NASA Technical Paper 2372

October 1984

Comparison of Numerical Techniques for Integration of Stiff Ordinary Differential Equations Arising in Combustion Chemistry

Krishnan Radhakrishnan

NASA

NASA Technical Paper 2372

1984

Comparison of Numerical Techniques for Integration of Stiff Ordinary Differential Equations Arising in Combustion Chemistry

Krishnan Radhakrishnan

Lewis Research Center
Cleveland, Ohio

National Aeronautics and Space Administration

Scientific and Technical Information Branch

Summary

The efficiency and accuracy of several algorithms recently developed for the efficient numerical integration of stiff ordinary differential equations are compared. 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 does not result in significant errors. In addition, this method can be more efficient than evaluating the temperature by integrating its time derivative.

Significant reductions in computational work are realized by updating the rate constants ($k = AT^N \exp(-E/RT)$) only when the temperature change exceeds an amount ΔT that is problem dependent. An approximate expression for the automatic evaluation of ΔT is derived and is shown to result in increased efficiency.

Introduction

The major problem associated with the simultaneous numerical integration of large sets of chemical kinetic rate equations is that of stiffness. Although stiffness does not have a simple definition (see, e.g., Shampine, refs. 1 and 2), it is characterized by widely varying time constants. For example, in hydrogen-air combustion the induction time is of the order of microseconds whereas the nitric oxide formation time is of the order of milliseconds. These widely different time constants present classical methods (such as the popular explicit Runge-Kutta method) with the following difficulty: to ensure stability of the numerical solution, these methods are restricted to using very short time steps that are determined by the smallest time constant. However, the time for all chemical species to reach near-equilibrium values is determined by the largest time constant. As a result, classical methods require excessive amounts of computer time to solve stiff systems of ordinary differential equations (ODE's).

Several approaches for the solution of stiff ODE's have been proposed; for details, see the reviews by Lomax and Bailey (ref. 3), Seinfeld, et al. (ref. 4), Enright and Hull (ref. 5), and Shampine and Gear (ref. 6). Of all these techniques the general-purpose codes EPISODE and LSODE (refs. 7 to 10) are regarded as the best available "packaged" codes for the solution of stiff systems of ODE's. However, although these codes may be the best available for solving an arbitrary system of ODE's, it may be possible to construct a superior method for solving a particular system of ODE's governing the behavior of a specific problem. In this vein, Young and Boris (ref. 11), Pratt (refs. 12 to 15), and Zeleznik and McBride (ref. 16) have developed codes for the specific purpose of integrating large systems of chemical kinetic rate equations.

The objective of the present investigation is to identify the fastest algorithm currently available for the numerical integration of combustion kinetic rate equations.[1] The motivation behind this work is the increasing interest in (1) modeling the reaction mechanisms describing the consumption of fuels and pollutant formation and destruction and (2) multidimensional modeling of reactive flows, which includes the equations of fluid motion. The former results in the need to integrate large systems of nonlinear ODE's (reaction rates). The latter results in the need to integrate these rate equations at several thousand grid points. To make such calculations practicable, it is necessary to have a very fast homogeneous batch chemistry integrator.

In the present report, currently available techniques are examined by application to two test problems drawn from combustion kinetics. A detailed comparison of the efficiency and accuracy of these techniques is presented, and recommendations are made for ways to increase the speed of general-purpose codes, as applied to the present problem.

[1] In this report, attention is restricted to adiabatic, constant pressure (hence, isenthalpic), exothermic chemical reactions.

Symbols

A_j, A_{-j} preexponential constants in forward and reverse rate equations for reaction j (eq. (5)), units depend on reaction type

B_j, B_{-j} exponent-on-ten in modified Arrhenius preexponential factor, $B_j = \log_{10} A_j$, $B_{-j} = \log_{10} A_{-j}$, arbitrary units

CPU total CPU time required on IBM 370/3033, s

$c_{p,i}$ constant-pressure specific heat of species i, J/kmole K

E_i estimated local error in independent variable i (eq. (17))

E_j, E_{-j} activation energy in forward and reverse rate equations for reaction j (eq. (5)), cal/mole

EPS for all methods, except EPISODE and DASCRU: local relative error tolerance; for EPISODE: local relative error tolerance for species with initially nonzero mole numbers and for the temperature, and local absolute error tolerance for species with initially zero mole numbers; for DASCRU: local relative error tolerance for a variable whose magnitude is greater than or equal to 10^{-3}, and local absolute error tolerance for a variable whose magnitude is less than 10^{-3}

ERMAX relative error tolerance for Newton-Raphson iteration for temperature

f_i rate of formation of species i (eq. (2)), kmole i/kg mixture s

g_i^0 1-atmosphere molal-specific Gibbs function of species i, J/kmole

H_0 initial mass-specific enthalpy of mixture, J/kg

H0 initial step length to be attempted by integrator, s

h_i molal-specific enthalpy of species i, J/kmole

IERROR error control indicator for EPISODE

ITMAX maximum number of corrector iterations to be attempted by CHEMEQ and CREK1D

ITOL error control indicator for LSODE

J Jacobian matrix: for temperature method A of size NS×NS: $J_{ik} = \partial f_i / \partial n_k$, $i,k = 1,\text{NS}$ (eq. (11)); for temperature method B, of size (NS+1)×(NS+1): $J_{ik} = \partial f_i / \partial n_k$, $i,k = 1,\text{NS}$ (eq. (11)); $J_{i,\text{NS}+1} = \partial f_i / \partial T$, $i = 1,\text{NS}$ (eq. (13)); $J_{\text{NS}+1,k} = \partial / \partial n_k \; (dT/dt)$, $k = 1,\text{NS}$ (eq. (14)); $J_{\text{NS}+1,\text{NS}+1} = \partial / \partial T \; (dT/dt)$ (eq. (15))

JJ total number of distinct elementary reactions in reaction mechanism

k_j, k_{-j} forward and reverse rate constants for reaction j (eq. (5)), units depend on reaction type

MF integration method to be used by EPISODE and LSODE

N_j, N_{-j} temperature exponent in forward and reverse rate constants for reaction j (eq. (5))

NFE total number of functional (i.e., derivative) evaluations

NJE total number of Jacobian matrix evaluations

NRRC total number of reaction rate constant evaluations

NS total number of distinct chemical species in gas mixture

NSTEP total number of steps required to solve problem

n_i mole number of species i, kmole species i/kg mixture

n_j', n_j'' molecularities of forward and reverse reactions j (eq. (12))

n_m reciprocal of mixture mean molar mass (eq. (10)), kmole/kg

Δn_j difference in molecularities of reverse and forward reactions j (eq. (6c))

p pressure, N/m^2

R universal gas constant, 8314.3 J/kmole K (1.9872 cal/mole K)

R_j, R_{-j} molar forward and reverse rates per unit volume for reaction j, kmole/m^3 s

T temperature, K

T_j, T_{-j} activation temperatures in forward and reverse rate constants for reaction j, $T_j = E_j/R$; $T_{-j} = E_{-j}/R$, K

ΔT maximum temperature change allowed before reaction rate constants are updated. For CREK1D, CHEMEQ-B, and DASCRU-B thermodynamic data are also updated only for temperature changes greater than ΔT, K

t time, s

t_{switch} time at which error control performed by EPISODE is switched from semirelative to pure relative, s

Y_i mole fraction of species i

ϵ_{abs} local absolute error tolerance (eq. (18))

$\epsilon_{\text{abs},i}$ local absolute error tolerance for species i (eq. (19))

ϵ_{rel} local relative error tolerance (eq. (17))

ν'_{ij}, ν''_{ij} stoichiometric coefficients for species i in forward and reverse reactions j (eq. (1)); number of kilomoles i in elementary reaction j as a reactant and as a product, respectively

ρ mixture mass density, kg/m^3

Problem Statement

The (initial value) problem may be stated as follows: given (1) a set of initial conditions (n_i ($i=1$,NS), where n_i is the mole number of species i (kmole species i/kg mixture) and NS is the total number of distinct chemical species involved in the combustion reaction and the temperature T (K)) at time $t=0$, (2) the pressure p, and (3) the reaction mechanism, find the mixture composition and temperature at the end of a prescribed time interval.

All chemical reactions are assumed to be elementary reactions of the type

$$\sum_{i=1}^{NS} \nu'_{ij} X_i \rightleftarrows \sum_{i=1}^{NS} \nu''_{ij} X_i \qquad j=1,JJ \tag{1}$$

where ν'_{ij} and ν''_{ij} are the stoichiometric coefficients of species i (with chemical formula X_i) in reaction j as a reactant and as a product, respectively, and JJ is the total number of distinct elementary reactions in the given reaction mechanism.

The ordinary differential equations describing adiabatic, homogeneous, gas-phase combustion reactions are as follows:

$$\left.\begin{aligned} &\frac{dn_i}{dt} = f_i(n_k, T) \qquad i,k=1,NS \\ &n_i(t=0) = \text{given} \\ &T\,(t=0) = \text{given} \end{aligned}\right\} \tag{2}$$

where

$$f_i = -\rho^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} - \nu''_{ij})(R_j - R_{-j}) \tag{3}$$

where the molar reaction rates per unit volume R_j and R_{-j} are given by

$$\begin{aligned} R_j &= A_j T^{N_j} \exp\left(\frac{-E_j}{RT}\right) \prod_{k=1}^{NS} (\rho n_k)^{\nu'_{kj}} \\ &= A_j T^{N_j} \exp\left(\frac{-T_j}{T}\right) \prod_{k=1}^{NS} (\rho n_k)^{\nu'_{kj}} \\ &= k_j \prod_{k=1}^{NS} (\rho n_k)^{\nu'_{kj}} \end{aligned} \tag{4a}$$

and

$$\begin{aligned} R_{-j} &= A_{-j} T^{N_{-j}} \exp\left(\frac{-E_{-j}}{RT}\right) \prod_{k=1}^{NS} (\rho n_k)^{\nu''_{kj}} \\ &= A_{-j} T^{N_{-j}} \exp\left(\frac{-T_{-j}}{T}\right) \prod_{k=1}^{NS} (\rho n_k)^{\nu''_{kj}} \\ &= k_{-j} \prod_{k=1}^{NS} (\rho n_k)^{\nu''_{kj}} \end{aligned} \tag{4b}$$

In equations (2) to (4), ρ is the mixture density (kg/m^3) and A_j, A_{-j}, N_j, N_{-j}, E_j, T_j ($=E_j/R$) E_{-j}, and T_{-j} ($=E_{-j}/R$) are constants in the modified Arrhenius rate expressions of the forward R_j and reverse R_{-j} rates of reaction j. The forward k_j and reverse k_{-j} rate constants for reaction j are given by

$$k_j = A_j T^{N_j} \exp\left(\frac{-T_j}{T}\right) \tag{5a}$$

$$k_{-j} = A_{-j} T^{N_{-j}} \exp\left(\frac{-T_{-j}}{T}\right) \tag{5b}$$

In this study the reverse rate constants are calculated from the forward rate constants and the concentration equilibrium constant by using the principle of detailed balancing (ref. 17). The resulting equations are

$$T_{-j} = T_j + \frac{\sum_{i=1}^{NS} (\nu'_{ij} - \nu''_{ij}) h_i}{R} \tag{6a}$$

$$k_{-j} = k_j (RT)^{\Delta n_j} \exp\left[\frac{\sum_{i=1}^{NS} (\nu''_{ij} - \nu'_{ij}) g_i^0}{RT}\right] \tag{6b}$$

where h_i is the molal-specific enthalpy of species i (J/kmole), g_i^0 is the 1-atmosphere molal-specific Gibbs function of species i (J/kmole) and Δn_j is given by

$$\Delta n_j = \sum_{i=1}^{NS} (\nu''_{ij} - \nu'_{ij}) \tag{6c}$$

Equality of the temperature exponents in equations (5b) and (6b) for k_{-j} together with equation (5a) results in the following relation for N_{-j}:

$$N_{-j} = N_j + \Delta n_j \tag{6d}$$

For a constant-pressure, adiabatic combustion reaction, the following enthalpy conservation equation constitutes an algebraic constraint on equations (2) to (4):

$$\sum_{i=1}^{NS} n_i h_i = H_0 = \text{constant} \tag{7}$$

where H_0 is the mass-specific enthalpy of the mixture (J/kg).

Time differentiation of equation (7) provides the following equation for the time rate of change of temperature:

$$\frac{dT}{dt} = \frac{-\sum_{i=1}^{NS} f_i h_i}{\sum_{i=1}^{NS} n_i c_{p,i}} \tag{8}$$

where $c_{p,i}$ is the constant-pressure specific heat of species i (J/kmole K).

The density ρ of the mixture is given by the equation of state for an ideal gas mixture

$$\rho = \frac{p}{RTn_m} \tag{9}$$

where p is the absolute pressure (N/m^2), R the universal gas constant (J/kmole K), and the reciprocal of the mean molar mass of the mixture

$$n_m = \sum_{i=1}^{NS} n_i \tag{10}$$

Some of the techniques used in the present study require the evaluation of the Jacobian matrix J ($J_{ik} = \partial f_i / \partial n_k$; $i,k = 1,\text{NS}$). Differentiation of equation (3) with respect to n_k gives the following expression for J_{ik}:

$$J_{ik} = -(\rho n_k)^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} - \nu''_{ij})(\nu'_{kj} R_j - \nu''_{kj} R_{-j}) + \frac{f_i}{n_m} + (\rho n_m)^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} - \nu''_{ij})(n'_j R_j - n''_j R_{-j}) \tag{11}$$

where n'_j and n''_j are, respectively, the molecularities of the forward and reverse reaction j and are given by

$$\left.\begin{aligned} n'_j &= \sum_{i=1}^{NS} \nu'_{ij} \\ n''_j &= \sum_{i=1}^{NS} \nu''_{ij} \end{aligned}\right\} \tag{12}$$

When temperature is included as an explicit independent variable (see the section Evaluation of Temperature for more details), the Jacobian elements $\partial f_i / \partial T$ ($i = 1,\text{NS}$), $\partial / \partial n_k$ (dT/dt) ($k = 1,\text{NS}$), and $\partial / \partial \text{T}(dT/dt)$ are also needed. These are obtained by differentiating equation (3) with respect to T and equation (8) with respect to n_k and T. These operations yield

$$\frac{\partial f_i}{\partial T} = \frac{f_i}{T} - \frac{1}{\rho T} \sum_{j=1}^{JJ} (\nu'_{ij} - \nu''_{ij}) \left[R_j \left(N_j + \frac{T_j}{T} - n'_j \right) - R_{-j} \left(N_{-j} + \frac{T_{-j}}{T} - n''_j \right) \right] \tag{13}$$

$$\frac{\partial}{\partial n_k} \left(\frac{dT}{dt} \right) = - \frac{\sum_{i=1}^{NS} \frac{\partial f_i}{\partial n_k} h_i + c_{p,k} \left(\frac{dT}{dt} \right)}{\sum_{i=1}^{NS} n_i c_{p,i}} \tag{14}$$

$$\frac{\partial}{\partial T} \left(\frac{dT}{dt} \right) = - \frac{\sum_{i=1}^{NS} \frac{\partial f_i}{\partial T} h_i + \sum_{i=1}^{NS} f_i c_{p,i} + \left(\frac{dT}{dt} \right) \sum_{i=1}^{NS} n_i \frac{dc_{p,i}}{dt}}{\sum_{i=1}^{NS} n_i c_{p,i}} \tag{15}$$

Methods Studied

The methods examined in this study include the general-purpose packages EPISODE and LSODE (refs. 7 to 10), developed for an arbitrary system of ODE's, and the specialized codes CHEMEQ (ref. 11), CREK1D (ref. 15), and GCKP84 (refs. 16 and 18), developed specifically to integrate chemical kinetic rate equations. In addition, an explicit Runge-Kutta-Merson differential equation solver (ref. 19) (IMSL routine DASCRU) is used to illustrate the difficulty associated with integrating chemical kinetic rate equations by classical methods. These methods are summarized in table I and discussed in more detail in appendix A.

The packages EPISODE and LSODE, based on the methods of Gear (refs. 20 and 21), consist of a variable-step, variable-order implicit Adams method (suitable for nonstiff problems) and a variable-step, variable-order backward differentiation method (suitable for stiff problems). A range of corrector iteration methods—from functional iteration to chord method (a variant of Newton's method) with a banded Jacobian generated internally—is included in these packages. The user selects both the basic method and the corrector iteration method by means of a method flag MF.

In CHEMEQ the ODE's are separated into two classes, stiff and normal, at the beginning of each time step. A classical second-order predictor-corrector method is used for equations classified as normal. For stiff equations a simple asymptotic integration formula is used.

CREK1D is based on the exponentially fitted trapezoidal rule developed by Liniger and Willoughby (ref. 22) and by Brandon (refs. 23 and 24). The algorithm includes special treatment of ill-posed initial conditions and automatic selection of functional iteration or Newton iteration.

GCKP84, developed by Bittker and Scullins (ref. 18), is a new general chemical kinetics program that supersedes their previous GCKP codes (ref. 25). The new code uses the integration technique developed by Zeleznik and McBride (ref. 16) specifically to integrate chemical kinetic rate equations. Details of this integration technique are not yet available.

DASCRU is an explicit fourth-order Runge-Kutta-Merson ODE solver. This method requires five derivative evaluations per step. The additional derivative evaluation provides an estimate of the local truncation error (ref. 19).

Test Problems

The algorithms summarized in the preceding section were applied to two test problems drawn from combustion kinetics. Both problems include all three regions of interest to a combustion researcher: induction, heat release, and equilibration.

Test problem 1, taken from Pratt (ref. 12), 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 10 atmospheres and an initial temperature of 1000 K. It comprises 12 reactions (table II) that describe the time evolution of 11 species. Test problem 2, taken from Bittker and Scullins (ref. 18), describes the ignition and subsequent combustion of a stoichiometric mixture of hydrogen and air, at a pressure of 2 atmospheres and an initial temperature of 1500 K. It involves 30 reactions (table III) that describe the time evolution of 15 species.

The initial values for the species mole fractions and temperature are given in tables IV and V for test problems 1 and 2, respectively. Also given in these tables are the equilibrium values for the species mole fractions and temperature calculated by using a Gibbs function minimization routine (ref. 26). Both test problems were

TABLE I.—SUMMARY OF METHODS STUDIED

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.—REATION MECHANISMS AND RATE CONSTANTS USED IN TEST PROBLEM 1

Reaction	Rate constants[a]		
	B	*N*	*E*, kcal/mole
$CO+OH=CO_2+H$	11.49	0	0.596
$H+O_2=O+OH$	14.34	↓	16.492
$H_2+O=H+OH$	13.48	↓	9.339
$H_2O+O=OH+OH$	13.92	↓	18.121
$H+H_2O=H_2+OH$	14.0	↓	19.870
$N+O_2=NO+O$	9.81	1.0	6.250
$N_2+O=N+NO$	13.85	0	75.506
$NO+M=N+O+M$	20.60	−1.5	149.025
$H+H+M=H_2+M$	18.00	−1.0	0
$O+O+M=O_2+M$	18.14	−1.0	.34
$H+OH+M=H_2O+M$	23.88	−2.6	0
$H_2+O_2=OH+OH$	13.00	0	43.0

[a]Rate constant $k=10^{B}T^{N}\exp(-E/RT)$.

