Preliminary Investigations of Superdetonative and Transdetonative Ram Accelerator Propulsion Modes

by

Alan Kull

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Engineering

University of Washington

1990

Approved by _____
(Chairperson of Supervisory Committee)

Program Authorized
to Offer Degree _____

Date _____

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Preliminary Investigations of Superdetonative and Transdetonative Ram Accelerator Propulsion Modes

by Alan Kull

Chairperson of the Supervisory Committee: Professor Abraham Hertzberg
Department of Aeronautics and
Astronautics

The results of high-velocity experiments with a ram accelerator are presented. Also included is a computer program that models finite rate chemistry, an important phenomenon in high-velocity ram accelerator operation. The ram accelerator is a ramjet-in-tube projectile accelerator whose principle of operation is similar to that of a supersonic airbreathing ramjet. The projectile resembles the centerbody of a ramjet and travels through a tube filled with a premixed gaseous fuel and oxidizer mixture. The tube acts as the outer cowling of the ramjet, and the combustion process travels with the projectile, generating a pressure field which produces forward thrust on the projectile. Several different modes of combustion are possible. Subsonic, thermally choked combustion theoretically allows a projectile to be accelerated to the Chapman-Jouguet (C-J) detonation velocity of a particular gas mixture. In the superdetonative regime the projectile is accelerated while always traveling faster than the C-J velocity, and in the transdetonative regime (85-115% of C-J speed) the projectile may transition smoothly from a subsonic to a superdetonative combustion mode. Experimental confirmation of superdetonative ram accelerator operation is demonstrated, in which 70 gm projectiles are accelerated through a 12.2 m long, 38 mm bore accelerator tube to velocities near 2500 m/s. Experimental evidence of acceleration in the transdetonative regime is also introduced.

The chemical kinetics model solves a general hydrocarbon / oxygen combustion reaction mechanism and is applied to several cases. The kinetics model provides a tool to investigate the effects of different propellant gas chemistries in all propulsion modes.

Table of Contents

List of Figures

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Abraham Hertzberg and Professor Adam P. Bruckner for the guidance, motivation, and salary they have given me throughout my graduate studies at the University of Washington.

I am also indebted to my fellow ram accelerator colleagues and graduate students. Without them, the ram accelerator would be only an idea, and my memories less rich. These people are, in chronological order, Carl Knowlen, Alfred "Violence" Alvares, Edward Burnham, Scott O'Brien, Kelley Scott, Dai Murakami, Peter Kaloupis, Gilbert "Bah" Chew, Erik "Chris" Christofferson, Andrew Berschauer, Amy Prochko, Barbrina Dunmire, Jackie Auzias de Turenne, and Robert McIntosh.

Special thanks also go to Linda Tait and Jack Jinneman who somehow tolerated occupying living quarters with me. Finally, I cannot express the deep feelings I have for the Apple Computer Corporation whose fine product, the Macintosh IIx, decided to kill most of my thesis four days before it was due. The resulting document is orders of magnitude leaner than it should be. Whenever I am asked about dependability, I will always remember Apple.

DEDICATION

To my parents, whose love and support are invaluable

Chapter 1

Introduction

At the University of Washington experimental and theoretical research is being carried out on the acceleration of projectiles to high velocities using a ramjet-in-tube concept called the "ram accelerator." [1-7] The projectile resembles the centerbody of a conventional ramjet and is accelerated by combustion through a stationary tube filled with a reactive gas mixture . There is no propellant on board the projectile. The pressure, composition, chemical energy density, and speed of sound of the mixture can be controlled to optimize the ballistic efficiency (defined here as the ratio of the rate of change of kinetic energy of the projectile to the rate of expenditure of chemical energy). The concept has the potential for a number of applications, such as hypervelocity impact studies [8] direct launch to orbit of acceleration-insensitive payloads [9,10] and hypersonic testing.

Chapter 2

Experimental Investigations

Several modes of ram accelerator operation, which span the velocity range of 0.7-12 km/sec, have been proposed [1]. These include, among others, a thermally choked subsonic combustion mode, shown in Fig. 2.1, and two oblique detonation modes, one of which is shown in Fig. 2.2. The thermally choked subsonic combustion mode has been extensively studied experimentally [1-5], and has attained velocities in excess of 2500 m/sec with 70 gm projectiles in a 12.2 m long 38 mm bore accelerator tube. In the oblique detonation modes the gasdynamic principles of the ram accelerator are similar to those of the Oblique Detonation Wave Engine (ODWE) [11,12], which has been proposed as an alternative to the scramjet engine for propelling hypersonic airbreathing vehicles such as the National Aerospace Plane (NASP).

To operate in the oblique detonation modes the projectile must fly at superdetonative speeds, i.e., speeds above the local Chapman-Jouguet (C-J) detonation speed of the gas mixture. The cone angle of the nose, the projectile velocity, and the speed of sound of the mixture are selected so that the first conical shock does not initiate combustion, but the reflected shock does. In operation the reflected shock becomes an oblique detonation wave. Theoretical studies by the authors and their colleagues have indicated that by judicious selection of gas mixtures the ram accelerator will operate superdetonatively in the velocity range 2-12 km/sec [3,7]. Experimental results are presented for superdetonative ram accelerator operation in the velocity range 2000-2500 m/sec, in an ethylene-based propellant mixture having a C-J detonation speed of 1650 m/sec [13].

In addition it has been experimentally observed that while operating in the subdetonative regime, at velocities greater than 85% of C-J speed, the projectile often accelerates at a higher rate than is predicted by theoretical models for subsonic, thermally choked combustion. Further experiments have shown that in this transdetonative regime (85-115% of C-J speed), the projectile can accelerate smoothly from subdetonative to superdetonative speeds in a single gas mixture. Transdetonative behavior has been

observed in four different propellant gas mixtures [14]. These results may have interesting implications for any ramjet engine which must operate over a wide range of Mach numbers.

2.1 Ram Accelerator Experimental Facility

The ram accelerator facility (Fig. 2.3) consists of a light gas gun, ram accelerator section, final dump tank and projectile decelerator. The 38 mm bore, single-stage light gas gun is capable of accelerating the sabot and projectile combination (typical combined mas ~60-100 gm) to speeds up to approximately 1350 m/s. The muzzle of the gun is connected to a perforated-wall tube that passes through an evacuated tank which serves as a dump for the helium drive gas.

The ram accelerator section consists of seven steel tubes with a bore of 38 mm, and O.D. of 100 mm and an overall length of 12.2 m. There are a total of 32 pairs of diametrically opposed instrumentation ports disposed at 28 regular intervals along the accelerator tube. At four axial stations there are two pairs of opposing ports at right angles to each other, permitting the simultaneous use of four transducers. Piezoelectric pressure transducers are located at each of up to 20 observation ports. The remaining ports are used to mount electromagnetic velocity transducers (copper wire coiled around a stainless steel core) and fiber-optic light guides. A 20-channel, 1 MHz digital data acquisition system (DAS) is used to process the data. Multiplexing permits processing the 50 separate input signals currently being monitored.

The ram accelerator tube is designed to operate at propellant fill pressures up to 50 atm. Thin Mylar diaphragms are used to close off each end and to separate sections of the tube filled with different propellant mixtures. The fuel, oxidizer, and diluent gases are metered using sonic orifices, combined in a mixing chamber and directed to the appropriate section of the ram accelerator tube.

The end of the accelerator tube is connected be a 0.76 m long drift tube of equal bore to a 2.4 m long evacuated dump tank, where the projectile flies free. The tank has a pair of 25 cm dia. viewing ports for spark photography and two pairs of smaller ports for a two-beam laser velocity measuring system (Fig. 2.3). The free-flying projectile is brought to a stop in tightly packed rug remnants in an 18 cm I.D. x 1.8 m long tube attached to the far end of the dump tank.

2.2 Projectile Configuration

The basic projectile configuration that has been used in the majority of the experimental work to date is illustrated in Fig. 2.4. The projectile is fabricated from magnesium in two separate units: the nose cone and the body with integral fins. The nose and body are hollow. The fins serve only to center the projectile in the tube and the octagonal cross section of the body is simply a machining convenience. At the threaded joint between the nose and body is sandwiched a thin sheet of flexible magnetic material. A second magnet is affixed to the interior of the projectile at its base. These magnets interact with the electromagnetic transducers, providing time of flight data from which the velocity history of the projectile can be determined. The projectile has a maximum diameter of 28.9 mm. In the 38 mm bore tube the resulting diffuser has a flow area ratio of 2.37.

Two variations of the basic projectile configuration have been experimented with extensively and are referred to in Fig. 2.4 as type A and type B. The differences between the two projectile geometries used lie in the angle of the nose (10° and 12.5°) and the length of the body (71 mm and 84 mm). The longer body is used with the 10° nose. The masses of the projectiles used have been in the range of 45 to 70 gm, depending on the external and internal configurations. The lexan launching sabot has a mass of 13 gm.

2.3 Experimental Results

In the current experiments the first 8.5 m of tube are configured into a three-stage thermally choked ram accelerator which accelerates the 70 gm projectile to the 2000-2200 m/sec range, using methane-based propellant mixtures at nominal fill pressures of 25-30 atm, as described in Refs. 4 and 5. The last 3.66 m of the accelerator are filled with a mixture of $0.9C_2H_4 + 3O_2 + 5CO_2$ at 15-20 atm, which has an experimentally measured C-J detonation speed of 1650 m/sec (theoretical C-J speed is 1550 m/sec, based on equilibrium combustion). The projectile thus enters the final mixture at a velocity 20-30% higher than the C-J speed.

2.3.1 Thermally Choked Mode

Figure 2.5 illustrates typical transducer outputs obtained in the thermally choked combustion mode (subdetonative regime) in a tube segment containing $3.5CH_4 + 2O_2 + 6.5He$ at 25 atm. The projectile velocity and Mach number are 2020 m/s and 3.7, respectively. Time is measured from the instant of data acquisition system triggering, and pressure is measured in atmospheres. The pressure (middle) trace is typical of the thermally choked mode. The first pressure pulse is generated by the oblique shock system in the projectile diffuser section. There then follows a series of pulses which increase the pressure to a peak of approximately 430 atm, after which the pressure decays. The increase in pressure after the initial oblique shocks represents the normal shock, which is assumed to consist of a complex system of oblique and normal shocks similar to that observed in supersonic flow in long ducts [15]. The flow entering the combustion zone is subsonic. The decay in pressure following the peak is due to subsonic heat addition accelerating the flow to choking and the subsequent nonsteady expansion of the combustion products behind the choking point.

The upper trace in Fig. 2.5 displays the output of an electromagnetic transducer located at the same axial station as the pressure transducer. The zero crossing of the first signal identifies the passage by the sensor of the annular magnetic disk located at the projectile throat. The later zero crossing identifies the rear of the projectile. These signals provide convenient reference points from which the position of the shock system on the projectile can be determined. A profile of the projectile scaled to the local velocity is shown for reference.

The bottom trace in Fig. 2.5 shows the output from a fiber-optic probe located at the same station as the pressure and electromagnetic probes. The fiber-optic probes are used to examine the luminosity emitted as the projectile and combustion gases pass by the the probe. The primary radiation is assumed to be that from carbon particles generated by the fuel-rich combustion of methane and oxygen in the subsonic zone behind the projectile. The carbon particles emit blackbody radiation whose peak intensity occurs at the highest gas temperature.

2.3.2 Superdetonative Regime

Figure 2.6 displays the outputs from a pressure transducer and an electromagnetic sensor at a point 0.5 m from the entrance to the last stage, where the projectile is operating

in the superdetonative regime. The propellant mixture consists of $0.9C_2H_4 + 3O_2 + 5CO_2$ at a fill pressure of 16 atm. The projectile velocity and Mach number are 2070 m/sec and 7.1, respectively. The pressure trace, typical of superdetonative operation, is completely different from that observed in the thermally choked mode. In the present case there is an abrupt rise in pressure to 800 atm, i.e., a pressure ratio of 50, followed by a series of pressure pulses of decreasing amplitude. Eventually, a steady pressure plateau of 500 atm is reached.

Figure 2.7 displays the outputs from a pressure transducer and a light emission probe located 0.3 m ahead of the instruments in Fig. 2.6; the data in Fig. 2.7 are taken from the same experimental run as those in Fig. 6. The projectile velocity and Mach number are 2040 m/sec and 7.0, respectively. The features of the pressure trace are similar to those in Fig. 2.6. The light emission data are radically different from those in the thermally choked mode (Fig. 2.5). Along with the pressure traces, the light emission data suggest that combustion occurs mainly on the projectile body in contrast to the thermally choked mode, where all combustion activity occurs behind the projectile. The light emission behind the projectile in Fig. 2.7 may be a result of recombination or the formation of carbon particles. Currently, the exact combustion mode which drives the projectile in the superdetonative regime remains somewhat speculative, though recent CFD modeling indicates that shock-induced combustion may be the thrust-producing mechanism [6,7]. Regardless of the exact mechanism, the gas pressure is seen to rise during the combustion process, indicating supersonic heat addition.

The velocity of the projectile down the tube was deduced from the distance-time history of the electromagnetic transducer signals. The data obtained are curve fit with the highest order polynomial (typically fifth- to seventh-order) that closely matches the experimental data without producing excessive oscillations in the distance-velocity history obtained by differentiation. Figure 2.8 shows the experimentally determined velocity as a function of distance in the entire ram accelerator, including the first three stages of thermally choked operation. The solid line is the theoretical velocity-distance curve for the corresponding experimental run. The theoretical curve is plotted only for the subdetonative, thermally choked combustion regime. A model of superdetonative operation is the subject of current work. The operating conditions and the gas mixtures are noted in the figure. The short, dashed horizontal lines denote the C-J detonation speeds of the various gas mixtures. In this case superdetonative operation was observed over the

velocity range 2180-2475 m/sec. The peak Mach number attained in the superdetonative regime was 8.4, which, as far as the author knows, exceeds the performance of any ramjet published in the open literature.

2.3.3 Transdetonative Regime

In the higher Mach number ranges of the thermally choked mode (typically 4-4.5), it can be seen from the velocity vs. distance curve of Fig. 2.8 that the experimentally measured velocities remain higher than theory predicts, near the end of the third mixture. The theoretical model, described in detail elsewhere [1], is based on quasi-steady flow and predicts that the thrust on the projectile decreases as the projectile approaches the C-J detonation speed of the gas. This model further assumes that heat addition occurs only in the subsonic zone behind the projectile.

Accelerations much greater than that predicted by the thermally choked model are routinely observed when the projectiles attain velocities greater than 85% of the C-J speed of the propellant mixture. When close to the detonation velocity, the pressure waves on the rear half of the projectile often sweep forward through the projectile throat and unstart it. During this transient shock system activity the projectile velocity and acceleration increase abruptly before it unstarts. This behavior could be due to the initiation phase of a detonation wave at the rear of the projectile which gives it a boost before the wave completely washes over.

It was found that longer projectiles more closely approached the experimentally determined detonation speeds of the thermally choked mode propellant gases [5] and in some cases actually accelerated through and above the C-J detonation velocity. It is postulated that the frequent discrepancy between theory and experiment in this transdetonative regime (85%-115% of C-J speed) may be explained by different modes of propulsion which do not require a thermal choking point at full tube area to stabilize the driving pressure wave system on the projectile. Heat addition is believed to occur at least partially on the projectile body. Credence is given to this hypothesis by data from the light fiber probes showing luminosity emanating from the rear half of the projectile at the high Mach number ranges of thermally choked operation.

Transdetonative propulsion suggests that it may be possible for projectiles to transition smoothly from thermally choked to superdetonative operation in one mixture. Figure 2.9 is a velocity versus distance plot of an experiment wherein the projectile entered a mixture of $4.5CH_4 + 2O_2 + 2He$ at a speed of 1300 m/sec (Mach 2.8) and accelerated to 2250 m/s (Mach 5.0). The projectile had a mass of 65 grams and the tube fill pressure was 25 atm. The experimentally determined C-J speed for the mixture is 2050 m/s. Again, the solid line is the theoretical profile for the experiment. It shows good agreement with experiment up to about 85% of C-J speed, after which the experimental results outpace theory. The projectile accelerated through the C-J detonation speed, exceeding it by 10%. Similar results have been obtained in argon-diluted mixtures with C-J speeds in the 1600 m/s range.

A pressure and an electromagnetic transducer trace from the transdetonative regime are displayed in Fig. 2.10. These data were taken from the same experimental run plotted in Fig. 2.9. The projectile velocity and Mach number were 2150 m/s and 4.8, respectively. Although thermally choked, subdetonative theory predicts that the projectile loses thrust as it approaches the C-J speed due to the shock wave moving back on the body, the figure clearly shows that the shock system is well-attached to the projectile body. Also because of the higher Mach number, the initial oblique shock system is narrower and much stronger than that of Fig. 2.5.

The details of propulsion in the transdetonative regime remain speculative and are the subject of current research at the University of Washington. The exact mechanism by which heat is released during transdetonative operation is believed to be a "combined cycle" in which some heat is released on the projectile body and some in the recirculation zone behind it. The heat released on the body may come from partial shock-induced combustion (possibly supersonic), or the combustion may be transitioning from a subsonic to a supersonic (SCRAM) mode. Regardless of the mechanism, the existence of transdetonative propulsion may allow the projectile to be accelerated over a wide Mach number range -- from subdetonative to superdetonative -- in only one mixture. Transdetonative propulsion may also have applications for other ramjet engines that must operate over a broad Mach number range such as on the NASP.

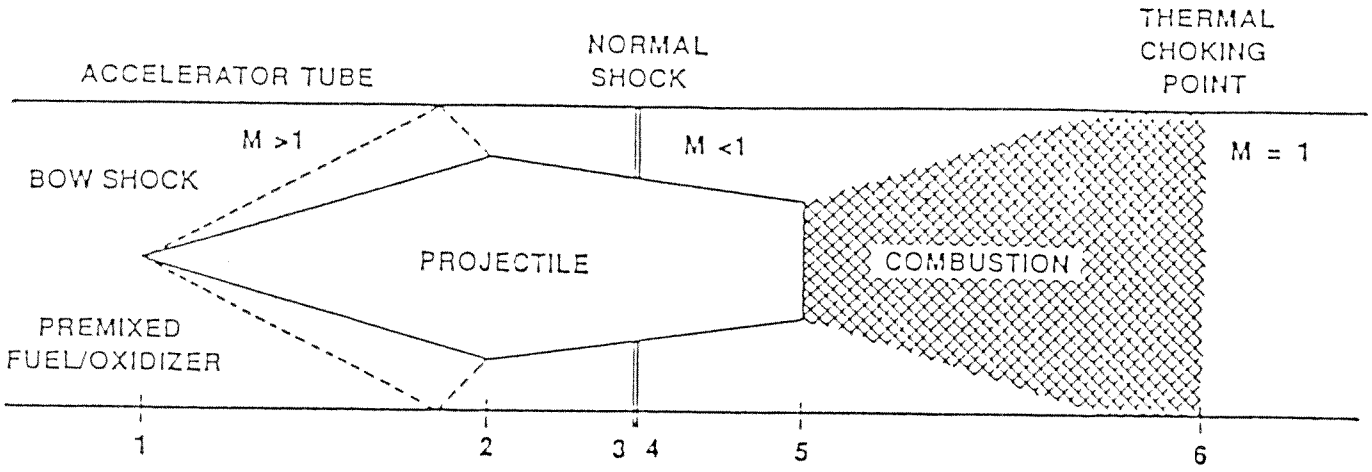


Figure 2.1: Subsonic combustion thermally choked ram accelerator mode.

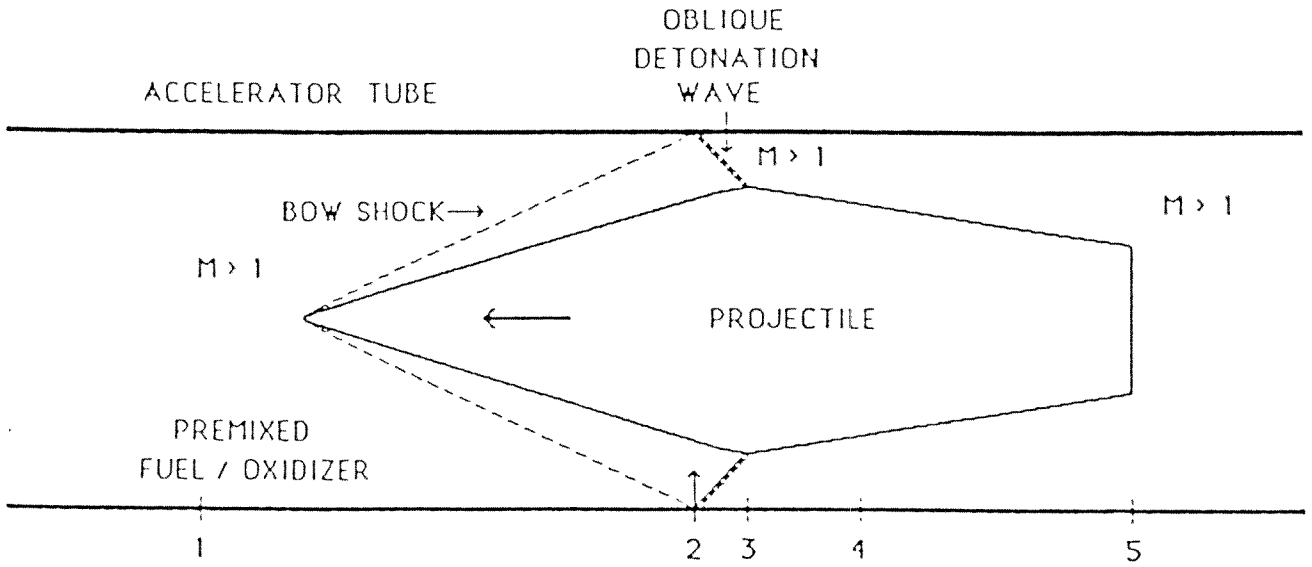


Figure 2.2: Oblique detonation wave ram accelerator mode.

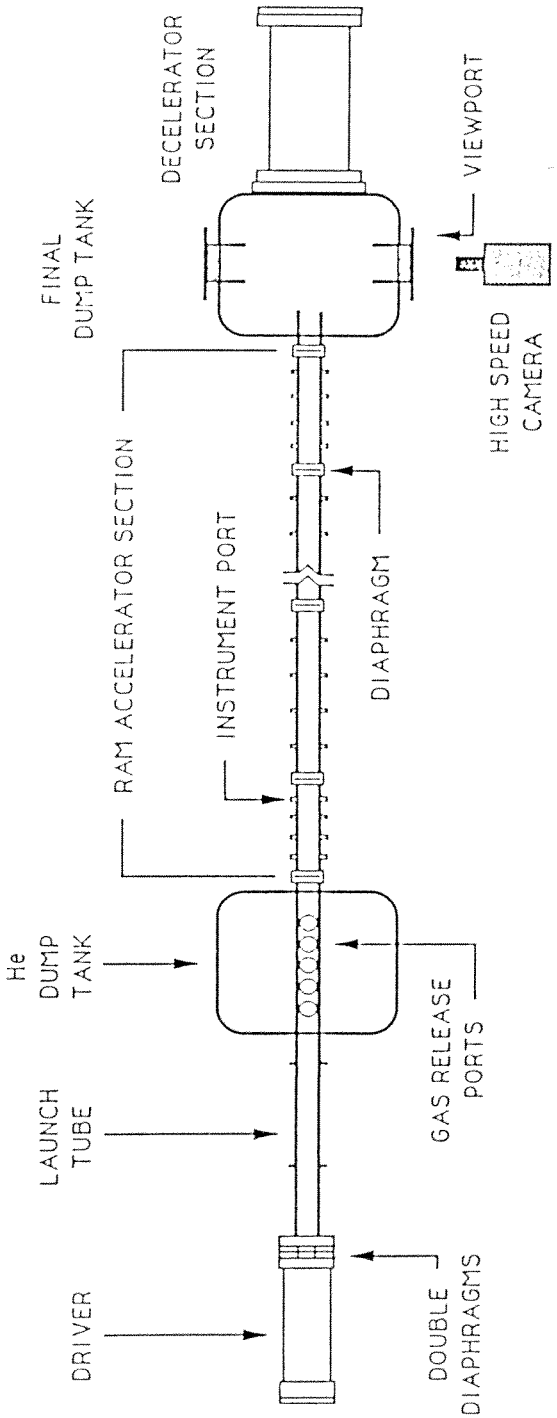
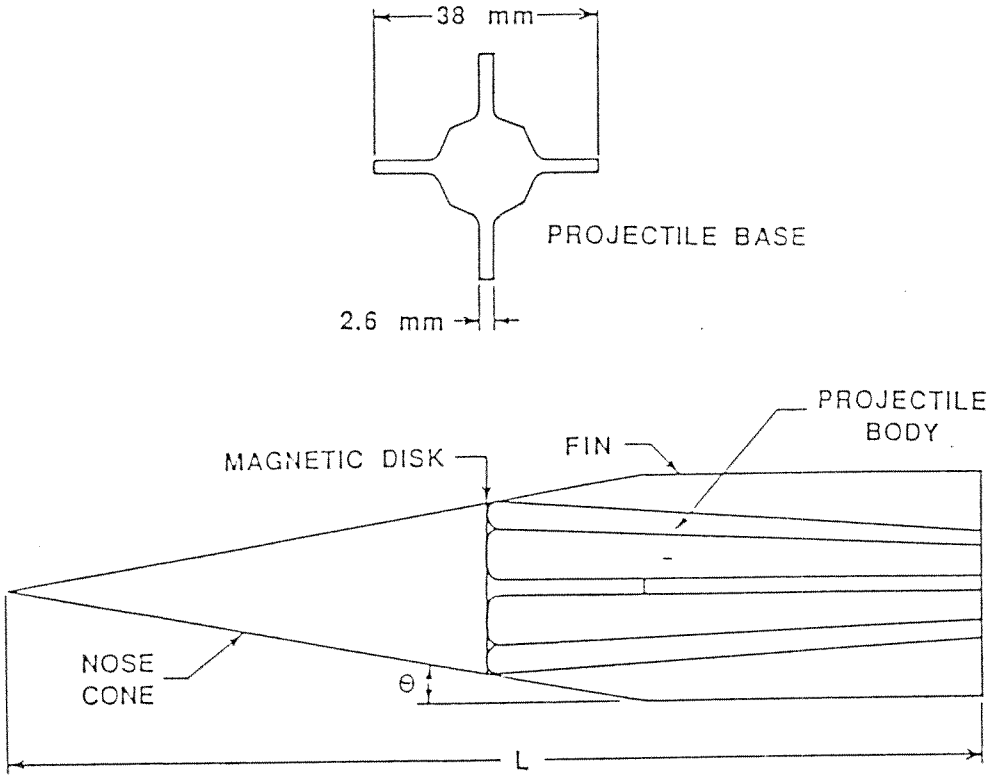


Figure 2.3: Ram accelerator test facility.



PROJECTILE

	TYPE A	TYPE B
L	138 mm	166 mm
Θ	12.5°	10°

Figure 2.4: Typical experimental projectile.

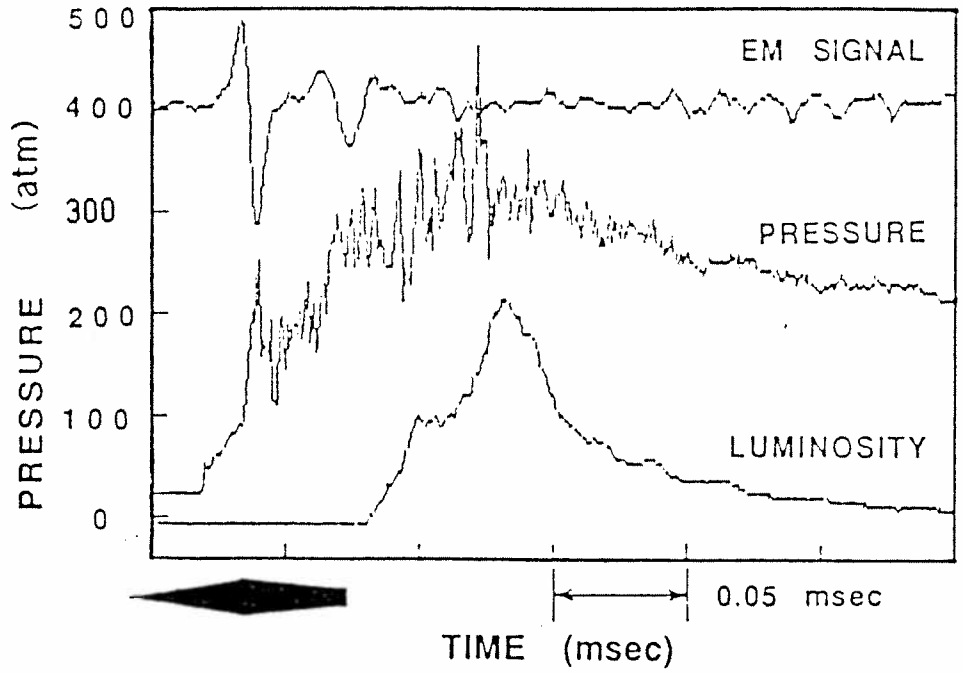


Figure 2.5: Electromagnetic, pressure, and fiber optic signals in ram accelerator operation.

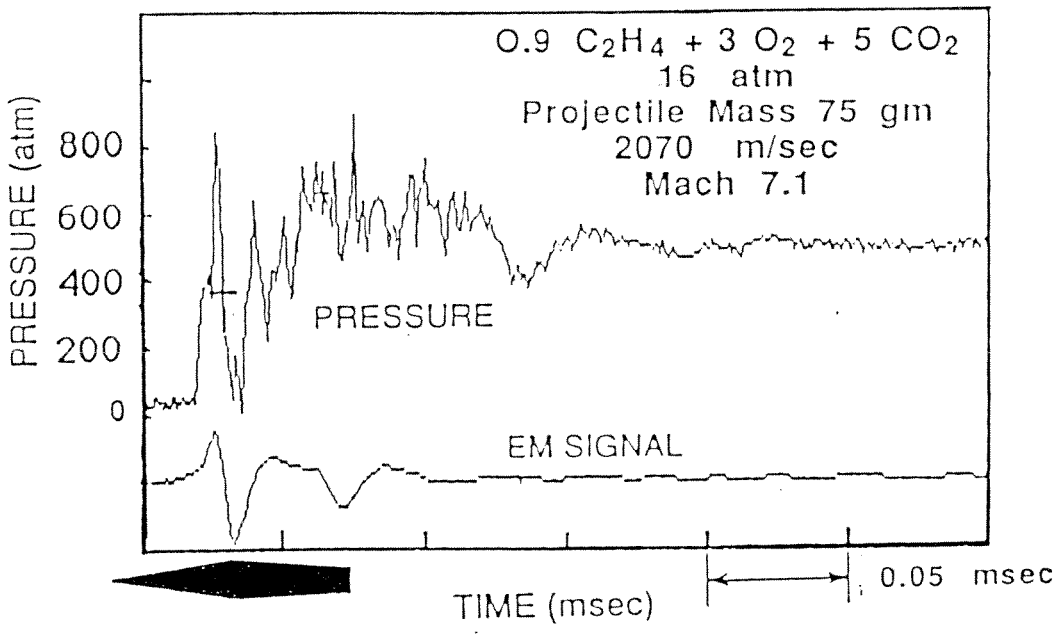


Figure 2.6: EM and pressure signals in superdetonative ram accelerator operation.

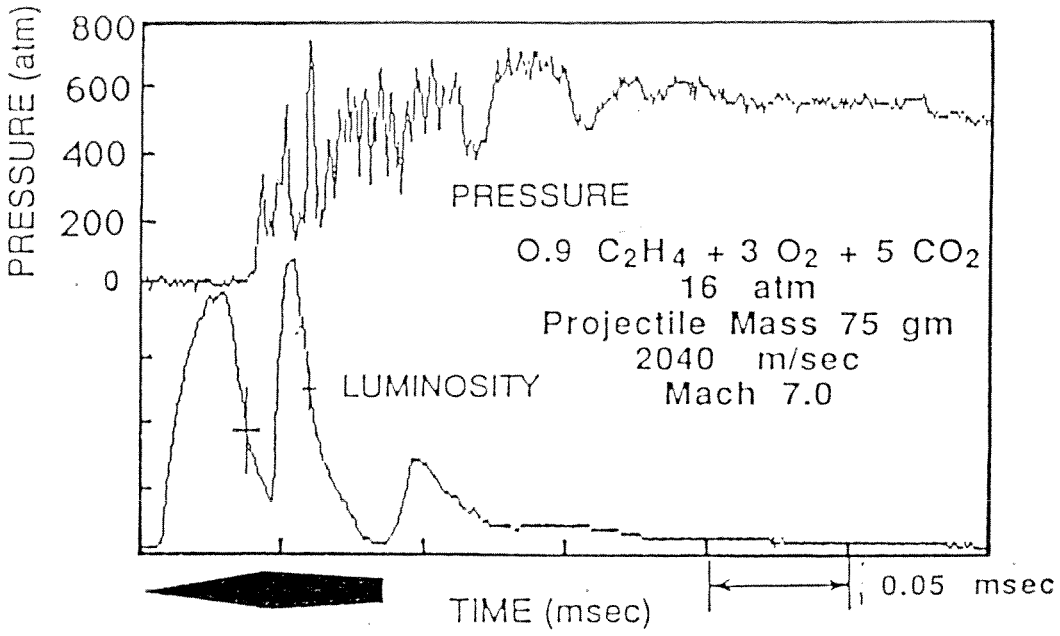
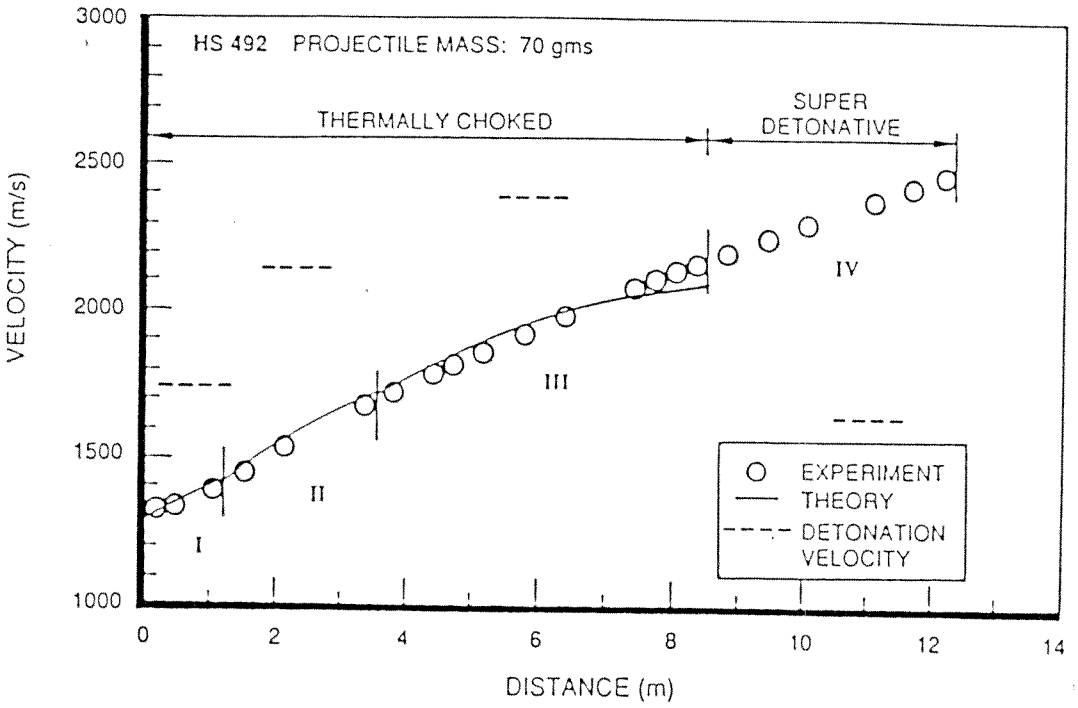
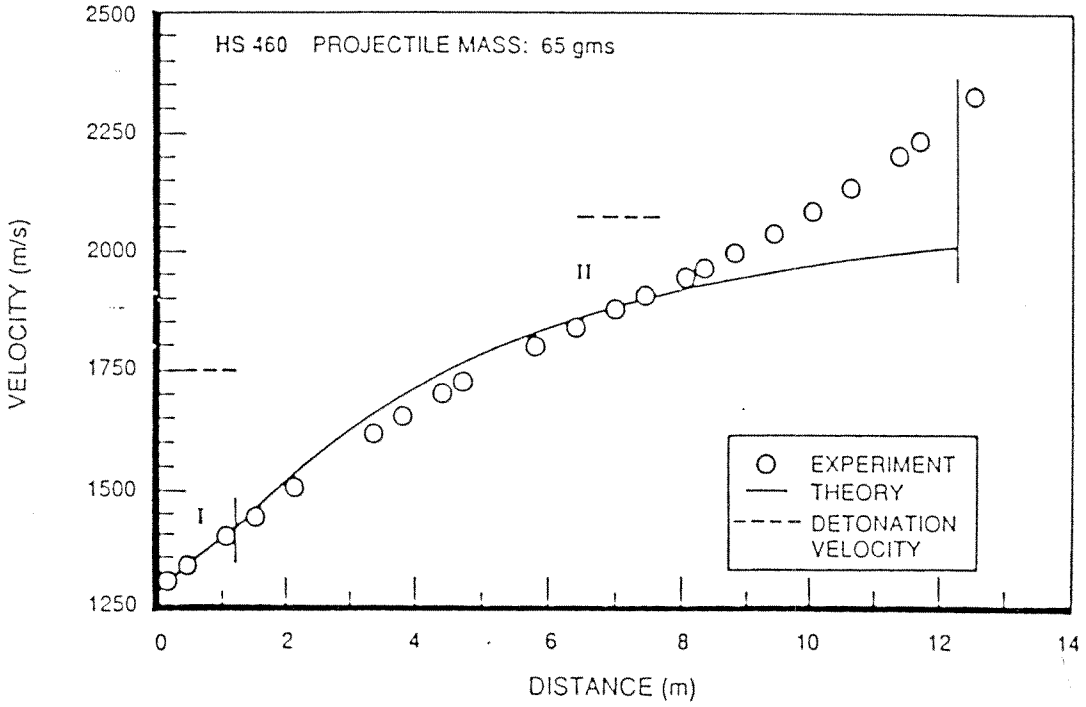


Figure 2.7: Fiber optic and pressure signals in superdetonative ram accelerator operation.



STAGE	MIX	SOUND SPEED	PRESSURE
I	2.6 CH_4 – 2 O_2 – 5.6 N_2	363 (m/s)	31 atm
II	4.5 CH_4 – 2 O_2 – 2 He	448 (m/s)	31 atm
III	3.5 CH_4 – 2 O_2 – 6.5 He	544 (m/s)	31 atm
IV	0.6 C_2H_4 – 2 O_2 – 3.3 CO_2	291 (m/s)	16 atm

Figure 2.8: Velocity profile in ram accelerator: sub- and superdetonative operation.



STAGE	MIX	SOUND SPEED	PRESSURE
I	2.6 CH_2 – 2O_2 – 5.6 N_4	363 (m/s)	25 atm
II	4.5 CH_4 – 2O_2 – 2 He	448 (m/s)	25 atm

Figure 2.9: Velocity profile in transdetonative ram accelerator operation.

