SAND REPORT
SAND2002-1448
Unlimited Release
Printed August, 2002

On the Accuracy of Operator-Splitting Methods for Problems with Multiple Time Scales

Louis A. Romero

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

25p.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2002-1448
Unlimited Release
Printed August 2002

On the Accuracy of Operator Splitting Methods for Problems with Multiple Time Scales

Louis A. Romero
Computational Mathematics and Algorithms Department
Sandia National Laboratories [1]
P.O. Box 5800
Albuquerque, NM 87185-1111

Abstract

In this paper we investigate the accuracy of operator splitting schemes when applied to operators that can be written as the sum of two operators each with its own characteristic time scale. When these time scales differ significantly it is shown that operator-splitting techniques can give much better accuracy than fully implicit methods. In order to get this benefit out of operator splitting one must integrate the fast operator more accurately than the slow operator. Both an analysis and numerical experiments are given to justify this statement. The analysis is used to construct a second order operator splitting scheme that works for arbitrary nonlinear equations.

[1]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

1 Introduction

There are several different reasons for using operator-splitting. The technique was first used in order to simplify the linear algebra involved in integrating multi-dimensional partial differential equations implicitly [1]. By splitting the multi-dimensional operator into a sum of one dimensional operators, and successively integrating these operators each time step, one obtains the stability of a fully implicit method while only doing the linear algebra on one dimensional systems.

More recently it has become standard to use operator-splitting on problems involving chemical reactions and diffusion (or some other transport mechanism). In these problems the "fast " time scale for the chemistry, t_f, is typically much shorter than the "slow" time scale for the transport, t_s. In this paper we will refer to the ratio of these two scales by

$$\epsilon = \frac{t_f}{t_s}.$$

When ϵ is small, the matrices that arise by integrating the equations fully implicitly are often very poorly conditioned, causing problems for standard matrix iterative solvers. Operator splitting overcomes this problem since if one splits the chemistry from the diffusion (or other transport mechanism) the resulting matrices are better conditioned. Also, as pointed out in [2] , by using operator-splitting one can use ones favorite technique for each individual operator. This can simplify the building and modifying of a large code.

It should also be pointed out that when one ignores the transport mechanisms, there is no coupling between the temperature and species concentrations at different grid cells. This means that the equations governing the chemistry can be integrated much quicker than the fully coupled equations involving both chemistry and transport.

There appears to be confusion in the literature as to how operator splitting affects the accuracy of the solutions. The situation is especially unclear when ϵ is small. In this case the operator with the fast time scale is frequently integrated with much smaller time steps than the operator with the slow time scale. Some authors imply that if one could efficiently invert the linear system for the two operators together , then there would be no advantage to doing operator splitting. However, other authors imply (incorrectly) that one can obtain meaningful results by integrating each operator on its own time scale. If the fast operator is easy to integrate compared to the full system of equations, this would clearly be beneficial. Unfortunatley, as pointed out in [2] , in order to obtain meaningful results one must integrate the slow operator using a time step small compared to t_f.

Given this restriction on the slow time step, it might appear that there is no advantage to integrating the fast operator more accurately than the slow operator. In this case one would be as well off integrating the equations fully implicitly (assuming one could invert the fully coupled equations as quickly as

the split equations). We will show that there is in fact an advantage gained in accuracy by using operator splitting. This results from the fact that if we were to use a first order method to integrate the equations fully implicily out to a time on the order of t_s, then we would need to take a time step that is small compared to ϵt_f. We will show that we can obtain accurate solutions using operator splitting when we integrate the fast operator using a time step that is small compared to ϵt_f, but using a time step for the slow operator that need only be small compared to t_f. As ϵ approaches zero, one can obtain solutions much more efficiently using operator splitting than by using a fully implicit technique. It should be emphasized that this advantage of operator splitting remains even if one could invert the fully coupled equations as efficiently as the split operators.

Most of the theoretical analysis of operator-splitting is based on the assumption that the same time step is taken for both operators. However, for chemically reacting flow problems one often takes a much shorter time step for the chemistry than for the diffusion. In this paper we present an analysis based on the assumption that the fast operator is integrated exactly. This is clearly not the case in practice, but the fast operator is often integrated with sufficiently high accuracy compared to the slow operator that it is a reasonable assumption. The analysis shows that when the fast time scale is integrated exactly one can achieve better accuracy using operator-splitting than a fully implicit technique. Here we are comparing the implicit method when it uses the same time step as is used for the slow operator in the operator-splitting scheme. Note that these benefits of operator-splitting go away if the two time scales do not differ greatly, and if one does not integrate the fast time scale with more accuracy than the slow time scale.

The analysis in this paper is for arbitrary linear systems that can be split into a slow and a fast operator. We make a change of variable using the fundamental matrix solution to the fast operator. Substituting this new variable into the differential equation we end up with a modified differential equation. We show that applying a backwards Euler scheme to this modified equation is identical to using operator-splitting on the original equation, but with the fast operator-integrated exactly. This way of deriving operator splitting allows us to see why one obtains better accuracy using operator-splitting than integrating the equations fully implicitly. It also suggests how to obtain higher order splitting schemes. By applying the trapezoidal rule to the modified equation one comes up with a second order splitting scheme that is similar to the method presented in [3]. The method presented here requires less computation since it involves only one implicit solve each time step.

It is shown both theoretically and by numerical experiments that the second order splitting method offers significant improvements over a fully implicit second order method as $\epsilon \to 0$. However, it should be pointed out that in order to get significant benefits out of this technique one must integrate the equations out to a time that is on the order of $\frac{t_s}{\epsilon}$. This shows that the benefits on accuracy

for a second order splitting method are not as significant as they are for a first order method.

The technique of consistant operator splitting has been proposed [4] in order to improve the efficiency of the standard operator splitting technique as one approaches a steady state. It is interesting to see whether this technique has the same improvement in accuracy over the fully implict method that the standard operator splitting technique has as $\epsilon \to 0$. Numerical experiments show that the method of consistant operator splitting also does well as $\epsilon \to 0$. The author does not have a convincing argument to show that this behavior should be expected.

We present a second order technique that results from a very minor modification of the first order splitting technique. Both the theory and numerical experiments confirm that this technique is second order, however, it is found that this method does not have any advantages over the fully implicit method as $\epsilon \to 0$.

