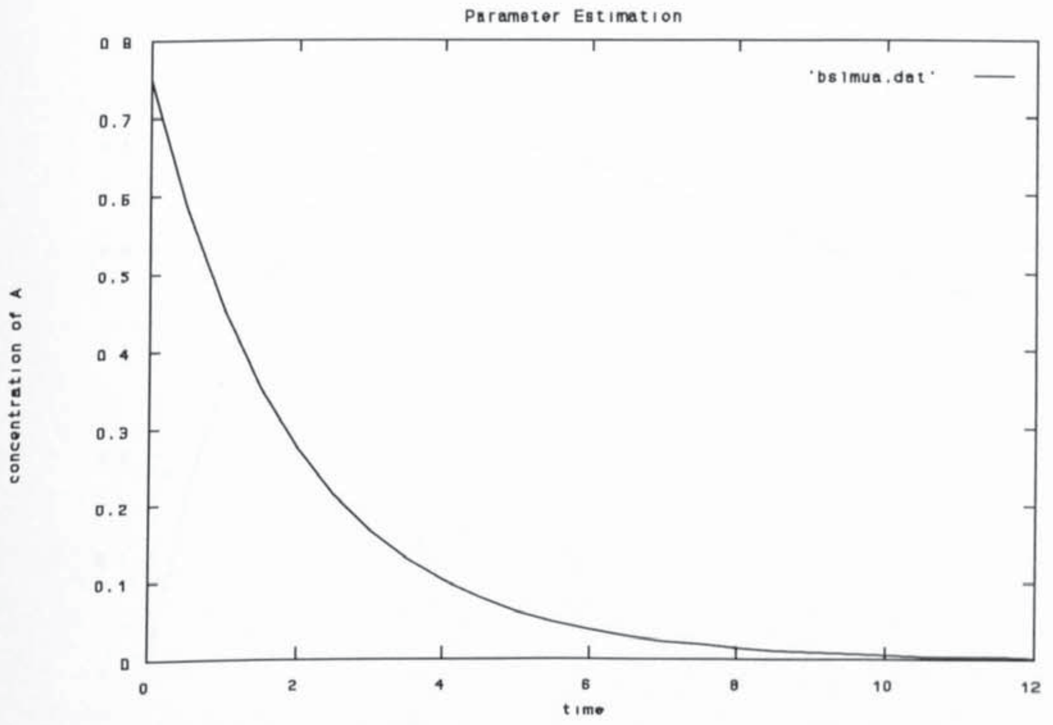

Fig. 7.31 concentration of A = f(time)

Figure 7.32 shows the optimal values of *B* (CVP BFGS run).

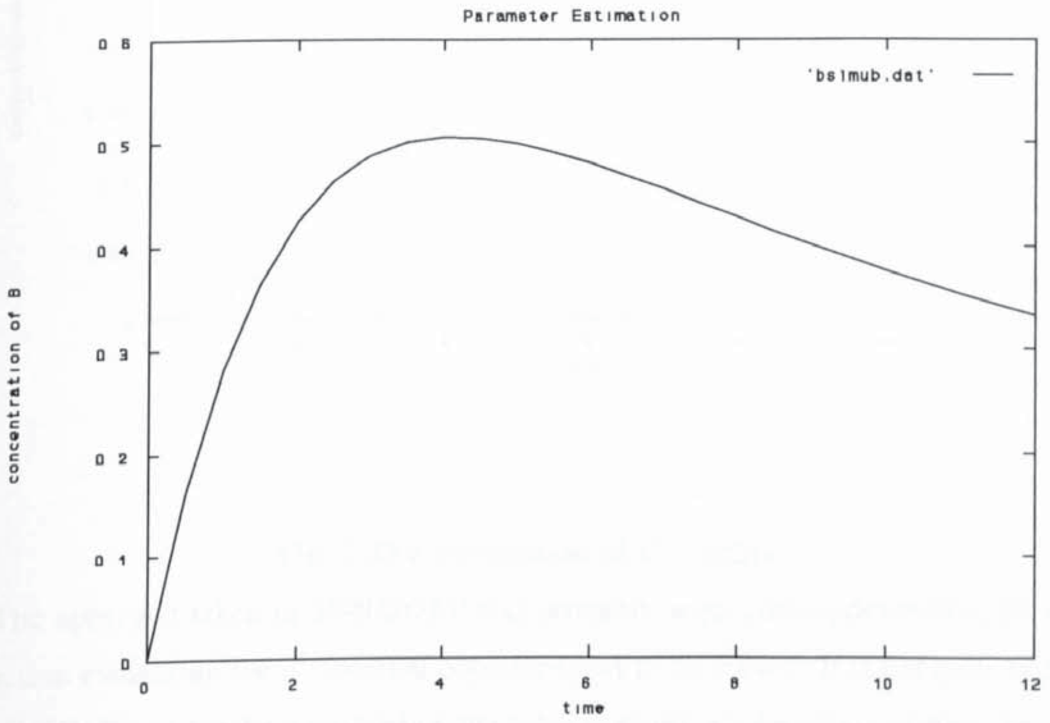

Fig. 7.32 concentration of B = f(time)

Figure 7.33 shows the optimal values of *C* (CVP BFGS run).

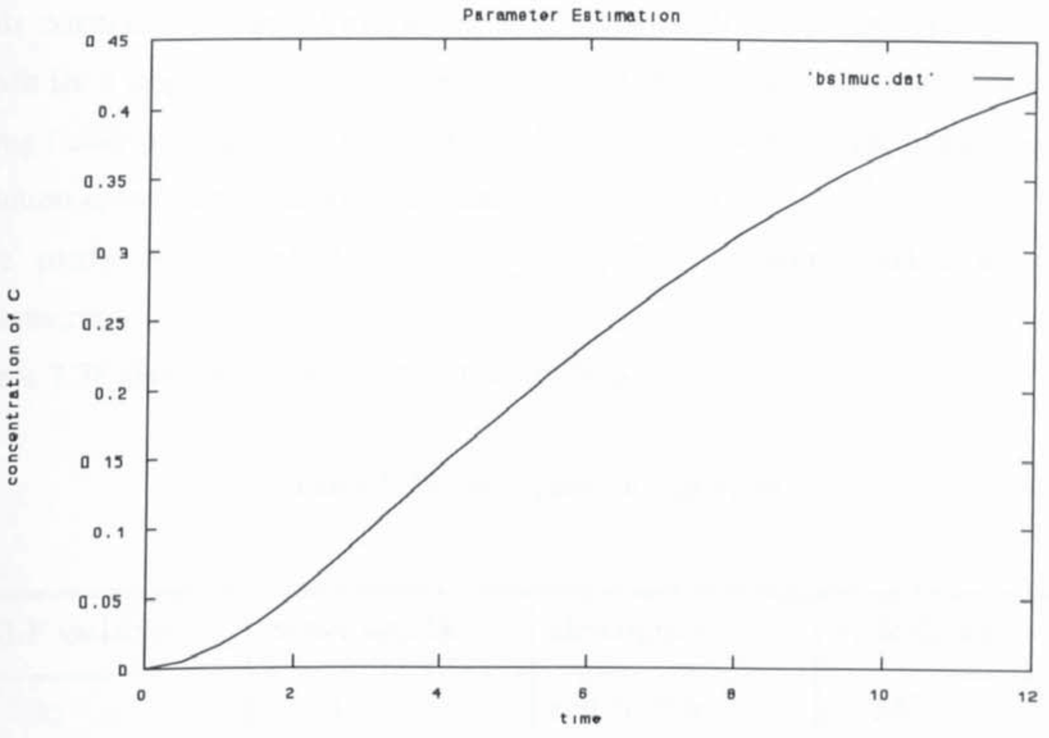

Fig. 7.33 concentration of C = f(time)

The approach taken in *SIMUSOLV* was probably sequential optimization, for each function evaluation the differential equations had to be solved. It is not sure whether *SIMUSOLV* can handle a case when the rate constants are functions of time, because they depend on temperature, which in turn is a function of time. In this work simultaneous simulation and optimization strategy and the sequential optimization strategy control vector parameterization were used.

7.5 Example 7 : A Nonlinear Programming Example

Optimization of an Electrolytic Cell

This example is concerned with a model of a chlor-alkali cell to determine maximum profit for a single cell. The model is described in [99]. In [99] the model was solved using *GRG2* package. Here it was solved also using the same package and the optimal solution closely approximates the solution given in [99].

The model consists of 40 variables, 37 equality constraints and 2 inequality constraints.

Table 7.28 shows the description of the variables.

Table 7.28 Description of Variables

NLP variable	model variable	Description	scale factor
x_1	I	cell current (kiloamperes)	10^{-3}
x_2	i_a	anodic current density (amps/ ft^2)	1
x_3	i_c	cathodic current density (amps/ft^2)	1
x_4	D	decomposition of salt per pass	10^2
x_5	U	current efficiency	10^2
x_6	m_f	sodium in feed brine (lbmoles/ hr)	1

NLP variable	model variable	Description	scale factor
x_7	M_f	salt molality of feed brine (gfw/kgm H_2O)	1
x_8	w_f	feed brine H_2O flow rate (lbs/ hr)	10^{-2}
x_9	w_b	feed brine mass flow rate (lbs/ hr)	10^{-2}
x_{10}	L	life of anode (days)	1
x_{11}	l_a	current day of anode life (day s)	1
x_{12}	θ	fractional age of anode	1
x_{13}	l_d	age of the nth diaphragm (days)	1
x_{14}	d_a	anode blade thi- ckness (feet)	10^2
x_{15}	d_{gap}	anode-diaphragm gap (feet)	10^3
x_{16}	m_{CL2}^+	chlorine evolv- ed	10^2

NLP variable	model variable	Description	scale factor
x_{17}	m^+_{H2O}	H_2O formed at anode (lbmoles/ hr)	10^2
x_{18}	T^+	anolyte tempe- rature (degrees F)	1
x_{19}	p_0^+	vapor pressure of H_2O at x_{18} (mm Hg)	1
x_{20}	M^+	salt molality of anolyte (gfw /kgm H_2O)	1
x_{21}	p^+	vapor pressure of anolyte (mm Hg)	1
x_{22}	m_{vap}^+	H_2O vaporized at anode (lbmol es/hr)	1
x_{23}	w_d	H_2O in anolyte flowing into diaphragm (lbs/ hr)	10^{-2}
x_{24}	η^+	chlorine over- voltage resis- tance (ohm)	10^6

NLP variable	model variable	Description	scale factor
x_{25}	p	resistance of anolyte liquor (ohm ft)	10^3
x_{26}	R_{gap}	resistance of anolyte in anolyte diaphragm gap (ohm)	10^6
x_{27}	$N^{\cdot}$	caustic molality of catholyte (gfw/kgm H_2O)	1
x_{28}	$M^{\cdot}$	salt molality of catholyte (gfw/kgm H_2O)	1
x_{29}	$T^{\cdot}$	catholyte temperature (degrees F)	1
x_{30}	$p_0^{\cdot}$	vapor pressure of H_2O at x_{29} (mm Hg)	1
x_{31}	$p^{\cdot}$	vapor pressure of catholyte (mm Hg)	1
x_{32}	$m_{vap}^{\cdot}$	water vaporized at cathode (lbmoles/hr)	1

NLP variable	model variable	Description	scale factor
x_{33}	w_c	mass flow rate of H_2O in cell liquor (lbs/hr)	10^{-2}
x_{34}	η^-	hydrogen overvoltage resistance (ohm)	10^6
x_{35}	R_d	diaphragm resistance (ohms)	10^6
x_{36}	E^+	thermodynamic rev voltage at anode (volts)	10^2
x_{37}	V^+	anode side half-cell voltage (volts)	1
x_{38}	E^-	thermodynamic rev voltage at cathode (volts)	1
x_{39}	V^-	cathode side half-cell voltage (volts)	1
x_{40}	V	total cell voltage (volts)	1

Model Formulation

The cell consists of 2 parts, anode side and cathode side, separated by a deposited asbestos diaphragm. The model consists primarily of mass, heat and voltage balances on each side of the cell. The derivation of the model is described in detail in [99].

Table 7.29 shows initial values, final values and bounds on the variables.

Table 7.29 Initial and Final Values and Bounds

NLP variable	model variable	initial value	final value	lower bound	upper bound
x_1	I	56	60	0	60
x_2	i_a	121.7	130.43	0	10^{30}
x_3	i_c	121.7	130.43	0	10^{30}
x_4	D	52	50.84	10^{-6}	100
x_5	U	95.8	96.30	0	100
x_6	m_f	8.49	9.34	0	10^{30}
x_7	M_f	5.7	6.88	0	10^{30}
x_8	w_f	14.9	13.57	0	10^{30}
x_9	w_b	19.86	19.04	0	10^{30}
x_{10}	L	259	240.55	0	10^{30}
x_{11}	l_a	0	0	0	10^{30}
x_{12}	Θ	0	0	0	1
x_{13}	l_d	0	0	0	10^{30}
x_{14}	d_a	10.42	10.41	2.08	10.42
x_{15}	d_{gap}	31.25	31.25	31.25	72.92
x_{16}	m_{CL2}^{+}	221	237.67	0	10^{30}
x_{17}	m_{H2O}^{+}	9.57	9.11	0	10^{30}
x_{18}	T^{+}	195	209.96	86	230

NLP variable	model variable	initial value	final value	lower bound	upper bound
x_{19}	p_0^+	537	729.85	0	10^{30}
x_{20}	M^+	4.39	5.73	0	10^{30}
x_{21}	p^+	452	574.33	0	750
x_{22}	m_{vap}^+	3.42	7.91	0	10^{30}
x_{23}	w_d	14.3	12.16	0	10^{30}
x_{24}	η^+	4.38	4.22	0	10^{30}
x_{25}	p	67.3	58.12	0	10^{30}
x_{26}	R_{gap}	5.31	4.58	0	10^{30}
x_{27}	N^-	3.57	4.84	0	12.5
x_{28}	M^-	3.29	4.68	0	10^{30}
x_{29}	T^-	197	215.36	68	230
x_{30}	p_0^-	562	812.18	0	10^{30}
x_{31}	p^-	427	532.18	0	750
x_{32}	m_{vap}^-	3.05	6.02	0	10^{30}
x_{33}	w_c	12.92	10.19	0	10^{30}
x_{34}	η^-	4.46	4.26	0	10^{30}
x_{35}	R_d	10.2	8.78	0	10^{30}
x_{36}	E^+	131	128.70	0	10^{30}
x_{37}	V^+	2	1.97	0	10^{30}
x_{38}	E^-	0.86	0.86	0	10^{30}
x_{39}	V^-	1.87	1.84	0	10^{30}

NLP variable	model variable	initial value	final value	lower bound	upper bound
x_{40}	V	3.87	3.82	0	10^{30}

Table 7.30 shows the description of the parameters.

Table 7.30 Description of Parameters

Parameter	Description
R	hardware resistance
GCU	hardware resistance
RG	hardware resistance
RHDWR	hardware resistance
F	Faraday's constant
F2	Faraday's constant
F4	Faraday's constant
AREAA	anodic area
AREAC	cathodic area
AREAD	diaphragm area
AREAE	electrolyte area
DTHKNS	diaphragm thickness
PRESS	barometer pressure
DRELCND	diaphragm relative conductivity
MWH2O	molecule weight

Parameter	Description
MWCL2	molecule weight
MWNACL	molecule weight
MWNAOH	molecule weight
MWH2	molecule weight
PAO	vapor pressure of water at anolyte
PA	vapor pressure of anolyte
RCL	chlorine overvoltage resistance
RHOO	resistivity of the anolyte solution
PCO	water pressure of H_2O at catholyte temperature
PC	water pressure at catholyte
RH2	hydrogen overvoltage resistance
D1	anolyte salt mass fraction
D2	catholyte caustic mass fraction
D3	anolyte salt mass fraction

Table 7.31 shows the description of the subroutines.

Table 7.31 Description of Subroutines

Subroutine	Description
RANOLT	calculates the resistivity of the anolyte
CLOVER	calculates anode chlorine overvoltage
H2OVER	calculates cathode hydrogen overvoltage

Subroutine	Description
VAPH2O	calculates the vapor pressure of water
VAPANO	calculates the anolyte vapor pressure
VAPCAT	calculates the catholyte vapor pressure
TBOIL	computes the approximate boiling point of an "M" molal aqueous solution of sodium chloride

Equality constraints

$$x_2 - \frac{x_1 10^3}{AREAA} = 0 \textit{ anodic current density}$$

$$x_3 - \frac{x_1 10^3}{AREAC} = 0 \textit{ cathodic current density}$$

$$scrat = \frac{1.461(1 - \frac{x_4}{100}}{x_4} 100 \textit{ salt/caustic ratio}$$

$$\left[\frac{x_5}{100} + 0.0805195(SCRAT)^2 - 0.295247(SCRAT) - 0.7066872 \right] 100 = 0$$
$$\textit{current efficiency scale factor}$$

$$x_6 - \frac{x_1 10^3 x_5}{x_4 F} = 0 \textit{ sodium input in feed brine}$$

$$\frac{\left[100x_8 - \frac{x_6 1000}{x_7}\right]}{100} = 0 \text{ mass flow rate of } H_2O \text{ in feed brine}$$

$$\frac{x_9 100 - (MW)_{NACL}x_6 - 100x_8}{10} = 0 \text{ feed brine mass flow}$$

$$x_{10} + 2.08333x_2 - 512.292 = 0 \text{ life of anode}$$

$$\left(x_{12} - \frac{x_{11}}{x_{10}}\right)100 = 0 \text{ fractional age of anode}$$

$$x_{13} - x_{11} = 0 \text{ age of nth diaphragm}$$

$$\left[\frac{x_{14}}{100} - \frac{1.25 - x_{12}}{12}\right]100 = 0 \text{ anode blade thickness}$$

$$\left[\frac{x_{15}}{1000} - \frac{1}{12} + \frac{x_{14}}{200}\right]100 = 0 \text{ anode diaphragm gap}$$

$$\left[\frac{x_{16}}{100} - \frac{x_{15}x_1 1000}{100F2}\right]10 = 0 \text{ chlorine gas evolved at anode}$$

$$\left[\frac{x_{17}}{100} - \left(1 - \frac{x_5}{100}\right)\frac{x_1\ 1000}{F2}\right]100 = 0\ H_2O \text{ formed by current inefficiency reaction}$$

$$x_{19} - p_{A0} = 0 \quad \begin{array}{l}\text{vapor pressure of water}\\ \text{at anolyte temperature}\end{array}$$

$$x_{21} - P_A = 0 \quad \begin{array}{l}\text{vapor pressure of water}\\ \text{above anolyte temperature}\end{array}$$

$$\left[x_{22} - \frac{1000x_1(1 + \frac{x_5}{100})x_{21}}{(PRESS - x_{21})F4} \right] 10 = 0$$ *water vaporized with anolyte gas*

$$x_{23}100 - 100x_8 - \left[\frac{x_{17}}{100} - x_{22} \right] (MW)_{H2O} = 0$$ *water in anolyte flowing into diaphragm*

$$\frac{\left[x_{20} - \frac{x_6 - x_{16}}{100} \right] 10^5}{100x_{23}} = 0$$ *salt molality of anolyte*

$$\left[\frac{x_{24}}{10^6} - R_{CL} \right] 10^6 = 0$$ *chlorine overvoltage resistance*

$$\left[\frac{x_{25}}{1000} - R_{HOO} \right] 1000 = 0$$ *resistance of anolyte liquor*

$$\left[\frac{x_{26}}{10^6} - \frac{x_{25}[0.905 + 0.009x_{12}]^{-1.5}x_{15}}{10^6(AREAE)} \right] 10^7 = 0$$ *resistance of anolyte in anolyte-diaphragm gap*

$$x_{28} - \frac{x_{27}\left(1 - \frac{x_4}{100} \right) 100}{x_4} = 0$$ *salt molality of catholyte*

$$x_{30} - P_{CO} = 0$$ *vapor pressure of water at catholyte temperature*

$$\frac{x_{31} - p_C}{10} = 0$$ *vapor pressure of H_2O above catholyte temperature*

$$\left[x_{32} - \frac{1000x_1x_{31}}{F2(PRESS) - x_{31})} \right] 10 = 0$$ *water vaporized with cathode gas*

$$\frac{100x_{33} - 100x_{23} + \left[\frac{1000x_1}{F} + x_{32} \right](MW)_{H2O}}{10} = 0$$ *flow rate of H_2O in cell liqour*

$$\left[x_{27} - \frac{1000x_1 1000}{F100x_{33}} \right] 10 = 0$$ *caustic molality of catholyte*

$$\left[\frac{x_{34}}{10^6} - R_{H2} \right] 10^7 = 0$$ *hydrogene overvoltage resistance*

$$\left[\frac{x_{35}}{10^6} - \frac{x_{25}(DTHKNS)}{1000(DRELCND)(AREAD)} - 2.132x10^{-10}x_{13}^2 + 5.414x10^{-9}x_{13} \right] 10^6 = 0$$

diaphragmresistance

$$Y_{CL2} = \frac{x_{16}}{\frac{100x_1 1000 \left(1 + \frac{x_5}{100} \right)}{F4} + x_{22}}$$

$$\left[\frac{x_{36}}{100} - 1.359 - 2.395210^{-5}(TAR)LOG \left(\frac{(Y_{CL2})(PRESS)}{0.81X_{20}^2X_{21}} \right) \right] 10^4 = 0$$ *thermodynamic reversible emf of anode*

$$\left[\frac{x_{37} - \frac{x_{36}}{100}}{1000x_1} - \frac{x_{26}}{10^6} + \frac{x_{24}}{10^6} + R_{GCU} + RG \right] 10^4 = 0$$ *anode half-cell equivalent resistance*

$$\left\{ x_{38} - 0.828 - 2.3952x10^{-5}[x_{29} + 459.7]LOG\left(\frac{Y_{H2}(PRESS)0.81x_{27}^{2}}{x_{31}}\right)\right\} 10^{3} = 0$$

thermodynamic reversible emf of cathode

$$\left[\frac{x_{39} - x_{38}}{1000x_{1}} - \frac{x_{34}}{10^{6}} + \frac{x_{35}}{10^{6}} + RHDWR \right] 10^{4} = 0$$ *cathode half-cell equivalent resistance*

$$x_{40} - x_{37} - x_{39} = 0$$ *total cell voltage*

$$\left\{ x_{18} - \frac{[\,68.2139x_{40} + 1.52602(D1) - 0.0936978(POWER) + 54.8611\,]\,9}{5} - 32 \right\} \frac{1}{10} = 0$$

anolyte temperature

$$x_{29} - \frac{[\,-3.26827x_{12} - 4.81807D1 + 339.217x_{40} - 0.466131(POWER) - 37.6628D2\,]\,9}{5}$$

$$+ \frac{[\,26.6881D3 + 95.4899\,]\,9}{5} - x_{18} = 0$$ *catholyte temperature*

$x_{11} = 0$ *force evalation of constraints at initial day of anode life*

Inequality Constraints

$x_{29} - 5.4 - x_{18} \geq 0$ *make sure anolyte temperature is at least 5.4 degrees less than the catholyte temperature*

$TB - x_{29} \geq 0$ *make sure catholyte temperature is not greater than its boiling point*

Objective Function

The objective function includes 3 terms:

(1) gross income from product sales

(2) cost of electricity

(3) cost of feed materials to the cell

$$OF = -0.06(POWER)5.x10^{-4} + \frac{145(5.x10^{-4})x_{16}(MW)_{CL2}}{100} + 175x_{27}x_{33}100(MW)_{NAOH}$$
$$(5.x10^{-7}) - 20.7x_{7}x_{8}100(MW)_{NACL}(5.x10^{-7})$$

Its optimal value was 10.077.

In [99], the optimal value of the objective function is 10.08.

7.6 Example 8 : Unsteady-State Diffusion in 2 Space Dimensions - Parabolic PDE in 3 Independent Variables

The initial-boundary-value problem described in [138]

$$\begin{aligned} u_t - (u_{xx} + u_{yy}) &= -e^{-t}xy \qquad 0<x<1,\ 0<y<1,\ t>0 \\ u(x,y,0) &= xy \\ u(0,y,t) &= 0 \\ u(x,0,t) &= 0 \\ u(1,y,t) &= ye^{-t} \\ u(x,1,t) &= xe^{-t} \\ u(0,0,0) &= 0 \\ u(1,1,t) &= e^{-t} \end{aligned} \tag{46}$$

has the exact solution xye^{-t}. Backward-in-time finite difference method was used to transform the *PDE* in 3 independent variables into a *PDE* in 2 independent variables. The time-step was chosen as $h = 0.1$ and the integration domain was [0,0.5]x[0,1]x[0,1]. The knot points in *x*-interval are 0.2,0.4,0.6,0.8 and in the *y*-interval are 0.2,0.4,0.6,0.8. The residual error of the discretized differential equations was evaluated at (0.125,0.125),(0.125,0.250),...,(0.875,0.875). There were 25 2-dimensional finite elements. The discretized differential equation was evaluated in the interior of the *x-y* integration domain, i.e. at internal collocation points. The finite differencing led to 5 dependent state variables, $u_2(x,y,0.1)$, $u_3(x,y,0.2)$, $u_4(x,y,0.3)$, $u_5(x,y,0.4)$ and $u_6(x,y,0.5)$.

The discretized differential equations in element (k,l) k=1,2,...,5 l=1,2,...,5 and at collocation points i=1,2,3 j=1,2,3 are

$$
\begin{aligned}
&\frac{u_{2ij}^{kl} - xy}{0.1} - \frac{1}{\Delta x_k^2}\sum_{n=1}^{3} u_{2nj}^{kl}\frac{d^2 l_n(x_i')}{dx'^2} - \frac{1}{\Delta y_l^2}\sum_{n=1}^{3} u_{2in}^{kl}\frac{d^2 l_n(y_j')}{dy'^2} + e^{-0.1}xy\\
&\frac{u_{3ij}^{kl} - u_{2ij}^{kl}}{0.1} - \frac{1}{\Delta x_k^2}\sum_{n=1}^{3} u_{3ij}^{kl}\frac{d^2 l_n(x_i')}{dx'^2} - \frac{1}{\Delta y_l^2}\sum_{n=1}^{3} u_{3in}^{kl}\frac{d^2 l_n(y_j')}{dy'^2} + e^{-0.2}xy\\
&\frac{u_{4ij}^{kl} - u_{3ij}^{kl}}{0.1} - \frac{1}{\Delta x_k^2}\sum_{n=1}^{3} u_{4nj}^{kl}\frac{d^2 l_n(x_i')}{dx'^2} - \frac{1}{\Delta y_l^2}\sum_{n=1}^{3} u_{4in}^{kl}\frac{d^2 l_n(y_j')}{dy'^2} + e^{-0.3}xy\\
&\frac{u_{5ij}^{kl} - u_{4ij}^{kl}}{0.1} - \frac{1}{\Delta x_k^2}\sum_{n=1}^{3} u_{5nj}^{kl}\frac{d^2 l_n(x_i')}{dx'^2} - \frac{1}{\Delta y_l^2}\sum_{n=1}^{3} u_{5in}^{kl}\frac{d^2 l_n(y_j')}{dy'^2} + e^{-0.4}xy \quad (47)\\
&\frac{u_{6ij}^{kl} - u_{5ij}^{kl}}{0.1} - \frac{1}{\Delta x_k^2}\sum_{n=1}^{3} u_{6nj}^{kl}\frac{d^2 l_n(x_i')}{dx'^2} - \frac{1}{\Delta y_l^2}\sum_{n=1}^{3} u_{6in}^{kl}\frac{d^2 l_n(y_j')}{dy'^2} + e^{-0.5}xy
\end{aligned}
$$

$$
\begin{aligned}
&k = 1,2,\ldots,4 \; l = 1,2,\ldots,4 \; i = 2,3 \; j = 2,3\\
&k = 1,2,\ldots,4 \; l = 5 \; i = 2,3 \; j = 2\\
&k = 5 \; l = 1,2,\ldots,4 \; i = 2 \; j = 2,3\\
&k = 5 \; l = 5 \; i = 2 \; j = 2
\end{aligned}
$$

The differential equations were not collocated at the boundary of the *x-y* integration domain, where initial-boundary conditions were given. The discretized initial-boundary conditions are:

$$
\begin{aligned}
&u_{2ij}^{kl} = 0 \; k = 1 \; l = 1 \; i = 1,2,3 \; j = 1\\
&u_{2ij}^{kl} = 0 \; k = 2,\ldots,5 \; l = 1 \; i = 2,3 \; j = 1\\
&u_{2ij}^{kl} = 0 \; k = 1 \; l = 1,2,\ldots,5 \; i = 1 \; j = 2,3\\
&u_{2ij}^{kl} = xe^{-0.1} \; k = 1,2,\ldots,4 \; l = 5 \; i = 2,3 \; j = 3\\
&u_{2ij}^{kl} = xe^{-0.1} \; k = 5 \; l = 5 \; i = 2 \; j = 3\\
&u_{2ij}^{kl} = ye^{-0.1} \; k = 5 \; l = 1,2,\ldots,4 \; i = 3 \; j = 2,3\\
&u_{2ij}^{kl} = ye^{-0.1} \; k = 5 \; l = 5 \; i = 3 \; j = 2\\
&u_{2ij}^{kl} = e^{-0.1} \; k = 5 \; l = 5 \; i = 3 \; j = 3
\end{aligned}
\qquad (48)
$$

$$\begin{array}{l} u_{3ij}^{kl} = 0 \; k = 1 \; l = 1 \; i = 1,2,3 \; j = 1 \\ u_{3ij}^{kl} = 0 \; k = 2,\ldots,5 \; l = 1 \; i = 2,3 \; j = 1 \\ u_{3ij}^{kl} = 0 \; k = 1 \; l = 1,2,\ldots,5 \; i = 1 \; j = 2,3 \\ u_{3ij}^{kl} = xe^{-0.2} \; k = 1,2,\ldots,4 \; l = 5 \; i = 2,3 \; j = 3 \\ u_{3ij}^{kl} = xe^{-0.2} \; k = 5 \; l = 5 \; i = 2 \; j = 3 \\ u_{3ij}^{kl} = ye^{-0.2} \; k = 5 \; l = 1,2,\ldots,4 \; i = 3 \; j = 2,3 \\ u_{3ij}^{kl} = ye^{-0.2} \; k = 5 \; l = 5 \; i = 3 \; j = 2 \\ u_{3ij}^{kl} = e^{-0.2} \; k = 5 \; l = 5 \; i = 3 \; j = 3 \end{array} \tag{49}$$

$$\begin{array}{l} u_{4ij}^{kl} = 0 \; k = 1 \; l = 1 \; i = 1,2,3 \; j = 1 \\ u_{4ij}^{kl} = 0 \; k = 2,\ldots,5 \; l = 1 \; i = 2,3 \; j = 1 \\ u_{4ij}^{kl} = 0 \; k = 1 \; l = 1,2,\ldots,5 \; i = 1 \; j = 2,3 \\ u_{4ij}^{kl} = xe^{-0.3} \; k = 1,2,\ldots,4 \; l = 5 \; i = 2,3 \; j = 3 \\ u_{4ij}^{kl} = xe^{-0.3} \; k = 5 \; l = 5 \; i = 2 \; j = 3 \\ u_{4ij}^{kl} = ye^{-0.3} \; k = 5 \; l = 1,2,\ldots,4 \; i = 3 \; j = 2,3 \\ u_{4ij}^{kl} = ye^{-0.3} \; k = 5 \; l = 5 \; i = 3 \; j = 2 \\ u_{4ij}^{kl} = e^{-0.3} \; k = 5 \; l = 5 \; i = 3 \; j = 3 \end{array} \tag{50}$$

$$\begin{array}{l} u_{5ij}^{kl} = 0 \; k = 1 \; l = 1 \; i = 1,2,3 \; j = 1 \\ u_{5ij}^{kl} = 0 \; k = 2,\ldots,5 \; l = 1 \; i = 2,3 \; j = 1 \\ u_{5ij}^{kl} = 0 \; k = 1 \; l = 1,2,\ldots,5 \; i = 1 \; j = 2,3 \\ u_{5ij}^{kl} = xe^{-0.4} \; k = 1,2,\ldots,4 \; l = 5 \; i = 2,3 \; j = 3 \\ u_{5ij}^{kl} = xe^{-0.4} \; k = 5 \; l = 5 \; i = 2 \; j = 3 \\ u_{5ij}^{kl} = ye^{-0.4} \; k = 5 \; l = 1,2,\ldots,4 \; i = 3 \; j = 2,3 \\ u_{5ij}^{kl} = ye^{-0.4} \; k = 5 \; l = 5 \; i = 3 \; j = 2 \\ u_{5ij}^{kl} = e^{-0.4} \; k = 5 \; l = 5 \; i = 3 \; j = 3 \end{array} \tag{51}$$

$$\begin{aligned}
u_{6ij}^{kl} &= 0 \quad k = 1 \quad l = 1 \quad i = 1,2,3 \quad j = 1\\
u_{6ij}^{kl} &= 0 \quad k = 2,\ldots,5 \quad l = 1 \quad i = 2,3 \quad j = 1\\
u_{6ij}^{kl} &= 0 \quad k = 1 \quad l = 1,2,\ldots,5 \quad i = 1 \quad j = 2,3\\
u_{6ij}^{kl} &= xe^{-0.5} \quad k = 1,2,\ldots,4 \quad l = 5 \quad i = 2,3 \quad j = 3\\
u_{6ij}^{kl} &= xe^{-0.5} \quad k = 5 \quad l = 5 \quad i = 2 \quad j = 3\\
u_{6ij}^{kl} &= ye^{-0.5} \quad k = 5 \quad l = 1,2,\ldots,4 \quad i = 3 \quad j = 2,3\\
u_{6ij}^{kl} &= ye^{-0.5} \quad k = 5 \quad l = 5 \quad i = 3 \quad j = 2\\
u_{6ij}^{kl} &= e^{-0.5} \quad k = 5 \quad l = 5 \quad i = 3 \quad j = 3
\end{aligned} \tag{52}$$

The problem to be solved has been transcribed as shown below

```
3 2 5 5 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
*10 NICP NEEDED WITH ONE SUBINTERVAL
* THIS IS A COMMENT
ocfepdesqppddzapdebdt05p420.dat
0.125 0.125 0.5 0.5
1 0 0 0 0 0 0 5 40 7 0 1
4 4 4 4 4 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 3 3 3 3 3 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 9
0 0 0 0 1 1 1 1 1 1 1 1
```

```
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
'U2' 'U3' 'U4' 'U5' 'U6'
'X' 'Y'
0 0 0 0
0. 1. 0. 1. 0. 0. 0. 0.
0.2 0.4 0.6 0.8
0.2 0.4 0.6 0.8
pi=3.141592
(U2(X,Y)-X*Y)/0.1-D2U2(X,Y)/DX2-D2U2(X,Y)/DY2+exp(-0.1)*X*Y
(U3(X,Y)-U2(X,Y))/0.1-D2U3(X,Y)/DX2-D2U3(X,Y)/DY2+exp(-0.2)*X*Y
(U4(X,Y)-U3(X,Y))/0.1-D2U4(X,Y)/DX2-D2U4(X,Y)/DY2+exp(-0.3)*X*Y
(U5(X,Y)-U4(X,Y))/0.1-D2U5(X,Y)/DX2-D2U5(X,Y)/DY2+exp(-0.4)*X*Y
(U6(X,Y)-U5(X,Y))/0.1-D2U6(X,Y)/DX2-D2U6(X,Y)/DY2+exp(-0.5)*X*Y
U2(X,Y) AT X=0 AT Y=0
U2(X,Y) AT X=0 AT Y=1
U2(X,Y) AT X=1 AT Y=0
U2(X,Y)-exp(-0.1) AT X=1 AT Y=1
U2(X,Y) AT X=0
U2(X,Y)-Y*exp(-0.1) AT X=1
U2(X,Y) AT Y=0
U2(X,Y)-X*exp(-0.1) AT Y=1
U3(X,Y) AT X=0 AT Y=0
U3(X,Y) AT X=0 AT Y=1
U3(X,Y) AT X=1 AT Y=0
U3(X,Y)-exp(-0.2) AT X=1 AT Y=1
U3(X,Y) AT X=0
U3(X,Y)-Y*exp(-0.2) AT X=1
U3(X,Y) AT Y=0
U3(X,Y)-X*exp(-0.2) AT Y=1
```

```
U4(X,Y) AT X=0 AT Y=0
U4(X,Y) AT X=0 AT Y=1
U4(X,Y) AT X=1 AT Y=0
U4(X,Y)-exp(-0.3) AT X=1 AT Y=1
U4(X,Y) AT X=0
U4(X,Y)-Y*exp(-0.3) AT X=1
U4(X,Y) AT Y=0
U4(X,Y)-X*exp(-0.3) AT Y=1
U5(X,Y) AT X=0 AT Y=0
U5(X,Y) AT X=0 AT Y=1
U5(X,Y) AT X=1 AT Y=0
U5(X,Y)-exp(-0.4) AT X=1 AT Y=1
U5(X,Y) AT X=0
U5(X,Y)-Y*exp(-0.4) AT X=1
U5(X,Y) AT Y=0
U5(X,Y)-X*exp(-0.4) AT Y=1
U6(X,Y) AT X=0 AT Y=0
U6(X,Y) AT X=0 AT Y=1
U6(X,Y) AT X=1 AT Y=0
U6(X,Y)-exp(-0.5) AT X=1 AT Y=1
U6(X,Y) AT X=0
U6(X,Y)-Y*exp(-0.5) AT X=1
U6(X,Y) AT Y=0
U6(X,Y)-X*exp(-0.5) AT Y=1
U6(X,Y)
```

Table 7.32 shows the starting values and bounds for the state variables.