TABLE III.—REACTION MECHANISMS AND RATE CONSTANTS USED FOR TEST PROBLEM 2

Reaction	Rate constants[a]		
	B	*N*	*E*, kcal/mole
$H+O_2=OH+O$	14.342	0	16.790
$O+H_2=OH+H$	10.255	1.0	8.900
$H_2+OH=H_2O+H$	13.716	0	6.500
$OH+OH=O+H_2O$	12.799	↓	1.093
$H+O_2+M=HO_2+M$	15.176	↓	−1.000
$O+O+M=O_2+M$	13.756	↓	−1.788
$H+H+M=H_2+M$	17.919	−1.0	0
$H+OH+M=H_2O+M$	21.924	−2.0	0
$H_2+HO_2=H_2O+OH$	11.857	0	18.700
$H_2O_2+M=OH+OH+M$	17.068	↓	45.500
$H_2+O_2=OH+OH$	13.000	↓	43.000
$H+HO_2=OH+OH$	14.398	↓	1.900
$O+HO_2=OH+O_2$	13.699	↓	1.000
$OH+HO_2=H_2O+O_2$	13.699	↓	1.000
$HO_2+HO_2=H_2O_2+O_2$	12.255	↓	0
$OH+H_2O_2=H_2O+HO_2$	13.000	↓	1.800
$O+H_2O_2=OH+HO_2$	13.903	↓	1.000
$H+H_2O_2=H_2O+OH$	14.505	↓	9.000
$HO_2+NO=NO_2+OH$	13.079	↓	2.380
$O+NO_2=NO+O_2$	13.000	↓	.596
$NO+O+M=NO_2+M$	15.750	↓	−1.160
$NO_2+H=NO+OH$	14.462	↓	.795
$N+O_2=NO+O$	9.806	1.0	6.250
$O+N_2=NO+N$	14.255	0	76.250
$N+OH=NO+H$	13.602	↓	0
$N_2O+M=N_2+O+M$	14.152	↓	51.280
$O+N_2O=N_2+O_2$	13.794	↓	24.520
$O+N_2O=NO+NO$	13.491	↓	21.800
$N+NO_2=NO+NO$	12.556	↓	0
$OH+N_2=N_2O+H$	12.505	↓	80.280

[a]Rate constant $k=10^{B}T^{N}\exp(-E/RT)$.

integrated over a time of 1 ms to obtain near equilibration of all species.

Discussion of Results

In this section, we discuss the computational work (which we shall take as a measure of the efficiency of the algorithm) and the accuracy of the techniques selected for our study. All of the codes 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.

Computational Procedure

A typical computational run consisted of initializing the time (t, set equal to 0), species mole numbers, temperature, and the CPU time.[2] The integrator was then called with values for the necessary input parameters, which are discussed in the section Efficiency Comparisons, and the elapsed time (1 ms for both problems) at which the solution is to be terminated. The integrator returns to the main calling program with the computed solutions for the mole numbers and, for some methods, the temperature. The integrator also returns with values for the following parameters, which give a measure of the computational work required to solve the problem: total number of steps NSTEP, total number of functional (i.e., derivative) evaluations NFE, total number of Jacobian matrix evaluations (NJE = 0 for CHEMEQ and DASCRU), and, for reasons presented later, total number of rate constant evaluations NRRC. On return from the integrator the computer time CPU required to solve the problem was calculated.

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 by using one of two different methods, labeled as methods A and B.[3]

[2]Before this was done, various preprocessors were called to read in thermochemical and reaction rate data and to compute the initial mixture properties and the equilibrium composition and temperature. This does not affect the work required by the integrator. The storage and work requirements of these preprocessors are therefore not included in this study.

[3]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.).

TABLE IV.—COMPOSITIONS AND TEMPERATURES FOR TEST PROBLEM 1

[Solution generated with LSODE-B and EPS = 10^{-5}.]

Component and temperature	Time, t, s							
	0	9×10^{-6}	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}	∞
Composition, species mole fraction								
CO	8.319×10^{-2}	8.311×10^{-2}	8.292×10^{-2}	3.123×10^{-2}	2.562×10^{-2}	1.802×10^{-2}	1.754×10^{-2}	1.798×10^{-2}
CO_2	0	8.359×10^{-5}	2.886×10^{-4}	6.078×10^{-2}	6.715×10^{-2}	7.557×10^{-2}	7.610×10^{-2}	7.564×10^{-2}
H	0	1.400×10^{-3}	4.572×10^{-3}	5.958×10^{-3}	3.131×10^{-3}	1.177×10^{-3}	1.084×10^{-3}	1.059×10^{-3}
H_2	1.664×10^{-1}	1.641×10^{-1}	1.588×10^{-1}	1.373×10^{-2}	1.030×10^{-2}	6.559×10^{-3}	6.357×10^{-3}	6.580×10^{-3}
H_2O	0	1.519×10^{-3}	5.221×10^{-3}	1.608×10^{-1}	1.682×10^{-1}	1.762×10^{-1}	1.767×10^{-1}	1.768×10^{-1}
N	0	2.419×10^{-19}	2.419×10^{-19}	3.258×10^{-7}	4.117×10^{-7}	3.778×10^{-7}	3.364×10^{-7}	2.209×10^{-7}
NO	0	5.501×10^{-16}	1.725×10^{-13}	6.295×10^{-5}	1.993×10^{-4}	9.975×10^{-4}	1.666×10^{-3}	5.347×10^{-3}
N_2	6.256×10^{-1}	6.256×10^{-1}	6.257×10^{-1}	6.919×10^{-1}	6.975×10^{-1}	7.033×10^{-1}	7.034×10^{-1}	7.014×10^{-1}
O	0	1.699×10^{-4}	5.678×10^{-4}	3.803×10^{-3}	2.173×10^{-3}	8.779×10^{-4}	7.986×10^{-4}	7.048×10^{-4}
OH	0	4.569×10^{-5}	1.508×10^{-4}	1.301×10^{-2}	1.082×10^{-2}	7.552×10^{-3}	7.238×10^{-3}	6.762×10^{-3}
O_2	1.248×10^{-1}	1.239×10^{-1}	1.217×10^{-1}	1.878×10^{-2}	1.484×10^{-2}	9.752×10^{-3}	9.171×10^{-3}	7.826×10^{-3}
Temperature, T, K								
T_A	1000	1001	1006	2407	2512	2623	2629	2619
T_B	----	1001	1006	2407	2512	2623	2628	----

TABLE V.—COMPOSITIONS AND TEMPERATURES FOR TEST PROBLEM 2

[Solution generated for LSODE-B and EPS = 10^{-5}.]

Component and temperature	Time, t, s								
	0	3×10^{-6}	5×10^{-6}	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}	∞
Composition, species mole fraction									
Ar	6.571×10^{-3}	6.572×10^{-3}	6.605×10^{-3}	6.744×10^{-3}	7.080×10^{-3}	7.204×10^{-3}	7.323×10^{-3}	7.324×10^{-3}	7.326×10^{-3}
CO_2	2.110×10^{-4}	2.111×10^{-4}	2.121×10^{-4}	2.166×10^{-4}	2.274×10^{-4}	2.314×10^{-4}	2.352×10^{-4}	2.352×10^{-4}	2.353×10^{-4}
H	0	3.006×10^{-3}	1.073×10^{-1}	9.848×10^{-2}	4.542×10^{-2}	3.063×10^{-2}	1.870×10^{-2}	1.854×10^{-2}	1.853×10^{-2}
HO_2	0	3.261×10^{-5}	1.183×10^{-5}	5.878×10^{-6}	5.121×10^{-6}	8.674×10^{-6}	1.447×10^{-5}	1.398×10^{-5}	1.391×10^{-5}
H_2	2.951×10^{-1}	2.896×10^{-1}	9.272×10^{-2}	6.201×10^{-2}	6.063×10^{-2}	5.544×10^{-2}	4.924×10^{-2}	4.966×10^{-2}	4.974×10^{-2}
H_2O	0	3.807×10^{-3}	1.429×10^{-1}	1.793×10^{-1}	2.171×10^{-1}	2.353×10^{-1}	2.549×10^{-1}	2.549×10^{-1}	2.549×10^{-1}
H_2O_2	0	3.442×10^{-8}	3.790×10^{-6}	5.603×10^{-6}	2.629×10^{-6}	1.380×10^{-6}	6.377×10^{-7}	6.185×10^{-7}	6.160×10^{-7}
N	0	1.718×10^{-12}	2.039×10^{-10}	4.927×10^{-9}	3.084×10^{-7}	9.124×10^{-7}	3.544×10^{-6}	4.067×10^{-6}	4.144×10^{-6}
NO	0	4.894×10^{-12}	3.769×10^{-9}	2.925×10^{-7}	1.428×10^{-4}	8.095×10^{-4}	7.615×10^{-3}	9.752×10^{-3}	1.008×10^{-2}
NO_2	0	1.639×10^{-15}	2.112×10^{-19}	3.520×10^{-11}	1.928×10^{-8}	1.374×10^{-7}	1.568×10^{-6}	1.973×10^{-6}	2.035×10^{-6}
N_2	5.501×10^{-1}	5.502×10^{-1}	5.530×10^{-1}	5.646×10^{-1}	5.926×10^{-1}	6.027×10^{-1}	6.094×10^{-1}	6.084×10^{-1}	6.082×10^{-1}
N_2O	0	1.090×10^{-9}	6.095×10^{-8}	1.149×10^{-7}	1.704×10^{-7}	3.010×10^{-7}	6.382×10^{-7}	6.350×10^{-7}	6.345×10^{-7}
O	0	6.325×10^{-4}	2.533×10^{-2}	2.841×10^{-2}	1.690×10^{-2}	1.204×10^{-2}	7.113×10^{-3}	6.866×10^{-3}	6.833×10^{-3}
OH	0	3.358×10^{-4}	1.452×10^{-2}	2.440×10^{-2}	3.495×10^{-2}	3.485×10^{-2}	3.068×10^{-2}	3.011×10^{-2}	3.003×10^{-2}
O_2	1.480×10^{-1}	1.456×10^{-1}	5.738×10^{-2}	3.583×10^{-2}	2.495×10^{-2}	2.078×10^{-2}	1.480×10^{-2}	1.417×10^{-2}	1.408×10^{-2}
Temperature, T, K									
T_A	1500	1503	1634	1909	2512	2724	2911	2909	2908
T_B	----	1503	1634	1909	2512	2723	2911	2909	----

In method A the temperature was calculated from the mole numbers and the initial mixture enthalpy by using the algebraic enthalpy equation (eq. (7)). A Newton-Raphson iteration technique (with a user-specified relative error tolerance ERMAX) was used to solve equation (7) for the temperature. In this method the temperature does not enter into the problem as an explicit independent variable, so the number of independent ODE's is equal to the number of species NS and the Jacobian matrix is of size NS by NS. The integrator therefore tracks only the solution for the mole numbers. As shown later, this did not introduce significant errors because the species concentrations changed faster than the temperature (figs. 1 and 2) and the step length was determined by the rate at which the variables changed. Since the integrator did not compute a solution for the temperature, it was evaluated from the solution for the mole numbers returned by the integrator. In addition, the temperature was computed whenever the species time derivatives (dn_i/dt) and the Jacobian matrix ($J_{ik} = \partial f_i/\partial n_k$; $i,k = 1,\mathrm{NS}$) were evaluated.

In method B the temperature was treated as an additional independent variable and evaluated by integrating its time derivative (eq. (8)). This increased the number of independent ODE's to NS + 1, and the computation of the Jacobian matrix (of size (NS + 1) by (NS + 1)) involved the calculation of 2NS + 1 additional terms. In this method, the integrator tracks the solutions for both the temperature and the species mole numbers.

Accuracy Estimate

To compare the accuracy of the algorithms, standard solutions must be established since exact solutions for the test cases are not known. The solutions used as standards were those generated by using LSODE-B (which is LSODE using method B for computing the temperature) because LSODE and its predecessors have been

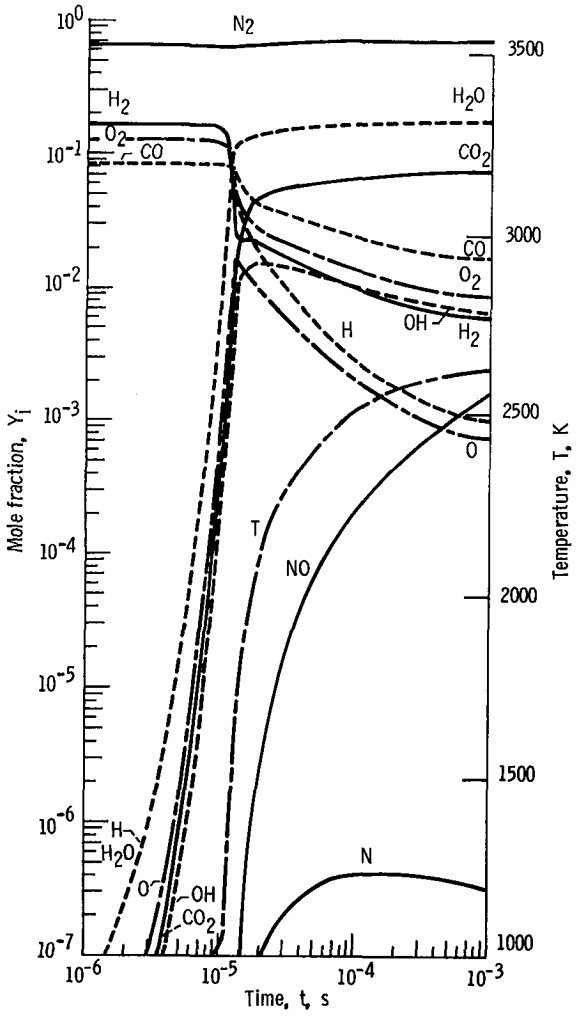


Figure 1.—Variation with time of species mole fractions and temperature for test problem 1. Solution generated with LSODE-B and EPS = 10^{-5}.

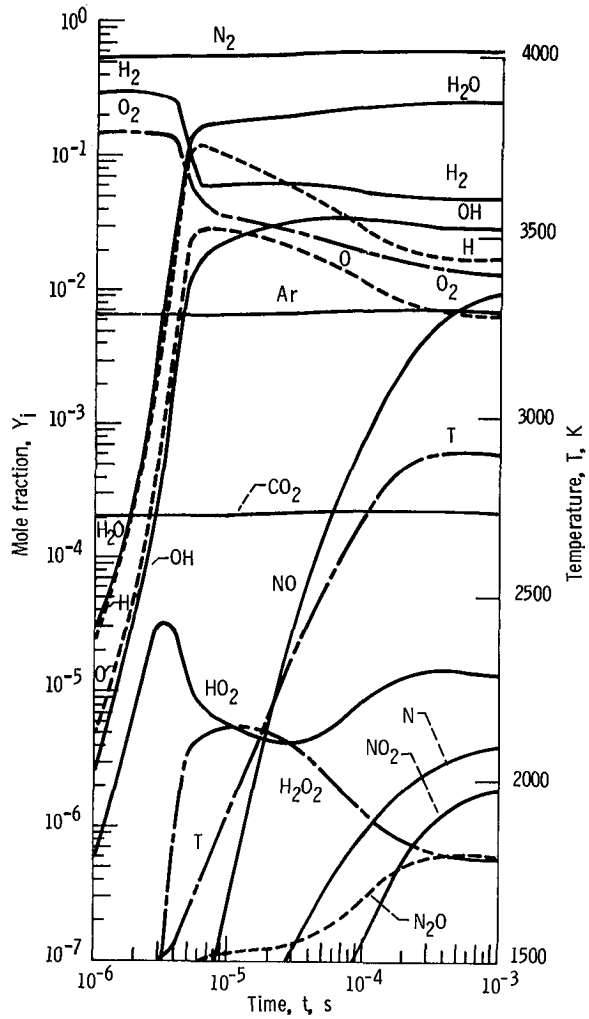


Figure 2.—Variation with time of species mole fractions and temperature for test problem 2. Solution generated with LSODE-B and EPS = 10^{-5}.

extensively tested on a wide variety of problems (refs. 7, 8, 27 to 29). A low relative error tolerance (EPS = 10^{-5}) was used, and the method flag MF was set equal to 21 (stiff method, user-supplied analytic evaluation of the complete Jacobian matrix). As discussed in appendix B, LSODE requires the specification of values for both the local relative and absolute error tolerances. In the test problems examined in this study the species mole numbers and temperature differed widely (tables IV and V), so relative error control was appropriate. Pure relative (or absolute) error control can be realized by using a value of zero for the absolute (or relative) error tolerance. However, since some of the mole numbers had zero initial values, a truly relative error criterion could not be used. To make the local error control mostly relative, low values for the absolute tolerance for species mole numbers and zero for temperature were used. Values for the absolute error tolerances for the species mole numbers were obtained by progressively decreasing them until the temperature-time trace showed essentially no change with a further decrease. The values used for the absolute error tolerances for the species mole numbers were 10^{-14} and 10^{-11} for test problems 1 and 2, respectively.

To determine if the use of solutions generated with LSODE as standards biases the test in favor of LSODE, we explore the use of other standards in the section Efficiency Comparisons and show that these do not give different results.

The standard solutions (for the mole fractions and temperature) are shown in figures 1 and 2 for test problems 1 and 2, respectively. The values for the mole fractions and temperatures at various times are shown in tables IV and V for problems 1 and 2, respectively. The third column in these tables presents immediate postinduction (i.e., end of the essentially isothermal reaction period) values, and the remaining columns show development of postinduction profiles. The last column shows the equilibrium solution obtained by using a Gibbs function minimization routine (ref. 26).

Although these standard solutions were generated by using temperature method B, method A was also used during these computations to evaluate the temperature, using only the solution for the mole numbers returned by the integrator. This was done to see if any consistent differences could be observed between values generated by using method A (T_A in tables IV and V) and those generated by using method B (T_B in tables IV and V). The consistently excellent agreement between T_A and T_B indicates that either method A or B would suffice for computing the temperature.

Storage Requirement and Startup Time

Multidimensional models for reactive flows require the integration of large sets of reaction rate equations at several thousand grid points for relatively short times between fluid mechanic time steps (ref. 11). These models generally also have large storage requirements. Hence reaction rate integration techniques with both a low storage requirement and a low initialization (startup) time are needed.

The storage requirements (for 20 species and 36 reactions) for the single-precision versions of CREK1D, LSODE, EPISODE, CHEMEQ, and DASCRU are given in table VI. The storage requirements for LSODE, EPISODE, and DASCRU are for versions that included both methods A and B for calculating the temperature. The temperature method to be used was specified via a temperature-method flag. Also, for LSODE the given figure does not include the storage required by subroutines (included in this package) that are not essential for its execution with method flag MF = 21. The storage shown in table VI for GCKP84 is that required by the double-precision version. We note that the special-purpose codes CHEMEQ and CREK1D required much less storage than the general-purpose codes EPISODE and LSODE. GCKP84 required more storage than the other codes because of the precision used and also because of the various options built into it. Although GCKP84 is a special-purpose code in that it has been developed for chemical kinetics problems, it can be used to solve a wider variety of problems than CHEMEQ and CREK1D (ref. 18).

The CPU time required by each code to successfully complete the first step was taken as a measure of the startup time. Each code was run with a low value for the elapsed time to ensure that only one step was taken to complete the problem. Note, however, that because of the automatic filtering of initial conditions (appendix A), CREK1D took two steps to return with a solution. The CPU times (in milliseconds) required for the first step (and the first two steps for CREK1D) are given in table VII. The general-purpose codes required longer initialization times than the special-purpose codes. GCKP84 required a much longer startup time than the other codes for the same reasons that its storage requirement is greater and also because of the extra work

TABLE VI.—STORAGE REQUIREMENTS

Method	Storage size, bytes
GKCP84[a]	205 948
CREK1D	26 304
LSODE	47 152
EPISODE	35 116
CHEMEQ-A	10 832
CHEMEQ-B	10 552
DASCRU	10 928

[a]Double-precision version.

TABLE VII.—CENTRAL PROCESSING UNIT TIMES REQUIRED TO SUCCESSFULLY COMPLETE FIRST STEP

Method	Problem 1	Problem 2
	CPU time, CPU ms,	
GCKP84	32	44
CREK1D	[a]9	[a]17
LSODE-A	20	24
LSODE-B	19	24
EPISODE-A	14	18
EPISODE-B	13	18
CHEMEQ-A	6	10
CHEMEQ-B	6	8
DASCRU-A	7	11
DASCRU-B	6	10

[a]Time required for first two steps.

involved in handling species whose mole numbers have zero initial values (see ref. 18 for details).

