NASA Technical Memorandum 83590

A Comparison of the Efficiency of Numerical Methods for Integrating Chemical Kinetic Rate Equations

Krishnan Radhakrishnan
Lewis Research Center
Cleveland, Ohio

Prepared for the
1984 JANNAF Propulsion Meeting
New Orleans, Louisiana, February 7-9, 1984

NASA

A COMPARISON OF THE EFFICIENCY OF NUMERICAL METHODS FOR INTEGRATING CHEMICAL KINETIC RATE EQUATIONS[1]

K. Radhakrishnan[2]
NASA Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

A comparison of the efficiency of several algorithms recently developed for the efficient numerical integration of stiff ordinary differential equations is presented. The methods examined include two general-purpose codes EPISODE and LSODE and three codes (CHEMEQ, CREK1D and GCKP84) developed specifically to integrate chemical kinetic rate equations. The codes are applied to two test problems drawn from combustion kinetics. The comparisons show that LSODE is the fastest code currently available for the integration of combustion kinetic rate equations.

An important finding is that an iterative solution of the algebraic energy conservation equation to compute the temperature can be more efficient than evaluating the temperature by integrating its time-derivative.

INTRODUCTION

Many practical problems arising in chemically reacting flows require the simultaneous numerical integration of large sets of chemical kinetic rate equations. Examples of such problems include the development and validation of reaction mechanisms, combustion of fuel-air mixtures, and pollutant formation and destruction. The rate equations for chemical species constitute a set of coupled first order ordinary differential equations (ode's) of the type

$$\left.\begin{aligned} &\frac{dn_i}{dt} = f_i\,(n_k, T) \quad i, k = 1 - NS \\ &n_i(t = 0) = n_{i,0} \\ &T(t = 0) = T_0 \end{aligned}\right\} \qquad (1)$$

where, n_i is the mole number of species i (kmole i/kg mixture), T is the temperature and NS is the total number of species involved in the reaction; the initial values $n_{i,0}$ (i = 1 - NS) and T_0 and the function f_i (i = 1 - NS) are given.

The initial value problem may be stated as follows. Given, (i) at time t = 0, initial values for n_i (i = 1 - NS) and temperature, (ii) the pressure, and (iii) the reaction mechanism; find the mixture composition and temperature at the end of a prescribed time interval[3].

Multi-dimensional modeling of reactive flows requires the integration of the system of ode's given by equation (1) at several thousand grid points. In addition, at each grid point, the solution to equation (1) may be required several times per numerical simulation. To make such calculations practicable, it is necessary to have a very fast batch chemistry integrator.

The major problem associated with the numerical solution of the system (1) of equations by classical methods (such as the popular explicit Runge-Kutta method) is as follows. These equations are often characterized by widely varying time constants. To insure stability of the numerical solution, classical methods are restricted to using very small steplengths which are determined by the smallest time constants. However, the time for all chemical species to reach near-equilibrium values is

---

1 Work partially funded by NASA Grant NAG3-147.

2 NRC-NASA Research Associate; on leave from The University of Michigan, Dept. of Mechanical Engineering and Applied Mechanics, Ann Arbor, Michigan 48109.

3 In this paper attention is restricted to adiabatic, constant pressure (hence, isenthalpic), exothermic chemical reactions.

determined by the largest time constant. As a result, the computation time required to solve a practical chemical kinetics problem by classical methods can become excessive.

In the present study we examine several techniques that have been proposed for the integration of differential equations with widely different time constants. The codes examined in this work include the general-purpose codes EPISODE and LSODE[1-3], and the special-purpose (for chemical kinetic calculations) codes CHEMEQ[4], CREK1D[5-7], and GCKP84[8,9]. In addition, the explicit fourth-order Runge-Kutta-Merson differential equation solver (IMSL Routine DASCRU) is used to illustrate the problems associated with the solution of the system (1) of ode's by a classical method. These codes are summarized in Table I. The above codes are applied to two test problems drawn from combustion kinetics and details of the computational work (including computer time), required by these methods are presented. In this paper, the total computer time required by each code to solve the test problems is used as a measure of its efficiency.

Discussions with Prof. D. T. Pratt of the University of Washington were most helpful. Dr. A. C. Hindmarsh of Lawrence Livermore Laboratory provided copies of EPISODE and LSODE.

TEST PROBLEMS

The algorithms summarized in Table I were applied to two test problems drawn from combustion kinetics. Both problems describe adiabatic, constant pressure transient batch chemical reaction and include all three regions of interest to a combustion researcher -- induction, heat release, and equilibration.

Test problem 1, taken from Pratt[10], describes the ignition and subsequent combustion of a mixture of 33 percent carbon monoxide and 67 percent hydrogen with 100 percent theoretical air, at a pressure of ten atmospheres and 1000 K initial temperature. It is comprised of 12 reactions which describe the time evolution of eleven species. Test problem 2, taken from Bittker and Scullin[9], describes the ignition and subsequent combustion of a stoichiometric mixture of hydrogen and air, at a pressure of two atmospheres and 1500 K initial temperature. It involves 30 reactions which describe the time evolution of fifteen species. The reaction mechanisms for both test problems are given in Radhakrishnan[11]. Both test problems were integrated over a time interval of 1 ms in order to obtain near-equilibration of all chemical species.

Figures 1 and 2 present the variations with time of the temperature and the chemical species mole fractions for test problems 1 and 2, respectively. These solutions were generated with LSODE using a low value (10^{-5}) for the relative error tolerance.

EVALUATION OF TEMPERATURE

Of the codes tested, only CREK1D and GCKP84 were written explicitly for the integration of exothermic, non-isothermal, combustion rate equations. These therefore have built-in procedures for calculating the temperature. For the other codes, the temperature was computed using one of two different methods, labelled as methods A and B, and described below.[4]

In method A, the temperature was calculated from the mole numbers and the initial mixture enthalpy using the enthalpy conservation equation

$$\sum_{i=1}^{NS} n_i h_i = h_0 = \text{constant} \qquad (2)$$

where, h_i is the molal-specific enthalpy of species i (J/kmol) and h_0 is the mixture mass-specific enthalpy (J/kg). The algebraic equation (2) was solved for the temperature using a Newton-Raphson iteration technique with a user-supplied relative error tolerance, ERMAX. In this method, the temperature is not an explicit independent variable so the number of independent ode's is equal to the number (NS) of species and the Jacobian matrix ($J_{ij} = \partial r_i/\partial n_j$; i,j = 1 - NS) is of size NS x NS. The integrator therefore tracks only the solution for the species mole numbers.