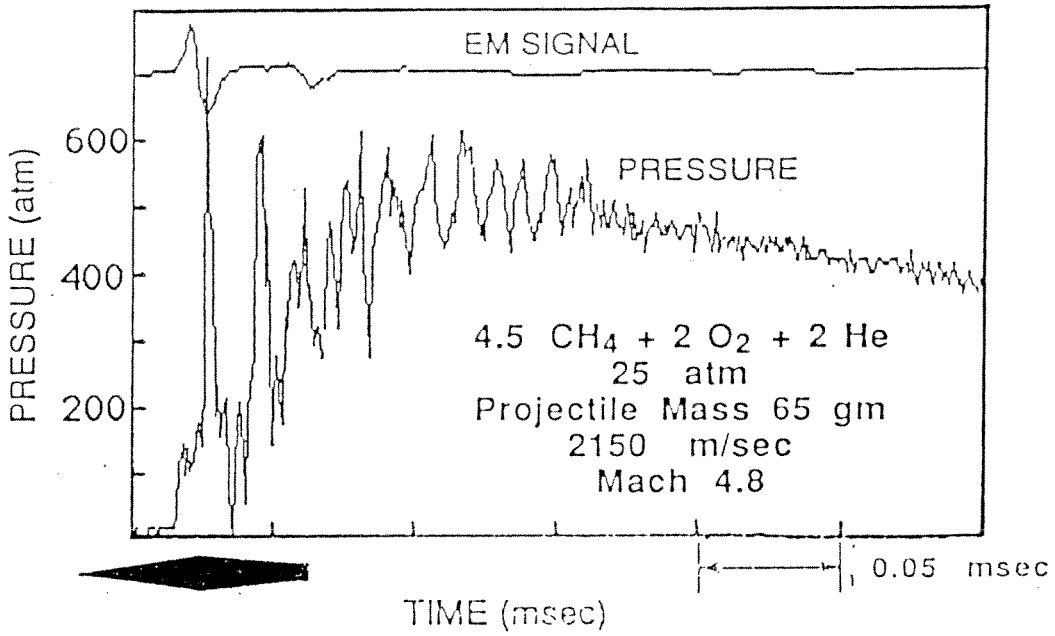


Figure 2.10: Pressure and EM signals in transdetonative ram accelerator operation.

Chapter 3

Chemical Kinetics and the Ram Accelerator

Current models of ram accelerator operation are based on the assumptions of steady flow and equilibrium combustion chemistry. It is believed, however, that in the transdetonative regime, unsteady fluid dynamic effects [16] and finite rate chemistry are of paramount importance. Yungster [6,7] has shown, using a 2-D Euler code and simple hydrogen kinetics, that in the superdetonative regime, finite rate chemistry is required to model the upper and lower operating Mach number limits within a particular gas propellant mixture. His results show that in the low Mach number ranges of the superdetonative regime, heat is released on the projectile body by shock-induced combustion. As the flight Mach number increases, the shock-induced combustion strengthens in magnitude until one of the reflected waves on the rear of the projectile becomes a detonation wave. Further increases in Mach number result in the detonation moving up from being at the third or fourth reflected wave to being the second or third and so on, until the first reflection of the bow wave off the tube wall is a detonation wave. The limiting upper Mach number for superdetonative operation would then be when the conical bow shock is strong enough to induce enough heat release that the flow choked at the projectile throat. Current experimental superdetonative work is being carried out at velocities that are too low for any reasonable hydrogen / oxygen / diluent mixture. As a result, the propellant of choice has been ethylene. Ethylene requires relatively little energy to initiate a detonation, [17] and has a high heat of combustion. In fact, the ethylene-based propellant gas mixture used in the superdetonative work cited above, has twice the chemical energy density of the most energetic thermally-choked mixtures. It would therefore be desirable to be able to predict the upper and lower operating Mach numbers for these mixtures. The computer model developed in this work is meant to be a tool to explore the combustion kinetics of general hydrocarbon / oxygen / diluent reactions.

In the thermally choked mode, a requirement a propellant gas mixture must meet is that it is not ignited by the conical shocks in the diffuser or by the normal shock system on

the projectile body. The gas must also not detonate in the ignition process. Methane has been the primary fuel for all the thermally choked operation to date, primarily because of its docile detonation characteristics. Methane is known to exhibit a delay to explosion, in shock tube experiments, which is nearly 10 times the ignition delay of higher alkanes [17]. The primary need for knowledge of finite rate chemistry behavior in thermally choked mode research is in ignition modeling and investigating the effects certain diluents, such as hydrogen, have on the detonability of the propellant gases.

As mentioned before, the unexpected transdetonative performance is somewhat irreproducible, but very attractive. The ability to accelerate from Mach 3 to Mach 8 in a single gas mixture would be enormously beneficial. Current experimental evidence suggests that transdetonative propulsion involves combustion on the projectile body, caused by shock heating, boundary layer combustion, magnesium ablation, and / or shock-boundary layer interaction. To tame transdetonative propulsion, it is necessary to have an adequate model of the combustion processes that cause it. An integral part of this model must be the chemical kinetics. It may be possible to induce combustion on the projectile body by adding small amounts of a higher hydrocarbon, such as ethane. Westbrook [18] shows that methane burns by first stripping off a hydrogen atom to form methyl radicals which then combine to form ethane. The hydrogen atoms are much easier to strip off ethane than methane so the reaction proceeds quickly after the long induction phase. Westbrook found that adding small amounts of ethane to methane / oxygen mixtures sharply reduced the induction time compared to pure methane / oxygen. For example, when ethane is 5% of the fuel, the induction time is roughly half that for pure methane. This reduction in induction time by a factor of two corresponds to a reduction in the critical energy for direct initiation of detonation by a factor of 8 [19]. This effect is quite large and illustrates the need for a chemical kinetics analysis of such factors.

Chapter 4

Chemical Kinetics Model

4.1 Mathematical Formulation

In this chapter, the chemical kinetics model is developed. A given set of M chemical reactions and reaction rate constants for N chemical species is converted to a set of N ordinary differential equations. These coupled, nonlinear equations are mathematically "stiff," meaning the solutions of some equations (chemical reactions) vary rapidly with time while others vary slowly and / or at intermediate rates. Because of the stiffness of the equations, special solution methods must be used [20]. The method chosen for this model is a modified version of Gear's [21]. The equations are solved for species concentrations as a function of time. Thermochemical data for each species is used to determine the temperature of the reacting mixture at each time step.

4.1.1 Reaction Rates

A one-step chemical reaction of arbitrary complexity can be represented by the following stoichiometric equation:

$$\sum_{i=1}^{N} \upsilon_i' M_i \rightarrow \sum_{i=1}^{N} \upsilon_i'' M_i \qquad (4.1)$$

where υ_i' are the stoichiometric coefficients of the reactants, υ_i'' the stoichiometric coefficients of the products, M_i the species chemical formula, and N the total number of compounds involved [22]. The law of mass action states that the rate of disappearance of a chemical species is proportional to the products of the concentrations of the reacting species, each concentration being raised to the power equal to the corresponding stoichiometric coefficient [23]. Thus the reaction rate is given as

$$RR = k\prod_{i=1}^{N}(C_{M_i})^{\upsilon_i'} \tag{4.2}$$

where k is the reaction constant and C_{M_i} are the species concentrations in units of moles per volume. The reaction constants are given in Arrhenius form:

$$k = BT^{\alpha}\exp\left(-\frac{E_a}{R_u T}\right) \tag{4.3}$$

where B, α, and E_a are empirical constants that depend on the nature and frequency of molecular collisions, R_u is the universal gas constant, and T is the temperature The net rate of production of species M_i is then given by the differential equation:

$$\frac{dC_{M_i}}{dt} = \left(\upsilon_i'' - \upsilon_i'\right)RR = \left(\upsilon_i'' - \upsilon_i'\right)k\prod_{i=1}^{N}(C_{M_i})^{\upsilon_i'} \tag{4.4}$$

Equation (4.4) can be written for each species participating in a given reaction. For a reaction set involving N species, the rate equations for each species are summed to yield a set of N ordinary differential equations.

4.1.2 Solution of Differential Equations

The N differential equations of the form given by Equation (4.4) are coupled, highly non-linear, and, as mentioned before, are stiff. The computer program, listed in Appendix A, solves the set of equations using a multi-value method based on the method of Gear [20,21]. The species concentrations are determined over time intervals specified by the user. After each time step the temperature is determined by the following procedure. Thermochemical data, in the form of enthalpy versus temperature for each species, are initially read into the program [24]. These data are then curve fit using piecewise cubic Hermite polynomial interpolation [20]. The initial (before combustion) species concentrations are multiplied by the sum of the enthalpy (at the initial temperature) and the enthalpy of formation for each species. These products are then summed to give the initial enthalpy of the gas mixture. It is assumed that the reactions occur adiabatically, so that the enthalpy remains constant. Thus

$$\sum_{i=1}^{n}\left\{C_i\left[\left(\bar{h}^0-\bar{h}^0_{298}\right)_i+\left(\Delta\bar{h}^0_{f,298}\right)_i\right]\right\}_{\text{REACTANTS}}=\sum_{i=1}^{n}\left\{C_i\left[\left(\bar{h}^0-\bar{h}^0_{298}\right)_i+\left(\Delta\bar{h}^0_{f,298}\right)_i\right]\right\}_{\text{PRODUCTS}} \quad (4.5)$$

where $\left(\bar{h}^0-\bar{h}^0_{298}\right)$ is the enthalpy minus the reference enthalpy at 298 K for species i and $\Delta\bar{h}^0_{f,298}$ is the enthalpy of formation for species i. Since the reactions are adiabatic and the initial species concentrations and temperature are known, the left hand side is easily determined and is constant. In the right hand side, the species concentrations are determined at each time step by the ordinary differential equation (ODE) solver. The enthalpies of formation for the species are known and do not change with time or temperature, so only the enthalpy remains to be found. This is achieved iteratively by guessing a product temperature, evaluating the polynomial interpolants to determine the enthalpies, summing the enthalpies over the species, and checking to see if equation { } balances. Usually, on first attempt, the assumed temperature is incorrect and must be modified, the entire procedure then repeated. This iterative process is accomplished very efficiently using a procedure that is a combination between the method of bisection and the secant rule [20]. The iteration usually converges within 3 or 4 cycles to within a relative error of 1%. This method is the same as that used to find adiabatic flame temperature in an equilibrium combustion problem. The updated temperature is used in the next time step of the ODE solver, affecting the solution through the temperature dependence of the kinetic rate constants.

4.2 Numerical Results

The program was run for many cases, one of which will be presented. Westbrook [18] has developed a kinetic model for the oxidation of methane, ethane, and oxygen that includes 75 reaction equations. Guirguis [25] took this reaction set and developed a much simpler, four equation model, primarily useful for studying methane ignition. This reaction set is given by:

$CH_4 + O_2 \rightarrow CH_3 + HO_2$	$k = 10^{13}\exp(-56 / RT)$
$CH_3 + O_2 \rightarrow CH_3O + O$	$k = 2.4*10^{13}\exp(-29 / RT)$
$CH_3 + H + X \rightarrow CH_4 + X$	$k = 2.0*10^{14}$
$CO_2 + 2H_20 \rightarrow CH_3 + 1/2OH + 1/2H$	$k = 10^{7}\exp(-50 / RT)$

This reaction set was solved first for the early part of the reaction, i.e. the induction zone. A step function in temperature of 2000 K was used to model shock-initiated ignition. The initial concentrations were 0.1 moles / cm^3 of CH_4 , 0.2 moles / cm^3 of O_2 and 0.3 moles of X, which is the sum of all species present. X represents any third body that is non-participating in the reaction; it merely supplies or extracts enough collisional energy to facilitate a reaction.

Figure 4.1 shows the concentrations of oxygen and methane as a function of time throughout the induction phase of the reaction. As expected, the concentrations decrease as these compounds are consumed to form radical species and products. Figure 4.2 shows the concomitant rise in the concentration of products, water and carbon dioxide.

Figure 4.3 follows the concentration of the hydroxyl radical with time. The amount of OH present quickly jumps from zero then changes to a much gentler slope as the hydroxyl radicals attack the unconsumed reactants. Figure 4.4 shows the concentration of CH_3O as a function of time. It's behavior is similar to the hydroxyl radical in that it quickly establishes an initial concentration, yet it nearly plateaus for most of the rest of the induction phase. Note that Fig. 13 is a plot of the logarithm of the concentration of OH, whereas Fig. 4.4 is simply a linear plot.

Figure 4.5 illustrates the behavior of the methyl radical through the induction zone. CH_3 has a very sharp peak in concentration very early in the flame, then quickly is consumed. Evidently the methyl radical is the dominant chain branching species in the very early stages of the reaction. Figure 4.6 follows the concentration of monatomic hydrogen with time. Like the methyl radical, hydrogen atoms peak very early then decay.

Figure 4.7 shows the concentration of HO_2 , which behaves like the hydroxyl concentration. Similar behavior is exhibited by monatomic oxygen in Fig. 4.8.

Finally, figure 4.9 shows the temperature history throughout the induction zone. The temperature drops quickly as the random thermal energy is used to break apart the initial species. When this case is run for longer times, the temperature climbs back up to about 3200 K which is slightly higher than experimentally-measured adiabatic flame temperatures for stoichiometric methane / oxygen mixtures [17].

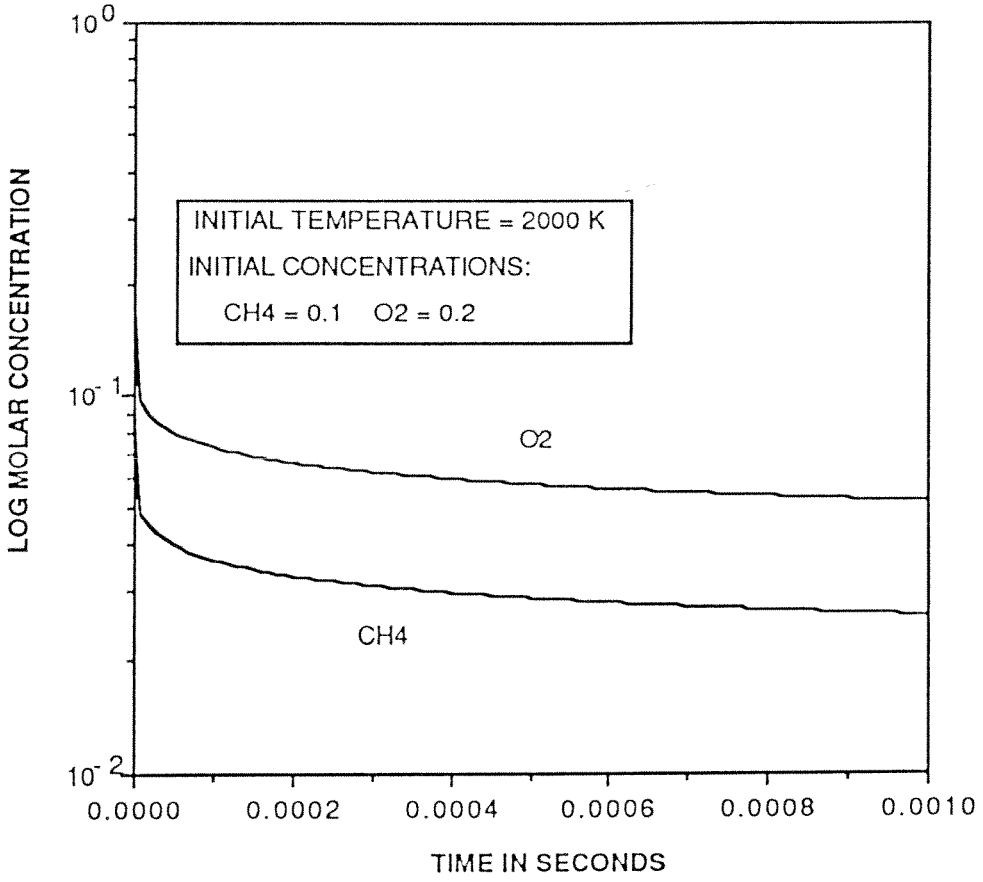


Figure 4.1: Concentration of methane and oxygen in induction zone.

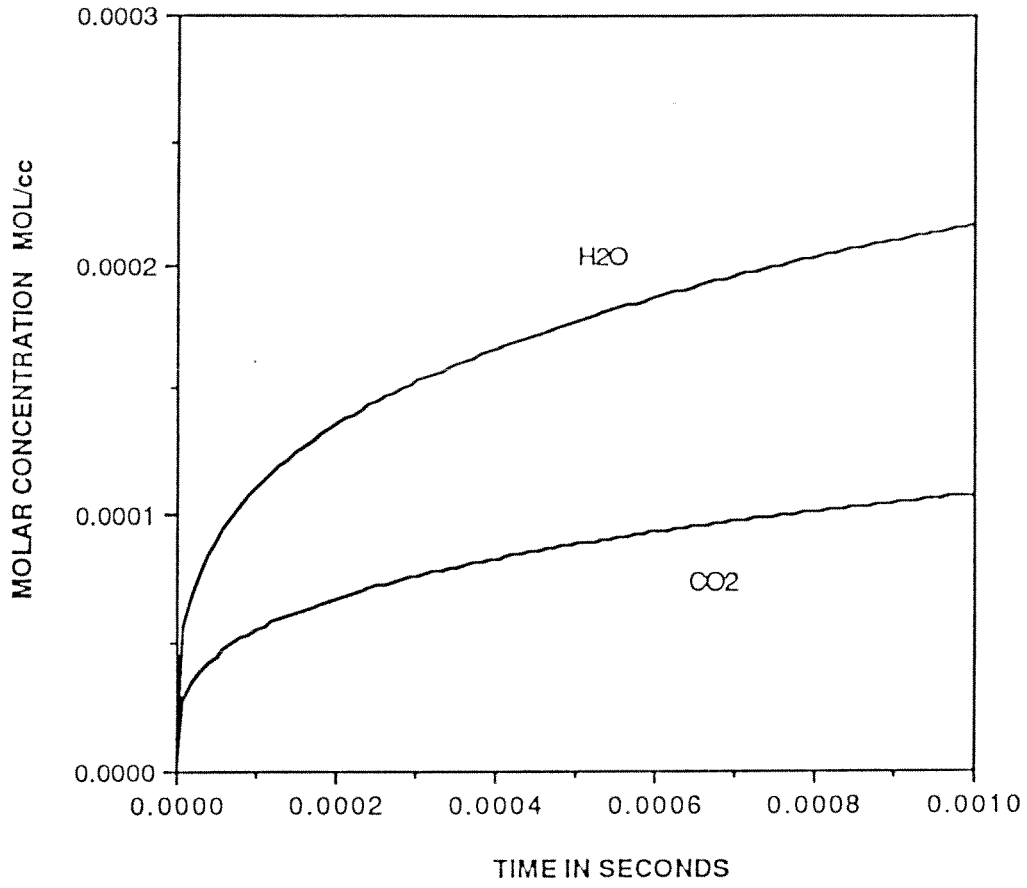


Figure 4.2: Concentration of water and carbon dioxide in induction zone.

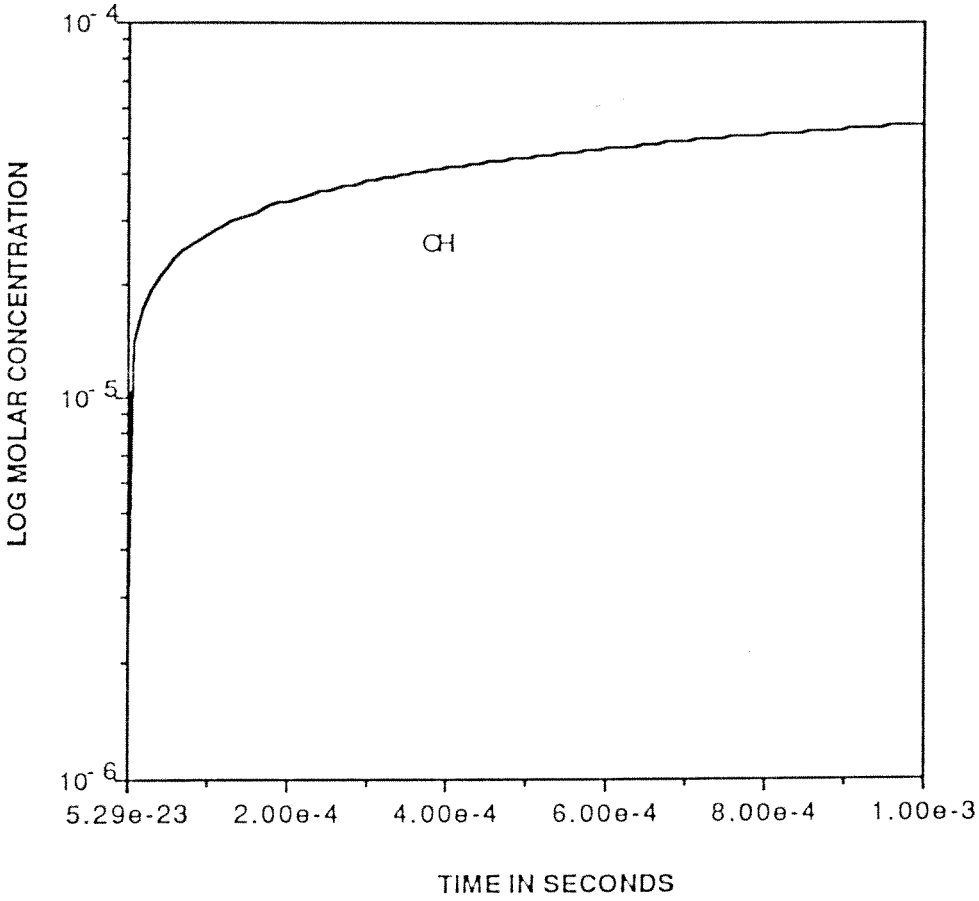


Figure 4.3: Concentration of hydroxyl radical in induction zone.

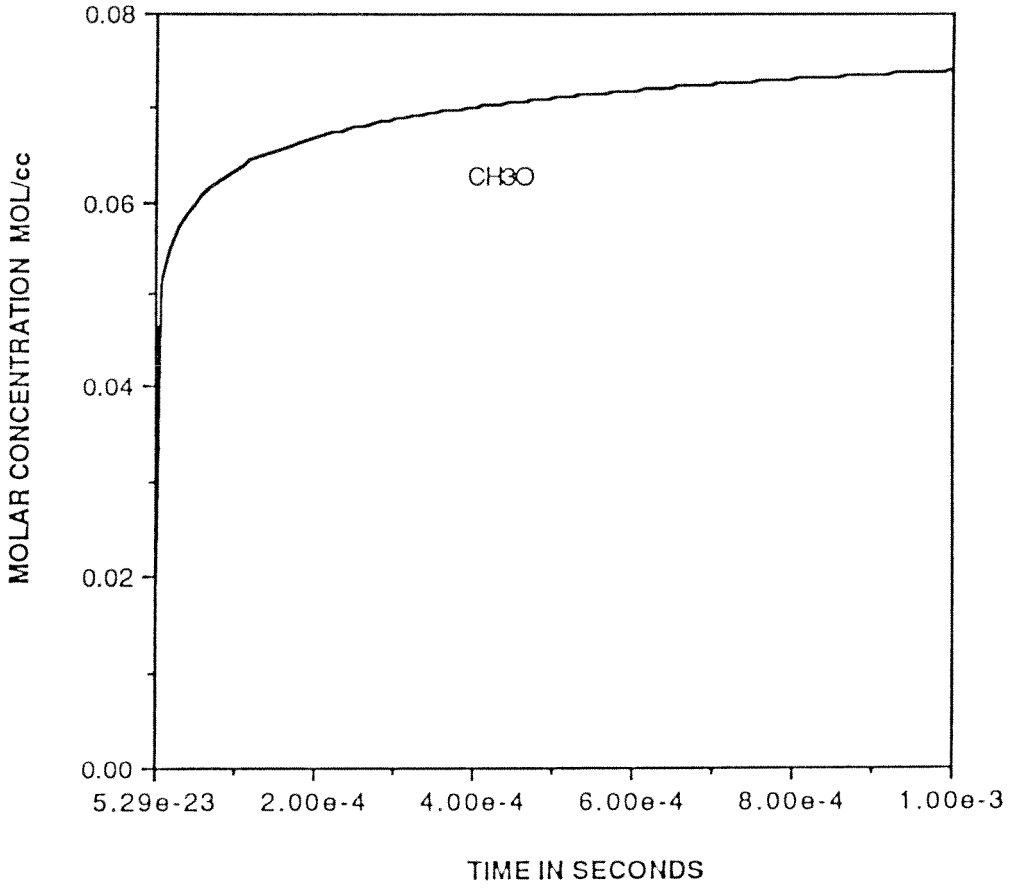


Figure 4.4: Concentration of CH3O in induction zone.

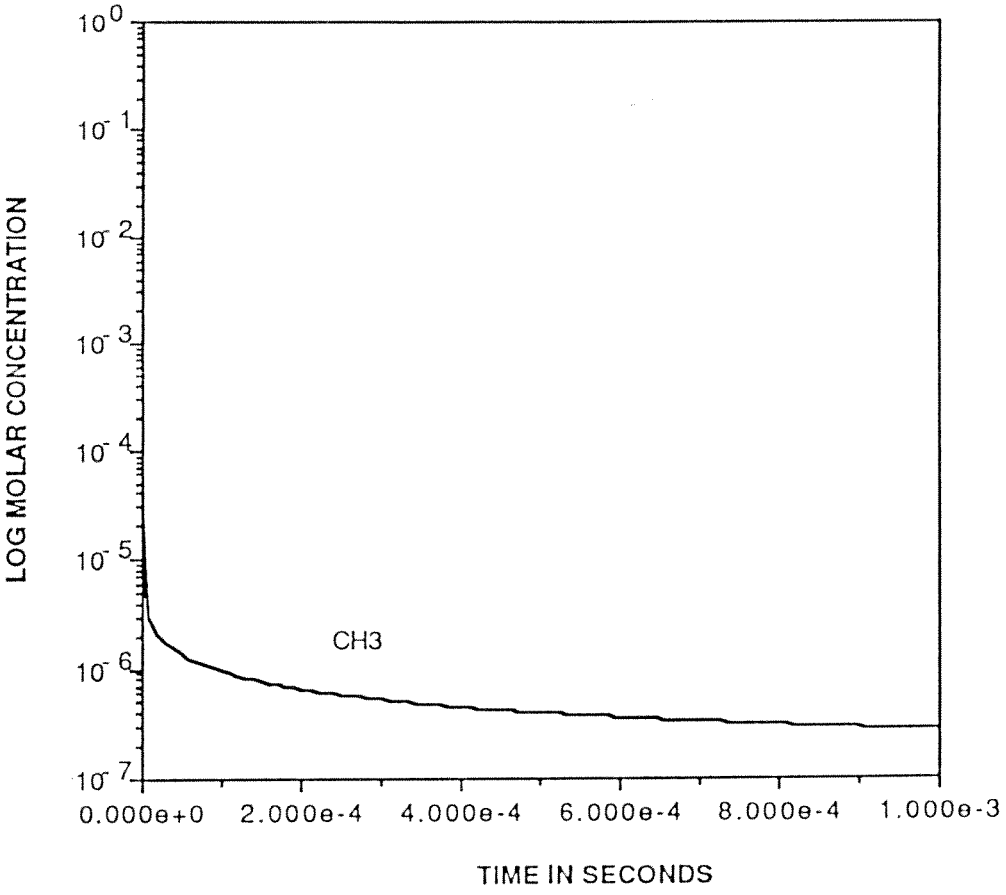


Figure 4.5: Concentration of methyl radical in induction zone.

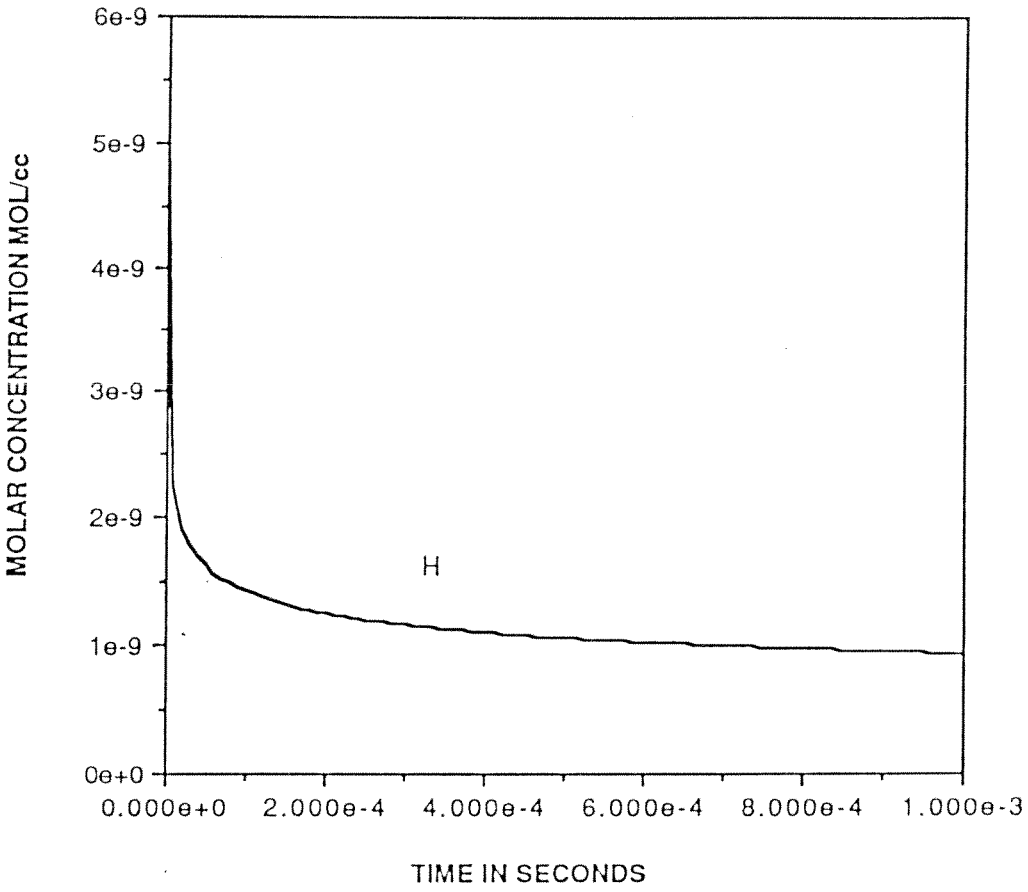


Figure 4.6: Concentration of atomic hydrogen in induction zone.

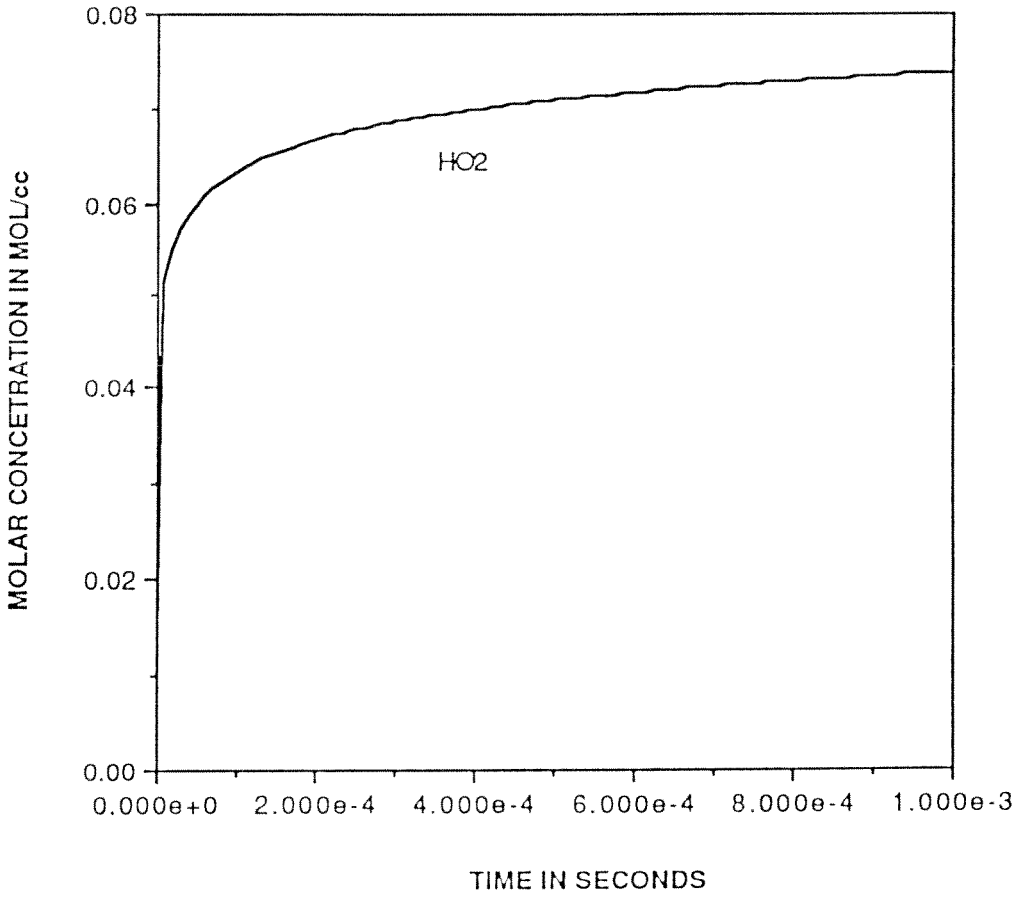


Figure 4.7: Concentration of HO2 in induction zone.

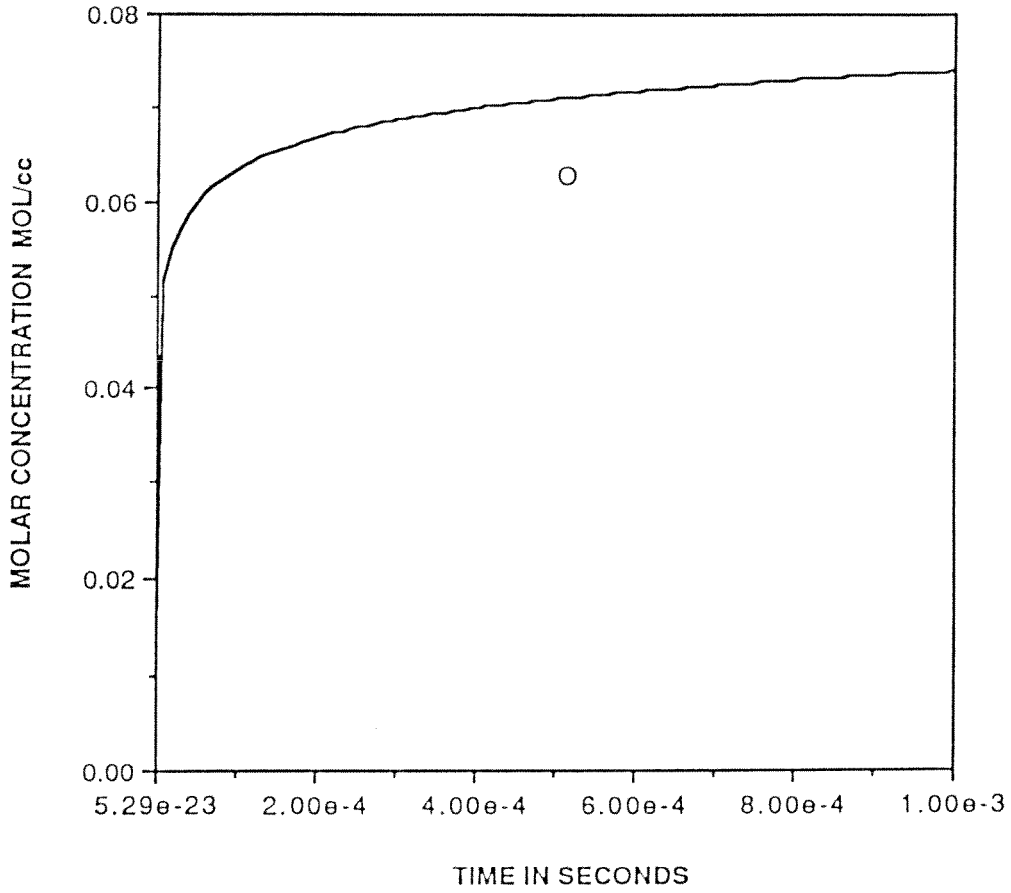


Fig. 4.8: Concentration of atomic oxygen in induction zone.

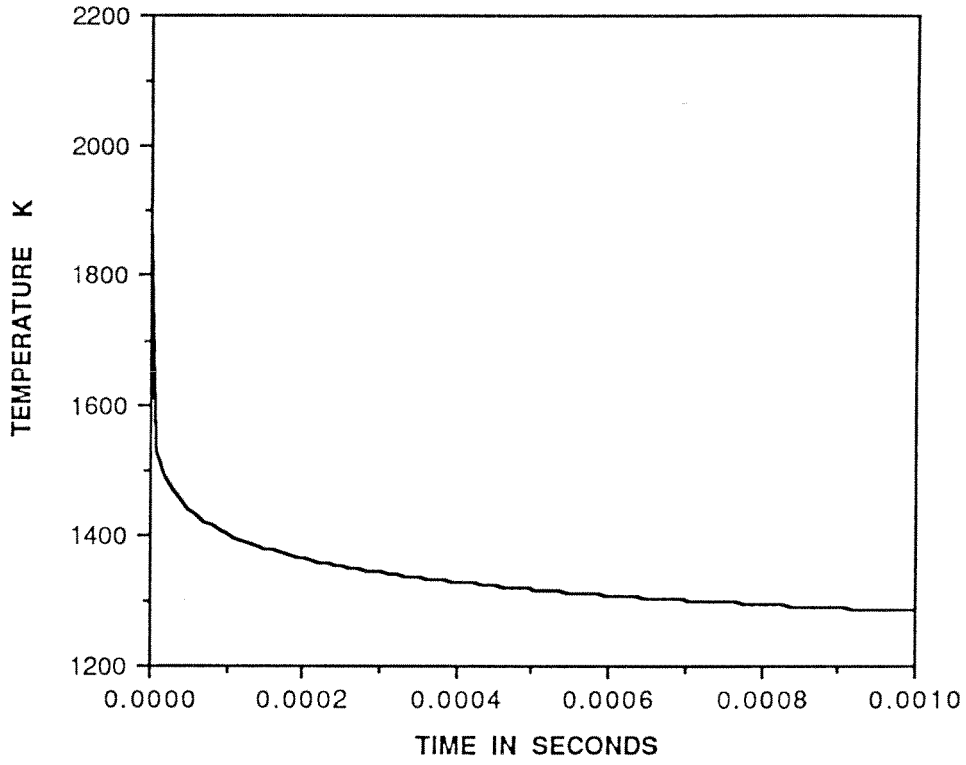


Figure 4.9: Temperature throughout induction zone.

Chapter 5

Conclusion

Acceleration of projectiles by ram accelerator propulsive modes at velocities greater than the local C-J detonation speed has been experimentally demonstrated in both methane and ethylene-based propellant mixtures. Projectiles were accelerated through the velocity range of 2000 m/sec to 2500 m/sec by an ethylene-oxygen-carbon dioxide propellant mixture having an experimentally determined detonation speed of 1650 m/sec. In several methane-based propellant mixtures, projectiles have been smoothly accelerated through the transdetonative regime (85%-115% C-J detonation velocity). All propellant mixtures used in the thermally choked propulsive mode have demonstrated extraordinary accelerations when the projectiles have been allowed to approach the detonation velocity of the mixture, and in several experiments the projectiles actually passed through the detonation velocity and continued to accelerate. These experiments suggest that smooth acceleration from a low Mach number, subdetonative regime to a hypersonic, superdetonative regime may be possible in a single propellant mixture.

The inability to accurately model the combustion phenomena involved in transdetonative and superdetonative ram accelerator propulsion modes demonstrates the need for more sophisticated tools. The computer program contained herein will hopefully be used as a tool to further investigate finite rate chemistry phenomena important to the ram accelerator. It is recommended that future work will be directed toward examining large reaction schemes, and reducing them to a few reactions that are important to the particular conditions being investigated. Also, it would be very beneficial to find experimentally-measured high pressure kinetics models of methane and higher hydrocarbon combustion.

Bibliography

1. Hertzberg, A., Bruckner, A.P. and Bogdanoff, D.W., "Ram Accelerator: A New Chemical Method for Accelerating Projectiles to Ultrahigh Velocities," AIAA J., Vol. 26, pp. 195-203, 1988.

2. Bruckner, A.P., Bogdanoff, D.W., Knowlen, C. and Hertzberg, A.,"Investigation of Gasdynamic Phenomena Associated with the Ram Accelerator Concept," AIAA Paper 87-1327, 1987.

3. Knowlen, C., Bruckner, A.P., Bogdanoff, D.W. and Hertzberg, A., "Performance Capabilities of the Ram Accelerator," AIAA Paper 87-2152, 1987.

