An Analysis of the TR-BDF2 integration scheme

by

Sohan Dharmaraja

Submitted to the School of Engineering
in Partial Fulfillment of the Requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author .... [illegible] ....
School of Engineering
July 13, 2007

Certified by .... [illegible] ....
W. Gilbert Strang
Professor of Mathematics
Thesis Supervisor

Accepted by .... [illegible] ....
Jaume Peraire
Professor of Aeronautics and Astronautics
Co-Director, Computation for Design and Optimization

An Analysis of the TR-BDF2 integration scheme

by

Sohan Dharmaraja

Submitted to the School of Engineering
on July 13, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

We intend to try to better our understanding of how the combined L-stable 'Trapezoidal Rule with the second order Backward Difference Formula' (TR-BDF2) integrator and the standard A-stable Trapezoidal integrator perform on systems of coupled non-linear partial differential equations (PDEs). It was originally Professor Klaus-Jürgen Bathe who suggested that further analysis was needed in this area. We draw attention to numerical instabilities that arise due to insufficient numerical damping from the Crank-Nicolson method (which is based on the Trapezoidal rule) and demonstrate how these problems can be rectified with the TR-BDF2 scheme.

Several examples are presented, including an advection-diffusion-reaction (ADR) problem and the (chaotic) damped driven pendulum. We also briefly introduce how the ideas of splitting methods can be coupled with the TR-BDF2 scheme and applied to the ADR equation to take advantage of the excellent modern day explicit techniques to solve hyperbolic equations.

Thesis Supervisor: W. Gilbert Strang
Title: Professor of Mathematics

Acknowledgments

First and foremost, I would like to thank my family. I would not be the person I am today if not for them. In many ways I was always the one who was at risk of not living up to my potential but over the years I've made strides in the right direction. So once again: thank you Mom, thank you Dad, thank you Malli for the never ceasing, never diminishing flow of comments, criticism and confidence.

Thank you to Professor Strang. Apart from teaching me how to write a thesis, he has taught me how to be a researcher. I am privileged to have worked with him and to have learned his methods. His advice on life and the personal interest he has taken in my future will always be appreciated. Thank you again sir, for taking an chance on me when I walked into your office those many months ago, looking for 'something interesting to work on'.

Finally, thank you to my friends, old and new. My time here at the Massachusetts Institute of Technology would have undoubtebly been less enjoyable if not for them. I will always look back fondly on these memories and friendships.

Contents

List of Figures

Chapter 1

Introduction

In the context of ordinary differential equations (ODEs) the trapezoidal rule is a popular method of choice and has already been implemented in the well known circuit simulator SPICE [1]. However, it is well known that for extremely stiff problems this method is not efficient: the stability region[1] of the trapezoidal rule forces a drastic reduction in the maximum allowable time step in order to compute a numerical solution. What makes things especially tricky is that the majority of nonlinear dynamics problems can transition between being stiff and non-stiff. It would be ideal to have a method of numerical integration that would encapsulate the ease of implementation of the trapezoidal rule as well as second order accuracy, while not having to worry about our nonlinear system becoming excessively stiff.

The method we study is in some sense a cyclic linear multistep method, consisting of an initial step of the Trapezoidal rule, followed by a second order backward differentiation formula. This method is more commonly known as the TR-BDF2 scheme (originally devised by Bank et al [2]) and is second order accurate. More importantly, it is L-stable (explained in section 3.3). This combination has a significant advantage over the Trapezoidal rule alone (which will be shown to be only A-stable) when it

[1]The stability region of a numerical method is defined to be the values of $z = \lambda_i h$ (possibly complex) which ensure that the amplification factor of numerical errors in our solution is strictly less than one. Here, λ_i are the eigenvalues of the Jacobian of our differential operator and h is the stepsize we are using.

comes to numerical damping of oscillations in our solution.

We begin this thesis by familiarizing ourselves with some well established concepts. We first introduce the notion of a multistep method, after which we elaborate on the nonlinear solver (Newton-Raphson) that we use throughout our analysis. Then we explain how these two ideas together can be used to integrate systems of differential equations. We proceed to define exactly what it means for a method to A-stable and L-stable and the implications and possible hazards of the former.

The section on numerical results is intended to highlight one of the key points made throughout this thesis: the numerical damping provided by widely accepted integration methods (Crank-Nicolson and the Trapezoidal rule, which are both only A-stable) is insufficient in extremely stiff systems. We apply the TR-BDF2 and Trapezoidal integrators to a wide range of problems arising in nonlinear dynamics. The systems we considered were primarily pendulum examples (chaotic, double and elastic). We end by briefly touching on how the idea of operator splitting can be used to very efficiently solve a coupled system of partial differential equations (PDEs) - namely the advection-diffusion-reaction equation, by applying a different integration method to each subprocess.

1.1 Multistep methods

1.1.1 Adams Methods

It is standard practice when faced with a partial differential equation involving space and time, to first discretize the spatial variable(s). After this initial discretization (more formally, a *semi-discretization*) we are left with an initial value problem for a set of ordinary differential equations. ODEs arise as a result of Fourier analysis and splitting methods, when applied to PDEs. The focus of this thesis will be mainly on *linear multistep* methods, which are powerful techniques to solve ODEs approximately. Runge-Kutta algorithms and Adams methods are perhaps the most popular choices, however we will be focusing on the Adams schemes.

The least accurate Adams schemes are the Forward and Backward Euler algorithms (which are *single step methods*). The idea behind these multistep methods is that we may better approximate the local gradient by taking into account an increasing number of previous/future values of the function to be integrated. For example, consider solving the ordinary differential equation

$$\frac{dy}{dt} = f(y,t) \tag{1.1}$$

We can step forward in time by calculating

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y,t)\,dt \tag{1.2}$$

The Adams family of multistep methods is derived by computing the integrand approximately, via an interpolating polynomial through previously computed values of the function f. For example, consider using the value of f_n and f_{n-1} to interpolate our function f and hence compute the integrand. Define

$$P(t) = f(y_{n-1}, t_{n-1})\frac{t-t_n}{t_{n-1}-t_n} + f(y_n, t_n)\frac{t-t_{n-1}}{t_n-t_{n-1}} \tag{1.3}$$

Assuming that we have a uniform discretization with stepsize h, say, our integrand becomes

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} f(y,t)\,dt &\approx \int_{t_n}^{t_{n+1}} P(t)\,dt && (1.4)\\
&= \int_{t_n}^{t_{n+1}} f(y_n,t_n)\frac{t-t_{n-1}}{h} - f(y_{n-1},t_{n-1})\frac{t-t_n}{h}\,dt && (1.5)\\
&= \frac{h}{2}f(y_n,t_n)\left[\frac{t-t_{n-1}}{h}\right]^2\Bigg|_{t_n}^{t_{n+1}} - \frac{h}{2}f(y_{n-1},t_{n-1})\left[\frac{t-t_n}{h}\right]^2\Bigg|_{t_n}^{t_{n+1}} && (1.6)\\
&= \frac{h}{2}\left[3f(y_n,t_n) - f(y_{n-1},t_{n-1})\right] && (1.7)
\end{aligned}$$

So finally we are left with an explicit expression for y_{n+1}:

$$y_{n+1} = y_n + \frac{h}{2}\left[3f(y_n,t_n) - f(y_{n-1},t_{n-1})\right] \tag{1.8}$$

Explicit schemes of this kind are known as *Adams-Bashforth* methods. In general, an n^{th} order scheme can be expressed as

$$y_{n+1} = y_n + \sum_{i=1}^{n} w_i f(y_{n+1-i}, t_{n+1-i}) \tag{1.9}$$

It is clear why these methods are coined as explicit: when we know the values of our function f at the required stages of the method, we can directly calculate the value of y_{n+1}. This is not the the case for an *implicit method* where we will be required to solve an equation to obtain the value of our next iterate, y_{n+1}.

It is straightforward to calculate the next few expressions for increasingly higher order approximations. For completeness we list them here up to third order (the first order Adams-Bashforth method is more commonly known as the *Forward Euler* method).

$$y_{n+1} = y_n + hf(y_n, t_n) \tag{1.10}$$

$$y_{n+1} = y_n + \frac{h}{2}\left[3f(y_n, t_n) - f(y_{n-1}, t_{n-1})\right] \tag{1.11}$$

$$y_{n+1} = y_n + \frac{h}{12}\left[23f(y_n, t_n) - 16f(y_{n-1}, t_{n-1} + 5f(y_{n-2}, t_{n-2})\right] \tag{1.12}$$

For the *implicit* Adams methods (more commonly known as the *Adams-Moulton* methods) there is one additional term in the multistep equation:

$$y_{n+1} = y_n + w_0 f(y_{n+1}, t_{n+1}) + \sum_{i=1}^{n} w_i f(y_{n+1-i}, t_{n+1-i}) \tag{1.13}$$

Note now that since y_{n+1} appears on both sides of 1.13 we need to solve this equation for y_{n+1} (hence the term implicit). If the function f is nonlinear, a nonlinear solver such as Newton-Raphson may be required.

Once again, with the same technique as above (using an interpolating polynomial to approximate our integrand) it is trivial to calculate the weights w_i for these implicit schemes. The first, second and third order Adams-Moulton methods are listed below:

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \tag{1.14}$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right] \tag{1.15}$$

$$y_{n+1} = y_n + \frac{h}{12}\left[5f(y_{n+1}, t_{n+1}) + 8f(y_n, t_n) - f(y_{n-1}, t_{n-1})\right] \tag{1.16}$$

The first order Adams-Moulton method is more commonly known as the *Backward Euler* while the second is known as the *Trapezoidal rule.*

1.1.2 Backwards Difference Formulae (BDF)

We derived the Adams methods described earlier by approximating the derivative $f'(y,t)$ at time $y = y_n$, $t = t_n$ with an interpolating polynomial which we then integrated to get the value of y_{n+1}

In Backwards Differentiation methods, we approximate $y' = f(y,t)$ at $t = t_{n+1}$ using a backwards differencing formula. For example,

$$\frac{y_{n+1} - y_n}{h} \approx f(y_{n+1}, t_{n+1}) \tag{1.17}$$

We recognize this as the Backward Euler formula from earlier (the first order Adams Moulton method).

