P. Deuflhard U. Nowak

Efficient Numerical Simulation and Identification of Large Chemical Reaction Systems

Invited talk at the Bunsen meeting, May 8-10, 1986,
University of Heidelberg

Preprint SC-86-1 (September 1986)

Konrad Zuse-Zentrum für Informationstechnik;
Heilbronner Straße 10; D-1000 Berlin 31

Contents

Abstract

New, highly efficient techniques for the numerical simulation and parameter identification of large chemical reaction systems are surveyed. The survey particularly addresses to chemists, which are understood to be potential users of the distributed software packages. These packages are written in the form of interactive dialogue systems – thus enabling chemists to concentrate on the chemistry of their problem. Large scale examples from chemical research environments are included.

1. Introduction

In recent years, the numerical simulation and identification of chemical reaction systems has played an increasing role in physical chemistry and already in chemical engineering - with the aim of optimization and optimal control of more and more complex chemical processes. Originally, the numerical work in this context has been done by chemists themselves, putting together known pieces software and adding user-oriented interfaces. As it turned out, however, the increasing complexity of the problems often led to limitations both from the accessible hardware and from the performance of the thus produced software.

At this point, numerical analysts got interested in the subject - as a scientific subject of its own. Of course, substantial progress in this direction could only be expected (and made) in reasonable cooperation with the eventual chemical users. The present article surveys certain efforts that have been made by the author and his research group towards the development of efficient, mathematically sound software for simulation and identification of large reaction networks. As the paper addresses to chemists mainly, the underlying mathematics will be widely suppressed. The reader should realize, however, that most of the progress to be reported has been made on the very basis of rather sophisticated mathematical thinking.

The authors are well aware of the fact that the whole scientific endeavor of realistic modeling of chemical reaction systems heavily relies on the quality of experimental measurements and the insight into possible chemical mechanisms. In this situation, the chemist is expected to need not just mathematical basis software but *expert systems* in order to be able to concentrate on the chemistry under consideration. The expert systems to be described have been written as interactive dialogue systems, which do not even require the knowledge of a high level programming language from the side of the chemical user. At the same time, the mathematical state-of-the-art is implemented in the core routines.

Mathematically speaking, a chemical reaction network is modeled by an initial value problem (IVP) for n ordinary differential equations (ODE's), say

$$y' = f(y, p), \qquad y(0) = y_0, \tag{1.1}$$

where the vector y represents chemical concentrations and the vector p represents reaction rate coefficients. If p is given, then the task is to integrate (1.1) numerically - which is the so-called *simulation* to be treated in section 2. If some of the parameters are unknown, then the task is to estimate these parameters from a comparison with experimental data - which is the so-called *identification* to be treated in section 3.

2. Large Scale Simulation

In this section, the software system LARKIN is described (mnemotechnically for LARge chemical KINetics). This package has been especially designed for *large* chemical networks - a situation that will arise as soon as the models get more and more realistic. The first two versions of LARKIN have been developed by the authors together with Bader - see [12,3]. These versions cover the case of constant temperature, pressure, and volume. A rather recent extension, which has been developed by Nowak and Walkowiak, also includes the thermodynamics see [26,30]. LARKIN consists of about 8.000 cards in version 2 and 11.000 cards in version 3 (with about 1/3 comment cards).

The main success of LARKIN depends upon a special chemical compiler, new stiff integrators, and a well-tuned sparse solver - parts that will be discussed now.