Table 7.32 Starting Values

Function/ variable	lower bound	upper bound	starting point

u_2	0	1	0
u_3	0	1	0
u_4	0	1	0
u_5	0	1	0
u_6	0	1	0

The state variables are unbounded, the bounds are only used for scaling. The numbering of u_6 is shown in Figure 7.34.

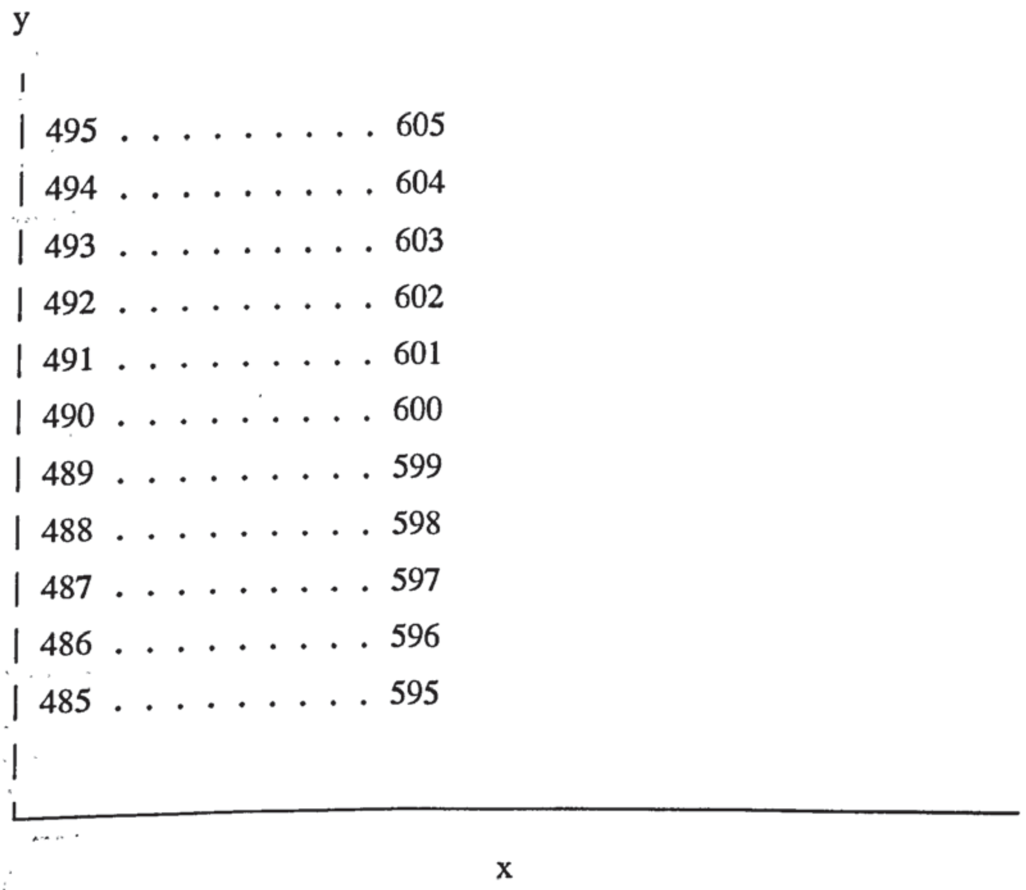

Fig. 7.34 NLP Variables for u_6

Table 7.33 shows the number of *NLP* variables derived from the state variables.

Table 7.33 Relationship between Continuous and *NLP* Variables

Function/variable	NLP dimension
u_2	121
u_3	121
u_4	121
u_5	121
u_6	121

The problem was solved using *VF13AD* on *SUN SPARC 2000* workstation. The numerical solution closely approximates the exact solution given in [138].

Table 7.34 shows the parameters of the collocation method applied and the *NLP* solution statistics.

Table 7.34 Collocation Parameters and *NLP* Statistics

N	5
M	5
N_x	5
N_y	5
m	605
n	605
niter	2
acc	0.0001
CPU	00:25:27

err	0.01
averr	0.0007

where			
	N	=	number of internal collocation points in x-interval
	M	=	number of internal collocation points in y-interval
	N_x	=	number of finite elements in x-interval
	N_y	=	number of finite elements in y-interval
	m	=	number of *NLP* constraints
	n	=	number of *NLP* variables
	niter	=	number of *NLP* iterations
	acc	=	accuracy of *NLP* solution
	CPU	=	*CPU* time
	err	=	the maximum point-wise residual error
	averr	=	the average point-wise residual error

Table 7.35 shows the exact solution at t=0.5.

Table 7.35 Exact Solution

x/y	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.
.0	0	0	0	0	0	0	0	0	0	0	0
.1	0	.01	.01	.02	.02	.03	.04	.04	.05	.05	.06
.2	0	.01	.02	.04	.05	.06	.07	.08	.10	.11	.12
.3	0	.02	.04	.05	.07	.09	.11	.13	.15	.16	.18
.4	0	.02	.05	.07	.10	.12	.15	.17	.19	.22	.24
.5	0	.03	.06	.09	.12	.15	.18	.21	.24	.27	.30
.6	0	.04	.07	.11	.15	.18	.22	.25	.29	.33	.36

.7	0	.04	.08	.13	.17	.21	.25	.30	.34	.38	.42
.8	0	.05	.10	.15	.19	.24	.29	.34	.39	.44	.49
.9	0	.05	.11	.16	.22	.27	.33	.38	.44	.49	.55
1.	0	.06	.12	.18	.24	.30	.36	.42	.49	.55	.61

Table 7.36 shows the numerical solution at t=0.5.

Table 7.36 Numerical Solution

x/y	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.
.0	0	0	0	0	0	0	0	0	0	0	0
.1	0	.01	.01	.02	.03	.03	.04	.04	.05	.06	.06
.2	0	.01	.03	.04	.05	.06	.08	.09	.10	.11	.12
.3	0	.02	.04	.06	.08	.09	.11	.13	.15	.17	.18
.4	0	.03	.05	.08	.10	.13	.15	.18	.20	.22	.24
.5	0	.03	.06	.09	.13	.16	.19	.22	.25	.28	.30
.6	0	.04	.08	.11	.15	.19	.23	.26	.30	.33	.36
.7	0	.04	.09	.13	.18	.22	.26	.31	.35	.39	.42
.8	0	.05	.10	.15	.20	.25	.30	.35	.40	.44	.49
.9	0	.06	.11	.17	.22	.28	.33	.39	.44	.49	.55
1.	0	.06	.12	.18	.24	.30	.36	.42	.49	.55	.61

Figure 7.35 shows the values of $u(t,x,y)$ at $t = 0.5$.

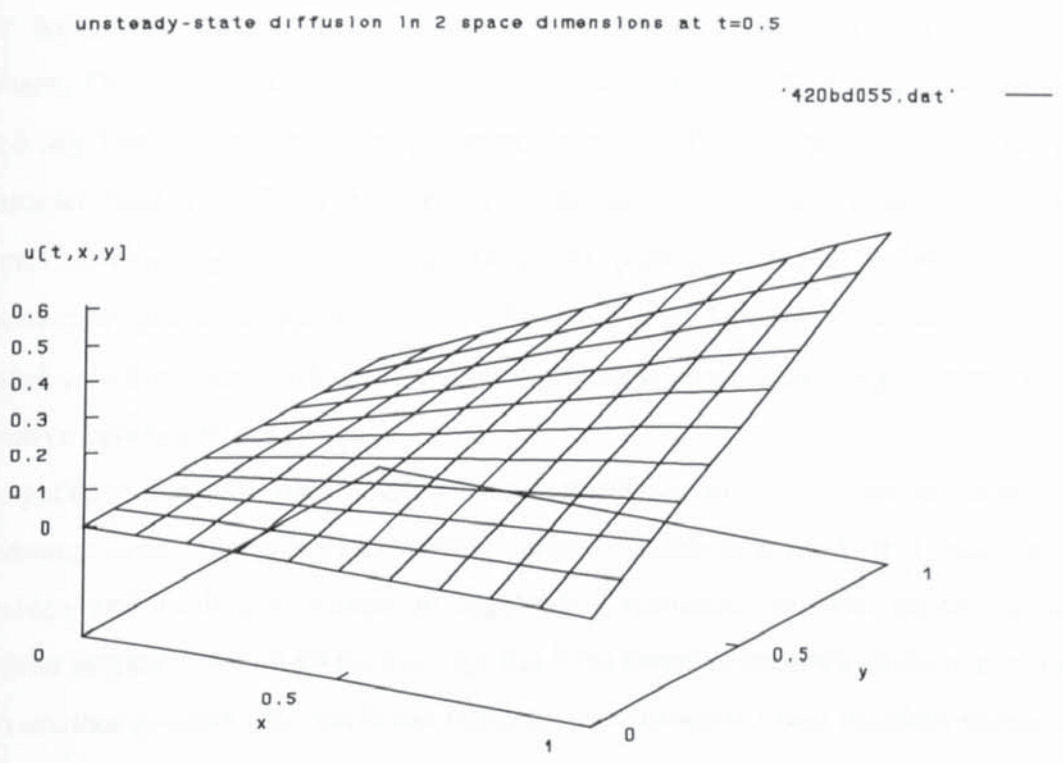

Fig. 7.35 u=u(t,x,y) at t=0.5

This example shows that the method and its computer implementation is capable of solving partial differential equations in 3 independent variables. The user has to transform the 3-dimensional problem into a 2-dimensional one using finite difference method. The finite difference method cannot be applied if second derivatives of each state variable appear in the model and there are no initial-boundary conditions for any of the state variables, because the finite difference formula for $u_{xx}(x_n,y,z)$ requires the functions $u_{n-1}(y,z)$, $u_n(y,z)$ and $u_{n+1}(y,z)$ in the differential equation. For $u_0(y,z)$ at $x = 0$ the finite difference formula would require the function $u_{-1}(y,z)$ which would lie outside of the integration domain. Similarly in the differential equation for $u_n(y,z)$ at $x = x_n$ where x_n is the boundary of the x-interval, the finite difference formula would require the function $u_{n+1}(y,z)$ which would lie outside of the integration domain.

7.7 Conclusions

The 2 NLP packages proved to be very reliable. Computer run experience did not show either of them to be consistently superior to the other, but to be equally good, 2 of the best packages currently available. Computer run experience has also shown that the residual error is smaller if the control function is a piecewise constant function over the finite elements and if there is one internal collocation point in a finite element. The package has solved both dynamic simulation and optimization problems involving lumped or distributed parameter systems. The package solved a lumped parameter system involving parameter estimation, time delay and equality constraint containing time-dependent state variables. The package has also solved a realistic nonlinear programming problem. The package has solved models involving bounds on control variables. The package has solved problems containing integral terms in the objective functional.

The package has not been used to solve problems involving general inequality constraints containing state and control variables dependent on 1 or 2 independent variables and problems containing algebraic constraints, realistic problems with irregular domains. Although the package has been tested to handle variables that come from another process unit, but it has not been used to solve a real problem involving disturbances or variables coming from another process unit.

Chapter 8
Conclusions and Recommendations

8.1 Conclusions

Computer runs have established that orthogonal collocation on finite elements is a viable method to convert, into a nonlinear programming problem, models containing algebraic equations and inequalities, ordinary differential equations, partial differential equations containing state or control functions having 1 or 2 independent variables, equations and inequalities containing state and control functions dependent on 1 or 2 independent variables.

The packages *GRG2* and *SQP* have solved the resulting large nonlinear programming problems without any difficulty.

8.2 Contributions of this Research

1. Although techniques currently exist to solve optimization problems containing *PDEs* with 2 independent variables, this work extends the class of problems that can be solved by also making possible the handling of equalities and inequalities containing state and control functions dependent on 1 or 2 independent variables and algebraic equations and inequalities.
2. Control vector parameterization and 2 *NLP* packages have been integrated into one optimization package.
3. For solving a differential-algebraic optimization problem by collocation, the *NLP* model is generated automatically without any user intervention. From model to solution, the intermediate stages are completely automated.
4. Residual error can be evaluated at a priori selected grid points.
5. The state and control functions can be tabulated.
6. The package has a commercial potential. Subject to royalties being paid to the University of Texas, USA, and to the Harwell Physical Laboratory, England, it should be made available both to the industry and, through a computer network, to Universities.
7. As, from an engineering standpoint, this work is quite general, in the instances where the models conform to a model solvable by this work, it should be of benefit not only to chemical engineering, but to all other engineering disciplines as well as to other disciplines, to science, to economics, to ecology,

to mathematics and to medical science.

8.3 Recommendations for Future Work

1. The package could be extended to handle more than 2 independent variables, say, *t,x,y*, where every state or control function is a function of only 2 of the independent variables, eg., the model could contain state functions $c_1(t,x)$, $c_2(t,y)$, $c_3(x,y)$, but in every differential equation there may appear only state and control functions and auxiliary variables depending on the same 2 independent variables. For example, if c_1 appears in a differential equation, c_2 and c_3 may not appear in the same equation.
2. The package could be extended to handle split boundary conditions, such as, when, e.g., between 0 and α a boundary condition is different from the one existing between α and t_f.
3. The package could be extended to handle those of the time delays where the time delay is not only a function of time but also that of the state variables.
4. The package could be used for on-line simultaneous control and optimization, if some type of feedback mechanism is included within the overall strategy.
5. A variant of the *OCFE* package could be implemented using *Hermitian* interpolates instead of Lagrangian interpolates.
6. A variant of the *OCFE* package could be implemented using *splines* interpolates instead of Lagrangian interpolates.
7. The package could be extended to handle modules where the validity of a different equation and the associated state variables is extended only over a part of the integration domain, e.g. there could be more than 1 phases where different laws would apply.
8. The package could be extended to allow state variables, control variables and parameters dependent on independent variables and the associated differential equations to have their own dedicated independent variables. Independent variables could have their own integration domain. This extension is an alternative to extension described in paragraph 7.
9. The *OCFE* program could be enhanced to check automatically whether the model equations contained in OCFEINP.DAT are independent and whether the initial and boundary conditions are consistent and sufficient.

10. A version of *OCFE* could be developed, which would result in the state function having continuous first derivatives. This could be achieved either by reactivating the code that generated equations making the derivatives equal at finite element boundaries, or by using cubic *Hermite* interpolation polynomials having continuous first derivatives or using cubic splines having continuous second derivatives, or by introducing a new state variable and add a differential equation to make this variable equal to the first derivative of an original state variable. Since the solution function of a state variable is a continuous function, this additional differential equation would make the derivative function of the original state variable continuous.

Nomenclature

In this section is a list of Nomenclature which defines many of the quantities and formulations used in this text. Also presented is a list of Acronyms.

Greek characters

Ω	integration domain
$\partial\Omega$	boundary of Ω
λ	vector of adjoint profile coefficients
δ	Dirac delta
μ,λ	coefficient vectors
α	scalar parameter of search direction
Ω_1	integration domain of the first independent variable
Ω_2	integration domain of the second independent variable
Ω_3	integration domain of (t,x)
$\partial\Omega_1$	boundary of Ω_1
$\partial\Omega_2$	boundary of Ω_2
$\partial\Omega_3$	boundary of Ω_3
τ	time delay

Roman characters

I	objective functional
G,G1	function at t_f and/or x_f, term in the objective functional
G2,G3,G4,F	integrand of the integral performance index
A,B,Q,R	matrices in linear/quadratic optimal control problem
$\mathbf{x}$	vector of state functions
$\mathbf{x}$	vector of real-valued variables in an NLP problem
x_i	an element of the vector $\mathbf{x}$ in an NLP problem
$\mathbf{u}(t),\mathbf{u}_2(t)$	control function vectors
$\mathbf{u}_3(x)$	control function vector
$\mathbf{u}_1$	a vector of real-valued control variables
$\mathbf{u}_4(t,x)$	control function vector
V	Liapunov function $[R^n \text{ -> } C^1]$
$\mathbf{f}$	a vector of right-hand side functions in a system of first-order quasilinear differential equations

$\mathbf{f}$	a system of differential equations
f	in an NLP real-valued function of $\mathbf{x}$ [R^n -> R^1]
$\mathbf{f}_1,\mathbf{f}_2$	a system of ordinary differential equations [R^1 -> C^2]
$\mathbf{f}_3$	a system of partial differential equations [R^2 -> C^2]
N_x	number of finite elements along the first independent variable
N_y	number of finite elements along the second independent variable
N	number of internal collocation points along the first independent variable in a finite element
N_k	number of internal collocation points along the first independent variable in the k-th finite element
M	number of internal collocation points along the second independent variable in a finite element
M_l	number of internal collocation points along the second independent variable in the l-th finite element
c	state variable
w	control variable
p	auxiliary variable
u,v	independent variables transformed into the interval [0,1]
$\mathbf{h}$	equality constraint vector in an NLP problem [R^n -> R^m]
∇	gradient vector
l	Lagrangian
f	objective function
M	tangent plane
t_f,x_f	final value of integration
t_0,x_0	initial value of integration
t,x,y	independent variables
R^1	1-dimensional Euclidean space
R^n	n-dimensional Euclidean space
C	function space of continuous functions over the domain of integration
C^1	function space of functions having continuous first derivatives over the domain of integration
H	the Hamiltonian function

F,f	function of **x** and **u** [R^{n+m} -> R^1]
a,b	coefficients
H	the Heaviside function
f_{ij}	real-valued scalar function
g_{ij}	real-valued scalar function
h_{ij}	real-valued scalar function
Φ_j	real-valued scalar function
w_j	linear combination coefficient
R	residual
f	differential equation
$l_n(u_i)$	the n-th Lagrange interpolation polynomial evaluated at collocation point u_i
$\Delta t_k, \Delta x_k$	length of the k-th finite element along the first independent variable
Δy_l	length of the l-th finite element along the second independent variable
L	the matrix of second partial derivatives of the Lagrangian
F	the Hessian of f
$\mathbf{H}_i$	the Hessian of the i-th equality constraint
g	inequality constraints vector in an NLP problem
$\mathbf{G}_i$	the Hessian of the i-th inequality constraint
B,C	matrix of the gradient vectors of equality constraints
d	search direction vector
a_j	j-th lower bound
b_j	j-th upper bound
∇^2	Hessian of f
q	real-valued scalar function
$\mathbf{c}_1$	a vector of real-valued state variables
$\mathbf{c}_2(t)$	state function vector [R^1 -> C]
$\mathbf{c}_3(x)$	state function vector [R^1 -> C]
$\mathbf{c}_4(t,x)$	state function vector [R^2 -> C]
$\mathbf{p}_1$	a vector of real-valued parameters variables
$\mathbf{p}_2(t)$	parameter function vector
$\mathbf{p}_3(x)$	parameter function vector

$\mathbf{p}_4(t,x)$	parameter function vector
$\mathbf{c}_0$	initial value vector of state variables
$\bar{c}_1,\bar{c}_2,\bar{c}_3,\bar{c}_4$	boundary value vector of state variables
$\mathbf{h},\mathbf{g}$	real-valued constraint function vectors of the independent variables
$\mathbf{u}_1^L$	lower bound of $\mathbf{u}_1$
$\mathbf{u}_1^U$	upper bound of $\mathbf{u}_1$
$\mathbf{u}_2^L(t)$	lower bound of $\mathbf{u}_2(t)$
$\mathbf{u}_2^U(t)$	upper bound of $\mathbf{u}_2(t)$
$\mathbf{u}_3^L(x)$	lower bound of $\mathbf{u}_3(x)$
$\mathbf{u}_3^U(x)$	upper bound of $\mathbf{u}_3(x)$
$\mathbf{u}_4^L(x,y)$	lower bound of $\mathbf{u}_4(x,y)$
$\mathbf{u}_4^U(x,y)$	upper bound of $\mathbf{u}_4(x,y)$
LB_1,UB_1	left and right endpoints of the first independent variable
LB_2,UB_2	left and right endpoints of the second independent variable
a_i	i-th state variable or its measurements
lb	lower bound
ub	upper bound
g_i	real-valued constraint function in an NLP problem $[R^n \rightarrow R^1]$
Superscripts	
$*$	upper bound
n	dimensionality of an Euclidean space
1	indicates continuity of first derivatives in a function space
k	iteration number
k	k-th finite element
l	l-th finite element
0	indicates initial estimate in an NLP problem
L	lower bound
U	upper bound
Subscripts	
0	indicates that a state function be evaluated at the initial value of integration
$*$	lower bound

f	indicates that a state function be evaluated at the final value of integration
i	i-th component of a vector
ij	ij-th element of a matrix
i	i-th collocation point
j	j-th collocation point

Major mathematical formulations and acronyms

NLP	nonlinear programming problem
GRG	generalized reduced gradient method
GRG2	package received from Professor Leon Lasdon from the University of Texas
SQP	successive quadratic programming
VF13AD	package received from the Harwell Physical Laboratory
CSTR	continuous stirred tank reactor
EVOP	evolutionary optimization
PDE	partial differential equation
ODE	ordinary differential equation
QP	quadratic programming
QDMC	quadratic dynamic matrix control
DAOP	Differential-Algebraic Optimization Problem
BVP	Boundary Value Problem
MWR	Method of Weighted Residuals

References

1. Pearson J. D., "Approximation Methods in Optimal Control", J. Electronic Control, Vol. 13, p. 453, (1962)
2. Burghart J. H., "A Technique for Suboptimal Feedback Control of Nonlinear Systems", IEEE Trans. on Autom. Control AC-14, p. 530, (1963)
3. Weber A. P. and Lapidus L., "Suboptimal Control of Nonlinear Systems", AIChE Journal Vol. 17 pp. 641-658, (1971)
4. Durbeck R. C., "An Approximation Technique for Suboptimal Control", IEEE Trans. Autom. Control AC-10, p. 144, (1965)
5. Garrard W. L., "An Approach to Sub-optimal Fedback Control of Non-linear Systems", Mc Clamrock N. H. and Clark L. G., Int. J. of Control, Vol. 5., pp. 425-435, (1967)
6. Bukreev V. Z., Automatika i Telmekhanika, Vol. 11, p. 5, (1968)
7. Luus R. and Lapidus L., Optimal Control of Engineering Processes, Blaisdell, (1967)
8. Koepecke R. and Lapidus L., "Dynamic control of chemical engineering processes using a method of Lyapunov", Chem. Eng. Sci., Vol. 16, pp. 252-266, (1961)
9. Paradis W. O. and Perlmutter D. D., "Part 1: Optimality and Computational Feasibility in Transient Control. Part 2: Feedback Control of a Distributed Parameter process.", AIChE Journal, Vol. 12, pp. 876-890, (1966).
10. Morshedi A. M., "Universal Dynamic Matrix Control", Seminar 6, paper no. 2, Chemical Process Control Conf. 3, Asilomar, California, (1986).
11. Bryson A. E. and Ho. Y. C., Applied Optimal Control, Ginn/Blaisdel, New York, (1968).
12. Miele A., "Gradient Algorithms for the Optimization of Dynamic Systems" in Control and Dynamic Systems, Leondes C. T., [ed.], Academic Press (1980).
13. Atham M. and Falb P. L., Optimal Control, New York, Mc Graw Hill, (1966).
14. Noton A. R. M., Modern Control Engineering, New York, Pergamon (1972).
15. Irving M. R., Boland F. M. and Nicholson H., "Optimal control of the argon-oxygen decarbonizing steelmaking process", Proc. Institution of Electrical Engineers, Vol. 126, pp. 198-203, (1979).

16. Edwards J. B. and Eren H., Proc. "Controlling electric traction drives for minimum energy wastage", Institution of Electrical Engineers, Vol. 126, pp. 254-260, (1979).

17. Jones D. I. and Finch J. W., Proceedings Institution of Electrical Engineers, Vol. 130, p. 175, (1983).

18. Kelley H. J., "Optimization Techniques", [ed.] Leitmann G., London, Academic Press, (1962).

19. Gibson J. A. and Lowinger J. F., "A predictive Min-H method to improve convergence to optimal solutions", International Journal of Control, Vol. 19, pp. 575-592, (1974).

20. Kumar V., "A control averaging technique for solving a class of simple optimal control problems", International Journal of Control, Vol. 23, pp. 361-380, (1976).

21. Quintana V. H. and Davison E. J., Proceedings Joint Automatic Control Conf., p. 43, (1970).

22. Ray W. H., Advanced Process Control, Mc Graw-Hill, New-York, (1981).

23. Ray W. H. and Szekely J., Process Optimization, John Wiley & Sons, (1973).

24. Crowe C. M. et al., Chemical Plant Simulation, Prentice Hall, (1971).

25. Paynter J. D., Dranoff J. S. and Bankoff S. G., "Application of a Suboptimal Design Method to a Distributed-Parameter Reactor Problem", Ind. Eng. Chem. Proc. Des., Vol. 9, pp. 303-309, (1970).

26. Denn M. M., "Optimal Boundary Control for a Non-linear Distributed System", Int. J. Control, Vol. 4, pp. 167-178, (1966).

27. Jackson R., Trans. Inst. Chem. Engn., Vol. 45, T 160-168, (1967).

28. Chang K. S. and Bankoff S. G., "Optimal Control of Tubular Reactors", AIChE J. Vol. 15, pp. 410-414, (1969).

29. Bertran D. R. and Chang K. S., "Optimal Feedforward Control of Concurrent Tubular Reactors", AIChE J. Vol. 16, pp. 897-902, (1970).

30. Ogunye A. F. and Ray W. H., "Optimal Control Policies for Tubular Reactors Experiencing Catalyst Decay : 2. Multiple Bed", AIChE J. Vol. 17, pp. 365-371, (1971).

31. Ogunye A. F. and Ray W. H., "Optimization of a Vinyl Chloride Monomer

Reactor", I and EC Process Des., Vol. 9, pp. 619-624, (1970).

32. Ogunye A. F. and Ray W. H., "Optimization of Cyclic Tubular Reactors with Catalyst Delay", I & EC Proc. Des. Dev., Vol. 10, pp. 410-416, (1971).

33. Hasdorff L., Gradient Optimization and Nonlinear Control, Wiley-International, New York, (1976).

34. Dixon L. C. W. and Szego G. P. [eds.], Numerical Optimization of Dynamic Systems, North Holland, Amsterdam, (1980).

35. Lasdon L.S., Mitter S. K. and Warren A. D., "The conjugate gradient method for optimal control problems", IEEE Trans. of Automatic Control Vol. AC-12, 2, p. 132, (1967).

36. Jones D. I. and Finch J. W., "Comparison of Optimization Algorithms", Int. J. Cont., Vol. 40, pp. 747-761, (1984).

37. Padmanabhan L. and Bankoff S. G., Proc. Joint Autom. Control Conf., (1969).

38. Bachman G. and Narici L., Functional Analysis, Academic Press, 1966

39. Pagurek B. and Woodside C. M., Automatica, Vol. 4, p. 337, (1968).

40. Kopp R. E. and Mayer H. G., Advanced Control Systems, Vol. 4, (1966).

41. Quintana V. H. and Davison E. J., "Clipping-off gradient algorithms to compute optimal controls with constrained magnitude", Int. J. Control, Vol. 20, pp. 243-255, (1974).

42. Tripatki S. S. and Narendra K. S., "Optimization Using Conjugate Gradient Methods", IEEE Transaction Autom. Control, Vol. 15, pp. 268-270, (1970).

43. Lasdon L. L., "Conjugate Direction Methods for Optimal Control", IEEE Trans. Autom. Control, Vol. 15, pp. 267-268, (1970).

44. Jacobson D. H., "Second-Order and Second-Variation Methods for Determining Optimal Control. A Comparative Study Using Differential Dynamic Programming", Int. J. Control, Vol. 7, pp. 175-196, (1968).

45. Lapidus L. and Luus R., Optimal Control of Engineering Processes, Gimm/Blaisdel, (1967).

46. Oh S. H. and Luus R., "Use of Orthogonal Collocation Method in Optimal Control Problems", Int. J. Control, Vol. 26, No. 5, pp. 657-673, (1977).

47. Lynn L. L., Parkin E. S. and Zahrednik R. L., "Near-Optimal Control by Trajectory Optimization", I & EC Fund. 9, 1, p. 58, (1970).

48. Hicks G. and Ray W. H., "Approximation Methods for Optimal Control Synthesis", Canadian J. Chem. Eng., Vol. 49, pp. 522-528, (1971).

49. Asselmeyer B., "Optimal Control for Nonlinear Systems Calculated with small Computers", J. Opt. Theory & Appl., Vol. 45, pp. 533-543, (1985).

50. Sargent and Sullivan, "The Development of an Efficient Optimal Control Package", "Proceedings of the 8th IFIP Conference on Optimization Techniques", Wurzburg, (J. Stoer, ed.), (1977), Part 2, Springer-Verlag, Berlin, (1978).

51. Ritz W., "Uber eine neue Methode zur Losung gewisser Variationsprobleme der mathematischen Physik", Journal fur reine und angewandte Mathematik, Vol. 135, p. 1, (1908).

52. Zahrednik R. L. and Lynn L. L., Proc. JACC, paper 22E, (1970).

53. Bosarge E., Proceedings IFAC Symposium on the Control of Distributed Parameter Systems, Banff, Canada, June 1971.

54. Sage A. P., Optimum Systems Control, Prentice-Hall, Englewood Cliffs, N. J., (1968).

55. Bukovski A. G., Automatika i Telemekhanika, Vol. 22, p. 1565, (1961).

56. Bukovski A. G., Automatika i Telemekhanika, Vol. 24, p. 1217, (1963).

57. Zahradnik R. L. and Parkin E. S., Computing Methods in Optimization Problems, Academic Press, (1970).

58. Bosarge W. E. and Johnson O. G., Proceedings JACC, Vol. 51, (1970).

59. Prenter P. M., Splines and Variational Methods, John Wiley & Sons, Inc, (1975).

60. Rosenbrock H. H. and Storey C., Computational Techniques for Chemical Engineers, Pergamon, (1966).

61. Walder T. J. and Storey C., "Numerical Solution of an Optimal Temperature Problem", Chem. Eng. J., Vol. 1, pp. 120-128, (1970).