---

[4] The following convention was adopted in naming these other codes: those using temperature method A were given the suffix A (e.g. LSODE-A, EPISODE-A, etc.) and those using temperature method B were given the suffix B (e.g. CHEMEQ-B, DASCRU-B, etc).

In method B, the temperature was treated as an additional independent variable and evaluated by integrating its time-derivative obtained by differentiating equation (2) and given by

$$\frac{dT}{dt} = \frac{-\sum_{i=1}^{NS} f_i h_i}{\sum_{i=1}^{NS} n_i c_{p,i}} \qquad (3)$$

where, $c_{p,i}$ is the constant-pressure specific heat of species i (J/kmol K). This increases the number of independent ode's to NS+1, and the computation of the Jacobian matrix (of size NS+1 x NS+1) involves the calculation of 2NS+1 additional terms. In this method, the integrator tracks the solutions for both the temperature and the species mole numbers.

RESULTS

The numerical techniques summarized in Table I were applied to the two test problems discussed above. All codes were run on the NASA Lewis Research Center's IBM 370/3033 computer using single-precision accuracy, except GCKP84 which was in double precision. A typical computational run consisted of initializing the species mole numbers, temperature and CPU time. The integrator was then called with values for the necessary input parameters[5]. On return from the integrator the total computer time (CPU) required to solve the problem was calculated. In addition, the following performance indicators were recorded: total number of steps (NSTEP), total number of functional (i.e. derivative) evaluations (NFE), and total number of Jacobian matrix evaluations (NJE, = 0 for CHEMEQ and DASCRU).

Figures 3 and 4 present the computational work (expressed as the CPU time in seconds required on the NASA Lewis Research Center's IBM 370/3033 computer) plotted against the relative error tolerance, EPS, for test problems 1 and 2, respectively. Note that for EPISODE, EPS is a mixed relative and absolute error criterion -- relative for species with initially non-zero mole numbers and for temperature (method B); and absolute for species with initially zero mole numbers. Also shown on figures 3 and 4 are the CPU times required by the explicit Runge-Kutta method for one value of EPS. For this study, the value of ERMAX (the relative error allowed in the Newton-Raphson iteration procedure used in method A to solve the algebraic energy equation) was set equal to EPS, to make comparisons between methods A and B meaningful. For the same reason, with LSODE-B, the absolute error tolerance for the temperature was set equal to zero.

To facilitate comparisons of efficiency, the values for the performance parameters NSTEP, NFE, and NJE are presented in Tables II and III for test problems 1 and 2, respectively. For each method (except DASCRU) and problem, these values correspond to the value of EPS that resulted in the least CPU time to solve the problem.

For test problem 1, very small values for EPS had to be used for EPISODE (fig. 3). For values of EPS $\geq 5\times10^{-6}$, EPISODE predicted little or no change in the composition and temperature after an elapsed time of 1 ms. Similar remarks apply to test problem 2 (fig. 4), for which values of 10^{-4} and 10^{-3} had to be used for EPISODE-A and EPISODE-B, respectively. Although the runs with EPISODE-B and EPS $\geq 5\times10^{-4}$ were successfully completed, the solutions (especially for minor species) were significantly different from those given in figure 2. With GCKP84 and EPS = 10^{-2}, the solution for test problem 1 exhibited serious instability and so this run was terminated. A more detailed discussion of the accuracy of the codes tested in this study can be found in Radhakrishnan[11].

Figures 3 and 4 and Tables II and III illustrate the difficulty associated with using a classical method (in this case the explicit Runge-Kutta method) to integrate combustion kinetic rate equations. The CPU times required for the two test problems are approximately 1 and 16 minutes respectively. The use of this technique would make multidimensional modeling of practical combustion devices prohibitively expensive

Examination of figure 3 shows that the difference in computational work required by methods A and B is small for test problem 1, with method B being more efficient. For test problem 2 (figure 4), the difference is small for large values of EPS. But for small values of EPS the difference is more marked, with method A being significantly superior to method B. A comparison of figures 1 and 2 shows that the temperature-time profile is steeper for test problem 1 indicating a stronger coupling between the species and the temperature. This may explain why the inclusion of the temperature as an additional independent variable works well for test problem 1. But for test problem 2 the additional

---

[5]For a detailed discussion of the parameters required as input by each code see Radhakrishnan[11].

work in computing the temperature rate and the temperature dependent terms in the Jacobian matrix does not lead to increased efficiency.

Figures 3 and 4 and Tables II and III show that LSODE and CREK1D are superior to the other codes. Although GCKP84 takes significantly fewer steps than CREK1D, LSODE and EPISODE, it requires longer CPU times. This implies that GCKP84 requires much more work per step. However, as shown in reference 9, GCKP84 is an efficient code for performing a wide variety of chemical kinetics calculations. For test problem 2, EPISODE is superior to the other codes. However, in using EPISODE, a word of caution is in order. The computational work can be strongly dependent on the value for the initial steplength (HO) selected by the user. An incorrect guess for HO can make EPISODE prohibitively expensive to use. Table IV illustrates this behavior for test problem 2. Note an order of magnitude increase in the CPU time for a change in HO from 10^{-7} to 10^{-8}. Although not shown here, the solution was also found to be adversely affected by an incorrect choice for HO. In addition, some values of HO resulted in problems with solution instability.

All codes used in the present study automatically select a steplength during the course of the integration. Some of the codes (GCKP84, DASCRU and EPISODE) required a user-supplied initial value to be tried. The other codes automatically selected the value for the initial steplength. The size of the step successfully used by the code indicates both the efficiency of the code and regions where difficulties due to stiffness arise. Figures 5 to 8 present plots of the steplength used by each code through the course of each problem.

Figures 5 and 8 illustrate the small steps that classical methods have to use to insure solution stability. For both test problems, the explicit Runge-Kutta technique uses small steplengths to track the solutions through induction and heat release. During equilibration the steplengths continue to remain small, thus requiring prohibitive amounts of computer time. The difficulties with CHEMEQ (figures 6 and 8) include the selection of a very small initial steplength, the continued use of small steplengths because of the very small increases allowed after satisfactory convergence, and its inability to select a suitable steplength during equilibration. Much computer time is wasted in the search for an appropriate steplength. In addition, this search is restricted to very small values for the steplength. These factors make CHEMEQ very expensive to use.

We note that all codes use small steplengths during induction and early heat release. In these regimes the species and temperature change rapidly (see figs. 1 and 2). Most of the species and temperature have positive time constants indicating that the differential equations are unstable. Hence, the steplengths are constrained to small values.