Chemical compiler. The input to be required from a chemical user will naturally just be a (possibly rather complex) system of chemical reactions together with given rate coefficients. For an illustration, see Figure 2.1, where each reaction line is followed by $(lnA, \Delta E)$.

```
*C
*C   HYDROGEN ABSTRACTION
*C
H + CH4           =>   H2 + CH3        (11.1, 12600)
H + C2K6          =>   H2 + C2B5       (11.1,  9700)
H + C3H8          =>   H2 + 1-C3H7     (11.1,  9700)
H + C3H8          =>   H2 + 2-C3H7     (11.0,  8300)
H + C4H10         =>   H2 + 1-C4H9     (11.1,  9700)
H + C4H10         =>   H2 + 2-C4H9     (11.0,  8400)
H + C6H14         =>   H2 + 1-C6H13    (11.0,  9700)
H + C6H14         =>   H2 + 2-C6H13    (10.7,  8400)
H + C6H14         =>   H2 + 3-C6H13    (10.4,  8400)
H + C2H4          =>   H2 + C2H3       ( 8.5,  4500)
H + C3H6          =>   H2 + C3H5       ( 8.5,  4500)
H + 1-C4H8        =>   H2 + 1-M-AL     ( 8.5,  4500)
H + 1-C4H8        =>   H2 + 3-BEN      ( 9.1, 11600)
H + 1-C5H10       =>   H2 + 1-E-AL     ( 7.7,  4500)
H + 1-C5H10       =>   H2 + 1-M-BEN    ( 8.5, 11000)
H + 1-C5H10       =>   H2 + 4-PEN      ( 9.0, 11600)
CH3 + H2          =>   CH4 + H         ( 9.5, 10200)
CH3 + CH4         =>   CH4 + CH3       ( 8.6, 14000)
CH3 + C2H6        =>   CH4 + C2H5      ( 9.0, 10600)
CH3 + C3H8        =>   CH4 + 1-C3H7    ( 9.0, 10100)
CH3 + C3H8        =>   CH4 + 2-C2H7    ( 8.9, 10100)
CH3 + C4H10       =>   CH4 + 1-C4H9    ( 9.1, 11600)
```

Figure 2.1: Reaction system input from [22] - 22 reactions out of 240.

From such an input, the chemical compiler automatically generates the right-hand side f in 1.1, the Jacobian (n,n)-matrix f_y and its associated sparse pattern. The routines for evaluating f, f_y are realized by setting pointers in formal subroutines – a technique, which permits treatment up to very large systems. In order to give a rough idea, a small part of an ODE system associated with Figure 2.1 is presented in Figure 2.2

```
 R(  1)=C(  1)*Y( 30)
 R(  2)=C(  2)*Y( 31)
 R(  3)=C(  3)*Y( 31)
 R(  4)=C(  4)*Y( 34)
      .
      .
      .
 DY(  1) = +R(  1)+R(  2)+R(  4)+R( 11)+R( 12)+R( 15)-R( 23)-R( 24)
@   -R( 25)-R( 26)-R( 27)-R( 28)-R( 29)-R( 30)-R( 31)-R( 32)-R( 33)
@   -R( 34)-R( 35)-R( 36)-R( 37)-R( 38)+R( 39)+R( 56)+R( 73)+R( 83)
@   +R( 98)-R(114)-R(115)-R(116)-R(117)-R(118)-R(119)-R(120)-R(121)
@   +R(151)+R(152)+R(154)+R(155)+R(158)+R(160)+R(176)+R(178)-R(186)
@   -R(186)-R(187)-R(188)
 DY(  2) = +R(  1)+R(  3)+R(  3)+R(  5)+R(  6)+R(  8)+R( 13)+R( 14)
@   +R( 17)+R( 23)-R( 39)-R( 41)-R( 42)-R( 43)-R( 44)-R( 45)-R( 46)
@   -R( 47)-R( 48)-R( 49)-R( 50)-R( 51)-R( 52)-R( 53)-R( 54)-R( 55)
@   +R( 57)+R( 74)+R( 84)+R( 99)-R(122)-R(123)-R(124)-R(125)-R(126)
@   -R(127)-R(128)-R(129)+R(153)+R(157)+R(159)+R(161)+R(164)+R(165)
@   +R(170)+R(177)+R(181)+R(185)-R(189)-R(189)-R(190)-R(191)-R(192)
 DY(  2) = DY(  2)-R(193)-R(194)-R(210)-R(211)-R(212)-R(213)
 DY(  3) = +R( 11)+R( 13)+R( 19)+R( 20)+R( 20)+R( 21)+R( 32)+R( 49)
@   +R( 66)+R( 81)-R( 83)-R( 84)-R( 85)-R( 86)-R( 87)-R( 88)-R( 89)
@   -R( 90)-R( 91)-R( 92)-R( 93)-R( 94)-R( 95)-R( 96)-R( 97)+R(108)
@   +R(121)-R(140)-R(141)-R(142)-R(176)+R(179)+R(180)+R(182)-R(187)
@   -R(191)-R(196)-R(201)-R(201)-R(202)-R(203)-R(211)-R(215)-R(216)
@   -R(221)-R(221)-R(222)-R(223)
```

Figure 2.2: Part of ODE system automatically generated from the reaction system of Figure 2.1 (written in explicit form just for presentation purposes).

This chemical compiler allows one to include several checks for the consistency of the input - thus helping to avoid input errors by the user up to some level.