4. Bruckner, A.P., Knowlen, C., Scott, K.A. and Hertzberg, A., "High Velocity Modes of the Thermally Choked Ram Accelerator," AIAA Paper 88-2925, 1988.

5. Bruckner, A.P., Knowlen, C., Scott, K.A., Hertzberg, A., and Bogdanoff, D.W., "Operational Characteristics of the Thermally Choked Ram Accelerator," to be published in Journal of Propulsion and Power.

6. Yungster, S., Eberhardt, S. and Bruckner, A.P., "Numerical Simulation of Shock-Induced Combustion Generated by High-Speed Projectiles in Detonable Gas Mixtures," AIAA Paper 89-0673, 1989.

7. Yungster, S. and Bruckner, A.P., "A Numerical Study of the Ram Accelerator Concept in the Superdetonative Velocity Range," AIAA Paper 89-2677, 1989.

8. Hertzberg, A., Bruckner, A.P. and Mattick, A.T., "A Chemical Method for Achieving Acceleration of Macroparticles to Ultrahigh Velocities," Final Report, UWAERP/15, Department of Energy Grant No. DE FG06 85ER13382, Aerospace and Energetics Research Program, University of Washington, Seattle, WA, 1987.

9. Bruckner, A.P. and Hertzberg, A., "Ram Accelerator Direct Launch System for Space Cargo," IAF Paper 87-211, 1987.

10. Kaloupis, P. and Bruckner, A.P., "The Ram Accelerator: A Chemically Driven Mass Launcher," AIAA Paper 88-2968, 1988.

11. Ostrander, M.J., Hyde, M.F., Young, R.D. and Kissinger, R.D., "Standing Oblique Detonation Wave Engine Performance," AIAA Paper 87-2002, 1987.

12. Pratt, D.T., Humphrey, J.W., and Glenn, D.E., "Morphology of a Standing Oblique Detonation Wave," AIAA Paper 87-1785, June 1987.

13. Kull, A.E., Burnham, E.A., Knowlen, C., Bruckner, A.P., and Hertzberg, A.,"Experimental Studies of Superdetonative Ram Accelerator Modes," AIAA Paper 89-2632, 1989.

14. Burnham, E.A., Kull, A.E., Knowlen, C., Bruckner, A.P., and Hertzberg, A.,"Operation of the Ram Accelerator in the Transdetonative Velocity Regime," AIAA Paper 90-1985, 1990.

15. Shapiro, A.H., The Dynamics and Thermodynamics of Compressible Fluid Flow, Vol I, John Wiley and Sons, New York, 1953, pp. 135-137.

16. Burnham, E.A., Unpublished Work, 1990.

17. Strehlow, R.A., Combustion Fundamentals, McGraw-Hill, New York, 1984, pp. 211-254.

18. Westbrook, C.K. and Haselman, L.C., "Chemical Kinetics in LNG Detonations," in Gasdynamics of Detonations and Explosions, Bowen, J.R., ed., Volume 75 Progress in Astronautics and Aeronautics, AIAA, New York, 1981, pp. 193-206.

19. Lee, J.H. (1977) "Initiation of Gaseous Detonation," *Ann. Rev. of Phys. Chem.* **28**, pp. 75-104.

20. Kahaner, D., Moler, C., and Nash, S., Numerical Methods and Software, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

21. Gear, C.W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Englewood Cliffs, New Jersey, 1971.

22. Kuo, K.K., Principles of Combustion, John Wiley and Sons, New York, 1986, pp. 109-122.

23. Vincenti, W.G. and Kruger, C.H., Introduction to Physical Gas Dynamics, Krieger Publishing, Malabar, Florida, 1982, pp. 210-216.

24. Stull, D.R. and Prophet, H., "JANAF Thermochemical Tables, 2nd. Ed.," NSRDS-Report 37, National Bureau of Standards, June 1971.

25. Guirguis,R.H., Oppenheim,A.K., Karasalo,I., and Creighton,J.R., "Thermochemistry of Methane Ignition," in Combustion in Reactive Systems, Bowen, J.R., ed., Volume 76 Progress in Astronautics and Aeronautics, AIAA, New York, 1981, pp. 134-153.