62. Eisenberg B. R. and Sage A. P., "Closed Loop Optimization of Fixed Configuration Systems", Int. J. Control, Vol. 3, pp. 183-194, (1966).

63. Tabak D. and Kuo B. C., Optimal Control by Mathematical Programming, Prentice-Hall, Englewood Cliffs, New Yersey (1971).

64. Hertzberg T. and Asbjornsen O. A., "Parameter Estimation in Nonlinear

Differential Equations", Computer Applications in the Analysis of Data and Plants, Science Press, Princeton, (1977).

65. Tsang T. H., Himmelblau D. M. and Edgar T. F., "Optimal Control via Collocation and Nonlinear Programming", Int. J. Control, Vol. 21, No. 5, pp. 763-768, (1975).

66. Luus R., "Time Optimal Control of Linear Systems", Can. J. Chem. Eng., Vol. 52, pp. 98-102, (1974).

67. Neumann C. P. and Casasayas L. G., "Parameter Identification Using the Galerkin Procedure in Nonlinear Boundary Value Problems", Trans. ASME, J. Dyn. Sys. & Control, p. 310, (1972).

68. Neumann C. P. and Sen A., "A Suboptimal Control Algorithm for Constrained Problems Using Cubic Splines", Automatica, Vol. 9, pp. 601-613, (1973).

69. Neumann C. P. and Sen A., "Weighted Residual Methods in Optimal Control", IEEE Trans. Aut. Control, p. 67, Feb., (1974).

70. Wang K. T. and Luus R., "Time suboptimal Feedback Control of Systems Described by Linear Parabolic Partial Differential Equations", Optimal Control Applications & Method, Vol. 3, pp. 177-185 (1982).

71. Lynn L. L. and Zahradnik R. L., "The Use of Orthogonal Polynomials in the Near-Optimal Control of Distributed Systems by Trajectory Optimization", Int. J. Control, Vol. 12, No. 6, pp. 1079-1087, (1970).

72. Renfro J. G., PhD Thesis, "Computational Studies in the Optimization of Systems Described by Differential/Algebraic Equations, University of Houston, UMI, 300 N. Zeeb Road, Ann Arbor, Michigan 48106 800-521-600 or 313/761-4700

73. Cuthrell J. E., PhD Thesis, "On the Optimization of Differential-Algebraic System of Equations in Chemical Engineering", Carnegie Mellon University, UMI, 300 N. Zeeb Road, Ann Arbor, Michigan 48106 800-521-600 or 313/761-4700

74. Biegler L. T., "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation", paper no. 27f, AIChE 97th National Meeting, San Francisco, California (1984).

75. Cuthrell J. E. and Biegler L. T., "On the Optimization of Differential-Algebraic

Process Systems", AIChE Journal, Vol. 33, No. 8, pp. 1257-1270 (1987).

76. Renfro J. G., Morshedi A. M. and Asbjornsen O. A., "Simultaneous Optimization and Solution of Systems Described by Differential/Algebraic Equations, Computers and Chemical Engineering, Vol. 11, No. 5, pp. 503-517, (1987).

77. Sarma P. V. L. N. and Reklaitis G. V., "Optimization of a Complex Chemical Process Using an Equation Oriented Model", Mathematical Programming Study, Vol. 20, pp. 113-160, (1982).

78. Biegler L. T., "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation", Computers and Chemical Engineering, VoL. 8, No. 3/4, pp. 243-248, (1984).

79. Cuthrell J. E. and Biegler L. T., "Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles", AIChE Journal, Vol. 33, No. 8, pp. 1-43, (1987).

80. Kalman R. E., Falb P. L. and Arbib M. A., Topics in Mathamatical System Theory, Mc Graw-Hill Book Company, (1969).

81. Biegler L. T., "On the Simultaneous Solution and Optimization of Large Scale Engineering Systems", 18. Congress on the Use of Computers in Chemical Engineering, EFCE, Giardini Noxos (Italy) 26-30 April 1987.

82. Bracken J. and Mc Cormick G. P., Selected Applications of Nonlinear Programming, John Wiley & Sons, Inc., New York, (1968).

83. Chang P. W. and Finlayson B. A., "Orthogonal Collocation on Finite Elements for Elliptic Equations", Mathematics and Computers in Simulation, 20, pp. 83-92, (1978).

84. Chang P. W. and Finlayson B. A., "Orthogonal Collocation on Finite Elements for Elliptic Equations", Advances in Computer Methods for Partial Differential Equations, Videhevetzky R. [ed.] IMACS (AICA) (1977).

85. Morbidelli M., Servida A., Storti G., Paludetto R., Carra S., "Application of the Orthogonal Collocation Method to some Chemical Engineering Problems", Ing. Chim. Ital., Vol. 19, No. 5-6, pp. 47-60, MAG.-GIU., (1983).

86. Finlayson B. A., "Orthogonal Collocation on Finite Elements - Progress and Potential", Mathematics and Computers in Simulation 22, pp. 11-17, (1980).

87. Houstis E. N., Mitchell W. F. and Rice J. R., "Collocation Software for Second-Order Elliptic Partial Differential Equations", ACM Transactions on Mathematical Software, Vol. 11, No. 4, pp. 379-412, (1985).

88. Ascher U., Christiansen J. and Russel R. D., "A Collocation Solver for Mixed Order Systems of Boundary Value Problems", Mathematics of Computation, Vol. 33, No. 146, pp. 659-679, (1979).

89. Carey G. F. and Finlayson B. A., "Orthogonal Collocation on Finite Elements", Chemical Engineering Science, Vol. 30, pp. 587-596, (1975).

90. Villadsen J. and Sorensen J. P., "Solution of Parabolic Differential Equations by a Double Collocation Method", Chemical Engineering Science, Vol. 24, pp. 1337-1349, (1969).

91. GRG2 User Guide, Professor Lasdon L. S., University of Texas, Austin, Texas

92. VF13AD User Guide, Harwell Physical Laboratory

93. Simusolv User Guide, Dow Chemical Company

94. Biegler L. T. and Grossmann I. E., "Strategies for the Optimization of Chemical Processes", Reviews in Chemical Engineering, Vol. 3, No. 1, pp. 1-47, (1985).

95. Fletcher J. P. and Ogbonda J. E., "A Modular Equation-Oriented Approach to Dynamic Simulation of Chemical Processes", The Use of Computers in Chemical Engineering, 18. Congress, Giardini Naxos (Italy), 26-30, April, (1987).

96. Parkin E. S. and Zahradnik R. L., "Computation of Near-Optimal Control Policies by Trajectory Approximation: Hyperbolic-Distributed Parameter Systems with Space-Independent Controls", AIChE Journal, Vol. 17., No. 2, pp. 409-412, (1971).

97. Homsy R. V. and Strohman R. D., "Diffusion and Chemical Reaction in a Tubular Reactor with Non-Newtonian Laminar Flow", AIChE Journal, Vol. 17, No. 1, pp. 215-219, (1971).

98. Cleland F. A. and Wilhelm R. H., "Diffusion and Reaction in Viscous-flow Tubular Reactor", AIChE Journal, Vol. 2, No. 4, pp. 489-497, (1956).

99. Stadtherr M. A., Cera G. D. and Alhire R. C., "Optimization of an Electrolytic Cell with Use of a GRG Algorithm", Computers and Chemical Engineering,

Vol. 7, No. 1, pp. 27-34, (1983).

100. Botha J. F. and Pinder G. F., Fundamental Concepts in the Numerical Solution of Differential Equations, John Wiley & Sons, (1983).

101. Berna T. J., Locke M. H. and Westerberg A. W., "A New Approach to Optimization of Chemical Processes", AIChE Journal, Vol. 26, No. 1, pp. 37-43, (1980).

102. Biegler L. T. and Hughes R. R., "Approximation Programming of Chemical Processes with Q/LAP CEP pp. 76-83, (1981).

103. Cuthrell J. E. and Biegler L. T., Simultaneous Solution and Optimization of Process Flowsheets with Differential Equation Models", Chem. Eng. Res. Des., Vol. 64, pp. 341-346, (1986).

104. Dennis J. E. Jr and More J. J., "Quasi-Newton Methods, Motivation and Theory", SIAM Review, Vol. 19, No. 1, pp. 46-89, (1977).

105. Finlayson B. A., Nonlinear Analysis in Chemical Engineering, Mc Graw-Hill International Book Company, (1980).

106. Fletcher R., "A New Approach to Variable Metric Algorithms", Computer Journal, Vol. 13, No. 3, pp. 317-322, (1970).

107. Gill P. E., Murray W. and Wright M. H., Practical Optimization, Academic Press, (1981).

108. Avriel M., "Nonlinear Programming", Chapter 11 in Holzman A. G. [editor], Mathematical Programming for Operation Researchers and Computer Scientists, Marcel Dekker, Inc., (1981).

109. Hong E. J., "Quadratic Approximation Methods for Constrained Nonlinear Programming", PhD Thesis, Georgia Institute of Technology (1982).

110. Han S. P., "A Globally Convergent Method for Nonlinear Programming", Journal of Optimization - Theory and Applications, Vol. 22, No. 3, pp. 297-309.

111. Himmelblau D. M., Applied Nonlinear Programming, Mc Graw-Hill Book Company, (1972).

112. Jang S. S., Joseph B. and Muhai H., "On-Line Optimization of Constrained Multivariable Chemical Processes", AIChE Journal, Vol. 33, No. 1, pp. 26-35, (1987).

113. Kaijaluoto S., Process Optimization by Flowsheet Simulation, Publication No. 20, Technical Research Center of Finland, (1984).

114. Kolm M. C., Practical Numerical Methods - Algorithms and Programs, Macmillan Publishing Company, (1987).

115. Kari R., "Parameterization and Comparative Analysis of the BFGS Optimization Algorithm for the Determination of Optimum Linear Coefficients", International Journal of Quantum Chemistry, Vol. 25, pp. 321-329, (1984).

116. Luenberger D. G., Linear and Nonlinear Programming, Addison-Wesley Publishing Company, (1984).

117. Himmelblau D. M., "A Uniform Evaluation of Unconstrained Optimization Techniques", Lootsma F. A. [editor], Numerical Methods for Nonlinear Optimization, Academic Press, (1972).

118. Locke M. H., Westerberg A. W. and Edahl R. H., "Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems", AIChE Journal, Vol. 29, No. 5, pp. 871-874, (1983).

119. Lang Y. D. and Biegler L. T., "A Unified Algorithm for Flowsheet Optimization", Computers and Chemical Engineering, Vol. 11, No. 2, pp. 143-158, (1987).

120. Lapidus L. and Pinder G. F., Numerical Solution of Partial Differential Equations in Science and Engineering, John Wiley & Sons, Inc., (1982).

121. Murray W. [editor], Numerical Methods for Unconstrained Optimization, Academic Press, (1972).

122. Murtagh B. A., "On the Simultaneous Solution and Optimization of Large-Scale Engineering Systems", Computers and Chemical Engineering, Vol. 6, No. 1, pp. 1-5, (1987).

123. Powell M. J. D., "Convergence Properties of Algorithms for Nonlinear Optimization", SIAM Review, Vol. 28, No. 4, pp. 487-500, (1986).

124. Reklaitis G. V., Ravindran A. and Ragsdell K. M., Engineering Optimization, John Wiley and Sons, (1983).

125. Ray W. H., "Multivariable Process Control - A Survey", Computers and Chemical Engineering, Vol. 7, No. 4, pp. 367-394, (1983).

126. Sarma P. V. L. N. and Reklaitis G. V., "Optimization of a Complex Chemical Process Using an Equation Oriented Model", Mathematical Programming Study, Vol. 20, pp. 113-160, (1982).

127. Schittkowski K., "The Current State of Constrained Optimization Software", Powell M. J. D. [editor], Nonlinear Optimization, Academic Press, New York, (1982).

128. Schittkowski K., "A Numerical Comparison of 13 Nonlinear Programming Codes with Randomly Generated Test Problems", Numerical Optimization of Dynamic Systems, Dixon L. C. W. and Szego G. P. [eds], North-Holland Publishing Company, (1980).

129. Schittkowski K., Nonlinear Programming Codes, Springer-Verlag Berlin (1980).

130. Villadsen J. and Michelsen M. L., Solution of Differential Equation Models by Polynomial Approximation, Prentice-Hall (1978).

131. Wolfe M. A., Numerical Methods for Unconstrained Optimization, Van Nostrand Reinhold Company (1978).

132. Wilde D. J., Optimum Seeking Methods, Prentice-Hall, Inc. (1964).

133. Williams T. J. and Otto R. E., "A generalized Chemical processing Model for the Investigation of Computer Control", AIEE Trans., Vol. 79, pp. 458-473, (1960).

134. Rao S. N. and Luus R., "Evaluation and Improvement of Control Vector Iteration Procedure", Can. J. Chem. Eng., Vol. 50, pp. 777-784, (1972).

135. Lee E. S., Quasi-Linearization and Invariant Imbedding, Academic Press, 1968.

136. Paterson W. R. and Creswell D. L., "A Simple Method for the Calculation of Effectiveness Factors", Chemical Engineering Science, Vol. 26, pp. 605-616, (1971).

137. Davis M. E., Numerical Methods and Modelling for Chemical Engineers, John Wiley & Sons, Inc., (1984).

138. DuChateau P. and Zachmann D., Applied Partial Differential Equations, Harper & Row, Publishers, Inc., (1989).

Appendix A - Optimizer User Guide

OPTIMIZER is a package written in SUN SPARC 2000 FORTRAN, which solves problems of the
following classes:

1. unconstrained optimization problem

$$\min g(x) \tag{P1}$$

2. linear or nonlinear programming problem with upper and lower bounds on the variables and on the constraints

$$\min g_k(x) \tag{P2}$$

subject to

$$lb_i \le x_i \le ub_i \qquad i=1,2,...,n$$

$$lb_{n+i} \le g_i(x) \le ub_{n+i} \qquad i=1,2,...,m \; i \ne k$$

3. linear or nonlinear programming problem

$$\min f(x) \tag{P3}$$

subject to

$$g_i(x) = 0 \qquad i=1,2,...,m_1$$

$$g_i(x) \ge 0 \qquad i=m_1+1,\ m_1+2,...,m$$

4. explicit quasilinear first-order system of ordinary differential equations (initial value problem)

$$\dot{c}(t) = f(c_1,c_2(t)) \ t\in\Omega \qquad c_2(0) = c_{20} \tag{P4}$$

5. first-order system of ordinary differential equations (initial or boundary value problem)

$$f(c_1,c_2(t),\dot{c}_2(t)) = \mathbf{0} \ t\in\Omega \qquad c_2 = \bar{c}_2 \ at \ t \in \partial\Omega \tag{P5}$$

6. second-order system of ordinary differential equations

$$f\left(c_1,c_2(t),\frac{dc_2(t)}{dt},\frac{d^2c_2(t)}{dt^2}\right) t\in\Omega \qquad c_2 = \bar{c}_2 \ at \ t \in \Omega \tag{P6}$$

7. first-order system of partial differential equations

$$f\left(c_1,c_4(x,y),\frac{\partial c_4(x,y)}{\partial x},\frac{\partial c_4(x,y)}{\partial y}\right) (x,y) \in \Omega \qquad c_4 = \bar{c}_4 \ at \ (x,y) \in \partial\Omega \tag{P7}$$

8. second-order system of partial differential equations

$$f(c_1,c_4(x,y),\frac{\partial c_4(x,y)}{\partial x},\frac{\partial c_4(x,y)}{\partial y},\frac{\partial^2 c_4(x,y)}{\partial x^2},\frac{\partial^2 c_4(x,y)}{\partial y^2},\frac{\partial^2 c_4(x,y)}{\partial x\partial y}) \quad (x,y) \in \partial\Omega \qquad c_4 = \bar{c}_4 \ at \ (x,y) \in \partial\Omega \tag{P8}$$

9. differential optimization problem where the constraint is a system of quasilinear first-order differential equations (initial value problem)

subject to

$$\min_{u_1,u_2(t)} G_1(t,c_1,c_2(t),p_1,p_2(t),u_1,u_2(t))\Big|_{t=t_f} + \int_{t_0}^{t_f} G_2(t,c_1,c_2(t),u_1,u_2(t),p_1,p_2(t))\ dt \qquad \text{(P9)}$$

$$\dot{c}_2(t) = f(t,c_1,c(_2t)),u_1,u_2(t),p_1,p_2(t))\ \ t \in \Omega$$

$$c_2(0) = c_{20}$$

10. differential optimization problem where the constraint is a system of quasilinear first-order differential equations (initial value problem) and the upper limit of the integration is also an optimizing variable

$$\min_{u_1,u_2(t),t_f} G_1(t,c_1,c_2(t),p_1,p_2(t),u_1,u_2(t))\Big|_{t=t_f} + \int_{t_0}^{t_f} G_2(t,c_1,c_2(t),u_1,u_2(t),p_1,p_2(t))\ dt \qquad \text{(P10)}$$

subject to

$$\dot{c}_2(t) = f(c_1,c(_2t)),u_1,u_2(t),p_1,p_2(t))\ \ t \in \Omega$$

$$c_2(0) = c_{20}$$

11. differential-algebraic optimization problem where the constraints are a system of second-order ordinary and/or partial differential equations(initial or boundary value problems), equations and/or inequalities containing state variables, control variables or parameters dependent on the independent variables, algebraic equations and/or inequalities

$$\min_{u_1,u_2(x),u_3(y),u_4(x,y)} G_1(x,y,c_1,c_2(x),c_3(y),c_4(x,y,p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y))\Big|_{x=x_f,y=y_f}$$
$$+\int_{x_0}^{x_f} G_2(c_1,c_2(x),u_1,u_2(x),c_4(x,y_f),p_1,p_2(x),x)\ dx$$
$$+\int_{y_0}^{y_f} G_3(c_1,c_3(y),u_1,u_3(y),c_4(x_f,y),p_1,p_3(y),y)\ dy$$
$$+\int_{x_0}^{x_f}\int_{y_0}^{y_f} G_4(c_1,c_2(x),c_3(y),c_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x,y)\ dxdy \qquad \text{(P11)}$$

subject to

$$f_1\left(c_1,c_2(x)),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},u_1,u_2(x),p_1,p_2(x),x\right) = 0 \ x\in\Omega_1$$
$$c_2 = \bar{c}_2 \ at \ x \in \partial\Omega_1$$

$$f_2\left(c_1,c_3(y)),\frac{dc_3(y)}{dy},\frac{d^2c_3(y)}{dy^2},u_1,u_3(y),p_1,p_3(y),y\right) = 0 \ y\in\Omega_2$$
$$c_3 = \bar{c}_3 \ at \ y \in \partial\Omega_2$$

$$f_3(c_1,c_2(x),c_3(y),c_4(x,y),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},\frac{dc_3(y)}{dy},\frac{d^2c_3(y)}{dy^2},$$
$$\frac{\partial c_4(x,y)}{\partial x},\frac{\partial^2 c_4(x,y)}{\partial x^2},\frac{\partial c_4(x,y)}{\partial y},\frac{\partial^2 c_4(x,y)}{\partial y^2},\frac{\partial^2 c_4(x,y)}{\partial x\partial y},$$
$$u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x.y) = 0 \ (x,y)\in\Omega_3$$
$$c_4 = \bar{c}_4 \ at \ (x,y) \in \partial\Omega_3$$

$$h^L \le h(x,y,c_1,c_2(x),c_3(y),c_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y)) \le h^U$$

$$z^L \le z \le z^U$$

$$u_1^L \le u_1 \le u_1^U$$

$$u_2^L \le u_2 \le u_2^U$$

$$u_3^L \le u_3 \le u_3^U$$

$$u_4^L \le u_4 \le u_4^U$$

12. differential-algebraic optimization problem where the constraints are a system of second-order ordinary and/or partial differential equations(initial or boundary value problems), equations and/or inequalities containing state variables, control variables or parameters dependent on the independent variables, algebraic equations and/or inequalities

$$\begin{aligned}\min_{u_1,u_2(x),u_3(y),u_4(x,y)} \; & G_1(x,y,c_1,c_2(x),c_3(y),c_4(x,y,p_1,p_2(x),p_3(y),p_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y))\big|_{x=x_f,y=y_f} \\ & + \int_{x_0}^{x_f} G_2(c_1,c_2(x),u_1,u_2(x),c_4(x,y_f),p_1,p_2(x),x)\; dx \\ & + \int_{y_0}^{y_f} G_3(c_1,c_3(y),u_1,u_3(y),c_4(x_f,y),p_1,p_3(y),y)\; dy \\ & + \int_{x_0}^{x_f}\int_{y_0}^{y_f} G_4(c_1,c_2(x),c_3(y),c_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x,y)dxdy \end{aligned} \qquad \text{(P12)}$$

subject to

$$f_1\left(c_1,c_2(x)),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},u_1,u_2(x),p_1,p_2(x),x\right) = 0 \;\; x\in\Omega_1$$

$$c_2 = \bar{c}_2 \;\; at \;\; x \in \partial\Omega_1$$

$$f_2\left(c_1, c_3(y)), \frac{dc_3(y)}{dy}, \frac{d^2c_3(y)}{dy^2}, u_1, u_3(y), p_1, p_3(y), y\right) = 0 \quad y \in \Omega_2$$

$$c_3 = \bar{c}_3 \ at \ y \in \partial\Omega_2$$

$$f_3 \ (c_1, c_2(x), c_3(y), c_4(x,y), \frac{dc_2(x)}{dx}, \frac{d^2c_2(x)}{dx^2}, \frac{dc_3(y)}{dy}, \frac{d^2c_3(y)}{dy^2},$$
$$\frac{\partial c_4(x,y)}{\partial x}, \frac{\partial^2 c_4(x,y)}{\partial x^2}, \frac{\partial c_4(x,y)}{\partial y}, \frac{\partial^2 c_4(x,y)}{\partial y^2}, \frac{\partial^2 c_4(x,y)}{\partial x \partial y},$$
$$u_1, u_2(x), u_3(y), u_4(x,y), p_1, p_2(x), p_3(y), p_4(x,y), x, y) = 0 \ (x,y) \in \Omega_3$$

$$c_4 = \bar{c}_4 \ at \ (x,y) \in \partial\Omega_3$$

$$h(x,y,c_1,c_2(x),c_3(y),c_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y)) = 0$$

$$g(x,y,c_1,c_2(x),c_3(y),c_4(x,y),u_1,u_2(x),u_3(y),u_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y)) \geq 0$$

$$u_1 - u_1^L \leq 0$$

$$u_1^U - u_1 \geq 0$$

$$u_2 - u_2^L \geq 0$$

$$u_2^U - u_2 \geq 0$$

$$u_3 - u_3^L \geq 0$$

$$u_3^U - u_3 \geq 0$$

$$u_4 - u_4^L \geq 0$$

$$u_4^U - u_4 \geq 0$$

N.B. Problem 11 is formulated in terms of the package GRG2, and problem 12 is formulated in terms of the package VF13AD. But they can easily be converted in each other, e.g. an individual bound on a variable can be replaced be 2 constraints, and an inequality constraint g(x) $\geq$ 0 can be considered a constraint having an upper bound 10^{38}, so that the same class of differential-algebraic problem can be solved either by GRG2 or by VF13AD.

13. parameter estimation problem using least-square method based on experimental data, where the constraint is a system of quasilinear first-order ordinary differential equations (initial value problem)

$$\min_{p_1} \sum_{i=1}^{m}\sum_{j=1}^{n} [c_j(t_i) - c_j'(t_i)]^2 \qquad \text{(P13)}$$

subject to

$$\dot{c}(t) = f(c(t),p_1) \quad t\in\Omega$$

$$c(0) = c_0$$

14. parameter estimation problem using least-square method based on experimental data, where the constraint is a system of second-order differential equations (initial or boundary value problem) , equations and/or inequalities containing state variables, control variables or parameters dependent on the independent variables, algebraic equations and/or inequalities

$$\min_{p_1,p_2(x),p_3(y),p_4(x,y)} \sum_{i=1}^{m}\sum_{j=1}^{n} [c_j(x_i,y_i) - c_j'(x_i,y_i)]^2 \qquad \text{(P14)}$$

subject to

$$f_1\left(c_1,c_2(x),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},p_1,p_2(x),x\right) = 0 \quad x\in\Omega_1$$

$$c_2 = \bar{c}_2 \ at\ x \in \partial\Omega_1$$

$$f_2\left(c_1,c_3(y),\frac{dc_3(y)}{dy},\frac{d^2c_2(y)}{dy^2},p_1,p_3(y),y\right) = 0 \quad y\in\Omega_2$$

$$c_3 = \bar{c}_3 \ at\ y \in \partial\Omega_2$$

$$lb_i \le h_i(z,c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x,y) \le ub_i \quad i=1,2,\ldots,m_1$$

$$lb_{i+m_1} \le g_i(c_1,p_1) \le ub_{i+m_1} \quad i=1,2,\ldots,m_2$$

$$f_3\ (c_1,c_2(x),c_3(y),c_4(x,y),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},\frac{dc_3(y)}{dy},\frac{d^2c_3(y)}{dy^2},$$
$$\frac{\partial c_4(x,y)}{\partial x},\frac{\partial c_4(x,y)}{\partial y},\frac{\partial^2 c_4(x,y)}{\partial x^2},\frac{\partial^2 c_4(x,y)}{\partial y^2},\frac{\partial^2 c_4(x,y)}{\partial x\partial y},$$
$$p_1,p_2(x),p_3(y),p_4(x,y),x,y) = 0 \quad (x,y)\in\Omega_3$$

$$c_4 = \bar{c}_4\ at\ (x,y) \in \partial\Omega_3$$

15. parameter estimation problem using least-square method based on experimental data, where the constraint is a system of second-order differential equations (initial or boundary value problem) , equations and/or inequalities containing state variables, control variables or parameters dependent on the independent variables, algebraic equations and/or inequalities

$$\min_{p_1,p_2(x),p_3(y),p_4(x,y)} \sum_{i=1}^{m}\sum_{j=1}^{n}[c_j(x_i,y_i) - c_j'(x_i,y_i)]^2 \qquad \text{(P15)}$$

subject to

$$f_1(c_1,c_2(x),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},p_1,p_2(x),x) = 0 \quad x\in\Omega_1$$

$$c_2 = \bar{c}_2\ at\ x \in \partial\Omega_1$$

$$f_2\left(c_1,c_3(y),\frac{dc_3(y)}{dy},\frac{d^2c_2(y)}{dy^2},p_1,p_3(y),y\right) = 0 \quad y\in\Omega_2$$

$$c_3 = \bar{c}_3\ at\ y \in \partial\Omega_2$$

$$f_3\ (c_1,c_2(x),c_3(y),c_4(x,y),\frac{dc_2(x)}{dx},\frac{d^2c_2(x)}{dx^2},\frac{dc_3(y)}{dy},\frac{d^2c_3(y)}{dy^2},$$
$$\frac{\partial c_4(x,y)}{\partial x},\frac{\partial c_4(x,y)}{\partial y},\frac{\partial^2 c_4(x,y)}{\partial x^2},\frac{\partial^2 c_4(x,y)}{\partial y^2},\frac{\partial^2 c_4(x,y)}{\partial x\partial y},$$
$$p_1,p_2(x),p_3(y),p_4(x,y),x,y) = 0 \quad (x,y)\in\Omega_3$$

$$c_4 = \bar{c}_4\ at\ (x,y) \in \partial\Omega_3$$

$$g_i(c_1,p_1) = 0 \quad i=1,2,\dots,m_1$$

$$g_i(c_1,p_1) \geq 0 \quad i=m_1+1,m_1+2,\dots,m$$

$$h_i(c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x,y) = 0 \quad i=1,2,\dots,m_2$$

$$h_{i+m_2}(c_1,c_2(x),c_3(y),c_4(x,y),p_1,p_2(x),p_3(y),p_4(x,y),x,y) \geq 0 \quad i=1,2,\dots,m_3$$

N.B. Problem 14 is formulated in terms of the package GRG2, and problem 15 is formulated in terms of the package VF13AD. But they can easily be converted in each other, e.g. an individual bound on a variable can be replaced be 2 constraints, and an inequality constraint $g(x) \geq 0$ can be considered a constraint having an upper bound 10^{38}, so that the same class of parameter estimation problem can be solved either by GRG2 or by VF13AD. Also Problems 14 and 15 are formulated as distributed parameter estimation problems, parameter estimation problems for lumped parameter systems can also be solved.

The package offers the following options:

1. Unconstrained optimization using Hooke-Jeeves method.
   It can be used to solve problem 1.

2. Unconstrained optimization using the Broyden-Fletcher-Goldfarb-Shanno method.
   It can be used to solve problem 1.

3. It has been superseded by options 15 and 17.

4. It has been superseded by options 15 and 17.

5. It has been superseded by option 6.

6. It uses BFGS method and control variable parameterization.
   It can be used to solve problem 4, 9, 10 and 13. If problem 4,9 or 13 has many differential equations, than option 6 is faster than option 16 or option 18, although it is easier to set up a problem for computer run using options 16 or 18.

7. It has been superseded by option 6.

8. It uses Hooke_Jeeves method and control variable parameterization.
   It can be used to solve problem 4, 9, 10 and 13. If problem 4,9 or 13 has many differential equations, than option 8 is faster than option 16 or option 18, although it is easier to set up a problem for computer run using options 16 or 18.

9. It has been superseded by options 16 and 18.

10. It has been superseded by options 16 and 18.

11. Parameter estimation using Hooke-Jeeves method and control variable parameterization.
    It can be used to solve problem 13 when there are no material balance equations.

12. Parameter estimation using BFGS method and control variable parameterization.
    It can be used to solve problem 13 when there are no material balance equations.

13. It has been superseded by options 16 and 18.

14. It has been superseded by options 16 and 18.

15. This option invokes the GRG2 package.
    It can be used to solve problems 1 and 2.

16. This option invokes the GRG2 package.
    The model has to be first input to the package OCFE.
    It can be used to solve problems 4, 5, 6, 7, 8, 9, 11 and 14.

17. This option invokes the Vf13AD package.
    It can be used to solve problem 3.

18. This option invokes the Vf13AD package.
    The model has to be first input to the package OCFE.
    It can be used to solve problems 4, 5, 6, 7, 8, 9, 12 and 15.

Program files and data files.

The package contains the following declaration in its subroutines:

```
IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
```

Program source files:

ocfe.for	preprocessor for options 16 and 18
optimizer.for	main program
dispatcher.for	main driving subroutine
hjbfgs.for	subroutines performing options 1-14
usersubs.for	user supplied subroutines computing the objective function or objective functional and derivatives for options 1-14
subint.for	subroutines comprising GRG2

grg21.for	subroutines comprising GRG2
grg22.for	subroutines comprising GRG2
sqp.for	subroutines comprising VF13AD
userint.for	user supplied subroutines containing various functions callable from GCOMP or SQPCONSTRAINTS when using options 16 or 18. e.g., if one has a boundary condition C=IF T < THETA THEN 0.5 ELSE 0, then such case can be handled by writing a function in userint.for. Functions handling disturbances and initializing state variables have to be put also in userint.for.
ocfeint.for	user supplied subroutines containing various functions callable from OCFE when using options 16 or 18. e.g., the function handling variable time delay VARTIMEDELAY and the irregular domain handling function WITHIN have to be in ocfeint.for.
sqpint.for	The user has to write REAL*8 FUNCTION SQPOF(X) and SUBROUTINE SQPCONSTRAINTS(C,U,N,M) when using option 17.
grg2int.for	The user has to write SUBROUTINE GCOMP(G,X) when using option 18.
evalerr.for	It is output by OCFE if options 16 or 18 are used. It computes the residual error after OPTIMIZER has been run and outputs ERROROUT.DAT. It reads in OPTSOLUTION.DAT
error.for	It is output by OCFE if options 16 or 18 are used. If one solves problems 5,6,7,8,9,11,12 using option 16 or 18, ERROR(X) has to be added to the objective functional if the problem is defined as minimization and ERROR(X) has to be subtracted from the objective functional if the problem is defined as maximization. Then variables cannot be named X. ERROR(X) may shift the optimum, so it is recommended that with optimization problems the user runs his/her problem with and without ERROR(X). One also can use weights like F(T)+0.005*ERROR(X). Before linking OPTIMIZER, error.for has to be compiled.
evalfunc.for	It is output by OCFE if options 16 or 18 are used. It computes the values of the state and control variables at equidistant points along the independent variables after OPTIMIZER has been run. It outputs FUNCOUT.DAT and reads OPTSOLUTION.DAT
plotprep.for	It generates data for GNUPLOT.

Executable modules:

ocfe.exe	It has to be run if options 16 or 18 are used. It reads OCFEINP.DAT and outputs OCFEOUT.DAT, evalerr.for, error.for, and evalfunc.for. If option 16 is used, it outputs grg2int.for and OCFEGRG2IF.DAT. If options 18 is used, it outputs sqpint.for and OCFESQPIF.DAT.
optimizer.exe	It reads OPTIMINP.DAT. If option 16 is used, it reads OCFEGRG2IF.DAT. If option 18 is used, it reads OCFESQPIF.DAT. It outputs OPTSOLUTION.DAT and OPTIMOUT.DAT. OPTSOLUTION.DAT is read by EVALERR and by EVALFUNC.