We now present a brief summary of the paper. In section 2 we derive the modified equation and show how the standard form for operator splitting can be derived by applying a backwards Euler method to the modified equation. In section 3) we show how higher order operator splitting methods can be derived for linear systems by applying higher order Adams methods to the modified equation. We specifically apply the trapezoidal rule to the modified equation to derive a second order operator splitting method, and explain how it can easily be generalized to nonlinear problems. In section 4) we summarize the methods that are tested numerically, in section 5) we present the results for several linear test problems, and in section 6) we present results obtained by applying the different techniques to the nonlinear test problem known as the Brusselator [6]. In section 7) we summarize our results.

2 The Modified Equation

Suppose we have an equation of the form

$$\underline{\dot{x}} = (A_0(t) + \epsilon A_1(t))\,\underline{x} + \underline{f}_0(t) + \epsilon \underline{f}_1(t). \tag{1}$$

Here $\underline{x}$,$\underline{f}_0$ and $\underline{f}_1$ are n dimensional vectors, and A_0 and A_1 are $n \times n$ matrices. The equations of interest frequently arise from a spatial discretization of a partial differential equation. We will not concern ourselves with the spatial discretization errors, but assume that we are interested in finding accurate solutions to eqn. (1) The analysis that we are about to present is easier to follow if one ignores the inhomogeneous terms and one assumes that the matrices are not dependent on time. We will present the general analysis, but suggest that the reader keep in mind the simpler analysis that arises when we ignore the inhomogeneous terms and the time dependence of the matrices.

Let $Q(t, t_0)$ be the fundamental matrix solution to the differential equation

when $\epsilon = 0$. That is, $Q(t, t_0)$ satisfies

$$\frac{d}{dt}Q(t, t_0) = A_0(t)Q(t, t_0), \tag{2a}$$

$$Q(t_0, t_0) = I. \tag{2b}$$

In what follows we will make use of the semi-group property of the fundamental matrix

$$Q(t, t_0) = Q(t, \alpha)Q(\alpha, t_0). \tag{3}$$

We now make a change of variable from $\underline{x}(t)$ to the variable $\underline{z}(t)$ that is defined through the equation

$$\underline{x}(t) = Q(t, t_0)\left(\underline{z}(t) + \int_{t_0}^{t} Q^{-1}(\tau, t_0)\underline{f}_0(\tau)d\tau\right) \tag{4}$$

This substitution is motivated by the fact that when $\epsilon = 0$ $x(t)$ is a solution to the equations if we put $\underline{z}(t) = constant$.

If $\underline{x}(t)$ is to satisfy eqn. (1), then $\underline{z}(t)$ must satisfy

$$\dot{\underline{z}} = \epsilon Q^{-1}(t, t_0)A_1(t)Q(t, t_0)\left(\underline{z}(t) + \int_{t_0}^{t} Q^{-1}(\tau, t_0)\underline{f}_0(\tau)d\tau\right) + \epsilon Q^{-1}(t, t_0)\underline{f}_1 \tag{5}$$

Assuming that we can integrate the fast equations exactly, we can determine the solution $\underline{x}(t)$ by integrating eqn. (5) to determine $\underline{z}(t)$ and then using (4) to find $\underline{x}(t)$. If we use the backwards Euler method to integrate eqn. (5) we find

$$\underline{z}^{n+1} = \underline{z}^n + \Delta t\epsilon Q^{-1}(t_{n+1}, t_0)\left(A_1(t_{n+1})\underline{x}^{n+1} + \underline{f}_1(t_{n+1})\right) \tag{6a}$$

where

$$t_{n+1} = t_n + \Delta t, \tag{6b}$$

and,

$$\underline{x}^{n+1} = Q(t_{n+1}, t_0)\left(\underline{z}^{n+1} + \int_{t_0}^{t_{n+1}} Q^{-1}(\tau, t_0)\underline{f}_0(\tau)d\tau\right) \tag{6c}$$

If we multiply both sides of eqn. (6a) by $Q(t_{n+1}, t_0)$ we find that

$$Q(t_{n+1}, t_0)\left(\underline{z}^{n+1} - \underline{z}^n\right) = \epsilon\Delta t A_1(t_{n+1})\underline{x}^{n+1} + \epsilon\underline{f}_1(t_{n+1})$$

A direct application of eqn. (4) shows that in terms of the original variable $\underline{x}$ we have

$$\underline{x}^{n+1} = \underline{x}^{n+1/2} + \epsilon\Delta t A_1(t_{n+1})\underline{x}^{n+1} + \epsilon\underline{f}_1(t_{n+1}) \tag{7a}$$

where

$$\underline{x}^{n+\frac{1}{2}} = Q(t_{n+1}, t_0)\left(Q^{-1}(t_n, t_0)\underline{x}^n + \int_{t_n}^{t_{n+1}} Q^{-1}(\tau, t_0)\underline{f}_0(\tau)d\tau\right).$$

Now using eqn. (3) we get

$$\underline{x}^{n+1/2} = Q(t_{n+1}, t_n)\left(\underline{x}^n + \int_{t_n}^{t_{n+1}} Q^{-1}(\tau, t_n)\underline{f}_0(\tau)d\tau\right) \tag{7b}$$

Note that the vector $\underline{x}^{n+1/2}$ is obtained by integrating the equation

$$\dot{\underline{x}} = A_0(t)\underline{x} + \underline{f}_0(t)$$

from t_n to t_{n+1} with an initial condition of $\underline{x}(t_n) = \underline{x}^n$.

Note that in order to arrive at the answer $\underline{x}^{n+1}$ using the modified equation we carry out exactly the procedure that is used in operator-splitting. We first integrate the fast equations from t_n to t_{n+1} with an initial value of $\underline{x}^n$. This gives us our intermediate solution $\underline{x}^{n+\frac{1}{2}}$. Here we have assumed that we have integrated these equations exactly. We now integrate the slow equations out one time step using an initial value of $\underline{x}^{n+\frac{1}{2}}$.

Note that if we integrate the equations (1) using a fully implicit backwards Euler method out to a time $T = \frac{T_1}{\epsilon}$, then in order to have an absolute error of less than δ we must choose the time step small enough so that

$$\Delta t \frac{T_1}{\epsilon} < K_1\delta$$

Here K_1 is a constant that is nearly independent of Δt and ϵ.