Appendix

Listing of FORTRAN Computer Program

```
C
C       This program solves a general chemical kinetics reaction
C       set. The reaction equations and rate constants are input
C       by the user. Thermochemical data from the JANAF tables
C       must be in a file named JANAF.DAT. These data must include
C       the species name, molecular weight, enthalpy of formation at
C       298 K, and a table of enthalpy minus (enthalpy at 298 K) vs.
C       temperature.

C       Program was written by Alan Kull in partial fulfillment of
C       the degree of Master of Engineering from the Department of
C       Aeronautics and Astronautics at the University of Washing-
C       ton. 8/13/90

C       Ramo Ergo Sum: It's not just a motto; it's a way of life.

        PROGRAM ODETOILET

        REAL TEMP,KF(80),Y(80),RU,YDOT(80),TIMAX,TIM,STOPRO(80,80),
     &    STOREC(80,80),WORK(7600),EWT,EPS,TOUT,AF(80),BF(80),CF(80),
     &    TEMPH(80,100),ENTHAL(80,100),MW(80),HFNOT(80),HTOT,YPT(80),
     &    HFITT(80,80),CTEMP(80),CENTH(80),D(8),WK(200),XVAL(1),
     &   FVAL(1),DVAL(1)
        INTEGER IWORK(200),LENW,LENIW,MINT,
     &    NROOT,MSTATE,N,NSPECS,NREACT,
     &     JKOUNT(80),NDUMMY,LWK,NVAL,IERR
        CHARACTER*8 NAME(80), SPENAM
        LOGICAL SPLINE
        COMMON /COM1/ HFITT,HTOT,HFNOT,NAME,TEMPH,ENTHAL,
     &          JKOUNT,NSPECS,YPT


        EXTERNAL FSUB,FTMP

        OPEN(UNIT=11,STATUS='NEW',FILE='SPECSOUT',ERR=5000)
        OPEN(UNIT=12,STATUS='UNKNOWN',FILE='NEWSET.DAT',ERR=5000)
        OPEN(UNIT=13,STATUS='OLD',FILE='JANAF.DAT',ERR=5000)
        OPEN(UNIT=14,STATUS='NEW',FILE='TEMPVTIME',ERR=5000)
C
C       The reactions are initially input interactively via screen commands
C       and keyboard input. These data are then stored in a file named
C       NEWSET.DAT. A previously-input reactions set stored under the
```

```
C     file name NEWSET.DAT can also be read by the program.  Be careful
C     to rename the file something other than NEWSET.DAT when you want
C     to save a particular reaction set and input another, else the new
C     set will overwrite the old one.
C
C     A note on portability of this program:  This program was originally
C     on an Apple Macintosh IIx computer using a Language Systems version
C     2.0 FORTRAN compiler.  When transporting this code to other machines,
C     there may be syntax conflicts.  First, the command ACCEPT *, {variable}
C     is used to input stuff from the screen / keyboard.  Also, the machine
C     constants used in subroutines R1MACH and I1MACH should be changed to
C     the particular computer you are using.  The constants for many
C     computers, such as IBM PC, DEC VAX, and CRAY are already in these
C     subroutines but are "commented out" by the use of a "C" in the first
C     column of code.  Only the machine constants for the Macintosh are
C     executable.  Simply, put a "C" in the first column of the Macintosh
C     numbers, and activate the appropriate
C     ones.
C
C     It is assumed that the rate constants are of Arrhenius form, i.e.
C                    k=B*T**A*exp(-C/(Ru*T)).
C     Usually, the constants B,T,and C are given.  Sometimes the activation
C     energy (the constant C) is given such that the rate constants are of
C     the form
C                    k=B*T**A*exp(-C/T)),
C     where the universal gas constant is already in the constant, C.  In
C     this case, select Ru=1. when queried by the program.  Further, when the
C     rate constants are given as forward and backward rate constants, simply
C     use the backward reactions as separate reactions.
C
C     Many reactions involve collisions with molecules that do not
C     participate in the chemical reaction.  For instance, oxygen dissociates
C     according to the reaction
C
C                    O2 + X -> 2O + X
C
C     where X is any of the species present.  It is important, when using this
C     program to name such non-participating species "X" .  However, when asked
C     by the program to give the total number of species in the reaction set,
C     include "X" in that number.
C
C
      ************************************************************************
*******
C     First executable statement of ODETOILET
C
 20   WRITE(6,*) 'IS THIS A NEW REACTION / SPECIES SET OR AN OLD ONE?'
      WRITE(6,*) 'TYPE IN ONE OF THE FOLLOWING:'
      WRITE(6,*) '1 = NEW SET TO BE INPUT FROM KEYBOARD'
      WRITE(6,*) '2 = PREVIOUSLY-INPUT SET TO BE READ FROM FILE THAT'
      WRITE(6,*) '    YOU NAMED NEWSET.DAT'
      ACCEPT *, NSTOR
```

```
      WRITE(6,*)
      WRITE(6,*) 'IS THE UNIVERSAL GAS CONSTANT '
      WRITE(6,*) 'IN CGS UNITS OR IS IT UNITY ?'
      WRITE(6,*) 'TYPE IN ONE OF THE FOLLOWING:'

      WRITE(6,*) '1 = CGS'
      WRITE(6,*) '3 = RU=1'
      ACCEPT *, NUNIT
      IF (NUNIT .EQ. 1) THEN
       RU = 1.987165E-3   !kcal/(Kelvin*mole)
      ELSEIF (NUNIT .EQ. 3) THEN
       RU = 1.
      ENDIF


      IF (NSTOR .EQ. 2) THEN            !Data will be read from file
        WRITE(6,*) 'READING IN DATA'
        CALL STORE (NREACT,NSPECS,NAME,STOREC,STOPRO,AF,BF,CF)
        GOTO 1070
      ELSEIF (NSTOR .EQ. 1) THEN        !Data will be input by user
        WRITE(6,*)
        GOTO 40
      ENDIF
      WRITE(6,*) 'WHAT"S THE MATTER?  IT"S EITHER 1 OR 2. TRY AGAIN.'
      GOTO 20


 40   WRITE(6,*) 'INPUT TOTAL NUMBER OF REACTIONS'
      ACCEPT *, NREACT
      WRITE (12,*) NREACT
      WRITE(6,*) 'INPUT TOTAL NUMBER OF SPECIES'
      ACCEPT *, NSPECS
      WRITE (12,*) NSPECS

C     INITIALIZE "NAME" ARRAY
      DO 50 K=1,NSPECS
      NAME(K) = ' '
 50   CONTINUE


C     ***************************************************************
C     This section is the interactive input of the reaction set
C
      WRITE(6,*)
      WRITE(6,*) 'WE WILL NOW INPUT THE REACTION SET'
      WRITE(6,*) 'REACTION EQUATIONS ARE OF THE FORM:'
      WRITE(6,*) '  aA + bB -> cC +dD  '
      WRITE(6,*) 'THERE MAY BE MORE OR FEWER TERMS ON'
      WRITE(6,*) 'EITHER SIDE.      THE SPECIES ON THE '
      WRITE(6,*) 'LEFT HAND SIDE OF THE REACTION ARE '
      WRITE(6,*) 'CALLED THE "REACTANTS," i.e., SPECIES'
      WRITE(6,*) 'A AND B.  THE LEFT SIDE OF THE EQUATIONS'
```

```
      WRITE(6,*) 'CONTAINS THE "PRODUCTS," C AND D. THE '
      WRITE(6,*) 'COEFFICIENTS OF THESE SPECIES ARE CALLED'
      WRITE(6,*) 'THE STOICHIOMETRIC COEFFICIENTS.  SO a '
      WRITE(6,*) 'IS THE STOICHIOMETRIC COEFFICIENT OF '
      WRITE(6,*) 'SPECIES A. I WILL ASK YOU TO INPUT, FOR '
      WRITE(6,*) 'EACH REACTION: THE NUMBER OF REACTANTS, THE'
      WRITE(6,*) 'NAMES OF THESE REACTANTS, AND THEIR RESPECTIVE'
      WRITE(6,*) 'STOICHIOMETRIC COEFFICIENTS. WE WILL DO THE SAME'
      WRITE(6,*) 'THING FOR THE PRODUCTS. AS AN EXAMPLE, LOOK AT'
      WRITE(6,*)
      WRITE(6,*) '              2 OH -> O + H2O              '
      WRITE(6,*)
      WRITE(6,*) 'THIS REACTION HAS 1 REACTANT, NAMED OH, WHOSE '
      WRITE(6,*) 'STOICHIOMETRIC COEFFICIENT IS 2.  THE REACTION'
      WRITE(6,*) 'HAS TWO PRODUCTS, O AND H20, BOTH OF WHICH HAVE'
      WRITE(6,*) 'STOICHIOMETRIC COEFFICIENTS EQUAL TO 1.'
      WRITE(6,*)
      WRITE(6,*) 'WELL, LET US BEGIN .....'
      WRITE(6,*)
      WRITE(6,*)

100   DO 1000 I=1,NREACT
118     WRITE(6,*)'-----------------------------------------------------------------'
        WRITE(6,*) 'REACTION NUMBER ',I
        WRITE(6,*) 'HOW MANY REACTANTS IN THIS REACTION?'
        ACCEPT *, NR
        WRITE (12,*) NR


      DO 200 K=1,NR

        WRITE(6,*) 'INPUT THE NAME OF REACTANT ',K
        ACCEPT *, SPENAM

        DO 150 L=1,NSPECS

          IF (NAME(L)(1:1).EQ.' ') THEN
          NAME(L)=SPENAM
          J=L
          GOTO 170
          ELSEIF (NAME(L).EQ.SPENAM) THEN
          J=L
          GOTO 170
          ENDIF

150     CONTINUE

170     WRITE(6,*) 'INPUT STOICHIOMETRIC COEFICIENT OF ',NAME(J)
        ACCEPT *, STOREC(I,J)
        WRITE (12,1005) J,NAME(J),STOREC(I,J)

200   CONTINUE   !THE REACTANT LOOP FOR REACTION I
```

```
      WRITE(6,*) 'HOW MANY PRODUCTS IN THIS REACTION?'
      ACCEPT *, NP
      WRITE (12,*) NP

      DO 300 K=1,NP

       WRITE(6,*) 'INPUT THE NAME OF PRODUCT ',K
       ACCEPT *, SPENAM

       DO 250 L=1,NSPECS

       IF (NAME(L)(1:1).EQ.' ') THEN
       NAME(L)=SPENAM
       J=L
       GOTO 270
       ELSEIF (NAME(L).EQ.SPENAM) THEN
       J=L
       GOTO 270
       ENDIF
250    CONTINUE

270    WRITE(6,*) 'INPUT STOICHIOMETRIC COEFICIENT OF ',NAME(J)
       ACCEPT *, STOPRO(I,J)
       WRITE (12,1005) J,NAME(J),STOPRO(I,J)

300   CONTINUE   !THE PRODUCT LOOP FOR REACTION I

      WRITE(6,*) 'THE FORWARD RATE CONSTANT IS OF ARRHENIUS
FORM:'
      WRITE(6,*) 'kf = B*T**A*exp(-C/(RuT))'
      WRITE(6,*) 'FOR REACTION ',I,' INPUT THE COEFFICIENTS AS
FOLLOWS'
      WRITE(6,*) 'INPUT THE COEFFICIENT, B'
      ACCEPT *, BF(I)
      WRITE(6,*) 'INPUT THE EXPONENT, A'
      ACCEPT *, AF(I)
      WRITE(6,*) 'INPUT THE ACTIVATION ENERGY, C'
      ACCEPT *, CF(I)
      WRITE (12,*) BF(I),AF(I),CF(I)

      WRITE(6,*) '*******************************************'
      WRITE(6,*) 'ANY CHANGES TO REACTION ',I,' ?'
      WRITE(6,*) '1 = YES'
      ACCEPT *, IOK
      IF ( IOK .EQ. 1) GOTO 118
      WRITE(6,*) '*******************************************'

1000  CONTINUE   !THE REACTION LOOP

1005  FORMAT(7X,I8,5X,A8,5X,F12.6)
```

```
      WRITE (6,*)
      WRITE (6,*)
      WRITE (6,*) 'THESE DATA WILL BE WRITTEN TO A FILE NAMED'
      WRITE (6,*) 'NEWSET.DAT .  IF YOU WANT TO INPUT AND SAVE'
      WRITE (6,*) 'A NEW SET OF REACTIONS AND SPECIES, CHANGE '
      WRITE (6,*) 'THE NAME OF THE SET YOU JUST TYPED IN, ELSE'
      WRITE (6,*) 'IT WILL BE WRITTEN OVER WHEN NEW SET IS INPUT.'
      WRITE (6,*)
      WRITE (6,*)

C     ****************************************************************
C     This section reads in data from the JANAF thermochemical tables.
C     We read the species name, molecular weight, enthalpy of formation
C     at 298 K, and enthalpy vs. temperature.  The enthalpy vs. temperature
C     is then curve fit with a piecewise Hermite cubic polynomial. These
C     functions are later iterated to determine the temperature of the
C     reacting species.
C

 1070 WRITE(6,*) 'READING IN THERMOCHEMICAL DATA.  PLEASE BE
PATIENT.'

      CALL THERMO (NAME,TEMPH,ENTHAL,JKOUNT,MW,HFNOT,NSPECS)

      WRITE(6,*)'NOW CURVE FITTING THE JANAF DATA.'

      DO 1080 J=1,NSPECS

       IF(NAME(J) .EQ. 'X') GOTO 1080          !NO THERMOCHEMICAL DATA
FOR X

       NDATA = JKOUNT(J)                 !NUMBER OF (X,Y) PAIRS

       DO 1077 LL=1,JKOUNT(J)
        CTEMP(LL) = TEMPH(J,LL)
        CENTH(LL) = ENTHAL(J,LL)

1077   CONTINUE

       SPLINE = .FALSE.
       CALL PCHEZ(NDATA,CTEMP,CENTH,D,SPLINE,WK,LWK,IERR)

       IF (IERR .LT. 0) THEN
        WRITE(6,*) 'ERROR CALLING PCHEZ: IERR = ',IERR
        STOP
       ENDIF

        DO 1078 KK=1,NDATA
          HFITT(J,KK) = D(KK)

1078    CONTINUE
```

```
1080  CONTINUE

C     ********************************************************
C     Input the initial species concentrations
C
      HTOT = 0.
      WRITE(6,*) 'INPUT THE INITIAL TEMPERATURE'
      ACCEPT *, TEMP

      WRITE(6,*)
      WRITE(6,*) ' INITIAL SPECIES CONCENTRATIONS'
      WRITE(6,*)
      WRITE(6,*) 'INPUT THE NUMBER OF NONZERO INITIAL SPECIES'
      ACCEPT *, NZSPEC

      DO 1500 K=1,NZSPEC

      WRITE(6,*)
1100  WRITE(6,*) 'INPUT THE NAME OF SPECIES ', K
      ACCEPT *, SPENAM

       DO 1250 L=1,NSPECS
       IF (NAME(L)(1:1).EQ.' ') THEN
       WRITE(6,*) 'NO SUCH SPECIES EXISTS. TRY AGAIN, BOZO.'
       GOTO 1100
       ELSEIF (NAME(L).EQ.SPENAM) THEN
       J=L
       GOTO 1270
       ENDIF
1250   CONTINUE
      WRITE(6,*) 'NO SUCH SPECIES EXISTS. TRY AGAIN, BOZO.'
      GOTO 1100

1270  WRITE(6,*) 'INPUT THE INITIAL CONCENTRATION OF ',NAME(J)
      ACCEPT *, Y(J)

C     Determine the enthalpy of the initial reactants
C     Real Function ENTLPY  evaluates the
C     polynomial curve fit of H vs. T.
      IF (NAME(J) .EQ. 'X' ) GOTO 1500

      HTOT = HTOT + Y(J)*(ENTLPY(J,TEMP,TEMPH,ENTHAL,
     &    HFITT,JKOUNT) + HFNOT(J))

1500  CONTINUE
      WRITE(6,*) 'REACTANT ENTHALPY = ',HTOT

      WRITE(6,*) 'INPUT THE TOTAL TIME INTERVAL OF INTEREST'
      ACCEPT *, TIMAX
      WRITE(6,*) 'INPUT THE NUMBER OF TIMESTEPS TO PRINT'
      ACCEPT *, NPRINT
```

```
C       ******* PRINT INITIAL CONDITIONS ************************
        NCRICK =1
        TIM=0.
        CALL CRICKETDUMP(TIM,Y,NSPECS,NCRICK,NAME,TEMP)

C       *********  INITIALIZE THE STIFF SOLVER  ***********
C
        DELTAT = TIMAX/FLOAT(NPRINT - 1)/10.
        DELTA1 = DELTAT

        TIM=0.              !SET INITIAL TIME
        EWT=1.0E-15               !SET FOR PURE RELATIVE ERROR
        N=NSPECS            !SET N=NSPECS ODE'S TO SOLVE
        MSTATE=1            !SETS PARAMETER USED IN ODE SOLVER
        NROOT = 0           !SETS PARAMETER USED IN ODE SOLVER
        EPS=.001            !SETS RELATIVE ACCURACY REQUIRED
        MINT=2              !TELLS ODE SOLVER TO USE STIFF SOLVER (GEAR)
        TOUT = DELTAT       !THE TIME AT WHICH FIRST SOLUTION IS DESIRED
        LENW=7600           !SETS PARAMETER USED IN ODE SOLVER
        LENIW=200           !SETS PARAMETER USED IN ODE SOLVER
        NCRICK = 0          !SETS SWITCH IN DUMP ROUTINE
        KOUNTR=0
        LINIT=0
        IPRIN=0

 2000   CALL SDRIV2
(N,TIM,Y,FSUB,TOUT,MSTATE,NROOT,EPS,EWT,MINT,WORK,
     &LENW,IWORK,LENIW,FSUB,TEMP,KF,CF,AF,RU,
     &NREACT,NSPECS,BF,STOREC,STOPRO)

C       Compute the temperature based on the new Y(J)

        BFZ = 0.
        CFZ = 6000.
        RNOT = TEMP
        RE = .0001
        AE = .1
        IFLAG = 0

        DO 2120 JP=1,NSPECS
         IF ( Y(JP) .LT. 0.) THEN
         Y(JP) = 0.
         ENDIF
         YPT(JP) = Y(JP)
 2120   CONTINUE

        CALL FZERO(FTMP,BFZ,CFZ,RNOT,RE,AE,IFLAG)

        IF (IFLAG .NE. 1) WRITE(6,*) 'ERROR CODE = ',IFLAG
        TEMP = BFZ
CCC     WRITE(6,*) 'TEMP = ',TEMP
```

```
      IF(MSTATE .NE. 2) THEN
      WRITE(6,*) 'WHOA, BOGUS, MSTATE IS  ',MSTATE,' AT ',KOUNTR
      ENDIF

      KOUNTR = KOUNTR + 1
      IPRIN = IPRIN + 1
      LINIT =1

      IF (10*INT(FLOAT(IPRIN)/10.) .EQ. IPRIN)THEN
    CALL CRICKETDUMP(TIM,Y,NSPECS,NCRICK,NAME,TEMP)
      ENDIF

      TIM=TOUT
      TOUT = TOUT + DELTAT
      IF (TOUT .GT. TIMAX) THEN
      GOTO 5000
      ENDIF
      GOTO 2000

 5000 CLOSE(UNIT=11)
      CLOSE(UNIT=14)
      STOP
      END



C     ** SUBROUTINE "FSUB" COMPUTES THE RATE OF CHANGE OF ******
C     ** SPECIES J  FOR ALL REACTIONS  ************************

      SUBROUTINE
FSUB(N,TIM,Y,YDOT,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
     1          STOREC,STOPRO)

      INTEGER N,NREACT,NSPECS
      REAL Y(80),YDOT(80),UDOT(80,80),A(80),TIM,TEMP,KF(80),
     1  CF(80),AF(80),RU,BF(80),STOREC(80,80),
     1  STOPRO(80,80)

C     COMPUTE THE RATE CONSTANTS *********************************

      DO 50, I=1,NREACT

       KF(I) = BF(I)*TEMP**AF(I)*EXP(-CF(I)/(RU*TEMP))

 50   CONTINUE

C     COMPUTE THE RATE OF CHANGE OF SPECIES J  ********************

      DO 500 I=1,NREACT

       A(I) = 1.0
```

```
      DO 100 J=1,NSPECS

          IF ( Y(J) .LT. 0. ) Y(J) = 0.
          IF ( Y(J) .EQ. 0. ) GOTO 100
          IF (STOREC(I,J) .EQ. 0.) GOTO 100

       A(I) = A(I)*Y(J)**STOREC(I,J)

100     CONTINUE

       DO 200 J=1,NSPECS

       UDOT(I,J) = (STOPRO(I,J)-STOREC(I,J))*KF(I)*A(I)

200    CONTINUE
500   CONTINUE

C      ADD THE RATES FOR EACH SPECIES OVER THE REACTIONS ********

       DO 670, J=1,NSPECS          !ZERO THE YDOT ARRAY
         YDOT(J) = 0.
670   CONTINUE


       DO 800, J=1,NSPECS
        DO 700, I=1,NREACT
         YDOT(J) = YDOT(J) +UDOT(I,J)
CCC      WRITE(6,*) 'J = ',J,' YDOT = ',YDOT(J)
700     CONTINUE

800   CONTINUE

       RETURN
       END

C      ***************************************************
C      Real function FTMP(X) computes the enthalpy for a given
C      temperature.  FTMP(X) is used by FZERO to find the root,
C      i.e. the temperature, given the species concentrations

       REAL FUNCTION FTMP (TEMP)

       REAL HFITT(80,80),TEMP,HTOT,HFNOT(80),YPT(80),
     &  TEMPH(80,100),ENTHAL(80,100),ENTJ
       INTEGER J,JKOUNT(80),NSPECS
       CHARACTER*8 NAME(80)
       COMMON /COM1/ HFITT,HTOT,HFNOT,NAME,TEMPH,ENTHAL,
     &        JKOUNT,NSPECS,YPT
       FTMP=0.
       DO 1000 J=1,NSPECS

        IF (NAME(J) .EQ. 'X' ) GOTO 1000
```

```
      ENTJ = YPT(J)*(ENTLPY(J,TEMP,TEMPH,ENTHAL,HFITT,
     &    JKOUNT)+HFNOT(J))
      FTMP = FTMP + ENTJ

 1000 CONTINUE

      FTMP = FTMP - HTOT
CCC   WRITE(6,*) 'IN FTMP'
CCC   WRITE(6,*)'FTMP = ',FTMP
CCC   WRITE(6,*)'TEMP = ',TEMP

      RETURN
      END


C     ************************************************************
C     Function ENTLPY evaluates the Hermite cubic polynomial
C     curve fit of enthalpy vs. temperature for species J.

      REAL FUNCTION ENTLPY(J,TEMP,TEMPH,ENTHAL,HFITT,JKOUNT)

      INTEGER J,JKOUNT(80),IERR,NVAL
      REAL TEMP, HFITT(80,80),TEMPH(80,100),ENTHAL(80,100),
     &  OUT(1),DOUT(1),XTEMP(80),FENTH(80),DHERM(80),XVAL(1)

      NDP = JKOUNT(J)
      DO 200 I=1,NDP
       XTEMP(I) = TEMPH(J,I)
       FENTH(I) = ENTHAL(J,I)
       DHERM(I) = HFITT(J,I)
 200  CONTINUE

      NVAL = 1
      XVAL(1) = TEMP
      CALL
PCHEV(NDP,XTEMP,FENTH,DHERM,NVAL,XVAL,OUT,DOUT,IERR)

      ENTLPY = OUT(1)

      RETURN
      END

C     ************************************************************
C     Subroutine     CRICKETDUMP  dumps the
C     data into a file that can be exported to CRICEKTGRAPH.

      SUBROUTINE CRICKETDUMP(TIM,Y,NSPECS,NCRICK,NAME,TEMP)
      INTEGER NSPECS,NCRICK
      REAL TIM, Y(*)
      CHARACTER*8 NAME(*)

      IF (NCRICK .EQ. 1) THEN
```

```
      WRITE(11,'(1H*)')   !TELL CRICKETGRAPH COLUMN NAMES ARE IN
FILE
      WRITE(14,'(1H*)')
      WRITE(11,'(9999(A11,A1))')
     & 'TIME',char(9),(NAME(I),char(9),i=1,NSPECS-1),NAME(NSPECS)
      WRITE(14,'(9999(A11,A1))')
     & 'TIME',char(9),'TEMP'
      ENDIF

      WRITE(11,'(9999(E,A1))') TIM,char(9),(Y(I),char(9),I=1,NSPECS-1),
    &Y(NSPECS)
      WRITE(14,'(9999(E,A1))') TIM,char(9),TEMP

    RETURN
      END

C     ************************************************************
C     Subroutine STORE writes the reaction set to the file
C     NEWSET.DAT

      SUBROUTINE
STORE(NREACT,NSPECS,NAME,STOREC,STOPRO,AF,BF,CF)

      REAL AF(80),BF(80),CF(80),STOREC(80,80),
     1 STOPRO(80,80)
      INTEGER NREACT,NSPECS
      CHARACTER*8 NAME(80)

      READ (12,*,ERR=1200) NREACT

      READ (12,*,ERR=1200) NSPECS

C     INITIALIZE "NAME" ARRAY
      DO 50 K=1,NSPECS+1
      NAME(K) = ' '
50    CONTINUE

100   DO 1000 I=1,NREACT

        READ (12,*,ERR=1200) NR

        DO 200 K=1,NR

          READ (12,FMT=1005,ERR=1200) J,NAME(J),STOREC(I,J)

200     CONTINUE          !THE REACTANT LOOP FOR REACTION I

        READ (12,*,ERR=1200) NP

        DO 300 K=1,NP

          READ (12,FMT=1005,ERR=1200) J,NAME(J),STOPRO(I,J)
```

```
300     CONTINUE          !THE PRODUCT LOOP FOR REACTION I

        READ (12,*,ERR=1200) BF(I),AF(I),CF(I)

1000  CONTINUE   !THE REACTION LOOP

1005  FORMAT(7X,I8,5X,A8,5X,F12.6)

      GOTO 1300

1200  WRITE(6,*) 'INPUT ERROR IN SUBROUTINE STORE.'
      WRITE(6,*) 'ERROR READING IN THE DATA.'
      CLOSE(UNIT=12)
      CLOSE(UNIT=11)
      STOP

1300  CLOSE(UNIT=12)
      RETURN
      END

C     ************************************************************
C     Subroutine THERMO reads in JANAF thermochemical data

      SUBROUTINE  THERMO
(NAME,TEMPH,ENTHAL,JKOUNT,MW,HFNOT,NSPECS)

      REAL TEMPH(80,100),ENTHAL(80,100),MW(80),HFNOT(80)
      INTEGER JKOUNT(80),NSPECS
      CHARACTER*8 NAME(*),SPENAM

      DO 250 L=1,100

       READ(13,FMT=200,ERR=5000) SPENAM,MOLWET,ENTHFM
200    FORMAT (1X,A8,1X,2F10.5)
       IF(SPENAM .EQ. 'EOF') GOTO 5200

            DO 204 J=1,NSPECS
         IF(SPENAM .EQ. NAME(J) ) GOTO 270
204         CONTINUE

       DO 220 K=1,80

        READ(13,FMT=205,ERR=5000) TDUM,ENDUM
205     FORMAT(1X,F5.0,4X,F10.4)
        IF (TDUM .LT. 0.) GOTO 250

220    CONTINUE
       GOTO 250

270   MW(J) = MOLWET
      HFNOT(J) = ENTHFM
```

```
      DO 275 JL=1,80
       READ(13,FMT=205,ERR=5000) TDUM,ENDUM

       IF (TDUM .LT. 0.) THEN
        JKOUNT(J) = JL -1
        GOTO 250
       ENDIF
       TEMPH(J,JL) = TDUM
       ENTHAL(J,JL) = ENDUM
275    CONTINUE

250    CONTINUE

       GOTO 5200

5000   WRITE(6,*) 'ERROR INPUTING THE THERMOCHEMICAL DATA'
       WRITE(6,*)   'SEE SUBROUTINE THERMO OR FILE JANAF'
       CLOSE(UNIT=13)
       STOP

5200   CLOSE (UNIT = 13)
       RETURN
       END

C      ************************************************************
      SUBROUTINE PCHEZ(N,X,F,D,SPLINE,WK,LWK,IERR)
C***BEGIN PROLOGUE  PCHEZ
C***DATE WRITTEN  870821  (YYMMDD)
C***REVISION DATE  870908  (YYMMDD)
C***CATEGORY NO.  E1B
C***KEYWORDS  CUBIC HERMITE MONOTONE INTERPOLATION, SPLINE
C         INTERPOLATION, EASY TO USE PIECEWISE CUBIC INTERPOLATION
C***AUTHOR  KAHANER, D.K., (NBS)
C         SCIENTIFIC COMPUTING DIVISION
C         NATIONAL BUREAU OF STANDARDS
C         GAITHERSBURG, MARYLAND 20899
C         (301) 975-3808
C***PURPOSE  Easy to use spline or cubic Hermite interpolation.
C***DESCRIPTION
C
C       PCHEZ:  Piecewise Cubic Interpolation, Easy to Use.
C
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C
C    Sets derivatives for spline (two continuous derivatives) or
C    Hermite cubic (one continuous derivative) interpolation.
C    Spline interpolation is smoother, but may not "look" right if the
C    data contains both "steep" and "flat" sections.  Hermite cubics
C    can produce a "visually pleasing" and monotone interpolant to
```

```
C    monotone data. This is an easy to use driver for the routines
C    by F. N. Fritsch in reference (4) below. Various boundary
C    conditions are set to default values by PCHEZ. Many other choices
C    are available in the subroutines PCHIC, PCHIM and PCHSP.
C
C    Use PCHEV to evaluate the resulting function and its derivative.
C
C-----------------------------------------------------------------------
C
C Calling sequence:  CALL  PCHEZ (N, X, F, D, SPLINE, WK, LWK, IERR)
C
C    INTEGER  N, IERR,  LWK
C    REAL  X(N), F(N), D(N), WK(*)
C    LOGICAL SPLINE
C
C Parameters:
C
C    N -- (input) number of data points.  (Error return if N.LT.2 .)
C         If N=2, simply does linear interpolation.
C
C    X -- (input) real array of independent variable values.  The
C         elements of X must be strictly increasing:
C              X(I-1) .LT. X(I),  I = 2(1)N.
C         (Error return if not.)
C
C    F -- (input) real array of dependent variable values to be inter-
C         polated.  F(I) is value corresponding to X(I).
C
C    D -- (output) real array of derivative values at the data points.
C
C    SPLINE -- (input) logical variable to specify if the interpolant
C         is to be a spline with two continuous derivaties
C         (set SPLINE=.TRUE.) or a Hermite cubic interpolant with one
C         continuous derivative (set SPLINE=.FALSE.).
C       Note: If SPLINE=.TRUE. the interpolating spline satisfies the
C         default "not-a-knot" boundary condition, with a continuous
C         third derivative at X(2) and X(N-1). See reference (3).
C           If SPLINE=.FALSE. the interpolating Hermite cubic will be
C         monotone if the input data is monotone. Boundary conditions are
C         computed from the derivative of a local quadratic unless this
C         alters monotonicity.
C
C    WK -- (scratch) real work array, which must be declared by the calling
C         program to be at least 2*N if SPLINE is .TRUE. and not used
C         otherwise.
C
C    LWK -- (input) length of work array WK. (Error return if
C         LWK.LT.2*N and SPLINE is .TRUE., not checked otherwise.)
C
C    IERR -- (output) error flag.
C         Normal return:
C           IERR = 0  (no errors).
```

```
C        Warning error:
C          IERR.GT.0  (can only occur when SPLINE=.FALSE.) means that
C            IERR switches in the direction of monotonicity were detected.
C            When SPLINE=.FALSE.,  PCHEZ guarantees that if the input
C            data is monotone, the interpolant will be too. This warning
C            is to alert you to the fact that the input data was not
C            monotone.
C        "Recoverable" errors:
C          IERR = -1  if N.LT.2 .
C          IERR = -3  if the X-array is not strictly increasing.
C          IERR = -7  if LWK is less than 2*N and SPLINE is .TRUE.
C         (The D-array has not been changed in any of these cases.)
C          NOTE:  The above errors are checked in the order listed,
C              and following arguments have **NOT** been validated.
C
C -----------------------------------------------------------------------
C***REFERENCES  1. F.N.FRITSCH AND R.E.CARLSON, 'MONOTONE
PIECEWISE
C              CUBIC INTERPOLATION,' SIAM J.NUMER.ANAL. 17, 2 (APRIL
C              1980), 238-246.
C            2. F.N.FRITSCH AND J.BUTLAND, 'A METHOD FOR CONSTRUCTING
C             LOCAL MONOTONE PIECEWISE CUBIC INTERPOLANTS,' LLNL
C             PREPRINT UCRL-87559 (APRIL 1982).
C            3. CARL DE BOOR, A PRACTICAL GUIDE TO SPLINES, SPRINGER-
C              VERLAG (NEW YORK, 1978).  (ESP. CHAPTER IV, PP.49-62.)
C            4. F.N.FRITSCH, 'PIECEWISE CUBIC HERMITE INTERPOLATION
C             PACKAGE, FINAL SPECIFICATIONS', LAWRENCE LIVERMORE
C            NATIONAL LABORATORY, COMPUTER DOCUMENTATION UCID-
30194,
C              AUGUST 1982.
C***ROUTINES CALLED  PCHIM,PCHSP
C***END PROLOGUE  PCHEZ
      INTEGER  N, LWK, IERR
      REAL  X(N), F(N), D(N), WK(LWK)
      LOGICAL SPLINE
C
C  DECLARE LOCAL VARIABLES.
C
      INTEGER IC(2), INCFD
      REAL  VC(2)
      DATA IC(1) /0/
      DATA IC(2) /0/
      DATA INCFD /1/
C
C
C***FIRST EXECUTABLE STATEMENT  PCHEZ
C
      IF ( SPLINE ) THEN
        CALL  PCHSP (IC, VC, N, X, F, D, INCFD, WK, LWK, IERR)
      ELSE
        CALL  PCHIM (N, X, F, D, INCFD, IERR)
      ENDIF
```

```
C
C  ERROR CONDITIONS ALREADY CHECKED IN PCHSP OR PCHIM

      RETURN
C------------ LAST LINE OF PCHEZ FOLLOWS ----------------------------
      END
      SUBROUTINE PCHIM(N,X,F,D,INCFD,IERR)
C***BEGIN PROLOGUE  PCHIM
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C***END PROLOGUE  PCHIM
C
C  DECLARE ARGUMENTS.
C
      INTEGER  N, INCFD, IERR
      REAL  X(N), F(INCFD,N), D(INCFD,N)
C
C  DECLARE LOCAL VARIABLES.
C
      INTEGER  I, NLESS1
      REAL  DEL1, DEL2, DMAX, DMIN, DRAT1, DRAT2, DSAVE,
     *      H1, H2, HSUM, HSUMT3, THREE, W1, W2, ZERO
      REAL  PCHST
      DATA  ZERO /0./,  THREE /3./
C
C  VALIDITY-CHECK ARGUMENTS.
C
C***FIRST EXECUTABLE STATEMENT  PCHIM
      IF ( N.LT.2 )  GO TO 5001
      IF ( INCFD.LT.1 )  GO TO 5002
      DO 1  I = 2, N
        IF ( X(I).LE.X(I-1) )  GO TO 5003
    1 CONTINUE
C
C  FUNCTION DEFINITION IS OK, GO ON.
C
      IERR = 0
      NLESS1 = N - 1
      H1 = X(2) - X(1)
      DEL1 = (F(1,2) - F(1,1))/H1
      DSAVE = DEL1
C
C  SPECIAL CASE N=2 -- USE LINEAR INTERPOLATION.
C
      IF (NLESS1 .GT. 1)  GO TO 10
      D(1,1) = DEL1
      D(1,N) = DEL1
      GO TO 5000
C
```

```
C  NORMAL CASE  (N .GE. 3).
C
  10 CONTINUE
     H2 = X(3) - X(2)
     DEL2 = (F(1,3) - F(1,2))/H2
C
C  SET D(1) VIA NON-CENTERED THREE-POINT FORMULA, ADJUSTED TO BE
C     SHAPE-PRESERVING.
C
     HSUM = H1 + H2
     W1 = (H1 + HSUM)/HSUM
     W2 = -H1/HSUM
     D(1,1) = W1*DEL1 + W2*DEL2
     IF ( PCHST(D(1,1),DEL1) .LE. ZERO)  THEN
       D(1,1) = ZERO
     ELSE IF ( PCHST(DEL1,DEL2) .LT. ZERO)  THEN
C       NEED DO THIS CHECK ONLY IF MONOTONICITY SWITCHES.
       DMAX = THREE*DEL1
       IF (ABS(D(1,1)) .GT. ABS(DMAX))  D(1,1) = DMAX
     ENDIF
C
C  LOOP THROUGH INTERIOR POINTS.
C
     DO 50  I = 2, NLESS1
       IF (I .EQ. 2)  GO TO 40
C
       H1 = H2
       H2 = X(I+1) - X(I)
       HSUM = H1 + H2
       DEL1 = DEL2
       DEL2 = (F(1,I+1) - F(1,I))/H2
  40   CONTINUE
C
C      SET D(I)=0 UNLESS DATA ARE STRICTLY MONOTONIC.
C
       D(1,I) = ZERO
       IF ( PCHST(DEL1,DEL2) )  42, 41, 45
C
C      COUNT NUMBER OF CHANGES IN DIRECTION OF MONOTONICITY.
C
  41   CONTINUE
       IF (DEL2 .EQ. ZERO)  GO TO 50
       IF ( PCHST(DSAVE,DEL2) .LT. ZERO)  IERR = IERR + 1
       DSAVE = DEL2
       GO TO 50
C
  42   CONTINUE
       IERR = IERR + 1
       DSAVE = DEL2
       GO TO 50
C
C      USE BRODLIE MODIFICATION OF BUTLAND FORMULA.
```

```
C
  45   CONTINUE
       HSUMT3 = HSUM+HSUM+HSUM
       W1 = (HSUM + H1)/HSUMT3
       W2 = (HSUM + H2)/HSUMT3
       DMAX = AMAX1( ABS(DEL1), ABS(DEL2) )
       DMIN = AMIN1( ABS(DEL1), ABS(DEL2) )
       DRAT1 = DEL1/DMAX
       DRAT2 = DEL2/DMAX
       D(1,I) = DMIN/(W1*DRAT1 + W2*DRAT2)
C
  50 CONTINUE
C
C  SET D(N) VIA NON-CENTERED THREE-POINT FORMULA, ADJUSTED TO BE
C     SHAPE-PRESERVING.
C
      W1 = -H2/HSUM
      W2 = (H2 + HSUM)/HSUM
      D(1,N) = W1*DEL1 + W2*DEL2
      IF ( PCHST(D(1,N),DEL2) .LE. ZERO)  THEN
        D(1,N) = ZERO
      ELSE IF ( PCHST(DEL1,DEL2) .LT. ZERO)  THEN
C       NEED DO THIS CHECK ONLY IF MONOTONICITY SWITCHES.
        DMAX = THREE*DEL2
        IF (ABS(D(1,N)) .GT. ABS(DMAX))  D(1,N) = DMAX
      ENDIF
C
C  NORMAL RETURN.
C
 5000 CONTINUE
      RETURN
C
C  ERROR RETURNS.
C
 5001 CONTINUE
C     N.LT.2 RETURN.
      IERR = -1
      CALL XERROR ('PCHIM -- NUMBER OF DATA POINTS LESS THAN TWO'
     *          , 44, IERR, 1)
      RETURN
C
 5002 CONTINUE
C     INCFD.LT.1 RETURN.
      IERR = -2
      CALL XERROR ('PCHIM -- INCREMENT LESS THAN ONE'
     *          , 32, IERR, 1)
      RETURN
C
 5003 CONTINUE
C     X-ARRAY NOT STRICTLY INCREASING.
      IERR = -3
      CALL XERROR ('PCHIM -- X-ARRAY NOT STRICTLY INCREASING'
```

```
     *        , 40, IERR, 1)
      RETURN
C------------ LAST LINE OF PCHIM FOLLOWS ----------------------------
      END
      REAL FUNCTION PCHST(ARG1,ARG2)
C***BEGIN PROLOGUE  PCHST
C***REFER TO  PCHCE,PCHCI,PCHCS,PCHIM
C***END PROLOGUE  PCHST
      REAL  ARG1, ARG2
C
C DECLARE LOCAL VARIABLES.
C
      REAL  ONE, ZERO
      DATA  ZERO /0./,  ONE /1./
C
C PERFORM THE TEST.
C
C***FIRST EXECUTABLE STATEMENT  PCHST
      PCHST = SIGN(ONE,ARG1) * SIGN(ONE,ARG2)
      IF ((ARG1.EQ.ZERO) .OR. (ARG2.EQ.ZERO))  PCHST = ZERO
C
      RETURN
C------------ LAST LINE OF PCHST FOLLOWS ----------------------------
      END
      SUBROUTINE PCHSP(IC,VC,N,X,F,D,INCFD,WK,NWK,IERR)
C***BEGIN PROLOGUE  PCHSP
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C            Prentice Hall 1988
C***END PROLOGUE  PCHSP
C
C DECLARE ARGUMENTS.
C
      INTEGER  IC(2), N, INCFD, NWK, IERR
      REAL  VC(2), X(N), F(INCFD,N), D(INCFD,N), WK(2,N)
C
C DECLARE LOCAL VARIABLES.
C
      INTEGER  IBEG, IEND, INDEX, J, NM1
      REAL  G, HALF, ONE, STEMP(3), THREE, TWO, XTEMP(4), ZERO
      REAL  PCHDF
C
      DATA  ZERO /0./,  HALF /0.5/,  ONE /1./,  TWO /2./,  THREE /3./
C
C VALIDITY-CHECK ARGUMENTS.
C
C***FIRST EXECUTABLE STATEMENT  PCHSP
      IF ( N.LT.2 )  GO TO 5001
      IF ( INCFD.LT.1 )  GO TO 5002
      DO 1  J = 2, N
```

```
      IF ( X(J).LE.X(J-1) )  GO TO 5003
    1 CONTINUE
C
      IBEG = IC(1)
      IEND = IC(2)
      IERR = 0
      IF ( (IBEG.LT.0).OR.(IBEG.GT.4) )  IERR = IERR - 1
      IF ( (IEND.LT.0).OR.(IEND.GT.4) )  IERR = IERR - 2
      IF ( IERR.LT.0 )  GO TO 5004
C
C  FUNCTION DEFINITION IS OK -- GO ON.
C
      IF ( NWK .LT. 2*N )  GO TO 5007
C
C  COMPUTE FIRST DIFFERENCES OF X SEQUENCE AND STORE IN WK(1,.).
ALSO,
C  COMPUTE FIRST DIVIDED DIFFERENCE OF DATA AND STORE IN WK(2,.).
      DO 5  J=2,N
        WK(1,J) = X(J) - X(J-1)
        WK(2,J) = (F(1,J) - F(1,J-1))/WK(1,J)
    5 CONTINUE
C
C  SET TO DEFAULT BOUNDARY CONDITIONS IF N IS TOO SMALL.
C
      IF ( IBEG.GT.N )  IBEG = 0
      IF ( IEND.GT.N )  IEND = 0
C
C  SET UP FOR BOUNDARY CONDITIONS.
C
      IF ( (IBEG.EQ.1).OR.(IBEG.EQ.2) )  THEN
        D(1,1) = VC(1)
      ELSE IF (IBEG .GT. 2)  THEN
C       PICK UP FIRST IBEG POINTS, IN REVERSE ORDER.
        DO 10  J = 1, IBEG
          INDEX = IBEG-J+1
C          INDEX RUNS FROM IBEG DOWN TO 1.
          XTEMP(J) = X(INDEX)
          IF (J .LT. IBEG)  STEMP(J) = WK(2,INDEX)
   10   CONTINUE
C              ---------------------------------
        D(1,1) = PCHDF (IBEG, XTEMP, STEMP, IERR)
C              ---------------------------------
        IF (IERR .NE. 0)  GO TO 5009
        IBEG = 1
      ENDIF
C
      IF ( (IEND.EQ.1).OR.(IEND.EQ.2) )  THEN
        D(1,N) = VC(2)
      ELSE IF (IEND .GT. 2)  THEN
C       PICK UP LAST IEND POINTS.
        DO 15  J = 1, IEND
          INDEX = N-IEND+J
```

```
C         INDEX RUNS FROM N+1-IEND UP TO N.
          XTEMP(J) = X(INDEX)
          IF (J .LT. IEND)  STEMP(J) = WK(2,INDEX+1)
   15   CONTINUE
C              --------------------------------
        D(1,N) = PCHDF (IEND, XTEMP, STEMP, IERR)
C              --------------------------------
        IF (IERR .NE. 0)  GO TO 5009
        IEND = 1
      ENDIF
C
C --------------------( BEGIN CODING FROM CUBSPL )--------------------
C
C  **** A TRIDIAGONAL LINEAR SYSTEM FOR THE UNKNOWN SLOPES S(J)
OF
C  F  AT X(J), J=1,...,N, IS GENERATED AND THEN SOLVED BY GAUSS ELIM-
C  INATION, WITH S(J) ENDING UP IN D(1,J), ALL J.
C     WK(1,.) AND WK(2,.) ARE USED FOR TEMPORARY STORAGE.
C
C  CONSTRUCT FIRST EQUATION FROM FIRST BOUNDARY CONDITION, OF
THE FORM
C           WK(2,1)*S(1) + WK(1,1)*S(2) = D(1,1)
C
      IF (IBEG .EQ. 0)  THEN
        IF (N .EQ. 2)  THEN
C         NO CONDITION AT LEFT END AND N = 2.
          WK(2,1) = ONE
          WK(1,1) = ONE
          D(1,1) = TWO*WK(2,2)
        ELSE
C         NOT-A-KNOT CONDITION AT LEFT END AND N .GT. 2.
          WK(2,1) = WK(1,3)
          WK(1,1) = WK(1,2) + WK(1,3)
          D(1,1) =((WK(1,2) + TWO*WK(1,1))*WK(2,2)*WK(1,3)
     *                   + WK(1,2)**2*WK(2,3)) / WK(1,1)
        ENDIF
      ELSE IF (IBEG .EQ. 1)  THEN
C       SLOPE PRESCRIBED AT LEFT END.
        WK(2,1) = ONE
        WK(1,1) = ZERO
      ELSE
C       SECOND DERIVATIVE PRESCRIBED AT LEFT END.
        WK(2,1) = TWO
        WK(1,1) = ONE
        D(1,1) = THREE*WK(2,2) - HALF*WK(1,2)*D(1,1)
      ENDIF
C
C  IF THERE ARE INTERIOR KNOTS, GENERATE THE CORRESPONDING
EQUATIONS AND
C  CARRY OUT THE FORWARD PASS OF GAUSS ELIMINATION, AFTER WHICH
THE J-TH
C  EQUATION READS    WK(2,J)*S(J) + WK(1,J)*S(J+1) = D(1,J).
```

```
C
      NM1 = N-1
      IF (NM1 .GT. 1)  THEN
        DO 20 J=2,NM1
          IF (WK(2,J-1) .EQ. ZERO)  GO TO 5008
          G = -WK(1,J+1)/WK(2,J-1)
          D(1,J) = G*D(1,J-1)
     *             + THREE*(WK(1,J)*WK(2,J+1) + WK(1,J+1)*WK(2,J))
          WK(2,J) = G*WK(1,J-1) + TWO*(WK(1,J) + WK(1,J+1))
   20   CONTINUE
      ENDIF
C
C  CONSTRUCT LAST EQUATION FROM SECOND BOUNDARY CONDITION, OF
THE FORM
C          (-G*WK(2,N-1))*S(N-1) + WK(2,N)*S(N) = D(1,N)
C
C     IF SLOPE IS PRESCRIBED AT RIGHT END, ONE CAN GO DIRECTLY TO
BACK-
C     SUBSTITUTION, SINCE ARRAYS HAPPEN TO BE SET UP JUST RIGHT FOR
IT
C     AT THIS POINT.
      IF (IEND .EQ. 1)  GO TO 30
C
      IF (IEND .EQ. 0)  THEN
        IF (N.EQ.2 .AND. IBEG.EQ.0)  THEN
C         NOT-A-KNOT AT RIGHT ENDPOINT AND AT LEFT ENDPOINT AND N =
2.
          D(1,2) = WK(2,2)
          GO TO 30
        ELSE IF ((N.EQ.2) .OR. (N.EQ.3 .AND. IBEG.EQ.0))  THEN
C         EITHER (N=3 AND NOT-A-KNOT ALSO AT LEFT) OR (N=2 AND *NOT*
C         NOT-A-KNOT AT LEFT END POINT).
          D(1,N) = TWO*WK(2,N)
          WK(2,N) = ONE
          IF (WK(2,N-1) .EQ. ZERO)  GO TO 5008
          G = -ONE/WK(2,N-1)
        ELSE
C         NOT-A-KNOT AND N .GE. 3, AND EITHER N.GT.3 OR  ALSO NOT-A-
C         KNOT AT LEFT END POINT.
          G = WK(1,N-1) + WK(1,N)
C         DO NOT NEED TO CHECK FOLLOWING DENOMINATORS (X-
DIFFERENCES).
          D(1,N) = ((WK(1,N)+TWO*G)*WK(2,N)*WK(1,N-1)
     *             + WK(1,N)**2*(F(1,N-1)-F(1,N-2))/WK(1,N-1))/G
          IF (WK(2,N-1) .EQ. ZERO)  GO TO 5008
          G = -G/WK(2,N-1)
          WK(2,N) = WK(1,N-1)
        ENDIF
      ELSE
C       SECOND DERIVATIVE PRESCRIBED AT RIGHT ENDPOINT.
        D(1,N) = THREE*WK(2,N) + HALF*WK(1,N)*D(1,N)
        WK(2,N) = TWO
```

```
        IF (WK(2,N-1) .EQ. ZERO)  GO TO 5008
        G = -ONE/WK(2,N-1)
      ENDIF
C
C  COMPLETE FORWARD PASS OF GAUSS ELIMINATION.
C
      WK(2,N) = G*WK(1,N-1) + WK(2,N)
      IF (WK(2,N) .EQ. ZERO)  GO TO 5008
      D(1,N) = (G*D(1,N-1) + D(1,N))/WK(2,N)
C
C  CARRY OUT BACK SUBSTITUTION
C
   30 CONTINUE
      DO 40 J=NM1,1,-1
        IF (WK(2,J) .EQ. ZERO)  GO TO 5008
        D(1,J) = (D(1,J) - WK(1,J)*D(1,J+1))/WK(2,J)
   40 CONTINUE
C ----------( END  CODING FROM CUBSPL )----------
C
C  NORMAL RETURN.
C
      RETURN
C
C  ERROR RETURNS.
C
 5001 CONTINUE
C     N.LT.2 RETURN.
      IERR = -1
      CALL XERROR ('PCHSP -- NUMBER OF DATA POINTS LESS THAN TWO'
     *           , 44, IERR, 1)
      RETURN
C
 5002 CONTINUE
C     INCFD.LT.1 RETURN.
      IERR = -2
      CALL XERROR ('PCHSP -- INCREMENT LESS THAN ONE'
     *           , 32, IERR, 1)
      RETURN
C
 5003 CONTINUE
C     X-ARRAY NOT STRICTLY INCREASING.
      IERR = -3
      CALL XERROR ('PCHSP -- X-ARRAY NOT STRICTLY INCREASING'
     *           , 40, IERR, 1)
      RETURN
C
 5004 CONTINUE
C     IC OUT OF RANGE RETURN.
      IERR = IERR - 3
      CALL XERROR ('PCHSP -- IC OUT OF RANGE'
     *           , 24, IERR, 1)
      RETURN
```

```
C
 5007 CONTINUE
C    NWK TOO SMALL RETURN.
     IERR = -7
     CALL XERROR ('PCHSP -- WORK ARRAY TOO SMALL'
    *         , 29, IERR, 1)
     RETURN
C
 5008 CONTINUE
C    SINGULAR SYSTEM.
C  *** THEORETICALLY, THIS CAN ONLY OCCUR IF SUCCESSIVE X-VALUES
***
C  *** ARE EQUAL, WHICH SHOULD ALREADY HAVE BEEN CAUGHT (IERR=-
3). ***
     IERR = -8
     CALL XERROR ('PCHSP -- SINGULAR LINEAR SYSTEM'
    *         , 31, IERR, 1)
     RETURN
C
 5009 CONTINUE
C    ERROR RETURN FROM PCHDF.
C  *** THIS CASE SHOULD NEVER OCCUR ***
     IERR = -9
     CALL XERROR ('PCHSP -- ERROR RETURN FROM PCHDF'
    *         , 32, IERR, 1)
     RETURN
C------------ LAST LINE OF PCHSP FOLLOWS ----------------------------
     END
     REAL FUNCTION PCHDF(K,X,S,IERR)
C***BEGIN PROLOGUE  PCHDF
C***REFER TO  PCHCE,PCHSP
C***END PROLOGUE  PCHDF
     INTEGER  K, IERR
     REAL  X(K), S(K)
C
C  DECLARE LOCAL VARIABLES.
C
     INTEGER  I, J
     REAL  VALUE, ZERO
     DATA  ZERO /0./
C
C  CHECK FOR LEGAL VALUE OF K.
C
C***FIRST EXECUTABLE STATEMENT  PCHDF
     IF (K .LT. 3)  GO TO 5001
C
C  COMPUTE COEFFICIENTS OF INTERPOLATING POLYNOMIAL.
C
     DO 10  J = 2, K-1
       DO 9  I = 1, K-J
         S(I) = (S(I+1)-S(I))/(X(I+J)-X(I))
  9    CONTINUE
```

```
   10 CONTINUE
C
C  EVALUATE DERIVATIVE AT X(K).
C
      VALUE = S(1)
      DO 20  I = 2, K-1
        VALUE = S(I) + VALUE*(X(K)-X(I))
   20 CONTINUE
C