We can increase the order of our BDF by taking into consideration more terms:

$$\frac{3}{2}y_{n+1} - 2y_n + \frac{1}{2}y_{n-1} = hf(y_{n+1}, t_{n+1}) \tag{1.18}$$

$$\frac{11}{6}y_{n+1} - 3y_n + \frac{3}{2}y_{n-1} - \frac{1}{3}y_{n-2} = hf(y_{n+1}, t_{n+1}) \tag{1.19}$$

We will see later on that BDF have an important role in solving stiff systems of differential equations. Loosely speaking, the term *stiff* means that the step size for the approximate solution is more severely limited by stability of the method of choice than by the accuracy of the method. This is explained in greater detail in section 3.1. *Stability* and *accuracy* are defined later in this chapter.

Backwards differentiation formulae are implicit, like the Adams-Moulton methods. In practice they are more stable than the Adams-Moulton schemes of the same order but less accurate. It also worth noting that BDF schemes of order greater than 6 are unstable [5].

1.1.3 Convergence Analysis

Let us define what exactly we mean for a multistep method to be convergent.

To recap, the goal is to solve the initial value problem

$$y' = f(y,t), \quad a \leq t \leq b$$
$$y(a) = \alpha$$

We have our multistep method

$$y_{n+1} = y_n + \sum_{i=0}^{n} w_i f(y_{n+1-i}, t_{n+1-i}) \tag{1.20}$$

where $w(0)$ could possibly be zero, depending on whether or not we are using an implicit scheme.

Then, given a stepsize h we say that a multistep method is *convergent* if we can achieve arbitrary accuracy with a sufficiently small h, i.e.

$$\lim_{h\to 0} |y_n - y(t_n)| = 0$$

With the goal of understanding convergence, we define the global error e_n

$$e_n = y_n - y(t_n) \tag{1.21}$$

where y_n is the approximate solution for y (from our multistep method) and $y(t_n)$ is the exact solution for y at time $t = t_n$. A natural question then is: how does the global error e_n behave? Does it tend to zero as our stepsize h tends to zero?

Before we begin to address these questions, it is useful to define the notion of *order of accuracy* of a multistep method. We first define the local truncation error (LTE)

$$l_{n+1} = y_{n+1} - y(t_{n+1}) \tag{1.22}$$

assuming that the previously computed value of y_n was identically equal to the exact solution for y at time $t = t_n$. We then say that the multistep method is of *order* p if

$$l_{n+1} = \boldsymbol{O}(h^{p+1})$$

For example, consider the previously derived expression for the Adams-Moulton second order method, (the Trapezoidal rule) equation 1.16

$$y_{n+1} = y_n + \frac{h}{2}\left[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right]$$

Then, our definition of the LTE yields

$$\begin{aligned} l_{n+1} &= y(t_n) + \frac{h}{2}\left[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right] - y(t_{n+1}) && (1.23) \\ &= y(t_n) + \frac{h}{2}\left[y'(t_{n+1}) + y'(t_n)\right] - y(t_{n+1}) && (1.24) \end{aligned}$$

We can combine $y(t_{n+1})$ and $y'(t_{n-1})$ by Taylor expanding about $t = t_n$

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \boldsymbol{O}(h^3) && (1.25) \\ y'(t_{n+1}) &= y'(t_n) + hy''(t_n) + \boldsymbol{O}(h^2) && (1.26) \end{aligned}$$

Substituting all this into the expression for the LTE:

$$\begin{aligned} l_{n+1} &= y(t_n) + \frac{h}{2}\left[y'(t_n) + hy''(t_n) + \boldsymbol{O}(h^2) + y'(t_n)\right] - \left[y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \boldsymbol{O}(h^3)\right] \\ &= \boldsymbol{O}(h^3) \end{aligned}$$

By definition, the Trapezoidal rule is indeed second order accurate. So now, with the goal of quantifying the global error we define the notion of stability for a multistep method

$$y_{n+1} = y_n + \sum_{i=1}^{s} \alpha_i y_{n+1-i} + h \sum_{j=-1}^{r} \beta f_{n-j} \tag{1.27}$$

We associate with each multistep scheme the *characteristic polynomials*

$$\rho(\xi) = \xi^s - \alpha_1 \xi^{s-1} - \alpha_2 \xi^{s-2} - \ldots - \alpha_s \tag{1.28}$$

$$\sigma(\eta) = \beta_r + \beta_{r-1}\eta + \beta_{r-2}\eta^2 + \ldots + \beta_{-1}\eta^{r+1} \tag{1.29}$$

(Note that for all of the Adams' methods, $\alpha_i = 0$)

We then say that the *root condition* of the multistep method is satisfied if the roots of $\rho(\xi) = 0$ lie on or inside the unit circle. If this condition holds, our multistep method is *stable.*

We say the method is *consistent* if

$$\rho(1) = 0$$
$$\rho'(0) = \sigma(1)$$

Then, we can finally say our multistep method is *convergent* if and only if the root condition is satisfied and the method is consistent.

The proof of this result is somewhat lengthy and the reader can refer to [3].

It is also common to define the notion of *strong* and *weak* stability. The method is considered to be *strongly stable* if $\xi = 1$ is a simple root and *weakly stable* otherwise.

After establishing the stability of the multistep method, we now need to determine the allowable step-sizes with which we can march forward in time. More precisely, given our computed approximate solution y_n, which satisfies our multistep equation, does the method allow perturbations

$$\tilde{y_n} = y_n + e_n$$

to grow. That is, does $\lim_{n\to\infty} |e_n| = \infty$?

Consider for example the Forward Euler algorithm. For simplicity, assume now that $f(y,t) = \lambda y + \tilde{f}(t)$ in equation 1.2. Then the multistep equation reads:

$$\frac{\tilde{y_{n+1}} - \tilde{y_n}}{h} = \lambda \tilde{y}_n + \tilde{f}(t_n) \tag{1.30}$$

$$\frac{y_{n+1} + e_{n+1} - y_n - e_n}{h} = \lambda(y_n + e_n) + \tilde{f}(t_n) \tag{1.31}$$

$$\tag{1.32}$$

Now we know that y satisfies the exact multistep equation. That is:

$$\frac{y_{n+1} - y_n}{h} = \lambda(y_n) + \tilde{f}(t_n)$$

So we are left with:

$$\frac{e_{n+1} - e_n}{h} = \lambda e_n$$
$$\Rightarrow e_{n+1} = (1 + \lambda h) e_n$$

So e_n satisfies the homogeneous difference equation. We now introduce an *amplification factor g*

$$e_{n+1} = g e_n \tag{1.33}$$

Thus if $|g| > 1$ then $\lim_{n\to\infty} |e_n| = \infty$.

Calculating g for the Forward Euler discretization yields: $g = 1 + \lambda h$. It is then common to visualize the values of λ and the stepsize h on the complex plane which ensure that $|g| < 1$. We set $z = \lambda h$ and denote this set of allowable values the *stability region* for the Forward Euler method.

For comparison, the stability regions for $p = 2$, and $p = 3$ are also plotted.

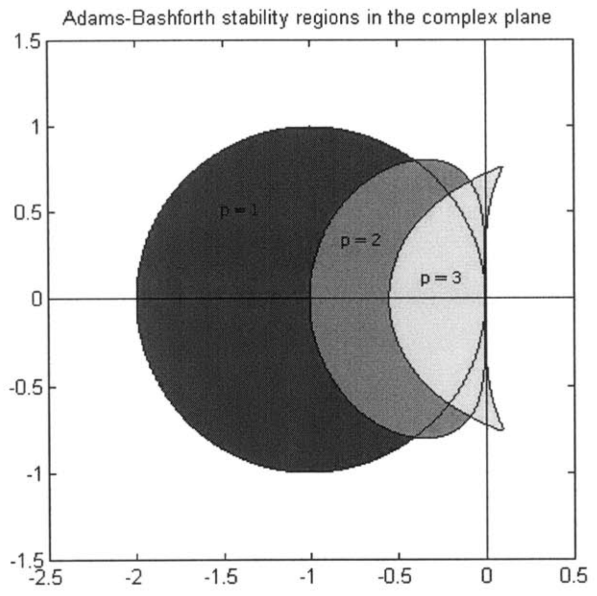
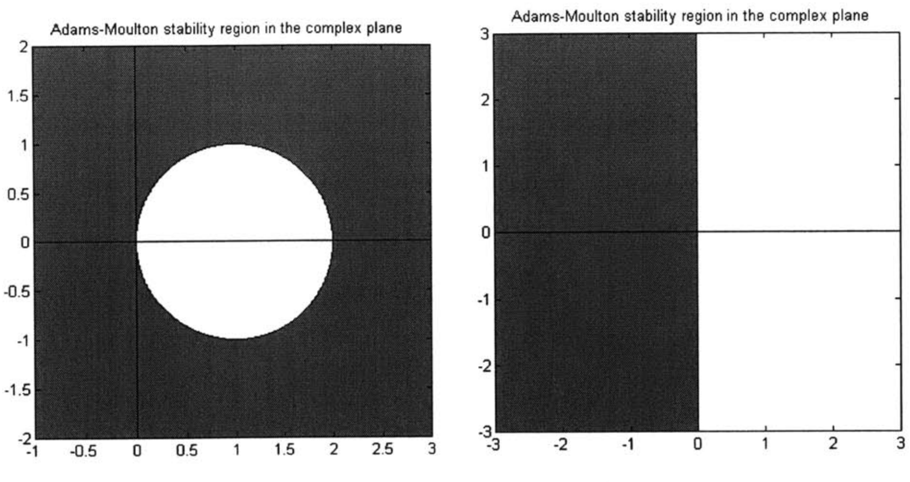


Figure 1-1: Adams Bashforth stability region



(a) Backward Euler (b) Trapezoidal rule

Figure 1-2: Adams-Moulton stability region for p = 1 and 2

We can just as easily derive the stability region for the Adams Moulton methods using the same method as before: Taylor expand to obtain the amplification factor g, then plot the values of z which ensure $|g| < 1$ on the complex plane.