Stiff integrator. ODE systems arising from chemical kinetics typically are known to be "stiff". In order to convey a rough idea of what "stiff integration" means, consider the trivial scalar ODE (cf. Dahlquist [5]):

$$y' = -40y, \qquad y(0) = 1 \tag{2.1}$$

Its analytical solution is well-known to be

$$y(t) = exp(-40t).$$

Important is that y is "rapidly decaying".

Numerical treatment of (2.1) will produce a discretized system, with stepsize $h = \Delta t$ say, replacing the continuous solution $y(t)$ by a discrete solution $v(t_k)$ over a grid with $t_k = k \cdot h, k = 0, 1, \ldots$. There are a variety of discretization methods to be applied. For illustration purposes, the so-called explicit and implicit Euler method are exemplified - with $v_k = v(t_k)$. *Explicit* Euler method applied to (2.1) yields

$$v_{k+1} = v_k + h(-40v_k) \tag{2.2}$$

whereas implicit Euler method yields

$$v_{k+1} = v_k + h(-40v_{k+1}) \tag{2.3}$$

Typical results of (2.2) and (2.3) with h=1/20 are graphically represented in Figure 2.3. together with y.

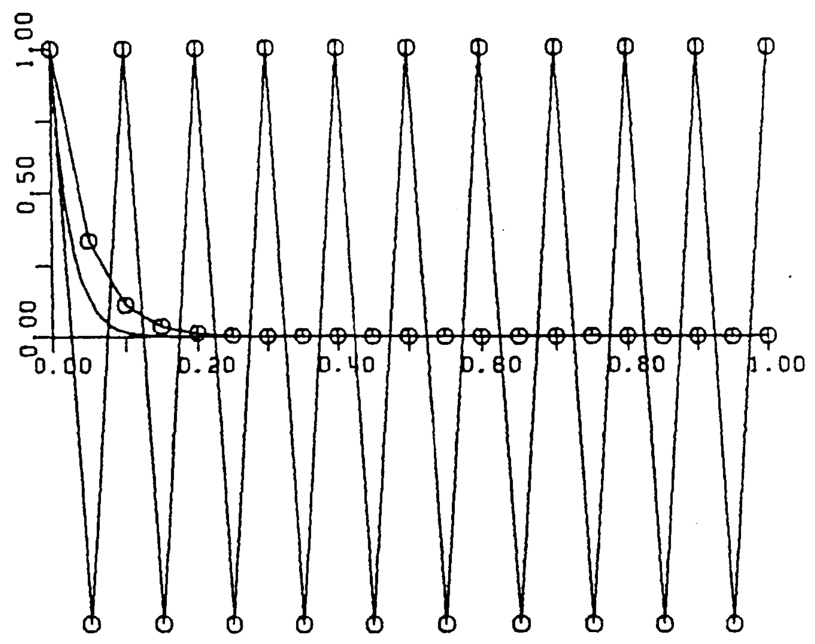


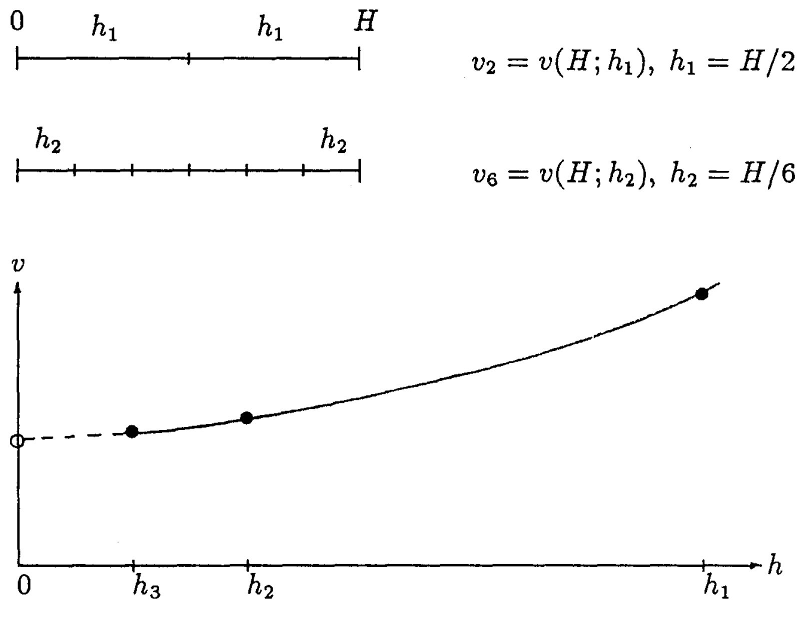
Figure 2.3: Explicit Euler method applied to (2.1) with h=1/20 yields oscillations, whereas implicit Euler method yields qualitatively correct behavior compared with the solution.