Efficiency Comparisons

The procedure described in the section Computational Procedure was used to study the computational work required by the different techniques examined in this study. For LSODE, EPISODE, CHEMEQ, and DASCRU both temperature methods A and B were attempted.

The codes examined in the present study require the specification of several parameters in addition to the local error tolerance EPS required of the solution. Values for these parameters that minimized the computational work required by each code were obtained by using a trial-and-error process. Following are the user-supplied parameters (excluding the local error tolerance EPS and the elapsed time at which the integration is to be terminated) required by each code (see appendix B for a detailed discussion)—for EPISODE: the method flag MF, the error control to be performed IERROR, and the initial step length H0 to be used; for LSODE: the method flag MF, the error control to be performed ITOL, and the values for the absolute error tolerances for the independent variables; for CHEMEQ: the maximum number of corrector iterations ITMAX allowed before nonconvergence is declared; for CREK1D: the maximum number of corrector iterations ITMAX allowed before nonconvergence is declared and the maximum temperature change ΔT allowed before thermodynamic properties and reaction rate constants are updated; for GCKP84: since details for this technique are not yet available, default values for all parameters to be used; for DASCRU: a guess for the initial step length H0 to be used.

The following selection procedure was adopted in using these techniques: each code was run with a value of 10^{-2} for the error tolerance EPS, and a solution (with output at various times) was generated. The computed temperature-time profile was then compared with the standard solution and was accepted if within 50 kelvins. Otherwise, it was rejected and a lower value for EPS was tried. This ensured that computational work was compared with codes of comparable accuracy. All of the results presented herein, except EPISODE and GCKP84 for test problem 1, satisfied this criterion. In other words, for these codes no EPS $\geq 10^{-6}$ resulted in acceptable agreement. With DASCRU the temperature did not satisfy this criterion during early heat release for test problem 1.

For test problem 2, some runs with DASCRU and LSODE predicted zero mole numbers for the nitrogen dioxide at times when the standard solutions had risen to measurable levels (of the order of 0.1 ppm). All codes were therefore required to satisfy the following additional criterion: a run was accepted only if it predicted nonzero mole fractions for all species whose standard solution values were greater than or equal to 10^{-7}. All results presented herein satisfied this criterion.

Each code was run with the maximum (and lower) values of EPS that resulted in acceptable agreement with the standard solution. The computational work was obtained by following the procedure outlined earlier. 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 local error tolerance EPS for test problems 1 and 2, respectively. For all codes except EPISODE and DASCRU, EPS is the local relative error tolerance required of the numerical solution. For EPISODE, EPS is a mixed relative and absolute error tolerance—relative for species with initially nonzero mole numbers and for temperature (method B) and absolute for species with initially zero mole numbers[4]. For DASCRU, EPS is absolute for variables whose magnitude is less than 10^{-3} and relative otherwise. For this study, because the maximum permissible local relative error in the temperature calculated by using method B was EPS, ERMAX (the relative error allowed in the Newton-Raphson iteration procedure used in

[4]Because some mole numbers had zero initial values, pure relative error control (option IERROR = 2) could not be used with EPISODE. The option IERROR = 3 was used, instead. This is a semirelative error control where, for a variable that is initially nonzero, the error control is relative. For a variable that is initially zero, the error control is absolute until the variable reaches 1 in magnitude, when the control becomes relative. Since none of the mole numbers reaches a value of unity, the error control is always absolute for species with initially zero mole numbers. In CHEMEQ and CREK1D, mole numbers less than 10^{-20} were set equal to 10^{-20}. In addition, in CREK1D the convergence test was applied only to species whose mole numbers are greater than 10^{-20} (appendix A).

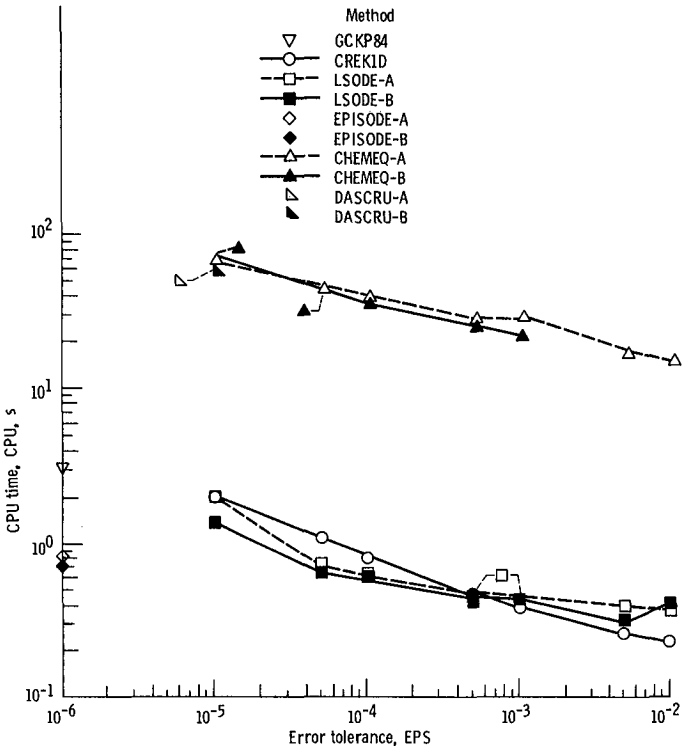


Figure 3.—Variation of the CPU time with error tolerance for test problem 1. All runs on IBM 370/3033.

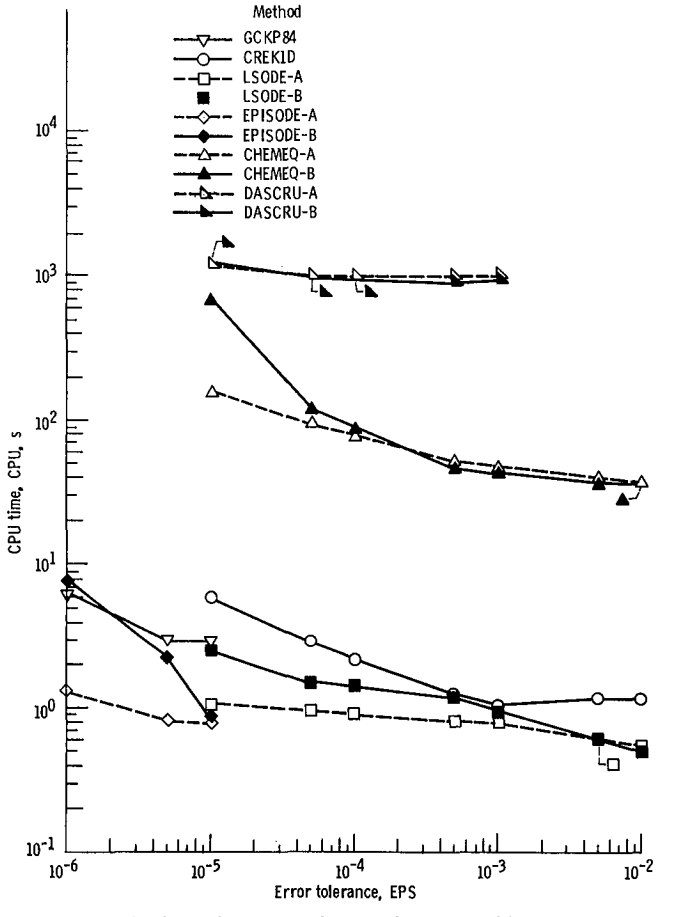


Figure 4.—Variation of the CPU time with error tolerance for test problem 2. All runs on IBM 370/3033.

method A to solve the algebraic energy equation) was set equal to EPS to make the two methods comparable. For the same reason, with LSODE-B the absolute error tolerance for the temperature was set equal to zero.

For test problem 1, very small values for the error tolerance had to be used for GCKP84, EPISODE, and DASCRU (fig. 3). Even with EPS = 10^{-6}, EPISODE-A, EPISODE-B, and GCKP84 did not accurately track the temperature during ignition and heat release, when the solution changed rapidly (fig. 1). For EPS $\geq 5\times10^{-6}$, EPISODE predicted physically meaningless results—little or no change from initial values after an elapsed time of 1 ms. However, with GCKP84 and DASCRU, although higher values of EPS resulted in poor solutions during ignition and heat release, the predicted solutions during equilibration were satisfactory. For example, with GCKP84 and DASCRU, values of EPS as high as 5×10^{-3} and 10^{-3}, respectively, were adequate to track the temperature in this regime with an error of less than 1 percent. The run with GCKP84 and EPS = 10^{-2} exhibited serious instability and so was terminated.

Similar remarks apply to test problem 2 where, again, small values of EPS had to be used with EPISODE-A, EPISODE-B, and GCKP84 to track correctly the temperature during ignition and heat release. During equilibration, higher values for EPS could be used without incurring error penalties. With EPISODE and GCKP84, values of EPS equal to 10^{-4} and 10^{-2}, respectively, were adequate to follow the temperature within 1 percent in this regime, although these runs did not satisfy the accuracy criterion during ignition and heat release. EPISODE-A and EPISODE-B predicted little or no change in the composition and temperature after an elapsed time of 1 ms for EPS greater than or equal to 5×10^{-4} and 5×10^{-3}, respectively. Although the runs with EPISODE-B and EPS of 5×10^{-4} and 10^{-3} were successfully completed, the solutions were considerably inaccurate. Also, these runs were less efficient than the run with EPS = 5×10^{-5}. For a more detailed study on the variation of the CPU time with EPS, see references 30 and 31.

The selection procedure used above required that all integrators track the solutions generated with LSODE. This accuracy criterion is subject to the following two objections: (1) it biases the test in favor of LSODE at the expense of the other integrators, and (2) there is no guarantee that LSODE is more accurate than the other codes examined in this study. The other integrators were therefore being required to track not the true solutions (which are not known) but solutions that could be less accurate than what they themselves generated. In other words, the use of solutions generated with another integrator might have resulted in the acceptance of some runs that were rejected above. This possibility was explored by further testing.

For codes[5] that failed to satisfy the comparison with LSODE, new standards were established. For every one of these codes the solutions used as the new standards were those generated by the code itself using a low value for EPS. All runs that had previously been rejected were then compared with these new standards, which were all found to be essentially the same as the standard solutions generated with LSODE. More specifically, the following comparisons were made: for test problem 1 the runs with GCKP84 and EPS $\geq 5\times10^{-6}$ were compared with the solution generated with GCKP84 and EPS = 10^{-8}, and the runs with CHEMEQ-B and EPS $\geq 5\times10^{-3}$ were compared with the solution generated with CHEMEQ-B and EPS = 10^{-5}. For test problem 2 the runs with GCKP84 and EPS $\geq 5\times10^{-5}$ were compared with the solution generated with GCKP84 and EPS = 10^{-8}, and the runs with EPISODE-A and EPISODE-B with EPS $\geq 5\times10^{-5}$ were compared with solutions generated with EPISODE-A and EPISODE-B, respectively, and EPS = 10^{-6}. In making these new comparisons the selection criterion remained the same: namely, accept the run if the predicted temperature-time profile is within 50 kelvins of the new standard. These additional comparisons did not result in the acceptance of any run that had previously been rejected. Furthermore, for test problem 1 the run with GCKP84 and EPS = 10^{-6} failed the 50 kelvin requirement when compared with GCKP84 and EPS = 10^{-8}. These additional comparisons not only support the use of solutions generated with LSODE as standards, but also imply that LSODE, CREK1D, and CHEMEQ-A are more accurate than the other codes.

Figures 3 and 4 illustrate the difficulty associated with using a classical method (in this case the explicit Runge-Kutta method used in DASCRU) to integrate combustion kinetic rate equations. The CPU times required for the two test problems were of the order of 1 and 15 min, respectively. Using this technique would make multidimensional modeling of practical combustion devices prohibitively expensive.

For test problem 1 the difference in computational work required by methods A and B was small (fig. 3), with method B being more efficient. For test problem 2 (fig. 4) the difference was small for large values of EPS. But for small values of EPS the difference was more marked, with method A being significantly superior to method B. The temperature-time profile was steeper for test problem 1 (figs. 1 and 2), indicating a stronger

[5]These include GCKP84 and CHEMEQ-B for test problem 1 and GCKP84, EPISODE-A, and EPISODE-B for test problem 2. Although some runs with DASCRU-A and DASCRU-B also failed to satisfy the comparisons with LSODE, the new tests were not applied to DASCRU. The main purpose in using DASCRU was to illustrate the difficulty associated with using classical methods to integrate combustion kinetic rate equations.

coupling between the species and the temperature. This may explain why the inclusion of the temperature as an additional independent variable worked well for test problem 1. But for test problem 2 the additional work in computing the temperature rate and the temperature-dependent terms in the Jacobian matrix did not lead to increased efficiency.

LSODE and CREK1D were superior to the other codes examined in this study (figs. 3 and 4). Comparing EPISODE and GCKP84 with CREK1D and LSODE was, however, difficult for the reasons discussed below. The error control performed by EPISODE is different from that performed by the other codes. As shown later, for problems of the type examined in this study, the error control used in EPISODE is inferior to that used in LSODE. To attain comparable accuracy at early times, EPISODE must use much lower values of EPS than LSODE, a result seen earlier. Nevertheless, when interest was restricted to computational efficiency, EPISODE was an attractive alternative to LSODE and CREK1D, especially for test problem 2 (see ref. 30 for details). In using EPISODE, however, a word of caution is in order. The computational work can depend strongly on the value for the initial step length H0 chosen by the user. A poor guess for H0 can make EPISODE prohibitively expensive to use, as shown for test problem 2 in table VIII. Note an order-of-magnitude increase in the CPU time for a change in H0 from 10^{-7} to 10^{-8}. Although not shown here, the solution was found to be adversely affected by a poor choice for H0. Also, some values for H0 resulted in problems with solution instability.

Comparing GCKP84 with CREK1D and LSODE was difficult because of the much lower values of EPS used by GCKP84. These low values were necessary to satisfy the two independent accuracy criteria used above. For test problem 1, GCKP84 was significantly slower than EPISODE for the same value of EPS. For test problem 2 its speed was comparable to those of LSODE-B and EPISODE-B for the same values of EPS. For larger values of EPS, however, GCKP84 was significantly slower than LSODE, EPISODE, and CREK1D (ref. 30). It should be emphasized that GCKP84 is a general-purpose chemical kinetics code designed to solve a variety of chemical kinetics problems. Consequently the overhead associated with functional and Jacobian evaluations is higher for GCKP84. In spite of the extra work required per step GCKP84 has been shown to be an efficient code for performing a wide variety of chemical kinetics calculations (ref. 18).

TABLE VIII.—EFFECT OF INITIAL STEP LENGTH ON WORK REQUIRED BY EPISODE-A (EPS = 10^{-5}) FOR TEST PROBLEM 2

Initial step length, H0, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total CPU time required, CPU, s
10^{-5}	129	237	33	0.79
10^{-6}	129	231	31	.78
10^{-7}	126	225	36	.79
10^{-8}	1168	2355	353	7.91
10^{-9}	1170	2394	362	8.04
10^{-10}	133	231	32	.77

Computational Tactics

A simple way of increasing the efficiency of the algorithms as applied to the present problem was explored. Experience with CHEMEQ-A showed that computing the rate constants k_j [$=A_j T^{N_j} \exp(-T_j/T)$] and k_{-j} [$=A_{-j} T^{N-j} \exp(-T_{-j}/T)$] every time the species time derivatives are evaluated is quite inefficient. Note that evaluating the rate constants necessitates computing both the exponential terms in the expressions for k_j and k_{-j} (eqs. 5(a) and 6(b)) and the fifth-order polynomial expression used for the Gibbs functions (ref. 26). To reduce the computational work associated with evaluating the rate constants, these were updated only when the temperature change exceeded a value ΔT. Shown in table IX are the CPU times for various values of ΔT using CHEMEQ-A on test problem 1. Note the substantial decrease in computational work—a value of $\Delta T = 0.1$ kelvin resulted in about a 40-percent decrease in both the number of steps and the number of function evaluations. Furthermore the number of rate constant evaluations NRRC decreased from 27 736 to 2578 for $\Delta T = 0.1$ kelvin. These reductions in computational work resulted in about a 70-percent decrease in the CPU time required to solve the problem.

TABLE IX.—EFFECT OF MAXIMUM TEMPERATURE CHANGE ALLOWED BEFORE REACTION RATE CONSTANT UPDATE ON WORK REQUIRED BY CHEMEQ-A (EPS = 10^{-3}) FOR TEST PROBLEM 1

Maximum temperature change, ΔT, kelvin	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
(a)	13 272	27 736	27 736	28.4
0	13 272	27 736	18 624	24.8
.05	8 061	17 131	3 693	10.4
.1	8 030	17 103	2 578	10.0
.5	13 582	28 394	953	14.4
1	13 974	29 300	627	14.7
(b)	8 027	17 107	2 827	9.3

[a]Not used.
[b]Calculated by using eq. (16).

In selecting a value for ΔT, care must be taken to avoid poor approximations in the resulting reaction rates, which leads to excessive computational work. Table X presents an example of such a case. A value of ΔT as small as 1 kelvin resulted in an order-of-magnitude increase in the CPU time required to solve the problem. The selection of an optimum value for ΔT—defined as that value which results in minimum computational work—is therefore a trial-and-error process, with it being a function of the problem and the error tolerance specified.

To avoid repeated runs of the program in search of the optimum value for ΔT, an approximation for it was developed. The results are given below and details are presented in appendix C. By requiring that the maximum relative error in the resultant reaction rates not exceed the required relative error tolerance EPS, the following expression for ΔT was derived:

$$\Delta T = \frac{\text{EPS T}}{\max_j \left| \frac{T_j}{T} + N_j ; \frac{T_{-j}}{T} + N_{-j} \right|} \qquad (16)$$

where T is the current value of the temperature, the bars | | denote absolute value, and the maximum is taken over all forward and reverse reactions.

Every time the reaction rate constants were evaluated, which occurred only when the temperature change (since the last update of the rate constants) exceeded ΔT, a new value of T could be calculated from equation (16). Thus equation (16) provides a simple expression for the automatic evaluation (through the history of the problem) of ΔT. The computational work resulting from the use of equation (16) is given in tables IX and X. Although using equation (16) did not necessarily result in the fastest algorithm, it did result in shorter CPU times. For CHEMEQ-A this decrease was significant; but for reasons discussed below, it was small for LSODE-A.

Using equation (16) resulted in significant savings for CHEMEQ and DASCRU for all values of EPS used in this study. But with EPISODE and LSODE the savings were found to be small, especially when low values of EPS were used. The decreases in CPU time were more significant for CHEMEQ and DASCRU because these two codes require many more reaction rate evaluations (NRRC in tables XI and XII) than EPISODE and LSODE. For EPISODE and LSODE the computational work required for evaluating ΔT by equation (16) is therefore a greater fraction of the work saved by not updating the rate constants than for CHEMEQ and DASCRU. In addition, when EPS was small, the resulting ΔT has such small values that the decrease in NRRC (and hence in computational work) is not sufficient to offset the work required to update ΔT. This update occurs more often as EPS is decreased.