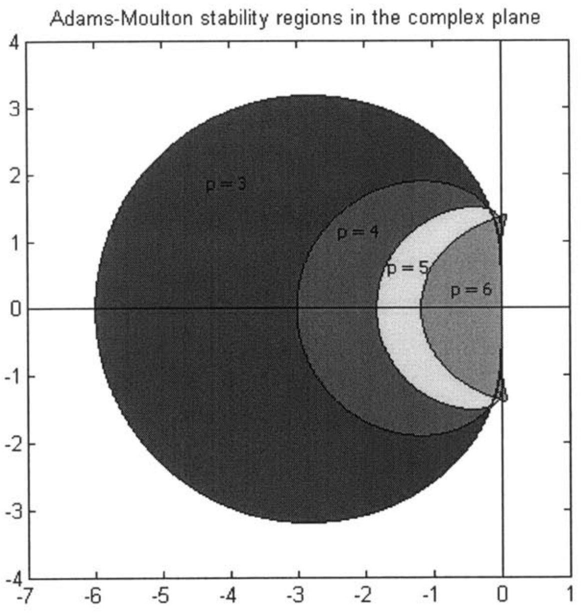


Figure 1-3: Adams Moulton stability region for p = 3, 4, 5 and 6

Similarly, we have the stability regions for the Backwards Differentiation Formulae derived earlier:

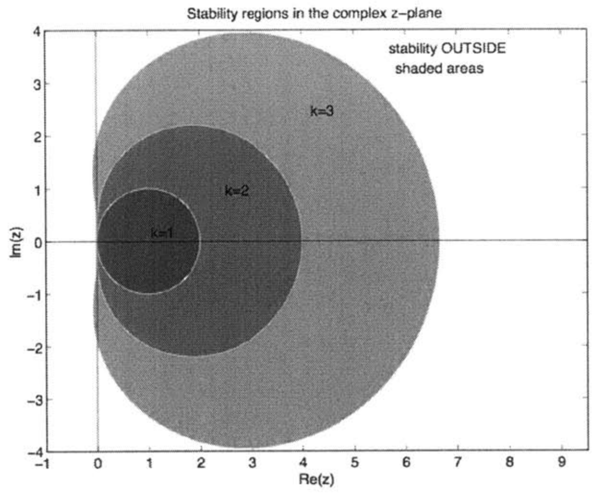


Figure 1-4: BDF stability region for p = 1, 2 and 3

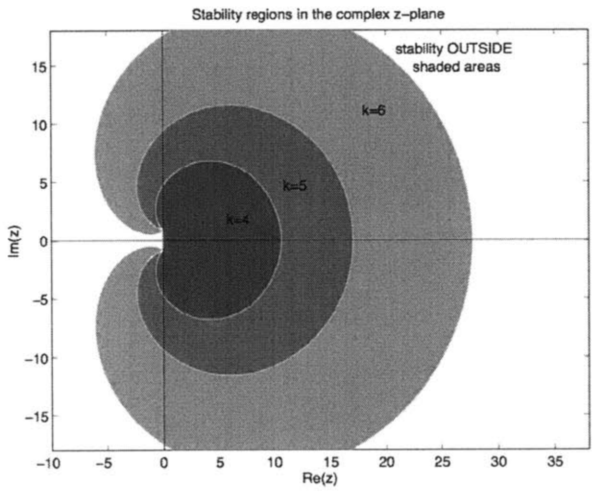


Figure 1-5: BDF stability region for p = 4, 5 and 6

Chapter 2

Nonlinear solvers

2.1 Newton's method for solving systems of non-linear systems of equations

Newton's method (or the Newton-Raphson method) is a well known iterative numerical technique for finding the (real) roots of a function f, say. We will see that it is a very efficient root finding algorithm, in the sense that if our current iterate is 'close' to a root, we will converge quadratically to it.

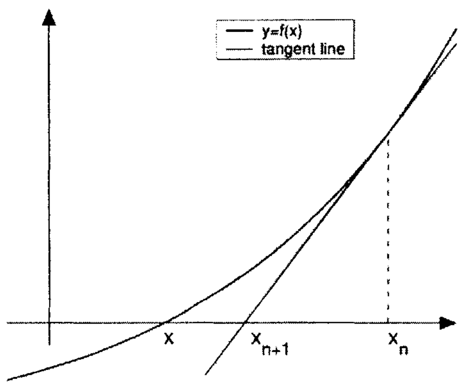


Figure 2-1: Newton's method in 1D

Newton's method is perhaps most easily explained with an illustration in one dimension. The idea is simple: at each iterate x_n we draw the tangent to the function and calculate where the tangent line intersects the x-axis. We take this value of x as our new iterate, x_{n+1}.

Mathematically speaking:

$$f'(x_n) = \frac{0 - f(x_n)}{x_{n+1} - x_n}$$

Rearranging this equation for our new iterate we obtain:

$$x_{n+1} = x_n - \frac{x_n}{f'(x_n)} \tag{2.1}$$

The idea of root solving in this way extends naturally to higher dimensions. Assume now that we are to solve n nonlinear equations. We can write these n equations as:

$$f(\underline{x}) = 0$$

where $\underline{x}$ is an n dimensional vector and $f : R^n \rightarrow R^n$

Proceeding in the analogous way as the one dimensional case, we replace the scalar derivative $f'(x_n)$ with a matrix of derivatives called the Jacobian.

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

We can demonstrate the method with an example: consider solving the system of equations with two unknowns

$$f_1(x,y) = xy^2 + sin(y) = 0$$

$$f_2(x,y) = cos(x) + y^3 = 0$$

Our Jacobian matrix is then

$$J(x,y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} y^2 & 2xy + cos(y) \\ -sin(x) & 3y^2 \end{bmatrix}$$

So the Newton-Raphson iterates are as follows:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - J^{-1} \begin{bmatrix} f_1(x_n, y_n) \\ f_2(x_n, y_n) \end{bmatrix}$$

substituting our expressions for J, $f_1(x_n, y_n)$ and $f_2(x_n, y_n)$ to give us the explicit iterative algorithm:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \begin{bmatrix} y_n^2 & 2x_n y_n + cos(y_n) \\ -sin(x_n) & 3y_n^2 \end{bmatrix}^{-1} \begin{bmatrix} x_n y_n^2 + sin(y_n) \\ cos(x_n) + y_n^3 \end{bmatrix}$$

n	x_n	y_n	$Residual f_1$	$Residual f_2$
1	6.863161e-001	3.985806e-001	1.841471e+000	1.540302e+000
2	1.672218e+000	-4.654934e-002	4.971432e-001	8.369068e-001
3	1.570680e+000	4.597843e-003	4.290910e-002	1.013492e-001
4	1.570796e+000	3.269765e-005	4.631031e-003	1.164770e-004
5	1.570796e+000	1.679210e-009	3.269933e-005	1.923241e-007
6	1.570796e+000	4.429249e-018	1.679210e-009	7.000528e-014
7	1.570796e+000	0.00000000000	4.429249e-018	6.123234e-017

2.1.1 Convergence analysis

An attractive feature of the Newton-Raphson method for solving systems of equations is that we can obtain quadratic convergence under certain assumptions (discussed at the end of this section) once we are sufficiently close to the solution. We briefly present the proof of this for the one dimensional case.

Assume the equation we wish to solve is $f(x) = 0$ with corresponding solution $x = x^*$. Then we can truncate the Taylor expansion about our current iterate x_k at the point x^* using the mean value theorem to yield

$$0 = f(x^*) = f(x_k) + \frac{df}{dx}(x_k)(x^* - x_k) + \frac{1}{2}\frac{d^2 f}{dx^2}(\tilde{x})(x^* - x_k)^2 \tag{2.2}$$

for some $\tilde{x} \in [x_k, x^*]$

We then truncate the Taylor expansion about the solution x_k at our *next* iterate, x_{k+1}

$$0 = f(x_k) + \frac{df}{dx}(x_k)(x_{k+1} - x_k) \tag{2.3}$$

Subtracting we obtain

$$\frac{df}{dx}(x_k)(x_{k+1} - x_k) = \frac{d^2 f}{dx^2}(\tilde{x})(x^* - x_k)_2 \tag{2.4}$$

If we assume that $\left[\frac{df}{dx}(x)\right]^{-1} \frac{d^2 f}{dx^2}(x) \leq L \; \forall x$ we then have

$$|x_{k+1} - x^*| \leq L|x_k - x^*|^2 \tag{2.5}$$

Hence we see quadratic convergence if L is bounded.

Chapter 3

Numerical integrators

3.1 The concept of stiffness

Stiffness is a phenomenon exhibited by certain ODEs. Physically, it signifies the fact that the solution has constituents which independently evolve on widely varying time-scales. We can elucidate this point with a concrete example. Consider the equation

$$y'' + 100y' + 99y = 0 \tag{3.1}$$

The eigenvalues are $\lambda_1 = -99$ and $\lambda_2 = -1$, so the solution is given by

$$y = Ae^{-t} + Be^{-99t} \tag{3.2}$$

Now, one would expect to be able to 'ignore' the transient Be^{-99t} term when it becomes 'insignificant' but stability theory dictates that we need our step length h, multiplied by our largest negative eigenvalue λ_1 to lie in our absolute stability region (as discussed in section 1.1.3). Should we use Forward Euler, this would restrict our choice of h to lie in the range

$$0 < h < \frac{2}{99} \tag{3.3}$$

Needless to say, this is a very severe constraint on our maximum allowable stepsize in order to guarantee stability. More often than not, to bound the LTE (and hence maintain a desired level of accuracy) when calculating a numerical solution, the step length predicted to meet these requirements is safely within the stability region. However, there are instances (as seen in example 3.1) that this is *not* the case - the maximum possible step length is determined by the boundary of the stability region. Problems with this feature are classified as *stiff.*

Mathematically, we can also say the ODE

$$y' = f(t, y)$$

is stiff if the eigenvalues of the Jacobian differ in magnitude by a significant amount. What this practically means, is that if an explicit method is used (such as *Forward Euler*) to meet stability requirements we have to use an extremely small step size h to march the solution y in time. It seems computationally inefficient to be doing so many steps to compute our numerical solution, which we already may know to be reasonably smooth.