The oscillations arising in the explicit case (2.2) can only be avoided by serious stepsize reduction - which means a significant and unnecessary increase of computational work. On the other hand, the implicit case in line (2.3), which yields qualitative agreement with the analytic solution, represents an algebraic equation, which needs to be solved. This shows the main feature: systems of "stiff" ODE's require the solution of systems of linear equations inside the discretization method. Even though this involves an increased amount of computing time per discretization step, the number of steps is significantly decreased (as compared to the explicit methods) - thus leading to an overall gain in computing time.

The special discretization methods used in LARKIN are of *extrapolation* type. The idea of extrapolation, which dates back at least) to Richardson in 1911 [28], may be explained as follows. Suppose the solution $y(0) = y_0$ is given, and $y(H)$ is wanted, where H is usually called the ***basic stepsize.*** Assume that a discretization method is applied repeatedly - first with a stepsize $h_1 = H/2$, second with $h_2 = H/6$, etc. Then both $v_2 = v(H; h_1)$, $v_6 = v(H; h_2)$ etc. are approximations of the solution. Now, recalling that with

$$y(H) = \lim_{h \to 0} v(H; h)$$

it is possible to construct an interpolating polynomial through $(h_1, v(H; h_1))$, $(h_2, v(H; h_2))$ etc. and evaluate it at $h = 0$ which means extrapolating it to $h = 0$. For illustration, see Figure 2.4.



0
h_1
h_1
H
$v_2 = v(H; h_1), \; h_1 = H/2$
h_2
h_2
$v_6 = v(H; h_2), \; h_2 = H/6$
v
0
h_3
h_2
h_1
h

Figure 2.4: Extrapolation from h_1, h_2, h_3, to $h = 0$

The extrapolation technique thus is an easy way of producing discretization methods of higher order. In addition whenever the discretization is formally invariant under the transformation $h \to -h$, then an h^2-extrapolation can be applied. In LARKIN, the following discretization methods are implemented:

(a) the semi-implicit mid-point rule due to [2] together with h^2- extrapolation,

(b) the semi-implicit Euler discretization from [7] together with h- extrapolation.

Both methods are combined with an automatic control of the (order and the) basic stepsize H, which is crucial for efficiency in chemical kinetics applications - see [6]. The choice of the integrators (a) or (b) is left to the user. Comparisons with the widely used BDF-type methods of Gear [20] and Hindmarsh [21] justify the implementation of the more recent extrapolation integrators - see [12,3,2,7].

Sparse solver. As already mentioned, the numerical integration of "stiff" ODE systems involves the solution of systems of linear equations. The arising matrices have the typical structure

$$M := I - \gamma H f_y$$

for some scalar factor $\gamma > 0$. In most complex reaction networks the matrix f_y contains comparitively few non-zero entries — M is then said to be "sparse". Examples of sparse patterns of such matrices are given in Figure 2.5.

In actual computation, computing time can be significantly saved, if the sparseness is exploited. In [14] one of the most efficient general linear sparse solvers has been adapted to the special needs of the stiff extrapolation integrators. The gain in efficiency is indicated below Figure 2.5: it is the more significant, the larger and sparser the arising matrices are. Finally, note that sparseness exploitation helps to save storage. If one included the typical thermodynamic ODE's in the way chemists are used to do it (compare [19]), then the sparse matrices would fill up. An alternative sparseness preserving treatment has been suggested in [30], which leads to so-called implicit ODE systems of the form

$$B(y)y' = f(y) \tag{2.4}$$

where B is another sparse matrix. This type of ODE's can be solved by special variants of the known stiff integrators - see [27] for the BDF method and [10,7] for extrapolation methods. The application to chemical kinetics including thermodynamics in LARKIN will be published in [26] together with further extensions.