If we use the backwards Euler method to integrate the equations (5) out to the same time T, then in order to achieve an absolute error of less than δ we must choose Δt so that

$$\Delta t\epsilon \frac{T_1}{\epsilon} < K_2\delta$$

When integrating the original equation (1) out to a time that is $O(\frac{1}{\epsilon})$ it is necessary to have $\Delta t\epsilon$ be small . When integrating the equation (5) out to the same time one can achieve the same absolute error with the less stringent requirement of making Δt be small. Once one has obtained the solution $\underline{z}$ from integrating the modified equation, one can then obtain the solution $\underline{x}$ using eqn. (4). Since we have shown that using operator-splitting (integrating the fast operator exactly) is equivalent to applying the backwards Euler method to (5) it is clear that one obtains better accuracy using operator-splitting than by integrating the equations fully implicitly. The bigger the difference in the time scales , the more efficient it becomes to use operator splitting. It should be pointed out that in order to accurately integrate the modified equations out to a time that is on the order of the slow time scale , it is still necessary to take time steps that are small compared to the fast time scale.

3 Higher Order Splitting Methods

In this section we obtain higher order operator splitting methods by applying higher order Adams methods to eqn. (5). These methods are valid for any linear problem, but it is not clear how to generalize them to an arbitrary nonlinear problem. This difficulty arises since the schemes distinguish between the homogeneous and non-homogeneous terms in the equations. For a nonlinear problem it is not clear how to make this distinction. However, for the particular case of the trapezoidal rule applied to eqn. (5) we can write the resulting method in a way that can be generalized to nonlinear problems.

When integrating the equation

$$\dot{\underline{x}} = \underline{g}(\underline{x}, t)$$

the general p-step Adams scheme uses

$$\underline{x}^{n+1} = \underline{x}^n + \Delta t \sum_{k=0}^{p} \alpha_k \underline{g}(\underline{x}^{n+1-k}, t_{n+1-k})$$

where,

$$t_{n+1-k} = t_n + \Delta t(1-k),$$

and α_k are suitably chosen constants.

When we apply an Adams method to eqn. (5) we get

$$\underline{z}^{n+1} = \underline{z}^n + \Delta t \epsilon \sum_{k=0}^{p} \alpha_k \underline{R}_k \tag{8a}$$

where

$$\underline{R}_k = Q^{-1}(t_{n+1-k}, t_0)\left(A_1(t_{n+1-k})Q(t_{n+1-k}, t_0)\left(\underline{z}^{n+1-k} + \int_{t_0}^{t_{n+1-k}} Q^{-1}(\tau, t_0)\underline{f}_0(\tau)d\tau\right) + \underline{f}_1(t_{n+1-k})\right) \tag{8b}$$

If we now write this in terms of $\underline{x}$ we get

$$\underline{x}^{n+1} = \underline{x}^{n+1/2} + \epsilon \Delta t \sum_{k=0}^{p} \alpha_k \Gamma_k \left(A_1(t_{n+1-k})\underline{x}^{n+1-k} + \underline{f}_1(t_{n+1-k})\right) \tag{9a}$$

$$\underline{x}^{n+1/2} = Q(t_{n+1}, t_n)\left(\underline{x}^n + \int_{t_n}^{t_{n+1}} Q^{-1}(\tau, t_n)\underline{f}_0(\tau)d\tau\right) \tag{9b}$$

$$\Gamma_k = Q(t_{n+1}, t_0)Q^{-1}(t_{n+1-k}, t_0) = Q(t_n, t_{n+1-k}) \tag{9c}$$

We have used the semi-group property in eqn. (3) in order to arrive at this result.

Note that the vector $\underline{x}^{n+1/2}$ is obtained by integrating the equation

$$\dot{\underline{x}} = A_0(t)\underline{x} + \underline{f}_0(t)$$

from t_n to t_{n+1} with an initial condition of $\underline{x}(t_n) = \underline{x}^n$.

Also, note that multiplying a vector $\underline{v}$ by Γ_k is equivalent to integrating the equation

$$\dot{\underline{x}} = A_0(t)\underline{x}$$

from t_{n+1-k} to t_{n+1} with an initial condition of $\underline{x}(t_{n+1-k}) = \underline{v}$.

It is clear that this scheme is a generalization of operator splitting.

In general it is not clear how to generalize these methods so they can be used on nonlinear problems. In particular, one has to know how to integrate the homogeneous part of the fast equations all by themselves. For a nonlinear problem it is not clear what is meant by this.

We now apply the trapezoidal rule to eqn. (5) and show that in this case the method can be straightforwardly generalized to nonlinear problems. We argue that the method also keeps its second order accuracy for nonlinear problems.

When we apply the trapezoidal rule to eqn. (5) we find that

$$\underline{x}^{n+1} = \underline{x}^{n+\frac{1}{2}} + \frac{1}{2}\epsilon\Delta t\left(Q(t_{n+1}, t_n)\left(A_1\underline{x}^n + \underline{f}_1(t_n)\right) + A_1\underline{x}^{n+1} + \underline{f}_1(t_{n+1})\right)$$

$$\underline{x}^{n+\frac{1}{2}} = Q(t_{n+1}, t_n)\left(\underline{x}_n + \int_{t_n}^{t_{n+1}} Q^{-1}(\tau, t_n)\underline{f}_0(\tau)d\tau\right)$$

This can be rewritten as

$$\underline{x}^{n+1} = \underline{x}^\star + \frac{1}{2}\epsilon\Delta t\left(A_1\underline{x}^{n+1} + \underline{f}_1(t_{n+1})\right) \tag{10a}$$

where

$$\underline{x}^\star = Q(t_{n+1}, t_n)\left(\hat{\underline{x}} + \int_{t_n}^{t_{n+1}} Q^{-1}(\tau, t_n)\underline{f}_0(\tau)d\tau\right) \tag{10b}$$

$$\hat{\underline{x}} = \underline{x}^n + \frac{1}{2}\Delta t\epsilon\left(A_1\underline{x}^n + \underline{f}_1(t_n)\right) \tag{10c}$$

This scheme can be described as follows.

- Starting with the vector $\underline{x}^n$ take half of a first order explicit time step using the slow operator. This gives us a vector $\hat{\underline{x}}$.
- Using $\hat{\underline{x}}$ as a starting value, integrate the fast equations from t_n to t_{n+1}. This gives us the vector $\underline{x}^\star$.
- Using the vector $\underline{x}^\star$ as a starting value take half of a first order implicit time step using the slow operator.