3.2 A-stable integration methods

The notion of stability as motivated by Dahlquist has a strong connection with the linear model problem:

$$y' = \lambda y \tag{3.4}$$

where λ is a complex constant with a negative real part.

We say that a numerical method is A-stable if its stability region contains the whole left half plane. This is a very strict requirement: we can immediately deduce that explicit methods (which have finite stability regions) cannot meet this criterion. For example, all of the explicit Runge-Kutta (RK) methods have finite stability regions[1]

It is easy to see that the trapezoidal rule is indeed A-stable. For our model problem, we have

$$y_{n+1} = \left|\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right| y_n \tag{3.5}$$

and for stability, $\left|\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right| < 1 \Leftrightarrow \text{Re}(h\lambda) < 0$. Therefore the trapezoidal rule is A-stable. It is straightforward to show that Backward Euler is also A-stable.

The trade-off in using an implicit method, which can guarantee A-stability at each iteration, is that we must solve a system of (possibly nonlinear) equations. The Newton-Raphson method is usually the method of choice because more often than not (provided we are using a reasonable step size h) we have a good initial guess, so quadratic convergence is observed almost immediately.

In the context of the linear model problem above: we know the system is stable for all $\text{Re}(\lambda) < 0$. Given this, it would be nice then, if our integration method was stable

[1]However there do exist implicit RK techniques [6] that *are* A-stable.

for any step length h, i.e. our stability region should be the entire left half plane - hence an A-stable integration method.

We will see later on, that for the nonlinear examples we consider, although A-stable methods can be used, they come with an important caveat: sometimes these integration methods do not provide adequate damping to maintain stability (in particular, the dynamical systems that exhibit chaotic behavior). Round-off errors can accumulate to the point where they cause our numerical solution to explode.

3.3 L-stable integration methods

As briefly motivated at the end of the last section, A-stable methods must be used with caution. We proceed to define what it means for an integration method to be L-stable and highlight examples of where A-stable methods fail to produce a numerical solution while L-stable methods do not break down.

Beneath is the solution to (stiff) equation 3.1 with the initial conditions chosen to ensure that $A = B = 1$ using the second order Trapezoidal rule and the first order Backward Euler method. The fact that the Trapezoidal rule is only A-stable manifests itself as the 'ringing' effect seen below.

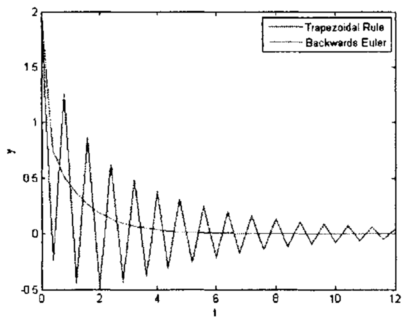


Figure 3-1: Solution to equation 3.1 with $dt = 0.4$

L-stability is a stronger requirement than A-stability. Once again, its definition is based on the linear test problem $y' = \lambda y$. We say that the integration method is L-stable if $y_{n+1}/y_n \rightarrow 0$ as $h\lambda \rightarrow -\infty$

We now show the trapezoidal rule (proven earlier to be A-stable) is *not* L-stable. We know

$$
\begin{aligned}
\frac{y_{n+1}}{y_n} &= \left|\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right| && (3.6)\\
&= \frac{1+hRe(\lambda)+\frac{1}{4}h^2|\lambda^2|}{1-hRe(\lambda)+\frac{1}{4}h^2|\lambda^2|} \rightarrow 1 \; as \; h\lambda \rightarrow -\infty && (3.7)
\end{aligned}
$$

There is however a way to overcome this difficulty using a extrapolation/smoothing technique [7] but that is not the focus of this thesis.

The L-stable method which we will be focusing on will be the composite second order trapezoidal rule/backward differentiation method (TR-BDF2). For simplicity, assume we already have a second order accurate discretization of the form

$$\frac{dy}{dt} = f(y) = Ay$$

Then, as intuition might suggest, the TR-BDF2 scheme consists of two steps: the first marches our solution from t_n to $t_{n+\gamma}$ using the (second order) Trapezoidal rule -

$$y_{n+\gamma} - \gamma\frac{\Delta t}{2}Ay_{n+\gamma} = y_n + \gamma\frac{\Delta t}{2}Ay_n \tag{3.8}$$

and the second step marches our solution from $t_{n+\gamma}$ to t_{n+1} using the second order Backward Differentiation Formula (BDF2)

$$y_{n+1} - \frac{1-\gamma}{2-\gamma}\Delta t A y_{n+1} = \frac{1}{\gamma(2-\gamma)}y_{n+\gamma} - \frac{(1-\gamma)^2}{\gamma(2-\gamma)}y_n \tag{3.9}$$

It is worth noting that the Jacobians for the Trapezoidal rule and the BDF scheme are:

$$J_{Trap} = I - \gamma \frac{\Delta t}{2} A$$
$$J_{BDF} = I - \frac{1-\gamma}{2-\gamma} \Delta t A$$

These matrices are identical if

$$\frac{2}{\gamma} = \frac{2-\gamma}{1-\gamma}$$

Solving this gives $\gamma = 2 - \sqrt{2}$. Using this value of γ means the Jacobian matrices for the Trapezoidal and BDF steps are the same. This means we only have to invert one matrix for the TR-BDF2 method, even though we are using two different implicit schemes. It was also proven that it is precisely this value that minimizes the LTE when implementing the TR-BDF2 scheme [2].

We now proceed to prove that the TR-BDF2 scheme as described above with this value of γ is indeed L-stable. From the Trapezoidal step, we have

$$y_{n+\gamma} = \frac{(2 - \gamma h \lambda)}{(2 + \gamma h \lambda)} y_n \tag{3.10}$$

Substituting this into the BDF2 formulae, we have

$$(2-\gamma) y_{n+1} - (1-\gamma) h \lambda y_{n+1} = \frac{1}{\gamma} \frac{(2 - \gamma h \lambda)}{(2 + \gamma h \lambda)} y_n - \frac{(1-\gamma)^2}{\gamma} y_n \tag{3.11}$$

Setting $z = \lambda h$ we can simplify this expression for y^{n+1}

$$y_{n+1} = \frac{2(2-\gamma) - z(1 + (1-\gamma)^2)}{z^2(\gamma - 1)\gamma + z(-\gamma^2 + 4\gamma - 2) + 2(2-\gamma)} y_n \tag{3.12}$$

We see now that the amplification factor, $A(z)$ is given by

$$A(z) = \frac{2(2-\gamma) - z(1+(1-\gamma)^2)}{z^2(\gamma-1)\gamma + z(-\gamma^2+4\gamma-2) + 2(2-\gamma)} \tag{3.13}$$

and this clearly tends to zero, as $|z| \to \infty$. By definition then, the TR-BDF2 scheme is L-stable. We can plot the stability regions (which are outside the bounded areas) for different values of the parameter γ. We see that when $\gamma = 1$, we are essentially using the Trapezoidal scheme, hence our stability region is the left-half plane.

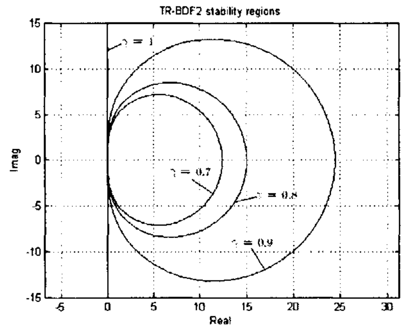


Figure 3-2: TR-BDF2 stability regions

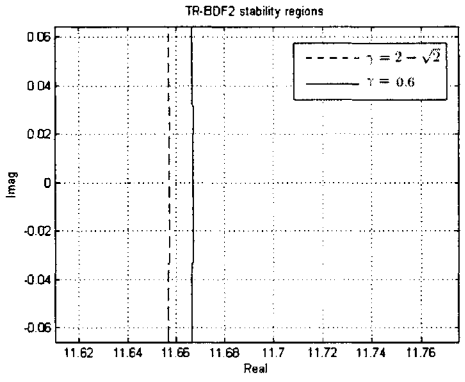


Figure 3-3: Magnified TR-BDF2 stability regions for $\gamma = 0.6$ and $\gamma = 2 - \sqrt{2}$

3.4 Integration schemes in commercial software

Given the numerous integration schemes that exist today, a natural question might be "which method(s) are *actually* being implemented in commercial software applications?" to solve ODE's which arise in industrial settings. Companies which specialize in circuit and structural analysis simulators clearly need accurate, robust solvers which will produce results in an acceptable amount of time. We briefly look at some of the industry-standard codes and which integration schemes they employ.

ADINA[2] (**A**utomatic **D**ynamic **I**ncremental **N**onlinear **A**nalysis) is one of the leading finite element analysis products available. It was founded in 1986 by Dr. Klaus-Jürgen Bathe, a professor at the Massachusetts Institute of Technology. Their software is currently being used for many applications [8]: at the California Department of Transportation in the design and testing of bridges for seismic analysis. ACE Controls, which manufactures shock absorbers (where linear deceleration is critical) for the automotive industry and theme parks chose ADINA from eleven other software providers. ADINA gives the user the option of using the TR-BDF2 integration scheme to solve fluid-structure, fluid flow and structural problems.