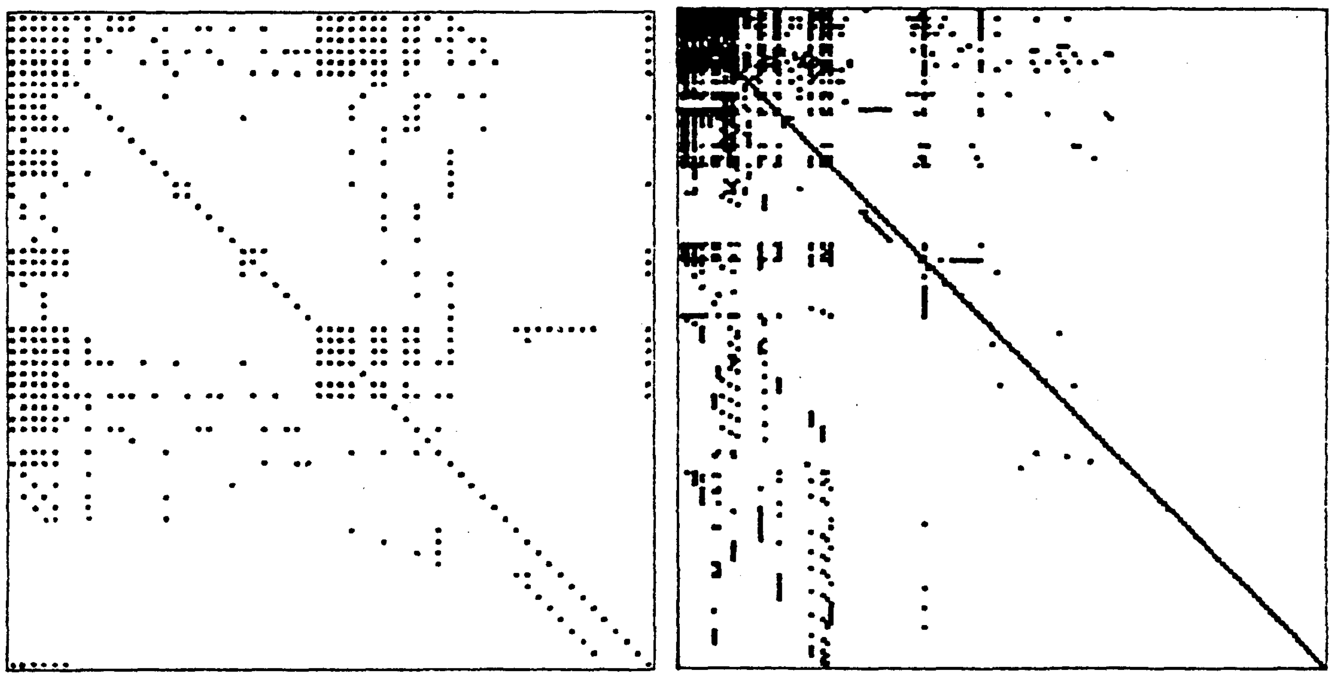


Figure 2.5: Examples of sparse patterns of reaction network matrices from [14]. Gain in linear equation solving time for sparse solver versus full mode solver: left 0.7:3.8, right 2.2:59.6 (sec. CPU on IBM 3081 D)

Quasi-stationary state approximation (QSSA). A rather popular technique in both mathematical analysis and physical chemistry is the so-called *singular perturbation* treatment, also called QSSA in the chemical literature. This technique involves active time rescaling for different reaction ODE's - leading to systems of the kind

$$\begin{aligned} y' &= f(y,z) \\ \epsilon z' &= g(y,z) \end{aligned} \tag{2.5}$$

for some "sufficiently small" parameter ϵ. Then QSSA means putting $\epsilon := 0$ deliberately, thus obtaining a differential algebraic system

$$\begin{aligned} y' &= f(y,z) \\ 0 &= g(y,z) \end{aligned} \tag{2.6}$$

Suppose, for the time being, that (2.6) can be solved uniquely. Then the following strategies are typical:

Table 2.1: [31]. Comparison of two QSSA approaches for the small n-hexane model in terms of CPU-times.

25 ODE's	0.48 sec.
13 ODE's, 12 AE's	no solution obtained
18 ODE's, 7 AE's	0.49 sec.

(I) Whenever the algebraic part can be solved analytically, yielding some $z = z(y)$, say, then a smaller ODE system can be obtained

$$y' = f(y, z(y)) = F(y) \tag{2.6'}$$

If this system is "non-stiff", then explicit discretization methods can be used, which may mean a gain in computing time.

(II) In most cases the above treatment is not possible. Then (2.6) must be solved directly by means of an integrator for differential-algebraic systems.

The main difficulty, however, in such a treatment is that (2.6) need not generally have a unique solution. Then both (I) and (II) will fail, (assuming that (II) uses a mathematically sound integrator). In order to illustrate the situation, an example is presented.