Note that the regular trapezoidal rule can be considered as taking half a time step explicitly followed by half a time step implicitly. This scheme does the same thing except it integrates the fast equations in between these two steps.

The statement of the above algorithm holds whether the equations are linear or nonlinear. The second order accuracy should persist for nonlinear problems. This follows since if we have a nonlinear system

$$\frac{d}{dt}\underline{x} = \underline{F}(\underline{x})$$

then we can linearize this system over the interval (t_n, t_{n+1}).

$$\frac{d}{dt}\underline{x} = \underline{F}(\underline{x}^n) + \underline{F}_{\underline{x}}(\underline{x}_n)\underline{x} + O((\Delta t)^2)$$

We can get a second order scheme by doing a second order integration of these linearized equations on each interval.

It should be noted that the method described in this section is similar to the method described by Strang in [3]. In the method described by Strang one takes half of a time step using a fully implicit second order method on the slow operator, then takes a full time step on the fast operator, and once more takes a fully implicit half time step on the slow operator. In the paper by Strang he does not discuss fast and slow operators, and he assumes that the same time step is taken for both operators. This algorithm generalizes when we integrate the chemistry very accurately to the following.

- Integrate the slow equations for half a time step using a second order implicit scheme such as the trapezoidal rule. This takes our initial vector $\underline{x}^n$ into a new vector $\hat{\underline{x}}$.
- Now integrate the fast equations over a whole time step using $\hat{\underline{x}}$ as the initial value. This gives us a vector $\underline{x}^\star$.
- Using $\underline{x}^\star$ as an inital value, integrate the slow equations over half a time step using a second order fully implicit method. This gives us the final answer

Both the method described by Strang and the method introduced at the beginning of this section can be written as

$$\underline{x}^{n+1} = L^1_{\Delta t/2} M_{\Delta t} L^2_{\Delta t/2} \underline{x}^n$$

Here $L^1_{\Delta t/2}$ and $L^2_{\Delta t/2}$ are numerical approximations that integrate the slow equations forward half a time step; and $M_{\Delta t}$ is a numerical approximation that integrates the fast equations forward a full time step. In Strangs method the operator and $L^2_{\Delta t/2}$ both come from applying a second order implicit method to the slow equations. In the method described in this section the operator $L^2_{\Delta t/2}$

comes from using the first order forward Euler method to integrate the slow equations over half a time step. The operator $L^1_{\Delta t/2}$ comes from using the first order backwards Euler formula to integrate the slow equations over half a time step.

Note that if we use a second order implicit method to integrate the equations (1) out to a time $\frac{T_1}{\epsilon}$ with an accuracy of δ, then we must choose our time step so that

$$(\Delta t)^2 < K_1 \delta \epsilon.$$

If we integrate eqn. (5) out to the same time and require the same accuracy we need only require that

$$(\Delta t)^2 < K_2 \delta$$

.

We see that the second order operator splitting method allows us to take a larger time step for the slow operator than when we use the fully implicit method. In order to get as good answers out of the implicit method one must chosse a times step that is about $\sqrt{\epsilon}$ as small as when using the operator splitting method. We see that the difference between the second order splitting method and the second order implicit method is not as pronounced as for the first order methods unless we integrate out to a time that is on the order of $\frac{t_f}{\epsilon^2}$.

4 The Methods to be Tested

We now describe 7 different codes that we will test in order to illustrate the accuracy of operator-splitting methods. When testing these codes we will compare the results when the time step Δt is constant and the same for all of the codes. In the codes that use a form of operator-splitting the fast equations are integrated using a library ordinary differential equation solver with the absolute error tolerance set to $1.d-9$. When comparing fully implicit methods to operator-splitting methods we take the same time step on the fully implicit method as we do on the slow time for the operator-splitting method.

In this paper we are only concerned with the accuracy of operator-splitting methods, not in the speed ups in the linear algebra. All of the codes described in this section are tested only on one dimensional partial differential equations. The linear systems are solved using a banded matrix solver from LINPACK. Newtons method is used to solve the nonlinear equations arising in the fully implicit methods.

We will now state the time stepping algorithms for the various schemes. When describing these schemes we will assume we are solving the system of equations

$$\frac{d}{dt}\underline{x} = \underline{F}_0(\underline{x}) + \underline{F}_1(\underline{x}) \tag{11}$$

In the description of the following algorithms $\underline{x}^n$ and $\underline{x}^{n+1}$ are the solutions at the times t_n and t_{n+1}; and $\Delta t = t_{n+1} - t_n$.

In most of the operator splitting methods we make use of the function $\underline{z}(t, \hat{t}, \underline{x})$ which is given by

$$\frac{d}{dt}\underline{z} = \underline{F}_0(\underline{z})$$
$$\underline{z}(\hat{t}, \hat{t}, \underline{x}) = \underline{x}$$

We now describe the algorithms we have tested. There are two fully implicit codes.

- IMP1 This is a fully implicit code that uses the backwards Euler method to integrate the system of ordinary differential equations.

$$\underline{x}^{n+1} - \underline{x}^n = \Delta t \left(\underline{F}_0(\underline{x}^{n+1}) + \underline{F}_1(\underline{x}^{n+1})\right)$$

- IMP2 This is a fully implicit scheme that uses the trapezoidal rule.

$$\underline{x}^{n+1} - \underline{x}^n = \frac{1}{2}\Delta t \left(\underline{F}_0(\underline{x}^{n+1}) + \underline{F}_1(\underline{x}^{n+1}) + \underline{F}_0(\underline{x}^n) + \underline{F}_1(\underline{x}^n)\right)$$

We also test the following three operator splitting methods.

- SPLIT1 This is an operator splitting scheme that uses the backwards Euler method to integrate the modified equations in eqn. (5). As already pointed out, this scheme is equivalent to applying a first order operator-splitting method to the equations (1) with the fast equations integrated very accurately.

$$\underline{x}^{n+1} - \underline{x}^{n+1/2} = \Delta t \underline{F}_1(\underline{x}^{n+1})$$

where,

$$\underline{x}^{n+1/2} = \underline{z}(t^{n+1}, t^n, \underline{x}^n)$$

- SPLIT2 This is an operator scheme that uses the trapezoidal rule to integrate the modified equations (5). This is equivalent to applying the second order operator splitting scheme described in the last section. This scheme has second order accuracy.

- STRANG This code uses the second order operator splitting scheme described in [3] and in the last section.