For test problem 1, CREK1D, GCKP84 and LSODE select steplengths of comparable magnitude, except immediately after ignition ($t \approx 10^{-5}$ s), when GCKP84 selects much larger steplengths (fig. 5). Although EPISODE uses larger steplengths in the post-ignition regime than the other codes, its difficulty in tracking the solution during induction makes it less efficient. The selection of a new steplength after every step results in EPISODE using larger steplengths in the post-ignition regime than the more conservative LSODE. For test problem 2, except at small times when EPSIODE selects larger steplengths, GCKP84 consistently uses larger steplengths than the other codes (fig. 7), thereby requiring far fewer steps. For longer, post-ignition times, the steplengths selected by CREK1D, LSODE and EPISODE are of comparable magnitude. However, at times preceding ignition ($t \approx 3\times10^{-6}$ s), EPISODE selects much larger steplengths than the other codes and is hence more efficient. CREK1D's inefficiency stems from its inability to select a suitable steplength at small times. Much effort is wasted in repeated attempts at selecting a larger steplength. This is reflected by the large number (138) of Jacobian evaluations required by CREK1D. In contrast, EPISODE and LSODE require only about 30 Jacobian evaluations.

The results discussed above indicate that the size of the steplength to be used is regime dependent; during induction and heat release, when the solution changes rapidly, small steplengths have to be taken to insure stability. During equilibration, however, when the solutions are more stable, larger steplengths can be used. These features should be exploited by and incorporated into special-purpose algorithms for the integration of combustion kinetic rate equations.

CONCLUSIONS

A comparison of the efficiency of several algorithms (GCKP84, CREK1D, LSODE, EPISODE, and CHEMEQ) utilized for the numerical integration of stiff ordinary differential equations arising in combustion chemistry has been made. To test these algorithms, two practical problems from combustion kinetics were selected: one involving eleven species and temperature with twelve reactions, and the other involving fifteen species and temperature with thirty reactions. Both problems included all three regimes of combustion: induction, heat release and equilibration.

This study has shown that the fastest package for integrating combustion kinetic rate equations available today is LSODE. This merits special note because LSODE was developed as a multi-purpose stiff differential equation solver, with no one particular application as its objective. EPISODE and CREK1D are attractive alternatives. However, an inaccurate guess for the initial step-length to be tried by the integrator can make EPISODE prohibitively expensive to use. It can also result in incorrect and unstable solutions. Some experimentation with different values for the initial steplength may be necessary to obtain its optimum value. The code CREK1D needs further refinement in the area of steplength selection before significant improvements in its speed can be realized.

An important conclusion from this study is that the use of an algebraic energy conservation equation for calculating the temperature does not result in significant inefficiencies. On the contrary, this method can be more efficient than evaluating the temperature by itegrating its time-derivative.

Nomenclature

$c_{p,i}$	constant pressure specific heat of species i, J/kmol K
h_i	molal-specific enthalpy of species i, J/kmol
h_o	mass-specific enthalpy of mixture, J/kg
n_i	mole number of species i, kmole i/kg mixture
t	time, s
EPS	for all methods, except EPISODE, local relative error tolerance; for EPISODE: relative error tolerance for species with initially non-zero mole numbers and for temperature, and absolute error tolerance for species with initially zero mole numbers
ERMAX	relative error tolerance for Newton Raphson iteration for temperature
HO	initial steplength to be attempted by integrator, s
NFE	total number of functional (i.e., derivative) evaluations
NJE	total number of Jacobian matrix evaluations
NS	number of distinct chemical species involved in the chemical reaction
NSTEP	total number of steps required to solve the problem
T	temperature, K
Y_i	mole fraction of species i

REFERENCES

1. Hindmarsh, A.C.; and Byrne, G.D.: EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations - Including Boundary Value Problems. UCID-30112-REV-1, Lawrence Livermore Laboratory, 1977.
2. Hindmarsh, A.C.: LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers. SIGNUM Newsletter, vol. 15, no. 4, Dec. 1980, pp. 10-11.
3. Hindmarsh, A.C.: ODEPACK, A Systematized Collection of ODE Solvers. UCRL-88007, Lawrence Livermore Laboratory, 1982.
4. Young, T.R.; and Boris, J.P.: A Numerical Technique for Solving Stiff Ordinary Differential Equations Associated with the Chemical Kinetics of Reactive-Flow Problems. J. Phys. Chem., vol. 81, Dec. 15, 1977, pp. 2424-2427.
5. Pratt, D.T.: Fast Algorithms for Combustion Kinetic Calculations. Presented at the International Conference on Stiff Computation, (Park City, UT.), April 12-14, 1982.
6. Pratt, D.T.: CREK-1D: A Computer Code for Transient, Gas-Phase Combustion Kinetics. WSCI Paper 83-21, April 1983.
7. Pratt, D.T.; and Radhakrishnan, K.: CREK-1D: A Computer Code for Transient, Gas-Phase Combustion Kinetics," to appear as a NASA Contractor Report, 1984.
8. Zeleznik, F.J.; and McBride, B.J.: Modelling the Internal Combustion Engine," NASA RP-1094, 1984.
9. Bittker, D.A.; and Scullin, V.J.: GCKP84-General Chemical Kinetics Code for Gas-Phase Flow and Batch Processes Including Heat Transfer," NASA TP, 1984.
10. Pratt, D.T.: New Computational Algorithms for Chemical Kinetics. 10th Materials Research Symposium on Characterization of High Temperature Vapors and Gases. Pt. II. National Bureau of Standards Special Publ. 561, 1979, pp. 1265-1279.
11. Radhakrishnan, K.: A Comparison of Numerical Techniques for the Integration of Stiff ODE's arising in Combustion Chemistry," to appear as a NASA Contractor Report, 1984.

TABLE I. - SUMMARY OF METHODS TESTED

Code or method	Description
GCKP84	Details not yet available.
CREK1D	Variable-step, predictor-corrector method based on an exponentially fitted trapezoidal rule; includes filtering of ill-posed initial conditions and automatic selection of functional iteration or Newton iteration.
LSODE EPISODE	Variable-step, variable-order backward differentiation method with a generalized Newton iteration[a].
CHEMEQ	Variable-step, second-order predictor-corrector method with an asymptotic integration formula for stiff equations.
DASCRU	Variable-step, fourth-order, explicit Runge-Kutta-Merson solver.

[a]Other options are included in these packages.