In incorporating equation (16) into codes that use temperature method B, the following attempt at further reducing the computational work was made. In addition to updating the rate constants only for temperature changes greater than ΔT, the thermodynamic properties h_i and $c_{p,i}$ (eq. (8)) were updated only for temperature changes greater than ΔT. This reduced the work associated with computing the fifth-order polynomial approximations used in evaluating h_i and $c_{p,i}$ (ref. 26). For CHEMEQ-B and DASCRU-B this resulted in increased efficiency for all values of EPS used in this study. But for LSODE-B and EPISODE-B no consistent differences could be observed. Hence with LSODE-B and EPISODE-B the thermodynamic properties were

TABLE X.—EFFECT OF MAXIMUM TEMPERATURE CHANGE ALLOWED BEFORE REACTION RATE CONSTANT UPDATE ON WORK REQUIRED BY LSODE-A (EPS = 10^{-4}) FOR TEST PROBLEM 1

Maximum temperature change, ΔT, kelvin	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
(a)	206	293	33	326	0.63
0	206	293	33	263	.58
.05	227	317	32	203	.58
.1	220	296	34	171	.54
.5	438	642	109	209	1.12
1	1224	2163	550	317	3.88
(b)	213	289	31	224	.57

[a]Not used.
[b]Calculated by using eq. (16).

TABLE XI.—WORK REQUIRED FOR TEST PROBLEM 1

[Given CPU times represent minimum time required by each code to solve test problem 1.]

Method	Error tolerance, EPS	Relative error tolerance, ERMAX	Initial step length, H0, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
GCKP84	10^{-6}	(a)	[b]10^{-6}	309	684	54	684	3.13
CREK1D	10^{-2}	(a)	(a)	84	280	32	91	.23
LSODE-A	10^{-2}	10^{-2}	(a)	114	165	27	105	.32
LSODE-B	5×10^{-3}	[c]0	(a)	101	156	27	113	.32
EPISODE-A	10^{-6}	10^{-2}	10^{-10}	244	463	41	435	.70
EPISODE-B	10^{-6}	(a)	10^{-9}	248	460	36	447	.68
CHEMEQ-A	10^{-2}	10^{-2}	(a)	6 741	13 963	0	735	6.58
CHEMEQ-B	10^{-3}	(a)	(a)	11 884	24 943	↓	2 638	9.82
DASCRU-A	10^{-5}	10^{-2}	10^{-7}	12 997	70 075		31 395	43.2
DASCRU-B	10^{-5}	(a)	10^{-7}	12 822	69 185		31 827	35.5

[a]Not needed.
[b]Default value.
[c]Absolute error tolerance for temperature.

TABLE XII.—WORK REQUIRED FOR TEST PROBLEM 2

[Given CPU times represent minimum time required by each code to solve test problem 2.]

Method	Error tolerance, EPS	Relative error tolerance, ERMAX	Initial step length, H0, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
GCKP84	10^{-5}	(a)	[b]10^{-6}	137	313	34	313	3.06
CREK1D	10^{-3}	(a)	(a)	140	439	138	214	1.04
LSODE-A	10^{-2}	10^{-2}	(a)	87	137	26	87	.47
LSODE-B	10^{-2}	[c]0	(a)	72	126	22	75	.45
EPISODE-A	10^{-5}	10^{-4}	10^{-6}	127	227	31	229	.71
EPISODE-B	10^{-5}	(a)	10^{-9}	145	303	34	273	.86
CHEMEQ-A	10^{-2}	10^{-2}	(a)	8 167	17 033	0	1 171	13.7
CHEMEQ-B	10^{-2}	(a)	(a)	8 745	18 213		974	12.0
DASCRU-A	10^{-3}	10^{-2}	10^{-9}	94 596	594 640		11 666	400
DASCRU-B	10^{-3}	(a)	10^{-9}	94 627	593 940		11 057	310

[a]Not needed.
[b]Default value.
[c]Absolute error tolerance for temperature.

updated every time the derivatives and the Jacobian matrix were evaluated. But with CHEMEQ-B and DASCRU-B these properties were evaluated only when the rate constants were updated.[6] We note here that CREK1D allows for a user-specified ΔT and both the rate constants and the thermodynamic properties are updated only for temperature changes greater than ΔT.

The changes discussed above were incorporated into CHEMEQ, DASCRU, EPISODE, and LSODE. The computational work required by each code for test problems 1 and 2 is presented in tables XI and XII, respectively. For temperature method A, ERMAX was allowed to be different from EPS (cf. figs. 3 and 4) and the value resulting in the least computational work was used. For LSODE-B the value for the absolute error tolerance for the temperature was chosen similarly. Tables XI and XII present the values for the user-specified parameters that result in the least computational work. The CPU times given in tables XI and XII represent the minimum time required by each code to solve test problems 1 and 2, respectively.

Comparing tables XI and XII with figures 3 and 4 shows the savings realized by using equation (16). For DASCRU (classical Runge-Kutta method) and CHEMEQ the decreases in CPU time were significant,

[6]With method A, since a Newton-Raphson iteration technique was used to compute the temperature, the thermodynamic properties were updated every time the derivatives and the Jacobian matrix were evaluated.

ranging from about 30 to about 65 percent. For LSODE and EPISODE the decreases were more modest, ranging from negligibly small to about 20 percent.

Tables XI and XII show that CREK1D was the fastest code for test problem 1, and LSODE for test problem 2. Note that although CREK1D is only marginally faster than LSODE for problem 1, it is significantly slower for problem 2. We therefore conclude that LSODE is the fastest code currently available for integrating chemical kinetic rate equations. LSODE is faster than CREK1D because of (1) better step length control, especially at early times (see the section Step Length Comparisons for details) and (2) less frequent update of the Jacobian. For test problem 2, CREK1D requires 138 Jacobian evaluations, whereas LSODE-B requires only 22 Jacobian evaluations (table XII). The main difficulty in using LSODE is the trial-and-error procedure necessary to obtain optimum values for the absolute error tolerances for a given value of EPS.

Accuracy Comparisons

The standard solutions established in the section Accuracy Estimate and given in tables IV and V were used to compare the accuracy of the algorithms examined in this study. Tables XIII and XIV give the differences from the standard solutions, in percent, for the solutions obtained with the optimized codes. The values chosen for the user-specified parameters are given in tables XI and XII for test problems 1 and 2, respectively. As expected, all temperature predictions were in close agreement with the standard solutions, except for the immediate postinduction solutions with GCKP84 and EPISODE. Unlike the other codes CHEMEQ tracked the temperature more accurately during induction and heat release than during equilibration. During equilibration all other codes tracked the temperature well, with GCKP84 and EPISODE being superior to the other codes because of the use of a much smaller error tolerance. These temperature differences show that even for solutions generated with large values of EPS, the differences between temperature methods A and B were insignificant. Furthermore, for test problem 1, solutions generated with EPISODE-A and DASCRU-A were as accurate as those generated, respectively, with EPISODE-B and DASCRU-B, although ERMAX was much larger than EPS (table XI). Similar remarks apply to test problem 2 and the runs with EPISODE and DASCRU.

The numbers in parentheses in tables XIII and XIV are values for the percent differences obtained by ignoring all species with standard solution values for mole fractions less than 10^{-7}. This was done because the selection criterion used in the section Efficiency Comparisons involved only species with mole fractions greater than or equal to 10^{-7}. All mole fractions were larger than 10^{-7} for times t greater than or equal to 5×10^{-5} and 10^{-4} for test problems 1 and 2, respectively.

The mole fraction differences from the standard solutions were largest at times immediately after ignition and during heat release because the solutions were changing rapidly in these regimes. During ignition and heat release GCKP84 and EPISODE were inferior to the other codes. However, at longer times they were superior to the other codes. These results for species mole fractions are consistent with the solutions for the temperature. In addition, as discussed in the section Efficiency Comparisons, although the use of larger error tolerances gave poor solutions during ignition and heat release, the predicted solutions were satisfactory at longer times. Hence, where the main objective of modeling is to predict pollutant (e.g., NO_x) formation, the user can specify fairly large error tolerances without incurring severe error penalties. Again, we note that the differences between the solutions generated with methods A and B (and the same value of EPS) are not significant. Among the codes that use a large error tolerance, CHEMEQ was the most accurate during ignition and early heat release; during late heat release and equilibration, LSODE and CREK1D were superior to the other codes.

EPISODE was inferior to LSODE and CREK1D at early times because of the difference in the error control performed by these codes. In contrast to the other codes for which EPS is the local relative error tolerance, EPS is a mixed error tolerance for EPISODE; that is, it is relative for species with nonzero initial mole numbers and absolute otherwise. Since most of the species had zero initial mole numbers for both problems examined in this study (tables IV and V), the error control performed by EPISODE was mostly absolute. For pure relative error control the estimated local truncation error E_i in species i satisfies

$$E_i \leq \epsilon_{rel} n_i \tag{17}$$

where ϵ_{rel} is the local relative error tolerance. For pure absolute error control E_i satisfies

$$E_i \leq \epsilon_{abs} \tag{18}$$

where ϵ_{abs} is the local absolute error tolerance. Equations (17) and (18) show that, since $n_i << 1$, relative error control is more accurate for the same value of the local error tolerance. In other words, to attain comparable accuracy for mole numbers that were initially zero, EPISODE requires lower values of EPS than the other codes. Equations (17) and (18) also show that the error controls are of comparable accuracy for n_i of order $O(\epsilon_{abs}/\epsilon_{rel})$, a result that can be used to examine the difference in accuracy attained by EPISODE and

TABLE XIII.—DIFFERENCES FROM STANDARD SOLUTIONS FOR TEST PROBLEM 1

[Numbers in parentheses are values obtained by ignoring species with mole fractions less than 10^{-7}.]

Method	Time, t, s					
	9×10^{-6}	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}
Difference in mole fractions, (rms), percent						
GCKP84	4.001×10^{3} (266.0)	3.866×10^{4} (228.8)	2.076	0.795	0.138	0.121
CREK1D	689.5 (3.965)	2.822×10^{3} (4.008)	.502	.515	.274	.257
LSODE-A	766.6 (10.05)	3.082×10^{3} (9.280)	.619	.700	.512	.122
LSODE-B	2.002×10^{6} (10.32)	1.044×10^{10} (9.687)	.871	.764	.335	.184
EPISODE-A	4.781×10^{3} (310.4)	5.659×10^{4} (267.0)	2.157	.784	.0873	.0701
EPISODE-B	7.702×10^{4} (317.2)	6.789×10^{7} (273.0)	2.182	.791	.0881	.0725
CHEMEQ-A	647.1 (0.305)	2.618×10^{3} (0.299)	10.48	15.64	32.72	42.17
CHEMEQ-B	649.4 (0.154)	2.622×10^{3} (0.150)	5.306	10.39	19.17	20.80
DASCRU-A	1.785×10^{3} (107.8)	9.228×10^{3} (94.66)	1.174	.435	.225	.350
DASCRU-B	1.819×10^{3} (110.8)	9.486×10^{3} (97.14)	1.183	.430	.232	.391
Difference in mole fractions (maximum), percent						
GCKP84	1.324×10^{4} (374.5)	1.282×10^{5} (330.7)	5.528	1.957	0.451	0.360
CREK1D	2.285×10^{3} (5.485)	9.359×10^{3} (5.746)	−.862	.893	−.584	.598
LSODE-A	2.541×10^{3} (13.72)	1.022×10^{4} (13.10)	−1.398	−1.498	−1.033	−.234
LSODE-B	6.641×10^{6} (14.42)	3.464×10^{10} (14.25)	−1.695	−1.624	−.697	.459
EPISODE-A	1.583×10^{4} (439.6)	1.877×10^{5} (386.5)	5.749	1.895	.285	.209
EPISODE-B	2.555×10^{5} (449.6)	2.252×10^{8} (395.1)	5.814	1.917	.288	.212
CHEMEQ-A	2.144×10^{3} (-0.446)	8.683×10^{3} (-0.440)	19.08	28.49	−56.32	−72.11
CHEMEQ-B	2.152×10^{3} (-0.236)	8.695×10^{3} (-0.208)	8.920	18.32	36.89	40.45
DASCRU-A	5.911×10^{3} (148.8)	3.060×10^{4} (135.2)	3.082	.953	−.397	−.595
DASCRU-B	6.022×10^{3} (153.0)	3.146×10^{4} (138.8)	3.114	.953	−.424	−.684
Difference in temperatures, percent						
GCKP84	0.899	4.572	0.208	0.0795	0	0.0381
CREK1D	.0497	.111	.0606	.0357	.0333	.0322
LSODE-A	.0587	.172	.0951	.0840	.0448	.0360
LSODE-B	.0588	.177	.136	.103	.0501	.0442
EPISODE-A	1.107	5.548	.204	.0628	.00629	.0236
EPISODE-B	1.137	5.697	.207	.0634	.00620	.0236
CHEMEQ-A	.0365	.0430	−1.466	−1.635	−1.917	−2.163
CHEMEQ-B	.0333	.0409	.449	.904	1.491	1.539
DASCRU-A	.300	1.590	.0831	.0398	−.0381	0
DASCRU-B	.300	1.590	.125	.0398	−.0381	−.0381

TABLE XIV.—DIFFERENCES FROM STANDARD SOLUTIONS FOR TEST PROBLEM 2

[Numbers in parentheses are values obtained by ignoring species with mole fractions less than 10^{-7}.]

Method	Time, t, s						
	3×10^{-6}	5×10^{-6}	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}
Difference in mole fractions (rms), percent							
GCKP84	143.6	8.772×10^{7}	8.447	0.731	0.615	0.852	0.835
	(103.2)	(15.98)	(5.925)	(0.723)			
CREK1D	4.106	3.570×10^{7}	0.322	0.0458	.0459	.213	.297
	(1.089)	(0.504)	(0.222)	(0.0435)			
LSODE-A	11.62	4.486×10^{7}	10.62	0.444	1.011	.0721	.170
	(8.642)	(1.977)	(8.086)	(0.380)			
LSODE-B	653.9	10.85	12.45	1.854	2.159	.563	.592
	(30.83)	(7.167)	(8.741)	(1.584)			
EPISODE-A	105.7	7.612×10^{7}	6.996	0.798	.276	.0358	.0449
	(76.26)	(14.11)	(4.806)	(0.621)			
EPISODE-B	157.3	8.743×10^{7}	8.542	0.947	.358	.0260	.0174
	(101.0)	(15.87)	(5.883)	(0.737)			
CHEMEQ-A	3.830	3.457×10^{7}	0.0528	2.885	10.25	26.71	37.85
	(0.198)	(0.0893)	(0.0382)	(2.570)			
CHEMEQ-B	3.839	3.447×10^{7}	0.103	1.591	5.990	25.40	38.50
	(0.234)	(0.138)	(0.0735)	(1.433)			
DASCRU-A	30.17	10.23	25.90	0.362	8.571	.477	9.640
	(23.24)	(7.541)	(1.829)	(0.280)			
DASCRU-B	30.18	1.247×10^{7}	2.615	0.261	.564	.641	7.164
	(23.24)	(7.085)	(1.787)	(0.201)			
Difference in mole fractions (maximum), percent							
GCKP84	317.5	3.397×10^{8}	21.03	1.471	−1.396	−1.622	−1.764
	(164.2)	(−26.75)	(21.03)	(1.471)			
CREK1D	−10.382	1.383×10^{8}	0.768	−0.101	−.122	−.446	−.609
	(1.728)	(−0.854)	(0.768)	(−0.101)			
LSODE-A	21.85	1.738×10^{8}	29.12	0.965	−1.706	.144	.382
	(13.64)	(−3.207)	(29.12)	(0.934)			
LSODE-B	1.987×10^{3}	30.55	36.45	4.051	3.932	1.229	1.203
	(48.95)	(−12.25)	(31.51)	(3.821)			
EPISODE-A	222.4	2.948×10^{8}	17.13	2.034	.717	.0684	.131
	(121.5)	(−24.06)	(17.13)	(1.820)			
EPISODE-B	309.8	3.386×10^{8}	20.97	2.422	.898	.0621	.0303
	(160.7)	(−26.87)	(20.97)	(2.179)			
CHEMEQ-A	−12.32	1.339×10^{8}	−0.151	−6.203	−20.52	−46.38	−63.40
	(−0.334)	(0.177)	(−0.129)	(−6.203)			
CHEMEQ-B	−12.51	1.335×10^{8}	−0.281	−3.034	11.18	44.45	65.14
	(−0.402)	(0.256)	(−0.260)	(−2.731)			
DASCRU-A	57.47	23.44	−100.0	0.934	33.19	.957	37.25
	(38.03)	(−14.62)	(6.530)	(0.840)			
DASCRU-B	57.50	4.830×10^{7}	6.359	0.674	−2.174	−1.244	27.24
	(38.03)	(−11.89)	(6.359)	(0.630)			
Difference in temperatures, percent							
GCKP84	0.333	2.203	0.733	0.0795	0	0	0
CREK1D	.00477	.0875	.0152	−.00135	.0218	.0126	−.00641
LSODE-A	.0306	.370	−.266	.146	.203	−.0179	−.0282
LSODE-B	.105	1.017	−.140	.310	.373	.0704	.0406
EPISODE-A	.261	1.856	.635	.0739	.0361	−.00448	−.00292
EPISODE-B	.344	2.194	.770	.0910	.0466	−.00447	−.00421
CHEMEQ-A	.00486	.0209	−.0227	−.524	−1.381	−1.025	−1.227
CHEMEQ-B	−.00189	.0108	−.0288	−.107	.0265	1.283	1.678
DASCRU-A	.0665	.796	.262	.0398	.0367	0	.0687
DASCRU-B	.0665	.796	.262	0	0	−.0687	−.138

LSODE, for mole numbers that are initially zero. Using the values for the error tolerances given in tables XI and XII, we found that the two codes were of comparable accuracy for n_i of order $O(10^{-4})$ and $O(10^{-3})$ for test problems 1 and 2, respectively. It should be emphasized that these values are only estimates because the two codes control the root-mean-square norm of the estimated local errors and not the estimated local error for each species.

These estimates for mole numbers correspond to mole fractions of order $O(10^{-3})$ and $O(10^{-2})$ for test problems 1 and 2, respectively, assuming a mixture mean molar mass of order O(10). Tables IV and V and figures 1 and 2 show that these values were not attained at early times by species with initial zero mole numbers. Hence EPISODE is expected to be inferior to LSODE during ignition and early heat release. However, during late heat release and equilibration, when many of the mole numbers exceed these values, EPISODE is expected to be superior to LSODE. Examination of tables XIII and XIV confirms this behavior. We note that, relative to LSODE, EPISODE was (1) inferior at short times, (2) superior at long times, and (3) comparable at intermediate times.

These observations imply that the error control used by EPISODE is unsatisfactory for problems of the type examined in this study. Many of the variables have zero initial values and they never reach unity. The nature of the error control performed by EPISODE therefore requires small values of EPS for acceptable accuracy at short times. The continued use of these small values of EPS at long times is wasteful because it results in solutions that are more accurate than required by the selection criterion. A simple method for increasing the efficiency of EPISODE is therefore to switch to pure relative error control (IERROR = 2) with a larger value of EPS once the species mole numbers have reached acceptable values. Tables XV and XVI present the effects of such a switch. In these tables, t_{switch} is the time at which the switch was made. The program was run up to

TABLE XV.—EFFECT OF SWITCH TO PURE RELATIVE ERROR CONTROL ON COMPUTATIONAL WORK REQUIRED FOR TEST PROBLEM 1

(a) EPISODE-A

Time at which error control is switched, t_{switch}, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
10^{-5}	121	223	39	175	0.37
1.5×10^{-5}	150	290	40	273	.48
2×10^{-5}	164	314	40	294	.51
2.5×10^{-5}	175	332	40	304	.53
5×10^{-5}	194	369	42	347	.58
10^{-4}	209	394	41	367	.62
5×10^{-4}	236	444	44	415	.68
[a]10^{-3}	244	463	41	435	.70

(b) EPISODE-B

Time at which error control is switched, t_{switch}, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
1.5×10^{-5}	140	257	37	240	0.43
2×10^{-5}	148	278	37	258	.45
2.5×10^{-5}	157	292	36	277	.47
5×10^{-5}	181	340	37	322	.53
10^{-4}	201	367	39	350	.57
5×10^{-4}	234	436	41	413	.66
[a]10^{-3}	248	460	36	447	.68

[a]No switching performed.