Finally, we test a first and a second order consistant operator splitting method.

- CONSIST1 This code uses the consistant operator splitting method described in [4] and below.

$$\underline{x}^{n+1} - \underline{x}^{n+1/2} = \Delta t \left(\underline{F}_1(\underline{x}^{n+1}) - \underline{F}_1(\underline{x}^n) \right)$$

$$\underline{x}^{n+1/2} = \underline{\phi}(t_{n+1})$$

where

$$\frac{d}{dt}\underline{\phi} = \underline{F}_0(\underline{\phi}) + \underline{F}_1(\underline{x}^n)$$

$$\underline{\phi}(t_n) = \underline{x}^n$$

- CONSIST2 This code uses a second order consistant operator splitting method described below.

$$\underline{x}^{n+1} - \underline{x}^{n+1/2} = \frac{1}{2}\Delta t \left(\underline{F}_1(\underline{x}^{n+1}) - \underline{F}_1(\underline{x}^n) \right)$$

$$\underline{x}^{n+1/2} = \underline{\phi}(t_{n+1})$$

where ϕ is the same function used for the first order consistant splitting method.

It is a straightforward exercise to confirm that the scheme used in CONSIST2 is second order for the general linear system described in eqn. (1). The second order consistant splitting method requires no more work than the first order, but the author is not aware of it being described previously in the literature.

5 Some Linear Test Problems

In this section we test the codes on several linear reaction diffusion equations. The equations are not motivated by any physical system, but are merely used to illustrate the how the different methods behave as the reaction and diffusion time scales grow apart.

All of the equations in this section arise from a second order spatial discretization of the equation

$$\frac{\partial T}{\partial t} = \epsilon \frac{\partial^2 T}{\partial z^2} - T\frac{d}{dt}f(t,z) + g(t,z) \tag{12a}$$

$$T(0,t) = T(1,t) = 0. \tag{12b}$$

$$T(z,0) = sin(\pi z) \tag{12c}$$

These equations can be written as

$$\frac{\partial T}{\partial t} = L_0 T + \epsilon L_1 T + f_0(z,t)$$

where

$$L_0(t)T(z,t) = -T(z,t)\frac{d}{dt}f(t,z)$$

$$L_1 T(z,t) = \frac{\partial^2 T}{\partial z^2}$$

and

$$f_0(z,t) = g(t,z).$$

We obtain a system of ordinary differential equations by dividing up the interval $[0,1]$ in N equally spaced intervals and defining the solution $T_i(t)$ at each grid point. We end up with a system of equations of the form in eqn. (1) where A_0 is a diagonal matrix representing the operator L_0, A_1 is a tridiagonal matrix approximating the operator L_1, and $\underline{f}_0(t)$ is a vector obtained by discretizing the function $f_0(z,t)$ at the grid points.

Note that we have included a spatial dependence in the operator L_0 in order to avoid having the operators L_0 and L_1 commute. If the operators commuted it would be a poor test problem for operator-splitting methods since in that case we rigorously get the same answers whether we integrate the equations together or one at a time.

In each of the test problems we use a time step of

$$\Delta t = \frac{1}{LSTEP \times 10} \tag{13}$$

for the slow operator. This means that when applying one of the operator splitting methods we integrate the fast equations for an interval of Δt, and then we correct our solution using the slow operator. When using a fully implicit method we use a time step of Δt. We integrate all of the equations from $t = 0$ to $t = \frac{2}{\epsilon}$. This means that as we decrease ϵ we are always integrating the equations out to a time that is the same when scaled according to the slow time scale t_s.

In order to determine the error in our answers we first integrate the equations using the second order operator splitting code SPLIT2 with a very small time step (a large value of LSTEP). The fact that $LSTEP$ was taken large enough is confirmed by looking at how the solutions with smaller values of $LSTEP$ compare to this solution. We compare different solutions by looking at the error in the flux at the left hand boundary. Let $T_1(t)$ be the numerical approximation to the solution at the first grid point, that is

$$T_1(t) \approx T(\frac{1}{2}\Delta z, t)$$

where

$$\Delta z = \frac{1}{N}.$$

The flux at the boundary is approximated by

$$\psi(t) = \frac{2T_1(t)}{\Delta z}$$

We will denote the flux at the left hand boundary obtained from the very accurate time integration of the discretized partial differential equation as $\psi_\infty(t)$. For a finite value of $LSTEP$ we calcuate the normalized error as

$$error = \frac{\sum_{k=0}^{NSTEPS} \mid \psi(t_k) - \psi_\infty(t_k) \mid}{\sum_{k=0}^{NSTEPS} \mid \psi_\infty(t_k) \mid}$$

where $NSTEPS$ is the number of time steps that were taken in order to reach $\frac{2}{\epsilon}$, and t_k is the time at the kth time step.

In all of the studies in this section we will use $LSTEP = 1$ when we are comparing the codes for different values of ϵ. We will use $\epsilon = \frac{1}{40}$ when we change $LSTEP$ in order to check the rate of convergence of the different schemes.

5.1 Problem 1

We set

$$\frac{d}{dt} f(t,z) = zcos(t) - \epsilon\pi^2$$

$$g(t) = 0.$$

Note that when we just consider the term involving $\frac{d}{dt}f$ in the equations we get an osillatory solution that has an exponential growth. We have chosen this growth so that it is just balanced by the exponential decay from the lowest order mode in the diffusion equation. When we integrate the full system we expect to get solutions that are oscillatory and that have relatively little growth or decay.

Table 1 shows the errors for the different codes at different values of ϵ. All of these results are for $LSTEP = 1$. For the first order methods the code CONS gives the smallest error at all values of ϵ. When ϵ is small, the fully implicit code IMP gives a smaller error than the operator splitting code SPL1, but as ϵ decreases, the error in the implicit code increases severely, while the error in the code SPL1 actually decreases a bit. For small values of ϵ the code SPL1 gives much smaller errors than the code IMP.

This is a clear example of what we would like to illustrate. When we fix the time step and decrease the ratio of the time scales ϵ , the error in the implicit method increases when we integrate the equations out to a time that is on the

order of the slow time scale. When we do the same for the splitting method SPL1, this does not happen. Our numerical experiments show that this does not happen for the consistant splitting code CONS.