C  NORMAL RETURN.
C
      IERR = 0
      PCHDF = VALUE
      RETURN
C
C  ERROR RETURN.
C
 5001 CONTINUE
C     K.LT.3 RETURN.
      IERR = -1
      CALL XERROR ('PCHDF -- K LESS THAN THREE'
     *          , 26, IERR, 1)
      PCHDF = ZERO
      RETURN
C------------- LAST LINE OF PCHDF FOLLOWS -----------------------------
      END

C       ***********************************************************
      SUBROUTINE PCHEV(N,X,F,D,NVAL,XVAL,FVAL,DVAL,IERR)
C***BEGIN PROLOGUE  PCHEV
C***DATE WRITTEN  870828  (YYMMDD)
C***REVISION DATE  870828  (YYMMDD)
C***CATEGORY NO.  E3,H1
C***KEYWORDS  CUBIC HERMITE OR SPLINE DIFFERENTIATION,CUBIC
HERMITE
C           EVALUATION,EASY TO USE SPLINE OR CUBIC HERMITE EVALUATOR
C***AUTHOR  KAHANER, D.K., (NBS)
C           SCIENTIFIC COMPUTING DIVISION
C           NATIONAL BUREAU OF STANDARDS
C           ROOM A161, TECHNOLOGY BUILDING
C           GAITHERSBURG, MARYLAND 20899
C           (301) 975-3808
C***PURPOSE  Evaluates the function and first derivative of a piecewise
C           cubic Hermite or spline function at an array of points XVAL,
C           easy to use.
C***DESCRIPTION
C
C        PCHEV:  Piecewise Cubic Hermite or Spline Derivative Evaluator,
C                Easy to Use.
C
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
```

```
C            Prentice Hall 1988
C
C     Evaluates the function and first derivative of the cubic Hermite
C     or spline function defined by  N, X, F, D, at the array of points XVAL.
C
C     This is an easy to use driver for the routines by F.N. Fritsch
C     described in reference (2) below. Those also have other capabilities.
C
C-----------------------------------------------------------------------
C
C Calling sequence: CALL  PCHEV (N, X, F, D, NVAL, XVAL, FVAL, DVAL, IERR)
C
C     INTEGER  N, NVAL, IERR
C     REAL  X(N), F(N), D(N), XVAL(NVAL), FVAL(NVAL), DVAL(NVAL)
C
C  Parameters:
C
C     N -- (input) number of data points.  (Error return if N.LT.2 .)
C
C     X -- (input) real array of independent variable values.  The
C          elements of X must be strictly increasing:
C            X(I-1) .LT. X(I),  I = 2(1)N. (Error return if not.)
C
C     F -- (input) real array of function values.  F(I) is
C          the value corresponding to X(I).
C
C     D -- (input) real array of derivative values.  D(I) is
C          the value corresponding to X(I).
C
C  NVAL -- (input) number of points at which the functions are to be
C          evaluated. ( Error return if NVAL.LT.1 )
C
C  XVAL -- (input) real array of points at which the functions are to
C          be evaluated.
C
C          NOTES:
C          1. The evaluation will be most efficient if the elements
C             of XVAL are increasing relative to X;
C             that is,   XVAL(J) .GE. X(I)
C             implies    XVAL(K) .GE. X(I),  all K.GE.J .
C          2. If any of the XVAL are outside the interval [X(1),X(N)],
C             values are extrapolated from the nearest extreme cubic,
C             and a warning error is returned.
C
C  FVAL -- (output) real array of values of the cubic Hermite function
C          defined by  N, X, F, D  at the points  XVAL.
C
C  DVAL -- (output) real array of values of the first derivative of
C          the same function at the points  XVAL.
C
C  IERR -- (output) error flag.
C          Normal return:
```

```
C           IERR = 0  (no errors).
C         Warning error:
C           IERR.GT.0  means that extrapolation was performed at
C              IERR points.
C         "Recoverable" errors:
C           IERR = -1  if N.LT.2 .
C           IERR = -3  if the X-array is not strictly increasing.
C           IERR = -4  if NVAL.LT.1 .
C         (Output arrays have not been changed in any of these cases.)
C           NOTE:  The above errors are checked in the order listed,
C              and following arguments have **NOT** been validated.
C           IERR = -5  if an error has occurred in the lower-level
C                    routine CHFDV.  NB: this should never happen.
C                    Notify the author **IMMEDIATELY** if it does.
C
C -----------------------------------------------------------------------
C***REFERENCES  1. F.N.FRITSCH AND R.E.CARLSON, 'MONOTONE
PIECEWISE
C              CUBIC INTERPOLATION,' SIAM J.NUMER.ANAL. 17, 2 (APRIL
C              1980), 238-246.
C            2. F.N.FRITSCH, 'PIECEWISE CUBIC HERMITE INTERPOLATION
C             PACKAGE, FINAL SPECIFICATIONS', LAWRENCE LIVERMORE
C             NATIONAL LABORATORY, COMPUTER DOCUMENTATION UCID-
30194,
C              AUGUST 1982.
C***ROUTINES CALLED  PCHFD
C***END PROLOGUE  PCHEV
      INTEGER  N, NVAL, IERR
      REAL  X(N), F(N), D(N), XVAL(NVAL), FVAL(NVAL), DVAL(NVAL)
C
C DECLARE LOCAL VARIABLES.
C
      INTEGER INCFD
      LOGICAL SKIP
      DATA SKIP /.TRUE./
      DATA INCFD /1/

C
C
C***FIRST EXECUTABLE STATEMENT  PCHEV
C
      CALL PCHFD(N,X,F,D,INCFD,SKIP,NVAL,XVAL,FVAL,DVAL,IERR)
C
C
 5000 CONTINUE
      RETURN
C
C------------ LAST LINE OF PCHEV FOLLOWS -----------------------------
      END
       SUBROUTINE PCHFD(N,X,F,D,INCFD,SKIP,NE,XE,FE,DE,IERR)
C***BEGIN PROLOGUE  PCHFD
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
```

```
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C***END PROLOGUE  PCHFD
C
C DECLARE ARGUMENTS.
C
      INTEGER  N, INCFD, NE, IERR
      REAL  X(N), F(INCFD,N), D(INCFD,N), XE(NE), FE(NE), DE(NE)
      LOGICAL  SKIP
C
C DECLARE LOCAL VARIABLES.
C
      INTEGER  I, IERC, IR, J, JFIRST, NEXT(2), NJ
C
C VALIDITY-CHECK ARGUMENTS.
C
C***FIRST EXECUTABLE STATEMENT  PCHFD
      IF (SKIP)  GO TO 5
C
      IF ( N.LT.2 )  GO TO 5001
      IF ( INCFD.LT.1 )  GO TO 5002
      DO 1  I = 2, N
        IF ( X(I).LE.X(I-1) )  GO TO 5003
    1 CONTINUE
C
C FUNCTION DEFINITION IS OK, GO ON.
C
    5 CONTINUE
      IF ( NE.LT.1 )  GO TO 5004
      IERR = 0
      SKIP = .TRUE.
C
C LOOP OVER INTERVALS.       (  INTERVAL INDEX IS  IL = IR-1  . )
C                       ( INTERVAL IS X(IL).LE.X.LT.X(IR) . )
      JFIRST = 1
      IR = 2
   10 CONTINUE
C
C    SKIP OUT OF LOOP IF HAVE PROCESSED ALL EVALUATION POINTS.
C
       IF (JFIRST .GT. NE)  GO TO 5000
C
C    LOCATE ALL POINTS IN INTERVAL.
C
       DO 20  J = JFIRST, NE
         IF (XE(J) .GE. X(IR))  GO TO 30
   20  CONTINUE
       J = NE + 1
       GO TO 40
C
```

```
C   HAVE LOCATED FIRST POINT BEYOND INTERVAL.
C
  30  CONTINUE
      IF (IR .EQ. N)  J = NE + 1
C
  40  CONTINUE
      NJ = J - JFIRST
C
C   SKIP EVALUATION IF NO POINTS IN INTERVAL.
C
      IF (NJ .EQ. 0)  GO TO 50
C
C   EVALUATE CUBIC AT XE(I),  I = JFIRST (1) J-1 .
C
C     ------------------------------------------------------------
      CALL CHFDV (X(IR-1),X(IR), F(1,IR-1),F(1,IR), D(1,IR-1),D(1,IR),
     *           NJ, XE(JFIRST), FE(JFIRST), DE(JFIRST), NEXT, IERC)
C     ------------------------------------------------------------
      IF (IERC .LT. 0)  GO TO 5005
C
      IF (NEXT(2) .EQ. 0)  GO TO 42
C       IF (NEXT(2) .GT. 0)  THEN
C         IN THE CURRENT SET OF XE-POINTS, THERE ARE NEXT(2) TO THE
C         RIGHT OF X(IR).
C
          IF (IR .LT. N)  GO TO 41
C           IF (IR .EQ. N)  THEN
C             THESE ARE ACTUALLY EXTRAPOLATION POINTS.
            IERR = IERR + NEXT(2)
            GO TO 42
  41      CONTINUE
C         ELSE
C           WE SHOULD NEVER HAVE GOTTEN HERE.
            GO TO 5005
C         ENDIF
C       ENDIF
  42  CONTINUE
C
      IF (NEXT(1) .EQ. 0)  GO TO 49
C       IF (NEXT(1) .GT. 0)  THEN
C         IN THE CURRENT SET OF XE-POINTS, THERE ARE NEXT(1) TO THE
C         LEFT OF X(IR-1).
C
          IF (IR .GT. 2)  GO TO 43
C           IF (IR .EQ. 2)  THEN
C             THESE ARE ACTUALLY EXTRAPOLATION POINTS.
            IERR = IERR + NEXT(1)
            GO TO 49
  43      CONTINUE
C         ELSE
C           XE IS NOT ORDERED RELATIVE TO X, SO MUST ADJUST
C           EVALUATION INTERVAL.
```

```
C
C          FIRST, LOCATE FIRST POINT TO LEFT OF X(IR-1).
           DO 44  I = JFIRST, J-1
             IF (XE(I) .LT. X(IR-1))  GO TO 45
  44       CONTINUE
C          NOTE-- CANNOT DROP THROUGH HERE UNLESS THERE IS AN
ERROR
C               IN CHFDV.
           GO TO 5005
C
  45       CONTINUE
C          RESET J.  (THIS WILL BE THE NEW JFIRST.)
           J = I
C
C          NOW FIND OUT HOW FAR TO BACK UP IN THE X-ARRAY.
           DO 46  I = 1, IR-1
             IF (XE(J) .LT. X(I)) GO TO 47
  46       CONTINUE
C          NB-- CAN NEVER DROP THROUGH HERE, SINCE XE(J).LT.X(IR-1).
C
  47       CONTINUE
C          AT THIS POINT, EITHER  XE(J) .LT. X(1)
C            OR     X(I-1) .LE. XE(J) .LT. X(I) .
C          RESET IR, RECOGNIZING THAT IT WILL BE INCREMENTED BEFORE
C          CYCLING.
           IR = MAX0(1, I-1)
C        ENDIF
C      ENDIF
  49   CONTINUE
C
       JFIRST = J
C
C    END OF IR-LOOP.
C
  50 CONTINUE
     IR = IR + 1
     IF (IR .LE. N)  GO TO 10
C
C  NORMAL RETURN.
C
 5000 CONTINUE
     RETURN
C
C  ERROR RETURNS.
C
 5001 CONTINUE
C     N.LT.2 RETURN.
     IERR = -1
     CALL XERROR ('PCHFD -- NUMBER OF DATA POINTS LESS THAN TWO'
    *        , 44, IERR, 1)
     RETURN
C
```

```
 5002 CONTINUE
C     INCFD.LT.1 RETURN.
      IERR = -2
      CALL XERROR ('PCHFD -- INCREMENT LESS THAN ONE'
     *        , 32, IERR, 1)
      RETURN
C
 5003 CONTINUE
C     X-ARRAY NOT STRICTLY INCREASING.
      IERR = -3
      CALL XERROR ('PCHFD -- X-ARRAY NOT STRICTLY INCREASING'
     *        , 40, IERR, 1)
      RETURN
C
 5004 CONTINUE
C     NE.LT.1 RETURN.
      IERR = -4
      CALL XERROR ('PCHFD -- NUMBER OF EVALUATION POINTS LESS THAN
ONE'
     *        , 50, IERR, 1)
      RETURN
C
 5005 CONTINUE
C     ERROR RETURN FROM CHFDV.
C  *** THIS CASE SHOULD NEVER OCCUR ***
      IERR = -5
      CALL XERROR ('PCHFD -- ERROR RETURN FROM CHFDV -- FATAL'
     *        , 41, IERR, 2)
      RETURN
C------------ LAST LINE OF PCHFD FOLLOWS -----------------------------
      END
       SUBROUTINE CHFDV(X1,X2,F1,F2,D1,D2,NE,XE,FE,DE,NEXT,IERR)
C***BEGIN PROLOGUE  CHFDV
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  CHFDV
C
C  DECLARE ARGUMENTS.
C
      INTEGER  NE, NEXT(2), IERR
      REAL  X1, X2, F1, F2, D1, D2, XE(NE), FE(NE), DE(NE)
C
C  DECLARE LOCAL VARIABLES.
C
      INTEGER  I
      REAL  C2, C2T2, C3, C3T3, DEL1, DEL2, DELTA, H, X, XMI, XMA, ZERO
      DATA  ZERO /0./
C
C  VALIDITY-CHECK ARGUMENTS.
```

```
C
C***FIRST EXECUTABLE STATEMENT  CHFDV
      IF (NE .LT. 1)  GO TO 5001
      H = X2 - X1
      IF (H .EQ. ZERO)  GO TO 5002
C
C  INITIALIZE.
C
      IERR = 0
      NEXT(1) = 0
      NEXT(2) = 0
      XMI = AMIN1(ZERO, H)
      XMA = AMAX1(ZERO, H)
C
C  COMPUTE CUBIC COEFFICIENTS (EXPANDED ABOUT X1).
C
      DELTA = (F2 - F1)/H
      DEL1 = (D1 - DELTA)/H
      DEL2 = (D2 - DELTA)/H
C                                 (DELTA IS NO LONGER NEEDED.)
      C2 = -(DEL1+DEL1 + DEL2)
      C2T2 = C2 + C2
      C3 = (DEL1 + DEL2)/H
C                           (H, DEL1 AND DEL2 ARE NO LONGER NEEDED.)
      C3T3 = C3+C3+C3
C
C  EVALUATION LOOP.
C
      DO 500  I = 1, NE
        X = XE(I) - X1
        FE(I) = F1 + X*(D1 + X*(C2 + X*C3))
        DE(I) = D1 + X*(C2T2 + X*C3T3)
C         COUNT EXTRAPOLATION POINTS.
        IF ( X.LT.XMI )  NEXT(1) = NEXT(1) + 1
        IF ( X.GT.XMA )  NEXT(2) = NEXT(2) + 1
C       (NOTE REDUNDANCY--IF EITHER CONDITION IS TRUE, OTHER IS
FALSE.)
  500 CONTINUE
C
C  NORMAL RETURN.
C
      RETURN
C
C  ERROR RETURNS.
C
 5001 CONTINUE
C     NE.LT.1 RETURN.
      IERR = -1
      CALL XERROR ('CHFDV -- NUMBER OF EVALUATION POINTS LESS THAN
ONE'
     *        , 50, IERR, 1)
      RETURN
```

```
C
 5002 CONTINUE
C    X1.EQ.X2 RETURN.
     IERR = -2
     CALL XERROR ('CHFDV -- INTERVAL ENDPOINTS EQUAL'
     *       , 33, IERR, 1)
     RETURN
C------------ LAST LINE OF CHFDV FOLLOWS ----------------------------
     END



C     ************************************************************
      SUBROUTINE FZERO(F,B,C,R,RE,AE,IFLAG)
C***BEGIN PROLOGUE  FZERO
C***DATE WRITTEN  700901  (YYMMDD)
C***REVISION DATE  860411  (YYMMDD)
C***CATEGORY NO.  F1B
C***KEYWORDS  BISECTION,NONLINEAR,ROOTS,ZEROS
C***AUTHOR  SHAMPINE,L.F.,SNLA
C         WATTS,H.A.,SNLA
C***PURPOSE  FZERO searches for a zero of a function F(X) in a given
C         interval (B,C).  It is designed primarily for problems
C         where F(B) and F(C) have opposite signs.
C***DESCRIPTION
C
C   From the book "Numerical Methods and Software"
C      by  D. Kahaner, C. Moler, S. Nash
C         Prentice Hall 1988
C
C   Based on a method by T J Dekker
C   written by L F Shampine and H A Watts
C
C         FZERO searches for a zero of a function F(X) between
C         the given values B and C until the width of the interval
C         (B,C) has collapsed to within a tolerance specified by
C         the stopping criterion, ABS(B-C) .LE. 2.*(RW*ABS(B)+AE).
C         The method used is an efficient combination of bisection
C         and the secant rule.
C
C   Description Of Arguments
C
C   F,B,C,R,RE and AE are input parameters
C   B,C and IFLAG are output parameters (flagged by an * below)
C
C      F     - Name of the real valued external function.  This name
C            must be in an EXTERNAL statement in the calling
C            program.  F must be a function of one real argument.
C
C     *B     - One end of the interval (B,C).  The value returned for
C            B usually is the better approximation to a zero of F.
C
```

```
C     *C   - The other end of the interval (B,C)
C
C      R   - A (better) guess of a zero of F which could help in
C            speeding up convergence.  If F(B) and F(R) have
C            opposite signs, a root will be found in the interval
C            (B,R); if not, but F(R) and F(C) have opposite
C            signs, a root will be found in the interval (R,C);
C            otherwise, the interval (B,C) will be searched for a
C            possible root.  When no better guess is known, it is
C            recommended that r be set to B or C; because if R is
C            not interior to the interval (B,C), it will be ignored.
C
C      RE  - Relative error used for RW in the stopping criterion.
C            If the requested RE is less than machine precision,
C            then RW is set to approximately machine precision.
C
C      AE  - Absolute error used in the stopping criterion.  If the
C            given interval (B,C) contains the origin, then a
C            nonzero value should be chosen for AE.
C
C     *IFLAG - A status code.  User must check IFLAG after each call.
C            Control returns to the user from FZERO in all cases.
C
C            1  B is within the requested tolerance of a zero.
C              The interval (B,C) collapsed to the requested
C              tolerance, the function changes sign in (B,C), and
C              F(X) decreased in magnitude as (B,C) collapsed.
C
C            2  F(B) = 0.  However, the interval (B,C) may not have
C              collapsed to the requested tolerance.
C
C            3  B may be near a singular point of F(X).
C              The interval (B,C) collapsed to the requested tol-
C              erance and the function changes sign in (B,C), but
C              F(X) increased in magnitude as (B,C) collapsed,i.e.
C                abs(F(B out)) .GT. max(abs(F(B in)),abs(F(C in)))
C
C            4  No change in sign of F(X) was found although the
C              interval (B,C) collapsed to the requested tolerance.
C              The user must examine this case and decide whether
C              B is near a local minimum of F(X), or B is near a
C              zero of even multiplicity, or neither of these.
C
C            5  Too many (.GT. 500) function evaluations used.
C***REFERENCES  L. F. SHAMPINE AND H. A. WATTS, *FZERO, A ROOT-
SOLVING
C              CODE*, SC-TM-70-631, SEPTEMBER 1970.
C            T. J. DEKKER, *FINDING A ZERO BY MEANS OF SUCCESSIVE
C              LINEAR INTERPOLATION*, 'CONSTRUCTIVE ASPECTS OF THE
C              FUNDAMENTAL THEOREM OF ALGEBRA', EDITED BY B. DEJON
C              P. HENRICI, 1969.
C***ROUTINES CALLED  R1MACH
```

```
C***END PROLOGUE  FZERO
      REAL A,ACBS,ACMB,AE,AW,B,C,CMB,ER,FA,FB,FC,FX,FZ,P,Q,R
     REAL RE,RW,T,TOL,Z
     INTEGER IC,IFLAG,KOUNT


C
C    ER IS TWO TIMES THE COMPUTER UNIT ROUNDOFF VALUE WHICH IS
C    DEFINED HERE BY THE FUNCTION R1MACH.
C
C***FIRST EXECUTABLE STATEMENT  FZERO
     ER = 2.0E0 * R1MACH(4)
C
C    INITIALIZE
C
     Z=R
      IF(R.LE.AMIN1(B,C).OR.R.GE.AMAX1(B,C)) Z=C
     RW=AMAX1(RE,ER)
     AW=AMAX1(AE,0.0)
     IC=0
     T=Z
     FZ=F(T)
     FC=FZ
     T=B
     FB=F(T)
     KOUNT=2
      IF(SIGN(1.0E0,FZ).EQ.SIGN(1.0E0,FB)) GO TO 1
     C=Z
     GO TO 2
   1 IF(Z.EQ.C) GO TO 2
     T=C
     FC=F(T)
     KOUNT=3
      IF(SIGN(1.0E0,FZ).EQ.SIGN(1.0E0,FC)) GO TO 2
     B=Z
     FB=FZ
   2 A=C
     FA=FC
     ACBS=ABS(B-C)
     FX=AMAX1(ABS(FB),ABS(FC))
C
   3 IF (ABS(FC) .GE. ABS(FB)) GO TO 4
C    PERFORM INTERCHANGE
     A=B
     FA=FB
     B=C
     FB=FC
     C=A
     FC=FA
C
   4 CMB=0.5*(C-B)
     ACMB=ABS(CMB)
```

```
      TOL=RW*ABS(B)+AW
C
C     TEST STOPPING CRITERION AND FUNCTION COUNT
C
      IF (ACMB .LE. TOL) GO TO 10
      IF(FB.EQ.0.E0) GO TO 11
      IF(KOUNT.GE.500) GO TO 14
C
C     CALCULATE NEW ITERATE IMPLICITLY AS B+P/Q
C     WHERE WE ARRANGE P .GE. 0.
C     THE IMPLICIT FORM IS USED TO PREVENT OVERFLOW.
C
      P=(B-A)*FB
      Q=FA-FB
      IF (P .GE. 0.) GO TO 5
      P=-P
      Q=-Q
C
C     UPDATE A AND CHECK FOR SATISFACTORY REDUCTION
C     IN THE SIZE OF THE BRACKETING INTERVAL.
C     IF NOT, PERFORM BISECTION.
C
    5 A=B
      FA=FB
      IC=IC+1
      IF (IC .LT. 4) GO TO 6
      IF (8.*ACMB .GE. ACBS) GO TO 8
      IC=0
      ACBS=ACMB
C
C     TEST FOR TOO SMALL A CHANGE
C
    6 IF (P .GT. ABS(Q)*TOL) GO TO 7
C
C     INCREMENT BY TOLERANCE
C
      B=B+SIGN(TOL,CMB)
      GO TO 9
C
C     ROOT OUGHT TO BE BETWEEN B AND (C+B)/2.
C
    7 IF (P .GE. CMB*Q) GO TO 8
C
C     USE SECANT RULE
C
      B=B+P/Q
      GO TO 9
C
C     USE BISECTION
C
    8 B=0.5*(C+B)
C
```

```
C   HAVE COMPLETED COMPUTATION FOR NEW ITERATE B
C
  9 T=B
    FB=F(T)
    KOUNT=KOUNT+1
C
C   DECIDE WHETHER NEXT STEP IS INTERPOLATION OR EXTRAPOLATION
C
    IF (SIGN(1.0,FB) .NE. SIGN(1.0,FC)) GO TO 3
    C=A
    FC=FA
    GO TO 3
C
C
C   FINISHED. PROCESS RESULTS FOR PROPER SETTING OF IFLAG
C
 10 IF (SIGN(1.0,FB) .EQ. SIGN(1.0,FC)) GO TO 13
    IF (ABS(FB) .GT. FX) GO TO 12
    IFLAG = 1
    RETURN
 11 IFLAG = 2
    RETURN
 12 IFLAG = 3
    RETURN
 13 IFLAG = 4
    RETURN
 14 IFLAG = 5
    RETURN
    END



      SUBROUTINE SDRIV2
(N,T,Y,F,TOUT,MSTATE,NROOT,EPS,EWT,MINT,WORK,

&LENW,IWORK,LENIW,G,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,STOREC,
STOPRO)
C***BEGIN PROLOGUE  SDRIV2
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  871105  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***KEYWORDS  ODE,STIFF,ORDINARY DIFFERENTIAL EQUATIONS,
C          INITIAL VALUE PROBLEMS,GEAR'S METHOD,
C          SINGLE PRECISION
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C       SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***PURPOSE  The function of SDRIV2 is to solve N ordinary differential
C          equations of the form dY(I)/dT = F(Y(I),T), given the
C          initial conditions Y(I) = YI.  The program has options to
C          allow the solution of both stiff and non-stiff differential
C          equations.  SDRIV2 uses single precision arithmetic.
C***DESCRIPTION
```

```
C  From the book "Numerical Methods and Software"
C    by  D. Kahaner, C. Moler, S. Nash
C      Prentice Hall 1988
C
C  I.  ABSTRACT  .....
C
C  The function of SDRIV2 is to solve N ordinary differential
C  equations of the form dY(I)/dT = F(Y(I),T), given the initial
C  conditions Y(I) = YI. The program has options to allow the
C  solution of both stiff and non-stiff differential equations.
C  SDRIV2 is to be called once for each output point of T.
C
C  II.  PARAMETERS  .....
C
C  The user should use parameter names in the call sequence of SDRIV2
C  for those quantities whose value may be altered by SDRIV2. The
C  parameters in the call sequence are:
C
C  N    = (Input) The number of differential equations.
C
C  T    = The independent variable. On input for the first call, T
C         is the initial point. On output, T is the point at which
C         the solution is given.
C
C  Y    = The vector of dependent variables. Y is used as input on
C         the first call, to set the initial values. On output, Y
C         is the computed solution vector. This array Y is passed
C         in the call sequence of the user-provided routines F and
C         G. Thus parameters required by F and G can be stored in
C         this array in components N+1 and above. (Note: Changes
C         by the user to the first N components of this array will
C         take effect only after a restart, i.e., after setting
C         MSTATE to +1(-1).)
C
C  F    = A subroutine supplied by the user. The name must be
C         declared EXTERNAL in the user's calling program. This
C         subroutine is of the form:
C              SUBROUTINE F (N, T, Y, YDOT)
C              REAL Y(*), YDOT(*)
C                .
C                .
C              YDOT(1) = ...
C                .
C                .
C              YDOT(N) = ...
C              END (Sample)
C         This computes YDOT = F(Y,T), the right hand side of the
C         differential equations. Here Y is a vector of length at
C         least N. The actual length of Y is determined by the
C         user's declaration in the program which calls SDRIV2.
C         Thus the dimensioning of Y in F, while required by FORTRAN
C         convention, does not actually allocate any storage. When
```

```
C         this subroutine is called, the first N components of Y are
C         intermediate approximations to the solution components.
C         The user should not alter these values. Here YDOT is a
C         vector of length N. The user should only compute YDOT(I)
C         for I from 1 to N. Normally a return from F passes
C         control back to  SDRIV2. However, if the user would like
C         to abort the calculation, i.e., return control to the
C         program which calls SDRIV2, he should set N to zero.
C         SDRIV2 will signal this by returning a value of MSTATE
C         equal to +6(-6). Altering the value of N in F has no
C         effect on the value of N in the call sequence of SDRIV2.
C
C   TOUT  = (Input) The point at which the solution is desired.
C
C   MSTATE = An integer describing the status of integration. The user
C         must initialize MSTATE to +1 or -1. If MSTATE is
C         positive, the routine will integrate past TOUT and
C         interpolate the solution. This is the most efficient
C         mode. If MSTATE is negative, the routine will adjust its
C         internal step to reach TOUT exactly (useful if a
C         singularity exists beyond TOUT.) The meaning of the
C         magnitude of MSTATE:
C           1  (Input) Means the first call to the routine. This
C              value must be set by the user. On all subsequent
C              calls the value of MSTATE should be tested by the
C              user. Unless SDRIV2 is to be reinitialized, only the
C              sign of MSTATE may be changed by the user. (As a
C              convenience to the user who may wish to put out the
C              initial conditions, SDRIV2 can be called with
C              MSTATE=+1(-1), and TOUT=T. In this case the program
C              will return with MSTATE unchanged, i.e.,
C              MSTATE=+1(-1).)
C           2  (Output) Means a successful integration. If a normal
C              continuation is desired (i.e., a further integration
C              in the same direction), simply advance TOUT and call
C              again. All other parameters are automatically set.
C           3  (Output)(Unsuccessful) Means the integrator has taken
C              1000 steps without reaching TOUT. The user can
C              continue the integration by simply calling SDRIV2
C              again. Other than an error in problem setup, the
C              most likely cause for this condition is trying to
C              integrate a stiff set of equations with the non-stiff
C              integrator option. (See description of MINT below.)
C           4  (Output)(Unsuccessful) Means too much accuracy has
C              been requested. EPS has been increased to a value
C              the program estimates is appropriate. The user can
C              continue the integration by simply calling SDRIV2
C              again.
C           5  (Output) A root was found at a point less than TOUT.
C              The user can continue the integration toward TOUT by
C              simply calling SDRIV2 again.
C           6  (Output)(Unsuccessful) N has been set to zero in
```

```
C            SUBROUTINE F.
C          7 (Output)(Unsuccessful) N has been set to zero in
C            FUNCTION G. See description of G below.
C
C  NROOT  = (Input) The number of equations whose roots are desired.
C        If NROOT is zero, the root search is not active. This
C        option is useful for obtaining output at points which are
C        not known in advance, but depend upon the solution, e.g.,
C        when some solution component takes on a specified value.
C        The root search is carried out using the user-written
C        function G (see description of G below.) SDRIV2 attempts
C        to find the value of T at which one of the equations
C        changes sign. SDRIV2 can find at most one root per
C        equation per internal integration step, and will then
C        return the solution either at TOUT or at a root, whichever
C        occurs first in the direction of integration. The index
C        of the equation whose root is being reported is stored in
C        the sixth element of IWORK.
C        NOTE: NROOT is never altered by this program.
C
C  EPS   = On input, the requested relative accuracy in all solution
C        components. EPS = 0 is allowed. On output, the adjusted
C        relative accuracy if the input value was too small. The
C        value of EPS should be set as large as is reasonable,
C        because the amount of work done by SDRIV2 increases as
C        EPS decreases.
C
C  EWT   = (Input) Problem zero, i.e., the smallest physically
C        meaningful value for the solution. This is used inter-
C        nally to compute an array YWT(I) = MAX(ABS(Y(I)), EWT).
C        One step error estimates divided by YWT(I) are kept less
C        than EPS. Setting EWT to zero provides pure relative
C        error control. However, setting EWT smaller than
C        necessary can adversely affect the running time.
C
C  MINT  = (Input) The integration method flag.
C         MINT = 1  Means the Adams methods, and is used for
C                   non-stiff problems.
C         MINT = 2  Means the stiff methods of Gear (i.e., the
C                   backward differentiation formulas), and is
C                   used for stiff problems.
C         MINT = 3  Means the program dynamically selects the
C                   Adams methods when the problem is non-stiff
C                   and the Gear methods when the problem is
C                   stiff.
C        MINT may not be changed without restarting, i.e., setting
C        the magnitude of MSTATE to 1.
C
C  WORK
C  LENW  = (Input)
C        WORK is an array of LENW real words used
C        internally for temporary storage. The user must allocate
```

```
C         space for this array in the calling program by a statement
C         such as
C                 REAL WORK(...)
C         The length of WORK should be at least
C           16*N + 2*NROOT + 204        if MINT is 1, or
C           N*N + 10*N + 2*NROOT + 204  if MINT is 2, or
C           N*N + 17*N + 2*NROOT + 204  if MINT is 3,
C         and LENW should be set to the value used.  The contents of
C         WORK should not be disturbed between calls to SDRIV2.
C
C  IWORK
C  LENIW  = (Input)
C         IWORK is an integer array of length LENIW used internally
C         for temporary storage.  The user must allocate space for
C         this array in the calling program by a statement such as
C                 INTEGER IWORK(...)
C         The length of IWORK should be at least
C           21     if MINT is 1, or
C           N+21   if MINT is 2 or 3,
C         and LENIW should be set to the value used.  The contents
C         of IWORK should not be disturbed between calls to SDRIV2.
C
C  G      = A real FORTRAN function supplied by the user
C         if NROOT is not 0.  In this case, the name must be
C         declared EXTERNAL in the user's calling program.  G is
C         repeatedly called with different values of IROOT to
C         obtain the value of each of the NROOT equations for which
C         a root is desired.  G is of the form:
C             REAL FUNCTION G (N, T, Y, IROOT)
C             REAL Y(*)
C             GO TO (10, ...), IROOT
C           10  G = ...
C               .
C               .
C               .
C             END (Sample)
C         Here, Y is a vector of length at least N, whose first N
C         components are the solution components at the point T.
C         The user should not alter these values.  The actual length
C         of Y is determined by the user's declaration in the
C         program which calls SDRIV2.  Thus the dimensioning of Y in
C         G, while required by FORTRAN convention, does not actually
C         allocate any storage.  Normally a return from G passes
C         control back to  SDRIV2.  However, if the user would like
C         to abort the calculation, i.e., return control to the
C         program which calls SDRIV2, he should set N to zero.
C         SDRIV2 will signal this by returning a value of MSTATE
C         equal to +7(-7).  In this case, the index of the equation
C         being evaluated is stored in the sixth element of IWORK.
C         Altering the value of N in G has no effect on the value of
C         N in the call sequence of SDRIV2.
C
C***LONG DESCRIPTION
```

```
C
C  III.  OTHER COMMUNICATION TO THE USER .....
C
C  A. The solver communicates to the user through the parameters
C     above.  In addition it writes diagnostic messages through the
C     standard error handling program XERROR.  That program will
C     terminate the user's run if it detects a probable problem setup
C     error, e.g., insufficient storage allocated by the user for the
C     WORK array.  Messages are written on the standard error message
C     file.  At installations which have this error handling package
C     the user should determine the standard error handling file from
C     the local documentation.  Otherwise the short but serviceable
C     routine, XERROR, available with this package, can be used.  That
C     program writes on logical unit 6 to transmit messages.  A
C     complete description of XERROR is given in the Sandia
C     Laboratories report SAND78-1189 by R. E. Jones.
C
C  B. The first three elements of WORK and the first five elements of
C     IWORK will contain the following statistical data:
C        AVGH    The average step size used.
C        HUSED   The step size last used (successfully).
C        AVGORD  The average order used.
C        IMXERR  The index of the element of the solution vector that
C                contributed most to the last error test.
C        NQUSED  The order last used (successfully).
C        NSTEP   The number of steps taken since last initialization.
C        NFE     The number of evaluations of the right hand side.
C        NJE     The number of evaluations of the Jacobian matrix.
C
C  IV.   REMARKS .....
C
C  A. On any return from SDRIV2 all information necessary to continue
C     the calculation is contained in the call sequence parameters,
C     including the work arrays.  Thus it is possible to suspend one
C     problem, integrate another, and then return to the first.
C
C  B. If this package is to be used in an overlay situation, the user
C     must declare in the primary overlay the variables in the call
C     sequence to SDRIV2.
C
C  C. When the routine G is not required, difficulties associated with
C     an unsatisfied external can be avoided by using the name of the
C     routine which calculates the right hand side of the differential
C     equations in place of G in the call sequence of SDRIV2.
C
C  V.   USAGE .....
C
C           PROGRAM SAMPLE
C           EXTERNAL F
C           PARAMETER(MINT = 1, NROOT = 0, N = ...,
C          8        LENW = 16*N + 2*NROOT + 204, LENIW = 21)
C                              N is the number of equations
```

```
C           REAL EPS, EWT, T, TOUT, WORK(LENW), Y(N)
C           INTEGER IWORK(LENIW)
C            OPEN(FILE='TAPE6', UNIT=6, STATUS='NEW')
C           T = 0.                  Initial point
C           DO 10 I = 1,N
C       10    Y(I) = ...            Set initial conditions
C           TOUT = T
C           EWT = ...
C           MSTATE = 1
C            EPS = ...
C       20  CALL SDRIV2 (N, T, Y, F, TOUT, MSTATE, NROOT, EPS, EWT,
C          8          MINT, WORK, LENW, IWORK, LENIW, F)
C                        Last argument is not the same
C                        as F if rootfinding is used.
C           IF (MSTATE .GT. 2) STOP
C           WRITE(6, 100) TOUT, (Y(I), I=1,N)
C           TOUT = TOUT + 1.
C           IF (TOUT .LE. 10.) GO TO 20
C       100 FORMAT(...)
C           END (Sample)
C
C***REFERENCES  GEAR, C. W., "NUMERICAL INITIAL VALUE PROBLEMS IN
C              ORDINARY DIFFERENTIAL EQUATIONS", PRENTICE-HALL, 1971.
C***ROUTINES CALLED  SDRIV3,XERROR
C***END PROLOGUE  SDRIV2
      EXTERNAL F,G
      REAL EPS, EWT, EWTCOM(1), G, HMAX, T, TOUT,
     8    WORK(*), Y(*)
          INTEGER N,NREACT,NSPECS
          REAL TEMP,KF(*),STOREC(80,80),
     8     STOPRO(80,80),CF(*),AF(*),RU,BF(*)
      INTEGER IWORK(*)
      CHARACTER MSG*81
      PARAMETER(IMPL = 0, MXSTEP = 1000)

C***FIRST EXECUTABLE STATEMENT  SDRIV2
      IF (MINT .LT. 1 .OR. MINT .GT. 3) THEN
        WRITE(MSG, '("SDRIV21FE Illegal input.  Improper value for ",
     8  "the integration method flag,", I8)') MINT
        CALL XERROR(MSG(1:81), 81, 21, 2)
        RETURN
      END IF
      IF (MSTATE .GE. 0) THEN
        NSTATE = MSTATE
        NTASK = 1
      ELSE
        NSTATE = - MSTATE
        NTASK = 3
      END IF
      EWTCOM(1) = EWT
      IF (EWT .NE. 0.E0) THEN
        IERROR = 3
```

```
      ELSE
        IERROR = 2
      END IF
      IF (MINT .EQ. 1) THEN
        MITER = 0
        MXORD = 12
      ELSE IF (MINT .EQ. 2) THEN
        MITER = 2
        MXORD = 5
      ELSE IF (MINT .EQ. 3) THEN
        MITER = 2
        MXORD = 12
      END IF
      HMAX = 2.E0*ABS(TOUT - T)
      CALL SDRIV3 (N, T, Y, F, NSTATE, TOUT, NTASK, NROOT, EPS, EWTCOM,
     8          IERROR, MINT, MITER, IMPL, ML, MU, MXORD, HMAX, WORK,
     8           LENW, IWORK, LENIW, F, F, NDE, MXSTEP, G, F,
     8           TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,STOREC,STOPRO)
      IF (MSTATE .GE. 0) THEN
        MSTATE = NSTATE
      ELSE
        MSTATE = - NSTATE
      END IF
      END
      SUBROUTINE SDCOR
     (DFDY,EL,FA,H,IMPL,IPVT,MATDIM,MITER,ML,MU,N,
     8  NDE,NQ,T,USERS,Y,YH,YWT,EVALFA,SAVE1,SAVE2,A,D,JSTATE)
C***BEGIN PROLOGUE  SDCOR
C***REFER TO  SDRIV3
C  Subroutine SDCOR is called to compute corrections to the Y array.
C  In the case of functional iteration, update Y directly from the
C  result of the last call to F.
C  In the case of the chord method, compute the corrector error and
C  solve the linear system with that as right hand side and DFDY as
C  coefficient matrix, using the LU decomposition if MITER is 1, 2, 4,
C  or 5.
C***ROUTINES CALLED  SGESL,SGBSL,SNRM2
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870401  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCOR
      REAL A(MATDIM,*), D, DFDY(MATDIM,*), EL(13,12), H,
     8  SAVE1(*), SAVE2(*), SNRM2, T, Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL EVALFA

C***FIRST EXECUTABLE STATEMENT  SDCOR
      IF (MITER .EQ. 0) THEN
        DO 100 I = 1,N
100     SAVE1(I) = (H*SAVE2(I) - YH(I,2) - SAVE1(I))/YWT(I)
```

```
      D = SNRM2(N, SAVE1, 1)/SQRT(REAL(N))
      DO 105 I = 1,N
105     SAVE1(I) = H*SAVE2(I) - YH(I,2)
    ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
      IF (IMPL .EQ. 0) THEN
       DO 130 I = 1,N
130      SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
      ELSE IF (IMPL .EQ. 1) THEN
       IF (EVALFA) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
       ELSE
        EVALFA = .TRUE.
       END IF
       DO 150 I = 1,N
150      SAVE2(I) = H*SAVE2(I)
       DO 160 J = 1,N
        DO 160 I = 1,N
160       SAVE2(I) = SAVE2(I) - A(I,J)*(YH(J,2) + SAVE1(J))
      ELSE IF (IMPL .EQ. 2) THEN
       IF (EVALFA) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
       ELSE
        EVALFA = .TRUE.
       END IF
       DO 180 I = 1,N
180      SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
      END IF
      CALL SGESL (DFDY, MATDIM, N, IPVT, SAVE2, 0)
      DO 200 I = 1,N
       SAVE1(I) = SAVE1(I) + SAVE2(I)
200    SAVE2(I) = SAVE2(I)/YWT(I)
      D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
    ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
      IF (IMPL .EQ. 0) THEN
       DO 230 I = 1,N
230      SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
      ELSE IF (IMPL .EQ. 1) THEN
       IF (EVALFA) THEN
        CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
       ELSE
```

```
        EVALFA = .TRUE.
       END IF
       DO 250 I = 1,N
250      SAVE2(I) = H*SAVE2(I)
       MW = ML + 1 + MU
       DO 260 J = 1,N
        I1 = MAX(ML+1, MW+1-J)
        I2 = MIN(MW+N-J, MW+ML)
        DO 260 I = I1,I2
         I3 = I + J - MW
260       SAVE2(I3) = SAVE2(I3) - A(I,J)*(YH(J,2) + SAVE1(J))
     ELSE IF (IMPL .EQ. 2) THEN
       IF (EVALFA) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
       ELSE
        EVALFA = .TRUE.
       END IF
       DO 280 I = 1,N
280      SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
     END IF
     CALL SGBSL (DFDY, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
     DO 300 I = 1,N
      SAVE1(I) = SAVE1(I) + SAVE2(I)
300    SAVE2(I) = SAVE2(I)/YWT(I)
     D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
   ELSE IF (MITER .EQ. 3) THEN
     IFLAG = 2
     CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
  8          N, NDE, IFLAG)
     IF (N .EQ. 0) THEN
      JSTATE = 10
      RETURN
     END IF
     DO 320 I = 1,N
      SAVE1(I) = SAVE1(I) + SAVE2(I)
320    SAVE2(I) = SAVE2(I)/YWT(I)
     D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
   END IF
   END
   SUBROUTINE SDCST (MAXORD,MINT,ISWFLG,EL,TQ)
C***BEGIN PROLOGUE  SDCST
C***REFER TO  SDRIV3
C  SDCST is called by SDNTL and sets coefficients used by the core