TABLE XVI.—EFFECT OF SWITCH TO PURE RELATIVE ERROR CONTROL ON COMPUTATIONAL WORK REQUIRED FOR TEST PROBLEM 2

(a) EPISODE-A

Time at which error control is switched, t_{switch}, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
3×10^{-6}	92	167	35	132	0.53
4×10^{-6}	119	212	35	172	.64
5×10^{-6}	101	186	31	168	.58
10^{-5}	112	204	36	188	.65
5×10^{-5}	118	218	34	204	.68
10^{-4}	125	227	35	215	.72
5×10^{-4}	128	233	35	219	.72
[a]10^{-3}	127	227	31	229	.71

(b) EPISODE-B

Time at which error control is switched, t_{switch}, s	Total number of steps required, NSTEP	Total number of functional evaluations, NFE	Total number of Jacobian matrix evaluations, NJE	Total number of reaction rate constant evaluations, NRRC	Total CPU time required, CPU, s
4×10^{-6}	88	173	30	140	0.54
5×10^{-6}	104	215	32	170	.62
10^{-5}	119	221	36	191	.67
5×10^{-5}	127	242	32	219	.71
10^{-4}	130	248	33	229	.73
5×10^{-4}	142	296	37	258	.84
[a]10^{-3}	145	303	34	273	.86

[a]No switching performed.

time $t=t_{switch}$ with IERROR = 3 and EPS of 10^{-6} and 10^{-5} for test problems 1 and 2, respectively. At $t=t_{switch}$ the integrator was reinitialized, the error control was switched to IERROR = 2, and EPS was increased to 10^{-2}. The problem was then run to completion with these new values for IERROR and EPS. Note the significant decreases in the computational work obtained by switching to pure relative error control. The resultant CPU times compare favorably with those required by LSODE. This switching process is, however, unsatisfactory for the following reasons:

(1) The value of t_{switch} that minimizes the CPU time is a function of the problem and the temperature method used. Moreover, a poor choice for t_{switch} can make the run prohibitively expensive. For example, for test problem 1 the run with EPISODE-B and $t_{switch}=10^{-5}$ was not successfully completed even after a CPU time of 2 min.

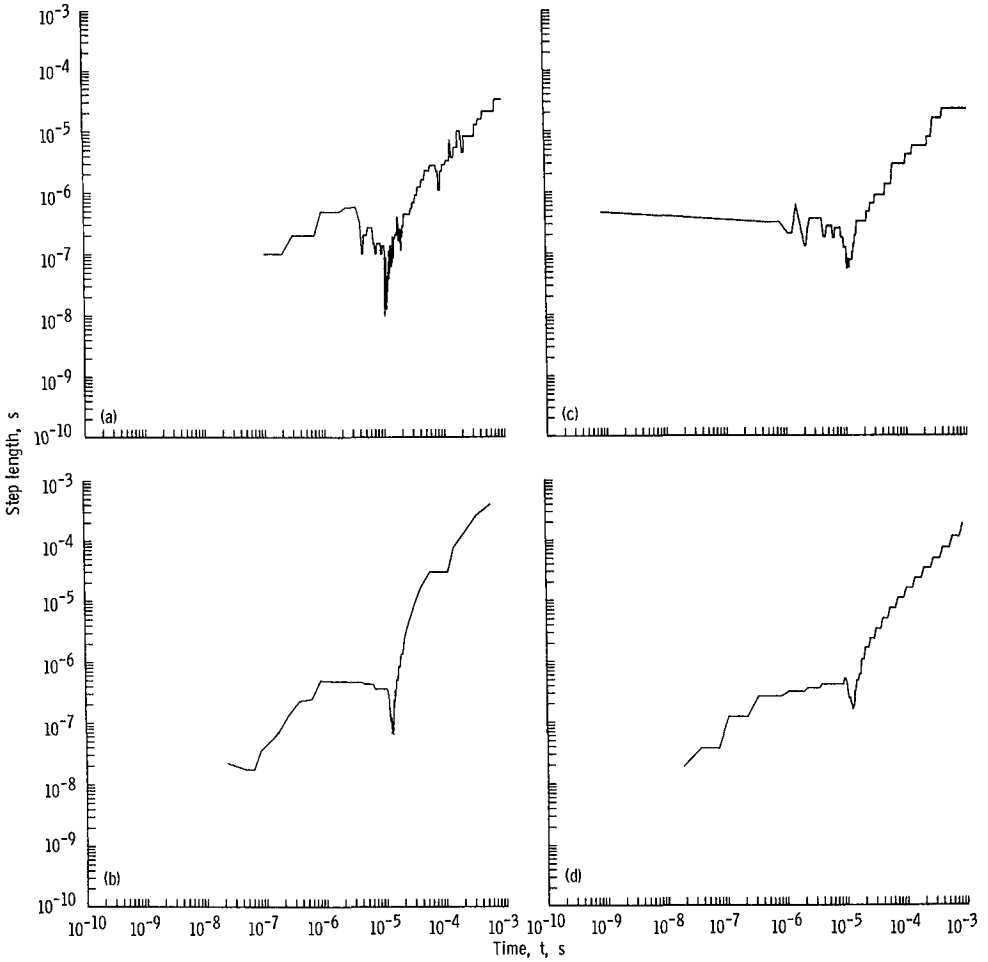
(2) The switching requires unnecessary computational work associated with the reinitialization and restart of the integrator.

(3) Using low values of EPS for $t \leq t_{switch}$ results in tighter error control than is necessary to satisfy the accuracy criterion for species with initially nonzero mole numbers and for the temperature.

A more appropriate error control for problems of the type examined in this study is therefore one that is absolute for mole numbers that are initially zero and that automatically becomes relative once these mole numbers reach acceptable values. The error control should also be relative for species with initially nonzero mole numbers and for the temperature. Such an error criterion can be realized by requiring E_i to satisfy

$$E_i \leq \epsilon_{rel} n_i + \epsilon_{abs,i} \tag{19}$$

where $\epsilon_{abs,i}$ is the local absolute error tolerance for



(a) GCKP84. (b) CREK1D. (c) EPISODE. (d) LSODE.

Figure 5.—Variation with time of the step length successfully used by GCKP84, CREK1D, EPISODE, and LSODE for test problem 1.

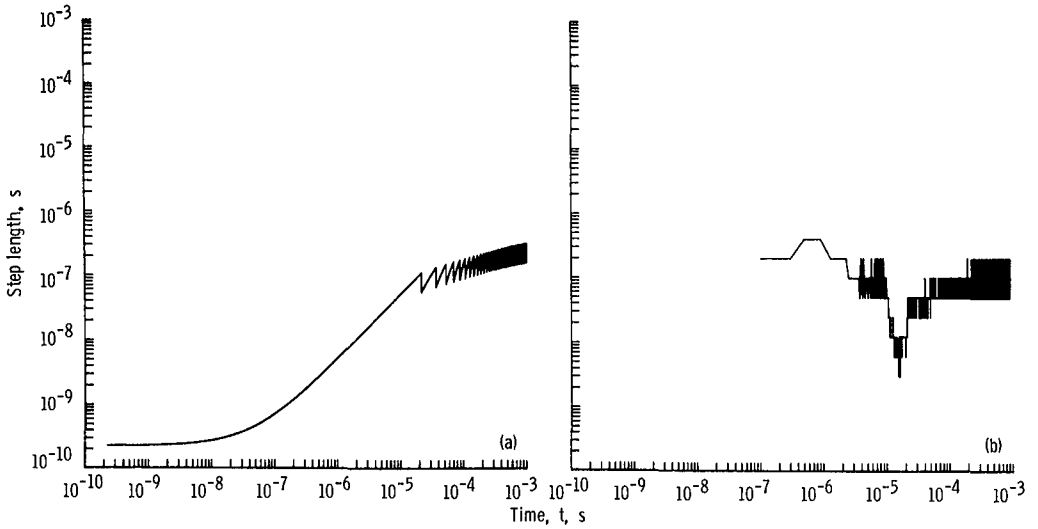
species i. For values of $n_i < (\epsilon_{abs,i}/\epsilon_{rel})$ the error control is mostly absolute, and for $n_i > (\epsilon_{abs,i}/\epsilon_{rel})$ it is mostly relative. This is, in fact, the nature of the error control used in LSODE; it is therefore more efficient than EPISODE for integrating combustion kinetic rate equations.

Step Length Comparisons

All codes used in the present study automatically select a step length during the course of the integration. Some of the codes (GCKP84, DASCRU, and EPISODE) require trying a user-supplied initial value. The other codes automatically select the value for the initial step length. 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 step length used by each code through the course of each problem. To facilitate comparisons (among the faster codes) at early times for test problem 2, figure 9 presents the variation of the step length between $t = 10^{-6}$ and $t = 10^{-5}$.

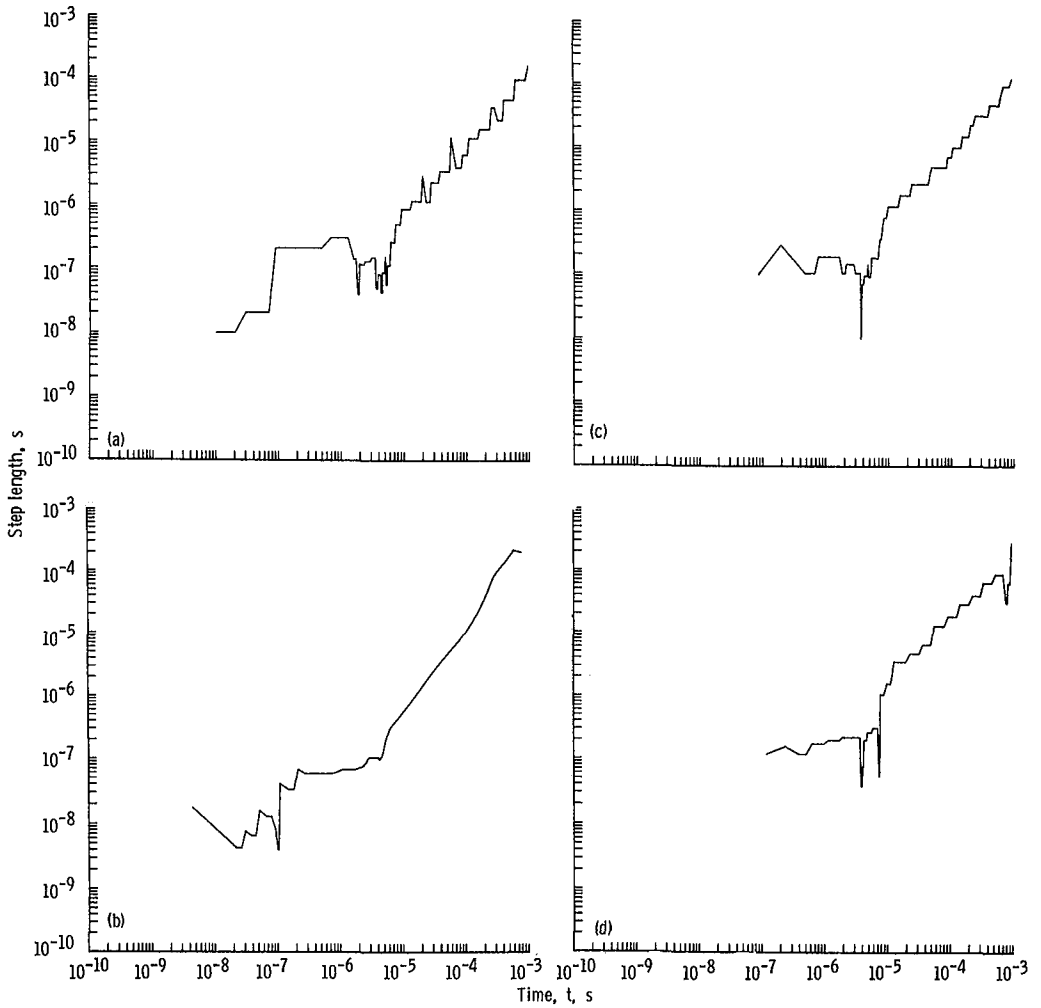
Figures 6 and 8 illustrate the small steps that classical methods have to use to ensure solution stability. For both test problems the explicit Runge-Kutta technique used small step lengths to track the solutions through induction and heat release. During equilibration the step lengths remained small, and thus prohibitive amounts of computer time were required. The difficulties with CHEMEQ (figs. 6 and 8) included the selection of a very small initial step length, the continued use of small step lengths because of the very small increases allowed after satisfactory convergence, and its inability to select a suitable step length during equilibration. Much computer time was wasted in the search for an appropriate step length. In addition, the search was restricted to very small step lengths. These factors make CHEMEQ very expensive to use.

We note that all codes use small steps during induction and early heat release. In these regimes the species and temperature change rapidly (figs. 1 and 2). Most of the species and temperature have positive time constants—an indication that the differential equations are unstable.



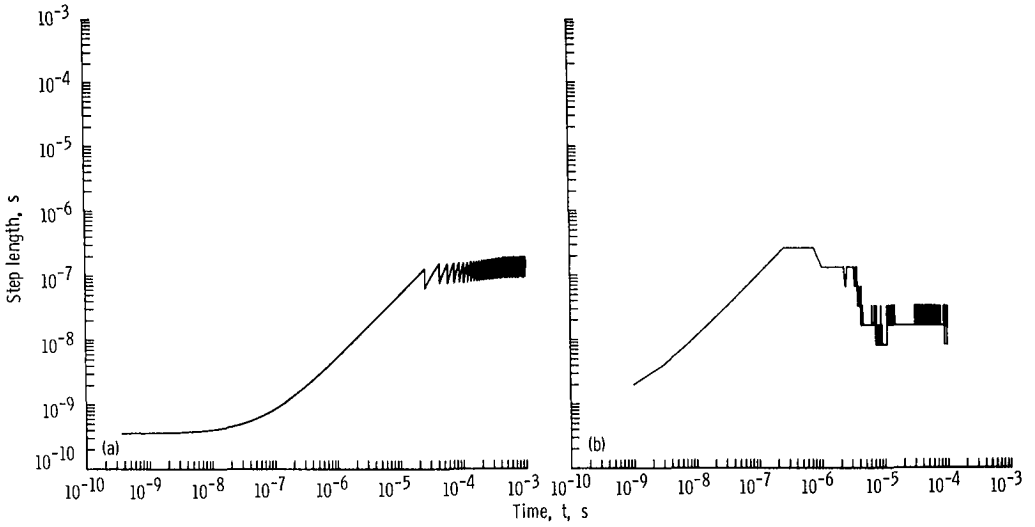
(a) CHEMEQ-A. (b) DASCRU-B.

Figure 6.—Variation with time of the step length successfully used by CHEMEQ-A and DASCRU-B for test problem 1.



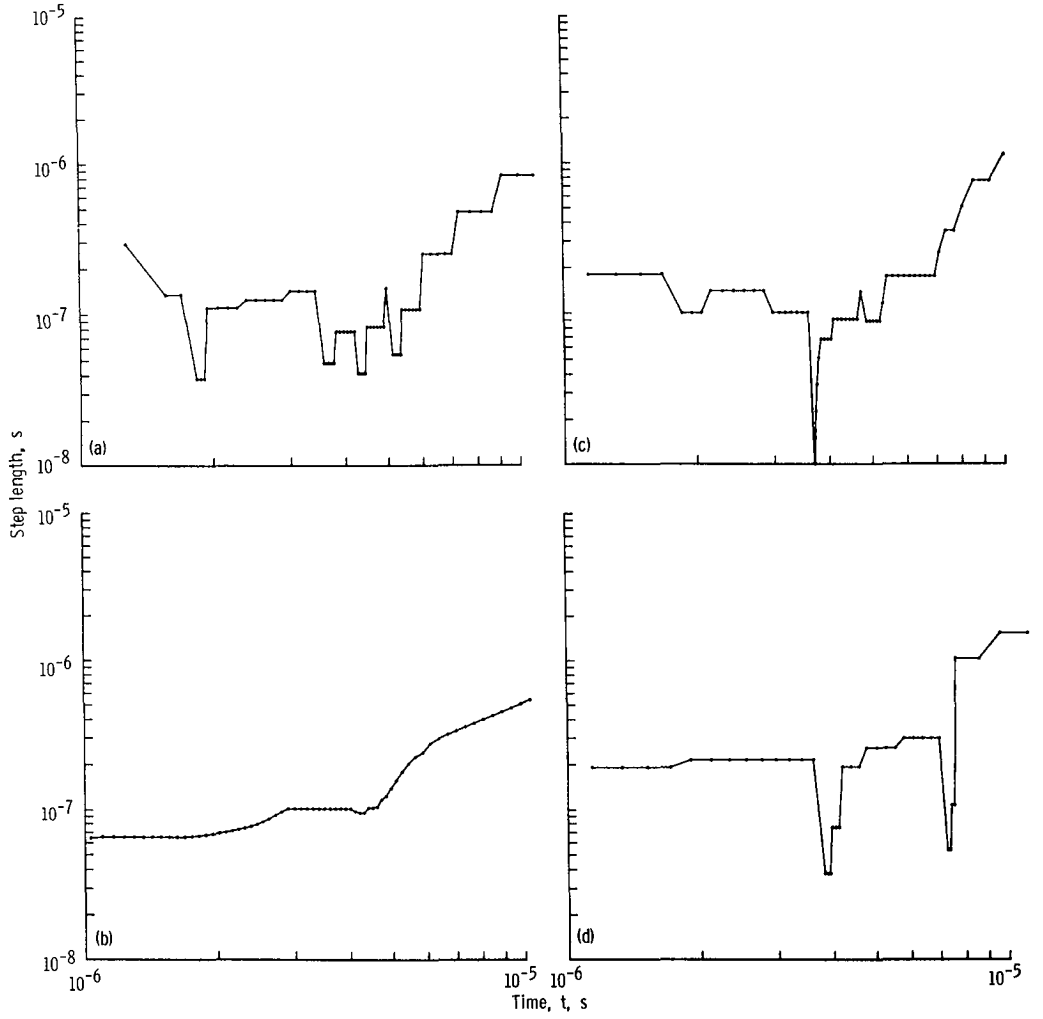
(a) GCKP84. (b) CREK1D. (c) EPISODE. (d) LSODE.

Figure 7.—Variation with time of the step length successfully used by GCKP84, CREK1D, EPISODE, and LSODE for test problem 2.



(a) CHEMEQ-B. (b) DASCRU-A.

Figure 8.—Variation with time of the step length successfully used by CHEMEQ-B and DASCRU-A for test problem 2.



(a) GCKP84. (b) CREK1D. (c) EPISODE-A. (d) LSODE-A.

Figure 9.—Variation with time of the step length successfully used at early times by GCKP84, CREK1D, EPISODE-A, and LSODE-A for test problem 2.

Hence the step lengths are constrained to small values.

For test problem 1, EPISODE was superior to GCKP84 and CREK1D was superior to LSODE. Note that although in the postinduction regime EPISODE selected steps comparable to those selected by LSODE, its difficulty in tracking the solution during ignition and early heat release made it less efficient than LSODE. For test problem 2, at long times, CREK1D took step lengths comparable to those of LSODE. However, at times preceding and immediately after ignition ($t \simeq 3$ μs), CREK1D took much smaller step lengths and was hence less efficient than LSODE. Although not presented herein, the run with CREK1D and EPS = 10^{-2} showed that this code had difficulty selecting a suitable step length at early times. Much computer time was wasted by repeatedly unsuccessful attempts at selecting a larger step length. This run was therefore less efficient than the one with EPS = 10^{-3} (fig. 4). The step lengths selected by EPISODE were comparable to those selected by GCKP84, although the former used larger steps initially.

The results discussed above indicate that the step length to be used is regime dependent: during induction and early heat release, when the solution changes rapidly, small steps have to be taken to ensure solution stability. The search for large step lengths is both futile and expensive. At later times, however, when the differential equations are more stable, larger step lengths can be used. In these regimes it is worth attempting to use a larger step length after every step.