This effect is not quite as pronounced when we compare the second order implicit code IMP2 to the second order splitting codes SPL2 and STR. The error in the code IMP2 is smaller than the erros from SPL2 and STR when ϵ is large, but as ϵ decreases, the error of the code IMP2 increases slightly , but the error in the codes SPL2 and STR decrease fairly significantly . Note that the second order consistant method CONS2 behaves much like the fully implicit method.

Table 1:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	9.41 D-2	6.89 D-1	1.80 D-2	2.91 D-4	8.50 D-3	2.03 D-3	1.45 D-3
20	1.71 D-1	3.02 D-1	1.46 D-2	2.92 D-4	2.04 D-3	5.27 D-4	1.57 D-3
40	3.78 D-1	1.38 D-1	1.83 D-2	2.76 D-4	3.23 D-4	2.14 D-4	1.79 D-3
80	9.93 D-1	6.15 D-2	1.49 D-2	3.59 D-4	6.34 D-5	1.15 D-4	1.76 D-3
160	3.48 D+0	2.89 D-2	6.70 D-3	4.01 D-4	4.16 D-5	6.13 D-5	1.71 D-3
320	2.72 D+1	1.39 D-2	1.55 D-3	4.13 D-4	2.73 D-5	3.31 D-5	1.70 D-3

Table 2 shows how each of the error in each of the methods decreases as we decrease the stepsize. All of these errors were for $\frac{1}{\epsilon} = 40$. .It is clear that the first order methods IMP,SPL1, and CONS are all showing first order convergence. Similarly the methods IMP2, SPL2 ,STR, and CONS2 are all showing second order convergence.

Table 2:

lSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	3.78 D-1	1.38 D-1	1.83 D-2	2.76 D-4	3.23 D-4	2.14 D-4	1.79 D-3
2	1.70 D-1	6.66 D-2	1.09 D-2	6.89 D-5	8.07 D-5	5.37 D-6	4.53 D-4
4	8.09 D-2	3.28 D-2	5.93 D-3	1.71 D-5	2.00 D-5	1.37 D-5	1.14 D-4
8	3.95 D-2	1.63 D-2	3.08 D-3	4.17 D-6	4.82 D-6	3.67 D-6	2.87D-5

5.2 Problem 2

This problem is the same as the last one except we force the partial differential with a source that is oscillatory and decays slowly in time. In particular we set

$$\frac{d}{dt}f(t,z) = zcos(t) - \epsilon\pi^2$$

$$g(t,z) = z^2 e^{-\epsilon t + cos(t)}$$

Table 3 shows the errors for the different codes. The trends for this problem are almost identical to those for the last problem. Table 4 shows how the errors decrease as one increases LSTEP.

Table 3:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	4.63 D-2	5.12 D-1	4.67 D-2	3.57 D-4	5.95 D-3	1.53 D-3	2.75 D-3
20	1.05 D-1	2.19 D-1	3.13 D-2	3.46 D-4	1.37 D-3	4.79 D-4	1.67 D-3
40	2.58 D-1	9.87 D-2	2.60 D-2	3.00 D-4	2.09 D-4	2.13 D-4	1.54 D-3
80	6.86 D-1	4.33 D-2	1.83 D-2	2.13 D-4	6.39 D-5	1.14 D-4	1.44 D-3
160	8.92 D-1	2.00 D-2	8.78 D-3	1.58 D-4	4.16 D-5	6.11 D-5	1.38 D-3
320	6.50 D-1	9.53 D-2	2.34 D-3	1.29 D-4	2.73 D-5	3.32 D-5	1.36 D-3

Table 4:

lSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	2.58 D-1	9.87 D-2	2.60 D-2	3.00 D-4	2.09 D-4	2.13 D-4	1.54 D-3
2	1.18 D-1	4.82 D-2	1.45 D-2	7.49 D-5	5.20 D-5	5.34 D-5	3.90 D-4
4	5.68 D-2	2.38 D-2	7.67 D-3	1.87 D-5	1.29 D-5	1.36 D-5	9.80 D-5
8	2.78 D-2	1.18 D-2	3.93 D-3	4.72 D-6	3.05 D-6	3.69 D-6	2.47 D-5

5.3 Problem 3

In the two previous problems we have balanced the decay from the diffusion with growth from the term involving $\frac{d}{dt}f$. In this problem we will not try to balance the decay from the diffusion. We set

$$\frac{d}{dt}f(t,z) = zcos(t)$$

$$g(t,z) = e^{-zcos(t)-\epsilon t}$$

Table 5 shows the errors for the different methods on this problem. Once again we see that errors in the fully implicit methods grow relative to the operator splitting methods, as ϵ decreases. There is one difference between this and the previous problems. In this problem the operator splitting method SPL1 does better than IMP and CONS even when ϵ is not small.

Table 6 confirms that we are getting the expected order of convergence for the different methods.

5.4 Problem 4

We solve a problem similar to the last one except we force the equations with a growing oscillatory term rather than a decaying one.

Table 5:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	9.80 D-3	4.79 D-3	8.37 D-3	2.40 D-4	1.02 D-4	1.68 D-2	2.30 D-4
20	1.27 D-2	3.72 D-3	6.80 D-3	2.40 D-4	8.50 D-3	4.26 D-3	1.53 D-4
40	1.86 D-2	3.94 D-3	4.72 D-3	2.05 D-4	7.11 D-5	1.08 D-3	1.02 D-4
80	3.93 D-2	3.04 D-3	2.88 D-3	1.60 D-4	5.15 D-5	2.74 D-4	8.57 D-5
160	8.63 D-2	2.11 D-3	1.89 D-3	1.22 D-2	3.52 D-5	7.57 D-5	8.06 D-5
320	2.04 D-1	1.43 D-3	1.29 D-3	9.52 D-5	2.42 D-5	3.20 D-5	7.90 D-5

Table 6:

KSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	1.86 D-2	3.94 D-3	4.72 D-3	2.05 D-4	7.11 D-5	1.08 D-3	1.02 D-4
2	9.23 D-2	1.99 D-2	2.40 D-3	5.11 D-5	1.77 D-5	2.70 D-4	2.56 D-5
4	4.60 D-3	1.00 D-3	1.21 D-3	1.72 D-5	4.32 D-6	6.76 D-5	6.32 D-6
8	2.30 D-3	5.02 D-4	6.08 D-4	3.09 D-6	9.99 D-7	1.69 D-5	1.52 D-6

$$\frac{d}{dt}f(t,z) = zcos(t)$$

$$g(t,z) = e^{-zsin(t)+\epsilon t}$$

Table 7 shows the results for the different methods. As for the last problem note that the code SPL1 does better than IMP even when ϵ is not small. It is does a little better than CONS for the larger values of ϵ, and a little worse for the smaller values of ϵ.