PISCES is another well known commercial piece of software, developed and maintained by the Integrated Circuits Laboratory at Stanford University. PISCES is a device modeling program which predicts component behavior in electrical circuits and is widely regarded [9] as "the industrial device simulator standard". Simucad, Synopsys and Crosslight sell commercial versions of the code which are used in the manufacturing process of integrated circuits as well as the perturbation analysis of individual components. Currently, six of the top ten electronics manufacturers [10] use code from Crosslight Software Inc.

PISCES gives the user the option of using either the TR-BDF2 scheme or the Backward Euler method. However the stepsize used is calculated automatically by

[2]The ADINA programs are developed in their entirety at ADINA R & D, Inc. and are the exclusive property of ADINA R & D, Inc.

the software and is dynamically adjusted during the simulation.

One might ask why we persistently use backwards differentiation schemes, given their natural dissipativeness. There are several reasons for this, the first being that these methods greatly reduce the number of functions evaluations required to compute a numerical solution. In [4], Gear describes in great detail the expense of these operations in circuit simulation. Secondly, we can very easily implement a higher order method without much extra cost.

The trapezoidal rule is not considered suitable for these applications because of its lack of damping. Since the trapezoidal rule conserves energy, oscillations are not damped at all (even the very high frequency ones which result from numerical noise) so errors propagate without being damped (see Figure 3.1). It is with this in mind that the TR-BDF2 scheme was constructed. With it, we can combine the advantages of each method: the energy conservation property of the trapezoidal rule as well as the damping properties of our backwards differentiation scheme.

Chapter 4

Numerical Examples

4.1 The Chaotic Pendulum

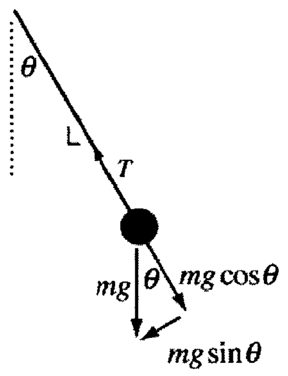


Figure 4-1: Free body force diagram for the simple pendulum

Throughout history, pendulums have had an exclusive role in the study of classical mechanics. In 1656, Christiaan Huygens invented the first pendulum clock which kept time accurately to an error of 10 seconds a day. In 1851, Jean Foucault used it to provide proof that the earth was rotating beneath us. Even today, pendulums serve as prime examples in the study of nonlinear (including chaotic) dynamics because of their natural oscillatory nature.

Under certain conditions, (namely a driving and damping force) we can make the simple pendulum exhibit chaotic behavior. By 'chaotic' we mean extreme sensitivity to initial conditions: the slightest of changes in how the system was initially configured can lead to large deviations in the long-term behavior of the system. The chaotic pendulum is an example of 'deterministic chaos' - meaning that at all times of the motion Newton's laws are obeyed. There is no randomness or stochastic element to the problem.

We begin by deriving the equation of motion for the damped, driven pendulum. When gravity is the only force present, the pendulum obeys

$$-mgl\ sin(\theta) = ml^2\ddot{\theta} \tag{4.1}$$

where θ is the angular displacement from the horizontal.

We then add a damping force $ml\beta\dot{\theta}$ and a periodic driving force $mlCsin(\omega_f t)$, so our equation of motion becomes

$$\frac{l}{g}\ddot{\theta} + \frac{\beta}{g}\dot{\theta} + sin(\theta) = \frac{C}{g}sin(\omega_f t) \tag{4.2}$$

We then rescale time by setting $t = s\sqrt{\frac{l}{g}}$. Finally we are left with

$$\ddot{\theta} + \mu\dot{\theta} + sin(\theta) = Fsin(\phi) \tag{4.3}$$

To apply our integration methods we write this as 3 first order differential equations:

$$\dot{\theta} = \omega \tag{4.4}$$

$$\dot{\omega} = -\mu\omega - sin\theta + Fsin(\phi) \tag{4.5}$$

$$\dot{\phi} = \omega_f \tag{4.6}$$

Here μ is the coefficient of damping and g is the amplitude of the (periodic) forcing. We now apply our usual trapezoidal step:

$$\theta_{t+\frac{1}{2}} - \frac{\Delta t}{2}\omega_{t+\frac{1}{2}} = \theta_t + \frac{\Delta t}{2}\omega_t \tag{4.7}$$

$$\omega_{t+\frac{1}{2}} - \frac{\Delta t}{2}\left(-\mu\omega_{t+\frac{1}{2}} - sin\theta_{t+\frac{1}{2}} + Fsin(\phi_{t+\frac{1}{2}})\right) = \omega_t + \frac{\Delta t}{2}\left(-\mu\dot{\theta}_t - sin\theta_t + Fsin(\phi_t)\right) \tag{4.8}$$

$$\phi_{t+\frac{1}{2}} - \frac{\Delta t}{2}\omega_f = \phi_t + \frac{\Delta t}{2}\omega_f \tag{4.9}$$

The *Jacobian matrix*, J is now found by taking partial derivatives of this nonlinear equation as described in the earlier section:

$$J = \frac{1}{2}\begin{pmatrix} 2 & -\Delta t & 0 \\ \Delta t\ cos(\theta) & 2+\mu\Delta t & -\Delta t F cos(\phi) \\ 0 & 0 & 2 \end{pmatrix} \tag{4.10}$$

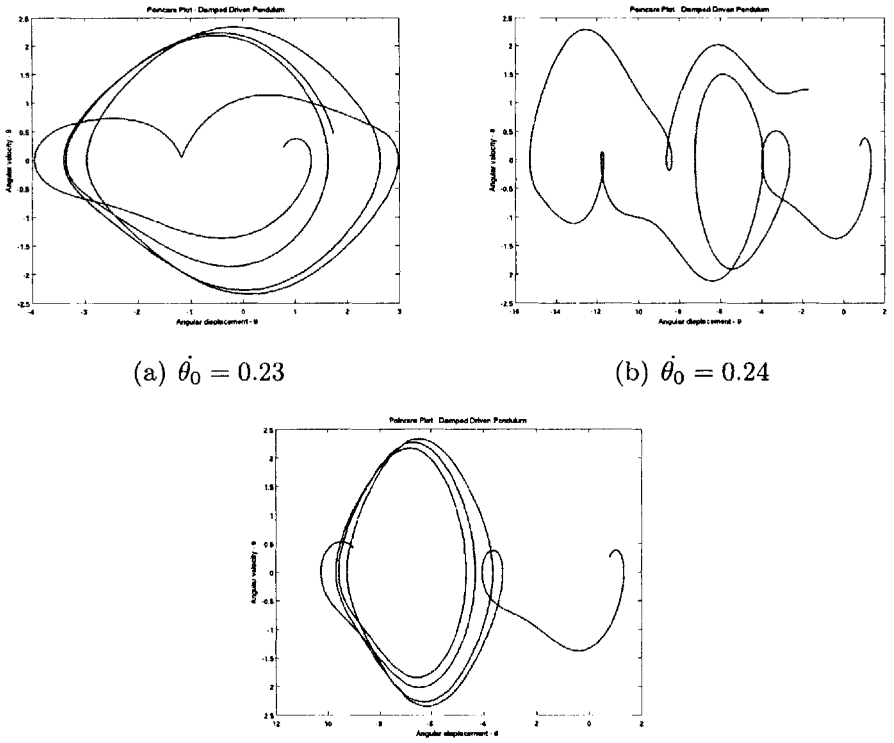
We use the scheme as above for the pure Trapezoidal scheme. For the TR-BDF2 method, we then use a 3 point backward Euler BDF2 step (given below) to evaluate the first derivative (which is the same operation for each unknown, U). The Jacobian for the BDF2 step (although a little more tedious) can be similarly derived.

$$\dot{U}_{t+1} = \frac{1}{\Delta t}U_t - \frac{4}{\Delta t}U_{t+\frac{1}{2}} + \frac{3}{\Delta t}U_{t+1} \tag{4.11}$$

Fortunately from the result in Section 3.3 we know we do *not* need to calculate the Jacobian matrix for the BDF2 step: by setting $\gamma = 2 - \sqrt{2}$ we ensure both matrices are the same, saving us an extra calculation.

In this section, with the intention of demonstrating the importance of choosing an appropriate numerical integration technique, we present the results of applying the Trapezoidal rule and the TR-BDF2 integration scheme to the chaotic (damped, driven) pendulum. Solutions were computed over an interval of 40 seconds. At each iteration, the coupled nonlinear equations were solved using the Newton-Raphson method to an accuracy of 10^{-8}.

The parameters selected for the following examples were intentionally chosen to lie in the chaotic regime of the motion. Note the Poincare plots for the pendulum, when the initial velocity is perturbed in increments of 0.01 -



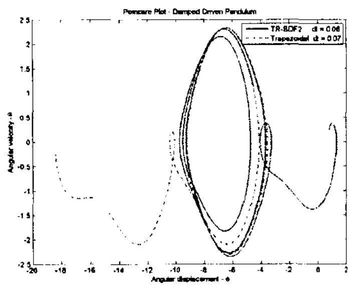
(a) $\dot{\theta}_0 = 0.23$

(b) $\dot{\theta}_0 = 0.24$

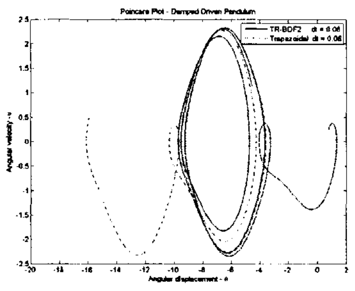
(c) $\dot{\theta}_0 = 0.25$

Figure 4-2: Chaotic behavior for the damped, driven pendulum with $\theta_0 = \frac{2\pi}{3}, \dot{\theta}_0 = [0.23, 0.24, 0.25], F = 1.18, \mu = \frac{1}{2}, \omega_f = \frac{2}{3}$ and $dt = 0.05$

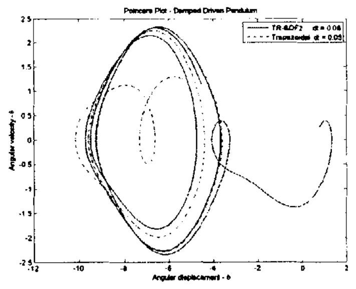
We now demonstrate the superior performance of the TR-BDF2 scheme over that of the Trapezoidal scheme for the last example given ($\theta_0 = \frac{2\pi}{3}, \dot{\theta}_0 = 0.25, F = 1.18, \mu = \frac{1}{2}$).