Summary of Results and Conclusions

Several algorithms (GCKP84, CREK1D, LSODE, EPISODE, CHEMEQ, and DASCRU) for numerically integrating stiff ordinary differential equations arising in combustion chemistry have been compared. The following performance indicators were recorded for purposes of this comparison:

(1) Total number of steps required to solve a problem
(2) Total number of derivative evaluations
(3) Total number of Jacobian matrix evaluations
(4) Total number of rate constant evaluations
(5) Total CPU time required to solve a problem

In addition, the errors relative to reference solutions (generated with LSODE with a very low relative error tolerance) were recorded at selected times. These tests were conducted on two combustion kinetics problems: one involving 11 species and temperature with 12 reactions, and the other involving 15 species and temperature with 30 reactions. Both problems included all three combustion regimes: induction, heat release, and equilibration.

This study has shown that for comparable accuracy the fastest package currently available for integrating combustion kinetic rate equations is LSODE. This merits special notice because LSODE was developed as a multipurpose stiff differential equation solver with no one particular application as its objective. Its disadvantages, however, include a large storage requirement and a relatively long startup time. Where storage is a significant problem, both EPISODE and CREK1D are attractive alternatives. However, a poor 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 step length may be necessary to obtain its optimum value. In addition, the error control performed by EPISODE was found to be unsatisfactory for problems of the type examined in this study. For acceptable accuracy at short times small values of the error tolerance had to be used. Significant decreases in computational work were obtained by switching to pure relative error control in the postinduction regimes. An error control that is more appropriate for combustion kinetics problems is required for additional increases in computational efficiency. The code CREK1D needs refinement in the area of step-length selection, especially at early times, before significant increases in its speed can be realized.

The step length to be used in integrating combustion kinetic rate equations was found to be regime dependent. During induction and early heat release, small steps had to be taken to ensure solution stability. At later times, however, when the differential equations were more stable, larger steps could be used.

An important conclusion from this study is that using an algebraic energy conservation equation to calculate the temperature does not result in significant errors or inefficiencies. On the contrary, this method can be more accurate and efficient than evaluating the temperature by integrating its time derivative.

Another important conclusion is that, where the objective of modeling is to predict pollutant (e.g., NO_x) formation, the user can specify large error tolerances without incurring significant error penalties.

A simple method for realizing significant efficiency increases was demonstrated. This involved updating the reaction rate constants only for temperature changes greater than an amount that was problem dependent. An expression for the maximum temperature change allowed before such an update was presented and shown to result in significant savings.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, June 6, 1984

Appendix A

Outline of Methods Studied

The ordinary differential equations (2) and (8) describing homogeneous gas-phase chemical reactions can be generalized as follows:

$$\left.\begin{aligned} \dot{y}_i = \frac{dy_i}{dt} = f_i(y_k) \qquad i,k = 1,N \\ y_i(t=0) = y_i(0) = \text{given} \end{aligned}\right\} \tag{A1}$$

where for temperature method A (see the section Evaluation of Temperature)

$$\left.\begin{aligned} y_i = n_i \qquad i = 1,\text{NS} \\ N = \text{NS} \end{aligned}\right\} \tag{A2}$$

and for temperature method B

$$\left.\begin{aligned} y_i = n_i \qquad i = 1,\text{NS} \\ y_{\text{NS}+1} = T \\ N = \text{NS}+1 \end{aligned}\right\} \tag{A3}$$

In vector notation, equation (A1) becomes

$$\left.\begin{aligned} \underline{\dot{y}} = \frac{d\underline{y}}{dt} = \underline{f}(\underline{y}) \\ \underline{y}(t=0) = \underline{y}(0) = \text{given} \end{aligned}\right\} \tag{A4}$$

where the underscore is used to denote a vector quantity. A matrix is denoted by a boldface letter. This notation is used throughout this appendix.

The initial-value problem is to determine values for y_i $(i = 1,N)$ at the end of a prescribed time interval, given f_i $(i = 1,N)$ and the initial values y_i (0) $(i = 1,N)$.

All methods examined in the present study are step-by-step methods. They compute approximations $\underline{y}_n (= y_{i,n};$ $i = 1,N)$ to the exact solution $\underline{y}(t_n)$ at discrete points t_1, t_2, t_3, ... The size h_n of the spacing $(= t_n - t_{n-1})$ may vary from one step to the next. Of the methods examined in this study, some (CHEMEQ, CREK1D, and DASCRU) are single-step methods; the rest (EPISODE, LSODE, and GCKP84) are multistep methods. A single-step method provides a rule for computing $\underline{y}_n$ at t_n from a knowledge of the step length h_n and the solution $\underline{y}_{n-1}$ at the previous step t_{n-1}. A multistep method, on the other hand, uses h_n and the solution at several earlier points to generate $\underline{y}_n$. For example, the K-step method uses h_n and the K earlier values $\underline{y}_{n-1}, \underline{y}_{n-2}, \ldots, \underline{y}_{n-K}$ (at times t_{n-1}, $t_{n-2}, \ldots, t_{n-K}$) to compute $\underline{y}_n$. Since at the outset only the initial conditions $\underline{y}(0)$ are known, multistep methods are not self-starting. EPISODE and LSODE resolve this difficulty at the initial point by starting with a single-step, first-order method. As the integration proceeds, the solutions that they generate provide the necessary values for a multistep method.

We now outline the methods studied in the present work. In particular, we examine how each method advances the solution by one step (of length h_n) from time t_{n-1} to time t_n. To avoid confusion between the time step length h_n and the molal-specific enthalpy for species n, the latter is denoted by $\tilde{h}_n$.

EPISODE and LSODE

Both EPISODE and LSODE use the integration formulas developed by Gear (refs. 20 and 21). These formulas involve linear multistep methods of the form

$$y_{i,n} = \sum_{j=1}^{K_1} \alpha_{n,j} y_{i,n-j} + h_n \sum_{j=0}^{K_2} \beta_{n,j} \dot{y}_{i,n-j} \tag{A5}$$

where $y_{i,n}$ is an approximation to the exact solution $y_i(t_n)$, $\dot{y}_{i,n} [= f_i(y_{k,n})]$ is an approximation to the exact derivative $\dot{y}_i(t_n) \{= f_i[y_k(t_n)]\}$, and the $\alpha_{n,j}$ and $\beta_{n,j}$ $(\beta_{n,0} > 0)$ are associated with the particular formula selected by the user. The options include implicit Adams predictor-corrector formulas and backward differentiation formulas (BDF). As discussed in appendix B, BDF's were found to be more efficient for problems examined in this study. The discussion is therefore restricted to BDF's. For a BDF of order q, $K_1 = q$, $K_2 = 0$, and equation (A5) reduces to

$$y_{i,n} = \sum_{j=1}^{q} \alpha_{n,j} y_{i,n-j} + h_n \beta_{n,0} \dot{y}_{i,n} \tag{A6}$$

The step length h_n $(= t_n - t_{n-1})$ can vary from one step to the next in EPISODE but is held constant for $q+1$

consecutive successful steps in LSODE. Hence, for EPISODE, $\alpha_{n,j}$ and $\beta_{n,j}$ can vary from one step to the next, but in LSODE they are predetermined constants corresponding to the order used. The predicted values of $\underline{y}_n$ and $\dot{\underline{y}}_n$, denoted by $\underline{y}_n^{(0)}$ and $\dot{\underline{y}}_n^{(0)}$ respectively, satisfy the equation

$$y_{i,n}^{(0)} - h_n\beta_{n,0}\dot{y}_{i,n}^{(0)} = \sum_{j=1}^{q} \alpha_{n,j} y_{i,n-j}$$

so that equation (A6) can be written as

$$y_{i,n} = y_{i,n}^{(0)} - h_n\beta_{n,0}\dot{y}_{i,n}^{(0)} + h_n\beta_{n,0}\dot{y}_{i,n} \tag{A7}$$

At each integration step equation (A7) must be solved for $\underline{y}_n$. This is accomplished as follows: let

$$\underline{G}(\underline{y}_n) = \underline{y}_n - h_n\beta_{n,0}\dot{\underline{y}}_n - \left[\underline{y}_n^{(0)} - h_n\beta_{n,0}\dot{\underline{y}}_n^{(0)}\right] \tag{A8}$$

then the solution to equation (A7) is equivalent to finding the zero of $\underline{G}$. Equation (A8) can be solved in a variety of ways. For stiff problems the most efficient is a variant of Newton's method:

$$\sum_{k=1}^{N} P_{ik,n}^{(m)}\left[y_{k,n}^{(m+1)} - y_{k,n}^{(m)}\right] = -G_i\left[\underline{y}_n^{(m)}\right] \qquad i = 1,N \tag{A9}$$

where $m+1$ is the current iteration number and the matrix $\mathbf{P}$ is given by

$$P_{ik,n}^{(m)} = \left(\frac{\partial G_i}{\partial y_k}\right)_{\underline{y}_n^{(m)}}$$

$$= \delta_{ik} - h_n\beta_{n,0}J_{ik,n}^{(m)}$$

where δ_{ik} is the Kronecker symbol,

$$\delta_{ik}\begin{cases} = 0 & i \neq k \\ = 1 & i = k \end{cases}$$

and $J_{ik,n}^{(m)}$ $(i,k = 1,N)$ are elements of the Jacobian matrix $\mathbf{J}$

$$J_{ik,n}^{(m)} = \left(\frac{\partial f_i}{\partial y_k}\right)_{\underline{y}_n^{(m)}}$$

For this method, much computation time is involved in forming the Jacobian matrix $\mathbf{J}$ and in doing the linear algebra necessary to solve equation (A9). To reduce this computational work, the matrix $\mathbf{P}$ is not updated at every iteration. For further savings, it is updated only when it has been determined to be absolutely necessary for convergence. Hence the iteration matrix is only accurate enough for the iteration to converge, and the codes may use the same matrix over several steps of the integration. In any case, both EPISODE and LSODE update $\mathbf{P}$ at least every twentieth step.

The predicted values $y_{i,n}^{(0)}$ and $\dot{y}_{i,n}^{(0)}$ are required before equation (A9) can be solved for $\underline{y}_n$. These are obtained by a Taylor series expansion as follows: the history of the solution is maintained in the Nordsieck array (which is a Taylor series array) $\mathbf{z}_n$ of size $N \times (q+1)$. The i^{th} row $\underline{z}_{i,n}$ contains the $q+1$ elements

$$\underline{z}_{i,n} = y_{i,n}, h_n\dot{y}_{i,n}, \frac{h_n^2}{2!}\ddot{y}_{i,n}, \ldots, \frac{h_n^q}{q!}y_{i,n}^{[q]}$$

where $y_{i,n}^{[j]}$ is the approximation to $(d^j y_i/dt^j)_{t_n}$. Thus, if $\mathbf{z}_{n-1}$ has been obtained,

$$\mathbf{z}_n^{(0)} = \mathbf{z}_{n-1}\mathbf{A}(q)$$

where $A_{jk}(q)$ is given by

$$A_{jk}(q) = \left\{ \begin{matrix} 0 & j<k \\ \binom{j}{k} & j \geq k \end{matrix} \right\} j,k = 0,1,2,\ldots,q \tag{A10}$$

where $\binom{j}{k}$ is the binomial coefficient, defined as

$$\binom{j}{k} = \frac{j!}{k!(j-k)!}$$

Thus the predicted values $\mathbf{z}_n^{(0)}$ are obtained by a simple Taylor series expansion by using equation (A10). Once the predicted values have been calculated, equation (A9) is iterated until convergence. The test for convergence of the iterates $y_{i,n}^{(m)}$ is based on successive differences of these quantities and the local error tolerance EPS as discussed below. For EPISODE a vector YMAX is constructed as follows:

IERROR = 1 (absolute error control):

$\text{YMAX}_i = 1 \qquad i = 1, 2, \ldots, N$

IERROR = 2 (pure relative error control):

$$\text{YMAX}_i = |y_{i,n-1}| \qquad i = 1, 2, \ldots, N$$

IERROR = 3 (semirelative error control):

$$\text{YMAX}_i = \max\{|y_{i,n-1}|, |y_{i,n-2}|\} \text{ for } i \text{ that satisfy } y_i(0) \neq 0$$

$$= \max\{1, |y_{i,n-1}|\} \qquad \text{for } i \text{ that satisfy } y_i(0) = 0$$

Convergence is said to occur if

$$d_m = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_{i,n}^{(m)} - y_{i,n}^{(m-1)}}{\text{YMAX}_i} \right)^2}{N}} \leq \epsilon' = c_E \text{EPS} \qquad \text{(A11)}$$

where ϵ' is proportional to EPS, the local error tolerance.

For LSODE and option ITOL = 2 (see appendix B for other options included in this package), an error vector EWT is constructed as follows:

$$\text{EWT}_i = \epsilon_{\text{rel}} |y_{i,n-1}| + \epsilon_{\text{abs},i} \qquad i = 1,2,\ldots,N$$

where ϵ_{rel}(= EPS) and $\epsilon_{\text{abs},i}$ are the local relative error tolerance and the local absolute error tolerance, respectively, for the variable i.

Convergence is said to occur if

$$d_m = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_{i,n}^{(m)} - y_{i,n}^{(m-1)}}{\text{EWT}_i} \right)^2}{N}} \leq c_L \qquad \text{(A12)}$$

In equations (A11) and (A12) the proportionality constants c_E and c_L are chosen to make the convergence tests consistent with the local truncation error. If convergence does not occur after the first iteration, both EPISODE and LSODE anticipate the magnitude of d_m one iteration in advance by assuming that the iterates converge linearly. Thus d_{m+1}, which does not yet exist, is estimated by

$$d_{m+1} = d_m \left(\frac{d_m}{d_{m-1}} \right)$$

If the corrector iteration fails to converge in three iterations, h_n is reduced if **P** is current and the step is retried; otherwise, **P** is updated and the step is retried. If the corrector converges after M ($M \leq 3$) iterations, an estimate of the truncation error is made and is accepted if it passes the following test:

EPISODE:

$$\sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_{i,n}^{(M)} - y_{i,n}^{(0)}}{YMAX_i} \right)^2}{N}} \leq t_E \text{EPS}$$

LSODE:

$$\sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_{i,n}^{(M)} - y_{i,n}^{(0)}}{EWT_i} \right)^2}{N}} \leq t_L$$

where t_E and t_L are test constants that depend on the order used. If the error test fails, the step is repeated with a different h_n or a lower order until either the test is passed or the situation is considered hopeless. If the test is passed, the step is accepted as successful, and the entire Nordsieck array is updated by

$$\mathbf{z}_n = \mathbf{z}_n^{(0)} + \underline{e}_n \underline{l}_n$$

where

$$\underline{e}_n = \underline{y}_n - \underline{y}_n^{(0)}$$

and the row vector $\underline{l}_n (l_{i,n}; i = 0,1,\ldots,q)$ is determined by the formulas used and satisfies $l_{0,n} = 1$ and $l_{1,n} = 1/\beta_{n,0}$. For EPISODE, $\underline{l}_n$ depends on the variable step length and is computed at the start of every step. For LSODE, $l_{i,n}$ are constants that are set at the beginning of the problem.

In summary, the predictor and corrector steps are given by

Predictor:

$$\mathbf{z}_n^{(0)} = \mathbf{z}_{n-1} \mathbf{A}(q)$$

Corrector:

$$\left.\begin{aligned} \mathbf{P}[\underline{y}_n^{(m+1)}] - \underline{y}_n^{(m)} &= \mathbf{P}[\Delta \underline{y}_n^{(m)}] \\ &= -\underline{G}[\underline{y}_n^{(m)}] \\ \underline{y}_n^{(m+1)} &= \underline{y}_n^{(m)} + \Delta \underline{y}_n^{(m)} \end{aligned}\right\} \quad m = 0,1,\ldots,M-1$$

$$\underline{y}_n = \underline{y}_n^{(M)} = \underline{y}_n^{(0)} + \underline{e}_n$$

$$\underline{e}_n = \sum_{m=0}^{M-1} \Delta \underline{y}_n^{(m)}$$

$$\mathbf{z}_n = \mathbf{z}_n^{(0)} + \underline{e}_n \underline{L}_n$$

CHEMEQ

In this technique, developed by Boris and Young (ref. 11), the rate equation (A2) is expressed as a difference between two positive-definite terms as follows:

$$\frac{dy_i}{dt} = f_i = Q_i - D_i \tag{A13}$$

where, for species i, the production rate Q_i and the destruction rate D_i can be derived from equation (3)

$$\left.\begin{aligned} Q_i &= \rho^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} R_{-j} + \nu''_{ij} R_j) \\ D_i &= \rho^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} R_j + \nu''_{ij} R_{-j}) \end{aligned}\right\} \tag{A14}$$

When temperature is an independent variable (method B), its rate equation (eq. (8)) can be cast in a similar form by combining equations (8) and (A13)

$$\frac{dy_{NS+1}}{dt} = \frac{dT}{dt} = \frac{-\sum_{k=1}^{NS} f_k \tilde{h}_k}{\sum_{k=1}^{NS} y_k c_{p,k}} = \frac{-\sum_{k=1}^{NS} (Q_k - D_k)\tilde{h}_k}{\sum_{k=1}^{NS} y_k c_{p,k}}$$

$$= Q_{NS+1} - D_{NS+1}$$

$$= Q_T - D_T$$

where

$$Q_T = \frac{\sum_{k=1}^{NS} D_k \tilde{h}_k}{\sum_{k=1}^{NS} y_k c_{p,k}}$$

and

$$D_T = \frac{\sum_{k=1}^{NS} Q_k \tilde{h}_k}{\sum_{k=1}^{NS} y_k c_{p,k}}$$

The objective of this decomposition is to enable factorization of y_i from D_i

$$D_i = L_i y_i = \frac{y_i}{\tau_i}$$

where L_i is obtained simply by dividing D_i by y (i.e., $L_i = D_i/y_i$). With this new notation, equation (A13) can be written as

$$\frac{dy_i}{dt} = Q_i - L_i y_i = Q_i - \frac{y_i}{\tau_i} \tag{A15}$$

which, for constant Q_i and L_i, can be solved to give

$$y_i(t_n) = y_i(t_{n-1} + h_n) = \frac{Q_i}{L_i} + \left[y_i(t_{n-1}) - \frac{Q_i}{L_i} \right] \times \exp(-L_i h_n) \tag{A16}$$

With this expression, it can be seen that $1/L_i (= \tau_i)$ describes how quickly the variable y_i reaches its equilibrium value.

In advancing the solution from time t_{n-1} to time t_n, all of the equations are separated into two classes, stiff and nonstiff, according to the criterion

$$\frac{h_n}{\tau_{i,n-1}} \begin{cases} \geq 1 & \text{stiff} \\ < 1 & \text{nonstiff} \end{cases}$$

where $\tau_{i,n-1}$ denotes the value of τ_i at time t_{n-1}. The two types of equations are integrated by separate predictor-corrector schemes. For equations classified as nonstiff, the improved Euler method—the Euler method as predictor and the modified Euler method (or trapezoidal rule) as corrector—is used. For equations classified as stiff a simple asymptotic formula is used.