Data Files:

OCFEINP.DAT	It contains the model if options 16 or 18 are used. It is read by OCFE.
OCFEOUT.DAT	It is output by OCFE.
OCFEGRG2IF.DAT	It is output by OCFE if option 16 is used.
OCFESQPIF.DAT	It is output by OCFE if option 18 is used.
OPTSOLUTION.DAT	It is output by OCFE.
OPTIMINP.DAT	It contains input to optimizer
OPTIMOUT.DAT	Output of OPTIMIZER
FUNCOUT.DAT	Output of EVALFUNC
ERROROUT.DAT	Output of EVALERR
PLOTPREP.DAT	It is input file to PLOTPREP, output by OCFE.
ROOT5.DAT	It is output by OCFE, read by DISPATCHER, used by INIT2 in userint.for to initialize state variables as a function of independent variables.
EXPER.DAT	It is output by OCFE, read by DISPATCHER, used for printing and comparing the experimental data and computed data in parameter estimation.
OCFETEMP.DAT	temporary file created by OCFE
OCFETEMP2.DAT	temporary file created by OCFE

OCFETEMP3.DAT	temporary file created by OCFE
OCFETEMP4.DAT	temporary file created by OCFE

Steps taken if option 16 is used:

1. set up OCFEINP.DAT and OPTIMINP.DAT

2. delete or rename ROOT5.DAT, EXPER.DAT, OCFETEMP.DAT, OCFETEMP2.DAT, OCFETEMP3.DAT, OCFETEMP4.DAT, OCFEOUT.DAT, EVALFUNC.FOR, PARESTRES.FOR, EVALERR.FOR, ERROR.FOR, GRG2INT.FOR, OCFEGRG2IF.DAT, PLOTPREP.DAT

3. ocfe.exe

4. lpr OCFEINP.DAT OCFEOUT.DAT OCFEGRG2IF.DAT

5. mv GRG2INT.FOR grg2int.for

6. f77 grg2int.for -c -x*l* -i4

7. f77 error.for -c -x*l* -i4 if option 16 is used for problems 4,5,6,7,8,9 or 11 and ERROR(X) or - ERROR(X) was added to the objective functional.

8. f77 ~illesj*l*/optimizer.o ~illesj*l*/dispatcher.o ~illesj*l*/hjbfgs.o ~illesj*l*/subint.o ~illesj*l*/grg21.o ~illesj*l*/grg22.o ~illesj*l*/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

9. delete or rename OPTIMOUT.DAT, OPTSOLUTION.DAT and DUMP.DAT if IDUMP=1 and IRESTART=0

10. at hh:mm
    time optimizer.exe > optimizer.log
    ctrl d
    logout

11. lpr OPTIMINP.DAT OPTIMOUT.DAT optimizer.log

12. mv EVALERR.FOR evalerr.for
    f77 evalerr.for -c -xl -i4
    f77 evalerr.o -o evalerr.exe
    evalerr.exe
    lpr ERROROUT.DAT

13. mv EVALFUNC.FOR evalfunc.for
    f77 evalfunc.for -c -xl -i4
    f77 evalfunc.o -o evalfunc.exe
    evalfunc.exe

```
e.g. 0.5 0.5 or 0.1 0.1
lpr FUNCOUT.DAT
```

Steps taken if option 18 is used:

1. set up OCFEINP.DAT and OPTIMINP.DAT

2. delete or rename ROOT5.DAT, EXPER.DAT, OCFETEMP.DAT, OCFETEMP2.DAT, OCFETEMP3.DAT, OCFETEMP4.DAT, OCFEOUT.DAT, EVALFUNC.FOR, PARESTRES.FOR, EVALERR.FOR, ERROR.FOR, SQPINT.FOR, OCFESQPIF.DAT, PLOTPREP.DAT

3. ocfe.exe

4. lpr OCFEINP.DAT OCFEOUT.DAT OCFESQPIF.DAT

5. mv SQPINT.FOR sqpint.for

6. f77 sqpint.for -c -x*l* -i4

7. f77 error.for -c -x*l* -i4 if option 18 is used for problems 4,5,6,7,8,9 or 12 and ERROR(X) or - ERROR(X) was added to the objective functional.

8. f77 ~illesj*l*/optimizer.o ~illesj*l*/dispatcher.o ~illesj*l*/hjbfgs.o ~illesj*l*/subint.o ~illesj*l*/grg21.o ~illesj*l*/grg22.o ~illesj*l*/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

9. delete or rename OPTIMOUT.DAT, OPTSOLUTION.DAT and DUMP.DAT if IDUMP=1 and IRESTART=0

10.
```
at hh:mm
time optimizer.exe > optimizer.log
ctrl d
logout
```

11. lpr OPTIMINP.DAT OPTIMOUT.DAT optimizer.log

12.
```
mv EVALERR.FOR evalerr.for
f77 evalerr.for -c -xl -i4
f77 evalerr.o -o evalerr.exe
evalerr.exe
lpr ERROROUT.DAT
```

13.
```
mv EVALFUNC.FOR evalfunc.for
f77 evalfunc.for -c -xl -i4
f77 evalfunc.o -o evalfunc.exe
evalfunc.exe
e.g. 0.5 0.5 or 0.1 0.1
```

lpr FUNCOUT.DAT

Steps taken if options 1 or 2 are used:

1. Set up the objective function in usersubs.for and OPTIMINP.DAT

2. f77 usersubs.for -c -x*l* -i4

3. f77 ~illesj*l*/optimizer.o ~illesj*l*/dispatcher.o ~illesj*l*/hjbfgs.o ~illesj*l*/subint.o ~illesj*l*/grg21.o ~illesj*l*/grg22.o ~illesj*l*/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

4. delete or rename OPTIMOUT.DAT

5. at hh:mm
   time optimizer.exe > optimizer.log
   ctrl d
   logout

6. lpr OPTIMINP.DAT OPTIMOUT.DAT

Steps taken if options 6,8,11 or 12 are used:

1. Set up the objective function in usersubs.for and OPTIMINP.DAT

2. Set up the computation of derivatives in the FUNCTION DX2 in usersubs.for

3. Set up the overall mass-balance equation if it is part of the model in the SUBROUTINE OVERALLMASSBALANCE in usersubs.for

4. f77 usersubs.for -c -x*l* -i4

5. f77 ~illesj*l*/optimizer.o ~illesj*l*/dispatcher.o ~illesj*l*/hjbfgs.o ~illesj*l*/subint.o ~illesj*l*/grg21.o ~illesj*l*/grg22.o ~illesj*l*/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

6. delete or rename OPTIMOUT.DAT

7. at hh:mm
   time optimizer.exe > optimizer.log
   ctrl d
   logout

8. lpr OPTIMINP.DAT OPTIMOUT.DAT

Steps taken if option 15 is used:

1. Set up SUBROUTINE GCOMP in grg2int.for and OPTIMINP.DAT

2. f77 grg2int.for -c -xl -i4

3. f77 ~illesjl/optimizer.o ~illesjl/dispatcher.o ~illesjl/hjbfgs.o ~illesjl/subint.o ~illesjl/grg21.o ~illesjl/grg22.o ~illesjl/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

4. delete or rename OPTIMOUT.DAT OPTSOLUTION.DAT and DUMP.DAT if IDUMP=1 and IRESTART=0

5. at hh:mm
   time optimizer.exe > optimizer.log
   ctrl d
   logout

6. lpr OPTIMINP.DAT OPTIMOUT.DAT

Steps taken if option 17 is used:

1. Set up FUNCTION SQPOF and SUBROUTINE SQPCONSTRAINTS in sqpint.for and OPTIMINP.DAT

2. f77 sqpint.for -c -xl -i4

3. f77 ~illesjl/optimizer.o ~illesjl/dispatcher.o ~illesjl/hjbfgs.o ~illesjl/subint.o ~illesjl/grg21.o ~illesjl/grg22.o ~illesjl/sqp.o usersubs.o userint.o sqpint.o grg2int.o error.o -o optimizer.exe

4. delete or rename OPTIMOUT.DAT OPTSOLUTION.DAT and DUMP.DAT if IDUMP=1 and IRESTART=0

5. at hh:mm
   time optimizer.exe > optimizer.log
   ctrl d
   logout

6. lpr OPTIMINP.DAT OPTIMOUT.DAT

When option 16 is selected, some of the input variables are read from OCFEGRG2IF.DAT and when option 18 is selected, some of the input variables are read from OCFESQPIF.DAT

Syntax of OCFEINP.DAT

DIFFEQCLASS IOPT NFE(1) NFE(2) NFE(3) NFE(4)

NICP1(1)...NICP1(NFE(1))

```
NICP1(1)...NICP2(NFE(2))

NICP3(1)...NICP3(NFE(3))

NICP4(1)...NICP4(NFE(4))

PROBLEMNAME

DELTAT DELTAX DELTAY DELTAZ

NAE1 NALGEQ NALGINEQ NALGVAR NTDEPEQ NTDEPINEQ NEQ NDE
NICBC NTDSCVPAR IS VILLORFIN

VARCLASS(1)...VARCLASS(NTDSCVPAR)

DO I=1;NTDSCVPAR
if VARCLASS(I) > 1 and VARCLASS(I) < 5 then
N01(I) N11(I) N02(I) N12(I) II00(I) II10(I) II01(I) II11(I) IBS0(I) IBS1(I) IBF0(I)
IBF1(I)
endif
enddo

N=NAE1+NALGEQ+NALGINEQ+NTDEPEQ+NTDEPINEQ+NEQ+NDE+NICBC+1

DECLASS(1)...DECLASS(N)

do I=1,N
if DECLASS(I) > 0 and DECLASS(I) < 4 or DECLASS(I) > 12 and DECLASS(I) <
16
then
M01(I) M11(I) M02(I) M12(I) JJ00(I) JJ10(I) JJ01(I) JJ11(I) JBS0(I) JBS1(I) JBF0(I)
JBF1(I)
endif
enddo

NAMEVAR(1)...NAMEVAR(NTDSCVPAR)

SATISFIED(1)...SATISFIED(4)

LIB(1) UIB(1) LIB(2) UIB(2) LIB(3) UIB(3) LIB(4) UIB(4)

if NFE(1) > 1 then
KNOT1(1)...KNOT1(NFE(1)-1)
endif

if NFE(2) > 1 then
KNOT2(1)...KNOT2(NFE(2)-1)
endif
```

```
INPUTTEXT(1)...INPUTTEXT(N)

if INPUTTEXT(N) = 'PARAMETER ESTIMATION' then
NLINES
INPUTTEXT(N)...INPUTTEXT(N+NLINES-1)
endif
```

Composition of INPUTTEXT

NAE2=NALGEQ+NALGINEQ

NTDC=NTDEPEQ+NTDEPINEQ

Option 16

NAE1 algebraic equations like R=1.9865

NAE2 algebraic constraints like X(1)+X(2)+X(3)=1

NTDC time-dependent consraints like $c_1+c_2+c_3=1$

NEQ equations like rate constant or feedback control law

NDE differential equations

NICBC boundary conditions

objective functional

if there is parameter estimation then
PARAMETER ESTIMATION
N
N lines containing the least-squares objective functional at sampled points of the independent variable

Option 18

NAE1 algebraic equations like R=1.9865

NALGEQ algebraic constraints like X(1)+X(2)+X(3)=1

NTDEPEQ time-dependent consraints like $c_1+c_2+c_3=1$

NEQ equations like rate constant or feedback control law

NDE differential equations

NICBC boundary conditions

NALGINEQ algebraic inequalities like X(1)+X(2)+X(3)>1

NTDEPINEQ time-dependent inequalities like 300 < temperature(t)
and 400 > temperature(t)

objective functional

if there is parameter estimation then
PARAMETER ESTIMATION
N
N lines containing the least-squares objective functional at sampled points of the independent variable

Description of input variables

Name	Role
DIFFEQCLASS	=1 ODE =3 PDE
IOPT	=1 use GRG2 =2 use VF13AD
NFE(1)	number of finite elements along the first independent variable
NFE(2)	number of finite elements along the second independent variable
NFE(3),NFE(4)	not used
NICP1(I)	number of internal collocation points along the first independent variable in the i-th subinterval
NICP2(I)	number of internal collocation points along the second independent variable in the i-th subinterval
NICP3,NICP4	not used
PROBLEMNAME	problem name
DELTAT	The residual error will be evaluated at DELTAT intervals along the first independent variable
DELTAX	The residual error will be evaluated at DELTA intervals along the second independent variable
DELTAY,DELTAZ	not used

NAE1	number of algebraic equations like R=1.9865
NALGEQ	number of algebraic equations like x(1)+x(2)=1
NALGINEQ	number of algebraic equations like x(1)+x(2)>0
NALGVAR	number of algebraic variables
NTDEPEQ	number of equations containing at least one variable dependent on the independent variable like $c_1(t)+c_2(t)+c_3(t)=1$
NTDEPINEQ	number of inequalities containing at least one variable dependent on the independent variable like c(t)>0
NEQ	number of equations like k=exp(E/RT)
NDE	number of differential equations
NICBC	number of boundary conditions
NTDSCVPAR	number of variables dependent on the independent variables
IS	should always be zero
VILLORFIN	=1 programs taken from Villadsen and Michelsen are used
	=2 programs taken from Finlayson are used
	if the number of internal collocation points > 5, select VILLORFIN=1
VARCLASS	=1 independent variable
	=2 dependent state variable like concentration dependent on the first independent variable
	=3 dependent state variable like concentration dependent on the second independent variable
	=4 dependent state variable like concentration dependent on both independent variables
	=5 control variable like temperature dependent on the first independent variable
	=6 control variable like temperature dependent on the second independent variable

=7	control variable like temperature dependent on both independent variables
=8	parameter like rate constant dependent on the first independent variable
=9	parameter like rate constant dependent on the second independent variable
=10	parameter like rate constant dependent on both independent variables
=11	control variable dependent on the first independent variable piecewise constant function over the subintervals
=12	control variable dependent on the second independent variable piecewise constant function over the subintervals
=13	control variable dependent on 2 independent variables, piecewise constant function over subintervals along the axis of the first independent variable
=14	control variable dependent on 2 independent variables, piecewise constant function over subintervals along the axis of the second independent variable
=15	parameter dependent on the first independent variable piecewise constant function over the subintervals
=16	parameter dependent on the second independent variable piecewise constant function over the subintervals
=17	parameter dependent on 2 independent variables, piecewise constant function over subintervals along the axis of the first independent variable
=18	parameter dependent on 2 independent variables, piecewise constant function over subintervals along the axis of the second independent variable
=19	user-defined function in userint.for
=20	variable in irregular domain
=21	control variable dependent on 2 independent variables,

		constant over 2-dimensional finite elements
	=22	parameter dependent on 2 independent variables, constant over 2-dimensional finite elements
	=23	control variable dependent on the first independent variable is a constant function over the integration domain
	=24	control variable dependent on the second independent variable is a constant function over the integration domain
	=25	control variable dependent on 2 independent variables constant over 2-dimensional integration domain
N01(i)	=1	if an initial condition is given in an ODE associated with the i-th state variable dependent on the first independent variable
N11(i)	=1	if a boundary condition is given in an ODE associated with the i-th state variable dependent on the first independent variable
N02(i)	=1	if an initial condition is given in an ODE associated with the i-th state variable dependent on the second independent variable
N12(i)	=1	if an boundary condition is given in an ODE associated with the i-th state variable dependent on the second independent variable
II00(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable at the start value of the first independent variable and at the start value of the second independent variable
II10(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable at the final value of the first independent variable and at the start value of the second independent variable
II01(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable at the start value of the first independent variable and at the final value of the second independent variable

II11(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable at the final value of the first independent variable and at the final value of the second independent variable
IBS0(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable along the axis of the first independent variable at the start value of the second independent variable
IBS1(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable along the axis of the first independent variable at the final value of the second independent variable
IBF0(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable along the axis of the second independent variable at the start value of the first independent variable
IBF1(i)	=1	if a boundary condition is given in a PDE associated with the i-th state variable along the axis of the second independent variable at the final value of the first independent variable
DECLASS	=1	ODE dependent on the first independent variable
	=2	ODE dependent on the second independent variable
	=3	PDE
	=4	boundary condition on the first independent variable
	=5	boundary condition on the first independent variable
	=6	boundary condition in a PDE
	=7	objective functional on an ODE associated with the first independent variable
	=8	objective functional on an ODE associated with the second independent variable
	=9	objective functional in a PDE containing at least one state variable dependent on 2 independent variables

	=10	constraint dependent on the first independent variable
	=11	constraint dependent on the second independent variable
	=12	constraint dependent on both independent variables
	=13	equations like the Arrhenius law for the rate constant dependent on the first independent variable
	=14	equations like the Arrhenius law for the rate constant dependent on the second independent variable
	=15	equations like the Arrhenius law for the rate constant dependent on both independent variable
	=16	PDE with irregular domain
	=17	ICBC with irregular domain
	=18	individual bound on control var of class 5
	=19	individual bound on control var of class 6
	=20	individual bound on control var of class 7
	=21	individual bound on control var of class 11
	=22	individual bound on control var of class 12
	=23	individual bound on control var of class 13
	=24	individual bound on control var of class 14
	=25	individual bound on control var of class 21
	=26	individual bound on control var of class 23
	=27	individual bound on control var of class 24
	=28	individual bound on control var of class 25
	=29	differential equation to be satisfied at a single point.
M01(i)	=1	if an initial condition is given in the i-th ODE dependent on the first independent variable
M11(i)	=1	if an boundary condition is given in the i-th ODE dependent on the first independent variable

M02(i)	=1	if an initial condition is given in the i-th ODE dependent on the second independent variable
M12(i)	=1	if an boundary condition is given in the i-th ODE dependent on the second independent variable
JJ00(i)	=1	if a boundary condition is given in the i-th PDE at the start value of the first independent variable and at the start of the second independent variable
JJ10(i)	=1	if a boundary condition is given in the i-th PDE at the final value of the first independent variable and at the start of the second independent variable
JJ01(i)	=1	if a boundary condition is given in the i-th PDE at the start value of the first independent variable and at the final of the second independent variable
JJ11(i)	=1	if a boundary condition is given in the i-th PDE at the final value of the first independent variable and at the final of the second independent variable
JBS0(i)	=1	if a boundary condition is given in the i-th PDE along the axis of the first independent variable at the start value of the second independent variable
JBS1(i)	=1	if a boundary condition is given in the i-th PDE along the axis of the first independent variable at the final value of the second independent variable
JBF0(i)	=1	if a boundary condition is given in the i-th PDE along the axis of the second independent variable at the start value of the first independent variable
JBF1(i)	=1	if a boundary condition is given in the i-th PDE along the axis of the second independent variable at the final value of the first independent variable
NAMEVAR		names of variables, maximum 3 characters, independent variables only 1 character
SATISFIED(1)	=1	if the differential equation is defined at the start value of the first independent variable
SATISFIED(2)	=1	if the differential equation is defined at the final value of the first independent variable
SATISFIED(3)	=1	if the differential equation is defined at the start value of the second independent variable

SATISFIED(4)	=1 if the differential equation is defined at the final value of the second independent variable
LIB(1)	start value of the first independent variable
UIB(1)	final value of the first independent variable
LIB(2)	start value of the second independent variable
UIB(2)	final value of the second independent variable
LIB(3)	unused, but required
UIB(3)	unused, but required
LIB(4)	unused, but required
UIB(4)	unused, but required
KNOT1(I)	i-th finite element knot along the first independent variable
KNOT2(I)	i-th finite element knot along the second independent variable
INPUTTEXT(I)	the i-th equation if an equation is longer than 80 characters then at the end of the line & must be used indicating that the next line is a continuation line

Rules, assumptions and limitations of the OCFE package

1. grg2int.for and sqpint.for are automatically generated using information contained in OCFEINP.DAT.

2. Mixed ODEs and PDEs are handled.

3. There must be at least 2 internal collocation points in each finite element, if there is a control variable, i.e., the problem is an optimization problem, and the control variable is not constant over the finite elements.

4. The number of internal collocation points can vary from finite element to finite element.

5. The maximum number of independent variables is 2, but problems containing 3 independent variables can be transformed into problems having 2 independent variables using finite differencing. If there is a second derivative term with respect to the variable z to be eliminated then there must be boundary conditions defining the value of the state function at the start point and at the end point of the integration with respect to z.

6. The maximum order of differential equations is 2.

7. The maximum number of NLP variables is 1500.

8. The maximum number of NLP constraints is 1500.

9. Lines starting with asterisk are treated as comments.

10. Currently only rectangular domains are handled, the code handling irregular domains has been implemented, but it has not been tested on real problem and it is not sure, that it will work on real problems.

11. Algebraic variables, i.e. variables not dependent on the independent variables must appear in OCFEINP.DAT subscripted starting from subscript 1, they have to be named x and x cannot be a variable name.

12. The objective functional and initial and boundary conditions can contain derivative terms.

13. NLP variables derived from state variables are not bounded.

14. For second-order differential equations the Lagrange interpolation polynomial must be at least cubic in order to prevent constraints becoming identical by having constant derivatives if there are no first derivative terms and independent variables do not appear in the differential equation.

15. OCFE in conjunction with GRG2 and VF13AD can be used to solve differential equations when there is no control variable, i.e., when the problem is not an optimization problem.

16. State variables dependent on 2 independent variables can be fixed at knot points or at the start or at the end of the integration interval of an independent variable. It occurs normally in integral terms of an objective functional when integration is performed along the edge of a rectangular integration domain.

17. Control variables and parameters can be constant over a finite element.

18. There can be no time delay in derivative terms and in parameters like rate constant and in objective functional and initial and boundary conditions.

19. c(0,t) or c(t,0) indicates that a state variable is fixed at start value of an independent variable. c(1,t) or c(t,1) indicates that a state variable is fixed at end value of an independent variable.

20. Differential equations, integral terms of an objective functional and initial and boundary conditions can contain state and control variables and parameters.

21. If a state variable is fixed at knot point, then it is indicated by putting 'L' instead of the independent variable, consequently 'L' cannot be the name of an independent variable.

22. If a variable c(t,x) appears in an integral term of the objective functional, as c(0,t)dt, the new variable derived from this integral term is defined at x=0 along the t-axis. It is dependent only on t.

23. There can be more than 1 integral term in an objective functional.

24. The independent variable must appear in parentheses following a state or control variable or a parameter.

25. Variable time delays should be defined in ocfeint.for in the function VARTIMEDELAY.

26. Both state and control variables can have time delays.

27. A PDE must contain at least 1 variable dependent on both independent variables.

28. If a model contains function references such that the argument is NLP variable x, then x cannot be a variable name, the function name cannot be a variable name, and the function must be defined in userint.for.

29. If a variable dependent on both independent variables appears in an ODE, then this variable must have one of its independent variables fixed.

30. For a PDE IC/BC conditions have to be specified separately for the corners and for the sides of the rectangular domain. Boundary conditions for a side exclude these end points. For instance,
BC AT T=0 AT X=0
BC AT T=0 AT X=1
BC AT T=1 AT X=0
BC AT T=1 AT X=1
BC AT T=0 [0≤X≤1 IMPLIED]
BC AT T=1 [0≤X≤1 IMPLIED]
BC AT X=0 [0≤T≤1 IMPLIED]
BC AT X=1 [0≤T≤1 IMPLIED]

are all the possible initial/boundary conditions for a state variable dependent on both independent variables.

31. A PDE can contain variables dependent only on 1 independent variable.

32. Independent variables can appear in any order in parentheses.

33. If, say, there are 2 spatial variables $0 < x < L_1$ and $0 < y < L_1$, then every dependent variable must be dependent on x or on y or on both over the entire

interval or rectangle. Also, in the model there can be no more than 2 independent variables. In one word, 2 units cannot be modeled together if both are modelled by differential equations with variables in both units dependent on spatial variables if the range of the spatial variables differs in units. But if only 1 unit contains differential equations dependent on a spatial variable and on time and the other units contain ordinary different equations dependent on time only, then this case can be handled. Also, if variables are transformed like $0 < z' < L$ and we transform z' as $z=z'/L$, then $0 < z < 1$ for each spatial independent variable and the above restriction is removed. Even if more than 1 units are modelled together, the lower and upper limit of integration is common for every unit. This implies that a unit cannot be started later or earlier and cannot be stopped later or earlier than other units.

34. A variable cannot be fixed and cannot have time delay at the same time.

35. If a control variable has time delay, then it cannot have an associated parameter, e.g., if k=A*exp(-E/R/Temperature), then not k, but A*exp(-E/R/Temperature) should appear in the differential equations.

36. If an integral term in the objective functional is to be maximized, then a - sign should precede the integral, like -INTEGRAL, if VF13AD is used. If GRG2 is used, either -INTEGRAL and 1 in OPTIMINP.DAT second line last number, or no minus sign and 2 in OPTIMINP.DAT last number must be given. If the objective functional is minimized, no - sign and if GRG2 is used, 1 in OPTIMINP.DAT second line last number must be given.

37. Variable names should not start with @.

38. If the letter D appears in a variable name, then there cannot be another variable name contained entirely in the first variable whose first character starts with a letter following the D in the first variable, because the first derivative of the second variable would be taken for the first variable, e.g. first variable = DAB and the second variable = AB.

39. If there is a time delay and both the time delay and the initial function is constant, it is indicated as e.g. t-0.5;0.5. otherwise as TIMEDELAY(i) where i is a label in a FUNCTION VARTIMEDELAY in ocfeint.for.

40. x cannot be a variable name if error(x) appears in the objective functional.

41. A state variable fixed at knot cannot appear in boundary condition and in the objective functional and in a parameter equation.

42. A state variable in derivative term cannot be fixed at knot or at boundary.

43. Disturbances and variables that come from another process unit and therefore cannot be manipulated must be specified as functions of time in userint.for and

the same name should occur in OCFEINP.DAT, the independent variables enclosed in close brackets, like FI[T] or FI[X,Y].

44. Boundary conditions must not contain the substring 'AT'.

45. If parameter K(T) is associated with diff. equ. of DECLASS=3 then the condition associated with K(T) [VARCLASS = 8 DECLASS = 13] should reflect the condition associated with the diff. equ. of DECLASS=3 e.g. JBF0=1 implies M01=1. Similarly if parameter K(X) is associated with diff. equ. of DECLASS=3 then the condition associated with K(X) [VARCLASS = 9 DECLASS = 14] should reflect the condition associated with the diff. equ. of DECLASS = 3 e.g. JBS0=1 implies M02=1.

46. A parameter definition equation must have the same set of conditions (JJ00,...JBF1) as the associated differential equation
47. Since second-order hyperbolic PDEs require 2 initial conditions, the number of equations is greater than the number of state variables, feasible point may not be found, i.e. the feasible region may be empty.

48. Constant definitions like R = 1.9865 can appear in OCFEINP.DAT.

49. Integral terms can occur only in the objective functional.

50. Boundary conditions can be split only if they are in the form "state variable = constant". The split has to be defined in userint.for and referenced in the model. e.g. in the model F1[T,X] and in userint.for a FUNCTION F1(T,X) should define F1 dependent on T and/or X.

51. State variables fixed at boundary cannot appear in a parameter equation.

52. It is possible to initialize state variables as a function of constants and independent variables if by this initilaization run time can be reduced by giving initial values closer to the optimal values or at least starting closer to the feasible region defined by the constraints. A user-written subroutine INIT2 has to be included in the program file userint.for.

53. It is possible to print out the experimental values when performing parameter estimation. The program PARESTRES has to be compiled, linked and run after the optimization run. The experimantal and estimated values are printed out when control vector parameterization is used to estimate parameters, but the model can only be an initial-value ODE problem. Zero values have to be given in the least-square objective functional, e.g. (A(T)-0)**2 at T = 0.

54. All differential equations must be integrated over the same integration domain.

55. Integration domain has to be fixed before simulation or optimization.

56. Conditional constraints cannot be handled automatically.

57. Boundary and initial conditions must be consistent.

58. Differentiation can be performed only with respect to independent variables.

59. A differential equation must be valid on the whole open integration domain.

60. If a differential equation is not defined at boundary, i.e. there is a division by an independent variable, then there must be a boundary condition for such portion of the boundary.

61. Boundary conditions can be specified only at the boundary of the integration domain.

62. If the differential equation is longer than 79 characters, then & should indicate in the position 79 that the equation is to be continued.

63. A differential equation can be defined over a single point.

The following table shows certain feasible and infeasible combinations.

	OF	I/B	DERIVATI-VE	PARAME-TER EQUATION
TIMEY DELAY	N	N	N	N
FIXED KNOT	N	N	N	N
FIXED BOUNDARY	Y	Y	N	N

Syntax of OCFEGRG2IF.DAT

```
N NFUN NOBJ NALGVAR NALGC NTDIQ NSTATE
TDVRANGE(1)...TDVRANGE(NTDVAR)
STATEVARIND(1)...STATEVARIND(NTDVAR)
TDCRANGE(1)...TDCRANGE(NTDC)
```

Syntax of OCFESQPIF.DAT

```
N M NOBJ MEQ NALGVAR NSTATE
STATEVARIND(1)...STATEVARIND(NTDVAR)
```

```
TDCRANGE(1)...TDCRANGE(NTDC)
```

Syntax of OPTIMINP.DAT

Option 1

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
OUTPUTLEVEL
LOG
if LOG = 1 THEN
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
HJEPSR DR
```

Option 2

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
OUTPUTLEVEL
LOG
if LOG = 1 THEN
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
MAXITER BFGSEPSR HDIFFR
```

Option 6

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
```

```
OUTPUTLEVEL
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
HJEPSR DR PENALTYR FEASIBILITYTOLR
MAXITER BFGSEPSR HDIFFR
HR T0 TN TFOPTION INTEGRALTERMFLAG
OVERALLMASBALANCEFLAG
if NCONTROLVAR > 0 then
LB1(1)...LB1(NCONTROLVAR)
UB1(1)...UB1(NCONTROLVAR)
endif
Y(1)...Y(NTIMEDEPSTATE)
TIMEDEPSVARNAME(1)...TIMEDEPSVARNAME(NTIMEDEPSTATE)
```

Option 8

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
OUTPUTLEVEL
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
HJEPSR DR PENALTYR FEASIBILITYTOLR
HR T0 TN TFOPTION INTEGRALTERMFLAG
OVERALLMASBALANCEFLAG
if NCONTROLVAR > 0 then
LB1(1)...LB1(NCONTROLVAR)
UB1(1)...UB1(NCONTROLVAR)
endif
```

```
Y(1)...Y(NTIMEDEPSTATE)
TIMEDEPSVARNAME(1)...TIMEDEPSVARNAME(NTIMEDEPSTATE)
```

Option 11

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
OUTPUTLEVEL
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
HJEPSR DR PENALTYR FEASIBILITYTOLR
HR T0 TN TFOPTION INTEGRALTERMFLAG
OVERALLMASBALANCEFLAG
if NCONTROLVAR > 0 then
LB1(1)...LB1(NCONTROLVAR)
UB1(1)...UB1(NCONTROLVAR)
endif
Y(1)...Y(NTIMEDEPSTATE)
TIMEDEPSVARNAME(1)...TIMEDEPSVARNAME(NTIMEDEPSTATE)
MISSINGDATA(I,J) J=1...M I=1...N
EXPERDATA(I,J) J=1...M I=1...N
```

Option 12

```
* comment
SELECTION
N NALGSTATE NTIMEDEPSTATE NCONTROLVAR NN M
'PROBLEMNAME'
OUTPUTLEVEL
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
```

```
U(1)...U(N)
endif
DVARNAME(1)...DVARNAME(N)
HJEPSR DR PENALTYR FEASIBILITYTOLR
MAXITER BFGSEPSR HDIFFR
HR T0 TN TFOPTION INTEGRALTERMFLAG
OVERALLMASBALANCEFLAG
if NCONTROLVAR > 0 then
LB1(1)...LB1(NCONTROLVAR)
UB1(1)...UB1(NCONTROLVAR)
endif
Y(1)...Y(NTIMEDEPSTATE)
TIMEDEPSVARNAME(1)...TIMEDEPSVARNAME(NTIMEDEPSTATE)
MISSINGDATA(I,J) J=1...M I=1...N
EXPERDATA(I,J) J=1...M I=1...N
```

Option 15

```
* comment
SELECTION
PNEWT PINIT PSTOP PSPIV PH1EP NSTOP ITLIM LMSER IPR IPN4 IPN5 IPN6
IPER
IDUMP IQUAD LDERIV MODCG IDEFAUL19 IRESTART
if any of PNEWT...IDEFAUL19 = -1 then default valued are used
'PROBLEMNAME'
N NFUN NOBJ
SCALEVAR SCALECON VARSCALELB VARSCALEUB CONSCALEFAC
ICONSTRAINTS
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
VARNAME(1)...VARNAME(N)
BLCON(1)...BLCON(NFUN)
BUCON(1)...BUCON(NFUN)
```

Option 16

```
* comment
SELECTION
PNEWT PINIT PSTOP PSPIV PH1EP NSTOP ITLIM LMSER IPR IPN4 IPN5 IPN6
IPER
```

```
IDUMP IQUAD LDERIV MODCG IDEFAUL19 IRESTART
if any of PNEWT...IDEFAUL19 = -1 then default valued are used
'PROBLEMNAME'
SCALEVAR SCALECON VARSCALELB VARSCALEUB CONSCALEFAC
ICONSTRAINTS

if NALGVAR > 0 then
LOG

if LOG = 1 THEN
IBOUND(1)...IBOUND(NALGVAR)
A(1)... A(NALGVAR)
B(1)...B(NALGVAR)
endif

LOG
if LOG=1 THEN
U(1)...U(NALGVAR)
endif
VARNAME(1)...VARNAME(NALGVAR)
BLCON(1)...BLCON(NALGC)
BUCON(1)...BUCON(NALGC)
endif

NTDVAR NTDC
TDVIV(1)...TDVIV(NTDVAR)
TDVLB(1)...TDVLB(NTDVAR)
TDVUB(1)...TDVUB(NTDVAR)
TDLC(1)...TDLC(NTDC)
TDUC(1)...TDUC(NTDC)
INITIALIZE(1)...INITIALIZE(NTDVAR)
```

Option 17

```
* comment
SELECTION
N M MEQ MAXFUN ACC IPRINT INF IDUMP IRESTART
if INF = -101 or INF = -111 then
W(M+1)...W(M+N)
endif
if INF = -110 or INF = -111 then
W(1)...W(M)
endif
SCALEVAR SCALECON VARSCALELB VARSCALEUB CONSCALEFAC
ICONSTRAINTS
LOG
if LOG = 1 THEN
```