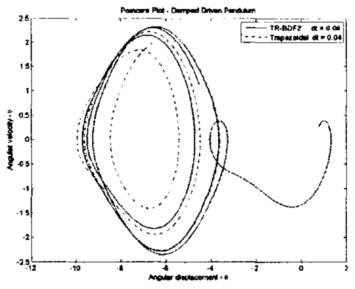
(a) Trapezoidal rule $dt = 0.07$



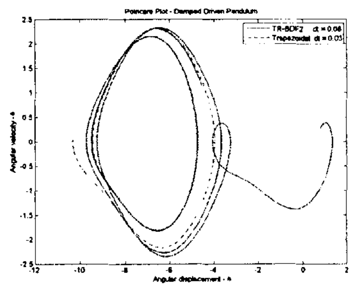
(b) Trapezoidal rule $dt = 0.06$



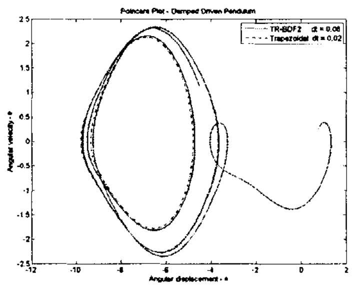
(c) Trapezoidal rule $dt = 0.05$



(d) Trapezoidal rule $dt = 0.04$



(e) Trapezoidal rule $dt = 0.03$



(f) Trapezoidal rule $dt = 0.02$

Figure 4-3: Higher accuracy of the TR-BDF2 scheme

A theoretical stability analysis of the model we have used for the chaotic pendulum would be very difficult to carry out because of the nonlinear nature of the coupled equations. Therefore we have to draw conclusions based on the results of our numerical simulation, since we cannot solve the model analytically.

Figure 4-3 is a very insightful one: with a stepsize of $dt = 0.05$ using the TR-BDF2 scheme, the numerical solution obtained is almost identical to that of the Trapezoidal scheme with a stepsize of $dt = 0.02$.

Several parameters were experimented with (F, μ, ω_f, dt as well as θ_0 and $\dot{\theta}_0$) but it proved difficult to find a parameter set which exposed the numerical instability of the Trapezoidal scheme (see Section 3.3). However our next example, the elastic pendulum, gives some compelling evidence to warrant our claim that purely A-stable methods should be implemented with caution: not just on the ground of 'greater accuracy', but based on the fact that they are genuinely more stable.

An animated movie of the chaotic pendulum with these parameters is available online from `http://www-math.mit.edu/18085/ChaoticPendulum.html` for both the Trapezoidal and the TR-BDF2 scheme. It can be seen that when the $\dot{r}$ and the $\dot{\theta}$ components were too high (relative to the selected time step) the numerical solution from the Trapezoidal rule blew up.

4.2 The Elastic Pendulum

The elastic pendulum is another model problem for us to use the ideas of numerical time stepping to predict evolution. The elastic pendulum, as the name might suggest is modeled as a heavy bob of mass m attached to a light elastic rod of length r and modulus of elasticity k which is allowed to stretch, but not bend. The system is free to oscillate in the vertical plane.

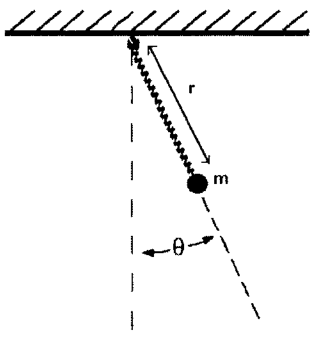


Figure 4-4: The elastic pendulum

Since the equations of motion are not as well known as the simple (inelastic) pendulum, we will derive them from Lagrange's equations. We begin by constructing the Lagrangian. The kinetic energy T is the sum of the transverse and radial kinetic energies of the bob:

$$T = \frac{m}{2}(\dot{r}^2 + r^2\dot{\theta}^2) \tag{4.12}$$

The potential energy, V is the sum of the elastic potential energy in the rod and the gravitational potential energy of the bob:

$$V = k(r - L)^2 - mgrcos\theta \tag{4.13}$$

So our Lagrangian L is then defined as $T - V$

$$L = \frac{m}{2}(\dot{r}^2 + r^2\dot{\theta}^2) - k(r - L)^2 + mgrcos\theta \tag{4.14}$$

The equations of motion are exactly the Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = 0 \tag{4.15}$$

where q and $\dot{q}$ are the state variables r and θ and their derivatives, respectively.

Hence the governing equations are:

$$\ddot{r} = gcos\theta - \frac{k}{m}(r - L) + r\dot{\theta}^2 \tag{4.16}$$

$$\ddot{\theta} = \frac{2\dot{r}\dot{\theta} + gsin\theta}{-r} \tag{4.17}$$

Linear analysis of this problem is very straightforward: we can integrate the governing equations of motion exactly using analytical techniques. However, this comes at a price because it is only when nonlinear effects are taken into account that we can accurately capture the coupling effects between the elastic oscillations in the rod and the rotational motion. We will see that solving the resulting (coupled) nonlinear equations has to be done numerically with an appropriate scheme (explicit methods are notoriously inefficient at solving systems of stiff nonlinear differential equations - see equation 3.1).

As described earlier, the trapezoidal rule can be used to time integration when we have a first order ordinary differential equation: hence we introduce new variables

$$R = \dot{r} \tag{4.18}$$

$$\Theta = \dot{\theta} \tag{4.19}$$

and we can then rewrite equations 4.16 and 4.17 as:

$$\begin{pmatrix} \theta \\ \Theta \\ r \\ R \end{pmatrix}' = \begin{pmatrix} \Theta \\ \frac{2\dot{r}\Theta + gsin\theta}{-r} \\ R \\ gcos\theta - \frac{k}{m}(r-L) + r\Theta^2 \end{pmatrix}$$

Now this is exactly in the form we require to use our trapezoidal integrator: comparing the above equation with an ODE in standard form:

$$\frac{dx}{dt} = f(x,t)$$

So substituting into the trapezoidal rule:

$$\frac{dx}{dt} = \frac{1}{2}(f'(x_n, t_n) + f'(x_{n+1}, t_{n+1}))$$

and using subscripts to denote the time at which we evaluate each term, we end up with:

$$U_{t+1} = U_t + \frac{\Delta t}{2}(V_t + V_{t+1})$$

We observe that now our variable x is identical to our state vector, U and $f(x,t)$ plays the role of our derivative vector, V.

$$U = \begin{pmatrix} \theta \\ \Theta \\ r \\ R \end{pmatrix}'$$

and

$$V = \begin{pmatrix} \Theta \\ \frac{2\dot{r}\Theta + gsin\theta}{-r} \\ R \\ gcos\theta - \frac{k}{m}(r-L) + r\Theta^2 \end{pmatrix}$$

What remains now is to illustrate how we use the Newton-Raphson process to solve this implicit equation for our new state vector, U_{t+1}.

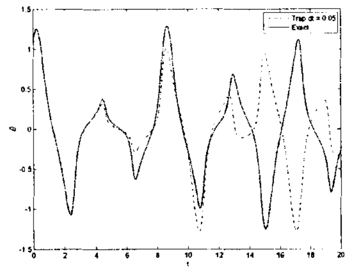
$$
\begin{aligned}
\theta_{t+1} - \frac{\Delta t}{2}\Theta_{t+1} &= \theta_t + \frac{\Delta t}{2}\Theta_t \\
\Theta_{t+1} - \frac{\Delta t}{2}\left(\frac{2R_{t+1}\Theta_{t+1} + gsin\theta_{t+1}}{-r_{t+1}}\right) &= \Theta_t + \frac{\Delta t}{2}\left(\frac{2R_t\Theta_t + gsin\theta_t}{-r_t}\right) \\
r_{t+1} - \frac{\Delta t}{2}R_{t+1} &= r_t + \frac{\Delta t}{2}R_t \\
R_{t+1} - \frac{\Delta t}{2}\left(gcos\theta_{t+1} - \frac{k}{m}(r_{t+1} - L) + r_{t+1}\Theta_{t+1}^2\right) &= R_t + \frac{\Delta t}{2}\left(gcos\theta_t - \frac{k}{m}(r_t - L) + r_t\Theta_t^2\right)
\end{aligned}
$$

The Jacobian matrix, J is again found by taking partial derivatives of these non-linear equations as described in the chapter on nonlinear solvers, Section 2.1

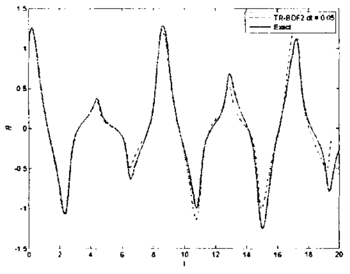
$$
J = \begin{pmatrix}
1 & -\frac{\Delta t}{2} & 0 & 0 \\
\frac{\Delta t}{2}\frac{gcos\theta}{r} & 1 + \frac{\Delta t}{2}R & -\frac{\Delta t}{2r^2}\left(2R\Theta + gsin\theta\right) & \frac{\Delta t}{r}\Theta \\
0 & 0 & 1 & -\frac{\Delta t}{2} \\
\frac{\Delta t}{2}\frac{gsin\theta}{r} & -R\Theta\Delta t & -\frac{\Delta t}{2}\Theta^2 + \frac{\Delta t}{2}\frac{k}{m} & 1
\end{pmatrix} \qquad (4.20)
$$

As mentioned in the previous example (the chaotic pendulum) the Jacobian matrix for the BDF2 step of the TR-BDF2 scheme is identical to that of the Trapezoidal rule when γ is set equal to $2 - \sqrt{2}$.