Example for QSSA [31]. Consider the small n-hexane model described in [31], which consists of 47 chemical reactions and 25 ODE's. Among the 25 chemical species, 13 species are stable, while 12 are free radicals. Hence, in a first QSSA treatment, one would come up with 13 ODE's and 12 algebraic equations (AE's). For the special example only strategy (II) is possible. As it turns out in this example, treating the 12 radicals by QSSA leads to strict non-uniqueness. Moreover, after all that analytical pre-processing, the computing time in the successful QSSA run was about the one of the standard ODE approach.

Generally speaking implicit (or semi-implicit) discretization intrinsically perform some internal rescaling. Indeed a successful ϵ-choice may be a rather difficult job already in mildly realistic examples - compare [23]. Therefore, LARKIN realizes the above mentioned discretization methods in a *scaling invariant* manner - see section 2.3 in [12]. For the performance, this feature is crucial. Last not least, QSSA may anyway yield a poor approximation for the true solution - compare [17].

Table 2.2: Typical computing times with LARKIN (FORTRAN double precision on IBM 3081 D) for various chemical reaction models.

example	number species (=ODE's)	number reactions	array storage	CPU-time [sec]
n-hexane [22]	59	240	20K	2.3
RNA [29]	352	770	120K	27.6
n-octane [30]	41	181	50K	11.3
$C_2H_2 - O_2$ [18]	138	959	150K	17.3

Summary. In order to give an idea, several typical examples that have been computed with LARKIN are listed in Table 2.2.

This may mean to chemists that now rather realistic complex reaction systems are open to a successful numerical simulation.

3. Large Scale Identification

This section deals with details of the software package PARKIN developed by the authors [24,25] (mnemotechnically for PARameter identification in chemical KINetics). The mathematical problem of identification (the so-called *inverse problem*) is much more complicated than that of simulation (the so-called *direct* problem, which is just an IVP for ODE's). Let $y(t, p)$ denote a smooth solution of (1.1) for a given parameter vector (of length q)

$$p = p(T) = (p_1, ..., p_q)$$

depending on the temperature T. Assume there is a set of experimental data

$$(t_1, z_1), ..., (t_m, z_m)$$

associated with corresponding measurement tolerances dz_j, where z_j, dz_j, are vectors of length n. Then the typical approach is defined via the special minimization problem

$$I(p) := \sum_{j=i}^{m} \sum_{i=1}^{n} (dy_i(t_j, p))^2 = min \tag{3.1}$$

where

$$dy_i(t_j, p) := (y_i(t_j, p) - z_{ji})/dz_{ji}$$

represents some relative deviation of model and data at point t_j.

Remark 1. Note that the underlying statistical reasoning (maximum likelihood) only applies, if the dz_j are given by the user *together* with the z_j an important feature, which is often overlooked by users. Only if all the dz_j are the same, then they can be dropped. Typically, *relative* accuracy of measurements is achieved, which implies that

$$dz_{ji} = \epsilon_z z_{ji} \tag{3.2}$$

for some estimated ϵ_z (mostly 10^{-1} to 10^{-3} in experiments).

Remark 2. In typical application, only part of the species are open to measurements: in that case, the corresponding dy_i just vanish formally.

Nonlinear least squares solution. The above problem (3.1) is highly nonlinear with respect to p, even though p enters just linearly into the right-hand side f of (1.1). Because of this feature, the *Gauss-Newton* method in the form suggested by the author [8] is applied - a method, which is well-known to be able to cope with highly nonlinear problems (from other areas of applications, e.g. space technology). If one writes (3.1) in the form

$$\begin{aligned} I(p) &= F(p)^T F(p) = \min \\ F(p) &: \bar{m} - \text{vector},\ \bar{m} \leq mn \end{aligned} \tag{3.1'}$$

then the Gauss-Newton iteration reads

$$\begin{array}{rrcl} a) & p^0 & : & \text{given starting guess} \\ & p^{k+1} & := & p^k + \lambda_k \Delta p^k, 0 < \lambda_k \leq 1 \\ b) & \Delta p^k & := & -F'(p^k)^\dagger F(p^k) \end{array} \tag{3.3}$$

herein $F'(p)$ denotes the Jacobian $(\bar{m}, q)$-matrix and the superscript † marks the Moore-Penrose pseudoinverse (which permits a formal representation for the solution of a *linear* least squares problem). The damping factors λ_k are chosen according to a special strategy given in [8] and especially extended to the problem class in [25]. Moreover, in order to observe the *positivity constraints* $p \geq 0$, the authors of PARKIN took inspiration from the chemical Arrhenius law and used the (then so-called) *Arrhenius transformation*