TABLE II. - COMPARISON OF WORK REQUIRED
FOR TEST PROBLEM 1

Method	EPS	NSTEP	NFE	NJE	CPU(s)
GCKP84	5×10^{-3}	53	170	30	0.846
CREK1D	10^{-2}	84	280	32	.227
LSODE-A	10^{-2}	93	155	26	.357
LSODE-B	10^{-2}	92	144	25	.344
EPISODE-A	10^{-6}	272	506	46	.894
EPISODE-B	10^{-6}	234	441	37	.708
CHEMEQ-A	10^{-2}	7198	14881	0	15.1
CHEMEQ-B	10^{-2}	8041	16589	0	15.5
DASCRU-A	10^{-4}	10700	59365	0	55.5
DASCRU-B	10^{-4}	10718	59760	0	48.7

TABLE III. - COMPARISON OF WORK REQUIRED
FOR TEST PROBLEM 2

Method	EPS	NSTEP	NFE	NJE	CPU(s)
GCKP84	5×10^{-3}	59	171	31	1.73
CREK1D	10^{-3}	140	439	138	1.04
LSODE-A	10^{-2}	98	157	32	.682
LSODE-B	10^{-2}	88	144	27	.617
EPISODE-A	10^{-4}	90	167	31	.584
EPISODE-B	5×10^{-5}	97	209	29	.669
CHEMEQ-A	10^{-2}	9038	18779	0	37.7
CHEMEQ-B	10^{-2}	9139	18990	0	36.3
DASCRU-A	10^{-4}	81457	567490	0	1078
DASCRU-B	10^{-4}	98594	596130	0	1026

TABLE IV. -- EXAMPLE OF EFFECT OF INITIAL STEPLENGTH (H0) ON WORK REQUIRED BY EPISODE-A (EPS = 10^{-5}) FOR TEST PROBLEM 2

H0(s)	NSTEP	NFE	NJE	CPU(s)
10^{-5}	129	237	33	0.786
10^{-6}	129	231	31	.783
10^{-7}	126	225	36	.791
10^{-8}	1168	2355	353	7.91
10^{-9}	1170	2394	362	8.04
10^{-10}	133	231	32	.772

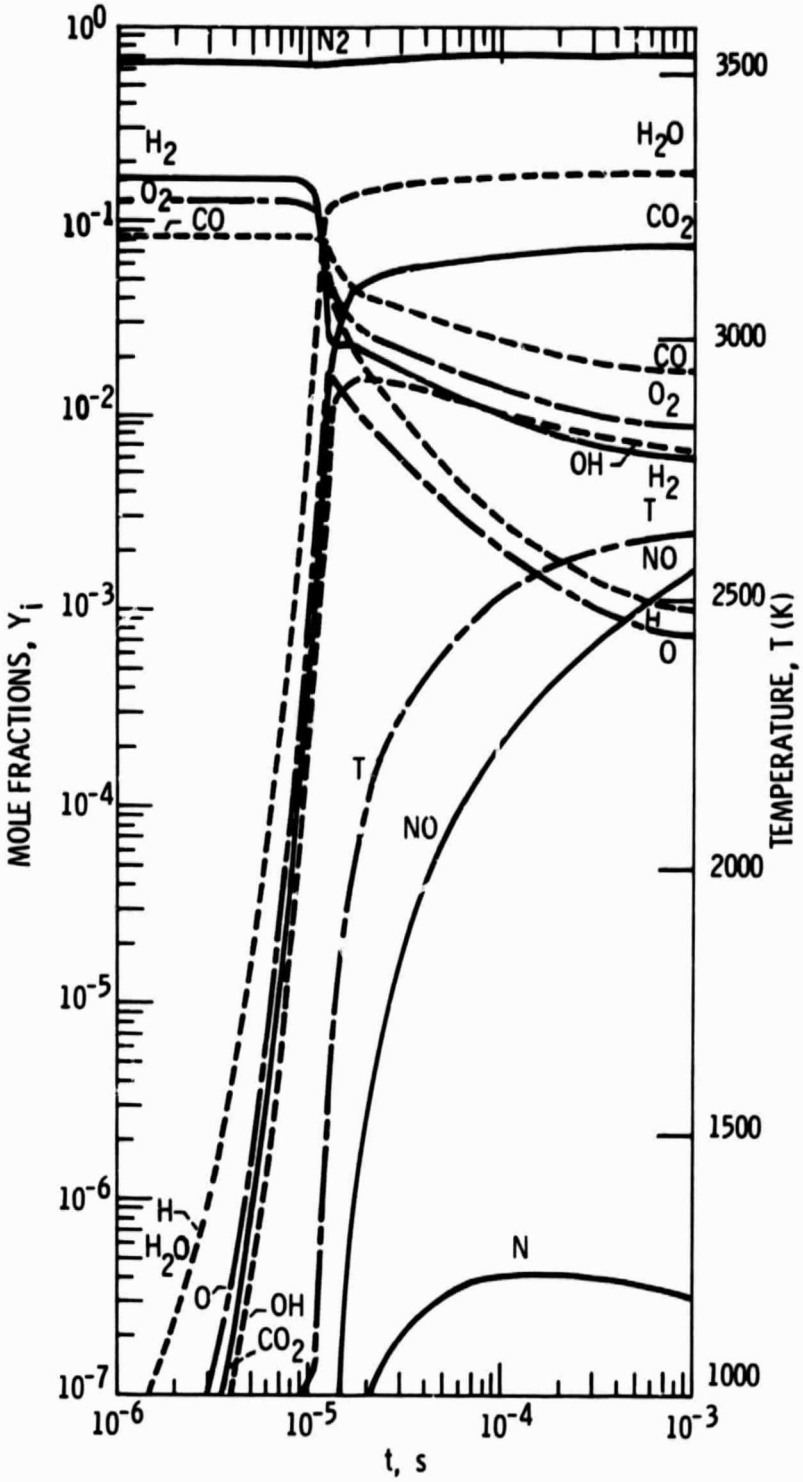

Figure 1. - Variation with time of temperature and species mole fractions for test problem 1. Solution generated with LSODE-B and EPS = 10^{-5}.