Predictor (nonstiff):

$$y_{i,n}^{(0)} = y_{i,n-1} + h_n f_{i,n-1}$$

Predictor (stiff):

$$y_{i,n}^{(0)} = \frac{y_{i,n-1}(2\tau_{i,n-1} - h_n) + 2h_n \tau_{i,n-1} Q_{i,n-1}}{2\tau_{i,n-1} + h_n} \qquad \text{(A17)}$$

Corrector (nonstiff):

$$y_{i,n}^{(m+1)} = y_{i,n-1} + \frac{h_n}{2}\left[f_{i,n-1} + f_{i,n}^{(m)}\right]$$

Corrector (stiff):

$$y_{i,n}^{(m+1)} = \left\{ \frac{h_n}{2}\left[\tau_{i,n}^{(m)} + \tau_{i,n-1}\right]\left[Q_{i,n}^{(m)} + Q_{i,n-1}\right] + y_{i,n-1}\left[\tau_{i,n}^{(m)} + \tau_{i,n-1} + h_n\right]\right\} \Big/ \left[\tau_{i,n}^{(m)} + \tau_{i,n-1} + h_n\right] \qquad \text{(A18)}$$

In equations (A17) and (A18), $m+1$ is the current iteration number. The zeroth iterate is the result of the predictor step. Also, $f_{i,n}^{(m)} = f_i[y_{k,n}^{(m)}]$. Convergence is ascertained by comparing $y_{i,n}^{(m+1)}$ with $y_{i,n}^{(m)}$ for all N equations using the relative error criterion

$$\frac{\left|y_{i,n}^{(m+1)} - y_{i,n}^{(m)}\right|}{\min\left[\left|y_{i,n}^{(m)}\right|, \left|y_{i,n}^{(m+1)}\right|\right]} \leq \text{EPS} \qquad \text{(A19)}$$

If after ITMAX iterations any of the N variables fails the test given by equation (A19), the step length is halved and the step repeated. If all N variables pass the test after M iterations ($M \leq$ ITMAX), the step is accepted as successful and the solution is updated

$$y_{i,n} = y_{i,n}^{(M)} \qquad i = 1, N$$

CREK1D

In CREK1D, attention is paid to the distinguishing physical and computational characteristics of the induction, heat release, and equilibration regimes. This code consists of two algorithms developed for the two distinctly different regimes identified in the section Accuracy Comparisons. These regimes are (a) induction and early heat release, when the ODE's are dominated by positive time constants, and (b) late heat release and equilibration, when the ODE's are more stable. Both algorithms are based on an exponentially fitted trapezoidal rule, but they use different iterative methods for convergence.

In the CREK1D method the temperature is not treated as an additional independent variable, so the number of ODE's is equal to NS. The temperature is calculated from the algebraic energy conservation equation (eq. (7)). In the following discussion the variables y_i ($i = 1$,NS) therefore refer only to the species mole numbers.

The species rate expression f_i (the right side of eq. (A1)) can be expanded in a first-order-truncated Taylor's series about the current approximate solution, $[t_{n-1}, \underline{y}_{n-1}$ $(= y_{i,n-1};\ i = 1, \text{NS}), T_{n-1}]$, as follows:

$$f_i = f_{i,n-1} + \left(\frac{df_i}{dy_i}\right)_{\underline{y}_{n-1}} (y_i - y_{i,n-1}) \qquad \text{(A20)}$$

where df_i/dy_i is the total derivative of f_i with respect to y_i and is given by the chain rule as

$$\frac{df_i}{dy_i} = \sum_{k=1}^{\text{NS}} \frac{\partial f_i}{\partial y_k}\frac{dy_k/dt}{dy_i/dt} + \frac{\partial f_i}{\partial t}\frac{dT/dt}{dy_i/dt}$$

$$= \frac{1}{f_i}\left(\sum_{k=1}^{\text{NS}} \frac{\partial f_i}{\partial y_k} f_k + \frac{\partial f_i}{\partial T}\frac{dT}{dt}\right)$$

To avoid confusion with the partial derivative of f_i with respect to y_i ($\partial f_i/\partial y_i$), a special notation is employed

$$Z_i \equiv \frac{df_i}{dy_i}$$

With this new notation equation (A20) becomes

$$f_i = f_{i,n-1} + Z_{i,n-1}(y_i - y_{i,n-1}) \qquad \text{(A21)}$$

or

$$\frac{dy_i}{dt} = f_{i,n-1} + Z_{i,n-1}(y_i - y_{i,n-1})$$

which can be integrated to give

$$y_{i,n}=y_{i,n-1}+h_n f_{i,n-1}\left[\frac{\exp(h_n Z_{i,n-1})-1}{h_n Z_{i,n-1}}\right] \tag{A22}$$

Consider now, an approximate solution to equation (2) based on a variation of the second-order trapezoidal rule, the tunable trapezoid,

$$y_{i,n}=y_{i,n-1}+h_n[U_{i,n}f_{i,n}+(1-U_{i,n})f_{i,n-1}] \tag{A23}$$

The substitution of equation (A22) into equation (A23) gives

$$U_{i,n}=\frac{1}{h_n Z_{i,n-1}}+\frac{1}{1-\exp(h_n Z_{i,n-1})} \tag{A24}$$

In order to maintain absolute A-stability of equation (A23) (i.e., $y_{i,n}$ remains bounded as h_n is increased indefinitely), U_i must be restricted to the interval (0.5, 1.0). For values of $Z_i>0$, equation (A24) gives $U_i<0.5$. CREK1D resolves this problem by setting $Z_i=0$ whenever it is greater than zero. This gives $U_i=0.5$, so that equation (A23) defaults to the second-order-accurate trapezoidal rule. However, for $Z_i\leq 0$, equations (A23) and (A24) together are equivalent to the locally exact or exponential solution, which has an equivalent polynomial accuracy of order six to eight. Thus equations (A23) and (A24), with the constraint $(0.5<U_i<1)$, constitute an exponentially fitted trapezoidal rule, a method which is A-stable and has a polynomial-order accuracy of at least two and as great as six to eight.

At each integration step, equation (A23) must be solved for $y_{i,n}$. This is accomplished by Newton-Raphson iteration in regime b and Jacobi-Newton iteration in regime a.

A Newton-Raphson functional $F_{i,n}^{(m)}$ $(i=1,\text{NS})$ for the species mole numbers is defined from equation (A23) by

$$F_{i,n}^{(m)}=\frac{y_{i,n}^{(m)}-y_{i,n-1}}{h_n U_{i,n}}-\left(\frac{1-U_{i,n}}{U_{i,n}}\right)f_{i,n-1}-f_{i,n}^{(m)} \tag{A25}$$

For temperature the functional $F_{T,n}^{(m)}$ is defined from the enthalpy conservation equation (7) as

$$F_{T,n}^{(m)}=\sum_{k=1}^{\text{NS}} y_{k,n}^{(m)}\tilde{h}_k[T_n^{(m)}]-H_0(T_0) \tag{A26}$$

where m is the iteration number, $T_n^{(m)}$ is the m^{th}-approximation to the exact value $T(t_n)$, $\tilde{h}_k\ [T_n^{(m)}]$ is the molal-specific enthalpy of species k at temperature $T_n^{(m)}$, and H_0 (T_0) is the initial mixture mass-specific enthalpy at the initial temperature T_0.

Newton-Raphson corrector equations with log variable corrections (for self-scaling of the widely varying mole numbers) are given by

$$\left.\begin{aligned}&\sum_{k=1}^{\text{NS}}\frac{\partial F_{i,n}^{(m)}}{\partial\log y_{k,n}^{(m)}}\Delta\log y_{k,n}^{(m)}+\frac{\partial F_{i,n}^{(m)}}{\partial\log T_n^{(m)}}\Delta\log T_n^{(m)}=-F_{i,n}^{(m)}\\&\sum_{k=1}^{\text{NS}}\frac{\partial F_{T,n}^{(m)}}{\partial\log y_{k,n}^{(m)}}\Delta\log y_{k,n}^{(m)}+\frac{\partial F_{T,n}^{(m)}}{\partial\log T_n^{(m)}}\Delta\log T_n^{(m)}=-F_{T,n}^{(m)}\end{aligned}\right\} \tag{A27}$$

To start the iteration process, the predicted values $y_{i,n}^{(0)}$ and $T_n^{(0)}$ are obtained quite simply by setting them equal to the values at the previous time step

$$y_{i,n}^{(0)}=y_{i,n-1}\qquad i=1,\text{NS}$$

$$T_n^{(0)}=T_{n-1}$$

The Jacobi-Newton iteration technique can be derived from the Newton-Raphson iteration procedure by neglecting the off-diagonal elements of the Jacobian matrix $\partial f_i/\partial y_k$. With this simplification, equations (A27) reduce to

$$\frac{\partial F_{i,n}^{(m)}}{\partial\log y_{i,n}^{(m)}}\Delta\log y_{i,n}^{(m)}=-F_{i,n}^{(m)} \tag{A28}$$

$$\frac{\partial F_{T,n}^{(m)}}{\partial\log T_n^{(m)}}\Delta\log T_n^{(m)}=-F_{T,n}^{(m)} \tag{A29}$$

The iteration procedure is further simplified as follows: the expression for $\partial F_{i,n}^{(m)}/\partial\log y_{i,n}^{(m)}$, derived from equation (A25), is given by

$$\frac{\partial F_{i,n}^{(m)}}{\partial\log y_{i,n}^{(m)}}=\frac{y_{i,n}^{(m)}}{h_n U_{i,n}}-y_{i,n}^{(m)}\frac{\partial f_{i,n}^{(m)}}{\partial y_{i,n}^{(m)}}$$

In deriving an expression for $\partial f_{i,n}^{(m)}/\partial y_{i,n}^{(m)}$, the partial derivatives with respect to the inverse mean molar mass n_m are assumed to be negligible in comparison with the

other terms. This results in the following approximation for $\partial f_{i,n}^{(m)}/\partial y_{i,n}^{(m)}$ (see eq. (11) for the exact expression for $\partial f_{i,n}^{(m)}/\partial y_{i,n}^{(m)}$):

$$\frac{\partial f_{i,n}^{(m)}}{\partial y_{i,n}^{(m)}} = -\left[\rho y_{i,n}^{(m)}\right]^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} - \nu''_{ij})(\nu'_{ij} R_j - \nu''_{ij} R_{-j})$$

For Jacobi-Newton iteration, this expression is further approximated by

$$\frac{\partial f_{i,n}^{(m)}}{\partial y_{i,n}^{(m)}} = -\left[\rho y_{i,n}^{(m)}\right]^{-1} \sum_{j=1}^{JJ} (\nu'_{ij} R_j + \nu''_{ij} R_{-j})$$

which when combined with equation (A14) gives

$$\frac{\partial f_{i,n}^{(m)}}{\partial y_{i,n}^{(m)}} = -\frac{D_{i,n}^{(m)}}{y_{i,n}^{(m)}}$$

where $D_{i,n}^{(m)}$ is the destruction rate of species i. With these simplifications equation (A28) can be solved explicitly for the iterative corrections

$$\Delta \log y_{i,n}^{(m)} = -\frac{F_{i,n}^{(m)}}{y_{i,n}^{(m)}/h_n U_{i,n} + D_{i,n}^{(m)}}$$

From equation (A26) the following expression can be derived:

$$\frac{\partial F_{T,n}^{(m)}}{\partial \log T_n^{(m)}} = T_n^{(m)} \sum_{k=1}^{NS} y_{k,n}^{(m)} c_{p,k}\left[T_n^{(m)}\right]$$

where $c_{p,k}\left[T_n^{(m)}\right]$ is the constant pressure molal-specific heat of species k at temperature $T_n^{(m)}$. Substituting the preceding equation into equation (A29) gives

$$\Delta \log T_n^{(m)} = \frac{-F_{T,n}^{(m)}}{T_n^{(m)} \sum_{k=1}^{NS} y_{k,n}^{(m)} c_{p,k}\left[T_n^{(m)}\right]}$$

To start the iteration process, the predicted values $y_{i,n}^{(0)}$ are obtained from equation (A22)

$$y_{i,n}^{(0)} = y_{i,n-1} + h_n f_{i,n-1}\left[\frac{\exp(h_n Z_{i,n-1}) - 1}{h_n Z_{i,n-1}}\right]$$

The predicted temperature $T_n^{(0)}$ is obtained by a single Newton-Raphson iteration

$$T_n^{(0)} = T_{n-1} + \frac{H_0(T_0) - \sum_{k=1}^{NS} y_{k,n}^{(0)} \tilde{h}_k T_{n-1}}{\sum_{k=1}^{NS} y_{k,n}^{(0)} c_{p,k} T_{n-1}}$$

For both Newton-Raphson (NR) and Jacobi-Newton (JN) iteration schemes the current values $y_{i,n}^{(m+1)}$ and $T_n^{(m+1)}$ are updated by the approximate equations

$$\left.\begin{aligned} y_{i,n}^{(m+1)} &= y_{i,n}^{(m)}\left[1 + \Delta \log y_{i,n}^{(m)}\right] \\ T_n^{(m+1)} &= T_n^{(m)}\left[1 + \Delta \log T_n^{(m)}\right] \end{aligned}\right\} \qquad \text{(A30)}$$

The test for convergence of the iterates $y_{i,n}^{(m+1)}$ is based on the value $\Delta \log y_{i,n}^{(m)}$ and is given by

$$\sqrt{\frac{\sum_{i=1}^{NS}\left[\Delta \log y_{i,n}^{(m)}\right]^2}{NS}} \leq EPS \qquad \text{(A31)}$$

This test is used only with variables whose magnitudes are greater than 10^{-20}; that is, the summation does not include species with mole numbers less than or equal to 10^{-20}. If convergence is not obtained after ITMAX iterations, the step length is reduced and the step retried. If convergence is achieved in M iterations ($M \leq$ ITMAX), the step is accepted as successful and the solution is updated

$$y_{i,n} = y_{i,n}^{(M)} \qquad i = 1, NS$$

$$T_n = T_n^{(M)}$$

CREK1D automatically selects the iteration scheme (JN or NR) to be used for solving equation (A23). During induction and heat release, when small step lengths are required for solution stability, the JN iteration is used to minimize computational work. During late heat release and equilibration, when the ODE's are more stable and larger step lengths can be used, NR iteration is preferred since it has a much larger radius of convergence than JN iteration. The regime identification test exploits the fact that during equilibration many reactions achieve a condition in which the forward and reverse rates are large but have vanishingly small differences. The actual test employed at the beginning of each time step is

$$|f_i| \leq 10^{-3}(Q_i + D_i) \qquad \text{(A32)}$$

where Q_i and D_i are the production and destruction terms, respectively, for species i (eq. (A14)). If any of the species satisfies equation (A32), regime b is obtained and the NR iteration is used for the step. If none of the species satisfies equation (A32), regime a is obtained and the JN iteration is used for the step.

For this method much computation time can be involved in calculating Z_i because it requires the evaluation of the Jacobian matrix. To avoid evaluating this matrix, $Z_{i,n-1}$ is estimated as follows: if equation (A21) is applied to the entire time step h_n

$$f_{i,n} = f_{i,n-1} + Z_{i,n-1}(y_{i,n} - y_{i,n-1})$$

which, when substituted into equation (A22), gives

$$f_{i,n} = f_{i,n-1}\exp(h_n Z_{i,n-1})$$

or

$$Z_{i,n-1} = \frac{1}{h_n}\log\left(\frac{f_{i,n}}{f_{i,n-1}}\right)$$

However, $f_{i,n}$ is not known at the start of the step, so approximations have to be developed for $Z_{i,n-1}$. For the NR iteration, $Z_{i,n-1}$ is approximated by

$$Z_{i,n-1} = \frac{1}{h_n}\log\left(\frac{f_{i,n-1}}{f_{i,n-2}}\right) \qquad \text{for } 0 < \frac{f_{i,n-1}}{f_{i,n-2}} < 1$$

$$= 0 \qquad \text{otherwise}$$

For the JN iteration, $Z_{i,n-1}$ is approximated by

$$Z_{i,n-1} = \frac{1}{h_n}\frac{f_{i,n-1} - f_{i,n-2}}{f_{i,n-1}} \qquad \text{for } f_{i,n-1}f_{i,n-2} > 0 \text{ and } f_{i,n-1} \neq 0$$

$$= 0 \qquad \text{otherwise}$$

CREK1D also includes an algorithm for filtering the initial conditions that may be ill posed. These ill-posed conditions may arise, for example, in multidimensional modeling because of the averaging of mole numbers over adjacent grid nodes. CREK1D therefore "filters" the initial conditions to provide physically meaningful initial mole numbers and net species production rates. For purposes of this filtering CREK1D uses the decomposition performed in CHEMEQ (eqs. (A13) and (A15)). On the first call to CREK1D it uses this formulation over one time step of length h_1, which is determined by

$$h_1 = \frac{1}{\max_i L_i}$$

The predictor-corrector algorithm uses equation (A15) as the predictor

$$y_{i,1}^{(0)} = y_i(0) + h_1 f_i(0)\left\{\frac{1 - \exp[-h_1 L_i(0)]}{h_1 L_i(0)}\right\}$$

An implicit Euler corrector is then iterated to convergence

$$y_{i,1}^{(m+1)} = y_i(0) + h_1 f_{i,1}^{(m+1)}$$

In these equations $y_i(0)$ are the initial values and the subscript 1 is used to indicate that this is the first step. Using equations (A13), (A15), and (A30), together with the approximations $Q_{i,1}^{(m+1)} = Q_{i,1}^{(m)}$ and $L_{i,1}^{(m+1)} = L_{i,1}^{(m)}$, the preceding corrector equation can be rewritten to provide the following expression for the log variable corrections $\Delta \log y_{i,1}^{(m)}$:

$$\Delta \log y_{i,1}^{(m)} = \frac{y_i(0) - y_{i,1}^{(m)} + h_1 f_{i,1}^{(m)}}{y_{i,1}^{(m)} + h_1 D_{i,1}^{(m)}} \qquad \text{(A33)}$$

Equation (A33) is iterated until convergence, which is ascertained by the criterion given by equation (A31). If convergence is not obtained after 10 iterations, the step length is halved and the step retried. If convergence is obtained after M iterations ($M \leq 10$), the step is accepted as successful, the solution for the mole numbers is updated

$$y_{i,1} = y_{i,1}^{(M)} \qquad i = 1, \text{NS}$$

and the temperature T_1 is obtained by a single Newton-Raphson iteration

$$T_1 = T_0 + \frac{H_0(T_0) - \sum_{k=1}^{NS} y_{k,1}\tilde{h}_k(T_0)}{\sum_{k=1}^{NS} y_{k,1}c_{p,k}(T_0)}$$

GCKP84

GCKP84 is a general-purpose chemical kinetics program designed to solve a wide variety of problems (ref. 18). It uses the integration technique developed by Zeleznik and McBride (ref. 16) specifically to integrate chemical kinetic rate equations. Details of this integration technique are not yet available.

DASCRU

DASCRU is an explicit fourth-order Runge-Kutta process that differs from the standard Runge-Kutta method as follows: the standard Runge-Kutta method requires four derivative evaluations per step. The main disadvantage with its use is the difficulty of estimating the local truncation error at each step, since its formal expression is excessively complicated. The Runge-Kutta-Merson method requires an additional derivative computation that serves to determine the estimated local error. This technique uses the following equations to advance the solution from time t_{n-1} to time t_n (ref. 19):

$$y_{i,n}^{(1)} = y_{i,n-1} + \frac{h_n}{3} f_{i,n-1}$$

$$y_{i,n}^{(2)} = y_{i,n-1} + \frac{h_n}{6}\left[f_{i,n-1} + f_{i,n}^{(1)}\right]$$

$$y_{i,n}^{(3)} = y_{i,n-1} + \frac{h_n}{8}\left[f_{i,n-1} + 3f_{i,n}^{(2)}\right]$$

$$y_{i,n}^{(4)} = y_{i,n-1} + \frac{h_n}{2}\left[f_{i,n-1} - 3f_{i,n}^{(2)} + 4f_{i,n}^{(3)}\right]$$

$$y_{i,n}^{(5)} = y_{i,n-1} + \frac{h_n}{6}\left[f_{i,n-1} + 4f_{i,n}^{(3)} + f_{i,n}^{(4)}\right]$$

In these equations

$$f_{i,n}^{(j)} = f_i\left[y_{k,n}^{(j)}\right]$$

A good estimate of the local error in computed $y_{i,n}^{(5)}$, the accepted value for $y_{i,n}$, is $(y_{i,n}^{(4)} - y_{i,n}^{(5)})/5$, if the time step is sufficiently small (ref. 19). However, DASCRU uses the following test to ascertain convergence:

$$\left.\begin{aligned} &\left|y_{i,n}^{(4)} - y_{i,n}^{(5)}\right| < 0.5\,\mathrm{EPS} \qquad \text{for} \left|y_{i,n}^{(5)}\right| < 10^{-3} \\ &\left|\frac{y_{i,n}^{(4)} - y_{i,n}^{(5)}}{y_{i,n}^{(5)}}\right| < 0.5\,\mathrm{EPS} \qquad \text{for} \left|y_{i,n}^{(5)}\right| \geq 10^{-3} \end{aligned}\right\} \quad \text{(A34)}$$

where EPS is the local error tolerance. If any of the variables fails to satisfy the convergence criterion (eq. (A34)), the step length is halved and the step is retried. If all of the variables satisfy the convergence criterion (eq. (A34)), the step is accepted as successful and the solution is updated

$$y_{i,n} = y_{i,n}^{(5)}$$

Appendix B
User-Supplied Parameters

The codes (EPISODE, LSODE, CHEMEQ, CREK1D, GCKP84, and DASCRU) examined in this study require the specification of several parameters in addition to the local error tolerance EPS and the elapsed time at which the integration is to be terminated. For each code, values for the user-supplied parameters that minimized the CPU time required by the code were obtained by a trial-and-error procedure. The parameters required by each code are discussed below.