C  integrator SDSTP.  The array EL determines the basic method.
C  The array TQ is involved in adjusting the step size in relation
C  to truncation error.  EL and TQ depend upon MINT, and are calculated
C  for orders 1 to MAXORD(.LE. 12).  For each order NQ, the coefficients
C  EL are calculated from the generating polynomial:
```

```
C   L(T) = EL(1,NQ) + EL(2,NQ)*T + ... + EL(NQ+1,NQ)*T**NQ.
C For the implicit Adams methods, L(T) is given by
C   dL/dT = (1+T)*(2+T)* ... *(NQ-1+T)/K,   L(-1) = 0,
C   where      K = factorial(NQ-1).
C For the Gear methods,
C   L(T) = (1+T)*(2+T)* ... *(NQ+T)/K,
C   where      K = factorial(NQ)*(1 + 1/2 + ... + 1/NQ).
C For each order NQ, there are three components of TQ.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870216  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCST
      REAL EL(13,12), FACTRL(12), GAMMA(14), SUM, TQ(3,12)

C***FIRST EXECUTABLE STATEMENT  SDCST
      FACTRL(1) = 1.E0
      DO 10 I = 2,MAXORD
 10     FACTRL(I) = REAL(I)*FACTRL(I-1)
C                                COMPUTE ADAMS COEFFICIENTS
      IF (MINT .EQ. 1) THEN
        GAMMA(1) = 1.E0
        DO 40 I = 1,MAXORD+1
          SUM = 0.E0
          DO 30 J = 1,I
 30         SUM = SUM - GAMMA(J)/REAL(I-J+2)
 40       GAMMA(I+1) = SUM
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        EL(2,2) = 1.E0
        EL(3,2) = 1.E0
        DO 60 J = 3,MAXORD
          EL(2,J) = FACTRL(J-1)
          DO 50 I = 3,J
 50         EL(I,J) = REAL(J-1)*EL(I,J-1) + EL(I-1,J-1)
 60       EL(J+1,J) = 1.E0
        DO 80 J = 2,MAXORD
          EL(1,J) = EL(1,J-1) + GAMMA(J)
          EL(2,J) = 1.E0
          DO 80 I = 3,J+1
 80         EL(I,J) = EL(I,J)/(REAL(I-1)*FACTRL(J-1))
        DO 100 J = 1,MAXORD
          TQ(1,J) = -1.E0/(FACTRL(J)*GAMMA(J))
          TQ(2,J) = -1.E0/GAMMA(J+1)
 100      TQ(3,J) = -1.E0/GAMMA(J+2)
C                                COMPUTE GEAR COEFFICIENTS
      ELSE IF (MINT .EQ. 2) THEN
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        DO 130 J = 2,MAXORD
```

```
      EL(1,J) = FACTRL(J)
      DO 120 I = 2,J
 120    EL(I,J) = REAL(J)*EL(I,J-1) + EL(I-1,J-1)
 130  EL(J+1,J) = 1.E0
    SUM = 1.E0
    DO 150 J = 2,MAXORD
      SUM = SUM + 1.E0/REAL(J)
      DO 150 I = 1,J+1
 150    EL(I,J) = EL(I,J)/(FACTRL(J)*SUM)
    DO 170 J = 1,MAXORD
      IF (J .GT. 1) TQ(1,J) = 1.E0/FACTRL(J-1)
      TQ(2,J) = REAL(J+1)/EL(1,J)
 170  TQ(3,J) = REAL(J+2)/EL(1,J)
  END IF
C                 Compute constants used in the stiffness test.
C                 These are the ratio of TQ(2,NQ) for the Gear
C                 methods to those for the Adams methods.
  IF (ISWFLG .EQ. 3) THEN
    MXRD = MIN(MAXORD, 5)
    IF (MINT .EQ. 2) THEN
      GAMMA(1) = 1.E0
      DO 190 I = 1,MXRD
        SUM = 0.E0
        DO 180 J = 1,I
 180      SUM = SUM - GAMMA(J)/REAL(I-J+2)
 190    GAMMA(I+1) = SUM
    END IF
    SUM = 1.E0
    DO 200 I = 2,MXRD
      SUM = SUM + 1.E0/REAL(I)
 200  EL(1+I,1) = -REAL(I+1)*SUM*GAMMA(I+1)
  END IF
  END
  SUBROUTINE SDNTL
(EPS,F,FA,HMAX,HOLD,IMPL,JTASK,MATDIM,MAXORD,
   8
MINT,MITER,ML,MU,N,NDE,SAVE1,T,UROUND,USERS,Y,YWT,H,MNTOLD,
   8
MTROLD,NFE,RC,YH,A,CONVRG,EL,FAC,IER,IPVT,NQ,NWAIT,RH,RMAX,
   8
SAVE2,TQ,TREND,ISWFLG,JSTATE,TEMP,KF,CF,AF,RU,NREACT,NSPECS
   8  ,BF,STOREC,STOPRO)
C***BEGIN PROLOGUE  SDNTL
C***REFER TO  SDRIV3
C  Subroutine SDNTL is called to set parameters on the first call
C  to SDSTP, on an internal restart, or when the user has altered
C  MINT, MITER, and/or H.
C  On the first call, the order is set to 1 and the initial derivatives
C  are calculated.  RMAX is the maximum ratio by which H can be
C  increased in one step.  It is initially RMINIT to compensate
C  for the small initial H, but then is normally equal to RMNORM.
C  If a failure occurs (in corrector convergence or error test), RMAX
```

```
C  is set at RMFAIL for the next increase.
C  If the caller has changed MINT, or if JTASK = 0, SDCST is called
C  to set the coefficients of the method.  If the caller has changed H,
C  YH must be rescaled.  If H or MINT has been changed, NWAIT is
C  reset to NQ + 2 to prevent further increases in H for that many
C  steps.  Also, RC is reset.  RC is the ratio of new to old values of
C  the coefficient L(0)*H.  If the caller has changed MITER, RC is
C  set to 0 to force the partials to be updated, if partials are used.
C***ROUTINES CALLED  SDCST,SDSCL,SGEFA,SGESL,SGBFA,SGBSL,SNRM2
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870810  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C        SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTL
      REAL A(MATDIM,*), EL(13,12), EPS, FAC(*), H, HMAX,
     8     HOLD, OLDL0, RC, RH, RMAX, RMINIT, SAVE1(*), SAVE2(*), SMAX,
     8     SMIN, SNRM2, SUM, SUM0, T, TQ(3,12), TREND, UROUND, Y(*),
     8     YH(N,*), YWT(*)
      INTEGER IPVT(*)
          INTEGER NREACT,NSPECS
          REAL TEMP,KF(*),CF(*),AF(*),RU,BF(*),
     1STOREC(80,80),STOPRO(80,80)
      LOGICAL CONVRG, IER
      PARAMETER(RMINIT = 10000.E0)

C***FIRST EXECUTABLE STATEMENT  SDNTL
      IER = .FALSE.
      IF (JTASK .GE. 0) THEN
        IF (JTASK .EQ. 0) THEN
          CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
          RMAX = RMINIT
        END IF
        RC = 0.E0
        CONVRG = .FALSE.
        TREND = 1.E0
        NQ = 1
        NWAIT = 3
          CALL F (N,T,Y,SAVE2,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
     8         STOREC,STOPRO)
        IF (N .EQ. 0) THEN
          JSTATE = 6
          RETURN
        END IF
        NFE = NFE + 1
        IF (IMPL .NE. 0) THEN
          IF (MITER .EQ. 3) THEN
            IFLAG = 0
            CALL USERS (Y, YH, YWT, SAVE1, SAVE2, T, H, EL, IMPL, N,
     8                  NDE, IFLAG)
            IF (N .EQ. 0) THEN
              JSTATE = 10
```

```
          RETURN
         END IF
        ELSE IF (IMPL .EQ. 1) THEN
         IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
           JSTATE = 9
           RETURN
          END IF
          CALL SGEFA (A, MATDIM, N, IPVT, INFO)
          IF (INFO .NE. 0) THEN
           IER = .TRUE.
           RETURN
          END IF
          CALL SGESL (A, MATDIM, N, IPVT, SAVE2, 0)
         ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
          CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
           JSTATE = 9
           RETURN
          END IF
          CALL SGBFA (A, MATDIM, N, ML, MU, IPVT, INFO)
          IF (INFO .NE. 0) THEN
           IER = .TRUE.
           RETURN
          END IF
          CALL SGBSL (A, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
         END IF
        ELSE IF (IMPL .EQ. 2) THEN
         CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
         IF (N .EQ. 0) THEN
          JSTATE = 9
          RETURN
         END IF
         DO 150 I = 1,NDE
          IF (A(I,1) .EQ. 0.E0) THEN
           IER = .TRUE.
           RETURN
          ELSE
           SAVE2(I) = SAVE2(I)/A(I,1)
          END IF
150       CONTINUE
        DO 155 I = NDE+1,N
155       A(I,1) = 0.E0
       END IF
      END IF
      DO 170 I = 1,NDE
170     SAVE1(I) = SAVE2(I)/YWT(I)
      SUM = SNRM2(NDE, SAVE1, 1)
      SUM0 = 1.E0/MAX(1.E0, ABS(T))
      SMAX = MAX(SUM0, SUM)
      SMIN = MIN(SUM0, SUM)
```

```
      SUM = SMAX*SQRT(1.E0 + (SMIN/SMAX)**2)/SQRT(REAL(NDE))
       H = SIGN(MIN(2.E0*EPS/SUM, ABS(H)), H)
      DO 180 I = 1,N
180     YH(I,2) = H*SAVE2(I)
      IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. ISWFLG .EQ. 3) THEN
       DO 20 I = 1,N
20       FAC(I) = SQRT(UROUND)
      END IF
     ELSE
      IF (MITER .NE. MTROLD) THEN
       MTROLD = MITER
       RC = 0.E0
       CONVRG = .FALSE.
      END IF
      IF (MINT .NE. MNTOLD) THEN
       MNTOLD = MINT
       OLDL0 = EL(1,NQ)
       CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
       RC = RC*EL(1,NQ)/OLDL0
       NWAIT = NQ + 2
      END IF
      IF (H .NE. HOLD) THEN
       NWAIT = NQ + 2
       RH = H/HOLD
       CALL SDSCL (HMAX, N, NQ, RMAX,  HOLD, RC, RH, YH)
      END IF
     END IF
     END
      SUBROUTINE SDNTP (H,K,N,NQ,T,TOUT,YH,Y)
C***BEGIN PROLOGUE  SDNTP
C***REFER TO  SDRIV3
C  Subroutine SDNTP interpolates the K-th derivative of Y at TOUT,
C  using the data in the YH array.  If K has a value greater than NQ,
C  the NQ-th derivative is calculated.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870216  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTP
      REAL FACTOR, H, R, T, TOUT, Y(*), YH(N,*)

C***FIRST EXECUTABLE STATEMENT  SDNTP
      IF (K .EQ. 0) THEN
       DO 10 I = 1,N
10       Y(I) = YH(I,NQ+1)
       R = ((TOUT - T)/H)
       DO 20 JJ = 1,NQ
        J = NQ + 1 - JJ
        DO 20 I = 1,N
20        Y(I) = YH(I,J) + R*Y(I)
```

```
      ELSE
        KUSED = MIN(K, NQ)
        FACTOR = 1.E0
        DO 40 KK = 1,KUSED
 40       FACTOR = FACTOR*REAL(NQ+1-KK)
        DO 50 I = 1,N
 50       Y(I) = FACTOR*YH(I,NQ+1)
        DO 80 JJ = KUSED+1,NQ
          J = K + 1 + NQ - JJ
          FACTOR = 1.E0
          DO 60 KK = 1,KUSED
 60         FACTOR = FACTOR*REAL(J-KK)
          DO 70 I = 1,N
 70         Y(I) = FACTOR*YH(I,J) + R*Y(I)
 80       CONTINUE
        DO 100 I = 1,N
 100      Y(I) = Y(I)*H**(-KUSED)
      END IF
      END
      SUBROUTINE SDPSC (KSGN,N,NQ,YH)
C***BEGIN PROLOGUE  SDPSC
C***REFER TO  SDRIV3
C     This subroutine computes the predicted YH values by effectively
C     multiplying the YH array by the Pascal triangle matrix when KSGN
C     is +1, and performs the inverse function when KSGN is -1.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  841119  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPSC
      REAL YH(N,*)

C***FIRST EXECUTABLE STATEMENT  SDPSC
      IF (KSGN .GT. 0) THEN
        DO 10 J1 = 1,NQ
          DO 10 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 10 I = 1,N
 10           YH(I,J) = YH(I,J) + YH(I,J+1)
      ELSE
        DO 30 J1 = 1,NQ
          DO 30 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 30 I = 1,N
 30           YH(I,J) = YH(I,J) - YH(I,J+1)
      END IF
      END
      SUBROUTINE SDPST
(EL,F,FA,H,IMPL,JACOBN,MATDIM,MITER,ML,MU,N,NDE,
```

```
     8
NQ,SAVE2,T,USERS,Y,YH,YWT,UROUND,NFE,NJE,A,DFDY,FAC,IER,IPVT,
     8   SAVE1,ISWFLG,BND,JSTATE,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
     8  STOREC,STOPRO)
C***BEGIN PROLOGUE  SDPST
C***REFER TO  SDRIV3
C  Subroutine SDPST is called to reevaluate the partials.
C  If MITER is 1, 2, 4, or 5, the matrix
C  P = I - L(0)*H*Jacobian is stored in DFDY and subjected to LU
C  decomposition, with the results also stored in DFDY.
C***ROUTINES CALLED  SGEFA,SGBFA,SNRM2
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870401  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPST
      REAL A(MATDIM,*), BL, BND, BP, BR, BU, DFDY(MATDIM,*),
     8    DFDYMX, DIFF, DY, EL(13,12), FAC(*), FACMAX, FACMIN, FACTOR,
     8    H, SAVE1(*), SAVE2(*), SCALE, SNRM2, T, UROUND, Y(*),
     8    YH(N,*), YJ, YS, YWT(*)
      INTEGER IPVT(*)
          INTEGER NREACT,NSPECS
          REAL TEMP,KF(*),CF(*),AF(*),RU,BF(*),
     1    STOREC(80,80),STOPRO(80,80)
      LOGICAL IER
      PARAMETER(FACMAX = .5E0)

C***FIRST EXECUTABLE STATEMENT  SDPST
      NJE = NJE + 1
      IER = .FALSE.
      IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
       IF (MITER .EQ. 1) THEN
        CALL JACOBN (N, T, Y, DFDY, MATDIM, ML, MU)
        IF (N .EQ. 0) THEN
         JSTATE = 8
         RETURN
        END IF
        IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)
        FACTOR = -EL(1,NQ)*H
        DO 110 J = 1,N
         DO 110 I = 1,N
110        DFDY(I,J) = FACTOR*DFDY(I,J)
      ELSE IF (MITER .EQ. 2) THEN
        BR = UROUND**(.875E0)
        BL = UROUND**(.75E0)
        BU = UROUND**(.25E0)
        BP = UROUND**(-.15E0)
        FACMIN = UROUND**(.78E0)
        DO 170 J = 1,N
         YS = MAX(ABS(YWT(J)), ABS(Y(J)))
120       DY = FAC(J)*YS
```

```
      IF (DY .EQ. 0.E0) THEN
       IF (FAC(J) .LT. FACMAX) THEN
        FAC(J) = MIN(100.E0*FAC(J), FACMAX)
        GO TO 120
       ELSE
        DY = YS
       END IF
      END IF
      IF (NQ .EQ. 1) THEN
       DY = SIGN(DY, SAVE2(J))
      ELSE
       DY = SIGN(DY, YH(J,3))
      END IF
      DY = (Y(J) + DY) - Y(J)
      YJ = Y(J)
      Y(J) = Y(J) + DY
       CALL F (N,T,Y,SAVE1,TEMP,KF,CF,AF,RU,NREACT,
     8         NSPECS,BF,STOREC,STOPRO)
      IF (N .EQ. 0) THEN
       JSTATE = 6
       RETURN
      END IF
      Y(J) = YJ
      FACTOR = -EL(1,NQ)*H/DY
      DO 140 I = 1,N
  140   DFDY(I,J) = (SAVE1(I) - SAVE2(I))*FACTOR
C                                        Step 1
      DIFF = ABS(SAVE2(1) - SAVE1(1))
      IMAX = 1
      DO 150 I = 2,N
       IF (ABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
        IMAX = I
        DIFF = ABS(SAVE2(I) - SAVE1(I))
       END IF
  150   CONTINUE
C                                        Step 2
      IF (MIN(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX))) .GT. 0.E0) THEN
       SCALE = MAX(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX)))
C                                        Step 3
       IF (DIFF .GT. BU*SCALE) THEN
        FAC(J) = MAX(FACMIN, FAC(J)*.1E0)
       ELSE IF (BR*SCALE .LE. DIFF .AND. DIFF .LE. BL*SCALE) THEN
        FAC(J) = MIN(FAC(J)*10.E0, FACMAX)
C                                        Step 4
       ELSE IF (DIFF .LT. BR*SCALE) THEN
        FAC(J) = MIN(BP*FAC(J), FACMAX)
       END IF
      END IF
  170   CONTINUE
     IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)/(-EL(1,NQ)*H)
     NFE = NFE + N
    END IF
```

```
      IF (IMPL .EQ. 0) THEN
       DO 190 I = 1,N
190      DFDY(I,I) = DFDY(I,I) + 1.E0
      ELSE IF (IMPL .EQ. 1) THEN
       CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
       IF (N .EQ. 0) THEN
        JSTATE = 9
        RETURN
       END IF
       DO 210 J = 1,N
        DO 210 I = 1,N
210       DFDY(I,J) = DFDY(I,J) + A(I,J)
      ELSE IF (IMPL .EQ. 2) THEN
       CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
       IF (N .EQ. 0) THEN
        JSTATE = 9
        RETURN
       END IF
       DO 230 I = 1,NDE
230      DFDY(I,I) = DFDY(I,I) + A(I,1)
      END IF
      CALL SGEFA (DFDY, MATDIM, N, IPVT, INFO)
      IF (INFO .NE. 0) IER = .TRUE.
     ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
      IF (MITER .EQ. 4) THEN
       CALL JACOBN (N, T, Y, DFDY(ML+1,1), MATDIM, ML, MU)
       IF (N .EQ. 0) THEN
        JSTATE = 8
        RETURN
       END IF
       FACTOR = -EL(1,NQ)*H
       MW = ML + MU + 1
       DO 260 J = 1,N
        I1 = MAX(ML+1, MW+1-J)
        I2 = MIN(MW+N-J, MW+ML)
        DO 260 I = I1,I2
260       DFDY(I,J) = FACTOR*DFDY(I,J)
     ELSE IF (MITER .EQ. 5) THEN
       BR = UROUND**(.875E0)
       BL = UROUND**(.75E0)
       BU = UROUND**(.25E0)
       BP = UROUND**(-.15E0)
       FACMIN = UROUND**(.78E0)
       MW = ML + MU + 1
       J2 = MIN(MW, N)
       DO 340 J = 1,J2
        DO 290 K = J,N,MW
         YS = MAX(ABS(YWT(K)), ABS(Y(K)))
280       DY = FAC(K)*YS
         IF (DY .EQ. 0.E0) THEN
          IF (FAC(K) .LT. FACMAX) THEN
           FAC(K) = MIN(100.E0*FAC(K), FACMAX)
```

```
          GO TO 280
         ELSE
          DY = YS
         END IF
        END IF
        IF (NQ .EQ. 1) THEN
         DY = SIGN(DY, SAVE2(K))
        ELSE
         DY = SIGN(DY, YH(K,3))
        END IF
        DY = (Y(K) + DY) - Y(K)
        DFDY(MW,K) = Y(K)
290       Y(K) = Y(K) + DY
        CALL F (N,T,Y,SAVE1,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
     8    STOREC,STOPRO)
       IF (N .EQ. 0) THEN
        JSTATE = 6
        RETURN
       END IF
       DO 330 K = J,N,MW
        Y(K) = DFDY(MW,K)
        YS = MAX(ABS(YWT(K)), ABS(Y(K)))
        DY = FAC(K)*YS
        IF (DY .EQ. 0.E0) DY = YS
        IF (NQ .EQ. 1) THEN
         DY = SIGN(DY, SAVE2(K))
        ELSE
         DY = SIGN(DY, YH(K,3))
        END IF
        DY = (Y(K) + DY) - Y(K)
        FACTOR = -EL(1,NQ)*H/DY
        I1 = MAX(ML+1, MW+1-K)
        I2 = MIN(MW+N-K, MW+ML)
        DO 300 I = I1,I2
         I3 = K + I - MW
300       DFDY(I,K) = FACTOR*(SAVE1(I3) - SAVE2(I3))
C                                    Step 1
       IMAX = MAX(1, K - MU)
       DIFF = ABS(SAVE2(IMAX) - SAVE1(IMAX))
       I1 = IMAX
       I2 = MIN(K + ML, N)
       DO 310 I = I1+1,I2
        IF (ABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
         IMAX = I
         DIFF = ABS(SAVE2(I) - SAVE1(I))
        END IF
310       CONTINUE
C                                    Step 2
       IF (MIN(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX))) .GT.0.E0) THEN
        SCALE = MAX(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX)))
C                                    Step 3
        IF (DIFF .GT. BU*SCALE) THEN
```

```
            FAC(K) = MAX(FACMIN, FAC(K)*.1E0)
           ELSE IF (BR*SCALE .LE.DIFF .AND. DIFF .LE.BL*SCALE) THEN
            FAC(K) = MIN(FAC(K)*10.E0, FACMAX)
C                                        Step 4
           ELSE IF (DIFF .LT. BR*SCALE) THEN
            FAC(K) = MIN(BP*FAC(K), FACMAX)
           END IF
          END IF
330        CONTINUE
340       CONTINUE
        NFE = NFE + J2
       END IF
       IF (ISWFLG .EQ. 3) THEN
        DFDYMX = 0.E0
        DO 345 J = 1,N
         I1 = MAX(ML+1, MW+1-J)
         I2 = MIN(MW+N-J, MW+ML)
         DO 345 I = I1,I2
345        DFDYMX = MAX(DFDYMX, ABS(DFDY(I,J)))
        BND = 0.E0
        IF (DFDYMX .NE. 0.E0) THEN
         DO 350 J = 1,N
          I1 = MAX(ML+1, MW+1-J)
          I2 = MIN(MW+N-J, MW+ML)
          DO 350 I = I1,I2
350         BND = BND + (DFDY(I,J)/DFDYMX)**2
         BND = DFDYMX*SQRT(BND)/(-EL(1,NQ)*H)
        END IF
       END IF
       IF (IMPL .EQ. 0) THEN
        DO 360 J = 1,N
360       DFDY(MW,J) = DFDY(MW,J) + 1.E0
       ELSE IF (IMPL .EQ. 1) THEN
        CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
        DO 380 J = 1,N
         I1 = MAX(ML+1, MW+1-J)
         I2 = MIN(MW+N-J, MW+ML)
         DO 380 I = I1,I2
380        DFDY(I,J) = DFDY(I,J) + A(I,J)
       ELSE IF (IMPL .EQ. 2) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
         JSTATE = 9
         RETURN
        END IF
        DO 400 J = 1,NDE
400       DFDY(MW,J) =  DFDY(MW,J) + A(J,1)
       END IF
```

```
      CALL SGBFA (DFDY, MATDIM, N, ML, MU, IPVT, INFO)
       IF (INFO .NE. 0) IER = .TRUE.
     ELSE IF (MITER .EQ. 3) THEN
      IFLAG = 1
      CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
    8          N, NDE, IFLAG)
      IF (N .EQ. 0) THEN
       JSTATE = 10
       RETURN
      END IF
     END IF
     END

      SUBROUTINE SDSCL (HMAX,N,NQ,RMAX,H,RC,RH,YH)
C***BEGIN PROLOGUE  SDSCL
C***REFER TO  SDRIV3
C  This subroutine rescales the YH array whenever the step size
C  is changed.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  850319  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSCL
      REAL H, HMAX, RC, RH, RMAX, R1, YH(N,*)

C***FIRST EXECUTABLE STATEMENT  SDSCL
      IF (H .LT. 1.E0) THEN
       RH = MIN(ABS(H)*RH, ABS(H)*RMAX, HMAX)/ABS(H)
      ELSE
       RH = MIN(RH, RMAX, HMAX/ABS(H))
      END IF
      R1 = 1.E0
      DO 10 J = 1,NQ
       R1 = R1*RH
       DO 10 I = 1,N
10       YH(I,J+1) = YH(I,J+1)*R1
      H = H*RH
      RC = RC*RH
      END
      SUBROUTINE SDSTP
(EPS,F,FA,HMAX,IMPL,JACOBN,MATDIM,MAXORD,MINT,
     8
MITER,ML,MU,N,NDE,YWT,UROUND,USERS,AVGH,AVGORD,H,HUSED,JTAS
K,
     8
MNTOLD,MTROLD,NFE,NJE,NQUSED,NSTEP,T,Y,YH,A,CONVRG,DFDY,EL,FA
C,
     8
HOLD,IPVT,JSTATE,NQ,NWAIT,RC,RMAX,SAVE1,SAVE2,TQ,TREND,ISWFLG,
```

```
     8
MTRSV,MXRDSV,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,STOREC,STOPRO)
C***BEGIN PROLOGUE  SDSTP
C***REFER TO  SDRIV3
C  SDSTP performs one step of the integration of an initial value
C  problem for a system of ordinary differential equations.
C  Communication with SDSTP is done with the following variables:
C
C   YH    An N by MAXORD+1 array containing the dependent variables
C         and their scaled derivatives.  MAXORD, the maximum order
C         used, is currently 12 for the Adams methods and 5 for the
C         Gear methods.  YH(I,J+1) contains the J-th derivative of
C         Y(I), scaled by H**J/factorial(J).  Only Y(I),
C         1 .LE. I .LE. N, need be set by the calling program on
C         the first entry.  The YH array should not be altered by
C         the calling program.  When referencing YH as a
C         2-dimensional array, use a column length of N, as this is
C         the value used in SDSTP.
C   DFDY  A block of locations used for partial derivatives if MITER
C         is not 0.  If MITER is 1 or 2 its length must be at least
C         N*N.  If MITER is 4 or 5 its length must be at least
C         (2*ML+MU+1)*N.
C   YWT   An array of N locations used in convergence and error tests
C   SAVE1
C   SAVE2  Arrays of length N used for temporary storage.
C   IPVT  An integer array of length N used by the linear system
C         solvers for the storage of row interchange information.
C   A     A block of locations used to store the matrix A, when using
C         the implicit method.  If IMPL is 1, A is a MATDIM by N
C         array.  If MITER is 1 or 2 MATDIM is N, and if MITER is 4
C         or 5 MATDIM is 2*ML+MU+1.  If IMPL is 2 its length is N.
C   JTASK  An integer used on input.
C         It has the following values and meanings:
C           .EQ. 0  Perform the first step.  This value enables
C                   the subroutine to initialize itself.
C           .GT. 0  Take a new step continuing from the last.
C                   Assumes the last step was successful and
C                   user has not changed any parameters.
C           .LT. 0  Take a new step with a new value of H and/or
C                   MINT and/or MITER.
C   JSTATE  A completion code with the following meanings:
C            1  The step was successful.
C            2  A solution could not be obtained with H .NE. 0.
C            3  A solution was not obtained in MXTRY attempts.
C            4  For IMPL .NE. 0, the matrix A is singular.
C          On a return with JSTATE .GT. 1, the values of T and
C          the YH array are as of the beginning of the last
C          step, and H is the last step size attempted.
C***ROUTINES CALLED  SDNTL,SDPST,SDCOR,SDPSC,SDSCL,SNRM2
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870810  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
```

```
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C        SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSTP
     EXTERNAL F, JACOBN, FA, USERS
     REAL A(MATDIM,*), AVGH, AVGORD, BIAS1, BIAS2, BIAS3,
    8    BND, CTEST, D, DENOM, DFDY(MATDIM,*), D1, EL(13,12), EPS,
    8    ERDN, ERUP, ETEST, FAC(*), H, HMAX, HN, HOLD, HS, HUSED,
    8    NUMER, RC, RCTEST, RH, RH1, RH2, RH3, RMAX, RMFAIL, RMNORM,
    8    SAVE1(*), SAVE2(*), SNRM2, T, TOLD, TQ(3,12), TREND, TRSHLD,
    8    UROUND, Y(*), YH(N,*), YWT(*), Y0NRM
     INTEGER IPVT(*)
         INTEGER NREACT,NSPECS
         REAL TEMP,KF(*),CF(*),AF(*),RU,BF(*),
    1STOREC(80,80),STOPRO(80,80)
     LOGICAL CONVRG, EVALFA, EVALJC, IER, SWITCH
     PARAMETER(BIAS1 = 1.3E0, BIAS2 = 1.2E0, BIAS3 = 1.4E0, MXFAIL = 3,
    8        MXITER = 3, MXTRY = 50, RCTEST = .3E0, RMFAIL = 2.E0,
    8        RMNORM = 10.E0, TRSHLD = 1.E0)
     DATA IER /.FALSE./

C***FIRST EXECUTABLE STATEMENT  SDSTP
     NSV = N
     BND = 0.E0
     SWITCH = .FALSE.
     NTRY = 0
     TOLD = T
     NFAIL = 0
     IF (JTASK .LE. 0) THEN
       CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
    8           MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
    8           UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
    8           YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
    8           RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE,TEMP,KF,CF,
    8           AF,RU,NREACT,NSPECS,BF,STOREC,STOPRO)
       IF (N .EQ. 0) GO TO 440
       IF (H .EQ. 0.E0) GO TO 400
       IF (IER) GO TO 420
     END IF
100  NTRY = NTRY + 1
     IF (NTRY .GT. MXTRY) GO TO 410
     T = T + H
     CALL SDPSC (1, N, NQ,  YH)
     EVALJC = ((ABS(RC - 1.E0) .GT. RCTEST) .AND. (MITER .NE. 0))
     EVALFA = .NOT. EVALJC
C
110  ITER = 0
     DO 115 I = 1,N
115    Y(I) = YH(I,1)
     CALL F (N, T, Y, SAVE2,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
    8        STOREC,STOPRO)
     IF (N .EQ. 0) THEN
      JSTATE = 6
```

```
      GO TO 430
     END IF
     NFE = NFE + 1
     IF (EVALJC .OR. IER) THEN
      CALL SDPST (EL, F, FA, H, IMPL, JACOBN, MATDIM, MITER, ML,
     8           MU, N, NDE, NQ, SAVE2, T, USERS, Y, YH, YWT, UROUND,
     8           NFE, NJE,  A, DFDY, FAC, IER, IPVT, SAVE1, ISWFLG,
     8            BND, JSTATE,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,
     8          STOREC,STOPRO)
      IF (N .EQ. 0) GO TO 430
      IF (IER) GO TO 160
      CONVRG = .FALSE.
      RC = 1.E0
     END IF
     DO 125 I = 1,N
 125   SAVE1(I) = 0.E0
C               Up to MXITER corrector iterations are taken.
C               Convergence is tested by requiring the r.m.s.
C               norm of changes to be less than EPS.  The sum of
C               the corrections is accumulated in the vector
C               SAVE1(I).  It is approximately equal to the L-th
C               derivative of Y multiplied by
C               H**L/(factorial(L-1)*EL(L,NQ)), and is thus
C               proportional to the actual errors to the lowest
C               power of H present (H**L).  The YH array is not
C               altered in the correction loop.  The norm of the
C               iterate difference is stored in D.  If
C               ITER .GT. 0, an estimate of the convergence rate
C               constant is stored in TREND, and this is used in
C               the convergence test.
C
 130  CALL SDCOR (DFDY, EL, FA, H, IMPL, IPVT, MATDIM, MITER, ML,
     8         MU, N, NDE, NQ, T, USERS, Y, YH, YWT,  EVALFA, SAVE1,
     8        SAVE2,  A, D, JSTATE)
      IF (N .EQ. 0) GO TO 430
     IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
      IF (ITER .EQ. 0) THEN
       NUMER = SNRM2(N, SAVE1, 1)
       DO 132 I = 1,N
 132     DFDY(1,I) = SAVE1(I)
       Y0NRM = SNRM2(N, YH, 1)
      ELSE
       DENOM = NUMER
       DO 134 I = 1,N
 134     DFDY(1,I) = SAVE1(I) - DFDY(1,I)
       NUMER = SNRM2(N, DFDY, MATDIM)
       IF (EL(1,NQ)*NUMER .LE. 100.E0*UROUND*Y0NRM) THEN
        IF (RMAX .EQ. RMFAIL) THEN
         SWITCH = .TRUE.
         GO TO 170
        END IF
       END IF
```

```
      DO 136 I = 1,N
136     DFDY(1,I) = SAVE1(I)
      IF (DENOM .NE. 0.E0)
     8  BND = MAX(BND, NUMER/(DENOM*ABS(H)*EL(1,NQ)))
       END IF
      END IF
      IF (ITER .GT. 0) TREND = MAX(.9E0*TREND, D/D1)
      D1 = D
      CTEST = MIN(2.E0*TREND, 1.E0)*D
      IF (CTEST .LE. EPS) GO TO 170
      ITER = ITER + 1
      IF (ITER .LT. MXITER) THEN
       DO 140 I = 1,N
140     Y(I) = YH(I,1) + EL(1,NQ)*SAVE1(I)
       CALL F (N, T, Y, SAVE2,TEMP,KF,CF,AF,RU,NREACT,NSPECS,
     8        BF,STOREC,STOPRO)
       IF (N .EQ. 0) THEN
        JSTATE = 6
        GO TO 430
       END IF
       NFE = NFE + 1
       GO TO 130
      END IF
C               The corrector iteration failed to converge in
C               MXITER tries.  If partials are involved but are
C               not up to date, they are reevaluated for the next
C               try.  Otherwise the YH array is retracted to its
C               values before prediction, and H is reduced, if
C               possible.  If not, a no-convergence exit is taken.
      IF (CONVRG) THEN
       EVALJC = .TRUE.
       EVALFA = .FALSE.
       GO TO 110
      END IF
160   T = TOLD
      CALL SDPSC (-1, N, NQ,  YH)
      NWAIT = NQ + 2
      IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
      IF (ITER .EQ. 0) THEN
       RH = .3E0
      ELSE
       RH = .9E0*(EPS/CTEST)**(.2E0)
      END IF
      IF (RH*H .EQ. 0.E0) GO TO 400
      CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
      GO TO 100
C                  The corrector has converged.  CONVRG is set
C                  to .TRUE. if partial derivatives were used,
C                  to indicate that they may need updating on
C                  subsequent steps.  The error test is made.
170   CONVRG = (MITER .NE. 0)
      DO 180 I = 1,NDE
```

```
 180   SAVE2(I) = SAVE1(I)/YWT(I)
      ETEST = SNRM2(NDE, SAVE2, 1)/(TQ(2,NQ)*SQRT(REAL(NDE)))
C
C                  The error test failed.  NFAIL keeps track of
C                  multiple failures.  Restore T and the YH
C                  array to their previous values, and prepare
C                  to try the step again.  Compute the optimum
C                  step size for this or one lower order.
      IF (ETEST .GT. EPS) THEN
       T = TOLD
       CALL SDPSC (-1, N, NQ,  YH)
       NFAIL = NFAIL + 1
       IF (NFAIL .LT. MXFAIL) THEN
         IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
        RH2 = 1.E0/(BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
        IF (NQ .GT. 1) THEN
         DO 190 I = 1,NDE
 190        SAVE2(I) = YH(I,NQ+1)/YWT(I)
        ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*SQRT(REAL(NDE)))
        RH1 = 1.E0/MAX(1.E0, BIAS1*(ERDN/EPS)**(1.E0/REAL(NQ)))
         IF (RH2 .LT. RH1) THEN
          NQ = NQ - 1
          RC = RC*EL(1,NQ)/EL(1,NQ+1)
          RH = RH1
         ELSE
          RH = RH2
         END IF
       ELSE
         RH = RH2
       END IF
       NWAIT = NQ + 2
       IF (RH*H .EQ. 0.E0) GO TO 400
       CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
       GO TO 100
      END IF
C           Control reaches this section if the error test has
C           failed MXFAIL or more times.  It is assumed that the
C           derivatives that have accumulated in the YH array have
C           errors of the wrong order.  Hence the first derivative
C           is recomputed, the order is set to 1, and the step is
C           retried.
      NFAIL = 0
      JTASK = 2
      DO 215 I = 1,N
 215    Y(I) = YH(I,1)
      CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
     8           MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
     8           UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
     8           YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
     8           RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE,TEMP,KF,CF,
     8           AF,RU,NREACT,NSPECS,BF,STOREC,STOPRO)
     RMAX = RMNORM
```

```
      IF (N .EQ. 0) GO TO 440
      IF (H .EQ. 0.E0) GO TO 400
      IF (IER) GO TO 420
      GO TO 100
     END IF
C                 After a successful step, update the YH array.
     NSTEP = NSTEP + 1
     HUSED = H
     NQUSED = NQ
     AVGH = (REAL(NSTEP-1)*AVGH + H)/REAL(NSTEP)
     AVGORD = (REAL(NSTEP-1)*AVGORD + REAL(NQ))/REAL(NSTEP)
     DO 230 J = 1,NQ+1
      DO 230 I = 1,N
 230    YH(I,J) = YH(I,J) + EL(J,NQ)*SAVE1(I)
     DO 235 I = 1,N
 235   Y(I) = YH(I,1)
C                              If ISWFLG is 3, consider
C                             changing integration methods.
     IF (ISWFLG .EQ. 3) THEN
      IF (BND .NE. 0.E0) THEN
       IF (MINT .EQ. 1 .AND. NQ .LE. 5) THEN
        HN = ABS(H)/MAX(UROUND, (ETEST/EPS)**(1.E0/REAL(NQ+1)))
        HN = MIN(HN, 1.E0/(2.E0*EL(1,NQ)*BND))
        HS = ABS(H)/MAX(UROUND,
   8     (ETEST/(EPS*EL(NQ+1,1)))**(1.E0/REAL(NQ+1)))
        IF (HS .GT. 1.2E0*HN) THEN
         MINT = 2
         MNTOLD = MINT
         MITER = MTRSV
         MTROLD = MITER
         MAXORD = MIN(MXRDSV, 5)
         RC = 0.E0
         RMAX = RMNORM
         TREND = 1.E0
         CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
         NWAIT = NQ + 2
        END IF
       ELSE IF (MINT .EQ. 2) THEN
        HS = ABS(H)/MAX(UROUND, (ETEST/EPS)**(1.E0/REAL(NQ+1)))
        HN = ABS(H)/MAX(UROUND,
   8     (ETEST*EL(NQ+1,1)/EPS)**(1.E0/REAL(NQ+1)))
        HN = MIN(HN, 1.E0/(2.E0*EL(1,NQ)*BND))
        IF (HN .GE. HS) THEN
         MINT = 1
         MNTOLD = MINT
         MITER = 0
         MTROLD = MITER
         MAXORD = MIN(MXRDSV, 12)
         RMAX = RMNORM
         TREND = 1.E0
         CONVRG = .FALSE.
         CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
```

```
          NWAIT = NQ + 2
        END IF
       END IF
      END IF
     END IF
     IF (SWITCH) THEN
      MINT = 2
      MNTOLD = MINT
      MITER = MTRSV
      MTROLD = MITER
      MAXORD = MIN(MXRDSV, 5)
      NQ = MIN(NQ, MAXORD)
      RC = 0.E0
      RMAX = RMNORM
      TREND = 1.E0
      CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
      NWAIT = NQ + 2
     END IF
C                Consider changing H if NWAIT = 1.  Otherwise
C                decrease NWAIT by 1.  If NWAIT is then 1 and
C                NQ.LT.MAXORD, then SAVE1 is saved for use in
C                a possible order increase on the next step.
C
     IF (JTASK .EQ. 0 .OR. JTASK .EQ. 2) THEN
      RH = 1.E0/MAX(UROUND, BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
      IF (RH.GT.TRSHLD) CALL SDSCL (HMAX, N, NQ, RMAX, H, RC, RH, YH)
     ELSE IF (NWAIT .GT. 1) THEN
      NWAIT = NWAIT - 1
      IF (NWAIT .EQ. 1 .AND. NQ .LT. MAXORD) THEN
       DO 250 I = 1,NDE
250      YH(I,MAXORD+1) = SAVE1(I)
      END IF
C          If a change in H is considered, an increase or decrease in
C          order by one is considered also.  A change in H is made
C          only if it is by a factor of at least TRSHLD.  Factors
C          RH1, RH2, and RH3 are computed, by which H could be
C          multiplied at order NQ - 1, order NQ, or order NQ + 1,
C          respectively.  The largest of these is determined and the
C          new order chosen accordingly.  If the order is to be
C          increased, we compute one additional scaled derivative.
C          If there is a change of order, reset NQ and the
C          coefficients.  In any case H is reset according to RH and
C          the YH array is rescaled.
     ELSE
      IF (NQ .EQ. 1) THEN
       RH1 = 0.E0
      ELSE
       DO 270 I = 1,NDE
270      SAVE2(I) = YH(I,NQ+1)/YWT(I)
       ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*SQRT(REAL(NDE)))
       RH1 = 1.E0/MAX(UROUND, BIAS1*(ERDN/EPS)**(1.E0/REAL(NQ)))
      END IF
```

```
      RH2 = 1.E0/MAX(UROUND, BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
      IF (NQ .EQ. MAXORD) THEN
        RH3 = 0.E0
      ELSE
        DO 290 I = 1,NDE
  290     SAVE2(I) = (SAVE1(I) - YH(I,MAXORD+1))/YWT(I)
        ERUP = SNRM2(NDE, SAVE2, 1)/(TQ(3,NQ)*SQRT(REAL(NDE)))
        RH3 = 1.E0/MAX(UROUND, BIAS3*(ERUP/EPS)**(1.E0/REAL(NQ+2)))
      END IF
      IF (RH1 .GT. RH2 .AND. RH1 .GE. RH3) THEN
        RH = RH1
        IF (RH .LE. TRSHLD) GO TO 380
        NQ = NQ - 1
        RC = RC*EL(1,NQ)/EL(1,NQ+1)
      ELSE IF (RH2 .GE. RH1 .AND. RH2 .GE. RH3) THEN
        RH = RH2
        IF (RH .LE. TRSHLD) GO TO 380
      ELSE
        RH = RH3
        IF (RH .LE. TRSHLD) GO TO 380
        DO 360 I = 1,N
  360     YH(I,NQ+2) = SAVE1(I)*EL(NQ+1,NQ)/REAL(NQ+1)
        NQ = NQ + 1
        RC = RC*EL(1,NQ)/EL(1,NQ-1)
      END IF
      IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
        IF (BND.NE.0.E0) RH = MIN(RH, 1.E0/(2.E0*EL(1,NQ)*BND*ABS(H)))
      END IF
      CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
      RMAX = RMNORM
  380 NWAIT = NQ + 2
    END IF
C           All returns are made through this section.  H is saved
C           in HOLD to allow the caller to change H on the next step
    JSTATE = 1
    HOLD = H
    RETURN
C
  400 JSTATE = 2
    HOLD = H
    DO 405 I = 1,N
  405 Y(I) = YH(I,1)
    RETURN
C
  410 JSTATE = 3
    HOLD = H
    RETURN
C
  420 JSTATE = 4
    HOLD = H
    RETURN
C
```

```
 430  T = TOLD
      CALL SDPSC (-1, NSV, NQ,  YH)
      DO 435 I = 1,NSV
 435    Y(I) = YH(I,1)
 440  HOLD = H
      RETURN
      END
      SUBROUTINE SDZRO
(AE,F,H,N,NQ,IROOT,RE,T,YH,UROUND,B,C,FB,FC,Y)
C***BEGIN PROLOGUE  SDZRO
C***REFER TO  SDRIV3
C    This is a special purpose version of ZEROIN, modified for use with
C    the SDRIV1 package.
C
C    Sandia Mathematical Program Library
C    Mathematical Computing Services Division 5422
C    Sandia Laboratories
C    P. O. Box 5800
C    Albuquerque, New Mexico  87115
C    Control Data 6600 Version 4.5, 1 November 1971
C
C    ABSTRACT
C      ZEROIN searches for a zero of a function F(N, T, Y, IROOT)
C      between the given values B and C until the width of the
C      interval (B, C) has collapsed to within a tolerance specified
C      by the stopping criterion, ABS(B - C) .LE. 2.*(RW*ABS(B) + AE).
C
C    Description of parameters
C      F    - Name of the external function, which returns a
C             real result.  This name must be in an
C             EXTERNAL statement in the calling program.
C      B    - One end of the interval (B, C).  The value returned for
C             B usually is the better approximation to a zero of F.
C      C    - The other end of the interval (B, C).
C      RE   - Relative error used for RW in the stopping criterion.
C             If the requested RE is less than machine precision,
C             then RW is set to approximately machine precision.
C      AE   - Absolute error used in the stopping criterion.  If the
C             given interval (B, C) contains the origin, then a
C             nonzero value should be chosen for AE.
C
C    REFERENCES
C      1. L F Shampine and H A Watts, ZEROIN, A Root-Solving Routine,
C         SC-TM-70-631, Sept 1970.
C      2. T J Dekker, Finding a Zero by Means of Successive Linear
C         Interpolation, "Constructive Aspects of the Fundamental
C         Theorem of Algebra", edited by B Dejon and P Henrici, 1969.
C***ROUTINES CALLED  SDNTP
C***DATE WRITTEN  790601  (YYMMDD)
C***REVISION DATE  870511  (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
```

```
C         SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDZRO
      REAL A, ACBS, ACMB, AE, B, C, CMB, ER, F, FA, FB, FC,
     8     H, P, Q, RE, RW, T, TOL, UROUND, Y(*), YH(N,*)

C***FIRST EXECUTABLE STATEMENT  SDZRO
      ER = 4.E0*UROUND
      RW = MAX(RE, ER)
      IC = 0
      ACBS = ABS(B - C)
      A = C
      FA = FC
      KOUNT = 0
C                                        Perform interchange
   10 IF (ABS(FC) .LT. ABS(FB)) THEN
        A = B
        FA = FB
        B = C
        FB = FC
        C = A
        FC = FA
      END IF
      CMB = 0.5E0*(C - B)
      ACMB = ABS(CMB)
      TOL = RW*ABS(B) + AE
C                                    Test stopping criterion
      IF (ACMB .LE. TOL) RETURN
      IF (KOUNT .GT. 50) RETURN
C                            Calculate new iterate implicitly as
C                             B + P/Q, where we arrange P .GE. 0.
C                     The implicit form is used to prevent overflow.
      P = (B - A)*FB
      Q = FA - FB