```
IBOUND(1)...IBOUND(N)
A(1)... A(N)
B(1)...B(N)
endif
LOG
if LOG=1 THEN
U(1)...U(N)
endif
VARNAME(1)...VARNAME(N)
HDIFFR VSMALL MAXLIN INFO3
```

Option 18

```
* comment
SELECTION
MAXFUN ACC IPRINT IDUMP IRESTART
SCALEVAR SCALECON VARSCALELB VARSCALEUB CONSCALEFAC
ICONSTRAINTS
if NALGVAR > 0 then
LOG
if LOG = 1 THEN
IBOUND(1)...IBOUND(NALGVAR)
A(1)... A(NALGVAR)
B(1)...B(NALGVAR)
endif
LOG
if LOG=1 THEN
U(1)...U(NALGVAR)
endif
VARNAME(1)...VARNAME(NALGVAR)
endif
HDIFFR VSMALL MAXLIN INFO3
NTDVAR
TDVIV(1)...TDVIV(NTDVAR)
TDVLB(1)...TDVLB(NTDVAR)
TDVUB(1)...TDVUB(NTDVAR)
INITIALIZE(1)...INITIALIZE(NTDVAR)
```

Description of input variables

Name	Range	Default	Description
SELECTION	1-18		selects optimizing option
PROBLEMNAME	1-60		name of the problem
OUTPUTLEVEL	0-3		controls the amount of output if options 1,2,6,8,11 or 12 are

		selected
LOG	0-1	if zero, defaults are assigned to scaling bounds or to initial values of the optimizing variables
A		lower scaling bounds of the optimizing variables
B		upper scaling bounds of the optimizing variables
DVARNAME	1-13	variable names for options 1,2,6,8,11 or 12
MAXITER	> 0	if the iteration count is > MAXITER in BFGS, then BFGS exits
HJEPSR	> 0	if the stepsize is < HJEPSR in HJ, then HJ exits
DR	> 0	initial stepsize for HJ
U		the vector of optimizer variables
EPSDIVR	> 0	played role in superseded options
BFGSEPSR	> 0	if the norm of the gradient vector of the optimizer variables is < BFGSEPSR, then BFGS exits
HDIFFR	> 0	stepsize of numerical diferentiation in BFGS
IBOUND	0-1	if IBOUND(I)=1 then the i-th optimizer variable is bounded
HR	> 0	integration stepsize
T0	> 0	initial value of integration
TN	> 0	final value of integration
TFOPTION	0-1	if TFOPTION=1 then TN is an optimizer variable

INTEGRALTERMFLAG	0-1		if =1 then there is an integral term in the objective functional
OVERALL-MASSBALANCEFLAG	0-1		if =1 then overall mass balance is used to compute the n-th time-dependent state variable
N	> 0		number of optimizer variables
NALGSTATE			used in defunct options
NTIMEDEPSTATE	> 0		number of time-dependent state variables
TIMEDEPSVARNAME			names of time dependent state variables for options 6,8,11 or 12
NCONTROLVAR	> 0		number of time-dependent control variables
PENALTYR	> 0		if a time-dependent control variable violates its constraint, the magnitude of violation is multiplied by this factor
FEASIBILITYTOLR	> 0		tolerance level of a time-dependent control variable constraint violation
LB1			lower bounds on time-dependent control variables
UB1			upper bounds on time-dependent control variables
Y			the vector of time-dependent state variables
NN	> 0		number of experimental values
M	> 0		number of components + 1
MISSINGDATA	0-1		= 1 if experimental value is missing
EXPERDATA			experimantal values including time
PNEWT		0.000001	a constraint is assumed to be

		binding if it is within this epsilon of one of its bounds. If a constraint is not binding and not within its bounds then the constraint is not satisfied
PINIT	0.000001	initial constraint tolerance
PSTOP	0.0001	If the fractional change in the objective is less than PSTOP for NSTOP consecutive iterations, the program will stop. The program will also stop if the Kuhn-Tucker optimality conditions are satisfied within PSTOP.
PSPIV	0.0001	If, in constructing the basis inverse, the absolute value of a prospective pivot element is less than PSPIV, the pivot will be rejected and another pivot element will be sought.
PH1EP	0.	If nonzero, the phase 1 objective is augmented by a multiple of the true objective.
NSTOP	10	If the fractional change in the objective is less than PSTOP for NSTOP consecutive iterations,the program will try some alternative strategies. If these do not produce an objective change greater than PSTOP, the program will stop.
ITLIM	100	If SUBROUTINE NEWTON takes ITLIM iterations without converging satisfactorily (in its attempt to solve the binding constraint equations for the basic variable values), the iterations are stopped and corrective action is taken.
LMSER	100000	If the number of completed one dimensional searches equals LMSER, optimization will terminate.

IPR	0-6	0	controls output level
IPN4		0	controls output level
IPN5		0	controls output level
IPN6		0	controls output level
IPER		0	controls output level
IDUMP	0-1		if=1 then current values of optimizer variables are written on a dump file
IQUAD	0-1	0	method for initial estimates of basic variables for each one dimensional search 0 - tangent vector and linear interpolation will be used 1 - quadratic extrapolation will be used
LDERIV	0-2	1	method for obtaining partial derivative 0 - forward difference approximation is used 1 - central difference approximation is used 2 - user supplied subroutine (PARSH) is used
MODCG	0-5	5	It controls the use of conjugate gradient method. Since MAXHES is set to N, always the quasi-Newton method will be used, its value is irrelevant.
IDEFAUL19		1	if =1 objective will be minimized, otherwise objective will be maximized
IRESTART	0-1		if = 1, program reads the latest values of optimizer variables from a dump file to be used as initial values

N		number of variables
NFUN		number of functions including objective
NOBJ		index of component of vector G in SUBROUTINE GCOMP corres- ponding to the objective function
NALGVAR		number of algebraic variables
NALGC		number of algebraic constraints
NTDIQ		number of NLP constraints derived from time-dependent constraints
NSTATE		number of NLP variables derived from state variables
TDCRANGE		number of NLP constraints generated from a time-dependent model constraint
SCALEVAR	0-1	if =1, variables will be scaled
SCALECON	0-1	if =1, constraints will be scaled
VARSCALELB and VARSCALEUB		variables are scaled into the interval [VARSCALELB, VARSCALEUB]. The GRG2 User Guide recommends that variables should be scaled so that a unit change represents a small but significant change in that variable. It also recommends that all problem functions be scaled to have absolute value less than 100. The VF13AD User Guide recommends that the values of the variables and the derivative vectors of the functions all have magnitude about unity.
CONSCALEFAC	> 0	if a constraint function is less than -CONSCALEFAC then it is set to -CONSCALEFAC. If it is

		greater than CONSCALEFAC then it is set to CONSCALEFAC.
ICONSTRAINTS	0-2	=1 if a variable is bounded and violates bounds it is set to the bound. =2 additionally, the changed values of variables are returned from GCOM or from SQPCONSTRAINTS to the calling subroutine.
INITIALIZE	0-1	if = 1, then the state variable is initialized in INIT2 in userint.for.
VARNAME		variable names
BLCON		lower bounds of functions
BUCON		upper bounds of functions
NTDVAR		number of time-dependent variables, i.e. number of variables dependent on the independent variable.
NTDC		number of time-dependent constraints, i.e. constraints that are not differential equations, but equations or inequalities containing at least 1 variable dependent on the independent variable
TDVIV		initial values of time-dependent variables
TDVLB		lower bounds of time-dependent variables
TDVUB		upper bounds of time-dependent variables
TDLC		lower bounds of time-dependent constraints
TDUC		upper bounds of time-dependent constraints
TDVRANGE		number of NLP variables a state or control variable maps.

STATEVARIND		=1 if the variable is a state variable
M		when option 17 is used, the number of functions excluding objective. When option 18 is used, then number of functions including objective
MEQ		number of equality constraints
MAXFUN		It limits the function and gradient evaluations that are called for.
ACC		It controls the accuracy of the calculation.
IPRINT		controls output level
INF		It should be set to a negative value initially by the user.
HDIFFR		stepsize of numerical differentiation.
VSMALL		used in VE17ED to check convergence and smallishness.
MAXLIN	20	bounds the call of VF13AD in a line search
INFO3		if =1, then VF13AD returns INF=1 instead of INF=3

Examples

Problem 1

$$\min f(x) = \frac{16x_1^2+16x_2^2-8x_1x_2-56x_1-256x_2+991}{15}$$

grg2int.for

```
      SUBROUTINE GCOMP(G,XX)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 G(*),XX(*),A(1000),B(1000),BLVAR(1000),BUVAR(1000),X(1000)
      INTEGER IBOUND(1000)
      COMMON/GRGA/N,NFUN,A,B,BLVAR,BUVAR,IBOUND,NOBJ
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED

 COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
 CALECON
     1,ICONSTRAINTS
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)

      ELSE
      DO 4 I=1,N
      X(I)=XX(I)
    4 CONTINUE
      ENDIF
      IF(ICONSTRAINTS.GT.0)THEN
      ALTERED=.FALSE.
      DO 1 I=1,N
      IF(IBOUND(I).EQ.1)THEN
      IF(X(I).LT.A(I))THEN
      X(I)=A(I)
      ALTERED=.TRUE.
      ENDIF
      IF(X(I).GT.B(I))THEN
      X(I)=B(I)
      ALTERED=.TRUE.
      ENDIF
      ENDIF
    1 CONTINUE
      ENDIF
*     UGRG2TEST1
      G(1)=(16.*X(1)*X(1)+16.*X(2)**2-8.*X(1)*X(2)-56.*X(1)-256.*X(2)+
     1991.)/15.
      IF(SCALECON.EQ.1)CALL CONSTRAINTSSCALING2(G,M,NOBJ)

      IF(ICONSTRAINTS.EQ.2)THEN
      IF(ALTERED)THEN
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLESCALING3(N,X,XX,A,B)
      ELSE
      DO 5 I=1,N
      XX(I)=X(I)
    5 CONTINUE
      ENDIF
      ENDIF
      ENDIF
      RETURN
      END

   OPTIMINP.DAT

* source of example
15
-1. -1. -1. -1. -1. -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 1 0
'UGRG2TEST1.DAT'
2 1 1
1 1 0. 100. 100. 0
1
0 0
0. 0.
100. 100.
1
10. 10.
'X1' 'X2'
0.
100.
```

Problem 2

Sources : 1) Proctor and Gamble Co.

2) Himmelblau problem number 11, p. 406

3) Colville problem number 3, p. 24

$$\min f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

s.t.

$$0 \le 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 92$$
$$90 \le 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \le 110$$
$$20 \le 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \le 25$$
$$78 \le x_1 \le 102$$
$$33 \le x_2 \le 45$$
$$27 \le x_3 \le 45$$
$$27 \le x_4 \le 45$$
$$27 \le x_5 \le 45$$

grg2int.for

```
      SUBROUTINE GCOMP(G,XX)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 G(*),XX(*),A(1000),B(1000),BLVAR(1000),BUVAR(1000),X(1000)
      INTEGER IBOUND(1000)
      COMMON/GRGA/N,NFUN,A,B,BLVAR,BUVAR,IBOUND,NOBJ
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED

COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
CALECON
     1,ICONSTRAINTS
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)

      ELSE
      DO 4 I=1,N
      X(I)=XX(I)
    4 CONTINUE
      ENDIF
      IF(ICONSTRAINTS.GT.0)THEN
      ALTERED=.FALSE.
      DO 1 I=1,N
      IF(IBOUND(I).EQ.1)THEN
      IF(X(I).LT.A(I))THEN
      X(I)=A(I)
      ALTERED=.TRUE.
      ENDIF
      IF(X(I).GT.B(I))THEN
      X(I)=B(I)
      ALTERED=.TRUE.
      ENDIF
      ENDIF
    1 CONTINUE
      ENDIF
*     MICROFICHE
      G(1)=5.3578547*X(3)**2+0.8356891*X(1)*X(5)+37.293239*X(1)-
     140792.141
      G(2)=85.334407+0.0056858*X(2)*X(5)+0.0006262*X(1)*X(4)-
     10.0022053*X(3)*X(5)
      G(3)=80.51249+0.0071317*X(2)*X(5)+0.0029955*X(1)*X(2)+
     10.0021813*X(3)**2
      G(4)=9.300961+0.0047026*X(3)*X(5)+0.0012547*X(1)*X(3)+
     10.0019085*X(3)*X(4)
      IF(SCALECON.EQ.1)CALL CONSTRAINTSSCALING2(G,M,NOBJ)

      IF(ICONSTRAINTS.EQ.2)THEN
      IF(ALTERED)THEN
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLESCALING3(N,X,XX,A,B)
      ELSE
      DO 5 I=1,N
      XX(I)=X(I)
    5 CONTINUE
      ENDIF
      ENDIF
      ENDIF
      RETURN
      END
```

OPTIMINP.DAT

```
* himmelblau problem no 11, p406, colville problem no 3, p24
15
-1. -1. -1. -1. -1. -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 1 0
'GRG2MICROFICHEEX3.DAT'
5 4 1
1 1 0. 100. 100. 1
1
1 1 1 1 1
78. 33. 27. 27. 27.
102. 45. 45. 45. 45.
1
78.62 23.44 31.07 44.18 35.32
'X1' 'X2' 'X3' 'X4' 'X5'
0. 0. 90. 20.
100. 92. 110. 25.
```

Problem 3
Sources: Math. Prog. 14(1978),pp 224
    Math. Prog. Study 16(1983),pp 84

$$\min f(x) = e^{x_1 x_2 x_3 x_4 x_5}$$

s.t.

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$x_2x_3 - 5x_4x_5 = 0$$
$$x_1^3 + x_2^3 + 1 = 0$$

grg2int.for

```
      SUBROUTINE GCOMP(G,XX)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 G(*),XX(*),A(1000),B(1000),BLVAR(1000),BUVAR(1000),X(1000)
      INTEGER IBOUND(1000)
      COMMON/GRGA/N,NFUN,A,B,BLVAR,BUVAR,IBOUND,M,NOBJ
*     GRG2RENFRONLPEX2.DAT
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)
      G(1)=EXP(X(1)*X(2)*X(3)*X(4)*X(5))
      G(2)=X(1)**2+X(2)**2+X(3)**2+X(4)**2+X(5)**2-10.
      G(3)=X(2)*X(3)-5.*X(4)*X(5)
      G(4)=X(1)*X(1)*X(1)+X(2)*X(2)*X(2)+1.
      CALL CONSTRAINTSSCALING2(G,NFUN,NOBJ)
      RETURN
      END
```

OPTIMINP.DAT

```
* renfro phd thesis ex2 p92
15
-1. -1. -1. -1. -1. -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 1 1 0
'GRG2RENFRONLPEX2.DAT'
5 4 1
1 1 0. 100. 100. 0
1
0 0 0 0 0
-10. -10. -10. -10. -10.
10. 10. 10. 10. 10.
1
-1.7 1.6 -1.8 0.76 -0.76
'X1' 'X2' 'X3' 'X4' 'X5'
0. 0. 0. 0.
100. 0. 0. 0.
```

sqpint.for

```
      REAL*8 FUNCTION SQPOF(XX,N)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 X(1000),XX(*),A(1000),B(1000)
      INTEGER*4 IBOUND(1000)
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED
      COMMON/SQPA/A,B,IBOUND

COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
CALECON
     1,ICONSTRAINTS
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)

      ELSE
      DO 4 I=1,N
      X(I)=XX(I)
    4 CONTINUE
      ENDIF
      IF(ICONSTRAINTS.GT.0)THEN
      ALTERED=.FALSE.
      DO 1 I=1,N
      IF(IBOUND(I).EQ.1)THEN
      IF(X(I).LT.A(I))THEN
      X(I)=A(I)
      ALTERED=.TRUE.
      ENDIF
      IF(X(I).GT.B(I))THEN
      X(I)=B(I)
      ALTERED=.TRUE.
      ENDIF
      ENDIF
    1 CONTINUE
      ENDIF
      SQPOF2=EXP(X(1)*X(2)*X(3)*X(4)*X(5))
      IF(SQPOF2.LT.-100.0D0)SQPOF2=-100.
      IF(SQPOF2.GT.100.0D0)SQPOF2=100.
      SQPOF=SQPOF2
      IF(ICONSTRAINTS.EQ.2)THEN
      IF(ALTERED)THEN
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLESCALING3(N,X,XX,A,B)
      ELSE
      DO 5 I=1,N
      XX(I)=X(I)
    5 CONTINUE
      ENDIF
      ENDIF
      ENDIF
      RETURN
      END
      SUBROUTINE SQPCONSTRAINTS(G,XX,N,M)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 G(*),X(1000),XX(*),A(1000),B(1000)
      INTEGER*4 IBOUND(1000)
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED
      COMMON/SQPA/A,B,IBOUND

COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
CALECON
     1,ICONSTRAINTS
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)

      ELSE
      DO 4 I=1,N
      X(I)=XX(I)
    4 CONTINUE
      ENDIF
      IF(ICONSTRAINTS.GT.0)THEN
      ALTERED=.FALSE.
      DO 1 I=1,N
      IF(IBOUND(I).EQ.1)THEN
      IF(X(I).LT.A(I))THEN
      X(I)=A(I)
      ALTERED=.TRUE.
      ENDIF
      IF(X(I).GT.B(I))THEN
      X(I)=B(I)
      ALTERED=.TRUE.
      ENDIF
      ENDIF
    1 CONTINUE
      ENDIF
       G(1)=X(1)**2+X(2)**2+X(3)**2+X(4)**2+(X(5)**2)-10.0
       G(2)=X(2)*X(3)-5.0*X(4)*X(5)
       G(3)=(X(1)**3)+(X(2)**3)+1.0
      IF(SCALECON.EQ.1)CALL CONSTRAINTSSCALING(G,M)

      IF(ICONSTRAINTS.EQ.2)THEN
      IF(ALTERED)THEN
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLESCALING3(N,X,XX,A,B)
      ELSE
      DO 5 I=1,N
      XX(I)=X(I)
    5 CONTINUE
      ENDIF
      ENDIF
      ENDIF
      RETURN
      END
```

OPTIMINP.DAT

```
* renfro phd thesis ex2 p92
17
5 3 3 10000 0.000001 1 1 1 0
1 1 0. 1. 100. 1
1
1 1 1 1 1
-2. -2. -2. -2. -2.
2. 2. 2. 2. 2.
1
-2. 2. 2. -1. -1.
'X1' 'X2' 'X3' 'X4' 'X5'
0.0001 1.0D-10 100000 1
```

Problem 8

$$u_z(z,y) + u_y(z,y) + u_{zz}(z,y) + u_{yy}(z,y) + u_{zy}(z,y) + u(z,y) - 2zy^2 - 2z^2y - 2y^2 - 2z^2 - 4zy - z^2y^2$$

$$u(z,y) = 0 \quad for\ z \in [0,1]\ y = 0$$

$$u(z,y) = 0 \quad for\ y \in [0,1]\ z = 0$$

$$u(z,y) = y^2 \quad for\ y \in [0,1]\ z = 1$$

$$u(z,y) = z^2 \quad for\ z \in [0,1]\ y = 1$$

OCFEINP.DAT

```
* made up by Josef Illes
3 1 2 2 0 0
2 2
2 2
OCFEPDEGRG2TEST1.DAT
0.5 0.5 0.5 0.5
0 0 0 0  0 0 0 1 8 3 0 1
4 1 1
0 0 0 0 1 1 1 1  1  1  1 1
3 6 6 6 6 6 6 6 6 9
0 0 0 0 1 1 1 1  1  1  1 1
'U' 'Z' 'Y'
```

```
1 1 1 1
0. 1. 0. 1. 0. 0. 0. 0.
0.7
0.7
DU(Z,Y)/DZ+DU(Z,Y)/DY+D2U(Z,Y)/DZ2+D2U(Z,Y)/DY2+D2U(Z,Y)/DZDY&
+U(Z,Y)-2*Z*Y**2-2*Z**2*Y-2*Y**2-2*Z**2-4*Z*Y-Z**2*Y**2
U(Z,Y) AT Z=0
U(Z,Y) AT Y=0
U(Z,Y)-Y**2 AT Z=1
U(Z,Y)-Z**2 AT Y=1
U(Z,Y) AT Z=0 AT Y=0
U(Z,Y) AT Z=0 AT Y=1
U(Z,Y) AT Z=1 AT Y=0
U(Z,Y)-1 AT Z=1 AT Y=1
INTEGRAL U(1,Y)DY+INTEGRAL U(Z,1)DZ+INTEGRAL U(Z,Y)DZDY
```

OPTIMINP.DAT

```
* made up by Josef Illes
16
-1. -1. -1. -1. -1. -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 0
'PDEGRG2TEST1.DAT'
1 1 0. 100. 100. 0
4 0
0. 0. 0. 0.
0. 0. 0. 0.
1. 1. 1. 1.
```

OCFEINP.DAT

```
* made up by Josef Illes
3 2 2 2 0 0
2 2
2 2
OCFEPDESQPTEST1.DAT
0.5 0.5 0.5 0.5
0 0 0 0  0 0 0 1 8 3 0 1
4 1 1
0 0 0 0 1 1 1 1  1  1  1 1
3 6 6 6 6 6 6 6 6 9
0 0 0 0 1 1 1 1  1  1  1 1
'U' 'Z' 'Y'
1 1 1 1
0. 1. 0. 1. 0. 0. 0. 0.
0.7
0.7
DU(Z,Y)/DZ+DU(Z,Y)/DY+D2U(Z,Y)/DZ2+D2U(Z,Y)/DY2+D2U(Z,Y)/DZDY&
+U(Z,Y)-2*Z*Y**2-2*Z**2*Y-2*Y**2-2*Z**2-4*Z*Y-Z**2*Y**2
```

```
U(Z,Y) AT Z=0
U(Z,Y) AT Y=0
U(Z,Y)-Y**2 AT Z=1
U(Z,Y)-Z**2 AT Y=1
U(Z,Y) AT Z=0 AT Y=0
U(Z,Y) AT Z=0 AT Y=1
U(Z,Y) AT Z=1 AT Y=0
U(Z,Y)-1 AT Z=1 AT Y=1
INTEGRAL U(1,Y)DY+INTEGRAL U(Z,1)DZ+INTEGRAL U(Z,Y)DZDY
```

OPTIMINP.DAT

```
* made up by Josef Illes
18
100000 0.000001 1 1 0
1 1 0. 1. 100. 1
0.0001 1.0D-10 100000 1
4
0. 0. 0. 0.
0. 0. 0. 0.
1. 1. 1. 1.
```

Problem 9

It is a batch reactor dynamic optimization problem, found as Example 3.3.2 pp 95-99 in W.H. RAY, Advanced Process Control

$$\max_{T(t)} [I = c_2(t)]$$

$$\frac{dc_1(t)}{dt} = -k_1(T)c_1(t)^2 \quad c_1(0) = 1$$

$$\frac{dc_2(t)}{dt} = k_1(T)c_1(t)^2 - k_2(T)c_2(t) \quad c_2(0) = 0$$

$$T_* \le T(t) \le T^*$$

OCFEINP.DAT

```
* ray: advanced process control p95
1 1 4 0 0 0
3 3 3 3
*10 NICP NEEDED WITH ONE SUBINTERVAL
* THIS IS A COMMENT
OCFE1GRG2RENFRODAOPEX1.DAT
0.1 0.1 0.1 0.1
```

```
6 0 0 0 0 0 2 2 2 6 0 1
2 2 11 15 15 1
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 13 13 1 1
4 4 7
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
'X1' 'X2' 'TEM' 'K1' 'K2'
'T'
1 1 1 1
0. 1. 0. 0. 0. 0. 0. 0.
0.1 0.3 0.6
REAL*8 A2(2),E(2),R
R=1.9865
A2(1)=4000.
E(1)=5000.
A2(2)=620000.
E(2)=10000.
K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
DX1(T)/DT+K1(T)*X1(T)**2
DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
X1(T)-1. AT T=0
X2(T) AT T=0
X2(T)

OCFEOUT.DAT

* renfo example 1
  1   1   4   0   0   0
   3    3    3    3
*10 NICP NEEDED WITH ONE SUBINTERVAL

* THIS IS A COMMENT
OCFE1GRG2RENFRODAOPEX1.DAT
  0.10   0.10   0.10   0.10
  6    0    0    0    0    0    2    2    2    6    0    1
  2    2   11   15   15    1
  1    0    0    0    0    0    0    0    0    0    0    0
  1    0    0    0    0    0    0    0    0    0    0    0
  0    0    0    0    0    0   13   13    1    1
  4    4    7
  1    0    0    0    0    0    0    0    0    0    0    0
  1    0    0    0    0    0    0    0    0    0    0    0
  1    0    0    0    0    0    0    0    0    0    0    0
  1   0   0   0   0   0   0   0   0   0   0   0
 X1   X2   TEM  K1   K2
 T
  1   1   1   1
   0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00
*0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
        4          0
        3
        0
.10D+00 .30D+00 .60D+00
*0.05 0.15 0.3 0.6
*0.03 0.1 0.3 0.5 0.75
*0.03 0.1 0.3 0.6 VERY GOOD WITH 4 4 4 4 4 AND 1 0
*0.05 0.15 0.3 0.6 VERY GOOD WITH 33333 AND 2 0

*0.03 0.1 0.3 0.6 VERY GOOD WITH 33333 AND 2 1
*0.001 0.1 0.3 0.6
*0.0125 0.025 0.0375 0.05 0.0625 0.075 0.0875 0.1 0.15 0.2
*0.3 0.4 0.5 0.6 0.7 0.8 0.9
*0.8 0.9
*0.001 0.01 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7
*0.8 0.9
*0.1 0.3 0.6
*0.001 0.01 0.1 0.3 0.6
*0.01 0.05 0.15 0.3 0.6
*0.1 0.2 0.3 0.6
*0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
*0.01 0.03 0.06 0.1 0.15 0.21 0.3 0.5 0.7
*0.5
*0.1 0.3 0.6
*0.05 0.3 0.6
*0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1 1 1 1 1 1
*0.1 0.2 0.3 0.4
* OK BUT NICP 3 2 0.3
* OK BUT NICP 3 2 2 .25 .5
*0.01 0.02 0.05 0.075 0.1 0.15 0.2 0.25 0.35 0.5
*0.7
*OK WITH NICP 2 2 2 2 2 2 0.1 0.2 0.3 0.5 0.7
*OK WITH NICP 2 2 2 2 2 0.1 0.2 0.4 0.7
*OKWITH NICP 2 2 2 2 2 2 2 2 2 2 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
* OK WITH NICP 1 1 1 1 1 1 1 1 1 1 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
REAL*8 A2(2),E(2),R
R=1.9865
A2(1)=4000.
E(1)=5000.
A2(2)=620000.
E(2)=10000.
K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
```

```
 K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
 DX1(T)/DT+K1(T)*X1(T)**2
 DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
 X1(T)-1. AT T=0
 X2(T) AT T=0
 X2(T)
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
-0.10000000D+01
 -0.53237900D+01    0.38729833D+01   0.20655911D+01  -0.12909944D+01
0.67620999D+00
  0.15000000D+01  -0.32274861D+01   0.46259293D-16   0.32274861D+01
-0.15000000D+01
 -0.67620999D+00    0.12909944D+01  -0.20655911D+01  -0.38729833D+01
0.53237900D+01
  0.10000000D+01  -0.18783611D+01   0.26666667D+01  -0.14788306D+02
0.13000000D+02
  0.84000000D+02  -0.12206317D+03   0.58666667D+02  -0.44603500D+02
0.24000000D+02
  0.53237900D+02  -0.73333333D+02   0.26666667D+02  -0.13333333D+02
0.67620999D+01
 -0.60000000D+01    0.16666667D+02  -0.21333333D+02    0.16666667D+02
-0.60000000D+01
  0.67620999D+01  -0.13333333D+02   0.26666667D+02  -0.73333333D+02
0.53237900D+02
  0.24000000D+02  -0.44603500D+02   0.58666667D+02  -0.12206317D+03
0.84000000D+02
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
-0.10000000D+01
 -0.53237900D+01    0.38729833D+01   0.20655911D+01  -0.12909944D+01
0.67620999D+00
  0.15000000D+01  -0.32274861D+01   0.46259293D-16   0.32274861D+01
-0.15000000D+01
 -0.67620999D+00    0.12909944D+01  -0.20655911D+01  -0.38729833D+01
0.53237900D+01
  0.10000000D+01  -0.18783611D+01   0.26666667D+01  -0.14788306D+02
0.13000000D+02
  0.84000000D+02  -0.12206317D+03   0.58666667D+02  -0.44603500D+02
0.24000000D+02
  0.53237900D+02  -0.73333333D+02   0.26666667D+02  -0.13333333D+02
0.67620999D+01
 -0.60000000D+01    0.16666667D+02  -0.21333333D+02    0.16666667D+02
-0.60000000D+01
  0.67620999D+01  -0.13333333D+02   0.26666667D+02  -0.73333333D+02
0.53237900D+02
  0.24000000D+02  -0.44603500D+02   0.58666667D+02  -0.12206317D+03
0.84000000D+02
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
-0.10000000D+01
```

```
 -0.53237900D+01     0.38729833D+01     0.20655911D+01  -0.12909944D+01
0.67620999D+00
  0.15000000D+01   -0.32274861D+01     0.46259293D-16     0.32274861D+01
-0.15000000D+01
 -0.67620999D+00     0.12909944D+01   -0.20655911D+01   -0.38729833D+01
0.53237900D+01
  0.10000000D+01   -0.18783611D+01     0.26666667D+01   -0.14788306D+02
0.13000000D+02
  0.84000000D+02   -0.12206317D+03     0.58666667D+02   -0.44603500D+02
0.24000000D+02
  0.53237900D+02   -0.73333333D+02     0.26666667D+02   -0.13333333D+02
0.67620999D+01
 -0.60000000D+01     0.16666667D+02   -0.21333333D+02     0.16666667D+02
-0.60000000D+01
  0.67620999D+01   -0.13333333D+02     0.26666667D+02   -0.73333333D+02
0.53237900D+02
  0.24000000D+02   -0.44603500D+02     0.58666667D+02   -0.12206317D+03
0.84000000D+02
 -0.13000000D+02     0.14788306D+02   -0.26666667D+01     0.18783611D+01
-0.10000000D+01
 -0.53237900D+01     0.38729833D+01     0.20655911D+01   -0.12909944D+01
0.67620999D+00
  0.15000000D+01   -0.32274861D+01     0.46259293D-16     0.32274861D+01
-0.15000000D+01
 -0.67620999D+00     0.12909944D+01   -0.20655911D+01   -0.38729833D+01
0.53237900D+01
  0.10000000D+01   -0.18783611D+01     0.26666667D+01   -0.14788306D+02
0.13000000D+02
  0.84000000D+02   -0.12206317D+03     0.58666667D+02   -0.44603500D+02
0.24000000D+02
  0.53237900D+02   -0.73333333D+02     0.26666667D+02   -0.13333333D+02
0.67620999D+01
 -0.60000000D+01     0.16666667D+02   -0.21333333D+02     0.16666667D+02
-0.60000000D+01
  0.67620999D+01   -0.13333333D+02     0.26666667D+02   -0.73333333D+02
0.53237900D+02
  0.24000000D+02   -0.44603500D+02     0.58666667D+02   -0.12206317D+03
0.84000000D+02
X1          2         1
X2          2        18
TEM         11        35
K1         15         0
K2         15         0
T           1         0
[          19         0
       0
REAL*8 A2(2),E(2),R
       0
 R=1.9865
         0
 A2(1)=4000.
         0
 E(1)=5000.
         0
 A2(2)=620000.
         0
 E(2)=10000.
         13
 K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
         13
 K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
         1
 DX1(T)/DT+K1(T)*X1(T)**2
         1
 DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
         4
 X1(T)-1.
         4
 X2(T)
         7
 X2(T)
   17   17    4
    1    1    0
  0.00000000D+00     0.11270167D-01     0.50000000D-01     0.88729833D-01
 0.10000000D+00
  0.12254033D+00     0.20000000D+00     0.27745967D+00     0.30000000D+00
 0.33381050D+00
  0.45000000D+00     0.56618950D+00     0.60000000D+00     0.64508067D+00
 0.80000000D+00
  0.95491933D+00  0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
 0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
 0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
 0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
 0.10000000D+01
```

grg2int.for

```
      SUBROUTINE GCOMP(G,XX)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 G(*),XX(*),X(1000),A(1000),B(1000),BLVAR(1000)
      REAL*8 BUVAR(1000)
      INTEGER*4 IBOUND(1000)
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED
      COMMON/GRGA/N,NFUN,A,B,BLVAR,BUVAR,IBOUND,NOBJ

COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
CALECON
     1,ICONSTRAINTS
      REAL*8 A2(2),E(2),R
      REAL*8 K11,K12,K13,K14,K21,K22,K23,K24
      IF(SCALEVAR.EQ.1)THEN
      CALL VARIABLEUNSCALING3(N,XX,X,A,B)