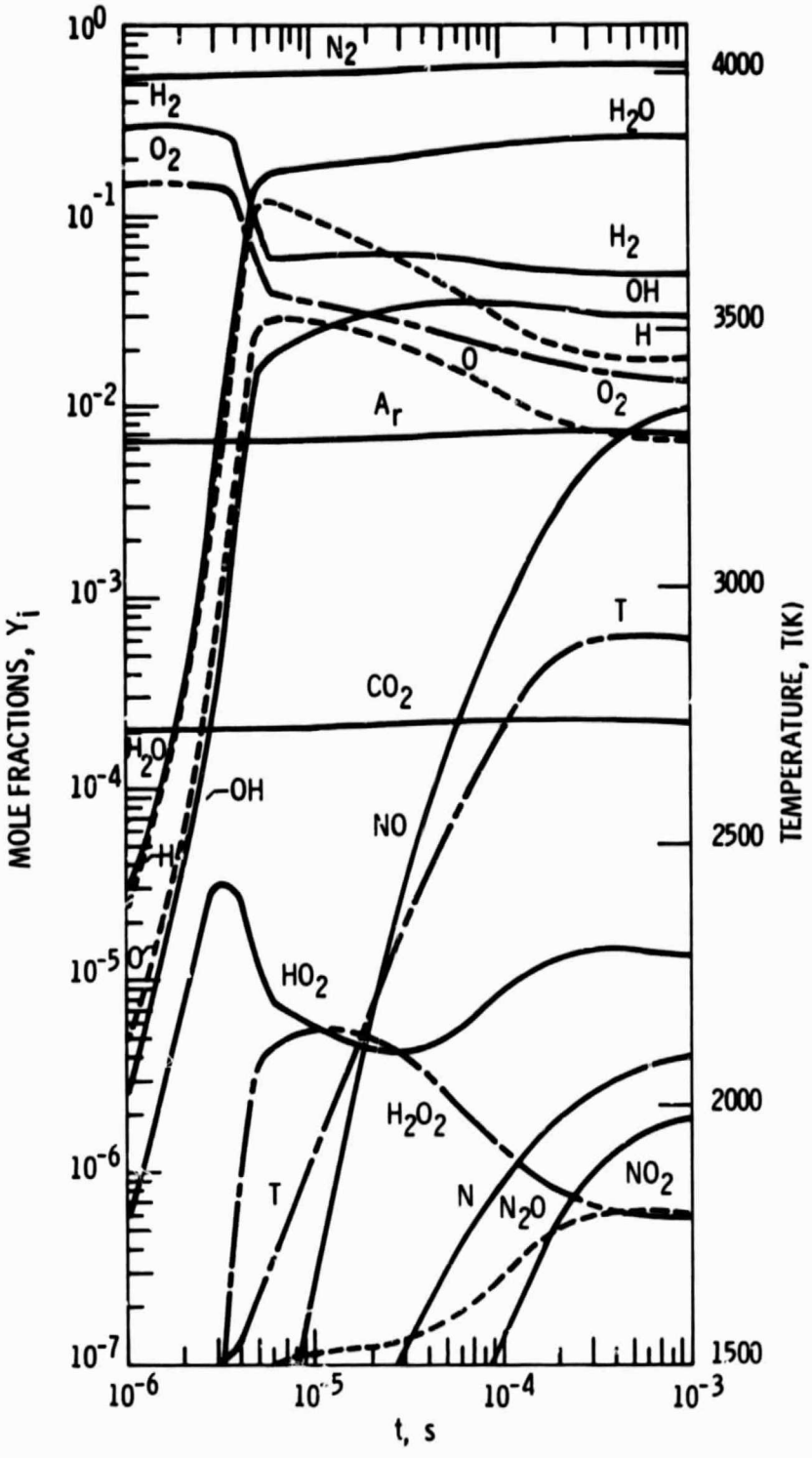

Figure 2. - Variation with time of temperature and species mole fractions for test problem 2. Solution generated with LSODE-B and EPS = 10^{-5}.

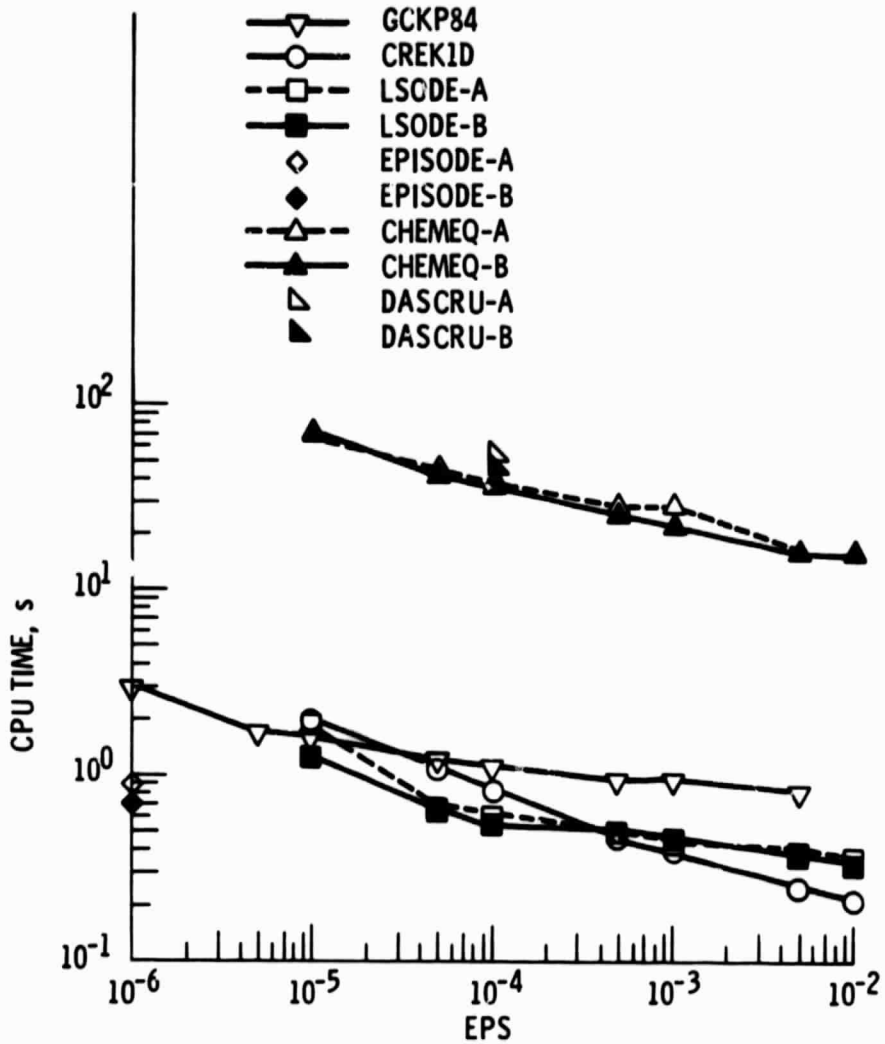

Figure 3. - Variation of the CPU time (s) with error tolerance EPS, for test problem 1. All runs on IBM 370/3033.