$$p = \exp(u) \tag{3.4}$$

This means that the iteration is realized in terms of u^k rather than p^k - which may induce a considerable improvement of performance as shown in [25]. This way of handling the inequalities is certainly rather natural in terms of chemistry. A further advantage of this transformation comes up when looking into details of the structure of the Jacobian matrix. In fact, $F'(u)$ has the form

$$F'(u) = \begin{bmatrix} y_u(t_1) \\ \vdots \\ y_u(t_m) \end{bmatrix}$$

where y_u denotes the *sensitivity* (n, q)-matrix, which is a solution of the so-called variational equation.

$$y_u' = f_y(y, u) y_u + f_u(y, u), \quad y_u(0) = 0 \tag{3.5}$$

Herein, as a consequence of the linear appearance of p in f, the matrix

$$f_u(y, u) = f_p(y) \frac{dp}{du} = f_p(y) p$$

consists only of terms already appearing in f - which permits clear savings of computing time and storage. Of course, the variational equations (3.5) can be solved numerically by means of the same techniques as described in section 2 (chemical compiler, stiff integrators, sparse techniques). Once more, significant savings of CPU time and storage are possible by taking the special structure of the problem class into account.

Identifiability. It is clear to every careful scientist that not each combination of model and data will permit an actual identification of the parameters p. This structure of the problem shows up in the performance of the algorithm. Even more, the selected algorithm in PARKIN permits 3 effective and cheap checks of identifiability:

a) The numerical solution of the linear least squares problems (3.3.b) yields an estimate of the Jacobian condition number, the so-called sub-condition number $sc(\cdot)$ - see [11]. Under the hypothesis (3.2), a reasonable solution of the problem (3.1) must satisfy

$$\epsilon_z sc(F'(u)) < 1$$

b) The convergence of the Gauss-Newton iteration (with $\kappa = 1$) is guaranteed only, if some so-called incompatibility factor κ satisfies

$$\kappa < 1$$

Note that $\kappa = 0$, if model and data are fully compatible (i.e. $F = 0$ at the solution).

c) After convergence of the iteration, PARKIN applies an a-posteriori perturbation analysis. However, if the linear analysis yields parameter perturbations dp from measurement perturbations, then the nonlinear analogon yields

$$dp/(1-\kappa)$$

This insight is due to Bock [4]. Whenever all three checks are passed satisfactorily, then the result of PARKIN can be safely interpreted by the chemist - but only in this situation. An illustrative example where these checks were not passed, can be found in [25].

A final word of warning: there are optimization routines around, which supply a solution by all means; however, if a chemist wants to rely on his results in terms of the underlying chemistry, then there is essentially no way round the above checks. On the other hand, just for this strict scientific requirement, PARKIN appears to be less popular than LARKIN. (Future developments of PARKIN will give special further information to the chemist, about where information is missing.)

Summary. The present form of PARKIN makes it certainly a important tool in a research environment of physical chemistry. As the treated inverse problem is much more complicated that the direct problem, the handling of PARKIN requires more skill than that of LARKIN.

Acknowledgment. This work was supported by the DFG under SFB 123 (University of Heidelberg).

References

[1] R. C. Aiken (ed.): *Stiff Computation.* Oxford Univ. Press (1985).

[2] G. Bader, P. Deuflhard: *A semi-implicit mid-point rule for stiff systems of ordinary differential equations.* Numer. Math. 41, pp. 373-398 (1983).

[3] G. Bader, U. Nowak, P. Deuflhard: *An advanced simulation package for large chemical reaction systems.* Section 5.5 in [1], pp. 255-264 (1985).

[4] H. G. Bock: *Numerical Treatment of Inverse Problems in Chemical Reaction Kinetics.* In [15], pp. 102-125 (1981).

[5] G. Dahlquist: *A special stability problem for linear multistep methods.* BIT 3, pp. 27-43 (1963).

[6] P. Deuflhard: *Order and Stepsize Control in Extrapolation Methods.* Numer. Math. 41, pp. 399-422 (1983).

[7] P. Deuflhard: *Recent Progress in Extrapolation Methods for Ordinary Differential Equations.* SIAM Rev. 27, pp. 505-535 (1985).