EPISODE

EPISODE requires the user to specify the method flag MF, the error control to be performed IERROR, and the guess for the initial step length H0. For both test problems all options (10, 11, 12, 13, 20, 21, 22, and 23) for MF were tried, and the stiff option with a user-supplied analytic evaluation of the complete Jacobian matrix (MF = 21) was found to be the most efficient. The type of error control to be performed is specified by the parameter IERROR, which has three possible values: 1, 2, and 3. For IERROR = 1, the error control is absolute; and for IERROR = 2, it is relative. In the test problems examined in this study, the variables differ widely (see tables IV and V), so relative error control is appropriate. However, since some of the mole numbers had zero initial values, pure relative error control could not be used. The option IERROR = 3 was used instead. This is a semirelative error control. It is relative for variables that are initially nonzero. For a variable that is initially zero, the error control is absolute until the variable reaches unity in magnitude, when the control becomes relative. Since none of the mole numbers reaches a value of unity, the error control is always absolute for species with initially zero mole numbers. The optimum value for H0 was found to be a function of the problem, the temperature method used, and the value for EPS. For test problem 2, values for H0 close to the optimum value resulted in vastly increased CPU times. This behavior is illustrated in table VIII. Note that a decrease in H0 from 10^{-7} to 10^{-8} s has resulted in an order-of-magnitude increase in the CPU time. Although not shown here, the solution was also found to be adversely affected by a poor choice for H0. Also, some values for H0 resulted in problems with solution instability.

LSODE

The user-specified parameters for LSODE include the method flag MF, the error control to be performed ITOL, and values for the local relative RTOL and absolute ATOL error tolerances. Both RTOL and ATOL can be specified either as (1) a scalar, so that the same error tolerance is used for all variables, or (2) an array, so that different values for the error tolerance can be used for different variables. The value of ITOL indicates whether RTOL and ATOL are scalars or arrays. ITOL has four possible values (1, 2, 3, and 4) which correspond to the types of RTOL and ATOL as follows:

ITOL = 1: scalar RTOL and scalar ATOL
ITOL = 2: scalar RTOL and array ATOL
ITOL = 3: array RTOL and scalar ATOL
ITOL = 4: array RTOL and array ATOL

The option ITOL = 2 (scalar RTOL and array ATOL) was used for the following reasons: since the same number of significant figures (given by RTOL) is acceptable for all components, RTOL was specified as a scalar. However, since the temperature has much larger values than the species mole numbers, ATOL can be much larger for the temperature than for the mole numbers. Hence ATOL was specified as an array.

For both test problems, the options (10, 11, 12, 13, 20, 21, 22, 23) for MF were tried and the method MF = 21 (backward differentiation method and user-supplied analytic evaluation of the complete Jacobian matrix) was found to be the fastest. For species mole numbers, values for the absolute error tolerances that resulted in minimum CPU times were found to be a function of the problem, the temperature method used, and the value of the relative error tolerance EPS. For values of EPS given in tables XI and XII for test problems 1 and 2, respectively, a value of zero for the absolute error tolerance for the temperature (required by LSODE-B) resulted in minimum CPU times for both test problems.

CHEMEQ

CHEMEQ requires the user to specify the maximum number of corrector iterations ITMAX to be attempted before nonconvergence is declared and a smaller step length tried. The optimum value for ITMAX was found to be 5 for both test problems. Several attempts at increasing the efficiency of CHEMEQ were made. These included (1) replacement of the Newton-Raphson iteration procedure for the temperature (method A) with a single Newton-Raphson iteration and (2) limiting the temperature change per time step. If the temperature change during any time step exceeded the maximum permitted, the step was shortened accordingly and retried.

Modification (1) resulted in less efficiency and more errors and was therefore abandoned. Test (2) was applied

(a) after each predictor step, (b) after each corrector step, and (c) after each predictor step and after each corrector step. For test problem 1 and temperature method A, option c was found to be the most efficient.

Various values for the maximum temperature change permitted per time step were tried, and the optimum value was in the range 7 to 8 kelvins for test problem 1. Although option c was found to be the best among the three options (a-c), it was only marginally faster than the basic algorithm (with no constraint on the temperature change per time step). Also, for test problem 2 with temperature method A, and for both test problems with temperature method B, none of the three options was more efficient than the basic algorithm. In addition, when the reaction rates were not continuously calculated but were updated only after an allowed temperature change ΔT, the basic method was found to be the most efficient. Hence modification 2 was also rejected.

CREK1D

The user-supplied parameters to CREK1D include the maximum number of corrector iterations ITMAX allowed before nonconvergence is declared; and the maximum temperature change DELT allowed before the thermodynamic properties h_i and $c_{p,i}$ ($i = 1$,NS) and the rate constants k_j [$= A_j T^{N_j} \exp(-T_j/T)$] and T_{-j}, N_{-j}, and k_{-j} [$= A_{-j} T^{N_{-j}} \exp(-T_{-j}/T)$] are updated. Note that T_{-j}, N_{-j}, and k_{-j} are calculated from the forward rate constants and the equilibrium constant. See Pratt and Wormeck (ref. 27) for details. The optimal value for ITMAX was in the range 5 to 7, depending on the test problem and the value of the relative error tolerance. For all runs presented in this report, a value of ITMAX = 10 was used. The optimal value for DELT was in the range 1 to 2 kelvins depending on the problem and on the value of the relative error tolerance.

GCKP84

Since details for this technique are not yet available, default values for all parameters were used.

DASCRU

This code requires the user to specify the guess for the initial step length H0. The optimum value for H0 was found to a function of the problem, the temperature method used, and the error tolerance required of the solution.

Appendix C

Approximation for Maximum Temperature Change Between Rate Constant Updates

As discussed in the section Computational Tactics, significant increases in efficiency are realized by updating the forward rate constants k_j [$= A_j T^{N_j} \exp(-T_j/T)$] and the backward rate constants T_{-j}, N_{-j}, and k_{-j} [$= A_{-j} T^{N_{-j}} \exp(-T_{-j}/T)$] only when the temperature change exceeds an amount ΔT that is problem dependent. In specifying a value for ΔT, care must be taken to avoid poor approximations in the resulting reaction rates because this leads to excessive computational work, as shown, for example, in table IX. To avoid the work associated with a trial-and-error search for an optimum value for ΔT, we now develop a simple approximation for it.

Consider rate constant k_j for reaction j. The exact expression for k_j is given by

$$k_j = A_j T^{N_j} \exp \frac{-T_j}{T} \tag{C1}$$

where A_j is the preexponential constant, T_j is the activation temperature ($= E_j/R$, where E_j is the activation energy and R the universal gas constant), and T is the current value of the temperature.

The idea behind using ΔT is that the work associated with computing k_j and the reverse rate constants (T_{-j}, N_{-j}, and k_{-j}) from the forward rate constants and the equilibrium constant is eliminated for changes in T not greater than ΔT. Hence, if the temperature changes from its current value of T by an amount not greater than ΔT, the rate constants k_j and k_{-j} are not reevaluated. The poorest approximation ($k_{j,\text{approx}}$) for the new rate constant is therefore given by equation (C1) as follows:

$$k_{j,\text{approx}} = A_j T^{N_j} \exp\left(\frac{-T_j}{T}\right) \tag{C2}$$

The exact expression for k_j ($k_{j,\text{exact}}$) is given by

$$k_{j,\text{exact}} = A_j (T \pm \Delta T)^{N_j} \exp\left(\frac{-T_j}{T \pm \Delta T}\right) \tag{C3}$$

where the plus sign denotes increasing temperature and the minus sign denotes decreasing temperature.

We now estimate the maximum allowable ΔT by limiting the relative error in the resulting reaction rate to be no greater than EPS, the error tolerance required of the solution,

$$\left| 1 - \frac{k_{j,\text{approx}}}{k_{j,\text{exact}}} \right| \leq \text{EPS} \tag{C4}$$

where the bars | | denote absolute value.

The division of equation (C2) by equation (C3) and rearrangement give

$$\frac{k_{j,\text{approx}}}{k_{j,\text{exact}}} = \frac{\left[\exp\left(\frac{-T_j}{T}\right)\right]^{1 - \frac{1}{1 \pm \epsilon_T}}}{(1 \pm \epsilon_T)^{N_j}} \tag{C5}$$

where $\epsilon_T = \Delta T/T$, the positive sign denotes increasing temperature, and the negative sign denotes decreasing temperature.

There are four possible cases (other than the trivial one $T_j = N_j = 0$) as follows:

(1) For increasing T

(a) $\left(\frac{T_j}{T} + N_j\right) > 0$

(b) $\left(\frac{T_j}{T} + N_j\right) < 0$

(2) For decreasing T

(a) $\left(\frac{T_j}{T} + N_j\right) > 0$

(b) $\left(\frac{T_j}{T} + N_j\right) < 0$

Consider case a. For increasing T and positive $(T_j/T + N_j)$, inequality (C4) becomes

$$\frac{\left[\exp\frac{-T_j}{T}\right]^{1 - \frac{1}{1+\epsilon_T}}}{(1+\epsilon_T)^{N_j}} \geq 1 - \text{EPS} \tag{C6}$$

For small EPS and ϵ_T, $\exp(-\text{EPS}) \cong 1 - \text{EPS}$, $\exp(\epsilon_T) \cong 1 + \epsilon_T$, and $1/(1+\epsilon_T) \cong 1 - \epsilon_T$, so that equation (C6) becomes

$$\exp\left(-\epsilon_T \frac{T_j}{T}\right) \exp\left(-\epsilon_T N_j\right) \geq \exp\left(-\text{EPS}\right)$$

or

$$-\epsilon_T\left(\frac{T_j}{T} + N_j\right) \geq -\text{EPS}$$

or

$$\epsilon_T \leq \frac{\text{EPS}}{\frac{T_j}{T} + N_j} \qquad \text{for } \epsilon_T, \text{ EPS} << 1; \quad \left(T_j/T + N_j\right) \neq 0$$

or

$$\Delta T \leq \frac{\text{EPS } T}{\frac{T_j}{T} + N_j}$$

Similar analyses for the other three cases lead to similar expressions for ΔT. These four expressions can be replaced by a single expression given by

$$\left|\Delta T\right|_{\max} = \frac{\text{EPS } T}{\left|\frac{T_j}{T} + N_j\right|} \tag{C7}$$

Equation (C7) gives the maximum value of ΔT that can be used without resulting in a relative error greater than EPS in the forward reaction rate R_j for reaction j. For each forward and reverse reaction a value for the maximum allowable ΔT can be calculated by using equation (C7). Using the minimum of these values will ensure that no forward or reverse reaction rate will have a relative error greater than EPS. The minimum value for ΔT over all forward and reverse reaction rates is given by

$$\Delta T = \min_j \left\{ \frac{\text{EPS } T}{\left|\frac{T_j}{T} + N_j\right|; \left|\frac{T_{-j}}{T} + N_{-j}\right|} \right\}$$

$$= \frac{\text{EPS } T}{\max_j \left[\left|\frac{T_j}{T} + N_j\right|; \left|\frac{T_{-j}}{T} + N_{-j}\right|\right]} \tag{C8}$$

Equation (C8) provides a simple expression for the automatic evaluation of ΔT throughout the history of the problem. Every time the rate constants k_j, T_{-j}, N_{-j}, and k_{-j} are evaluated, which occurs only when the temperature change since the last update of the rate constants exceeds ΔT, a new value for ΔT can be calculated by using equation (C8). Using this equation therefore avoids the work associated with a trial-and-error search for the optimum value of ΔT.

References

1. Shampine, L.F.: Type-Insensitive ODE Codes Based on Implicit A-Stable Formulas. Mathematics of Computation, vol. 36, no. 154, Apr. 1981, pp. 499-510.
2. Shampine, L.F.: What is Stiffness? Presented at the International Conference on Stiff Computation (Park City, Utah), Apr. 12-14, 1982.
3. Lomax, H.; and Bailey, H.E.: A Critical Analysis of Various Numerical Integration Methods for Computing the Flow of a Gas in Chemical Nonequilibrium. NASA TN D-4109, 1967.
4. Seinfeld, J.H.; Lapidus, L.; and Hwang, M.: Review of Numerical Integration Techniques for Stiff Ordinary Differential Equations. Ind. Eng. Chem. Fundam., vol. 9, May 1970, pp. 266-275.
5. Enright, W.H.; and Hull, T.E.: Comparing Numerical Methods for the Solution of Stiff Systems of ODEs Arising in Chemistry. Numerical Methods for Differential Systems, L. Lapidus and W.E. Schiesser, Eds., Academic Press Inc., 1976, pp. 45-66.
6. Shampine, L.F.; and Gear, C.W.: A User's View of Solving Stiff Ordinary Differential Equations, SIAM Rev., vol. 21, no. 1, July 1979, pp. 1-17.
7. Hindmarsh, A.C.; and Byrne, G.D.: Applications of EPISODE: An Experimental Package for the Integration of Systems of Ordinary Differential Equations. Numerical Methods for Differential Systems, L. Lapidus and W.E. Schiesser, Eds., Academic Press Inc., 1976, pp. 147-166.
8. Hindmarsh, A.C.; and Byrne, G.D.: EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations. UCID-30112, Lawrence Livermore Laboratory, Rev. 1, 1977.
9. 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.
10. Hindmarsh, A.C.: ODEPACK: A Systematized Collection of ODE Solvers. UCRL-88007, Lawrence Livermore Laboratory, 1982.
11. 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.
12. Pratt, D.T.: New Computational Algorithms for Chemical Kinetics. NBS Spec. Publ. 561, 1979, pp. 1265-1279.
13. Pratt, D.T. and Radhakrishnan, K.: Fast Algorithms for Combustion Kinetic Calculations. Presented at the Fall Meeting of the Western States Section of the Combustion Institute, Oct. 20, 1981, Paper WSCI 81-46.
14. Pratt, D.T.: Fast Algorithms for Combustion Kinetic Calculations. Presented at the International Conference on Stiff Computation (Park City, Utah), Apr. 12-14, 1982.
15. Pratt, D.T.: CREK-1D: A Computer Code for Transient, Gas-Phase Combustion Kinetics. Presented at the Spring Meeting of the Western States Section of the Combustion Institute, Apr. 11-12, 1983, Paper WSCI 83-21.
16. Zeleznik, F.J.; and McBride, B.J.: Modeling the Internal Combustion Engine. NASA RP-1094, 1984.
17. Pratt, G.L.: Gas Kinetics. Wiley, 1969.
18. Bittker, D.A.; and Scullin, V.J.: GCKP84—General Chemical Kinetics Code for Gas Phase Flow and Batch Processes Including Heat Transfer. NASA TP-2320, 1984.
19. Fox, L., ed.: Numerical Solution of Ordinary and Partial Differential Equations. Addison-Wesley, 1962.
20. Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, 1971.
21. Gear, C.W.: The Automatic Integration of Ordinary Differential Equations. Commun. ACM, vol. 14, no. 3, Mar. 1971, pp. 176-190.
22. Liniger, W.; and Willoughby, R.A.: Efficient Integration Methods for Stiff Systems of Ordinary Differential Equations, SIAM Journal on Numerical Analysis, vol. 7, Mar. 1970, pp. 47-66.
23. Brandon, D.M., Jr.: A New Single-Step Integration Algorithm with A-Stability and Improved Accuracy. Simulation, vol. 23, no. 1, July 1974, pp. 17-29.
24. Babcock, P.D.; Stutzman, L.F.; and Brandon, D.M., Jr.: Improvements in a Single-Step Integration Algorithm. Simulation, vol. 33, no. 1, July 1979, pp. 1-10.
25. Bittker, D.A.; and Scullin, V.J.: General Chemical Kinetics Computer Program for Static and Flow Reactions, with Application to Combustion and Shock-Tube Kinetics. NASA TN D-6586, 1972.
26. Pratt, D.T.; and Wormeck, J.J.: CREK—A Computer Program for Calculation of Combustion Reaction Equilibrium and Kinetics in Laminar or Turbulent Flow. WSU-ME-TEL-76-1, Department of Mechanical Engineering, Washington State University, 1976.
27. Byrne, G.D.; et al.: Comparative Test Results for Two ODE Solvers: EPISODE and GEAR. ANL-77-19, Argonne National Laboratory, 1977.
28. Byrne, G.D.; et al.: A Comparison of Two ODE Codes: GEAR and EPISODE. Comput. Chem. Eng., vol. 1, no. 2, 1977, pp. 133-147.
29. Petzold, L.R.: Multistep Methods: An Overview of Methods, Codes, and New Developments. SAND83-8673, Sandia National Laboratories, 1983.
30. Radhakrishnan, K.: A Comparison of the Efficiency of Numerical Methods for Integrating Chemical Kinetic Rate Equations. NASA TM-83590, 1984.
31. Radhakrishnan, K.: Fast Algorithms for Combustion Kinetics Calculations: A Comparison. Combustion Fundamentals Research. NASA CP-2309, 1984. pp. 257-267.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TP-2372		

4. Title and Subtitle	5. Report Date
Comparison of Numerical Techniques for Integration of Stiff Ordinary Differential Equations Arising in Combustion Chemistry	October 1984
	6. Performing Organization Code: 505-31-42

7. Author(s)	8. Performing Organization Report No.
Krishnan Radhakrishnan	E-2149
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	
	13. Type of Report and Period Covered: Technical Paper

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546	

15. Supplementary Notes

Krishnan Radhakrishnan, University of Michigan, Ann Arbor, Michigan (work partially funded by NASA grants NAG3-147 and NAG3-294), and National Research Council - NASA Research Associate.

16. Abstract

The efficiency and accuracy of several algorithms recently developed for the efficient numerical integration of stiff ordinary differential equations are compared. 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 interactive solution of the algebraic energy conservation equation to compute the temperature does not result in significant errors. In addition, this method can be more efficient than evaluating the temperature by integrating its time derivative. Significant reductions in computational work are realized by updating the rate constants ($k = AT^N \exp(-E/RT)$) only when the temperature change exceeds an amount ΔT that is problem dependent. An approximate expression for the automatic evaluation of ΔT is derived and is shown to result in increased efficiency.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Chemical kinetics; Stiff ordinary differential equations; Numerical solution	Unclassified - unlimited STAR Category 07

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	40	A03

*For sale by the National Technical Information Service, Springfield, Virginia 22161 NASA-Langley, 1984

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

THIRD-CLASS BULK RATE

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

NASA

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return