      ELSE
      DO 4 I=1,N
      X(I)=XX(I)
    4 CONTINUE
      ENDIF
      IF(ICONSTRAINTS.GT.0)THEN
      ALTERED=.FALSE.
      DO 1 I=1,N
      IF(IBOUND(I).EQ.1)THEN
      IF(X(I).LT.A(I))THEN
      X(I)=A(I)
      ALTERED=.TRUE.
      ENDIF
      IF(X(I).GT.B(I))THEN
      X(I)=B(I)
      ALTERED=.TRUE.
      ENDIF
      ENDIF
    1 CONTINUE
      ENDIF
      R=1.9865
      A2(1)=4000.
      E(1)=5000.
      A2(2)=620000.
      E(2)=10000.
      K11=A2(1)*EXP(-E(1)/R/X(35))
      K12=A2(1)*EXP(-E(1)/R/X(36))
      K13=A2(1)*EXP(-E(1)/R/X(37))
      K14=A2(1)*EXP(-E(1)/R/X(38))
      K21=A2(2)*EXP(-E(2)/R/X(35))
      K22=A2(2)*EXP(-E(2)/R/X(36))
      K23=A2(2)*EXP(-E(2)/R/X(37))
      K24=A2(2)*EXP(-E(2)/R/X(38))
      G(1)=(-0.5323790007724450D+01*X(1)+0.3872983346207417D+01*X(2)+0.2
X065591117977289D+01*X(3) -0.1290994448735806D+01*X(4)+0.6762099922
X755498D+00*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
X11*X(2)**2
 G(2)=(0.1500000000000000D+01*X(1) -0.3227486121839514D+01*X(2)+0.4
X625929269271486D-16*X(3)+0.3227486121839514D+01*X(4) -0.1500000000
X000000D+01*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
X11*X(3)**2
 G(3)=(-0.6762099922755499D+00*X(1)+0.1290994448735806D+01*X(2) -0.
X2065591117977289D+01*X(3) -0.3872983346207417D+01*X(4)+0.532379000
X7724450D+01*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+
XK11*X(4)**2
 G(4)=(0.1000000000000000D+01*X(1) -0.1878361089654305D+01*X(2)+0.2
X666666666666667D+01*X(3) -0.1478830557701236D+02*X(4)+0.1300000000
X000000D+02*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
X11*X(5)**2
 G(5)=(-0.5323790007724450D+01*X(5)+0.3872983346207417D+01*X(6)+0.2
X065591117977289D+01*X(7) -0.1290994448735806D+01*X(8)+0.6762099922
X755498D+00*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(6)**2
 G(6)=(0.1500000000000000D+01*X(5) -0.3227486121839514D+01*X(6)+0.4
X625929269271486D-16*X(7)+0.3227486121839514D+01*X(8) -0.1500000000
X000000D+01*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(7)**2
 G(7)=(-0.6762099922755499D+00*X(5)+0.1290994448735806D+01*X(6) -0.
X2065591117977289D+01*X(7) -0.3872983346207417D+01*X(8)+0.532379000
X7724450D+01*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+
XK12*X(8)**2
```

```
 G(8)=(0.1000000000000000D+01*X(5) -0.1878361089654305D+01*X(6)+0.2
X6666666666666667D+01*X(7) -0.1478830557701236D+02*X(8)+0.1300000000
X000000D+02*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(9)**2
 G(9)=(-0.5323790007724450D+01*X(9)+0.3872983346207417D+01*X(10)+0.
X2065591117977289D+01*X(11) -0.1290994448735806D+01*X(12)+0.6762099
X922755498D+00*X(13))/(0.6000000000000000D+00-0.3000000000000000D+0
X0)+K13*X(10)**2
 G(10)=(0.1500000000000000D+01*X(9) -0.3227486121839514D+01*X(10)+0
X.4625929269271486D-16*X(11)+0.3227486121839514D+01*X(12) -0.150000
X0000000000D+01*X(13))/(0.6000000000000000D+00-0.3000000000000000D+
X00)+K13*X(11)**2
 G(11)=(-0.6762099922755499D+00*X(9)+0.1290994448735806D+01*X(10) -
X0.2065591117977289D+01*X(11) -0.3872983346207417D+01*X(12)+0.53237
X90007724450D+01*X(13))/(0.6000000000000000D+00-0.3000000000000000D
X+00)+K13*X(12)**2
 G(12)=(0.1000000000000000D+01*X(9) -0.1878361089654305D+01*X(10)+0
X.2666666666666667D+01*X(11) -0.1478830557701236D+02*X(12)+0.130000
X0000000000D+02*X(13))/(0.6000000000000000D+00-0.3000000000000000D+
X00)+K13*X(13)**2
 G(13)=(-0.5323790007724450D+01*X(13)+0.3872983346207417D+01*X(14)+
X0.2065591117977289D+01*X(15) -0.1290994448735806D+01*X(16)+0.67620
X99922755498D+00*X(17))/(0.1000000000000000D+01-0.6000000000000000D
X+00)+K14*X(14)**2
 G(14)=(0.1500000000000000D+01*X(13) -0.3227486121839514D+01*X(14)+
X0.4625929269271486D-16*X(15)+0.3227486121839514D+01*X(16) -0.15000
X00000000000D+01*X(17))/(0.1000000000000000D+01-0.6000000000000000D
```

```
X+00)+K14*X(15)**2
 G(15)=(-0.6762099922755499D+00*X(13)+0.1290994448735806D+01*X(14)
X-0.2065591117977289D+01*X(15) -0.3872983346207417D+01*X(16)+0.5323
X790007724450D+01*X(17))/(0.1000000000000000D+01-0.6000000000000000
XD+00)+K14*X(16)**2
 G(16)=(0.1000000000000000D+01*X(13) -0.1878361089654305D+01*X(14)+
X0.2666666666666667D+01*X(15) -0.1478830557701236D+02*X(16)+0.13000
X00000000000D+02*X(17))/(0.1000000000000000D+01-0.6000000000000000D
X+00)+K14*X(17)**2
 G(17)=(-0.5323790007724450D+01*X(18)+0.3872983346207417D+01*X(19)+
X0.2065591117977289D+01*X(20) -0.1290994448735806D+01*X(21)+0.67620
X99922755498D+00*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(2)**2+K21*X(19)
 G(18)=(0.1500000000000000D+01*X(18) -0.3227486121839514D+01*X(19)+
X0.4625929269271486D-16*X(20)+0.3227486121839514D+01*X(21) -0.15000
X00000000000D+01*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(3)**2+K21*X(20)
 G(19)=(-0.6762099922755499D+00*X(18)+0.1290994448735806D+01*X(19)
X-0.2065591117977289D+01*X(20) -0.3872983346207417D+01*X(21)+0.5323
X790007724450D+01*X(22))/(0.1000000000000000D+00-0.0000000000000000
XD+00)-K11*X(4)**2+K21*X(21)
 G(20)=(0.1000000000000000D+01*X(18) -0.1878361089654305D+01*X(19)+
X0.2666666666666667D+01*X(20) -0.1478830557701236D+02*X(21)+0.13000
X00000000000D+02*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(5)**2+K21*X(22)
 G(21)=(-0.5323790007724450D+01*X(22)+0.3872983346207417D+01*X(23)+
X0.2065591117977289D+01*X(24) -0.1290994448735806D+01*X(25)+0.67620
X99922755498D+00*X(26))/(0.3000000000000000D+00-0.1000000000000000D
```

```
X+00)-K12*X(6)**2+K22*X(23)
 G(22)=(0.1500000000000000D+01*X(22)  -0.3227486121839514D+01*X(23)+
X0.4625929269271486D-16*X(24)+0.3227486121839514D+01*X(25)  -0.15000
X00000000000D+01*X(26))/(0.3000000000000000D+00-0.1000000000000000D
X+00)-K12*X(7)**2+K22*X(24)
 G(23)=(-0.6762099922755499D+00*X(22)+0.1290994448735806D+01*X(23)
X-0.2065591117977289D+01*X(24)  -0.3872983346207417D+01*X(25)+0.5323
X790007724450D+01*X(26))/(0.3000000000000000D+00-0.1000000000000000
XD+00)-K12*X(8)**2+K22*X(25)
 G(24)=(0.1000000000000000D+01*X(22)  -0.1878361089654305D+01*X(23)+
X0.2666666666666667D+01*X(24) -0.1478830557701236D+02*X(25)+0.13000
X00000000000D+02*X(26))/(0.3000000000000000D+00-0.1000000000000000D
X+00)-K12*X(9)**2+K22*X(26)
 G(25)=(-0.5323790007724450D+01*X(26)+0.3872983346207417D+01*X(27)+
X0.2065591117977289D+01*X(28) -0.1290994448735806D+01*X(29)+0.67620
X99922755498D+00*X(30))/(0.6000000000000000D+00-0.3000000000000000D
X+00)-K13*X(10)**2+K23*X(27)
 G(26)=(0.1500000000000000D+01*X(26)  -0.3227486121839514D+01*X(27)+
X0.4625929269271486D-16*X(28)+0.3227486121839514D+01*X(29)  -0.15000
X00000000000D+01*X(30))/(0.6000000000000000D+00-0.3000000000000000D
X+00)-K13*X(11)**2+K23*X(28)
 G(27)=(-0.6762099922755499D+00*X(26)+0.1290994448735806D+01*X(27)
X-0.2065591117977289D+01*X(28)  -0.3872983346207417D+01*X(29)+0.5323
X790007724450D+01*X(30))/(0.6000000000000000D+00-0.3000000000000000
XD+00)-K13*X(12)**2+K23*X(29)
 G(28)=(0.1000000000000000D+01*X(26)  -0.1878361089654305D+01*X(27)+
X0.2666666666666667D+01*X(28) -0.1478830557701236D+02*X(29)+0.13000
```

```
 X00000000000D+02*X(30))/(0.6000000000000000D+00-0.3000000000000000D
 X+00)-K13*X(13)**2+K23*X(30)
  G(29)=(-0.5323790007724450D+01*X(30)+0.3872983346207417D+01*X(31)+
 X0.2065591117977289D+01*X(32) -0.1290994448735806D+01*X(33)+0.67620
 X99922755498D+00*X(34))/(0.1000000000000000D+01-0.6000000000000000D
 X+00)-K14*X(14)**2+K24*X(31)
  G(30)=(0.1500000000000000D+01*X(30)  -0.3227486121839514D+01*X(31)+
 X0.4625929269271486D-16*X(32)+0.3227486121839514D+01*X(33)  -0.15000
 X00000000000D+01*X(34))/(0.1000000000000000D+01-0.6000000000000000D
 X+00)-K14*X(15)**2+K24*X(32)
  G(31)=(-0.6762099922755499D+00*X(30)+0.1290994448735806D+01*X(31)
 X-0.2065591117977289D+01*X(32)  -0.3872983346207417D+01*X(33)+0.5323
 X790007724450D+01*X(34))/(0.1000000000000000D+01-0.6000000000000000
 XD+00)-K14*X(16)**2+K24*X(33)
  G(32)=(0.1000000000000000D+01*X(30)  -0.1878361089654305D+01*X(31)+
 X0.2666666666666667D+01*X(32) -0.1478830557701236D+02*X(33)+0.13000
 X00000000000D+02*X(34))/(0.1000000000000000D+01-0.6000000000000000D
 X+00)-K14*X(17)**2+K24*X(34)
  G(33)=X(1)-1.
  G(34)=X(18)
  G(35)=X(34)
  IF(SCALECON.EQ.1)CALL CONSTRAINTSSCALING2(G,NFUN,NOBJ)

  IF(ICONSTRAINTS.EQ.2)THEN
  IF(ALTERED)THEN
  IF(SCALEVAR.EQ.1)THEN
  CALL VARIABLESCALING3(N,X,XX,A,B)
  ELSE
  DO 5 I=1,N
  XX(I)=X(I)
5 CONTINUE
  ENDIF
  ENDIF
  ENDIF
  RETURN
```

```
      END
```

OCFEGRG2IF.DAT

```
 38  35  35   0   0   0  34
  17   17    4
   1    1    0
```

PLOTPREP.DAT

```
 1
 3
 1 1 1
  0.00000000D+00   0.10000000D+01   0.00000000D+00   0.00000000D+00
```

OPTIMINP.DAT

```
* ray: advanced process control p95
16
0.000001 0.0001 -1. -1. -1. 100 1000 1000000 1 -1 -1 -1 -1 1 -1 -1 -1 2 0
'GRG2RENFRODAOPEX1.DAT'
1 1 0. 100. 100. 0
3 0
1. 0. 390.
0. 0. 298.
1. 1. 398.
```

OCFEINP.DAT

```
* ray: advanced process control p95
1 2 4 0 0 0
3 3 3 3 3
*10 NICP NEEDED WITH ONE SUBINTERVAL
* THIS IS A COMMENT
OCFE1SQPRENFRODAOPEX1.DAT
0.1 0.1 0.1 0.1
6 0 0 0 0 2 2 2 2 6 0 1
2 2 11 15 15 1
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 21 21 13 13
1 1 4 4 7
1 0 0 0 0 0 0 0 0 0 0 0
```

```
 1 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0
 'X1' 'X2' 'TEM' 'K1' 'K2'
 'T'
 1 1 1 1
 0. 1. 0. 0. 0. 0. 0. 0.
 0.1 0.3 0.6
 REAL*8 A2(2),E(2),R
 R=1.9865
 A2(1)=4000.
 E(1)=5000.
 A2(2)=620000.
 E(2)=10000.
 398.-TEM(T)
 TEM(T)-298.
 K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
 K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
 DX1(T)/DT+K1(T)*X1(T)**2
 DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
 X1(T)-1. AT T=0
 X2(T) AT T=0
 -X2(T)
```

OCFEOUT.DAT

```
* renfro example 1
  1  2  4  0  0  0
   3   3   3   3
*10 NICP NEEDED WITH ONE SUBINTERVAL

* THIS IS A COMMENT
OCFE1SQPRENFRODAOPEX1.DAT
   0.10    0.10    0.10    0.10
   6   0   0   0   0   2   2   2   2   6   0   1
   2   2  11  15  15   1
   1   0   0   0   0   0   0   0   0   0   0   0
   1   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0  21  21  13  13
   1   1   4   4   7
   1   0   0   0   0   0   0   0   0   0   0   0
   1   0   0   0   0   0   0   0   0   0   0   0
   1   0   0   0   0   0   0   0   0   0   0   0
   1   0   0   0   0   0   0   0   0   0   0   0
X1  X2  TEM  K1  K2
T
  1   1   1   1
```

```
   0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00
 *0.05 0.15 0.3 0.6
         4         0
         3
         0
 .10D+00 .30D+00 .60D+00
 *0.0000001 0.000001 0.00001 0.0001 0.001 0.01 0.03 0.1 0.3 0.5
 *0.75
 *0.03 0.1 0.3 0.5 0.75
 *0.05 0.15 0.3 0.6
 *0.1 0.3 0.6
 *0.001 0.003 0.006 0.01 0.03 0.06 0.1 0.3 0.6
 *0.05 0.3 0.6
 *0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1 1 1 1 1 1
 *0.1 0.2 0.3 0.4
 * OK BUT NICP 3 2 0.3
 * OK BUT NICP 3 2 2 .25 .5
 *0.01 0.02 0.05 0.075 0.1 0.15 0.2 0.25 0.35 0.5
 *0.7
 *OK WITH NICP 2 2 2 2 2 2 0.1 0.2 0.3 0.5 0.7
 *OK WITH NICP 2 2 2 2 2 0.1 0.2 0.4 0.7
 *OKWITH NICP 2 2 2 2 2 2 2 2 2 2 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
 * OK WITH NICP 1 1 1 1 1 1 1 1 1 1 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
 REAL*8 A2(2),E(2),R
 R=1.9865
 A2(1)=4000.
 E(1)=5000.
 A2(2)=620000.
 E(2)=10000.
 398.-TEM(T)
 TEM(T)-298.
 K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
 K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
 DX1(T)/DT+K1(T)*X1(T)**2
 DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
 X1(T)-1. AT T=0
 X2(T) AT T=0
 -X2(T)
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
-0.10000000D+01
 -0.53237900D+01    0.38729833D+01    0.20655911D+01  -0.12909944D+01
0.67620999D+00
  0.15000000D+01  -0.32274861D+01    0.46259293D-16    0.32274861D+01
-0.15000000D+01
 -0.67620999D+00    0.12909944D+01  -0.20655911D+01  -0.38729833D+01
0.53237900D+01
  0.10000000D+01  -0.18783611D+01    0.26666667D+01  -0.14788306D+02
0.13000000D+02
```

```
  0.84000000D+02  -0.12206317D+03    0.58666667D+02  -0.44603500D+02
 0.24000000D+02
  0.53237900D+02  -0.73333333D+02    0.26666667D+02  -0.13333333D+02
 0.67620999D+01
 -0.60000000D+01    0.16666667D+02  -0.21333333D+02    0.16666667D+02
 -0.60000000D+01
  0.67620999D+01  -0.13333333D+02    0.26666667D+02  -0.73333333D+02
 0.53237900D+02
  0.24000000D+02  -0.44603500D+02    0.58666667D+02  -0.12206317D+03
 0.84000000D+02
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
 -0.10000000D+01
 -0.53237900D+01    0.38729833D+01    0.20655911D+01  -0.12909944D+01
 0.67620999D+00
  0.15000000D+01  -0.32274861D+01    0.46259293D-16    0.32274861D+01
 -0.15000000D+01
 -0.67620999D+00    0.12909944D+01  -0.20655911D+01  -0.38729833D+01
 0.53237900D+01
  0.10000000D+01  -0.18783611D+01    0.26666667D+01  -0.14788306D+02
 0.13000000D+02
  0.84000000D+02  -0.12206317D+03    0.58666667D+02  -0.44603500D+02
 0.24000000D+02
  0.53237900D+02  -0.73333333D+02    0.26666667D+02  -0.13333333D+02
 0.67620999D+01
 -0.60000000D+01    0.16666667D+02  -0.21333333D+02    0.16666667D+02
 -0.60000000D+01
  0.67620999D+01  -0.13333333D+02    0.26666667D+02  -0.73333333D+02
 0.53237900D+02
  0.24000000D+02  -0.44603500D+02    0.58666667D+02  -0.12206317D+03
 0.84000000D+02
 -0.13000000D+02    0.14788306D+02  -0.26666667D+01    0.18783611D+01
 -0.10000000D+01
 -0.53237900D+01    0.38729833D+01    0.20655911D+01  -0.12909944D+01
 0.67620999D+00
  0.15000000D+01  -0.32274861D+01    0.46259293D-16    0.32274861D+01
 -0.15000000D+01
 -0.67620999D+00    0.12909944D+01  -0.20655911D+01  -0.38729833D+01
 0.53237900D+01
  0.10000000D+01  -0.18783611D+01    0.26666667D+01  -0.14788306D+02
 0.13000000D+02
  0.84000000D+02  -0.12206317D+03    0.58666667D+02  -0.44603500D+02
 0.24000000D+02
  0.53237900D+02  -0.73333333D+02    0.26666667D+02  -0.13333333D+02
 0.67620999D+01
 -0.60000000D+01    0.16666667D+02  -0.21333333D+02    0.16666667D+02
 -0.60000000D+01
  0.67620999D+01  -0.13333333D+02    0.26666667D+02  -0.73333333D+02
 0.53237900D+02
```

```
  0.24000000D+02   -0.44603500D+02      0.58666667D+02   -0.12206317D+03
0.84000000D+02
 -0.13000000D+02      0.14788306D+02   -0.26666667D+01      0.18783611D+01
-0.10000000D+01
 -0.53237900D+01      0.38729833D+01      0.20655911D+01   -0.12909944D+01
0.67620999D+00
  0.15000000D+01   -0.32274861D+01      0.46259293D-16      0.32274861D+01
-0.15000000D+01
 -0.67620999D+00      0.12909944D+01   -0.20655911D+01   -0.38729833D+01
0.53237900D+01
  0.10000000D+01   -0.18783611D+01      0.26666667D+01   -0.14788306D+02
0.13000000D+02
  0.84000000D+02   -0.12206317D+03      0.58666667D+02   -0.44603500D+02
0.24000000D+02
  0.53237900D+02   -0.73333333D+02      0.26666667D+02   -0.13333333D+02
0.67620999D+01
 -0.60000000D+01      0.16666667D+02   -0.21333333D+02      0.16666667D+02
-0.60000000D+01
  0.67620999D+01   -0.13333333D+02      0.26666667D+02   -0.73333333D+02
0.53237900D+02
  0.24000000D+02   -0.44603500D+02      0.58666667D+02   -0.12206317D+03
0.84000000D+02
X1         2        1
X2         2        18
TEM        11        35
K1         15        0
K2         15        0
T          1        0
[         19        0
       0
REAL*8 A2(2),E(2),R
       0
R=1.9865
       0
A2(1)=4000.
       0
E(1)=5000.
       0
A2(2)=620000.
       0
E(2)=10000.
       21
398.-TEM(T)
       21
TEM(T)-298.
       13
K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
       13
 K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
        1
 DX1(T)/DT+K1(T)*X1(T)**2
        1
 DX2(T)/DT-K1(T)*X1(T)**2+K2(T)*X2(T)
        4
 X1(T)-1.
        4
 X2(T)
        7
 -X2(T)
  17  17   4
   1   1   0
  0.00000000D+00     0.11270167D-01     0.50000000D-01     0.88729833D-01
0.10000000D+00
  0.12254033D+00     0.20000000D+00     0.27745967D+00     0.30000000D+00
0.33381050D+00
  0.45000000D+00     0.56618950D+00     0.60000000D+00     0.64508067D+00
0.80000000D+00
  0.95491933D+00  0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
0.10000000D+01
  0.00000000D+00     0.11270167D+00     0.50000000D+00     0.88729833D+00
0.10000000D+01
```

sqpint.for

```
      REAL*8 FUNCTION SQPOF(X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      DIMENSION X(*)
      SQPOF=0
      RETURN
      END
      SUBROUTINE SQPCONSTRAINTS(G,XX,N,M,NOBJ)

      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 G(*),X(1000),XX(*),A(1000),B(1000)
      INTEGER*4 IBOUND(1000)
      INTEGER*4 SCALEVAR,SCALECON
      LOGICAL ALTERED
      COMMON/SQPA/A,B,IBOUND

COMMON/SCX/VARSCALELB,VARSCALEUB,CONSCALEFAC,SCALEVAR,S
CALECON
```

```
   1,ICONSTRAINTS
    REAL*8 A2(2),E(2),R
    REAL*8 K11,K12,K13,K14,K21,K22,K23,K24
    IF(SCALEVAR.EQ.1)THEN
    CALL VARIABLEUNSCALING3(N,XX,X,A,B)

    ELSE
    DO 4 I=1,N
    X(I)=XX(I)
  4 CONTINUE
    ENDIF
    IF(ICONSTRAINTS.GT.0)THEN
    ALTERED=.FALSE.
    DO 1 I=1,N
    IF(IBOUND(I).EQ.1)THEN
    IF(X(I).LT.A(I))THEN
    X(I)=A(I)
    ALTERED=.TRUE.
    ENDIF
    IF(X(I).GT.B(I))THEN
    X(I)=B(I)
    ALTERED=.TRUE.
    ENDIF
    ENDIF
  1 CONTINUE
    ENDIF
    R=1.9865
    A2(1)=4000.
    E(1)=5000.
    A2(2)=620000.
    E(2)=10000.
    K11=A2(1)*EXP(-E(1)/R/X(35))
    K12=A2(1)*EXP(-E(1)/R/X(36))
    K13=A2(1)*EXP(-E(1)/R/X(37))
    K14=A2(1)*EXP(-E(1)/R/X(38))
    K21=A2(2)*EXP(-E(2)/R/X(35))
    K22=A2(2)*EXP(-E(2)/R/X(36))
    K23=A2(2)*EXP(-E(2)/R/X(37))
    K24=A2(2)*EXP(-E(2)/R/X(38))
    G(1)=(-0.5323790007724450D+01*X(1)+0.3872983346207417D+01*X(2)+0.2
   X065591117977289D+01*X(3) -0.1290994448735806D+01*X(4)+0.6762099922
   X755498D+00*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
   X11*X(2)**2
    G(2)=(0.1500000000000000D+01*X(1) -0.3227486121839514D+01*X(2)+0.4
X625929269271486D-16*X(3)+0.3227486121839514D+01*X(4) -0.1500000000
X000000D+01*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
X11*X(3)**2
 G(3)=(-0.6762099922755499D+00*X(1)+0.1290994448735806D+01*X(2) -0.
X2065591117977289D+01*X(3) -0.3872983346207417D+01*X(4)+0.532379000
X7724450D+01*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+
XK11*X(4)**2
 G(4)=(0.1000000000000000D+01*X(1) -0.1878361089654305D+01*X(2)+0.2
X666666666666667D+01*X(3) -0.1478830557701236D+02*X(4)+0.1300000000
X000000D+02*X(5))/(0.1000000000000000D+00-0.0000000000000000D+00)+K
X11*X(5)**2
 G(5)=(-0.5323790007724450D+01*X(5)+0.3872983346207417D+01*X(6)+0.2
X065591117977289D+01*X(7) -0.1290994448735806D+01*X(8)+0.6762099922
X755498D+00*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(6)**2
 G(6)=(0.1500000000000000D+01*X(5) -0.3227486121839514D+01*X(6)+0.4
X625929269271486D-16*X(7)+0.3227486121839514D+01*X(8) -0.1500000000
X000000D+01*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(7)**2
 G(7)=(-0.6762099922755499D+00*X(5)+0.1290994448735806D+01*X(6) -0.
X2065591117977289D+01*X(7) -0.3872983346207417D+01*X(8)+0.532379000
X7724450D+01*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+
XK12*X(8)**2
 G(8)=(0.1000000000000000D+01*X(5) -0.1878361089654305D+01*X(6)+0.2
X666666666666667D+01*X(7) -0.1478830557701236D+02*X(8)+0.1300000000
X000000D+02*X(9))/(0.3000000000000000D+00-0.1000000000000000D+00)+K
X12*X(9)**2
 G(9)=(-0.5323790007724450D+01*X(9)+0.3872983346207417D+01*X(10)+0.
X2065591117977289D+01*X(11) -0.1290994448735806D+01*X(12)+0.6762099
X922755498D+00*X(13))/(0.6000000000000000D+00-0.3000000000000000D+0
X0)+K13*X(10)**2
 G(10)=(0.1500000000000000D+01*X(9)  -0.3227486121839514D+01*X(10)+0
X.4625929269271486D-16*X(11)+0.3227486121839514D+01*X(12) -0.150000
X0000000000D+01*X(13))/(0.6000000000000000D+00-0.3000000000000000D+
X00)+K13*X(11)**2
 G(11)=(-0.6762099922755499D+00*X(9)+0.1290994448735806D+01*X(10) -
X0.2065591117977289D+01*X(11) -0.3872983346207417D+01*X(12)+0.53237
X90007724450D+01*X(13))/(0.6000000000000000D+00-0.3000000000000000D
X+00)+K13*X(12)**2
 G(12)=(0.1000000000000000D+01*X(9)  -0.1878361089654305D+01*X(10)+0
X.2666666666666667D+01*X(11) -0.1478830557701236D+02*X(12)+0.130000
X0000000000D+02*X(13))/(0.6000000000000000D+00-0.3000000000000000D+
X00)+K13*X(13)**2
 G(13)=(-0.5323790007724450D+01*X(13)+0.3872983346207417D+01*X(14)+
X0.2065591117977289D+01*X(15) -0.1290994448735806D+01*X(16)+0.67620
X99922755498D+00*X(17))/(0.1000000000000000D+01-0.6000000000000000D
X+00)+K14*X(14)**2
 G(14)=(0.1500000000000000D+01*X(13)  -0.3227486121839514D+01*X(14)+
X0.4625929269271486D-16*X(15)+0.3227486121839514D+01*X(16)  -0.15000
X00000000000D+01*X(17))/(0.1000000000000000D+01-0.6000000000000000D
X+00)+K14*X(15)**2
 G(15)=(-0.6762099922755499D+00*X(13)+0.1290994448735806D+01*X(14)
X-0.2065591117977289D+01*X(15)  -0.3872983346207417D+01*X(16)+0.5323
X790007724450D+01*X(17))/(0.1000000000000000D+01-0.6000000000000000
XD+00)+K14*X(16)**2
```

```
 G(16)=(0.1000000000000000D+01*X(13)  -0.1878361089654305D+01*X(14)+
X0.2666666666666667D+01*X(15) -0.1478830557701236D+02*X(16)+0.13000
X00000000000D+02*X(17))/(0.1000000000000000D+01-0.6000000000000000D
X+00)+K14*X(17)**2
 G(17)=(-0.5323790007724450D+01*X(18)+0.3872983346207417D+01*X(19)+
X0.2065591117977289D+01*X(20) -0.1290994448735806D+01*X(21)+0.67620
X99922755498D+00*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(2)**2+K21*X(19)
 G(18)=(0.1500000000000000D+01*X(18)  -0.3227486121839514D+01*X(19)+
X0.4625929269271486D-16*X(20)+0.3227486121839514D+01*X(21)  -0.15000
X00000000000D+01*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(3)**2+K21*X(20)
 G(19)=(-0.6762099922755499D+00*X(18)+0.1290994448735806D+01*X(19)
X-0.2065591117977289D+01*X(20)  -0.3872983346207417D+01*X(21)+0.5323
X790007724450D+01*X(22))/(0.1000000000000000D+00-0.0000000000000000
XD+00)-K11*X(4)**2+K21*X(21)
 G(20)=(0.1000000000000000D+01*X(18)  -0.1878361089654305D+01*X(19)+
X0.2666666666666667D+01*X(20) -0.1478830557701236D+02*X(21)+0.13000
X00000000000D+02*X(22))/(0.1000000000000000D+00-0.0000000000000000D
X+00)-K11*X(5)**2+K21*X(22)
 G(21)=(-0.5323790007724450D+01*X(22)+0.3872983346207417D+01*X(23)+
X0.2065591117977289D+01*X(24) -0.1290994448735806D+01*X(25)+0.67620
X99922755498D+00*X(26))/(0.3000000000000000D+00-0.1000000000000000D
X+00)-K12*X(6)**2+K22*X(23)
 G(22)=(0.1500000000000000D+01*X(22)  -0.3227486121839514D+01*X(23)+
X0.4625929269271486D-16*X(24)+0.3227486121839514D+01*X(25)  -0.15000
X00000000000D+01*X(26))/(0.3000000000000000D+00-0.1000000000000000D
```

```
X+00)-K12*X(7)**2+K22*X(24)
 G(23)=(-0.6762099922755499D+00*X(22)+0.1290994448735806D+01*X(23)
X-0.2065591117977289D+01*X(24) -0.3872983346207417D+01*X(25)+0.5323
X790007724450D+01*X(26))/(0.3000000000000000D+00-0.1000000000000000
XD+00)-K12*X(8)**2+K22*X(25)
 G(24)=(0.1000000000000000D+01*X(22) -0.1878361089654305D+01*X(23)+
X0.2666666666666667D+01*X(24) -0.1478830557701236D+02*X(25)+0.13000
X00000000000D+02*X(26))/(0.3000000000000000D+00-0.1000000000000000D
X+00)-K12*X(9)**2+K22*X(26)
 G(25)=(-0.5323790007724450D+01*X(26)+0.3872983346207417D+01*X(27)+
X0.2065591117977289D+01*X(28) -0.1290994448735806D+01*X(29)+0.67620
X99922755498D+00*X(30))/(0.6000000000000000D+00-0.3000000000000000D
X+00)-K13*X(10)**2+K23*X(27)
 G(26)=(0.1500000000000000D+01*X(26) -0.3227486121839514D+01*X(27)+
X0.4625929269271486D-16*X(28)+0.3227486121839514D+01*X(29) -0.15000
X00000000000D+01*X(30))/(0.6000000000000000D+00-0.3000000000000000D
X+00)-K13*X(11)**2+K23*X(28)
 G(27)=(-0.6762099922755499D+00*X(26)+0.1290994448735806D+01*X(27)
X-0.2065591117977289D+01*X(28) -0.3872983346207417D+01*X(29)+0.5323
X790007724450D+01*X(30))/(0.6000000000000000D+00-0.3000000000000000
XD+00)-K13*X(12)**2+K23*X(29)
 G(28)=(0.1000000000000000D+01*X(26) -0.1878361089654305D+01*X(27)+
X0.2666666666666667D+01*X(28) -0.1478830557701236D+02*X(29)+0.13000
X00000000000D+02*X(30))/(0.6000000000000000D+00-0.3000000000000000D
X+00)-K13*X(13)**2+K23*X(30)
 G(29)=(-0.5323790007724450D+01*X(30)+0.3872983346207417D+01*X(31)+
X0.2065591117977289D+01*X(32) -0.1290994448735806D+01*X(33)+0.67620
X99922755498D+00*X(34))/(0.1000000000000000D+01-0.6000000000000000D
```

```
X+00)-K14*X(14)**2+K24*X(31)
 G(30)=(0.1500000000000000D+01*X(30)  -0.3227486121839514D+01*X(31)+

X0.4625929269271486D-16*X(32)+0.3227486121839514D+01*X(33)  -0.15000

X00000000000D+01*X(34))/(0.1000000000000000D+01-0.6000000000000000D

X+00)-K14*X(15)**2+K24*X(32)
 G(31)=(-0.6762099922755499D+00*X(30)+0.1290994448735806D+01*X(31)

X-0.2065591117977289D+01*X(32)  -0.3872983346207417D+01*X(33)+0.5323

X790007724450D+01*X(34))/(0.1000000000000000D+01-0.6000000000000000

XD+00)-K14*X(16)**2+K24*X(33)
 G(32)=(0.1000000000000000D+01*X(30)  -0.1878361089654305D+01*X(31)+

X0.2666666666666667D+01*X(32) -0.1478830557701236D+02*X(33)+0.13000

X00000000000D+02*X(34))/(0.1000000000000000D+01-0.6000000000000000D

X+00)-K14*X(17)**2+K24*X(34)
 G(33)=X(1)-1.
 G(34)=X(18)
 G(35)=-X(34)
 G(36)=398.-X(35)
 G(37)=398.-X(36)
 G(38)=398.-X(37)
 G(39)=398.-X(38)
 G(40)=X(35)-298.
 G(41)=X(36)-298.
 G(42)=X(37)-298.
 G(43)=X(38)-298.
 IF(SCALECON.EQ.1)CALL CONSTRAINTSSCALING2(G,M,NOBJ)