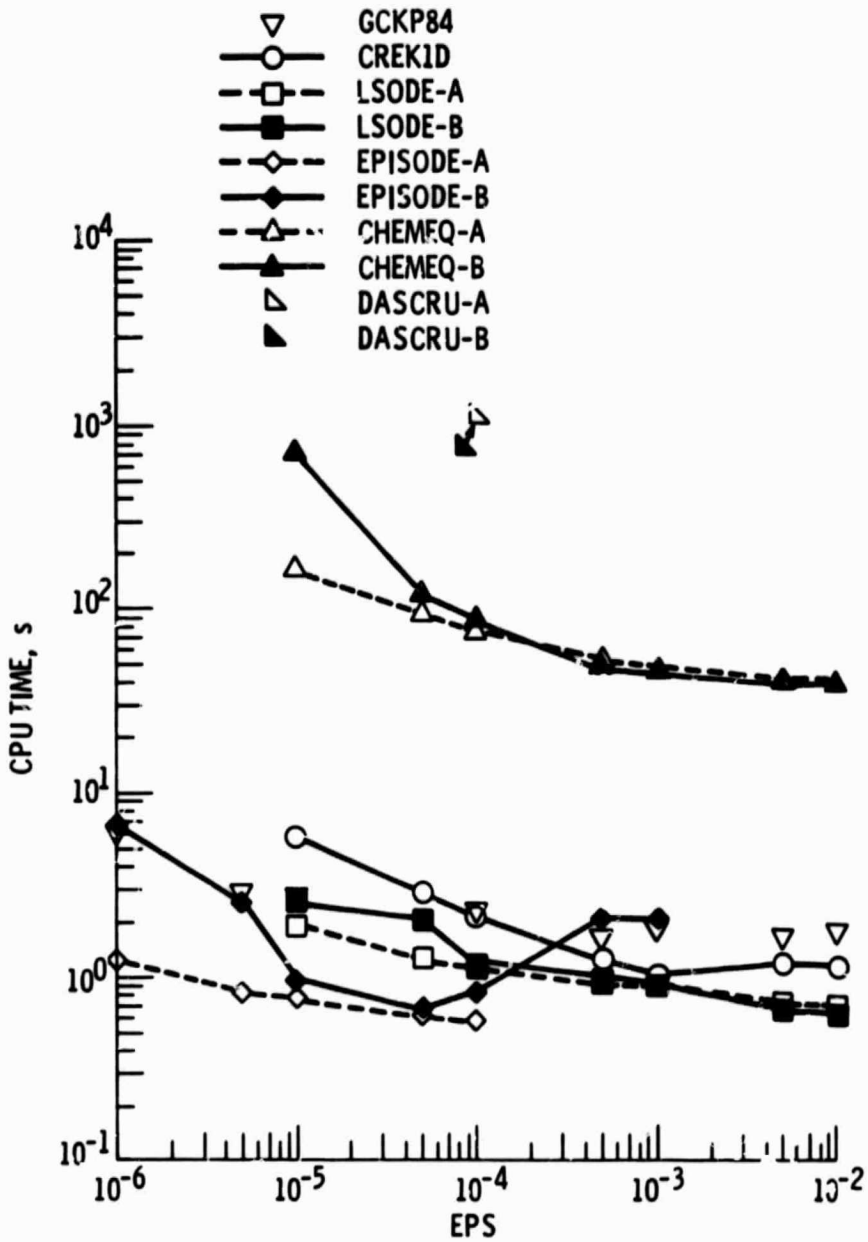

Figure 4. - Variation of the CPU time (s) with error tolerance, EPS, for test problem 2. All runs on IBM 370/3033.