      IF (P .LT. 0.E0) THEN
        P = -P
        Q = -Q
      END IF
C                     Update A and check for satisfactory reduction
C                     in the size of our bounding interval.
      A = B
      FA = FB
      IC = IC + 1
      IF (IC .GE. 4) THEN
        IF (8.E0*ACMB .GE. ACBS) THEN
C                                                 Bisect
          B = 0.5E0*(C + B)
          GO TO 20
        END IF
        IC = 0
      END IF
      ACBS = ACMB
C                               Test for too small a change
```

```
      IF (P .LE. ABS(Q)*TOL) THEN
C                                     Increment by tolerance
        B = B + SIGN(TOL, CMB)
C                                    Root ought to be between
C                                     B and (C + B)/2.
      ELSE IF (P .LT. CMB*Q) THEN
C                                          Interpolate
        B = B + P/Q
      ELSE
C                                            Bisect
        B = 0.5E0*(C + B)
      END IF
C                                  Have completed computation
C                                  for new iterate B.
 20   CALL SDNTP (H, 0, N, NQ, T, B, YH,  Y)
      FB = F(N, B, Y, IROOT)
      IF (N .EQ. 0) RETURN
      IF (FB .EQ. 0.E0) RETURN
      KOUNT = KOUNT + 1
C
C         Decide whether next step is interpolation or extrapolation
C
      IF (SIGN(1.0E0, FB) .EQ. SIGN(1.0E0, FC)) THEN
        C = A
        FC = FA
      END IF
      GO TO 10
      END
      SUBROUTINE SDRIV3
     (N,T,Y,F,NSTATE,TOUT,NTASK,NROOT,EPS,EWT,IERROR,
     8
     MINT,MITER,IMPL,ML,MU,MXORD,HMAX,WORK,LENW,IWORK,LENIW,JACO
     BN,
     8
     FA,NDE,MXSTEP,G,USERS,TEMP,KF,CF,AF,RU,NREACT,NSPECS,BF,STOREC
     ,
     8  STOPRO)
C***BEGIN PROLOGUE  SDRIV3
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C          Prentice Hall 1988
C
C***END PROLOGUE  SDRIV3
      EXTERNAL F, JACOBN, FA, G, USERS
      REAL AE, BIG, EPS, EWT(*), G, GLAST, H, HMAX, HSIGN,
     8     NROUND, RE, R1MACH, SIZE, SNRM2, SUM, T, TLAST, TOUT, TROOT,
     8     UROUND, WORK(*), Y(*)
      INTEGER IWORK(*)
          INTEGER NREACT,NSPECS
```

```
      REAL TEMP,KF(*),CF(*),AF(*),RU,BF(*),
     1STOREC(80,80),STOPRO(80,80)
     LOGICAL CONVRG
     CHARACTER MSG*205
     PARAMETER(NROUND = 20.E0)
     PARAMETER(IAVGH = 1, IHUSED = 2, IAVGRD = 3,
     8        IEL = 4, IH = 160, IHMAX = 161, IHOLD = 162,
     8        IHSIGN = 163, IRC = 164, IRMAX = 165, IT = 166,
     8        ITOUT = 167, ITQ = 168, ITREND = 204, IYH = 205,
     8        INDMXR = 1, INQUSD = 2, INSTEP = 3, INFE = 4, INJE = 5,
     8        INROOT = 6, ICNVRG = 7, IJROOT = 8, IJTASK = 9,
     8        IMNTLD = 10, IMTRLD = 11, INQ = 12, INRTLD = 13,
     8        INDTRT = 14, INWAIT = 15, IMNT = 16, IMTRSV = 17,
     8        IMTR = 18, IMXRDS = 19, IMXORD = 20, INDPRT = 21,
     8        INDPVT = 22)

C***FIRST EXECUTABLE STATEMENT  SDRIV3
     NPAR = N
     UROUND = R1MACH (4)
     IF (NROOT .NE. 0) THEN
      AE = R1MACH(1)
      RE = UROUND
     END IF
     IF (EPS .LT. 0.E0) THEN
       WRITE(MSG, '("SDRIV36FE Illegal input.  EPS,", E16.8,
     8  ", is negative.")') EPS
      CALL XERROR(MSG(1:60), 60, 6, 2)
      RETURN
     END IF
     IF (N .LE. 0) THEN
       WRITE(MSG, '("SDRIV37FE Illegal input.  Number of equations,",
     8  I8, ", is not positive.")') N
      CALL XERROR(MSG(1:72), 72, 7, 2)
      RETURN
     END IF
     IF (MXORD .LE. 0) THEN
       WRITE(MSG, '("SDRIV314FE Illegal input.  Maximum order,", I8,
     8  ", is not positive.")') MXORD
      CALL XERROR(MSG(1:67), 67, 14, 2)
      RETURN
     END IF
     IF ((MINT .LT. 1 .OR. MINT .GT. 3) .OR. (MINT .EQ. 3 .AND.
    8  (MITER .EQ. 0 .OR. MITER .EQ. 3 .OR. IMPL .NE. 0))
    8  .OR. (MITER .LT. 0 .OR. MITER .GT. 5) .OR.
    8  (IMPL .NE. 0 .AND. IMPL .NE. 1 .AND. IMPL .NE. 2) .OR.
    8  ((IMPL .EQ. 1 .OR. IMPL .EQ. 2) .AND. MITER .EQ. 0) .OR.
    8  (IMPL .EQ. 2 .AND. MINT .EQ. 1) .OR.
    8  (NSTATE .LT. 1 .OR. NSTATE .GT. 10)) THEN
       WRITE(MSG, '("SDRIV39FE Illegal input.  Improper value for ",
     8 "NSTATE(MSTATE), MINT, MITER or IMPL.")')
      CALL XERROR(MSG(1:81), 81, 9, 2)
      RETURN
```

```
      END IF
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
       LIWCHK = INDPVT - 1
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2 .OR. MITER .EQ. 4 .OR.
     8  MITER .EQ. 5) THEN
       LIWCHK = INDPVT + N - 1
      END IF
      IF (LENIW .LT. LIWCHK) THEN
        WRITE(MSG, '("SDRIV310FE Illegal input.  Insufficient ",
     8  "storage allocated for the IWORK array.  Based on the ")')
        WRITE(MSG(94:), '("value of the input parameters involved, ",
     8  "the required storage is", I8)') LIWCHK
       CALL XERROR(MSG(1:164), 164, 10, 2)
       RETURN
      END IF
C                                  Allocate the WORK array
C                               IYH is the index of YH in WORK
      IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
       MAXORD = MIN(MXORD, 12)
      ELSE IF (MINT .EQ. 2) THEN
       MAXORD = MIN(MXORD, 5)
      END IF
      IDFDY = IYH + (MAXORD + 1)*N
C                                IDFDY is the index of DFDY
C
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3)  THEN
       IYWT = IDFDY
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2)  THEN
       IYWT = IDFDY + N*N
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5)  THEN
       IYWT = IDFDY + (2*ML + MU + 1)*N
      END IF
C                                  IYWT is the index of YWT
      ISAVE1 = IYWT + N
C                               ISAVE1 is the index of SAVE1
      ISAVE2 = ISAVE1 + N
C                               ISAVE2 is the index of SAVE2
      IGNOW = ISAVE2 + N
C                                IGNOW is the index of GNOW
      ITROOT = IGNOW + NROOT
C                               ITROOT is the index of TROOT
      IFAC = ITROOT + NROOT
C                                  IFAC is the index of FAC
      IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. MINT .EQ. 3) THEN
       IA = IFAC + N
      ELSE
       IA = IFAC
      END IF
C                                    IA is the index of A
      IF (IMPL .EQ. 0 .OR. MITER .EQ. 3) THEN
       LENCHK = IA - 1
      ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 1 .OR. MITER .EQ. 2)) THEN
```

```
      LENCHK = IA - 1 + N*N
     ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 4 .OR. MITER .EQ. 5)) THEN
      LENCHK = IA - 1 + (2*ML + MU + 1)*N
     ELSE IF (IMPL .EQ. 2 .AND. MITER .NE. 3) THEN
      LENCHK = IA - 1 + N
     END IF
     IF (LENW .LT. LENCHK) THEN
       WRITE(MSG, '("SDRIV38FE Illegal input.  Insufficient ",
    8  "storage allocated for the WORK array.  Based on the ")')
       WRITE(MSG(92:), '("value of the input parameters involved, ",
    8  "the required storage is", I8)') LENCHK
      CALL XERROR(MSG(1:162), 162, 8, 2)
      RETURN
     END IF
     IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
      MATDIM = 1
     ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
      MATDIM = N
     ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
      MATDIM = 2*ML + MU + 1
     END IF
     IF (IMPL .EQ. 0 .OR. IMPL .EQ. 1) THEN
      NDECOM = N
     ELSE IF (IMPL .EQ. 2) THEN
      NDECOM = NDE
     END IF
     IF (NSTATE .EQ. 1) THEN
C                                  Initialize parameters
      IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
       IWORK(IMXORD) = MIN(MXORD, 12)
      ELSE IF (MINT .EQ. 2) THEN
       IWORK(IMXORD) = MIN(MXORD, 5)
      END IF
      IWORK(IMXRDS) = MXORD
      IF (MINT .EQ. 1 .OR. MINT .EQ. 2) THEN
       IWORK(IMNT) = MINT
       IWORK(IMTR) = MITER
       IWORK(IMNTLD) = MINT
       IWORK(IMTRLD) = MITER
      ELSE IF (MINT .EQ. 3) THEN
       IWORK(IMNT) = 1
       IWORK(IMTR) = 0
       IWORK(IMNTLD) = IWORK(IMNT)
       IWORK(IMTRLD) = IWORK(IMTR)
       IWORK(IMTRSV) = MITER
      END IF
      WORK(IHMAX) = HMAX
      H = (TOUT - T)*(1.E0 - 4.E0*UROUND)
      H = SIGN(MIN(ABS(H), HMAX), H)
      WORK(IH) = H
      HSIGN = SIGN(1.E0, H)
      WORK(IHSIGN) = HSIGN
```

```
      IWORK(IJTASK) = 0
      WORK(IAVGH) = 0.E0
      WORK(IHUSED) = 0.E0
      WORK(IAVGRD) = 0.E0
      IWORK(INDMXR) = 0
      IWORK(INQUSD) = 0
      IWORK(INSTEP) = 0
      IWORK(INFE) = 0
      IWORK(INJE) = 0
      IWORK(INROOT) = 0
      WORK(IT) = T
      IWORK(ICNVRG) = 0
      IWORK(INDPRT) = 0
C                                  Set initial conditions
      DO 30 I = 1,N
       JYH = I + IYH - 1
 30     WORK(JYH) = Y(I)
      IF (T .EQ. TOUT) RETURN
      GO TO 180
     END IF
C                              On a continuation, check
C                              that output points have
C                              been or will be overtaken.
     IF (IWORK(ICNVRG) .EQ. 1) THEN
      CONVRG = .TRUE.
     ELSE
      CONVRG = .FALSE.
     END IF
     T = WORK(IT)
     H = WORK(IH)
     HSIGN = WORK(IHSIGN)
     IF (IWORK(IJTASK) .EQ. 0) GO TO 180
C
C                         IWORK(IJROOT) flags unreported
C                         roots, and is set to the value of
C                         NTASK when a root was last selected.
C                         It is set to zero when all roots
C                         have been reported.  IWORK(INROOT)
C                         contains the index and WORK(ITOUT)
C                         contains the value of the root last
C                         selected to be reported.
C                         IWORK(INRTLD) contains the value of
C                         NROOT and IWORK(INDTRT) contains
C                         the value of ITROOT when the array
C                         of roots was last calculated.
     IF (NROOT .NE. 0) THEN
      JROOT = IWORK(IJROOT)
      IF (JROOT .GT. 0) THEN
C                           TOUT has just been reported.
C                           If TROOT .LE. TOUT, report TROOT.
       IF (NSTATE .NE. 5) THEN
        IF (TOUT*HSIGN .GE. WORK(ITOUT)*HSIGN) THEN
```

```
          TROOT = WORK(ITOUT)
          CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH),  Y)
          T = TROOT
          NSTATE = 5
          GO TO 580
         END IF
C                              A root has just been reported.
C                              Select the next root.
       ELSE
        TROOT = T
        IROOT = 0
        DO 50 I = 1,IWORK(INRTLD)
         JTROOT = IWORK(INDTRT) + I - 1
         IF (WORK(JTROOT)*HSIGN .LE. TROOT*HSIGN) THEN
C
C                                   Check for multiple roots.
C
          IF (WORK(JTROOT) .EQ. WORK(ITOUT) .AND.
     8     I .GT. IWORK(INROOT)) THEN
           IROOT = I
           TROOT = WORK(JTROOT)
           GO TO 60
          END IF
          IF (WORK(JTROOT)*HSIGN .GT. WORK(ITOUT)*HSIGN) THEN
           IROOT = I
           TROOT = WORK(JTROOT)
          END IF
         END IF
 50       CONTINUE
 60      IWORK(INROOT) = IROOT
       WORK(ITOUT) = TROOT
       IWORK(IJROOT) = NTASK
       IF (NTASK .EQ. 1) THEN
        IF (IROOT .EQ. 0) THEN
         IWORK(IJROOT) = 0
        ELSE
         IF (TOUT*HSIGN .GE. TROOT*HSIGN) THEN
          CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT,WORK(IYH),Y)
          NSTATE = 5
          T = TROOT
          GO TO 580
         END IF
        END IF
       ELSE IF (NTASK .EQ. 2 .OR. NTASK .EQ. 3) THEN
C
C                        If there are no more roots, or the
C                        user has altered TOUT to be less
C                        than a root, set IJROOT to zero.
C
        IF (IROOT .EQ. 0 .OR. (TOUT*HSIGN .LT. TROOT*HSIGN)) THEN
         IWORK(IJROOT) = 0
        ELSE
```

```
            CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH), Y)
            NSTATE = 5
            T = TROOT
            GO TO 580
           END IF
          END IF
         END IF
        END IF
       END IF
C
       IF (NTASK .EQ. 1) THEN
        NSTATE = 2
        IF (T*HSIGN .GE. TOUT*HSIGN) THEN
         CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
         T = TOUT
         GO TO 580
        END IF
       ELSE IF (NTASK .EQ. 2) THEN
C                                      Check if TOUT has
C                                      been reset .LT. T
        IF (T*HSIGN .GT. TOUT*HSIGN) THEN
          WRITE(MSG, '("SDRIV32WRN With NTASK=", I1, " on input, ",
     8     "T,", E16.8, ", was beyond TOUT,", E16.8, ".  Solution",
     8    " obtained by interpolation.")') NTASK, T, TOUT
          CALL XERROR(MSG(1:124), 124, 2, 0)
          CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
         T = TOUT
         NSTATE = 2
         GO TO 580
        END IF
C                          Determine if TOUT has been overtaken
C
       IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT)))
THEN
        T = TOUT
        NSTATE = 2
        GO TO 560
       END IF
C                                  If there are no more roots
C                                  to report, report T.
       IF (NSTATE .EQ. 5) THEN
        NSTATE = 2
        GO TO 560
       END IF
       NSTATE = 2
C                                      See if TOUT will
C                                      be overtaken.
       IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
        H = TOUT - T
        IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
        WORK(IH) = H
        IF (H .EQ. 0.E0) GO TO 670
```

```
        IWORK(IJTASK) = -1
       END IF
      ELSE IF (NTASK .EQ. 3) THEN
       NSTATE = 2
       IF (T*HSIGN .GT. TOUT*HSIGN) THEN
          WRITE(MSG, '("SDRIV32WRN With NTASK=", I1, " on input, ",
     8     "T,", E16.8, ", was beyond TOUT,", E16.8, ".  Solution",
     8    " obtained by interpolation.")') NTASK, T, TOUT
         CALL XERROR(MSG(1:124), 124, 2, 0)
         CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
         T = TOUT
         GO TO 580
       END IF
       IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT)))
THEN
        T = TOUT
        GO TO 560
       END IF
       IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
        H = TOUT - T
        IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
        WORK(IH) = H
        IF (H .EQ. 0.E0) GO TO 670
        IWORK(IJTASK) = -1
       END IF
      END IF
C                  Implement changes in MINT, MITER, and/or HMAX.
C
      IF ((MINT .NE. IWORK(IMNTLD) .OR. MITER .NE. IWORK(IMTRLD)) .AND.
     8  MINT .NE. 3 .AND. IWORK(IMNTLD) .NE. 3) IWORK(IJTASK) = -1
      IF (HMAX .NE. WORK(IHMAX)) THEN
        H = SIGN(MIN(ABS(H), HMAX), H)
        IF (H .NE. WORK(IH)) THEN
         IWORK(IJTASK) = -1
         WORK(IH) = H
        END IF
        WORK(IHMAX) = HMAX
      END IF
C
  180 NSTEPL = IWORK(INSTEP)
      DO 190 I = 1,N
       JYH = IYH + I - 1
  190   Y(I) = WORK(JYH)
      IF (NROOT .NE. 0) THEN
       DO 200 I = 1,NROOT
        JGNOW = IGNOW + I - 1
        WORK(JGNOW) = G (NPAR, T, Y, I)
        IF (NPAR .EQ. 0) THEN
         IWORK(INROOT) = I
         NSTATE = 7
         RETURN
        END IF
```

```
 200    CONTINUE
      END IF
      IF (IERROR .EQ. 1) THEN
       DO 230 I = 1,N
        JYWT = I + IYWT - 1
 230     WORK(JYWT) = 1.E0
       GO TO 410
      ELSE IF (IERROR .EQ. 5) THEN
       DO 250 I = 1,N
        JYWT = I + IYWT - 1
 250     WORK(JYWT) = EWT(I)
       GO TO 410
      END IF
C                              Reset YWT array.  Looping point.
 260  IF (IERROR .EQ. 2) THEN
       DO 280 I = 1,N
        IF (Y(I) .EQ. 0.E0) GO TO 290
        JYWT = I + IYWT - 1
 280     WORK(JYWT) = ABS(Y(I))
       GO TO 410
 290   IF (IWORK(IJTASK) .EQ. 0) THEN
        CALL F (NPAR, T, Y, WORK(ISAVE2),TEMP,KF,CF,AF,RU,
     8          NREACT,NSPECS,BF,STOREC,STOPRO)
       IF (NPAR .EQ. 0) THEN
        NSTATE = 6
        RETURN
       END IF
       IWORK(INFE) = IWORK(INFE) + 1
       IF (MITER .EQ. 3 .AND. IMPL .NE. 0) THEN
        IFLAG = 0
        CALL USERS(Y, WORK(IYH), WORK(IYWT), WORK(ISAVE1),
     8             WORK(ISAVE2), T, H, WORK(IEL), IMPL, NPAR,
     8             NDECOM, IFLAG)
        IF (NPAR .EQ. 0) THEN
         NSTATE = 10
         RETURN
        END IF
       ELSE IF (IMPL .EQ. 1) THEN
        IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
         CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
         IF (NPAR .EQ. 0) THEN
          NSTATE = 9
          RETURN
         END IF
         CALL SGEFA (WORK(IA), MATDIM, N, IWORK(INDPVT), INFO)
         IF (INFO .NE. 0) GO TO 690
         CALL
SGESL(WORK(IA),MATDIM,N,IWORK(INDPVT),WORK(ISAVE2),0)
        ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
         JAML = IA + ML
         CALL FA (NPAR, T, Y, WORK(JAML), MATDIM, ML, MU, NDECOM)
         IF (NPAR .EQ. 0) THEN
```

```
            NSTATE = 9
            RETURN
           END IF
           CALL SGBFA (WORK(IA),MATDIM,N,ML,MU,IWORK(INDPVT),INFO)
           IF (INFO .NE. 0) GO TO 690
           CALL SGBSL (WORK(IA), MATDIM, N, ML, MU, IWORK(INDPVT),
     8                 WORK(ISAVE2), 0)
          END IF
         ELSE IF (IMPL .EQ. 2) THEN
          CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
          IF (NPAR .EQ. 0) THEN
           NSTATE = 9
           RETURN
          END IF
          DO 340 I = 1,NDECOM
           JA = I + IA - 1
           JSAVE2 = I + ISAVE2 - 1
           IF (WORK(JA) .EQ. 0.E0) GO TO 690
 340        WORK(JSAVE2) = WORK(JSAVE2)/WORK(JA)
         END IF
        END IF
        DO 360 J = I,N
         JYWT = J + IYWT - 1
         IF (Y(J) .NE. 0.E0) THEN
          WORK(JYWT) = ABS(Y(J))
         ELSE
          IF (IWORK(IJTASK) .EQ. 0) THEN
           JSAVE2 = J + ISAVE2 - 1
           WORK(JYWT) = ABS(H*WORK(JSAVE2))
          ELSE
           JHYP = J + IYH + N - 1
           WORK(JYWT) = ABS(WORK(JHYP))
          END IF
         END IF
         IF (WORK(JYWT) .EQ. 0.E0) WORK(JYWT) = UROUND
 360     CONTINUE
       ELSE IF (IERROR .EQ. 3) THEN
        DO 380 I = 1,N
         JYWT = I + IYWT - 1
 380     WORK(JYWT) = MAX(EWT(1), ABS(Y(I)))
       ELSE IF (IERROR .EQ. 4) THEN
        DO 400 I = 1,N
         JYWT = I + IYWT - 1
 400     WORK(JYWT) = MAX(EWT(I), ABS(Y(I)))
       END IF
C
 410  DO 420 I = 1,N
       JYWT = I + IYWT - 1
       JSAVE2 = I + ISAVE2 - 1
 420   WORK(JSAVE2) = Y(I)/WORK(JYWT)
      SUM = SNRM2(N, WORK(ISAVE2), 1)/SQRT(REAL(N))
      IF (EPS .LT. SUM*UROUND) THEN
```

```
      EPS = SUM*UROUND*(1.E0 + 10.E0*UROUND)
       WRITE(MSG, '("SDRIV34REC At T,", E16.8, ", the requested ",
     8 "accuracy, EPS, was not obtainable with the machine ",
     8 "precision.  EPS has been increased to")') T
       WRITE(MSG(137:), '(E16.8)') EPS
       CALL XERROR(MSG(1:152), 152, 4, 1)
       NSTATE = 4
       GO TO 560
      END IF
      IF (ABS(H) .GE. UROUND*ABS(T)) THEN
       IWORK(INDPRT) = 0
      ELSE IF (IWORK(INDPRT) .EQ. 0) THEN
        WRITE(MSG, '("SDRIV35WRN At T,", E16.8, ", the step size,",
     8 E16.8, ", is smaller than the roundoff level of T.  ")') T, H
       WRITE(MSG(109:), '("This may occur if there is an abrupt ",
     8 "change in the right hand side of the differential ",
     8 "equations.")')
       CALL XERROR(MSG(1:205), 205, 5, 0)
       IWORK(INDPRT) = 1
      END IF
      IF (NTASK.NE.2) THEN
       IF ((IWORK(INSTEP)-NSTEPL) .GT. MXSTEP) THEN
         WRITE(MSG, '("SDRIV33WRN At T,", E16.8, ", ", I8,
     8   " steps have been taken without reaching TOUT,", E16.8)')
     8   T, MXSTEP, TOUT
        CALL XERROR(MSG(1:103), 103, 3, 0)
        NSTATE = 3
        GO TO 560
       END IF
      END IF
C
C     CALL SDSTP (EPS, F, FA, HMAX, IMPL, JACOBN, MATDIM, MAXORD,
C    8          MINT, MITER, ML, MU, N, NDE, YWT, UROUND, USERS,
C    8          AVGH, AVGORD, H, HUSED, JTASK, MNTOLD, MTROLD,
C    8          NFE, NJE, NQUSED, NSTEP, T, Y, YH,  A, CONVRG,
C    8          DFDY, EL, FAC, HOLD, IPVT, JSTATE, NQ, NWAIT, RC,
C    8          RMAX, SAVE1, SAVE2, TQ, TREND, ISWFLG, MTRSV, MXRDSV)
C
      CALL SDSTP (EPS, F, FA, WORK(IHMAX), IMPL, JACOBN, MATDIM,
     8          IWORK(IMXORD), IWORK(IMNT), IWORK(IMTR), ML, MU, NPAR,
     8          NDECOM, WORK(IYWT), UROUND, USERS,  WORK(IAVGH),
     8          WORK(IAVGRD), WORK(IH), WORK(IHUSED), IWORK(IJTASK),
     8          IWORK(IMNTLD), IWORK(IMTRLD), IWORK(INFE), IWORK(INJE),
     8          IWORK(INQUSD), IWORK(INSTEP), WORK(IT), Y, WORK(IYH),
     8          WORK(IA), CONVRG, WORK(IDFDY), WORK(IEL), WORK(IFAC),
     8          WORK(IHOLD), IWORK(INDPVT), JSTATE, IWORK(INQ),
     8          IWORK(INWAIT), WORK(IRC), WORK(IRMAX), WORK(ISAVE1),
     8          WORK(ISAVE2), WORK(ITQ), WORK(ITREND), MINT,
     8          IWORK(IMTRSV), IWORK(IMXRDS),TEMP,KF,CF,AF,RU,NREACT,
     8          NSPECS,BF,STOREC,STOPRO)
      T = WORK(IT)
      H = WORK(IH)
```

```
      GO TO (470, 670, 680, 690, 690, 660, 660, 660, 660, 660), JSTATE
  470 IWORK(IJTASK) = 1
C                       Determine if a root has been overtaken
      IF (NROOT .NE. 0) THEN
       IROOT = 0
       DO 500 I = 1,NROOT
        JTROOT = ITROOT + I - 1
        JGNOW = IGNOW + I - 1
        GLAST = WORK(JGNOW)
        WORK(JGNOW) = G (NPAR, T, Y, I)
        IF (NPAR .EQ. 0) THEN
         IWORK(INROOT) = I
         NSTATE = 7
         RETURN
        END IF
        IF (GLAST*WORK(JGNOW) .GT. 0.E0) THEN
         WORK(JTROOT) = T + H
        ELSE
         IF (WORK(JGNOW) .EQ. 0.E0) THEN
          WORK(JTROOT) = T
          IROOT = I
         ELSE
          IF (GLAST .EQ. 0.E0) THEN
           WORK(JTROOT) = T + H
          ELSE
           IF (ABS(WORK(IHUSED)) .GE. UROUND*ABS(T)) THEN
            TLAST = T - WORK(IHUSED)
            IROOT = I
            TROOT = T
            CALL SDZRO (AE, G, H, NPAR, IWORK(INQ), IROOT, RE, T,
     8                  WORK(IYH), UROUND,  TROOT, TLAST,
     8                  WORK(JGNOW), GLAST,  Y)
            DO 480 J = 1,N
  480         Y(J) = WORK(IYH + J -1)
            IF (NPAR .EQ. 0) THEN
             IWORK(INROOT) = I
             NSTATE = 7
             RETURN
            END IF
            WORK(JTROOT) = TROOT
           ELSE
            WORK(JTROOT) = T
            IROOT = I
           END IF
          END IF
         END IF
        END IF
  500   CONTINUE
      IF (IROOT .EQ. 0) THEN
       IWORK(IJROOT) = 0
C                                 Select the first root
      ELSE
```

```
      IWORK(IJROOT) = NTASK
      IWORK(INRTLD) = NROOT
      IWORK(INDTRT) = ITROOT
      TROOT = T + H
      DO 510 I = 1,NROOT
       JTROOT = ITROOT + I - 1
       IF (WORK(JTROOT)*HSIGN .LT. TROOT*HSIGN) THEN
        TROOT = WORK(JTROOT)
        IROOT = I
       END IF
510      CONTINUE
      IWORK(INROOT) = IROOT
      WORK(ITOUT) = TROOT
      IF (TROOT*HSIGN .LE. TOUT*HSIGN) THEN
       CALL SDNTP (H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH),  Y)
       NSTATE = 5
       T = TROOT
       GO TO 580
      END IF
     END IF
    END IF
C                    Test for NTASK condition to be satisfied
    NSTATE = 2
    IF (NTASK .EQ. 1) THEN
     IF (T*HSIGN .LT. TOUT*HSIGN) GO TO 260
     CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
     T = TOUT
     GO TO 580
C                    TOUT is assumed to have been attained
C                    exactly if T is within twenty roundoff
C                    units of TOUT, relative to max(TOUT, T).
    ELSE IF (NTASK .EQ. 2) THEN
     IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT)))
THEN
      T = TOUT
     ELSE
      IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
       H = TOUT - T
       IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
       WORK(IH) = H
       IF (H .EQ. 0.E0) GO TO 670
       IWORK(IJTASK) = -1
      END IF
     END IF
    ELSE IF (NTASK .EQ. 3) THEN
     IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT)))
THEN
      T = TOUT
     ELSE
      IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
       H = TOUT - T
       IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
```

```
          WORK(IH) = H
          IF (H .EQ. 0.E0) GO TO 670
          IWORK(IJTASK) = -1
         END IF
         GO TO 260
        END IF
       END IF
C                          All returns are made through this
C                          section.  IMXERR is determined.
 560  DO 570 I = 1,N
       JYH = I + IYH - 1
 570    Y(I) = WORK(JYH)
 580  IF (CONVRG) THEN
       IWORK(ICNVRG) = 1
      ELSE
       IWORK(ICNVRG) = 0
      END IF
      IF (IWORK(IJTASK) .EQ. 0) RETURN
      BIG = 0.E0
      IMXERR = 1
      IWORK(INDMXR) = IMXERR
      DO  590 I = 1,N
C                               SIZE = ABS(ERROR(I)/YWT(I))
       JYWT = I + IYWT - 1
       JERROR = I + ISAVE1 - 1
       SIZE = ABS(WORK(JERROR)/WORK(JYWT))
       IF (BIG .LT. SIZE) THEN
        BIG = SIZE
        IMXERR = I
        IWORK(INDMXR) = IMXERR
       END IF
 590   CONTINUE
      RETURN
C
 660  NSTATE = JSTATE
      RETURN
C                              Fatal errors are processed here
C
 670  WRITE(MSG, '("SDRIV311FE At T,", E16.8, ", the attempted ",
     8  "step size has gone to zero.  Often this occurs if the ",
     8  "problem setup is incorrect.")') T
      CALL XERROR(MSG(1:129), 129, 11, 2)
      RETURN
C
 680  WRITE(MSG, '("SDRIV312FE At T,", E16.8, ", the step size has",
     8  " been reduced about 50 times without advancing the ")') T
       WRITE(MSG(103:), '("solution.  Often this occurs if the ",
     8  "problem setup is incorrect.")')
      CALL XERROR(MSG(1:165), 165, 12, 2)
      RETURN
C
 690  WRITE(MSG, '("SDRIV313FE At T,", E16.8, ", while solving",
```

```
     8  " A*YDOT = F, A is singular.")') T
      CALL XERROR(MSG(1:74), 74, 13, 2)
      RETURN
      END
      SUBROUTINE SGBFA(ABD,LDA,N,ML,MU,IPVT,INFO)
C***BEGIN PROLOGUE  SGBFA
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C     From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C         Prentice Hall 1988
C
C***END PROLOGUE  SGBFA
      INTEGER LDA,N,ML,MU,IPVT(*),INFO
      REAL ABD(LDA,*)
C
      REAL T
       INTEGER I,ISAMAX,I0,J,JU,JZ,J0,J1,K,KP1,L,LM,M,MM,NM1

C
C***FIRST EXECUTABLE STATEMENT  SGBFA
      M = ML + MU + 1
      INFO = 0
C
C     ZERO INITIAL FILL-IN COLUMNS
C
      J0 = MU + 2
      J1 = MIN0(N,M) - 1
      IF (J1 .LT. J0) GO TO 30
      DO 20 JZ = J0, J1
        I0 = M + 1 - JZ
        DO 10 I = I0, ML
          ABD(I,JZ) = 0.0E0
   10   CONTINUE
   20 CONTINUE
   30 CONTINUE
      JZ = J1
      JU = 0
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 130
      DO 120 K = 1, NM1
        KP1 = K + 1
C
C       ZERO NEXT FILL-IN COLUMN
C
        JZ = JZ + 1
        IF (JZ .GT. N) GO TO 50
        IF (ML .LT. 1) GO TO 50
```

```
      DO 40 I = 1, ML
        ABD(I,JZ) = 0.0E0
 40     CONTINUE
 50   CONTINUE
C
C     FIND L = PIVOT INDEX
C
      LM = MIN0(ML,N-K)
      L = ISAMAX(LM+1,ABD(M,K),1) + M - 1
      IPVT(K) = L + K - M
C
C     ZERO PIVOT IMPLIES THIS COLUMN ALreadY TRIANGULARIZED
C
      IF (ABD(L,K) .EQ. 0.0E0) GO TO 100
C
C       INTERCHANGE IF NECESSARY
C
        IF (L .EQ. M) GO TO 60
          T = ABD(L,K)
          ABD(L,K) = ABD(M,K)
          ABD(M,K) = T
 60     CONTINUE
C
C       COMPUTE MULTIPLIERS
C
        T = -1.0E0/ABD(M,K)
        CALL SSCAL(LM,T,ABD(M+1,K),1)
C
C       ROW ELIMINATION WITH COLUMN INDEXING
C
        JU = MIN0(MAX0(JU,MU+IPVT(K)),N)
        MM = M
        IF (JU .LT. KP1) GO TO 90
        DO 80 J = KP1, JU
          L = L - 1
          MM = MM - 1
          T = ABD(L,J)
          IF (L .EQ. MM) GO TO 70
            ABD(L,J) = ABD(MM,J)
            ABD(MM,J) = T
 70       CONTINUE
          CALL SAXPY(LM,T,ABD(M+1,K),1,ABD(MM+1,J),1)
 80     CONTINUE
 90     CONTINUE
      GO TO 110
 100  CONTINUE
        INFO = K
 110  CONTINUE
 120 CONTINUE
 130 CONTINUE
    IPVT(N) = N
    IF (ABD(M,N) .EQ. 0.0E0) INFO = N
```

```
      RETURN
      END
      SUBROUTINE SGBSL(ABD,LDA,N,ML,MU,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGBSL
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C     From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C         Prentice Hall 1988
C
C***END PROLOGUE  SGBSL
      INTEGER LDA,N,ML,MU,IPVT(*),JOB
      REAL ABD(LDA,*),B(*)
C
      REAL SDOT,T
      INTEGER K,KB,L,LA,LB,LM,M,NM1

C***FIRST EXECUTABLE STATEMENT  SGBSL
      M = MU + ML + 1
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE L*Y = B
C
         IF (ML .EQ. 0) GO TO 30
         IF (NM1 .LT. 1) GO TO 30
           DO 20 K = 1, NM1
             LM = MIN0(ML,N-K)
             L = IPVT(K)
             T = B(L)
             IF (L .EQ. K) GO TO 10
               B(L) = B(K)
               B(K) = T
   10        CONTINUE
             CALL SAXPY(LM,T,ABD(M+1,K),1,B(K+1),1)
   20      CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
           K = N + 1 - KB
           B(K) = B(K)/ABD(M,K)
           LM = MIN0(K,M) - 1
           LA = M - LM
           LB = K - LM
           T = -B(K)
           CALL SAXPY(LM,T,ABD(LA,K),1,B(LB),1)
   40    CONTINUE
      GO TO 100
```

```
  50 CONTINUE
C
C      JOB = NONZERO, SOLVE  TRANS(A) * X = B
C      FIRST SOLVE  TRANS(U)*Y = B
C
       DO 60 K = 1, N
         LM = MIN0(K,M) - 1
         LA = M - LM
         LB = K - LM
         T = SDOT(LM,ABD(LA,K),1,B(LB),1)
         B(K) = (B(K) - T)/ABD(M,K)
  60   CONTINUE
C
C      NOW SOLVE TRANS(L)*X = Y
C
       IF (ML .EQ. 0) GO TO 90
       IF (NM1 .LT. 1) GO TO 90
         DO 80 KB = 1, NM1
           K = N - KB
           LM = MIN0(ML,N-K)
           B(K) = B(K) + SDOT(LM,ABD(M+1,K),1,B(K+1),1)
           L = IPVT(K)
           IF (L .EQ. K) GO TO 70
             T = B(L)
             B(L) = B(K)
             B(K) = T
  70       CONTINUE
  80     CONTINUE
  90   CONTINUE
 100 CONTINUE
     RETURN
     END


       SUBROUTINE XERROR(MESSG,NMESSG,NERR,LEVEL)
C***BEGIN PROLOGUE  XERROR
C***DATE WRITTEN  790801  (YYMMDD)
C***REVISION DATE  870930  (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes an error (diagnostic) message.
C***DESCRIPTION
C   From the book "Numerical Methods and Software"
C      by  D. Kahaner, C. Moler, S. Nash
C        Prentice Hall 1988
C    Abstract
C      XERROR processes a diagnostic message. It is a stub routine
C      written for the book above. Actually, XERROR is a sophisticated
C      error handling package with many options, and is described
C      in the reference below. Our version has the same calling sequence
C      but only prints an error message and either returns (if the
```

```
C      input value of ABS(LEVEL) is less than 2) or stops (if the
C      input value of ABS(LEVEL) equals 2).
C
C   Description of Parameters
C    --Input--
C      MESSG - the Hollerith message to be processed.
C      NMESSG- the actual number of characters in MESSG.
C            (this is ignored in this stub routine)
C      NERR  - the error number associated with this message.
C            NERR must not be zero.
C            (this is ignored in this stub routine)
C      LEVEL - error category.
C            =2 means this is an unconditionally fatal error.
C            =1 means this is a recoverable error.  (I.e., it is
C              non-fatal if XSETF has been appropriately called.)
C            =0 means this is a warning message only.
C            =-1 means this is a warning message which is to be
C              printed at most once, regardless of how many
C              times this call is executed.
C             (in this stub routine
C                 LEVEL=2 causes a message to be printed and then a
C                              stop.
C                 LEVEL<2 causes a message to be printed and then a
C                              return.
C
C   Examples
C      CALL XERROR('SMOOTH -- NUM WAS ZERO.',23,1,2)
C      CALL XERROR('INTEG  -- LESS THAN FULL ACCURACY ACHIEVED.',
C            43,2,1)
C     CALL XERROR('ROOTER -- ACTUAL ZERO OF F FOUND BEFORE
INTERVAL F
C   1ULLY COLLAPSED.',65,3,0)
C      CALL XERROR('EXP    -- UNDERFLOWS BEING SET TO ZERO.',39,1,-1)
C
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC
ERROR-
C            HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C            1982.
C***ROUTINES CALLED  XERRWV
C***END PROLOGUE  XERROR
    CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERROR

     CALL XERRWV(MESSG,NMESSG,NERR,LEVEL,0,0,0,0,0.,0.)
    RETURN
    END
     SUBROUTINE XERRWV(MESSG,NMESSG,NERR,LEVEL,NI,I1,I2,NR,R1,R2)
C***BEGIN PROLOGUE  XERRWV
C***DATE WRITTEN  800319  (YYMMDD)
C***REVISION DATE  870930  (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
```

```
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes error message allowing 2 integer and two real
C          values to be included in the message.
C***DESCRIPTION
C   From the book "Numerical Methods and Software"
C      by  D. Kahaner, C. Moler, S. Nash
C        Prentice Hall 1988
C    Abstract
C      XERRWV prints a diagnostic error message.
C      In addition, up to two integer values and two real
C      values may be printed along with the message.
C      A stub routine for the book above. The actual XERRWV is described
C      in the reference below and contains many other options.
C
C    Description of Parameters
C     --Input--
C      MESSG - the Hollerith message to be processed.
C      NMESSG- the actual number of characters in MESSG.
C            (ignored in this stub)
C      NERR  - the error number associated with this message.
C            NERR must not be zero.
C            (ignored in this stub)
C      LEVEL - error category.
C            =2 means this is an unconditionally fatal error.
C            =1 means this is a recoverable error.  (I.e., it is
C              non-fatal if XSETF has been appropriately called.)
C            =0 means this is a warning message only.
C            =-1 means this is a warning message which is to be
C              printed at most once, regardless of how many
C              times this call is executed.
C              (in this stub LEVEL=2 causes an error message to be
C                              printed followed by a stop,
C                        LEVEL<2 causes an error message to be
C                              printed followed by a return.)
C      NI    - number of integer values to be printed. (0 to 2)
C      I1    - first integer value.
C      I2    - second integer value.
C      NR     - number of real values to be printed. (0 to 2)
C      R1    - first real value.
C      R2    - second real value.
C
C    Examples
C       CALL XERRWV('SMOOTH -- NUM (=I1) WAS ZERO.',29,1,2,
C     1   1,NUM,0,0,0.,0.)
C       CALL XERRWV('QUADXY -- REQUESTED ERROR (R1) LESS THAN
MINIMUM (
C     1R2).,54,77,1,0,0,0,2,ERRREQ,ERRMIN)
C
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC
ERROR-
C              HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C              1982.
```

```
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  XERRWV
      CHARACTER*(*) MESSG

C***FIRST EXECUTABLE STATEMENT  XERRWV
      WRITE(*,*) MESSG
      IF(NI.EQ.2)THEN
        WRITE(*,*) I1,I2
      ELSEIF(NI.EQ.1) THEN
        WRITE(*,*) I1
      ENDIF
      IF(NR.EQ.2) THEN
        WRITE(*,*) R1,R2
      ELSEIF(NR.EQ.1) THEN
        WRITE(*,*) R1
      ENDIF
      IF(ABS(LEVEL).LT.2)RETURN
      STOP
      END



      SUBROUTINE SGEFS(A,LDA,N,V,ITASK,IND,WORK,IWORK,RCOND)
C***BEGIN PROLOGUE  SGEFS
C***DATE WRITTEN  800317  (YYMMDD)
C***REVISION DATE  870916  (YYMMDD)
C***CATEGORY NO.  D2A1
C***KEYWORDS  GENERAL SYSTEM OF LINEAR EQUATIONS,LINEAR
EQUATIONS
C***AUTHOR  VOORHEES, E., (LOS ALAMOS NATIONAL LABORATORY)
C***PURPOSE  SGEFS solves a GENERAL single precision real
C           NXN system of linear equations.
C***DESCRIPTION
C
C   From the book "Numerical Methods and Software"
C      by  D. Kahaner, C. Moler, S. Nash
C        Prentice Hall 1988
C
C   Subroutine SGEFS solves a general NxN system of single
C   precision linear equations using LINPACK subroutines SGECO
C   and SGESL.  That is, if A is an NxN real matrix and if X
C   and B are real N-vectors, then SGEFS solves the equation
C
C                     A*X=B.
C
C   The matrix A is first factored into upper and lower tri-
C   angular matrices U and L using partial pivoting.  These
C   factors and the pivoting information are used to find the
C   solution vector X.  An approximate condition number is
C   calculated to provide a rough estimate of the number of
C   digits of accuracy in the computed solution.
C
```

```
C   If the equation A*X=B is to be solved for more than one vector
C   B, the factoring of A does not need to be performed again and
C   the option to only solve (ITASK .EQ. 2) will be faster for
C   the succeeding solutions. In this case, the contents of A,
C   LDA, N and IWORK must not have been altered by the user follow-
C   ing factorization (ITASK=1). IND will not be changed by SGEFS
C   in this case. Other settings of ITASK are used to solve linear
C   systems involving the transpose of A.
C
C Argument Description ***
C
C  A     REAL(LDA,N)
C        on entry, the doubly subscripted array with dimension
C          (LDA,N) which contains the coefficient matrix.
C        on return, an upper triangular matrix U and the
C          multipliers necessary to construct a matrix L
C          so that A=L*U.
C  LDA   INTEGER
C        the leading dimension of the array A. LDA must be great-
C        er than or equal to N. (terminal error message IND=-1)
C  N     INTEGER
C        the order of the matrix A. The first N elements of
C        the array A are the elements of the first column of
C        the  matrix A.  N must be greater than or equal to 1.
C        (terminal error message IND=-2)
C  V     REAL(N)
C        on entry, the singly subscripted array(vector) of di-
C          mension N which contains the right hand side B of a
C          system of simultaneous linear equations A*X=B.
C        on return, V contains the solution vector, X .
C  ITASK  INTEGER
C        If ITASK=1, the matrix A is factored and then the
C          linear equation is solved.
C        If ITASK=2, the equation is solved using the existing
C          factored matrix A and IWORK.
C        If ITASK=3, the matrix is factored and A'x=b is solved
C        If ITASK=4, the transposed equation is solved using the
C          existing factored matrix A and IWORK.
C        If ITASK .LT. 1 or ITASK .GT. 4, then the terminal error
C          message IND=-3 is printed.
C  IND   INTEGER
C        GT. 0  IND is a rough estimate of the number of digits
C             of accuracy in the solution, X.
C        LT. 0  see error message corresponding to IND below.
C  WORK  REAL(N)
C        a singly subscripted array of dimension at least N.
C  IWORK  INTEGER(N)
C        a singly subscripted array of dimension at least N.
C  RCOND  REAL
C        estimate of 1.0/cond(A)
C
C Error Messages Printed ***
```

```
C
C   IND=-1  fatal   N is greater than LDA.
C   IND=-2  fatal   N is less than 1.