 IF(ICONSTRAINTS.EQ.2)THEN
 IF(ALTERED)THEN
 IF(SCALEVAR.EQ.1)THEN
 CALL VARIABLESCALING3(N,X,XX,A,B)
 ELSE
 DO 5 I=1,N
 XX(I)=X(I)
5 CONTINUE
 ENDIF
 ENDIF
 ENDIF
 RETURN
```

```
      END
```

OCFEGRG2IF.DAT

```
  38  43  35  34   0  34
   17   17    4
    1    1    0
```

PLOTPREP.DAT

```
 1
 3
 1 1 1
   0.00000000D+00   0.10000000D+01   0.00000000D+00   0.00000000D+00
```

OPTIMINP.DAT

```
* ray: advanced process control p95
18
100000 0.0001 1 1 0
1 1 0. 1. 100. 0
0.0001 1.0D-10 100000 1
3
1. 0. 390.
0. 0. 298.
1. 1. 393.
0 0 0
```

usersubs.for

```
      FUNCTIONF(UU,X,U,A,B,MISSINGDATA,EXPERDATA,COMPUTEDDATA)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      INTEGER*4 SELECTION
      INTEGER*4 DATAPOINTER
      INTEGER*4 MISSINGDATA(NN,M)
      REAL*8 EXPERDATA(NN,M),COMPUTEDDATA(NN,M)
      REAL*8 X(*),U(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/H/T,H
      COMMON/L/DATAPOINTER
      CALL UNSCALE3(N,UU,U,A,B)
*     UTEST1
*     CVPRENFRODAOPEX1 & DAPCVPTEST1
       F=-X(2)
*      SIMUSOLV
      RETURN
      END
      SUBROUTINE EC(UU,U,A,B,X,F,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR NLPS
*     SELECTION 3,4
*     NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 X(*),F(*),U(*),Y(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      CALL UNSCALE3(N,UU,U,A,B)
*     RENFRONLPEX2
      F(1)=X(1)**2+X(2)**2+X(3)**2+U(1)**2+U(2)**2-10.
      F(2)=X(2)*X(3)-5.*U(1)*U(2)
      F(3)=X(1)*X(1)*X(1)+X(2)*X(2)*X(2)+1.
*      MICROFICHE
      RETURN
      END
      SUBROUTINE EC2(UU,U,A,B,X,F,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR DAOPS
*     SELECTION 9,10,13,14
*     NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 X(*),F(*),U(*),UU(*),A(*),B(*)
      REAL*8 Y(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      CALL UNSCALE3(N,UU,U,A,B)
*     DAPCVPTEST1
*      F(1)=Y(1)+Y(2)+X(1)-1.
*     SIMUSOLV
      F(1)=0.75-Y(1)-Y(2)-X(1)
      RETURN
      END
      REAL*4 FUNCTION IC(UU,U,A,B,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR NLPS
*     SELECTION 3,4
      INTEGER*4 SELECTION
*     NOT INVOKED WITH THIS OPTION
      REAL*8 U(*),X(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
```

```
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      CALL UNSCALE3(N,UU,U,A,B)
      P=0.
*       MICROFICHE
      IC=P
      RETURN
      END
      REAL*4 FUNCTION IC2(UU,U,A,B,Y,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR DAOPS
*     SELECTION 9,10,13,14
      INTEGER*4 SELECTION
*     NOT INVOKED WITH THIS OPTION
      REAL*8 U(*),X(*),Y(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      CALL UNSCALE3(N,UU,U,A,B)
      P=0.
      IC2=P
      RETURN
      END
      FUNCTION DX2(I,Y,TIME,UU,U,A2,B,LB1,UB1,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 Y(*),U(*),C(5),E(5),K(5),A(2)
      REAL*8 LB1(*),UB1(*),UU(*),A2(*),B(*)
      INTEGER*4 SELECTION,TFOPTION
      LOGICAL NOTCALLEDFROMNUMDIFF
      REAL*8 X(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      COMMON/D/TFOPTION,TF
      COMMON/I/NOTCALLEDFROMNUMDIFF
      CALL UNSCALE3(N,UU,U,A2,B)
*     RENFRODAOPEX1 WITH TFOPTION = 1
      IF(TFOPTION.EQ.1)THEN
      IF(U(4).LT.0.)THEN
      PENALTYTERM=PENALTYTERM-U(4)*PENALTY
      ENDIF
      ENDIF
*     RENFRODAOPEX1 & DAPCVPTEST1
      R=1.9865
      A(1)=4000.
      A(2)=620000.
      E(1)=5000.
      E(2)=10000.
*      T=U(1)+U(2)*EXP(-U(3)*TIME)
```

```
      T=U(1)+U(2)*TIME+U(3)*TIME*TIME
      IF(T.LT.LB1(1))THEN
      PENALTYTERM=PENALTYTERM+(LB1(1)-T)*PENALTY
      IF(NOTCALLEDFROMNUMDIFF)T=LB1(1)
      ENDIF
      IF(T.GT.UB1(1))THEN
      PENALTYTERM=PENALTYTERM+(T-UB1(1))*PENALTY
      IF(NOTCALLEDFROMNUMDIFF)T=UB1(1)
      ENDIF
      DO 4 J=1,2
      K(J)=A(J)*EXP(-E(J)/(R*T))
*     ROSENBROCK
*      K(J)=C(J)*EXP(-E(J)*(1./T-1./658.)/R)
    4 CONTINUE
      GOTO (1,2),I
    1 DX2=-K(1)*Y(1)**2
      RETURN
    2 DX2=K(1)*Y(1)**2-K(2)*Y(2)
      RETURN
*     SIMUSOLV
      END
      SUBROUTINE CONTROLEVALUATION(UU,U,CONTROLV,T,N,A,B,N,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      DIMENSION UU(*),U(*),CONTROLV(*),A(*),B(*)
      CALL UNSCALE3(N,UU,U,A,B)
      CONTROLV(1)=U(1)+U(2)*T+U(3)*T**2
      RETURN
      END
      FUNCTION DX1(I,Y,UU,U,A2,B,LB1,UB1)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     NOT INVOKED WITH THIS OPTION
      REAL*8 Y(*),U(*),E(5),K(5),A(5)
      REAL*8 LB1(*),UB1(*),UU(*),A2(*),B(*)
      LOGICAL NOTCALLEDFROMNUMDIFF
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      COMMON/I/NOTCALLEDFROMNUMDIFF
      CALL UNSCALE3(N,UU,U,A2,B)
*     RENFRODAOPEX1
*      R=1.9865
*      A(1)=4000.
*      A(2)=620000.
*     E(1)=5000.
*     E(2)=10000.
*     T=U(1)
*     IF(T.LT.LB1(1))THEN
*     PENALTYTERM=PENALTYTERM+(LB1(1)-T)*PENALTY
*      IF(NOTCALLEDFROMNUMDIFF)T=LB1(1)
*      ENDIF
*      IF(T.GT.UB1(1))THEN
*      PENALTYTERM=PENALTYTERM+(T-UB1(1))*PENALTY
*      IF(NOTCALLEDFROMNUMDIFF)T=UB1(1)
*      ENDIF
*      DO 4 J=1,2
*      K(J)=A(J)*EXP(-E(J)/(R*T))
*    4 CONTINUE
*      GOTO (1,2),I
*    1 DX1=-K(1)*Y(1)**2
*      RETURN
*    2 DX1=K(1)*Y(1)**2-K(2)*Y(2)
*      RETURN
*    3 DX1=K(2)*Y(2)
*      RETURN
*     RAYEX336
      RETURN
      END
      SUBROUTINE OVERALLMASSBALANCE(Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 Y(*)
*     RENFRODAOPEX1 INSIDE-OUT
*      Y(3)=1.-Y(1)-Y(2)
*      IF(Y(3).LT.0.0)Y(3)=0.0
*     SIMUSOLV
      Y(3)=0.75-Y(1)-Y(2)
      IF(Y(3).LT.0.0)Y(3)=0.0
      RETURN
      END
```

OPTIMINP.DAT

```
8
3 0 2 1 0 0
'DCCVPHJRENFRODAOPEX1.DAT'
0
1
0 0 0
0. -100. -100.
500. 100. 100.
1
360. 10. 5.
'A0' 'A1' 'A2'
0.00001 0.1 1. 1.
0.01 0. 1. 0 0
0
298.
```

```
398.
1. 0.
'A' 'B'
```

OPTIMINP.DAT

```
6
3 0 2 1 0 0
'DCCVPBFGSRENFRODAOPEX1.DAT'
0
1
0 0 0
0. -100. -100.
500. 100. 100.
1
360. 10. 5.
'A0' 'A1' 'A2'
0.001 0.5 1. 0.1
100 0.00001 0.001
0.01 0. 1. 0 0
0
298.
398.
1. 0.
'A' 'B'
```

Problem 10

Same as problem 9, with the exception that the batch time as also an optimizer variable.

OPTIMINP.DAT

```
8
4 0 2 1 0 0
'DCCVPHJTFOPTIONRENFRODAOPEX1.DAT'
0
1
0 0 0 1
0. -100. -100. 0.
500. 100. 100. 10.
1
360. 10. 5. 1.
'A0' 'A1' 'A2' 'TF'
0.001 0.1 1. 1.
```

```
0.01 0. 1. 1 0
0
298.
398.
1. 0.
'A' 'B'
```

OPTIMINP.DAT

```
6
4 0 2 1 0 0
'DCCVPBFGSTFOPTIONRENFRODAOPEX1.DAT'
0
1
1 1 1 1
0. -100. -100. 0.
500. 100. 100. 10.
1
310. 40. -20. 2.
'A0' 'A1' 'A2' 'TF'
0.001 0.1 1. 0.1
10000 0.00001 0.001
0.01 0. 1. 1 0
0
298.
398.
1. 0.
'A' 'B'
```

Problem 11

It is a dynamic optimization problem, where the same chemical reaction as referred to in Problem 9 is carried out in a homogeneous tubular reactor. It is found as Problem 4.1 in W. H. Ray, Advanced Process Control

$$\max_{T(t)} \left\{ I = \int_0^{t_f} c_2(1,t)dt \right\}$$

$$\frac{\partial c_1(z,t)}{\partial t} = -\frac{1}{\theta}\frac{\partial c_1(z,t)}{\partial z} - k_1(T)c_1^2(z,t) \quad 0 \le z \le 1 \quad t > 0$$

$$\frac{\partial c_2(z,t)}{\partial t} = -\frac{1}{\theta}\frac{\partial c_2(z,t)}{\partial z} + k_1(T)c_1^2(z,t) - k_2(T)c_2(z,t) \quad 0 \le z \le 1 \quad t > 0$$

$$c_1(z,t) = 1 \quad c_2(z,t) = 0 \; for \; z = 0, \; for \; all \; t > 0$$

$$c_1(z,t) = 1 \quad c_2(z,t) = 0 \; for \; t = 0, \; for \; all \; z \in [0,1]$$

OCFEINP.DAT

```
* ray: advanced process control problem 4.1 of distributed systems
3 1 7 5 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
* THIS IS A COMMENT
OCFEPDEGRG2RAY41.DAT
0.4 0.25 0.05 0.05
7 0 0 0 0 0 2 2 10 7 0 1
4 4 11 15 15 1 1
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
0 0 0 0 0 0 0 13 13 3
3 6 6 6 6 6 6 6 6 6
6 9
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
'C1' 'C2' 'TEM' 'K1' 'K2'
'T' 'Z'
0 1 0 1
0. 2. 0. 1. 0. 0. 0. 0.
0.1 0.2 0.4 0.7 1.1 1.6
0.1 0.2 0.4 0.7
REAL*8 A2(2),E(2),R
TETA=1.
R=1.9865
A2(1)=4000.
E(1)=5000.
A2(2)=620000.
```

```
E(2)=10000.
K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
DC1(T,Z)/DT+DC1(T,Z)/DZ/TETA+K1(T)*C1(T,Z)**2
DC2(T,Z)/DT+DC2(T,Z)/DZ/TETA-K1(T)*C1(T,Z)**2+K2(T)*C2(T,Z)
C1(T,Z)-1 AT T=0 AT Z=0
C2(T,Z) AT T=0 AT Z=0
C1(T,Z)-1 AT T=1 AT Z=0
C2(T,Z) AT T=1 AT Z=0
C1(T,Z)-1 AT T=0 AT Z=1
C2(T,Z) AT T=0 AT Z=1
C1(T,Z)-1 AT Z=0
C2(T,Z) AT Z=0
C1(T,Z)-1 AT T=0
C2(T,Z) AT T=0
INTEGRAL C2(T,1)DT-1.25*ERROR(X)
```

OPTIMINP.DAT

```
* ray: advanced process control problem 4.1 of distributed systems
16
-1. -1. -1. -1. -1. -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 2 0
'PDEGRG2RAY41.DAT'
1 1 0. 100. 100. 0
4 0
1. 0. 350. 0.
0. 0. 298. 0.
1. 1. 398. 2.
0 0 0 0
```

Problem 12
Same as Problem 11, only solved by VF13AD instead of GRG2.

OCFEINP.DAT

```
* ray: advanced process control problem 4.1 of distributed systems
3 2 7 5 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
* THIS IS A COMMENT
OCFEPDESQPRAY41.DAT
0.4 0.25 0.05 0.05
7 0 0 0 0 2 2 2 10 7 0 1
4 4 11 15 15 1 1
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
0 0 0 0 0 0 0 21 21 13
13 3 3 6 6 6 6 6 6 6
```

```
6 6 6 9
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
1 0 0 0 1 1 1 0 1 0 1 0
'C1' 'C2' 'TEM' 'K1' 'K2'
'T' 'Z'
0 1 0 1
0. 2. 0. 1. 0. 0. 0. 0.
0.1 0.2 0.4 0.7 1.1 1.6
0.1 0.2 0.4 0.7
REAL*8 A2(2),E(2),R
TETA=1.
R=1.9865
A2(1)=4000.
E(1)=5000.
A2(2)=620000.
E(2)=10000.
398.-TEM(T)
TEM(T)-298.
K1(T)=A2(1)*EXP(-E(1)/R/TEM(T))
K2(T)=A2(2)*EXP(-E(2)/R/TEM(T))
DC1(T,Z)/DT+DC1(T,Z)/DZ/TETA+K1(T)*C1(T,Z)**2
DC2(T,Z)/DT+DC2(T,Z)/DZ/TETA-K1(T)*C1(T,Z)**2+K2(T)*C2(T,Z)
C1(T,Z)-1 AT T=0 AT Z=0
C2(T,Z) AT T=0 AT Z=0
C1(T,Z)-1 AT T=1 AT Z=0
C2(T,Z) AT T=1 AT Z=0
C1(T,Z)-1 AT T=0 AT Z=1
C2(T,Z) AT T=0 AT Z=1
C1(T,Z)-1 AT Z=0
C2(T,Z) AT Z=0
C1(T,Z)-1 AT T=0
C2(T,Z) AT T=0
- INTEGRAL C2(1,T)DT+1.25*ERROR(X)
```

OPTIMINP.DAT

```
* ray: advanced process control problem 4.1 of distributed systems
18
100000 0.0001 1 1 0
1 1 0. 1. 100. 0
0.0001 1.0D-10 100000 1
4
1. 0. 350. 0.
0. 0. 298. 0.
1. 1. 398. 2.
0 0 0 0
```

Problem 13

It is a parameter estimation problem, found in Simusolv User Guide as an example problem.

The data

Time	Concentration		
	A	B	C
0	0.75	0	0
2	0.266	0.412	0.0479
4	0.105	0.520	0.148
6		0.470	0.216
8	0.0095	0.420	0.329
10	0.00525	0.377	0.357

The mathematical model

$$\frac{dA}{dt} = -k_1 A$$

$$\frac{dB}{dt} = k_1 A + k_3 C - k_2 B$$

$$C = A_0 + B_0 + C_0 - A - B$$

The objective is to find the best values for the rate constants k_1, k_2 and k_3, given the experimental time-concentration data for A, B and C.

usersubs.for

```
REAL*8 FUNCTION
F(UU,X,U,A,B,MISSINGDATA,EXPERDATA,COMPUTEDDATA)
IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
INTEGER*4 SELECTION
INTEGER*4 DATAPOINTER
```

```
      INTEGER*4 MISSINGDATA(NN,M)
      REAL*8 EXPERDATA(NN,M),COMPUTEDDATA(NN,M)
      REAL*8 X(*),U(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/H/T,H
      COMMON/L/DATAPOINTER
      CALL UNSCALE3(N,UU,U,A,B)
*     UTEST1
*      SIMUSOLV
      SUM=0.
      DO 1 I=1,NN
      DO 2 J=2,M
      IF(MISSINGDATA(I,J).EQ.0)THEN
      SUM=SUM+(EXPERDATA(I,J)-COMPUTEDDATA(I,J))**2
      ENDIF
    2 CONTINUE
    1 CONTINUE
      F=SQRT(SUM/NN)
*     RAYEX336
*      F=X(3)
      RETURN
      END
      SUBROUTINE EC(UU,U,A,B,X,F,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR NLPS
*     SELECTION 3,4
*     NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 X(*),F(*),U(*),Y(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      CALL UNSCALE3(N,UU,U,A,B)
*     RENFRONLPEX2
      F(1)=X(1)**2+X(2)**2+X(3)**2+U(1)**2+U(2)**2-10.
      F(2)=X(2)*X(3)-5.*U(1)*U(2)
      F(3)=X(1)*X(1)*X(1)+X(2)*X(2)*X(2)+1.
*     MICROFICHE
      RETURN
      END
      SUBROUTINE EC2(UU,U,A,B,X,F,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*     THIS IS FOR DAOPS
*     SELECTION 9,10,13,14
*     NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 X(*),F(*),U(*),UU(*),A(*),B(*)
      REAL*8 Y(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      CALL UNSCALE3(N,UU,U,A,B)
*      DAPCVPTEST1
*       F(1)=Y(1)+Y(2)+X(1)-1.
*      SIMUSOLV
      F(1)=0.75-Y(1)-Y(2)-X(1)
      RETURN
      END
      REAL*8 FUNCTION IC(UU,U,A,B,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*      THIS IS FOR NLPS
*      SELECTION 3,4
*      NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 U(*),X(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      CALL UNSCALE3(N,UU,U,A,B)
      P=0.
*       MICROFICHE
      IC=P
      RETURN
      END
      REAL*8 FUNCTION IC2(UU,U,A,B,Y,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*      THIS IS FOR DAOPS
*      SELECTION 9,10,13,14
*      NOT INVOKED WITH THIS OPTION
      INTEGER*4 SELECTION
      REAL*8 U(*),X(*),Y(*),UU(*),A(*),B(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      CALL UNSCALE3(N,UU,U,A,B)
      P=0.
      IC2=P
      RETURN
      END
      REAL*8 FUNCTION DX2(I,Y,TIME,UU,U,A2,B,LB1,UB1,X)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 Y(*),U(*),C(5),E(5),K(5),A(2)
      REAL*8 LB1(*),UB1(*),UU(*),A2(*),B(*)
      INTEGER*4 SELECTION,TFOPTION
      LOGICAL NOTCALLEDFROMNUMDIFF
      REAL*8 X(*)
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      COMMON/D/TFOPTION,TF
      COMMON/I/NOTCALLEDFROMNUMDIFF
      CALL UNSCALE3(N,UU,U,A2,B)
*       SIMUSOLV
       IF(U(1).LT.0.0)PENALTYTERM=PENALTYTERM-U(1)*PENALTY
       IF(U(2).LT.0.0)PENALTYTERM=PENALTYTERM-U(2)*PENALTY
       IF(U(3).LT.0.0)PENALTYTERM=PENALTYTERM-U(3)*PENALTY
      IF(U(1).LT.0.0)U(1)=0.0
      IF(U(2).LT.0.0)U(2)=0.0
      IF(U(3).LT.0.0)U(3)=0.0
      GOTO (1,2),I
    1 DX2=-U(1)*Y(1)
      RETURN
*     SIMUSOLV MASSBALANCE
    2 DX2=U(1)*Y(1)+U(3)*Y(3)-U(2)*Y(2)
*     SIMUSOLV CONSTRAINT
*     2 DX2 = U(1)*Y(1)+U(3)*X(1)-U(2)*Y(2)
      RETURN
*     RAYEX336
      END
      SUBROUTINE CONTROLEVALUATION(UU,U,CONTROLV,T,N,A,B,N,Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      DIMENSION UU(*),U(*),CONTROLV(*),A(*),B(*)
      CALL UNSCALE3(N,UU,U,A,B)
      CONTROLV(1)=U(1)+U(2)*T+U(3)*T**2
      RETURN
      END
      REAL*8 FUNCTION DX1(I,Y,UU,U,A2,B,LB1,UB1)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 Y(*),U(*),E(5),K(5),A(5)
      REAL*8 LB1(*),UB1(*),UU(*),A2(*),B(*)
      LOGICAL NOTCALLEDFROMNUMDIFF
*     NOT INVOKED WITH THIS OPTION
      COMMON/A/SELECTION,N,NALGSTATE,NALGSTATE2,NTIMEDEPSTATE,
     1NCONTROLVAR,NN,M
      COMMON/B/PENALTY,PENALTYTERM,FEASIBILITYTOL
      COMMON/I/NOTCALLEDFROMNUMDIFF
      CALL UNSCALE3(N,UU,U,A2,B)
      RETURN
      END
      SUBROUTINE OVERALLMASSBALANCE(Y)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      REAL*8 Y(*)
*     RENFRODAOPEX1 INSIDE-OUT
*      Y(3)=1.-Y(1)-Y(2)
*      IF(Y(3).LT.0.0)Y(3)=0.0
*     SIMUSOLV
      Y(3)=0.75-Y(1)-Y(2)
      IF(Y(3).LT.0.0)Y(3)=0.0
      RETURN
      END
```

OPTIMINP.DAT

```
* simusolv user guide parameter estimation example
11
3 0 3 0 6 4
'DCCVPHJPEMBSIMUSOLVKINEX.DAT'
0
1
1 1 1
0. 0. 0.
1. 1. 1.
1
0.3 0.3 0.02
'K1' 'K2' 'K3'
0.001 0.5 1. 0.1
0.1 0. 12. 0 0
1
0.75 0. 0.
'A' 'B' 'C'
0 0 0 0
0 0 0 0
0 0 0 0
0 1 0 0
0 0 0 0
0 0 0 0
0. 0.75 0. 0.
2. 0.266 0.412 0.0479
4. 0.105 0.520 0.148
6. 0. 0.470 0.216
8. 0.0095 0.420 0.329
10. 0.00525 0.377 0.357
```

OPTIMINP.DAT

```
* simusolv user guide parameter estimation example
12
3 0 3 0 6 4
'DCCVPBFGSPEMBSIMUSOLVKINEX.DAT'
0
1
1 1 1
0. 0. 0.
```

```
1. 1. 1.
1
0.3 0.3 0.02
'K1' 'K2' 'K3'
0.001 0.5 1. 0.1
100 0.00001 0.001
0.1 0. 12. 0 0
1
0.75 0. 0.
'A' 'B' 'C'
0 0 0 0
0 0 0 0
0 0 0 0
0 1 0 0
0 0 0 0
0 0 0 0
0. 0.75 0. 0.
2. 0.266 0.412 0.0479
4. 0.105 0.520 0.148
6. 0. 0.470 0.216
8. 0.0095 0.420 0.329
10. 0.00525 0.377 0.357
```

Problem 14

Same as Problem 13, with the exception that it is solved by GRG2 instead of using control variable parameterization.

OCFEINP.DAT

```
* simusolv user guide parameter estimation example
1 1 12 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1
OCFE1GRG2SIMUSOLVEX.DAT
0.5 0.5 0.5 0.5
3 0 0 3 1 0 0 2 3 4 0 1
2 2 2 1
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 10 1 1 4 4 4 7
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
'A' 'B' 'C' 'T'
1 1 1 1
```

```
0. 12. 0. 0. 0. 0. 0. 0.
1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
11.
A0=0.75
B0=0.
C0=0.
C(T)-A0-B0-C0+A(T)+B(T)
DA(T)/DT+X(1)*A(T)
DB(T)/DT-X(1)*A(T)-X(3)*C(T)+X(2)*B(T)
A(T)-0.75 AT T=0
B(T) AT T=0
C(T) AT T=0
PARAMETER ESTIMATION
6
(A(T)-0.75)**2+(B(T)-0.0)**2+(C(T)-0.0)**2 AT T=0
(A(T)-0.266)**2+(B(T)-0.412)**2+(C(T)-0.0479)**2 AT T=2
(A(T)-0.105)**2+(B(T)-0.52)**2+(C(T)-0.148)**2 AT T=4
(B(T)-0.47)**2+(C(T)-0.216)**2 AT T=6
(A(T)-0.0095)**2+(B(T)-0.42)**2+(C(T)-0.329)**2 AT T=8
(A(T)-0.00525)**2+(B(T)-0.377)**2+(C(T)-0.357)**2 AT T=10
```

OPTIMINP.DAT

```
* simusolv user guide parameter estimation example
16
-1. -1. -1. -1. -1. -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 1 0
'GRG2SIMUSOLVEX.DAT'
1 1 0. 100. 100. 1
1
1 1 1
0. 0. 0.
10. 10. 10.
1
0.3 0.3 0.02
'K1' 'K2' 'K3'
3 1
0.75 0. 0.
0. 0. 0.
1. 1. 1.
0.
0.
0 0 0
```

Problem 15

Same as Problem 13, with the exception that it is solved by VF13AD instead of using control variable parameterization.

OCFEINP.DAT

```
* simusolv user guide parameter estimation example
1 2 12 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1
OCFE1SQPSIMUSOLVEX.DAT
0.5 0.5 0.5 0.5
3 0 3 3 1 0 0 2 3 4 0 1
2 2 2 1
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 10 1 1 4
4 4 7
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
'A' 'B' 'C' 'T'
1 1 1 1
0. 12. 0. 0. 0. 0. 0. 0.
1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
11.
A0=0.75
B0=0.
C0=0.
X(1)
X(2)
X(3)
C(T)-A0-B0-C0+A(T)+B(T)
DA(T)/DT+X(1)*A(T)
DB(T)/DT-X(1)*A(T)-X(3)*C(T)+X(2)*B(T)
A(T)-0.75 AT T=0
B(T) AT T=0
C(T) AT T=0
PARAMETER ESTIMATION
6
(A(T)-0.75)**2+(B(T)-0.0)**2+(C(T)-0.0)**2 AT T=0
(A(T)-0.266)**2+(B(T)-0.412)**2+(C(T)-0.0479)**2 AT T=2
(A(T)-0.105)**2+(B(T)-0.52)**2+(C(T)-0.148)**2 AT T=4
(B(T)-0.47)**2+(C(T)-0.216)**2 AT T=6
(A(T)-0.0095)**2+(B(T)-0.42)**2+(C(T)-0.329)**2 AT T=8
(A(T)-0.00525)**2+(B(T)-0.377)**2+(C(T)-0.357)**2 AT T=10
```

OPTIMINP.DAT

```
* simusolv user guide parameter estimation example
18
100000 0.000001 1 1 0
```

```
1 1 0. 1. 100. 1
1
1 1 1
0. 0. 0.
1. 1. 1.
1
0.3 0.3 0.02
'K1' 'K2' 'K3'
0.0001 1.0D-10 100000 1
3
0.75 0. 0.
0. 0. 0.
1. 1. 1.
0 0 0
```

example how to set up ocfeint.for

ocfeint.for

```
      REAL*8 FUNCTION VARTIMEDELAY(T,I,TDIF)
* HANDLES CONSTANT AND VARIABLE TIME DELAYS
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      GOTO(1,2,3,4)I
    1 VARTIMEDELAY2=0.02
*   1 VARTIMEDELAY2=0.0
      IF(T.LT.VARTIMEDELAY2)TDIF=1.
      VARTIMEDELAY=VARTIMEDELAY2
      RETURN
    2 VARTIMEDELAY2=0.015
*   2 VARTIMEDELAY2=0.0
      IF(T.LT.VARTIMEDELAY2)TDIF=-0.02
      VARTIMEDELAY=VARTIMEDELAY2
      RETURN
    3 VARTIMEDELAY2=0.5
      IF(T.LT.VARTIMEDELAY2)TDIF=T-0.5
      VARTIMEDELAY=VARTIMEDELAY2
    4 VARTIMEDELAY2=1
      IF(T.LT.VARTIMEDELAY2)TDIF=1
      VARTIMEDELAY=VARTIMEDELAY2
      RETURN
      END
      LOGICAL FUNCTION WITHIN(X,Y)
* CHECKS WHETHER (X,Y) IS WITHIN AN IRREGULAR DOMAIN
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
*      WRITE(6,*)X,Y,(X-0.5)**2+(Y-0.5)**2
      IF((X-0.5)**2+(Y-0.5)**2.LT.0.25)THEN
*      IF(X.GT.0.001.AND.X.LT.0.999.AND.Y.GT.0.001.AND.Y.LT.0.999)THEN
*      IF(X.GE.0..AND.X.LT.0.499.AND.Y.GE.0..AND.Y.LE.1.)THEN
```

```
      WITHIN=.TRUE.
      ELSE
      WITHIN=.FALSE.
      ENDIF
      RETURN
      END
```

example how to set up userint.for

userint.for