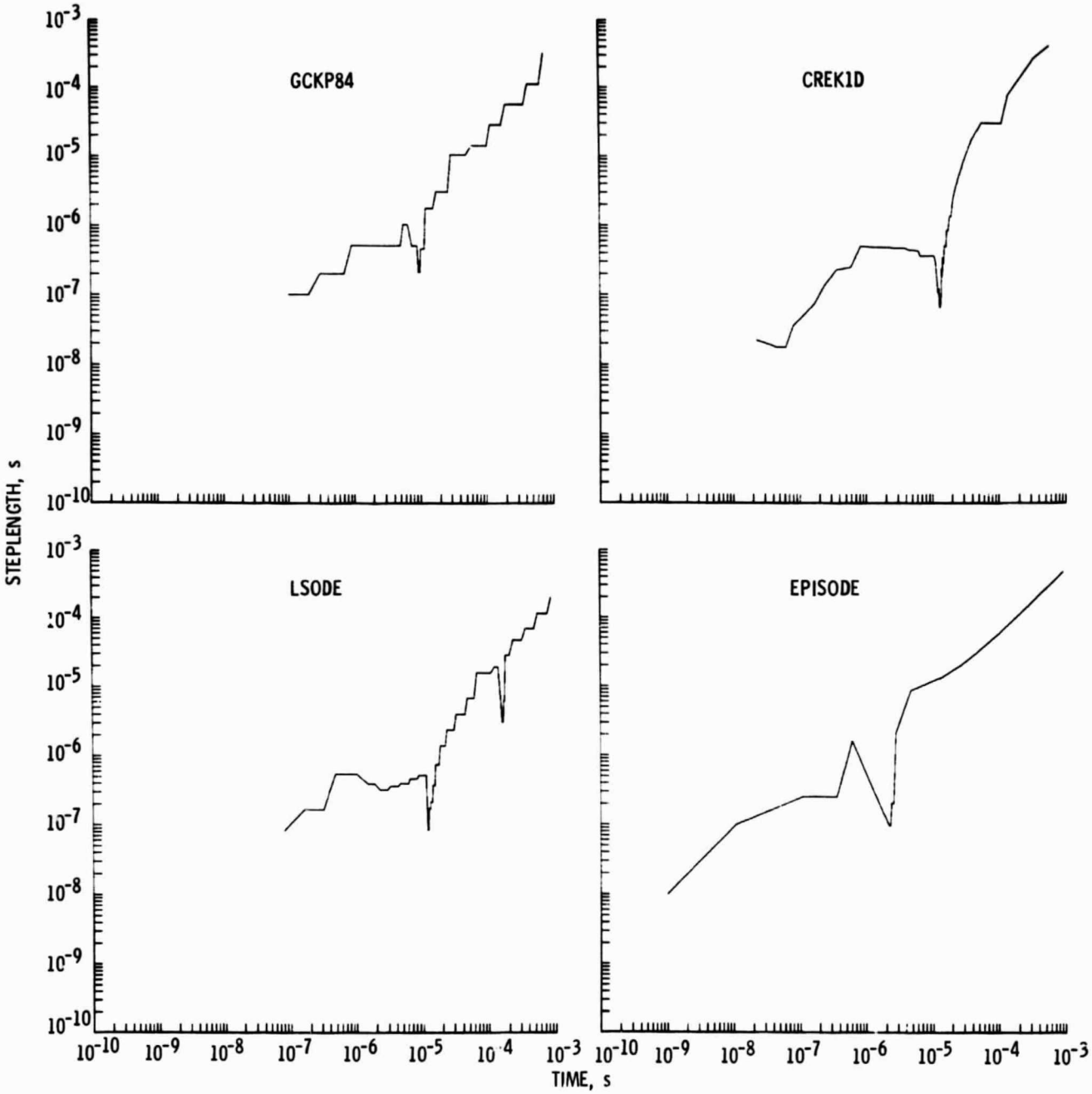

Figure 5. - Variation with time (s) of the steplength (s) successfully used by GCKP84, CREK1D, LSODE-B, and EPISODE-B for test problem 1.

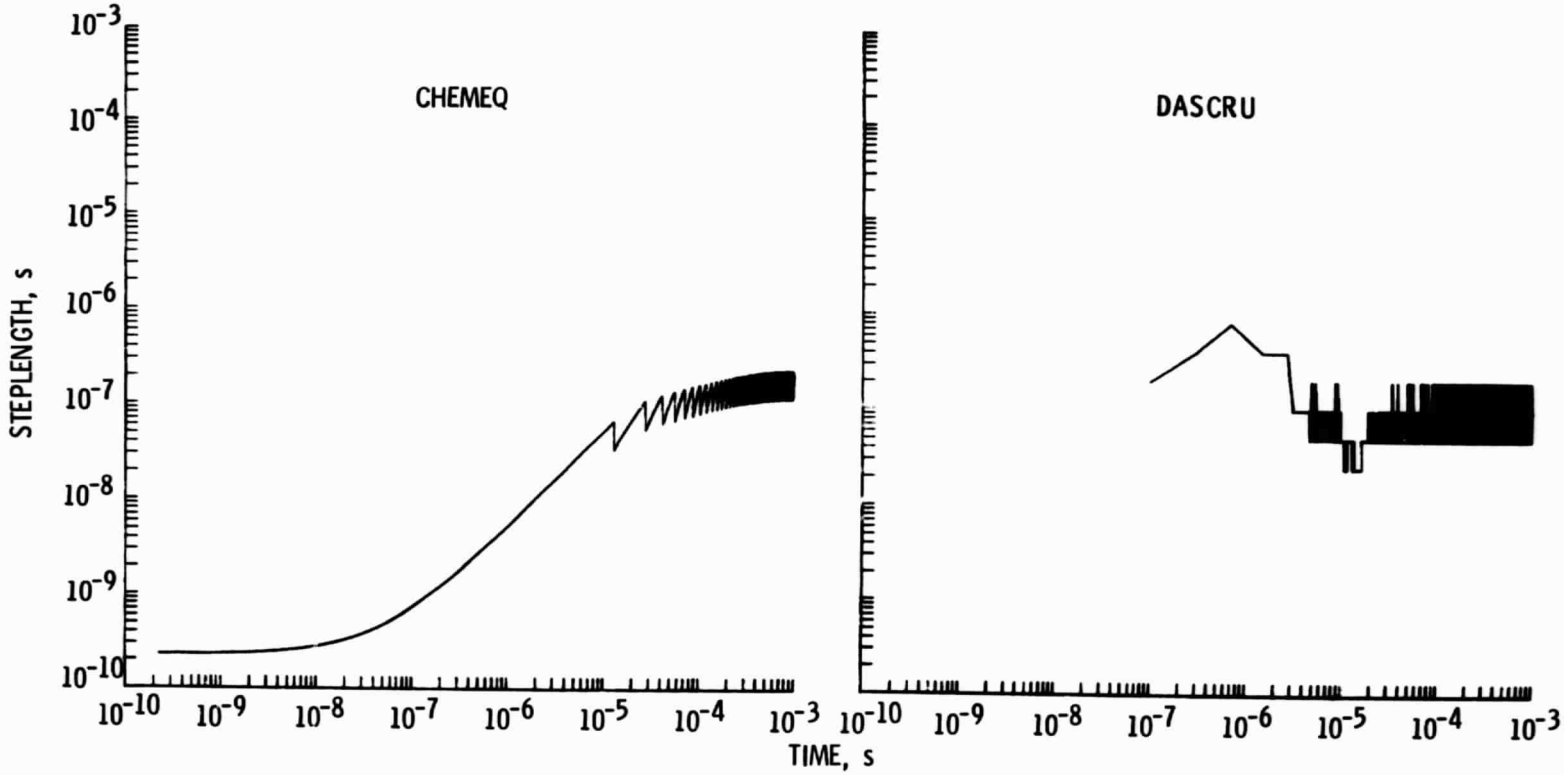

Figure 6. - Variation with time (s) of the steplength (s) successfully used by CHEMEQ-B and DASCRU-B for test problem 1.