Table 7:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	8.86 D-3	4.90 D-3	8.94 D-3	1.72 D-4	4.14 D-5	1.89 D-2	1.65 D-4
20	1.18 D-2	3.32 D-3	6.42 D-3	1.95 D-4	5.35 D-5	4.97 D-5	1.13 D-4
40	1.68 D-2	3.54 D-3	4.12 D-3	1.82 D-4	5.99 D-5	1.21 D-3	8.18 D-5
80	3.43 D-2	2.87 D-3	2.64 D-3	1.49 D-4	4.87 D-5	3.07 D-4	6.91 D-5
160	7.45 D-2	2.03 D-3	1.78 D-3	1.13 D-4	3.44 D-5	7.87 D-5	6.60 D-5
320	1.72 D-1	1.40 D-3	1.23 D-3	8.44 D-5	2.42 D-5	3.30 D-5	6.57 D-5

Table 8 shows that each of the methods are getting their predicted order of convergence.

5.5 Problem 5

The previous problems showed that the operator splitting code SPL1 worked better than the fully implicit code IMP even for relatively large values of ϵ when

Table 8:

KSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	1.65 D-2	3.54 D-3	4.12 D-3	1.82 D-4	5.99 D-5	1.21 D-3	8.18 D-5
2	8.35 D-3	1.79 D-3	2.10 D-3	4.54 D-5	1.48 D-5	3.02 D-4	2.04 D-5
4	4.17 D-3	8.98 D-4	1.06 D-3	1.13 D-5	3.60 D-6	7.55 D-5	5.02 D-6
8	2.08 D-2	4.51 D-4	5.35 D-4	2.76 D-6	8.46 D-7	1.88 D-5	1.22 D-6

the operotar L_0 gives us a growth that cancels out the decay due to diffusion. For this problem we will include a term in L_0 that adds an additional decay to that due to diffusion. In particular we set

$$\frac{d}{dt}f(t,z) = zcos(t) + \epsilon\pi^2$$

$$g(t,z) = e^{-zsin(t)+\epsilon t}$$

Table 9 shows the results for the different methods. This problem behaves much like the first two problems. In particular, the codes IMP and CONS do better than SPL1 when ϵ is not small. As ϵ decreases the splitting codes SPL1 and CONS lowere their errors, while the fully implicit code IMP raises its errors.

Table 9:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	5.15 D-3	2.96 D-2	4.39 D-3	1.08 D-4	5.48 D-4	3.29 D-2	1.45 D-4
20	7.51 D-3	1.51 D-2	3.40 D-3	1.34 D-4	1.43 D-4	8.53 D-3	7.50 D-5
40	9.63 D-3	7.36 D-3	2.54 D-3	1.43 D-4	5.01 D-5	2.13 D-3	4.83 D-5
80	1.23 D-2	3.69 D-3	1.93 D-3	1.30 D-4	4.11 D-5	5.39 D-4	3.67 D-5
160	2.59 D-2	2.13 D-3	1.44 D-3	1.05 D-4	3.09 D-5	1.35 D-4	2.75 D-5
320	5.54 D-2	1.39 D-3	1.06 D-3	8.08 D-5	2.24 D-5	3.75 D-5	2.18 D-5

Table 10 shows that the codes are all getting their predicted rates of convergence.

Table 10:

KSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	9.63 D-3	7.36 D-3	2.54 D-3	1.43 D-4	5.01 D-5	2.13 D-3	4.83 D-5
2	4.86 D-3	3.69 D-3	1.30 D-3	3.57 D-5	1.25 D-5	5.33 D-4	1.20 D-5
4	2.44 D-3	1.85 D-3	6.42 D-4	8.83 D-6	3.14 D-6	1.33 D-4	2.92 D-6
8	1.22 D-3	9.23 D-4	3.22 D-4	2.12 D-6	8.55 D-7	3.32 D-5	7.05 D-7

5.6 Problem 6

The previous problems have all had periodic terms in the equations, or terms that are periodic with an exponential growth or decay. In this problem we include several incommensurate frequencies in our equations.

$$\frac{d}{dt}f(t,z) = zcos(\pi t) + z^2 cos(t)$$

$$g = e^{cos(\sqrt{3}t)}$$

Table 11 shows the results for for the different codes. Note that in this problem there is no growth or decay coming from the operator L_0, and as in the previous problems where this was the case, the code SPL1 performs better than IMP even when ϵ is not small. It performs almost identically to CONS for all values of ϵ.

Table 11:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	2.14 D-2	1.10 D-2	1.19 D-2	8.22 D-4	4.20 D-4	1.78 D-2	4.69 D-4
20	1.74 D-2	6.15 D-3	6.94 D-3	6.44 D-4	2.62 D-4	4.52 D-3	2.85 D-4
40	2.40 D-2	4.80 D-3	5.11 D-3	4.49 D-4	1.70 D-4	1.12 D-3	1.94 D-4
80	5.33 D-2	3.04 D-3	3.08 D-3	3.30 D-4	1.05 D-4	2.90 D-4	1.45 D-4
160	1.24 D-1	1.58 D-3	1.54 D-3	2.46 D-4	6.13 D-5	9.05 D-5	1.25 D-4
320	3.27 D-1	8.01 D-4	7.49 D-4	1.96 D-4	3.34 D-5	3.68 D-5	1.21 D-4

Table 12 confirms the order of convergence for all of the methods.

Table 12:

KSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	2.40 D-2	4.80 D-3	5.11 D-3	4.49 D-4	1.70 D-4	1.12 D-3	1.94 D-4
2	1.19 D-2	2.43 D-3	2.62 D-3	1.12 D-4	4.22 D-5	2.79 D-4	4.83 D-5
4	5.90 D-3	1.22 D-3	1.33 D-3	2.80 D-5	1.04 D-5	6.97 D-5	1.20 D-5
8	2.94 D-3	6.14 D-4	6.68 D-4	6.94 D-6	2.51 D-6	1.74 D-5	2.89 D-6

5.7 Summary of Linear Test Problems

We have seen that for all of the linear test problems the codes IMP, SPL1 , and CONS show first order convergence as we decrease the step size Δt. As one decreases ϵ the accuracy of the splitting codes SPL1 and CONS always imporve when compared to the accuracy of IMP. When ϵ is not small, the codes IMP and CONS seem to do better than SPL1 when there is growth or decay coming from the operator L_0. When this is not the case, the code SPL1 does better than IMP for all values of ϵ, and is comparable to CONS for all values of ϵ.