```
*     PROGRAM TESTCZEQO
*     X=CZEQ0(0.0D+00)
*     WRITE(6,*)X
*     X=CZEQ0(0.3D+00)
*     WRITE(6,*)X
*     X=CZEQ0(0.11D-01)
*     WRITE(6,*)X
*     X=CZEQ0(0.1D+00)
*     WRITE(6,*)X
*     X=CZEQ0(0.12D+00)
*     WRITE(6,*)X
*     X=CZEQ0(0.27D+00)
*     WRITE(6,*)X
*     STOP
*     END
      REAL*8 FUNCTION F1(T,X)
* HANDLES VARIABLE BOUNDARY CONDITIONS (SARMIDI)
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      TETAF=0.5
      IF(T.LE.TETAF)THEN
      F1=0.5
      ELSE
      F1=0
      ENDIF
      RETURN
      END
      REAL*8 FUNCTION FLOWRATE(T)
* HANDLES DISTURBANCES AND OTHER VARIABLES THAT ARE NEITHER
STATE
* NOR CONTROL VARIABLES, E.G. VARIABLES THAT COME FROM
ANOTHER PROCESS
* AND ARE NOT AVAILABLE FOR MANIPULATION
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      I=999
      IF(T.LE.0.5)THEN
      FLOWRATE=50+RAN(I)
      ELSE
      FLOWRATE=60+RAN(I)
      ENDIF
      RETURN
      END
      REAL*8 FUNCTION CZEQ0(T)
* HANDLES DISTURBANCES AND OTHER VARIABLES THAT ARE NEITHER
STATE
* NOR CONTROL VARIABLES, E.G. VARIABLES THAT COME FROM
ANOTHER PROCESS
* AND ARE NOT AVAILABLE FOR MANIPULATION
* CHECKING EFFECT OF STEP CHANGE IN INPUT ON THE SOLUTION
* IT CAUSES DISCONTINUITY FOR FIRST-ORDER HYPERBOLIC EQUATIONS
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      IF(T.LE.0.1)THEN
      CZEQ0=1
      ELSE
*       CZEQ0=0.75
*       CZEQ0=0.95
      CZEQ0=0.9
*       CZEQ0=0.99
*       CZEQ0=1
      ENDIF
      RETURN
      END
      REAL*8 FUNCTION U(T)
* HANDLES DISTURBANCES AND OTHER VARIABLES THAT ARE NEITHER
STATE
* NOR CONTROL VARIABLES, E.G. VARIABLES THAT COME FROM
ANOTHER PROCESS
* AND ARE NOT AVAILABLE FOR MANIPULATION
* CHECKING EFFECT OF STEP CHANGE IN INPUT ON THE SOLUTION
* IT CAUSES DISCONTINUITY FOR FIRST-ORDER HYPERBOLIC EQUATIONS
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER*4 (I-N)
      OMEGA=3.
      U=DSIN(OMEGA*T)
      RETURN
      END
```

Things to try if the package is not working and useful hints

1. It is recommended to try out first a run on a small rectangle, say, [0.25]X[0.25]

2. It is recommended that constant control over a finite element be used to reduce the residual error.

Appendix B - Program Documentation

OCFE

This program discretizes equations depending on independent variables and transfers the model into interface routines for the package *GRG2* or *VF13AD*.

Description of Variables

Input variables are described in the User Guide.

OUTPUTSTRING	contains the transformed model equation	
MININD	points to the beginning of the variable name in the model equation	
WORKSTRING	contains the model equation	
VARIND1	= 1	first independent variable has been found between parentheses
	= 2	second independent variable has been found between parentheses

The first independent variable precedes the second independent variable in the array VARNAME.

MINJ	subscript of the next variable to be processed in the model equation	
INTEGRALTERM	= 1	if a differential equation derived from an integral term of an objective functional is processed
NONUNIQUEVAR	true if the name of the variable to be processed is followed by an alphanumeric character	
NONUNIQUEVAR2	true if the name of the variable to be processed is preceded by an alphanumeric character excluding 'D' and '2'	
WHICHEQ	= 1	differential equation or any other equation containing a variable dependent on an independent variable
	=2	initial/boundary condition
	=3	objective functional
INPUTTEXT	subscript of the model equation currently processed	

INTERFACE

This subroutine scans the equations read in from the problem definition file and calls subprograms to discretize equations containing continuous variables.

ADDSUBSCRIPTSTOPARAMETER

This subprogram replaces parameters like k (rate constant) by $k_1,\ldots,k_i,\ldots,k_n$, *i* is incremented for each collocation point. If it processes expressions like k = EXP(E/R/T), then it also declares k_i as a REAL*8 variable.

ADDSUBSCRIPTSTOPARAMETERB

This subprogram replaces parameters like k (rate constant) by $k_1,\ldots,k_i,\ldots,k_n$, *i* is incremented for each residual error evaluation point in the integration domain. If it processes expressions like k = EXP(E/R/T), then it also declares k_i as a REAL*8 variable.

ALLOCATEMEMORY

OCFE has been written to provide the appearance of dynamic memory allocation with *REAL*8* arrays set up as portions of *DPCORE* so that *OCFE* does not waste memory space on individual arrays and the memory is utilized most economically. This subroutine assigns pointers to *DPCORE* for each *REAL*8* array.

BREAKUP

This subprogram is called from *PROCESSEQUATION1,2,3,1B,2B,3B*. It searches for a string enclosed in parentheses and if the string is longer than 10 characters, generates x_i = string and replaces the string by x_i in the output string (the transformed model equation). It calls *BREAKUPANDWRITE* and *OPENCLOSEPAR*.

Description of variables

I3	points to the first character to be copied from *OUTPUTSTRING* to *STRING*.
J2	points to *))*
J1	points to *(*.
STRING	transformed *OUTPUTSTRING*, where each string longer than 51 is replaced by x_i.

When searching for *(*, it ignores *(* preceded by *X*.

OPENCLOSEPAR

This subprogram counts open and close parentheses.

BREAKUPANDWRITE

This subprogram breaks up a transformed model equation into a *FORTRAN* statement having continuation lines if necessary. It handles 3 types of equations:
TYPE = 1 e.g. R = 1.9895
TYPE = 2 e.g. DX1/DT + K1 * X1 = 0
TYPE = 3 least square objective function
It writes the generated *FORTRAN* statement in a temporary file. The statements will be eventually put into the files *GRG2INT.FOR* and *SQPINT.FOR*, to be part of the subroutines *GCOMP* or *SQPCONSTRAINTS*.

REPLACEPM

It replaces +- by - in the transformed model equation.

DISTURBANCE

It replaces variables coming from other process units. They are known functions of the independent variables. Called from *PROCESSEQUATION1*, *PROCESSEQUATION2* and *PROCESSEQUATION3*, if an open square bracket is encountered in a model equation. The open square bracket is considered a variable having *VARCLASS* = 19. For example, *FLOW[T]* in a boundary condition at $z = 0$ and $t = 0$ will be replaced by *FLOW(0.0)* and will be called from *GCOMP* or from *SQPCONSTRAINTS*. The *REAL*8 FUNCTION FLOW(T)* has to be put in *USERINT.FOR* by the user. It calls *GETTFROMLZ* to obtain the value of *T* from the finite element and from the collocation point.

EVALFUNC

It generates a program *EVALFUNC*. *EVALFUNC* will write tabulated values of the state and control variables at user-defined equidistant points *t*, *t + deltat*,... and *x*, *x + delta x*,.... *EVALFUNC* uses the optimal *NLP* variables to interpolate the continuous state and control variables. If, for example, the state variable *c* is dependent on 2 independent variables *t*,*z* and *t* falls in the second finite element along the *t-axis* and *z* falls in the first finite element along the *z-axis* and the number of internal collocation points is 3 in both finite elements, then

$$c^{21} = \sum_{i=1}^{5}\sum_{j=1}^{5} l_i(u) l_j(v) c_{ij}^{21} \tag{1}$$

where c_{ij}^{21} is an *NLP* variable corresponding to the *i-th* collocation point in the second finite element along the *t-axis* and to the *j-th* collocation point in the first finite element along the *z-axis* and

$$u = \frac{(t-t_1)}{(t_2-t_1)} \quad and \quad v = \frac{(z-z_0)}{(z_1-z_0)} \tag{2}$$

where $t_2 - t_1$ is the length of the second finite element along the *t-axis*, t_1 is the first knot, t_2 is the second knot along the *t-axis*, and z_1-z_0 is the length of the first finite element along the *z-axis* and z_0 is the start of the integration along the *z-axis* and z_1 is the first knot along the *z-axis*. If w is a control variable dependent on t, then if t falls in the second finite element along the *t-axis* and the number of internal collocation points is 3 in the finite element, then

$$w^2 = \sum_{i=2}^{4} l_i(u) w_i^2 \tag{3}$$

where w_i^2 is an *NLP* variable corresponding to the *i-th* collocation point in the second finite element along the *t-axis*. The summation for control variables excludes the end points of finite elements. If, for instance, the integration is to be performed from $t=0$ to $t=1$ and from $z=0$ to $z=1$ and $\Delta t = 0.1$ and $\Delta z = 0.1$, then *EVALFUNC* would write values for c and w in *FUNCOUT.DAT* as follows assuming that t is the first independent variable:

$$c(0,0),\ c(0,.01),...,c(1,1) \quad and$$

$$w(0),\ w(0.1),...,w(1).$$

EVALERR

This subprogram generates the program *EVALERR* in the file *EVALERR.FOR* to evaluate the differential equations at equidistant points $t, t+\Delta t$ and $x, x+\Delta x$, and the generated program *EVALERR* writes the residual error to *ERROROUT.DAT*. The *FORTRAN* statements to evaluate the residuals are generated by *PROCESSEQUATION1B,PROCESSEQUATION2B* and *PROCESSEQUATION3B* and written to *OCFETEMP4.DAT*. The generated program reads the optimal solutions from the file *OPTSOLUTION.DAT* written by *DISPATCHER*. *EVALERR* uses the optimal *NLP* variables to interpolate the continuous state and control variables. If, e.g., the integration is to be performed from $t=0$ to $t=1$ and from $z=0$ to $z=1$ and $\Delta t=0.1$ and $\Delta z=0.1$, and *SATISFIED(i)=1 i=1,...,4* and t is the first independent variable, then *EVALERR* would evaluate every differential equation in the model at $(t=0, z=0), (t=0, z=0.1),...,(t=1,z=1)$. It also calculates the estimate of the approximation error based on *B. A. Finlayson "Nonlinear Analysis in Chemical Engineering" p. 143* and *p. 305*.

ERROR

It generates the *REAL*8 FUNCTION ERROR* in the file *ERROR.FOR* to evaluate the squared residual error. The *FORTRAN* statements to evaluate the residuals are generated by *PROCESSEQUATION1B*,*PROCESSEQUATION2B* and *PROCESSEQUATION3B* and written to *OCFETEMP4.DAT*. If problems *4,5,6,7,8,9,10,11,12* are solved using options *16* or *18*, *ERROR(X)* has to be added to the objective functional if the problem is defined as minimization and *ERROR(X)* has to be subtracted from the objective functional if the problem is defined as maximization. Then variables cannot be named *x*. *ERROR(X)* may shift the optimum, so it is recommended that with optimization problems the user runs his/her problem with and without *ERROR(X)*. Weights can also be used like *F(T)-1.25*ERROR(X)*. Before linking *OPTIMIZER*, *ERROR.FOR* has to be compiled.

GRG2INTERFACE

It writes variables *NVAR*, *NFUN*, *NOBJ*, *NALGVAR*, *NALGC*, *NTDIQ*, *NSTATE* and arrays *TDVRANGE*, *STATEVARIND* and *TDCRANGE* into *OCFEGRG2IF.DAT*. This file will be read by *DISPATCHER*. *NVAR* is equal to the number of *NLP* variables. The other variables are described in the *USER GUIDE*.

SQPINTERFACE

It writes variables *NVAR*, *NFUN*, *NOBJ*, *MEQ*, *NALGVAR*, *NSTATE* and arrays *TDVRANGE* and *STATEVARIND* into *OCFESQPIF.DAT*. This file will be read by *DISPATCHER*. *NVAR* is equal to the number of *NLP* variables. The other variables are described in the *USER GUIDE*.

INSERT

It reads collocation equations generated by *PROCESSEQUATION1*, *PROCESSEQUATION2* and *PROCESSEQUATION3* from *OCFETEMP.DAT*, and assembles the *GRG2* user interface *GCOM* in the file *GRG2INT.FOR*.

INSERT2

It reads collocation equations generated by *PROCESSEQUATION1*, *PROCESSEQUATION2* and *PROCESSEQUATION3* from *OCFETEMP.DAT*, and assembles the *VF13AD* user interface subprograms *SQPOF* and *SQPCONSTRAINTS* in the file *SQPINT.FOR*.

ISUMNICPA

It sums up collocation points up to subinterval *L* exclusive.

ISUMNICPB

It sums up collocation points over the entire integration domain.

OUTPUTINDVAR

It outputs the roots of *Legendre* polynomials and their mapped values into the integration domain in the file *OCFEOUT.DAT.*

PROCESSDERIVATIVE

It replaces derivatives of the state variables as a linear combination of the derivatives of the *Lagrange* polynomials previously evaluated at collocation points and stored in arrays *A, B, A2* and *B2,* where

A(I,J,K)	first derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the first independent variable
B(I,J,K)	second derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the first independent variable
A2(I,J,K)	first derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the second independent variable
B2(I,J,K)	second derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the second independent variable

In the *kl-th* element the unknown *c* is approximated by

$$c^{kl}(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{M} l_i(u)l_j(v)c_{ij}^{kl} \tag{4}$$

$$u = \frac{(x-x_k)}{(x_{k+1}-x_k)} \qquad v = \frac{(y-y_l)}{(y_{l+1}-y_l)} \tag{5}$$

Here c_{ij}^{kl} is the value of *c* at the collocation point (u_i,v_j) in the *kl-th* element.
$x_{k+1}-x_k = \Delta x_k$ is the size of the *k-th* finite element in the *x*-direction.
$y_{l+1}-y_l = \Delta y_l$ is the size of the *l-th* finite element in the *y*-direction.

$$\frac{\partial c^{kl}(x,y)}{\partial x \partial y} = \frac{1}{\Delta x_k \Delta y_l}\sum_{i=1}^{N+2}\sum_{j=1}^{M+2} A(m,i,k)A2(n,j,l)c_{ij}^{kl} \tag{6}$$

when evaluated at collocation point (u_m,v_n)

$$\frac{\partial c^{kl}(x,y)}{\partial x} = \frac{1}{\Delta x_k} \sum_{i=1}^{N+2} A(m,i,k)c_{in}^{kl} \tag{7}$$

when evaluated at collocation point (u_m, v_n)

$$\frac{\partial c^{kl}(x,y)}{\partial y} = \frac{1}{\Delta y_l} \sum_{j=1}^{M+2} A2(n,j,l)c_{mj}^{kl} \tag{8}$$

when evaluated at collocation point (u_m, v_n)

$$\frac{\partial^2 c^{kl}(x,y)}{\partial x^2} = \frac{1}{\Delta x_k^2} \sum_{i=1}^{N+2} B(m,i,k)c_{in}^{kl} \tag{9}$$

when evaluated at collocation point (u_m, v_n)

$$\frac{\partial^2 c^{kl}(x,y)}{\partial y^2} = \frac{1}{\Delta y_l^2} \sum_{j=1}^{M+2} B2(n,j,l)c_{mj}^{kl} \tag{10}$$

when evaluated at collocation point (u_m, v_n)

$$\frac{dc^{k}(x)}{dx} = \frac{1}{\Delta x_k} \sum_{i=1}^{N+2} A(m,i,k)c_{i}^{k} \tag{11}$$

when evaluated at collocation point (u_m)

$$\frac{dc^{l}(y)}{dy} = \frac{1}{\Delta y_l} \sum_{j=1}^{M+2} A2(n,j,l)c_{j}^{l} \tag{12}$$

when evaluated at collocation point (v_n)

$$\frac{d^2c^k(x)}{dx^2} = \frac{1}{\Delta x_k^2} \sum_{i=1}^{N+2} B(m,i,k)c_i^k \tag{13}$$

when evaluated at collocation point (u_m)

$$\frac{d^2c^l(y)}{\partial y^2} = \frac{1}{\Delta y_l^2} \sum_{j=1}^{M+2} B2(n,j,l)c_j^l \tag{14}$$

when evaluated at collocation point (v_n)

Local variables

I	=	subscript of the *NLP* variable associated with the finite element and the subscript of *c* corresponding to a collocation point
STRING	=	the string replacing the derivative term in the model equation

PROCESSDERIVATIVEB

It is called for error evaluation. It replaces derivatives of the state variables as a linear combination of the *Lagrange* polynomials computed analytically at grid points $(x+i\Delta x, y+j\Delta y)$. Since the grid points where the derivatives have to be evaluated do not normally coincide with collocation points, arrays *A, B, A2, B2* cannot be used, *REAL*8 FUNCTION DERIV1* and *DERIV2* have to be invoked instead.

DERIV1

It computes the first derivative of the *i-th Lagrange* polynomial in finite element *L* and at collocation point *z* analytically.

DERIV2

It computes the second derivative of the *i-th Lagrange* polynomial in finite element *L* and at collocation point *z* analytically.

MIXEDDERIVATIVE

It checks a mixed derivative term in a model equation, whether *VARNAME* contains the 2 independent variables in the derivative term.

PROCESSEQUATION1

It discretizes model equations depending only on the first independent variable, replacing variables and derivatives. It handles differential equations, initial/boundary conditions, objective functionals and any other model equations dependent on the first independent variable. Initial and boundary conditions and objective functionals are discretized only at collocation points according to the scope of the equations. Differential equations are discretized along the integration domain of the first independent variable excluding initial and boundary conditions. The model equation is scanned repeatedly for each collocation point.

Algorithm

```
LOOP :    FOR each finite element selected DO :
          LOOP :    FOR each collocation point selected DO :
                    find the next unprocessed variable dependent on the
                    independent variable in the model equation
                    IF found THEN
                    CASE type of substring to be replaced
                    derivative : process derivative
                    state variable : process state variable
                    control variable : process control variable
                    parameter : process parameter
                    independent variable :    process independent
                                              variable
                    END CASE
                    ELSE
                    write out the transformed equation
                    ENDIF
          END LOOP
END LOOP
```

PROCESSEQUATION1B

It is similar to *PROCESSEQUATION1*, except for the fact that boundary conditions and objective functionals are not processed and looping is performed over equidistant points for the purposes of residual error evaluation.

PROCESSEQUATION2

It is like *PROCESSEQUATION1*, but processing model equations depending only on the second independent variable.

PROCESSEQUATION2B

It is like *PROCESSEQUATION1B*, but processing model equations depending only on the second independent variable.

PROCESSEQUATION3

It is like *PROCESSEQUATION1*, except for the fact that discretization is performed over a rectangular integration domain and it processes model equations which contains at least one variable dependent on both independent variables.

PROCESSEQUATION3B

It is similar to *PROCESSEQUATION3*, except for the fact that boundary conditions and objective functionals are not processed and looping performed over a rectangular domain over equidistant points for the purposes of residual error evaluation.

PROCESSOF

```
CASE type of objective functional
LEAST SQUARE : process least square objective functional
INTEGRAL : LOOP for each integral term
                convert the integral term into a differential equation. Set
                initial/boundary conditions of the differential equation. Set
                attributes of the artificial variable derived from the integral
                term. Set attributes of the differential equation and the initial
                condition.
           END LOOP
END CASE
```

PARES1

It processes a least-square objective functional in case of parameter estimation.

PARES2

It passes the sample time to *PARES3* in case of parameter estimation.

PARES3

It processes one line of the least-square objective functional replacing variables by a linear combination of *Lagrange* interpolation polynomials evaluated at sample time after the sample time has been mapped into the interval *[0,1]*.

GETNUM

It returns in *I1* the start position of and in *I2* the end position of a number string delimited by spaces.

READCOMMENTS

It treats lines in *OCFEINP.DAT* as comment lines if the first character in the line is an asterisk.

REPLACEINDEPENDENTVAR

This subprogram maps a collocation point, i.e. the root of a *Legendre* polynomial in a finite element into the integration domain.

REPLACEINDEPENDENTVARB

This subprogram replaces the name of an independent variable by its value at the point where the residual is to be evaluated.

REPLACESCVAR

This subprogram replaces a state or control variable with its *NLP* equivalent taking account of a possible time delay. If the collocation point - time delay < the start of the integration then the variable takes on its initial value. If a variable is fixed at knot or boundary then *REPLACESCVAR* adjusts the finite element and the collocation point where the variable is to be evaluated. If a control variable is to be evaluated at the beginning or at the end of a finite element then the control variable is approximated as a linear combination of the *NLP* equivalents of the control variable evaluated at internal collocation points.

REPLACECONTROLVAR

REPLACECONTROLVAR is called from *REPLACESCVAR* when the control variable is of *VARCLASS 5,6,7* and it has to be replaced in the model equation at the end-point or edge of a finite element. If w is a control variable of *VARCLASS=5* in finite element k, then if the number of internal collocation points in the *k-th* finite element along the axis of the first independent variable is N, then since the collocation point is at the end-point of a finite element, z equals either 0 or 1 and

$$w^{k}(x) = \sum_{i=2}^{N+1} l_i(z) w_{NLP(k,i)} \tag{15}$$

If w is a control variable of *VARCLASS=6* in finite element l, then if the number of internal collocation points in the *l-th* finite element along the axis of the second independent variable is M, then since the collocation point is at the end-point of a finite element, $z2$ equals either 0 or 1 and

$$w^{l}(y) = \sum_{j=2}^{M+1} l_j(z2) w_{NLP(l,j)} \tag{16}$$

Similarly, if w is a control variable of *VARCLASS=7*, then

$$w^{kl}(x,y) = \sum_{i=2}^{N+1} \sum_{j=2}^{M+1} l_i(z) l_j(z2) w_{NLP(k,i,l,j)} \tag{17}$$

REPLACESCVARB

REPLACESCVARB is called for error evaluation. It approximates state and control variables by a linear combination of *Lagrange* interpolation polynomials, taking account of a possible time delay. If the collocation point - time delay < the start of the integration, then its initial value is assigned to the variable. The linear combination coefficients, i.e. the state or control variables at the collocation points are replaced by their *NLP* equivalents. If the variable is fixed at knot or at boundary then *REPLACESCVARB* sets the finite element and the collocation point in accordance with the fixed value of the independent variable.

A state variable of *VARCLASS=2* is approximated by

$$c^k(x) = \sum_{i=1}^{N+2} l_i(u) c_i^k \quad where\ u = \frac{(x - x_k)}{(x_{k+1} - x_k)} \tag{18}$$

$l_i(u)$ is the *i-th Lagrange* polynomial evaluated at u. The number of internal collocation points in the *k-th* element is N. c_i^k is the value of c at collocation point u_i in the *k-th* element. x_k is the left end-point of the *k-th* finite element.

A state variable of *VARCLASS=3* is approximated by

$$c^l(y) = \sum_{j=1}^{M+2} l_j(v) c_j^l \quad where\ v = \frac{(y - y_l)}{(y_{l+1} - y_l)} \tag{19}$$

$l_j(v)$ is the *j-th Lagrange* polynomial evaluated at v. The number of internal collocation points in the *l-th* element is M. c_j^l is the value of c at collocation point v_j in the *l-th* element. y_l is the left end-point of the *l-th* finite element.

A state variable of *VARCLASS=4* is approximated by

$$c^{kl}(x,y) = \sum_{i=1}^{N+2} \sum_{j=1}^{M+2} l_i(u) l_j(v) c_{ij}^{kl} \tag{20}$$

c_{ij}^{kl} is the value of c at collocation point (u_i, v_j) in the *kl-th* element.

A control variable w of *VARCLASS=5* is approximated by

$$w^k(x) = \sum_{i=2}^{N+1} l_i(u) w_i^k \tag{21}$$

A control variable of *VARCLASS=6* is approximated by

$$w^l(y) = \sum_{j=2}^{M+1} l_j(v) w_j^l \tag{22}$$

A control variable of *VARCLASS=7* is approximated by

$$w^{kl}(x,y) = \sum_{i=2}^{N+1}\sum_{j=2}^{M+1} l_i(u)l_j(v)w_{ij}^{kl} \tag{23}$$

Control variables of *VARCLASS=11,12,21* are constant over a finite element, they are not approximated by a linear combination of *Lagrange* interpolation polynomials, they have an *NLP* equivalent.

Control variable of *VARCLASS=13* is approximated by

$$w^{kl}(x,y) = \sum_{j=2}^{M+1} l_j(v)w_j^{kl} \tag{24}$$

Control variable of *VARCLASS=14* is approximated by

$$w^{kl}(x,y) = \sum_{i=2}^{N+1} l_i(u)w_i^{kl} \tag{25}$$

Local variables

T,T2	value of the first independent variable
x,x2	value of the second independent variable
zz	the value of the first independent variable mapped into the corresponding finite element
zz2	the value of the second independent variable mapped into the corresponding finite element
L,L2,L3	finite elements along the first independent variable
LL,LL2,LL3	finite elements along the second independent variable

INDEX2

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=4* if *NLPVARSUBSCRIPTS* is added to it.

INDEX3

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=7* if *NLPVARSUBSCRIPTS* is added to it.

INDEX6

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=13* if *NLPVARSUBSCRIPTS* is added to it.

INDEX7

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=14* if *NLPVARSUBSCRIPTS* is added to it.

INDEX4

This subprogram returns the subscript added to a parameter of *VARCLASS=10.*

INDEX8

This subprogram returns the subscript added to a parameter of *VARCLASS=17.*

INDEX9

This subprogram returns the subscript added to a parameter of *VARCLASS=18.*

INDEX10

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=21* if *NLPVARSUBSCRIPTS* is added to it. It also returns the subscript added to a parameter of *VARCLASS=22.*

INDEX11

This subprogram returns the subscript added to a parameter of *VARCLASS=22.*

INDEX5

This subprogram returns the *NLP* subscript of a variable of *VARCLASS=2,3,4,5,6,7,11,12,13,14* or *21.*

RLAGRANGE

This subprogram evaluates the *i-th Lagrange* polynomial at *z* in the finite element *L* with *N* internal collocation points. If a state variable is approximated then the summation runs from *1* to *N+2.* If a control variable is approximated, the summation runs from *2* to *N+1.*

GETLZFROMT

Given a value of an independent variable, this subprogram returns the finite element and variable value mapped into the interval *[0,1].*

GETTFROMLZ

Given a variable mapped into the interval *[0,1]* in a finite element, this subprogram returns the original value of an independent variable.

TIMEDELAY

This subprogram returns the value of a time delay, i.e. decodes a number string into a value.

WHICHINDVAR

Given an independent variable name, it returns in *IND1* the subscript of the first independent variable in *VARCLASS* and sets *VARIND1* to *1* if the given string is the first independent variable. Given an independent variable name, it returns in *IND2* the subscript of the second independent variable in *VARCLASS* and sets *VARIND1* to *2* if the given string is the second independent variable.

IYTCOMMAX

IYTCOMMAX checks the arguments of a variable between parentheses and returns the following values:

1	the variable is dependent on the first independent variable
2	the variable is dependent on the second independent variable
3	the variable is dependent on both independent variables
4	the variable is dependent on the first independent variable at lower bound and on the second independent variable
5	the variable is dependent on the first independent variable at upper bound and on the second independent variable
6	the variable is dependent on the second independent variable at lower bound and on the first independent variable
7	the variable is dependent on the second independent variable at upper bound and on the first independent variable
8	the first independent variable is fixed at knot
9	the second independent variable is fixed at knot

The setting of the argument *ITIMEDELAY*:

0	no time delay
1	time delay at the first independent variable
2	time delay at the second independent variable

Valid combinations of *DECLASS,VARCLASS* and *return value*:

DECLASS	VARCLASS	return value
1,4,7,10,13	2,5	1
2,5,8,11,14	3,6	2

DECLASS	VARCLASS	return value
1,4,7,10,13	4,7	6,7
2,5,8,11,14	4,7	4,5
3,6,9,12,15	4,7	3
3,6,9,12,15	2,5	1
3,6,9,12,15	3,6	2

The values of the local variable *IYTCOMMAX2*:

-1	before comma variable at low boundary
-2	before comma variable at high boundary
-3	before comma first independent variable
-4	before comma second independent variable
-5	before comma variable fixed at knot

The values of the local variable *IYTCOMMAX3*:

-1	after comma variable at low boundary
-2	after comma variable at high boundary
-3	after comma first independent variable
-4	after comma second independent variable
-5	after comma variable fixed at knot

SETUPAB

This subprogram stores the first and second derivatives of the *Lagrange* polynomial at collocation points for each finite element.

A(i,j,k)	first derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the first independent variable
B(i,j,k)	second derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the first independent variable
A2(i,j,k)	first derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the second independent variable
B2(i,j,k)	second derivative of the *j-th Lagrange* polynomial at the *i-th* collocation point in the *k-th* finite element along the second independent variable

SETVARRANGE

This subprogram sets subscripts of the first *NLP* variable derived from state and control variables dependent on independent variables.

LANGE

This subprogram returns the dynamic length of a string variable or string array element.

JCOBI

This subprogram evaluates roots of *Lagrange* polynomials. [Villadsen and Michelsen, Solution of Differential Equation Models by Polynomial Approximation].

DFOPR

This subprogram evaluates derivatives of *Lagrange* polynomials. Villadsen and Michelsen, Solution of Differential Equation Models by Polynomial Approximation].

PLANAR

This subprogram evaluates roots and derivatives of *Lagrange* polynomials. [Finlayson, Nonlinear Analysis in Chemical Engineering].

DISPATCHER

This subprogram is the control module of the optimization package. It calls modules using unconstrained optimization, sequential optimization and solution strategy using control vector parameterization or modules associated with *VF13AD* and *GRG2* using simultaneous optimization and solution strategy.

PROCESSGRG2NLP

This module reads in *OPTIMINP.DAT* and invokes the *GRG2* package. The *GRG2* interface subroutine *GCOMP* has to be set up by the user. This module is invoked to solve unconstrained or constrained linear or nonlinear programming problems.

PROCESSGRG2DAP

This module reads in *OPTIMINP.DAT* and *OCFEGRG2IF.DAT* and invokes the *GRG2* package. This module is invoked to solve differential-algebraic optimization problems.

PROCESSSQPNLP

This module reads in *OPTIMINP.DAT* and invokes the *VF13AD* package. The *VF13AD* interface subroutines *SQPOF* and *SQPCONSTRAINTS* have to be set up by the user. This module is invoked to solve constrained linear or nonlinear programming problems.

PROCESSSQPDAP

This module reads in *OPTIMINP.DAT* and *OCFESQPIF.DAT* and invokes the *VF13AD* package. This module is invoked to solve differential-algebraic optimization problems.

INTERFACE

This module calls unconstrained optimization modules based on the *Hooke-Jeeves* or *Broyden-Fletcher-Goldfarb-Shanno* algorithm if option *1,2,6,8,11* or *12* selected.

HJBFGSSETUP

This module provides the appearance of dynamic memory allocation setting up *INTEGER*4* and *REAL*8* arrays as segments of *INTCORE* and *DPCORE* so that *HJBFGS* does not waste memory space on individual arrays and the memory is utilized most economically. This subroutine assigns pointers to *DPCORE* for each *REAL*8* array and to *INTCORE* for each *INTEGER*4* array.

INPUT

This subprogram reads *OPTIMINP.DAT* if option *1,2,6,8,11* or *12* is selected.

OUTPUT

This subprogram writes the optimal solution to *OPTIMOUT.DAT.*

SCALE

This subprogram scales optimizer variables from the interval *[a,b]* into the interval *[0,1]*.

VARIABLESCALING

This subprogram scales optimizer variables from the interval *[a,b]* into the interval *[L,U]*.

SCALEDX

This subprogram scales an optimizer variable from the interval *[a,b]* into the interval *[L,U]*.

UNSCALE

This subprogram unscales optimizer variables from the interval *[0,1]* into the interval *[a,b]*.

VARIABLEUNSCALING

This subprogram unscales optimizer variables from the interval *[L,U]* into the interval *[a,b]*.

UNSCALEDX

This subprogram unscales an optimizer variable from the interval *[L,U]* into the interval *[a,b]*.

SCALE3

This subprogram scales optimizer variables from the interval *[a,b]* into the interval *[0,1]* leaving the array of the unscaled variables unscaled.

VARIABLESCALING3

This subprogram scales optimizer variables from the interval *[a,b]* into the interval *[L,U]* leaving the array of the unscaled variables unscaled.

UNSCALE3

This subprogram unscales optimizer variables from the interval *[0,1]* into the interval *[a,b]* leaving the array of the scaled variables scaled.

VARIABLEUNSCALING3

This subprogram unscales optimizer variables from the interval *[L,U]* into the interval *[a,b]* leaving the array of the scaled variables scaled.

SCALE2

This subprogram scales an optimizer variable from the interval *[a,b]* into the interval *[0,1]*.

UNSCALE2

This subprogram unscales an optimizer variable from the interval *[0,1]* into the interval *[a,b]*.

CONSTRAINTSSCALING

```
If constraint < a then
constraint:=a
else if constraint > b then
constraint:=b
endif
```

CONSTRAINTSSCALING2

It is like *CONSTRAINTSSCALING* except leaving the objective function unscaled.

DIFF(a,b)

It returns DIFF=a-b. This subprogram is used to test underflow. It is called to increase the stepsize of the numerical differentiation, if it is too small. It is called as follows:

```
400     IF(1+HDIFF,1).EQ.0)THEN
        HDIFF = 10*HDIFF
        GOTO 400
        ENDIF
```

HJBFGS

The modules contained in *HJBFGS.FOR* are called if the user wants to solve unconstrained optimization problems by the *Hooke-Jeeves* or by the *BFGS* method, or differential-algebraic optimization problems containing initial-value ordinary differential equations by sequential optimization and solution strategy using control vector parameterization.

HJ

This subprogram finds the minimum of an unconstrained function using *Hooke-Jeeves* method. It is called if option *1* is selected.

BFGS

This subprogram finds the minimum of an unconstrained function using the *BFGS* method. It is called if option *2* is selected. The code is based on the article by J. E. Dennis, J. R. Jorge and J. More, SIAM Review, Vol. 19, No. 1, January 1977. The line search code is based on the algorithm in Reklaitis, Engineering Optimization.

HJ2

This subprogram uses *Hooke-Jeeves* method with sequential optimization and solution strategy to find the minimum of an unconstrained function using control vector parameterization. For each function evaluation, calls a 4-th order *Runge-Kutta* differential equation solver.

BFGS2

This subprogram uses *BFGS* method with sequential optimization and solution strategy to find the minimum of an unconstrained function using control vector parameterization. For each function evaluation, calls a 4-th order *Runge-Kutta* differential equation solver.

NUMDIFF

This subprogram returns the gradient vector of the objective function using central difference approximation. It is called from *BFGS*.

NUMDIFF2

This subprogram returns the gradient vector of the objective function using central difference approximation. It is called from *BFGS2*.

RK

This subprogram solves a system of initial value quasilinear ordinary differential equations using the fourth-order *Runge-Kutta* method. The user must provide the derivative evaluation module.

Appendix C - Mathematical Background

Euclidean Space

n-component vectors of n-tuples of real numbers span an n-dimensional Euclidean space. The vectors can be

multiplied by a scalar, there is an addition and scalar product $\sum_{i=1}^{n} x_i y_i$ defined, the magnitude of a vector is $|x| = (x^T x)^{1/2}$. For any two vector x and y in E^n the Cauchy-Schwartz Inequality holds. : $|x^T y| \leq |x| \cdot |y|$. The distance of 2 vectors x and y is defined as $\left(\sum (x_i - y_i)^2 \right)^{1/2}$.

Functional

A functional is a mapping of a function into the real line.

Hamilton-Jacobi-Bellman Equation

U	set of input values (controls)
B, B_x	Banach spaces
B^*	space of continuous linear functionals on B
X	state set
$\mathbb{R}$	real line
$\lambda, \partial V / \partial x$	continuous linear transformations of B_x into $\mathbb{R}$, that is, $\in B_x^*$
V(t,x)	a real-valued function defined on [T1,T2] $\times$ X
[T1,T2]	closed time-interval
$\Re \subset [T1,T2] \times X$	event space
S	target set $\subset \Re$
u^*	an admissible control that transfers (t_o, x_o) to S along a path lying entirely in $\Re$.
X	$\subset B_x$
x	state
t	time
u	control

The real-valued function H defined on $X \times B_x^* \times U \times [T1,T2]$ by the relation

$$H(x,\lambda,u,t) = L(x,u,t) + \langle \lambda, f(x,u,t) \rangle$$

is called the Hamiltonian

where $\langle \lambda, f(x,u,t) \rangle$ is the operation of λ on $f(x,u,t) \in B$

The partial differential equation
is called the Hamilton-Jacobi-Bellman equation.

$$\frac{\partial V(t,x)}{\partial t} + H\left(x,\frac{\partial V(t,x)}{\partial x},u^\circ\left(t,x,\frac{\partial V(t,x)}{\partial x}\right),t\right) = 0$$

Normed Linear Space

A vector space over which a norm $\|\mathbf{x}\|$ is defined called normed linear space having the following properties:

$$\|x\| = \sqrt{\sum x_i^2}$$

$\|\mathbf{x}\| \geq \mathbf{0}$

$\|\mathbf{x}\| = \mathbf{0}$ if and only if $\mathbf{x} = \mathbf{0}$

$\|\alpha\mathbf{x}\| = |\alpha|\ \|\mathbf{x}\|$

$(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$

$\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$

$\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$

$1\mathbf{x} = \mathbf{x}$

where α,β are scalars and $\mathbf{x},\mathbf{y}$ are vectors.

Metric Space

A vector space over which a distance $d(\mathbf{x},\mathbf{y})$ is defined is called a metric space having the following properties:

$d(\mathbf{x},\mathbf{y}) \geq 0$

$d(\mathbf{x},\mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$

$d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$

$d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z})$

Cauchy Sequence

If, in a metric space (X,d), the sequence $\{\mathbf{x}_n\}$ has the property that for any $\epsilon > 0$, there exists an integer N such that for all n and m greater than N, $d(\mathbf{x}_n,\mathbf{x}_m) < \epsilon$, the sequence is said to be a Cauchy sequence.

Complete Metric Space

If a metric space has the property that every Cauchy sequence converges, the space is called a complete metric space.

Banach Space

Let X be a normed linear space. Let the distance between any two points $\mathbf{x},\mathbf{y} \in X$ be $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|$. If (X,d) is a complete metric space, X is said to be a Banach space.