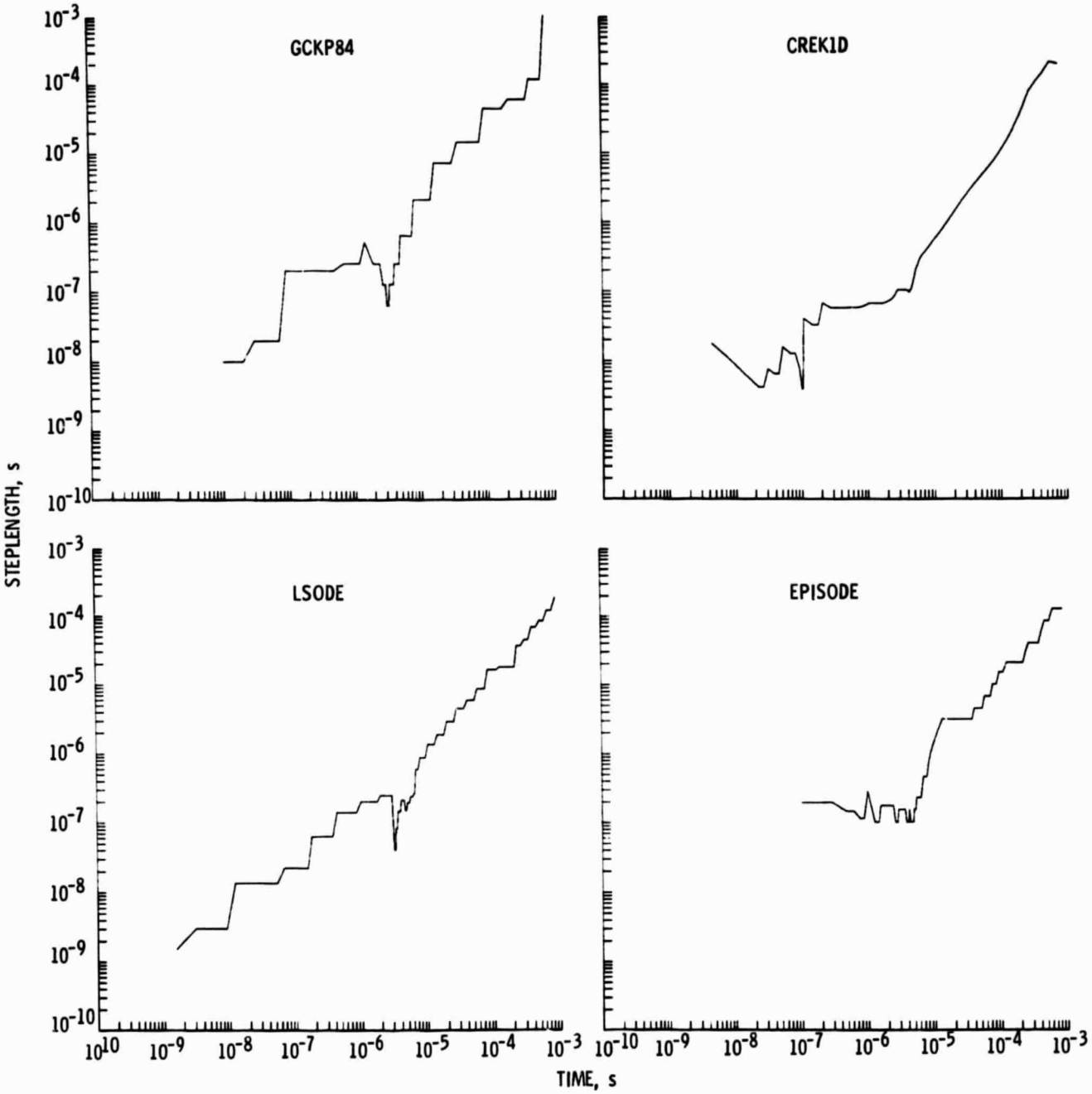

Figure 7. - Variation with time (s) of the steplength (s) successfully used by GCKP84, CREK1D, LSODE-A, and EPISODE-A for test problem 2.

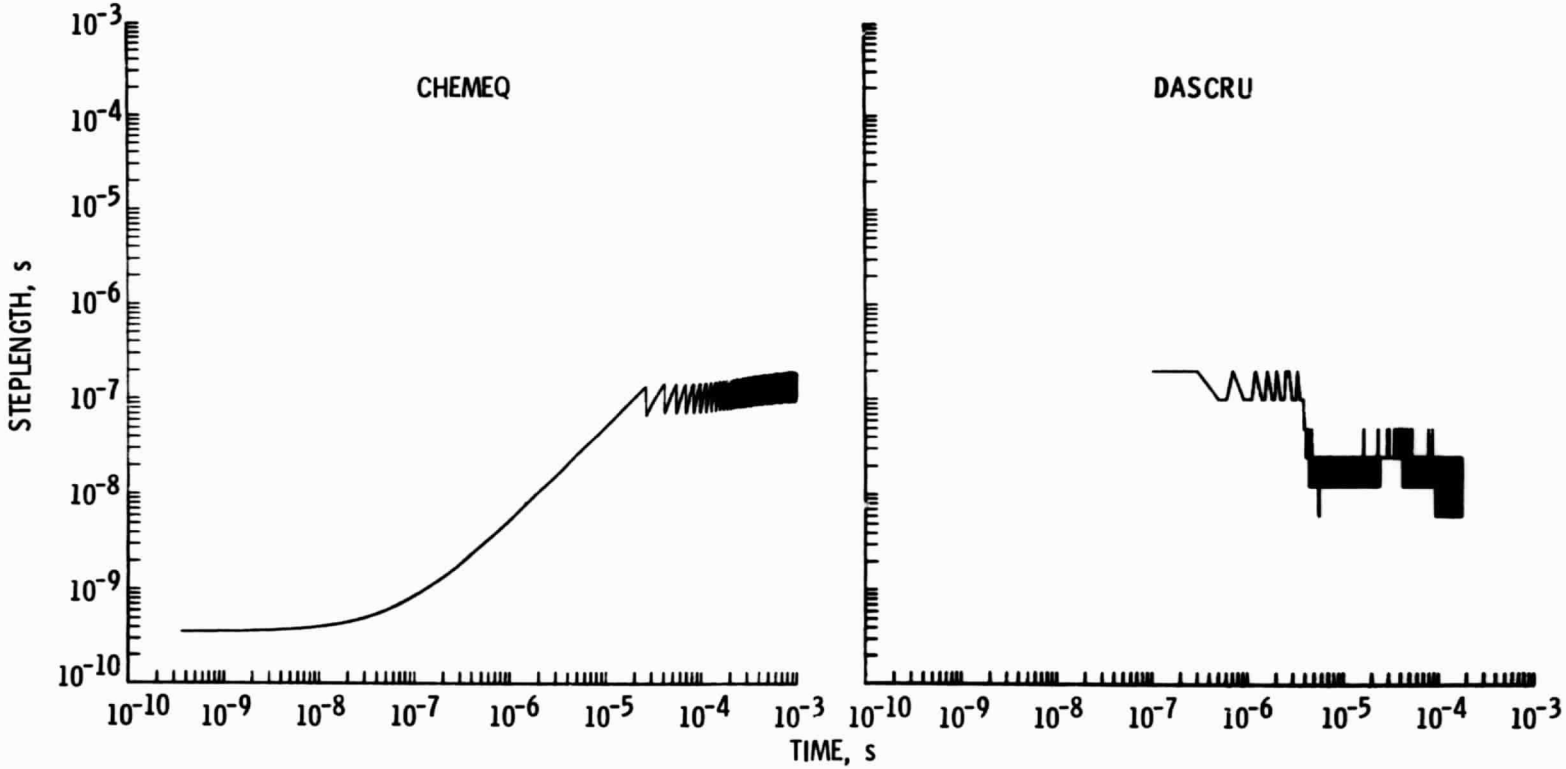

Figure 8. - Variation with time (s) of the steplength (s) successfully used by CHEMEQ-A and DASCRU-A for test problem 2.