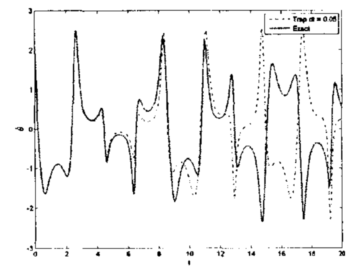
Now we present the results of applying the Trapezoidal rule and the TR-BDF2 integration scheme to a several instances of the elastic pendulum. The 'spring constant' is taken to be 10 (in non-dimensional units) for all of our examples. With each set of initial conditions, we solve for the 4 unknowns, $\theta, \dot{\theta}, r$ and $\dot{r}$ over an interval of 20 seconds. At each iteration, the nonlinear equations were solved using the Newton-Raphson method to an accuracy of 10^{-8}.



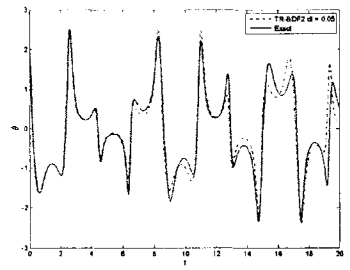
(a) θ evolution - Trapezoidal



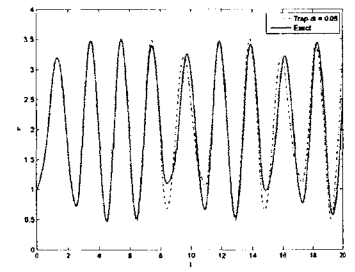
(b) θ evolution - TR-BDF2



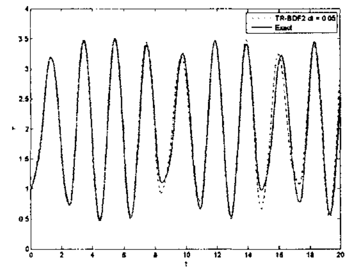
(c) $\dot{\theta}$ evolution - Trapezoidal



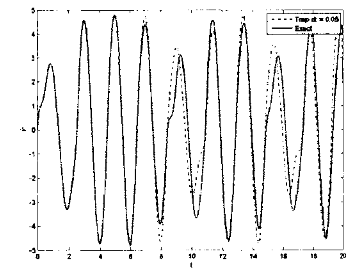
(d) $\dot{\theta}$ evolution - TR-BDF2



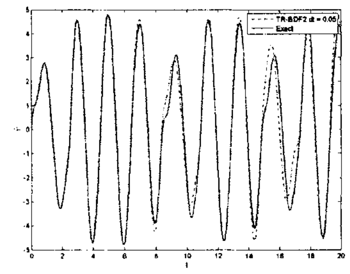
(e) r evolution - Trapezoidal



(f) r evolution - TR-BDF2

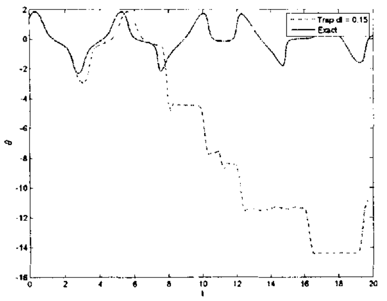


(g) $\dot{r}$ evolution - Trapezoidal

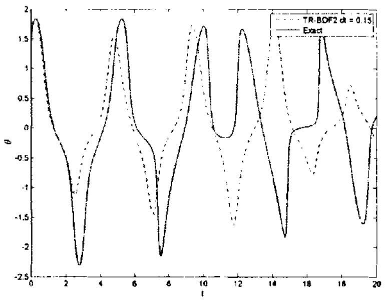


(h) $\dot{r}$ evolution - TR-BDF2

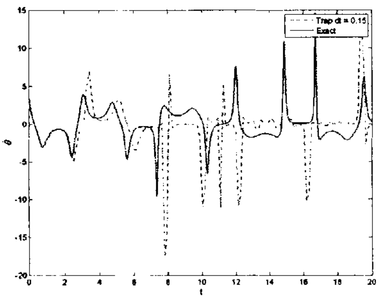
Figure 4-5: Plots for $\theta, \dot{\theta}, r$ and $\dot{r}$ evolution with the Trapezoidal and TR-BDF2 scheme with $\theta_0 = \frac{\pi}{3}, \dot{\theta}_0 = 2, r_0 = 1, \dot{r}_0 = 0, dt = 0.05$



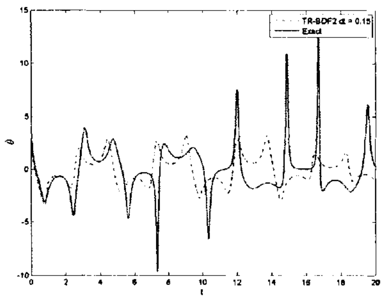
(a) θ evolution - Trapezoidal



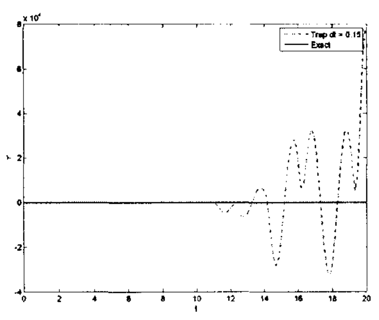
(b) θ evolution - TR-BDF2



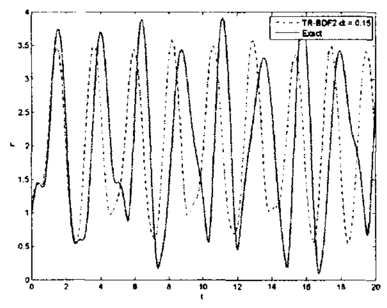
(c) $\dot{\theta}$ evolution - Trapezoidal



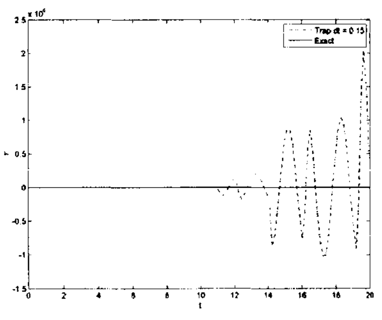
(d) $\dot{\theta}$ evolution - TR-BDF2



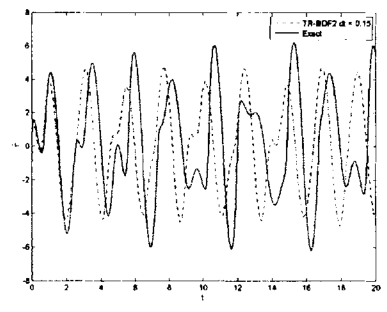
(e) r evolution - Trapezoidal



(f) r evolution - TR-BDF2

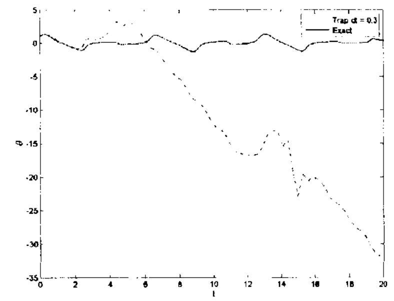


(g) $\dot{r}$ evolution - Trapezoidal

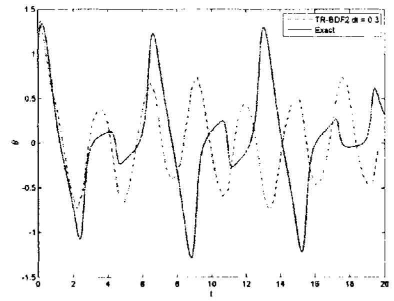


(h) $\dot{r}$ evolution - TR-BDF2

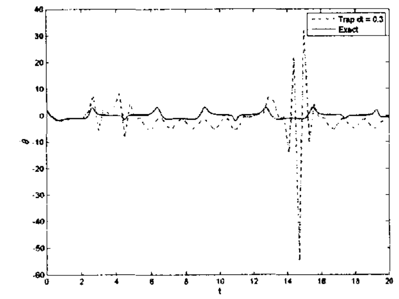
Figure 4-6: Plots for $\theta, \dot{\theta}, r$ and $\dot{r}$ evolution with the Trapezoidal and TR-BDF2 scheme with $\theta_0 = \frac{4\pi}{9}, \dot{\theta}_0 = 3.5, r_0 = 1, \dot{r}_0 = 0.8, dt = 0.15$



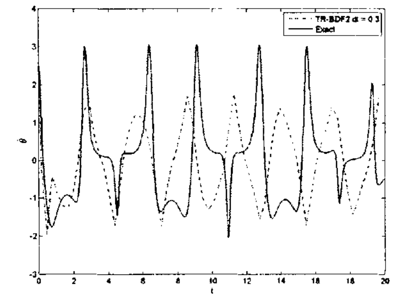
(a) θ evolution - Trapezoidal



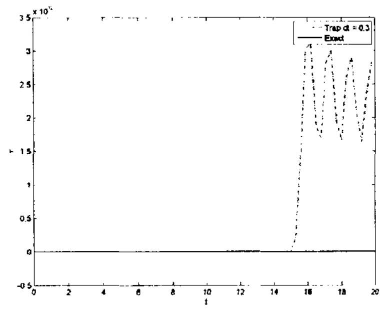
(b) θ evolution - TR-BDF2



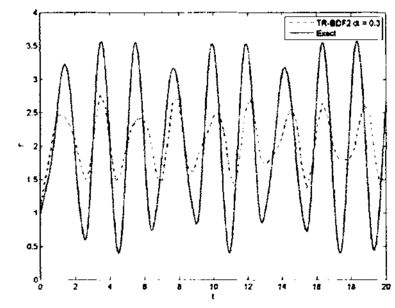
(c) $\dot{\theta}$ evolution - Trapezoidal