[8] P. Deuflhard, V. Apostolescu: *A Study of the Gauss-Newton Method for the Solution of Nonlinear Least Squares Problems.* In: Frehse, Pallaschke, Trottenberg (ed): Special Topics of Applied Mathematics. North-Holland, pp. 129-150 (1980)

[9] P. Deuflhard, E. Hairer (ed.): *Numerical Treatment of Inverse Problems in Differential and Integral Equations.* Birkhäuser, Progr. Sci. Comp. 2 (1983).

[10] P. Deuflhard, U. Nowak: *Extrapolation integrators for quasilinear implicit ODE's.* Univ. Heidelberg, SFB 123: Tech. Rep. Nr. 332 (1985).

[11] P. Deuflhard, W. Sautter: *On Rank-Deficient Pseudoinverses.* Lin. Alg. Appl. 29, pp. 91-111 (1980)

[12] P. Deuflhard, G. Bader, U. Nowak: *LARKIN - A Software Package for the Numerical Simulation of LARge Systems Arising in Chemical Reaction KINetics.* In [15], pp. 38-55 (1981).

[13] P. Deuflhard, E. Hairer, J. Zugck: *One-Step and Extrapolation Methods for Differential-Algebraic Systems.* Univ. Heidelberg, SFB 123: Tech. Rep. Nr. 318 (1985).

[14] I. S. Duff, U. Nowak: *On Sparse Solvers in a Stiff Integrator of Extrapolation Type.* Harwell, Tech. Rep. CSS 144 (1985).

[15] K. H. Ebert, P. Deuflhard, W. Jaeger (ed.): *Modeling of Chemical Reaction Systems.* Springer Ser. Chem. Phys. 18 (1981)

[16] C. Esser, U. Maas, J. Warnatz: *Chemistry of the auto-ignition in hydrocarbon-air- mixtures up to octane and its relation to engine knock.* International Symposium on Diagnostics and Modeling of Combustion in Reciprocating Engines, Tokyo, (1985)

[17] L. A. Farrow, D. Edelson: *The steady-state approximation: fact or fiction?* Int. J. Chem. Kin. 6, pp. 787-800 (1974)

[18] M. Frenklach et al.: *Mechanism of soot formation in acetylene-oxygene mixtures.* In: Combust. Sci. Technol., in press, (1986)

[19] W. C. Gardiner, B. F. Walker, C. B. Wakefield: *Mathematical methods for modeling chemical reactions in shock waves.* In: A. Lifshitz (ed.): Shock Waves in Chemistry (1981).

[20] C. W. Gear: *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, NJ. (1971).

[21] A. C. Hindmarsh: *GEAR-ordinary differential equation system solver.* Lawrence Livermore Nat. Lab. Tech. Rep. UCID-30001, Rev. 3 (1974).

[22] G. Isbarn, H. J. Ederer, K. H. Ebert: *The Thermal Decomposition of n-Hexane: Kinetics, Mechanism, and Simulation.* In [15], pp. 235-248 (1981).

[23] Mao Zhu-fan: *SCALE - A Fortran Program for a Scaling Procedure.* Roy. Inst. Technol. Stockholm, Dep. Numer. Anal. Comp. Sci.: Tech. Rep. TRITA-NA-8314 (1983).

[24] U. Nowak, P. Deuflhard: *Towards Parameter Identification for Large Chemical Reaction Systems.* In [6], pp. 13-26 (1983).

[25] U. Nowak, P. Deuflhard: *Numerical Identification of Selected Rate Constants in Large Chemical Reaction Systems.* Appl. Num. Math. 1, pp. 59-75 (1985)

[26] U. Nowak, D. Walkowiak: *Extension of LARKIN - Rev. 3.1.* In preparation.

[27] L. Petzold: *A Description of DASSL: A Differential/ Algebraic System Solver.* Proc. IMACS World Congress 1982.

[28] C. Richardson: *The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam.* Phil. Trans. Roy. Soc. London, A 210, 307-357 (1911).

[29] F. W. Schneider, M. Heinrichs, T. Poll: Private communication. (1980).

[30] D. Walkowiak: *Numerische Behandlung grosser adiabater chemischer Reaktionssysteme.* Univ. Heidelberg, Inst. Angew. Math.: Diplomarbeit (1986).

[31] J. Zugck: *Numerische Behandlung linear-impliziter Differentialgleichungen mit Extrapolationsmethoden.* Univ. Heidelberg, Inst. Angw. Math.: Diplomarbeit (1984).