C   IND=-3  fatal   ITASK is less than 1 or greater than 4.
C   IND=-4  fatal   The matrix A is computationally singular.
C                   A solution has not been computed.
C   IND=-10 warning    The solution has no apparent significance.
C                   The solution may be inaccurate or the matrix
C                   A may be poorly scaled.
C
C***REFERENCES  SUBROUTINE SGEFS WAS DEVELOPED BY GROUP C-3,
LOS ALAMOS
C           SCIENTIFIC LABORATORY, LOS ALAMOS, NM 87545.
C           THE LINPACK SUBROUTINES USED BY SGEFS ARE DESCRIBED IN
C           DETAIL IN THE *LINPACK USERS GUIDE* PUBLISHED BY
C           THE SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS
C           (SIAM) DATED 1979.
C***ROUTINES CALLED  R1MACH,SGECO,SGESL,XERROR
C***END PROLOGUE  SGEFS
C
      INTEGER LDA,N,ITASK,IND,IWORK(*)
      REAL A(LDA,*),V(*),WORK(*),R1MACH
      REAL RCOND
      CHARACTER MSG*54

C***FIRST EXECUTABLE STATEMENT  SGEFS
      IF (LDA.LT.N)  GO TO 101
      IF (N.LE.0)  GO TO 102
      IF (ITASK.LT.1) GO TO 103
      IF (ITASK.GT.4) GO TO 103
      IF (ITASK.EQ.2 .OR. ITASK.GT.3) GO TO 20
C
C     FACTOR MATRIX A INTO LU
      CALL SGECO(A,LDA,N,IWORK,RCOND,WORK)
C
C     CHECK FOR COMPUTATIONALLY SINGULAR MATRIX
      IF (RCOND.EQ.0.0)  GO TO 104
C
C     COMPUTE IND (ESTIMATE OF NO. OF SIGNIFICANT DIGITS)
      IND=-INT(ALOG10(R1MACH(4)/RCOND))
C
C     CHECK FOR IND GREATER THAN ZERO
      IF (IND.GT.0)  GO TO 20
      IND=-10
      CALL XERROR( 'SGEFS ERROR (IND=-10) -- SOLUTION MAY HAVE NO
SIGNIF
     1ICANCE',58,-10,0)
C
C     SOLVE AFTER FACTORING
   20 JOB=0
      IF (ITASK.GT.2) JOB=1
      CALL SGESL(A,LDA,N,IWORK,V,JOB)
```

```
      RETURN
C
C     IF LDA.LT.N, IND=-1, FATAL XERROR MESSAGE
  101 IND=-1
      WRITE(MSG, '(
     * "SGEFS ERROR (IND=-1) -- LDA=", I5, " IS LESS THAN N=",
     *    I5      )'  ) LDA, N
      CALL XERROR(MSG(1:54), 54, -1, 0)
      RETURN
C
C     IF N.LT.1, IND=-2, FATAL XERROR MESSAGE
  102 IND=-2
      WRITE(MSG, '(
     * "SGEFS ERROR (IND=-2) -- N=", I5, " IS LESS THAN 1.") ')N
      CALL XERROR(MSG(1:47), 47, -2, 0)
      RETURN
C
C     IF ITASK.LT.1, IND=-3, FATAL XERROR MESSAGE
  103 IND=-3
      WRITE(MSG, '(
     * "SGEFS ERROR (IND=-3) -- ITASK=", I5, " IS LT 1 OR GT 4.")
     *           ') ITASK
      CALL XERROR(MSG(1:52), 52, -3, 0)
      RETURN
C
C     IF SINGULAR MATRIX, IND=-4, FATAL XERROR MESSAGE
  104 IND=-4
      CALL XERROR( 'SGEFS ERROR (IND=-4) -- SINGULAR MATRIX A - NO
SOLUT
     1ION',55,-4,0)
      RETURN
C
      END
      SUBROUTINE SGECO(A,LDA,N,IPVT,RCOND,Z)
C***BEGIN PROLOGUE  SGECO
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***ROUTINES CALLED  SASUM,SAXPY,SDOT,SGEFA,SSCAL
C***END PROLOGUE  SGECO
      INTEGER LDA,N,IPVT(*)
      REAL A(LDA,*),Z(*)
      REAL RCOND
C
      REAL SDOT,EK,T,WK,WKM
      REAL ANORM,S,SASUM,SM,YNORM
      INTEGER INFO,J,K,KB,KP1,L

C
C     COMPUTE 1-NORM OF A
```

```
C
C***FIRST EXECUTABLE STATEMENT  SGECO
      ANORM = 0.0E0
      DO 10 J = 1, N
        ANORM = AMAX1(ANORM,SASUM(N,A(1,J),1))
   10 CONTINUE
C
C     FACTOR
C
      CALL SGEFA(A,LDA,N,IPVT,INFO)
C
C     RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
C     ESTIMATE = NORM(Z)/NORM(Y) WHERE  A*Z = Y  AND  TRANS(A)*Y = E .
C     TRANS(A)  IS THE TRANSPOSE OF A .  THE COMPONENTS OF  E  ARE
C     CHOSEN TO CAUSE MAXIMUM LOCAL GROWTH IN THE ELEMENTS OF W
WHERE
C     TRANS(U)*W = E .  THE VECTORS ARE FREQUENTLY RESCALED TO
AVOID
C     OVERFLOW.
C
C     SOLVE TRANS(U)*W = E
C
      EK = 1.0E0
      DO 20 J = 1, N
        Z(J) = 0.0E0
   20 CONTINUE
      DO 100 K = 1, N
        IF (Z(K) .NE. 0.0E0) EK = SIGN(EK,-Z(K))
        IF (ABS(EK-Z(K)) .LE. ABS(A(K,K))) GO TO 30
          S = ABS(A(K,K))/ABS(EK-Z(K))
          CALL SSCAL(N,S,Z,1)
          EK = S*EK
   30   CONTINUE
        WK = EK - Z(K)
        WKM = -EK - Z(K)
        S = ABS(WK)
        SM = ABS(WKM)
        IF (A(K,K) .EQ. 0.0E0) GO TO 40
          WK = WK/A(K,K)
          WKM = WKM/A(K,K)
        GO TO 50
   40   CONTINUE
          WK = 1.0E0
          WKM = 1.0E0
   50   CONTINUE
        KP1 = K + 1
        IF (KP1 .GT. N) GO TO 90
          DO 60 J = KP1, N
            SM = SM + ABS(Z(J)+WKM*A(K,J))
            Z(J) = Z(J) + WK*A(K,J)
            S = S + ABS(Z(J))
   60       CONTINUE
```

```
            IF (S .GE. SM) GO TO 80
              T = WKM - WK
              WK = WKM
              DO 70 J = KP1, N
                Z(J) = Z(J) + T*A(K,J)
   70         CONTINUE
   80       CONTINUE
   90     CONTINUE
          Z(K) = WK
  100 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
C
C     SOLVE TRANS(L)*Y = W
C
      DO 120 KB = 1, N
        K = N + 1 - KB
        IF (K .LT. N) Z(K) = Z(K) + SDOT(N-K,A(K+1,K),1,Z(K+1),1)
        IF (ABS(Z(K)) .LE. 1.0E0) GO TO 110
          S = 1.0E0/ABS(Z(K))
          CALL SSCAL(N,S,Z,1)
  110   CONTINUE
        L = IPVT(K)
        T = Z(L)
        Z(L) = Z(K)
        Z(K) = T
  120 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
C
      YNORM = 1.0E0
C
C     SOLVE L*V = Y
C
      DO 140 K = 1, N
        L = IPVT(K)
        T = Z(L)
        Z(L) = Z(K)
        Z(K) = T
        IF (K .LT. N) CALL SAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
        IF (ABS(Z(K)) .LE. 1.0E0) GO TO 130
          S = 1.0E0/ABS(Z(K))
          CALL SSCAL(N,S,Z,1)
          YNORM = S*YNORM
  130   CONTINUE
  140 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
C     SOLVE  U*Z = V
C
```

```
      DO 160 KB = 1, N
        K = N + 1 - KB
        IF (ABS(Z(K)) .LE. ABS(A(K,K))) GO TO 150
          S = ABS(A(K,K))/ABS(Z(K))
          CALL SSCAL(N,S,Z,1)
          YNORM = S*YNORM
  150   CONTINUE
        IF (A(K,K) .NE. 0.0E0) Z(K) = Z(K)/A(K,K)
        IF (A(K,K) .EQ. 0.0E0) Z(K) = 1.0E0
        T = -Z(K)
        CALL SAXPY(K-1,T,A(1,K),1,Z(1),1)
  160 CONTINUE
C     MAKE ZNORM = 1.0
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
      IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
      RETURN
      END
      SUBROUTINE SGEFA(A,LDA,N,IPVT,INFO)
C***BEGIN PROLOGUE  SGEFA
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  SGEFA
      INTEGER LDA,N,IPVT(*),INFO
      REAL A(LDA,*)
C
      REAL T
      INTEGER ISAMAX,J,K,KP1,L,NM1

C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C***FIRST EXECUTABLE STATEMENT  SGEFA
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
        KP1 = K + 1
C
C       FIND L = PIVOT INDEX
C
        L = ISAMAX(N-K+1,A(K,K),1) + K - 1
        IPVT(K) = L
C
C       ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
```

```
      IF (A(L,K) .EQ. 0.0E0) GO TO 40
C
C        INTERCHANGE IF NECESSARY
C
         IF (L .EQ. K) GO TO 10
           T = A(L,K)
           A(L,K) = A(K,K)
           A(K,K) = T
   10    CONTINUE
C
C        COMPUTE MULTIPLIERS
C
         T = -1.0E0/A(K,K)
         CALL SSCAL(N-K,T,A(K+1,K),1)
C
C        ROW ELIMINATION WITH COLUMN INDEXING
C
         DO 30 J = KP1, N
           T = A(L,J)
           IF (L .EQ. K) GO TO 20
             A(L,J) = A(K,J)
             A(K,J) = T
   20        CONTINUE
             CALL SAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30      CONTINUE
       GO TO 50
   40   CONTINUE
        INFO = K
   50   CONTINUE
   60 CONTINUE
   70 CONTINUE
     IPVT(N) = N
     IF (A(N,N) .EQ. 0.0E0) INFO = N
     RETURN
     END
     SUBROUTINE SGESL(A,LDA,N,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGESL
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C             Prentice Hall 1988
C***END PROLOGUE  SGESL
     INTEGER LDA,N,IPVT(*),JOB
     REAL A(LDA,*),B(*)
C
     REAL SDOT,T
     INTEGER K,KB,L,NM1

C***FIRST EXECUTABLE STATEMENT  SGESL
     NM1 = N - 1
     IF (JOB .NE. 0) GO TO 50
```

```
C
C     JOB = 0 , SOLVE  A * X = B
C     FIRST SOLVE  L*Y = B
C
      IF (NM1 .LT. 1) GO TO 30
      DO 20 K = 1, NM1
        L = IPVT(K)
        T = B(L)
        IF (L .EQ. K) GO TO 10
          B(L) = B(K)
          B(K) = T
   10   CONTINUE
        CALL SAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
   20 CONTINUE
   30 CONTINUE
C
C     NOW SOLVE  U*X = Y
C
      DO 40 KB = 1, N
        K = N + 1 - KB
        B(K) = B(K)/A(K,K)
        T = -B(K)
        CALL SAXPY(K-1,T,A(1,K),1,B(1),1)
   40 CONTINUE
    GO TO 100
   50 CONTINUE
C
C     JOB = NONZERO, SOLVE  TRANS(A) * X = B
C     FIRST SOLVE  TRANS(U)*Y = B
C
      DO 60 K = 1, N
        T = SDOT(K-1,A(1,K),1,B(1),1)
        B(K) = (B(K) - T)/A(K,K)
   60 CONTINUE
C
C     NOW SOLVE TRANS(L)*X = Y
C
      IF (NM1 .LT. 1) GO TO 90
      DO 80 KB = 1, NM1
        K = N - KB
        B(K) = B(K) + SDOT(N-K,A(K+1,K),1,B(K+1),1)
        L = IPVT(K)
        IF (L .EQ. K) GO TO 70
          T = B(L)
          B(L) = B(K)
          B(K) = T
   70   CONTINUE
   80 CONTINUE
   90 CONTINUE
  100 CONTINUE
    RETURN
    END
```

```
C       THIS WAS MODIFIED TO WORK ON THE MAC.  WHY SHOULD YOU
CARE?
C       PROBABLY BECAUSE IT IS WORTHLESS.

      REAL FUNCTION R1MACH(I)
C***BEGIN PROLOGUE  R1MACH
C***DATE WRITTEN  790101  (YYMMDD)
C***REVISION DATE  831014  (YYMMDD)
C***CATEGORY NO.  R1
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  FOX, P. A., (BELL LABS)
C         HALL, A. D., (BELL LABS)
C         SCHRYER, N. L., (BELL LABS)
C***PURPOSE  Returns single precision machine dependent constants
C***DESCRIPTION
C    From the book, "Numerical Methods and Software" by
C              D. Kahaner, C. Moler, S. Nash
C             Prentice Hall, 1988
C
C
C    R1MACH can be used to obtain machine-dependent parameters
C    for the local machine environment.  It is a function
C    subroutine with one (input) argument, and can be called
C    as follows, for example
C
C        A = R1MACH(I)
C
C    where I=1,...,5.  The (output) value of A above is
C    determined by the (input) value of I.  The results for
C    various values of I are discussed below.
C
C  Single-Precision Machine Constants
C  R1MACH(1) = B**(EMIN-1), the smallest positive magnitude.
C  R1MACH(2) = B**EMAX*(1 - B**(-T)), the largest magnitude.
C  R1MACH(3) = B**(-T), the smallest relative spacing.
C  R1MACH(4) = B**(1-T), the largest relative spacing.
C  R1MACH(5) = LOG10(B)
C***REFERENCES  FOX, P.A., HALL, A.D., SCHRYER, N.L, *FRAMEWORK
FOR
C             A PORTABLE LIBRARY*, ACM TRANSACTIONS ON MATHE-
C             MATICAL SOFTWARE, VOL. 4, NO. 2, JUNE 1978,
C             PP. 177-188.
C***ROUTINES CALLED  XERROR
C***END PROLOGUE  R1MACH
C
    INTEGER SMALL(2)
    INTEGER LARGE(2)
    INTEGER RIGHT(2)
    INTEGER DIVER(2)
    INTEGER LOG10(2)
```

```
C
      REAL RMACH(5)
C
      EQUIVALENCE (RMACH(1),SMALL(1))
      EQUIVALENCE (RMACH(2),LARGE(1))
      EQUIVALENCE (RMACH(3),RIGHT(1))
      EQUIVALENCE (RMACH(4),DIVER(1))
      EQUIVALENCE (RMACH(5),LOG10(1))
C
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 170 SERIES (FTN5).
C
C     DATA RMACH(1) / O"0001400000000000000" /
C     DATA RMACH(2) / O"3776777777777777777" /
C     DATA RMACH(3) / O"1640400000000000000" /
C     DATA RMACH(4) / O"1641400000000000000" /
C     DATA RMACH(5) / O"1716464202324117572O" /
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 200 SERIES
C
C     DATA RMACH(1) / X'9000400000000000' /
C     DATA RMACH(2) / X'6FFF7FFFFFFFFFFF' /
C     DATA RMACH(3) / X'FFA3400000000000' /
C     DATA RMACH(4) / X'FFA4400000000000' /
C     DATA RMACH(5) / X'FFD04D104D427DE8' /
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C     DATA RMACH(1) / 00564000000000000000B /
C     DATA RMACH(2) / 37767777777777777776B /
C     DATA RMACH(3) / 16414000000000000000B /
C     DATA RMACH(4) / 16424000000000000000B /
C     DATA RMACH(5) / 17164642023241175720B /
C
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA RMACH(1) / 20003400000000000000B /
C     DATA RMACH(2) / 57767777777777777776B /
C     DATA RMACH(3) / 37722400000000000000B /
C     DATA RMACH(4) / 37723400000000000000B /
C     DATA RMACH(5) / 37777464202324117572OB /
C
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86  AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA RMACH(1) / Z00100000 /
C     DATA RMACH(2) / Z7FFFFFFF /
C     DATA RMACH(3) / Z3B100000 /
C     DATA RMACH(4) / Z3C100000 /
C     DATA RMACH(5) / Z41134413 /
```

```
C
C    MACHINE CONSTANTS FOR THE IBM PC FAMILY (D. KAHANER NBS)
C
C     DATA RMACH/1.18E-38,3.40E+38,0.595E-07,1.19E-07,0.30102999566/

C       MACHINE CONSTANTS FOR THE 68882

        DATA RMACH/1.2E-38,3.4E+38,0.595E-07,1.19E-07,0.30102999566/
C
C    MACHINE CONSTANTS FOR THE PDP-10 (KA OR KI PROCESSOR).
C
C    DATA RMACH(1) / "000400000000 /
C    DATA RMACH(2) / "377777777777 /
C    DATA RMACH(3) / "146400000000 /
C    DATA RMACH(4) / "147400000000 /
C    DATA RMACH(5) / "177464202324 /
C
C
C    MACHINE CONSTANTS FOR THE SUN-3 (INCLUDES THOSE WITH 68881
CHIP,
C      OR WITH FPA BOARD. ALSO INCLUDES SUN-2 WITH SKY BOARD. MAY
ALSO
C      WORK WITH SOFTWARE FLOATING POINT ON EITHER SYSTEM.)
C
C    DATA SMALL(1) / X'00800000' /
C    DATA LARGE(1) / X'7F7FFFFF' /
C    DATA RIGHT(1) / X'33800000' /
C    DATA DIVER(1) / X'34000000' /
C    DATA LOG10(1) / X'3E9A209B' /
C
C
C    MACHINE CONSTANTS FOR THE VAX 11/780
C   (EXPRESSED IN INTEGER AND HEXADECIMAL)
C *** THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS***
C
C    DATA SMALL(1) /      128 /
C    DATA LARGE(1) /   -32769 /
C    DATA RIGHT(1) /    13440 /
C    DATA DIVER(1) /    13568 /
C    DATA LOG10(1) / 547045274 /
C
C ***THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX
SYSTEMS***
C    DATA SMALL(1) / Z00000080 /
C    DATA LARGE(1) / ZFFFF7FFF /
C    DATA RIGHT(1) / Z00003480 /
C    DATA DIVER(1) / Z00003500 /
C    DATA LOG10(1) / Z209B3F9A /
C
C
C***FIRST EXECUTABLE STATEMENT  R1MACH
     IF (I .LT. 1  .OR.  I .GT. 5)
```

```
      1   CALL XERROR ( 'R1MACH -- I OUT OF BOUNDS',25,1,2)
C
      R1MACH = RMACH(I)
      RETURN
C
      END
      DOUBLE PRECISION FUNCTION D1MACH(I)
C***BEGIN PROLOGUE  D1MACH
C***DATE WRITTEN  750101  (YYMMDD)
C***REVISION DATE  831014  (YYMMDD)
C***CATEGORY NO.  R1
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  FOX, P. A., (BELL LABS)
C          HALL, A. D., (BELL LABS)
C          SCHRYER, N. L., (BELL LABS)
C***PURPOSE  Returns double precision machine dependent constants
C***DESCRIPTION
C    From the book, "Numerical Methods and Software" by
C              D. Kahaner, C. Moler, S. Nash
C              Prentice Hall, 1988
C
C
C    D1MACH can be used to obtain machine-dependent parameters
C    for the local machine environment.  It is a function
C    subprogram with one (input) argument, and can be called
C    as follows, for example
C
C         D = D1MACH(I)
C
C    where I=1,...,5.  The (output) value of D above is
C    determined by the (input) value of I.  The results for
C    various values of I are discussed below.
C
C Double-precision machine constants
C D1MACH( 1) = B**(EMIN-1), the smallest positive magnitude.
C D1MACH( 2) = B**EMAX*(1 - B**(-T)), the largest magnitude.
C D1MACH( 3) = B**(-T), the smallest relative spacing.
C D1MACH( 4) = B**(1-T), the largest relative spacing.
C D1MACH( 5) = LOG10(B)
C***REFERENCES  FOX P.A., HALL A.D., SCHRYER N.L.,*FRAMEWORK FOR
A
C              PORTABLE LIBRARY*, ACM TRANSACTIONS ON MATHEMATICAL
C               SOFTWARE, VOL. 4, NO. 2, JUNE 1978, PP. 177-188.
C***ROUTINES CALLED  XERROR
C***END PROLOGUE  D1MACH
C
      INTEGER SMALL(4)
      INTEGER LARGE(4)
      INTEGER RIGHT(4)
      INTEGER DIVER(4)
      INTEGER LOG10(4)
C
```

```
      DOUBLE PRECISION DMACH(5)
C
      EQUIVALENCE (DMACH(1),SMALL(1))
      EQUIVALENCE (DMACH(2),LARGE(1))
      EQUIVALENCE (DMACH(3),RIGHT(1))
      EQUIVALENCE (DMACH(4),DIVER(1))
      EQUIVALENCE (DMACH(5),LOG10(1))
C
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 170 SERIES (FTN5).
C
C     DATA SMALL(1) / O"0060400000000000000" /
C     DATA SMALL(2) / O"0000000000000000000" /
C
C     DATA LARGE(1) / O"3776777777777777777" /
C     DATA LARGE(2) / O"3716777777777777777" /
C
C     DATA RIGHT(1) / O"1560400000000000000" /
C     DATA RIGHT(2) / O"1500000000000000000" /
C
C     DATA DIVER(1) / O"1561400000000000000" /
C     DATA DIVER(2) / O"1501000000000000000" /
C
C     DATA LOG10(1) / O"1716464202324117571" /
C     DATA LOG10(2) / O"1636757142174254654" /
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 200 SERIES
C
C     DATA SMALL(1) / X'9000400000000000' /
C     DATA SMALL(2) / X'8FD1000000000000' /
C
C     DATA LARGE(1) / X'6FFF7FFFFFFFFFFF' /
C     DATA LARGE(2) / X'6FD07FFFFFFFFFFF' /
C
C     DATA RIGHT(1) / X'FF74400000000000' /
C     DATA RIGHT(2) / X'FF45000000000000' /
C
C     DATA DIVER(1) / X'FF75400000000000' /
C     DATA DIVER(2) / X'FF46000000000000' /
C
C     DATA LOG10(1) / X'FFD04D104D427DE7' /
C     DATA LOG10(2) / X'FFA17DE623E2566A' /
C
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C     DATA SMALL(1) / 00564000000000000000B /
C     DATA SMALL(2) / 00000000000000000000B /
C
C     DATA LARGE(1) / 37757777777777777777B /
C     DATA LARGE(2) / 37157777777777777777B /
C
```

```
C     DATA RIGHT(1) / 1562400000000000000B /
C     DATA RIGHT(2) / 000000000000000000000B /
C
C     DATA DIVER(1) / 1563400000000000000B /
C     DATA DIVER(2) / 000000000000000000000B /
C
C     DATA LOG10(1) / 17164642023241175717B /
C     DATA LOG10(2) / 16367571421742254654B /
C
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA SMALL(1) / 20135400000000000000B /
C     DATA SMALL(2) / 000000000000000000000B /
C
C     DATA LARGE(1) / 57776777777777777777B /
C     DATA LARGE(2) / 000007777777777777774B /
C
C     DATA RIGHT(1) / 37643400000000000000B /
C     DATA RIGHT(2) / 000000000000000000000B /
C
C     DATA DIVER(1) / 37644400000000000000B /
C     DATA DIVER(2) / 000000000000000000000B /
C
C     DATA LOG10(1) / 37777464202324117571 7B /
C     DATA LOG10(2) / 00000757142174225465 4B /
C
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA SMALL(1),SMALL(2) / Z00100000, Z00000000 /
C     DATA LARGE(1),LARGE(2) / Z7FFFFFFF, ZFFFFFFFF /
C     DATA RIGHT(1),RIGHT(2) / Z33100000, Z00000000 /
C     DATA DIVER(1),DIVER(2) / Z34100000, Z00000000 /
C     DATA LOG10(1),LOG10(2) / Z41134413, Z509F79FF /
C
C     MACHINE CONSTATNS FOR THE IBM PC FAMILY (D. KAHANER NBS)
C
C      DATA DMACH/2.23D-308,1.79D+308,1.11D-16,2.22D-16,
C     * 0.301029995663981195D0/

C        DATA FOR MAC IIx  PROBABLY NONSENSE

         DATA DMACH/2.3D-308,1.70D+308,1.11D-16,2.22D-16,
     * 0.301029995663981195D0/
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).
C
C     DATA SMALL(1),SMALL(2) / "033400000000, "000000000000 /
C     DATA LARGE(1),LARGE(2) / "377777777777, "344777777777 /
C     DATA RIGHT(1),RIGHT(2) / "113400000000, "000000000000 /
```

```
C    DATA DIVER(1),DIVER(2) / "114400000000, "000000000000 /
C    DATA LOG10(1),LOG10(2) / "177464202324, "144117571776 /
C
C    MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).
C
C    DATA SMALL(1),SMALL(2) / "000400000000, "000000000000 /
C    DATA LARGE(1),LARGE(2) / "377777777777, "377777777777 /
C    DATA RIGHT(1),RIGHT(2) / "103400000000, "000000000000 /
C    DATA DIVER(1),DIVER(2) / "104400000000, "000000000000 /
C    DATA LOG10(1),LOG10(2) / "177464202324, "476747767461 /
C
C
C    MACHINE CONSTANTS FOR THE SUN-3 (INCLUDES THOSE WITH 68881
CHIP,
C     OR WITH FPA BOARD. ALSO INCLUDES SUN-2 WITH SKY BOARD. MAY
ALSO
C     WORK WITH SOFTWARE FLOATING POINT ON EITHER SYSTEM.)
C
C     DATA SMALL(1),SMALL(2) / X'00100000', X'00000000' /
C     DATA LARGE(1),LARGE(2) / X'7FEFFFFF', X'FFFFFFFF' /
C     DATA RIGHT(1),RIGHT(2) / X'3CA00000', X'00000000' /
C     DATA DIVER(1),DIVER(2) / X'3CB00000', X'00000000' /
C     DATA LOG10(1),LOG10(2) / X'3FD34413', X'509F79FF' /
C
C
C    MACHINE CONSTANTS FOR VAX 11/780
C    (EXPRESSED IN INTEGER AND HEXADECIMAL)
C   *** THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS***
C
C    DATA SMALL(1), SMALL(2) /      128,         0 /
C    DATA LARGE(1), LARGE(2) /   -32769,        -1 /
C    DATA RIGHT(1), RIGHT(2) /     9344,         0 /
C    DATA DIVER(1), DIVER(2) /     9472,         0 /
C    DATA LOG10(1), LOG10(2) /  546979738,  -805796613 /
C
C   ***THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX
SYSYEMS***
C    DATA SMALL(1), SMALL(2) / Z00000080, Z00000000 /
C    DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C    DATA RIGHT(1), RIGHT(2) / Z00002480, Z00000000 /
C    DATA DIVER(1), DIVER(2) / Z00002500, Z00000000 /
C    DATA LOG10(1), LOG10(2) / Z209A3F9A, ZCFF884FB /
C
C  MACHINE CONSTANTS FOR VAX 11/780 (G-FLOATING)
C    (EXPRESSED IN INTEGER AND HEXADECIMAL)
C   *** THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS***
C
C    DATA SMALL(1), SMALL(2) /       16,         0 /
C    DATA LARGE(1), LARGE(2) /   -32769,        -1 /
C    DATA RIGHT(1), RIGHT(2) /    15552,         0 /
C    DATA DIVER(1), DIVER(2) /    15568,         0 /
C    DATA LOG10(1), LOG10(2) /  1142112243, 2046775455 /
```

```
C
C   ***THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX
SYSYEMS***
C    DATA SMALL(1), SMALL(2) / Z00000010, Z00000000 /
C    DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C    DATA RIGHT(1), RIGHT(2) / Z00003CC0, Z00000000 /
C    DATA DIVER(1), DIVER(2) / Z00003CD0, Z00000000 /
C    DATA LOG10(1), LOG10(2) / Z44133FF3, Z79FF509F /
C
C
C***FIRST EXECUTABLE STATEMENT  D1MACH
     IF (I .LT. 1  .OR.  I .GT. 5)
    1   CALL XERROR( 'D1MACH -- I OUT OF BOUNDS',25,1,2)
C
     D1MACH = DMACH(I)
     RETURN
C
     END
     INTEGER FUNCTION I1MACH(I)
C***BEGIN PROLOGUE  I1MACH
C***DATE WRITTEN   750101   (YYMMDD)
C***REVISION DATE  840405   (YYMMDD)
C***CATEGORY NO.  R1
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  FOX, P. A., (BELL LABS)
C          HALL, A. D., (BELL LABS)
C          SCHRYER, N. L., (BELL LABS)
C***PURPOSE  Returns integer machine dependent constants
C***DESCRIPTION
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  These machine constant routines must be activated for
C  a particular environment.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C    I1MACH can be used to obtain machine-dependent parameters
C    for the local machine environment.  It is a function
C    subroutine with one (input) argument, and can be called
C    as follows, for example
C
C         K = I1MACH(I)
C
C    where I=1,...,16.  The (output) value of K above is
C    determined by the (input) value of I.  The results for
C    various values of I are discussed below.
C
C  I/O unit numbers.
C    I1MACH( 1) = the standard input unit.
C    I1MACH( 2) = the standard output unit.
C    I1MACH( 3) = the standard punch unit.
C    I1MACH( 4) = the standard error message unit.
C
```

```
C  Words.
C   I1MACH( 5) = the number of bits per integer storage unit.
C   I1MACH( 6) = the number of characters per integer storage unit.
C
C  Integers.
C   assume integers are represented in the S-digit, base-A form
C
C              sign ( X(S-1)*A**(S-1) + ... + X(1)*A + X(0) )
C
C               where 0 .LE. X(I) .LT. A for I=0,...,S-1.
C   I1MACH( 7) = A, the base.
C   I1MACH( 8) = S, the number of base-A digits.
C   I1MACH( 9) = A**S - 1, the largest magnitude.
C
C  Floating-Point Numbers.
C   Assume floating-point numbers are represented in the T-digit,
C   base-B form
C              sign (B**E)*( (X(1)/B) + ... + (X(T)/B**T) )
C
C               where 0 .LE. X(I) .LT. B for I=1,...,T,
C              0 .LT. X(1), and EMIN .LE. E .LE. EMAX.
C   I1MACH(10) = B, the base.
C
C  Single-Precision
C   I1MACH(11) = T, the number of base-B digits.
C   I1MACH(12) = EMIN, the smallest exponent E.
C   I1MACH(13) = EMAX, the largest exponent E.
C
C  Double-Precision
C   I1MACH(14) = T, the number of base-B digits.
C   I1MACH(15) = EMIN, the smallest exponent E.
C   I1MACH(16) = EMAX, the largest exponent E.
C
C  To alter this function for a particular environment,
C  the desired set of DATA statements should be activated by
C  removing the C from column 1.  Also, the values of
C  I1MACH(1) - I1MACH(4) should be checked for consistency
C  with the local operating system.
C***REFERENCES  FOX P.A., HALL A.D., SCHRYER N.L.,*FRAMEWORK FOR
A
C              PORTABLE LIBRARY*, ACM TRANSACTIONS ON MATHEMATICAL
C               SOFTWARE, VOL. 4, NO. 2, JUNE 1978, PP. 177-188.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  I1MACH
C
    INTEGER IMACH(16),OUTPUT

    EQUIVALENCE (IMACH(4),OUTPUT)
C
C
C    MACHINE CONSTANTS FOR THE CDC CYBER 170 SERIES (FTN5).
C
```

```
C     DATA IMACH( 1) /    5 /
C     DATA IMACH( 2) /    6 /
C     DATA IMACH( 3) /    7 /
C     DATA IMACH( 4) /    6 /
C     DATA IMACH( 5) /   60 /
C     DATA IMACH( 6) /   10 /
C     DATA IMACH( 7) /    2 /
C     DATA IMACH( 8) /   48 /
C     DATA IMACH( 9) / O"00007777777777777777" /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   48 /
C     DATA IMACH(12) / -974 /
C     DATA IMACH(13) / 1070 /
C     DATA IMACH(14) /   96 /
C     DATA IMACH(15) / -927 /
C     DATA IMACH(16) / 1070 /
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 200 SERIES
C
C     DATA IMACH( 1) /     5 /
C     DATA IMACH( 2) /     6 /
C     DATA IMACH( 3) /     7 /
C     DATA IMACH( 4) /     6 /
C     DATA IMACH( 5) /    64 /
C     DATA IMACH( 6) /     8 /
C     DATA IMACH( 7) /     2 /
C     DATA IMACH( 8) /    47 /
C      DATA IMACH( 9) / X'00007FFFFFFFFFFF' /
C     DATA IMACH(10) /     2 /
C     DATA IMACH(11) /    47 /
C     DATA IMACH(12) / -28625 /
C     DATA IMACH(13) /  28718 /
C     DATA IMACH(14) /    94 /
C     DATA IMACH(15) / -28625 /
C     DATA IMACH(16) /  28718 /
C
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C     DATA IMACH( 1) /    5 /
C     DATA IMACH( 2) /    6 /
C     DATA IMACH( 3) /    7 /
C     DATA IMACH( 4) /6LOUTPUT/
C     DATA IMACH( 5) /   60 /
C     DATA IMACH( 6) /   10 /
C     DATA IMACH( 7) /    2 /
C     DATA IMACH( 8) /   48 /
C     DATA IMACH( 9) / 00007777777777777777B /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   47 /
C     DATA IMACH(12) / -929 /
C     DATA IMACH(13) / 1070 /
```

```
C     DATA IMACH(14) /  94 /
C     DATA IMACH(15) / -929 /
C     DATA IMACH(16) / 1069 /
C
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA IMACH( 1) /   100 /
C     DATA IMACH( 2) /   101 /
C     DATA IMACH( 3) /   102 /
C     DATA IMACH( 4) /   101 /
C     DATA IMACH( 5) /    64 /
C     DATA IMACH( 6) /     8 /
C     DATA IMACH( 7) /     2 /
C     DATA IMACH( 8) /    63 /
C     DATA IMACH( 9) / 777777777777777777777B /
C     DATA IMACH(10) /     2 /
C     DATA IMACH(11) /    47 /
C     DATA IMACH(12) / -8189 /
C     DATA IMACH(13) /  8190 /
C     DATA IMACH(14) /    94 /
C     DATA IMACH(15) / -8099 /
C     DATA IMACH(16) /  8190 /
C
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA IMACH( 1) /   5 /
C     DATA IMACH( 2) /   6 /
C     DATA IMACH( 3) /   7 /
C     DATA IMACH( 4) /   6 /
C     DATA IMACH( 5) /  32 /
C     DATA IMACH( 6) /   4 /
C     DATA IMACH( 7) /  16 /
C     DATA IMACH( 8) /  31 /
C     DATA IMACH( 9) / Z7FFFFFFF /
C     DATA IMACH(10) /  16 /
C     DATA IMACH(11) /   6 /
C     DATA IMACH(12) / -64 /
C     DATA IMACH(13) /  63 /
C     DATA IMACH(14) /  14 /
C     DATA IMACH(15) / -64 /
C     DATA IMACH(16) /  63 /
C
C     MACHINE CONSTANTS FOR THE IBM PC FAMILY (D. KAHANER NBS)
C
      DATA IMACH/5,6,0,6,32,4,2,31,2147483647,2,24,
     * -125,127,53,-1021,1023/
C           NOTE! I1MACH(3) IS NOT WELL DEFINED AND IS SET TO ZERO.
C
C
```

```
C    MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).
C
C    DATA IMACH( 1) /   5 /
C    DATA IMACH( 2) /   6 /
C    DATA IMACH( 3) /   5 /
C    DATA IMACH( 4) /   6 /
C    DATA IMACH( 5) /  36 /
C    DATA IMACH( 6) /   5 /
C    DATA IMACH( 7) /   2 /
C    DATA IMACH( 8) /  35 /
C    DATA IMACH( 9) / "377777777777 /
C    DATA IMACH(10) /   2 /
C    DATA IMACH(11) /  27 /
C    DATA IMACH(12) / -128 /
C    DATA IMACH(13) /  127 /
C    DATA IMACH(14) /  54 /
C    DATA IMACH(15) / -101 /
C    DATA IMACH(16) /  127 /
C
C    MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).
C
C    DATA IMACH( 1) /   5 /
C    DATA IMACH( 2) /   6 /
C    DATA IMACH( 3) /   5 /
C    DATA IMACH( 4) /   6 /
C    DATA IMACH( 5) /  36 /
C    DATA IMACH( 6) /   5 /
C    DATA IMACH( 7) /   2 /
C    DATA IMACH( 8) /  35 /
C    DATA IMACH( 9) / "377777777777 /
C    DATA IMACH(10) /   2 /
C    DATA IMACH(11) /  27 /
C    DATA IMACH(12) / -128 /
C    DATA IMACH(13) /  127 /
C    DATA IMACH(14) /  62 /
C    DATA IMACH(15) / -128 /
C    DATA IMACH(16) /  127 /
C
C
C    MACHINE CONSTANTS FOR THE SUN-3 (INCLUDES THOSE WITH 68881
CHIP,
C     OR WITH FPA BOARD. ALSO INCLUDES SUN-2 WITH SKY BOARD. MAY
ALSO
C     WORK WITH SOFTWARE FLOATING POINT ON EITHER SYSTEM.)
C
C    DATA IMACH( 1) /   5 /
C    DATA IMACH( 2) /   6 /
C    DATA IMACH( 3) /   6 /
C    DATA IMACH( 4) /   0 /
C    DATA IMACH( 5) /  32 /
C    DATA IMACH( 6) /   4 /
C    DATA IMACH( 7) /   2 /
```

```
C     DATA IMACH( 8) /   31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   24 /
C     DATA IMACH(12) / -125 /
C     DATA IMACH(13) /  128 /
C     DATA IMACH(14) /   53 /
C     DATA IMACH(15) / -1021 /
C     DATA IMACH(16) /  1024 /
C
C
C     MACHINE CONSTANTS FOR THE VAX 11/780
C
C     DATA IMACH(1) /    5 /
C     DATA IMACH(2) /    6 /
C     DATA IMACH(3) /    5 /
C     DATA IMACH(4) /    6 /
C     DATA IMACH(5) /   32 /
C     DATA IMACH(6) /    4 /
C     DATA IMACH(7) /    2 /
C     DATA IMACH(8) /   31 /
C     DATA IMACH(9) /2147483647 /
C     DATA IMACH(10)/    2 /
C     DATA IMACH(11)/   24 /
C     DATA IMACH(12)/ -127 /
C     DATA IMACH(13)/  127 /
C     DATA IMACH(14)/   56 /
C     DATA IMACH(15)/ -127 /
C     DATA IMACH(16)/  127 /
C
C***FIRST EXECUTABLE STATEMENT  I1MACH
      IF (I .LT. 1  .OR.  I .GT. 16)
     1   CALL XERROR ( 'I1MACH -- I OUT OF BOUNDS',25,1,2)
C
      I1MACH=IMACH(I)
      RETURN
C
      END

      INTEGER FUNCTION ISAMAX(N,SX,INCX)
C***BEGIN PROLOGUE  ISAMAX
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  ISAMAX
C
      REAL SX(*),SMAX,XMAG
C***FIRST EXECUTABLE STATEMENT  ISAMAX
      ISAMAX = 0
      IF(N.LE.0) RETURN
```

```
      ISAMAX = 1
      IF(N.LE.1)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      SMAX = ABS(SX(1))
      NS = N*INCX
      II = 1
         DO 10 I=1,NS,INCX
         XMAG = ABS(SX(I))
         IF(XMAG.LE.SMAX) GO TO 5
         ISAMAX = II
         SMAX = XMAG
    5    II = II + 1
   10    CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
   20 SMAX = ABS(SX(1))
      DO 30 I = 2,N
         XMAG = ABS(SX(I))
         IF(XMAG.LE.SMAX) GO TO 30
         ISAMAX = I
         SMAX = XMAG
   30 CONTINUE
      RETURN
      END
      REAL FUNCTION SASUM(N,SX,INCX)
C***BEGIN PROLOGUE  SASUM
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  SASUM
C
      REAL SX(*)
C***FIRST EXECUTABLE STATEMENT  SASUM
      SASUM = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      NS = N*INCX
         DO 10 I=1,NS,INCX
         SASUM = SASUM + ABS(SX(I))
   10    CONTINUE
      RETURN
C
```

```
C      CODE FOR INCREMENTS EQUAL TO 1.
C
C
C      CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 6.
C
   20 M = MOD(N,6)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SASUM = SASUM + ABS(SX(I))
   30 CONTINUE
      IF( N .LT. 6 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,6
        SASUM = SASUM + ABS(SX(I)) + ABS(SX(I + 1)) + ABS(SX(I + 2))
     1  + ABS(SX(I + 3)) + ABS(SX(I + 4)) + ABS(SX(I + 5))
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SAXPY
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  SAXPY
C
      REAL SX(*),SY(*),SA

C***FIRST EXECUTABLE STATEMENT  SAXPY
      IF(N.LE.0.OR.SA.EQ.0.E0) RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C      CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SY(IY) + SA*SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 4.
C
```

```
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SY(I) + SA*SX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        SY(I) = SY(I) + SA*SX(I)
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
   50 CONTINUE
      RETURN
C
C       CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
   60 CONTINUE
      NS = N*INCX
         DO 70 I=1,NS,INCX
         SY(I) = SA*SX(I) + SY(I)
   70    CONTINUE
      RETURN
      END
      SUBROUTINE SCOPY(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SCOPY
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  SCOPY
C
      REAL SX(*),SY(*)

C***FIRST EXECUTABLE STATEMENT  SCOPY
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C       CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
```

```
C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 7.
C
   20 M = MOD(N,7)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SX(I)
   30 CONTINUE
      IF( N .LT. 7 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,7
        SY(I) = SX(I)
        SY(I + 1) = SX(I + 1)
        SY(I + 2) = SX(I + 2)
        SY(I + 3) = SX(I + 3)
        SY(I + 4) = SX(I + 4)
        SY(I + 5) = SX(I + 5)
        SY(I + 6) = SX(I + 6)
   50 CONTINUE
      RETURN
C
C      CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
   60 CONTINUE
      NS = N*INCX
         DO 70 I=1,NS,INCX
         SY(I) = SX(I)
   70    CONTINUE
      RETURN
      END
      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SDOT
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C              Prentice Hall 1988
C***END PROLOGUE  SDOT
C
      REAL SX(*),SY(*)
C***FIRST EXECUTABLE STATEMENT  SDOT
      SDOT = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1)5,20,60
    5 CONTINUE
C
C      CODE FOR UNEQUAL INCREMENTS OR NONPOSITIVE INCREMENTS.
C
      IX = 1
```

```
      IY = 1
       IF(INCX.LT.0)IX = (-N+1)*INCX + 1
       IF(INCY.LT.0)IY = (-N+1)*INCY + 1
       DO 10 I = 1,N
        SDOT = SDOT + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
       RETURN
C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
       SDOT = SDOT + SX(I)*SY(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
       SDOT = SDOT + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) +
     1  SX(I + 2)*SY(I + 2) + SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4)
   50 CONTINUE
      RETURN
C
C      CODE FOR POSITIVE EQUAL INCREMENTS .NE.1.
C
   60 CONTINUE
      NS=N*INCX
      DO 70 I=1,NS,INCX
       SDOT = SDOT + SX(I)*SY(I)
   70  CONTINUE
      RETURN
      END
      REAL FUNCTION SNRM2(N,SX,INCX)
C***BEGIN PROLOGUE  SNRM2
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C             Prentice Hall 1988
C***END PROLOGUE  SNRM2
      INTEGER         NEXT
      REAL   SX(*),  CUTLO, CUTHI, HITEST, SUM, XMAX, ZERO, ONE

      DATA   ZERO, ONE /0.0E0, 1.0E0/
C
      DATA CUTLO, CUTHI / 4.441E-16,  1.304E19 /
C***FIRST EXECUTABLE STATEMENT  SNRM2
```

```
      IF(N .GT. 0) GO TO 10
        SNRM2  = ZERO
        GO TO 300
C
   10 ASSIGN 30 TO NEXT
      SUM = ZERO
      NN = N * INCX

C                                   BEGIN MAIN LOOP
      I = 1
   20   GO TO NEXT,(30, 50, 70, 110)
   30 IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 50 TO NEXT
      XMAX = ZERO
C
C                   PHASE 1.  SUM IS ZERO
C
   50 IF( SX(I) .EQ. ZERO) GO TO 200
      IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
C
C                         PREPARE FOR PHASE 2.
      ASSIGN 70 TO NEXT
      GO TO 105
C
C                         PREPARE FOR PHASE 4.
C
  100 I = J
      ASSIGN 110 TO NEXT

      SUM = (SUM / SX(I)) / SX(I)
  105 XMAX = ABS(SX(I))
      GO TO 115
C
C              PHASE 2.  SUM IS SMALL.
C                   SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   70 IF( ABS(SX(I)) .GT. CUTLO ) GO TO 75
C
C               COMMON CODE FOR PHASES 2 AND 4.
C               IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  110 IF( ABS(SX(I)) .LE. XMAX ) GO TO 115
        SUM = ONE + SUM * (XMAX / SX(I))**2
        XMAX = ABS(SX(I))
        GO TO 200
C
  115 SUM = SUM + (SX(I)/XMAX)**2
      GO TO 200
C
C
C              PREPARE FOR PHASE 3.
```

```
C
  75 SUM = (SUM * XMAX) * XMAX
C
C
C    FOR REAL OR D.P. SET HITEST = CUTHI/N
C    FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
  85 HITEST = CUTHI/FLOAT( N )
C
C                PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
     DO 95 J =I,NN,INCX
     IF(ABS(SX(J)) .GE. HITEST) GO TO 100
  95   SUM = SUM + SX(J)**2
     SNRM2 = SQRT( SUM )
     GO TO 300
C
 200 CONTINUE
     I = I + INCX
     IF ( I .LE. NN ) GO TO 20
C
C          END OF MAIN LOOP.
C
C          COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
     SNRM2 = XMAX * SQRT(SUM)

 300 CONTINUE

     RETURN
     END
      SUBROUTINE SSCAL(N,SA,SX,INCX)
C***BEGIN PROLOGUE  SSCAL
C    THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C    FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C    From the book "Numerical Methods and Software"
C         by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C***END PROLOGUE  SSCAL
C
     REAL SA,SX(*)

C***FIRST EXECUTABLE STATEMENT  SSCAL
     IF(N.LE.0)RETURN
     IF(INCX.EQ.1)GOTO 20
C
C       CODE FOR INCREMENTS NOT EQUAL TO 1.
C
     NS = N*INCX
        DO 10 I = 1,NS,INCX
        SX(I) = SA*SX(I)
  10    CONTINUE
```

```
      RETURN
C
C       CODE FOR INCREMENTS EQUAL TO 1.
C
C
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SX(I) = SA*SX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SX(I) = SA*SX(I)
        SX(I + 1) = SA*SX(I + 1)
        SX(I + 2) = SA*SX(I + 2)
        SX(I + 3) = SA*SX(I + 3)
        SX(I + 4) = SA*SX(I + 4)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SSWAP(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SSWAP
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  SSWAP
C
      REAL SX(*),SY(*),STEMP1,STEMP2,STEMP3

C***FIRST EXECUTABLE STATEMENT  SSWAP
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C       CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP1 = SX(IX)
        SX(IX) = SY(IY)
        SY(IY) = STEMP1
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
```

```
      RETURN
C
C     CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C     CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 3.
C
   20 M = MOD(N,3)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        STEMP1 = SX(I)
        SX(I) = SY(I)
        SY(I) = STEMP1
   30 CONTINUE
      IF( N .LT. 3 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,3
        STEMP1 = SX(I)
        STEMP2 = SX(I+1)
        STEMP3 = SX(I+2)
        SX(I) = SY(I)
        SX(I+1) = SY(I+1)
        SX(I+2) = SY(I+2)
        SY(I) = STEMP1
        SY(I+1) = STEMP2
        SY(I+2) = STEMP3
   50 CONTINUE
      RETURN
   60 CONTINUE
C
C     CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
      NS = N*INCX
        DO 70 I=1,NS,INCX
        STEMP1 = SX(I)
        SX(I) = SY(I)
        SY(I) = STEMP1
   70   CONTINUE
      RETURN
      END
```