Almost identical trends occur for the second order splitting methods although the deterioration of the fully implicit method IMP2 is not as severe when we let ϵ get small. Also the second order splitting code CONS2 does not behave any better than IMP2 as ϵ gets small.

6 A Nonlinear Test Problem

In this section we test our codes on the equations for a Bursselator [6]. These equations are a system of nonlinear reaction diffusion equations that in certain parameter regimes give rise to oscillatory behavior. The equations for the Brusselator are

$$\frac{\partial T}{\partial t} = D_1 \frac{\partial^2 T}{\partial x^2} + \alpha - (\beta + 1)T + T^2 C \tag{14a}$$

$$\frac{\partial C}{\partial t} = D_2 \frac{\partial^2 C}{\partial x^2} + \beta T - T^2 C \tag{14b}$$

along with the boundary conditions

$$T(0,t) = T(1,t) = \alpha \tag{14c}$$

$$C(0,t) = c(1,t) = \frac{\beta}{\alpha} \tag{14d}$$

In order to begin integrating these equations we must also include some initial conditions on the functions $C(x,t)$ and $T(x,t)$. Before specifying these initial conditions we mention that these equations have the steady state solutions

$$T(x,t) = \alpha$$

and

$$C(x,t) = \frac{\beta}{\alpha}.$$

We will assume that

$$D_1 = D_2 = \epsilon.$$

In this case the analysis in [6] shows that the steady solution has an oscillatory instability provided

$$2\pi^2\epsilon + \alpha^2 + 1 - \beta < 0, \tag{15a}$$

and,

$$\left(\alpha^2 + 1 - \beta - 4\gamma^2\epsilon\right)\left(\alpha^2 + 1 - \beta\right) - 4\alpha^2 < 0. \tag{15b}$$

We will use the constants

$$\alpha = .6,$$

and

$$\beta = 2.$$

It can be verified that with these values of α and β the conditions in eqn. (15) are satisfied provided ϵ is small enough. When ϵ is small enough we will get an oscillatory solution. In this section we will test the various codes on how they track the approach to this oscillatory solution.

We use the initial conditions

$$T(x,0) = \alpha + x(1-x)$$

$$C(x,0) = \frac{\beta}{\alpha} + x^2(1-x).$$

As in the previous section we will define our time step as

$$\Delta t = \frac{1}{10 \times LSTEP}$$

and we will integrate our equations from $t = 0$ to $t = \frac{2}{\epsilon}$. When we are comparing the different codes for different values of ϵ we will choose $LSTEP = 1$. When we are checking the convergence of the different codes by varying $LSTEP$ we will choose $\frac{!}{\epsilon} = 40$.

The error in the solution is computed as in the case of the linear test problems except that there are now two unknowns, so we add the absolute values of the errors in the flux for both C and T at the left hand boundary.

6.1 Problem 1

This problem solves the equations for the Brusselator with no forcing terms. When $\frac{1}{\epsilon} = 10$ or 20 the steady state solution is stable and our solution eproaches this steady state. All of the cases where $\frac{1}{\epsilon}$ is bigger than 20 eventually approach an oscillatory solution.

Table 13 shows errors in the different methods as we vary ϵ. Note that as ϵ gets smaller the implicit method IMP1 steadily degenerates, but the operator splitting code SPL1 shows a slight improvement in its error.

Table 14 shows that all of the methods are in fact getting the expected order o convergence.

References

[1] N.N. Yanenko, The Method of Fractional Steps, Springer ,1971 .

Table 13:

$\frac{1}{\epsilon}$	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
10	8.85 D-2	3.30 D-1	1.20 D-1	4.67 D-3	1.53 D-2	7.71 D-3	1.60 D-2
20	1.41 D-1	6.83 D-2	1.25 D-1	4.69 D-3	2.19 D-3	4.37 D-3	4.84 D-3
40	2.56 D-1	5.41 D-2	5.38 D-1	1.35 D-2	8.18 D-4	2.63 D-3	9.00 D-3
80	8.00 D-1	4.45 D-2	6.59 D-1	2.43 D-2	1.92 D-3	3.12 D-3	9.91 D-3
160	1.44 D+0	2.95 D-2	6.93 D-1	2.24 D-2	1.84 D-3	2.82 D-3	7.57 D-3

Table 14:

KSTEP	IMP1	SPL1	CONS	IMP2	SPL2	STR	CONS2
1	2.56 D-1	5.41 D-2	5.38 D-1	1.35 D-2	8.18 D-4	2.63 D-3	9.00 D-3
2	1.24 D-1	2.80 D-2	2.87 D-1	3.38 D-3	2.01 D-4	6.55 D-4	2.24 D-3
4	6.06 D-2	1.42 D-2	1.46 D-1	8.45 D-4	4.99 D-5	1.64 D-4	5.59 D-4
8	3.00 D-2	7.17 D-3	7.39 D-2	2.11 D-4	1.24 D-6	4.12 D-5	1.40 D-5

[2] E.S. Oran and J.P. Boris, Numerical Simulation of Reactive Flow, Elsevier, New York, 1987 .

[3] Gilbert Strang, On the Construction and Comparison of Difference Schemes, SIAM J. Numer. Anal., vol 5 no 3, pp 506-517 (1963).

[4] Not known yet

[5] I.S. Wichman, On the Use of Operator-Splitting Methods for the Equations of Combustion, Combustion and Flame , vol 83, pp240-252, 1991 .

[6] Hermann Haken, Synergetics, Springer-Verlag, New York Heidelberg Berlin, 1983

æ

Distribution List:

1	MS 0847	Sudip Dosanjh, 09233
1	MS 1110	David Womble, 09214
1	MS 1110	Richard Lehoucq, 09214
1	MS 0847	Curt Ober, 09233
5	MS 1110	Louis Romero, 09214
1	MS 1110	John Shadid, 09233
1	MS 9018	Central Technical Files, 8945-1
2	MS 0899	Technical Library, 9616
1	MS 0612	Review & Approval Desk, 9612
		For DOE/OSTI
1	MS 0161	Patent & Licensing Office, 11500