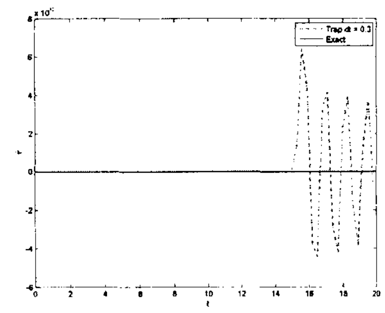
(d) $\dot{\theta}$ evolution - TR-BDF2



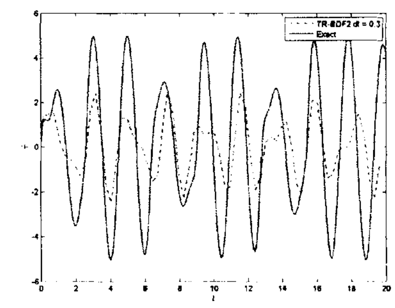
(e) r evolution - Trapezoidal



(f) r evolution - TR-BDF2



(g) $\dot{r}$ evolution - Trapezoidal



(h) $\dot{r}$ evolution - TR-BDF2

Figure 4-7: Plots for $\theta, \dot{\theta}, r$ and $\dot{r}$ evolution with the Trapezoidal and TR-BDF2 scheme with $\theta_0 = \frac{\pi}{3}, \dot{\theta}_0 = 2.5, r_0 = 1, \dot{r}_0 = 0.2, dt = 0.3$

In the first example, Figure 4-5 we verify in some sense what we already knew: that the TR-BDF2 scheme is more stable that the Trapezoidal rule for a given time step. This is especially clear from the plots of θ and $\dot{\theta}$ against time. The solution from the Trapezoidal rule rapidly diverges from the exact one around 13 seconds.

In the second example, Figure 4-6 with an increased time step $dt = 0.15$ we see the problem forewarned earlier: numerical errors lead to a blow up of the solutions obtained. Although this is not apparent from the plots of θ and $\dot{\theta}$, clearly the solutions for r and $\dot{r}$ indicate some non-physical behavior. Although the TR-BDF2 scheme is not very accurate with the larger time step at least we have stability.

In the third and last example, Figure 4-7 with an even larger time step $dt = 0.3$ the problem with the Trapezoidal scheme is even clearer - now the method does not conserve energy, even approximately. Again, the TR-BDF2 scheme is inaccurate with this large time step, but again stability is observed.

The data from these simulations was used to produce an animated movie (available from `http://www-math.mit.edu/18085/ElasticPendulum.html`) for both the Trapezoidal and the TR-BDF2 scheme. We note that when the $\dot{r}$ and the $\dot{\theta}$ components were too high (relative to the time step) the numerical solution from the Trapezoidal blew up.

4.3 The Double Pendulum

The double pendulum is a simple dynamical system which demonstrates complex behaviour. As the name suggests, it consists of one pendulum attached to the end of another and each pendulum is modeled as a point mass at the end of a light rod. Both pendulums are confined to oscillate in the same plane.

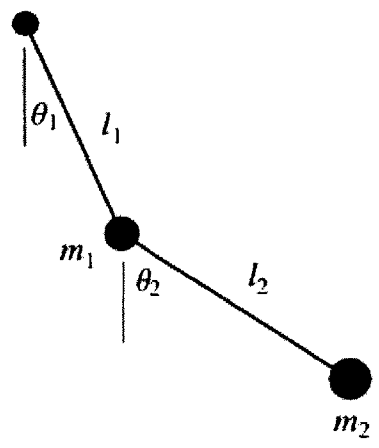


Figure 4-8: The double pendulum

We could derive the equations of motion in Cartesian coordinates, but this is somewhat tedious. Instead, as in the elastic pendulum we use polar coordinates to construct the Lagrangian, from which we will derive the Euler-Lagrange equations. Had we used cartesian coordinates, we would need to impose the constraint that the length of each rod is fixed:

$$\sqrt{x_i^2 + y_i^2} = l_i \tag{4.21}$$

However, we can forget about the constraints by choosing our coordinates appropriately:

$$x_1 = L_1 sin(\theta_1) \tag{4.22}$$

$$y_1 = L_1 cos(\theta_1) \tag{4.23}$$

$$x_2 = L_1 sin(\theta_1) + L_2 sin(\theta_2) \tag{4.24}$$

$$y_2 = L_1 cos(\theta_1) + L_2 cos(\theta_2) \tag{4.25}$$

Using these coordinates, we transform our unconstrained Lagrangian:

$$\tilde{L} = \frac{m_1}{2}(\dot{x}_1^2 + \dot{y}_1^2 - 2gy_1) + \frac{m_2}{2}(\dot{x}_2^2 + \dot{y}_2^2 - 2gy_2) \tag{4.26}$$

to the new Lagrangian, which captures the fact that the rod lengths are fixed:

$$L = \frac{m_1}{2} l_1^2 \dot{\theta_1}^2 + \frac{m_2}{2}(l_1^2 \dot{\theta_1}^2 + 2 l_1 l_2 cos(\theta_1 - \theta_2)\dot{\theta}_1\dot{\theta}_2 + l_2 \dot{\theta_2}^2) + m_1 g l_1 cos(\theta_1) + m_2 g(l_1 cos(\theta_1) + l_2 cos(\theta_2)) \tag{4.27}$$

These yield the two Euler-Lagrange equations we would have obtained had we substituted our constraint equations and eliminated the tensions in the rods:

$$\ddot{\theta_1} = \frac{-g(2m_1 + m_2) sin\theta_1 - m_2 g sin(\theta_1 - 2\theta_2) - 2 sin(\theta_1 - \theta_2) m_2 [\dot{\theta}_2^2 L_2 + \dot{\theta}_1^2 L_1 cos(\theta_1 - \theta_2)]}{L_1(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} \tag{4.28}$$

$$\ddot{\theta_2} = \frac{2 sin(\theta_1 - \theta_2)(\dot{\theta}_1^2 L_1 (m_1 + m_2) + g(m_1 + m_2) cos\theta_1 + \dot{\theta}_2^2 L_2 m_2 cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} \tag{4.29}$$

As described earlier, the trapezoidal rule can be used for the time integration when we have a system of first order ordinary differential equation: hence we introduce new variables

$$\omega_1 = \dot{\theta}_1 \tag{4.30}$$

$$\omega_2 = \dot{\theta}_2 \tag{4.31}$$

and we can then rewrite equations 4.28 and 4.29 as:

$$\begin{pmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \end{pmatrix}' = \begin{pmatrix} \omega_1 \\ \frac{-g(2m_1+m_2)sin\theta_1 - m_2 g sin(\theta_1-2\theta_2) - 2sin(\theta_1-\theta_2)m_2[\dot{\theta}_2^2 L_2 + \dot{\theta}_1^2 L_1 cos(\theta_1-\theta_2)]}{L_1(2m_1+m_2-m_2cos(2\theta_1-2\theta_2))} \\ \omega_2 \\ \frac{2sin(\theta_1-\theta_2)(\dot{\theta}_1^2 L_1(m_1+m_2)+g(m_1+m_2)cos\theta_1+\dot{\theta}_2^2 L_2 m_2 cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2cos(2\theta_1-2\theta_2))} \end{pmatrix}$$

Now this is exactly in the form we require to use our trapezoidal integrator: comparing the above equation with an ODE in standard form:

$$\frac{dx}{dt} = f(x,t)$$

We observe that now our variable x is identical to our state vector, U and $f(x,t)$ plays the role of our derivative vector, V.

$$U = \begin{pmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \end{pmatrix}'$$

and

$$V = \begin{pmatrix} \omega_1 \\ \frac{-g(2m_1+m_2)sin\theta_1 - m_2 g sin(\theta_1-2\theta_2) - 2sin(\theta_1-\theta_2)m_2[\dot{\theta}_2^2 L_2 + \dot{\theta}_1^2 L_1 cos(\theta_1-\theta_2)]}{L_1(2m_1+m_2-m_2cos(2\theta_1-2\theta_2))} \\ \omega_2 \\ \frac{2sin(\theta_1-\theta_2)(\dot{\theta}_1^2 L_1(m_1+m_2)+g(m_1+m_2)cos\theta_1+\dot{\theta}_2^2 L_2 m_2 cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2cos(2\theta_1-2\theta_2))} \end{pmatrix}$$

So substituting into the trapezoidal rule:

$$\frac{dx}{dt} = \frac{1}{2}(f'(x_n, t_n) + f'(x_{n+1}, t_{n+1}))$$

and using subscripts to denote the time at which we evaluate each term, we end up with:

$$U_{t+1} = U_t + \frac{\Delta t}{2}(V_t + V_{t+1})$$

What remains now is to use the Newton-Raphson process to solve this implicit equation for our new state vector, U_{t+1}. Clearly the Jacobian of this system is large and unsightly, so we avoid writing it out explicitly. Essentially, the steps are the same as those described in the earlier sections: construct the matrix by taking partial derivatives of the discretized equations.

Again, to implement the TR-BDF2 scheme we need to write out the backwards difference step and construct the Jacobian matrix but this is the same as the Jacobian for the Trapezoidal rule when γ is chosen to be $2 - \sqrt{2}$.

In this section we present the results of applying the Trapezoidal rule and the TR-BDF2 integration scheme to a sample double pendulum problem. With each set of initial conditions, solutions were computed over an interval of 30 seconds. As before, at each iteration, the nonlinear equations were solved using the Newton-Raphson method to an accuracy of 10^{-8}.

Test 1 - Initial data: $\theta_{10} = \theta_{20} = \frac{2\pi}{3}, \dot{\theta}_{10} = \dot{\theta}_{20} = 0, dt = 0.04$

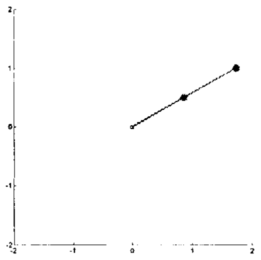
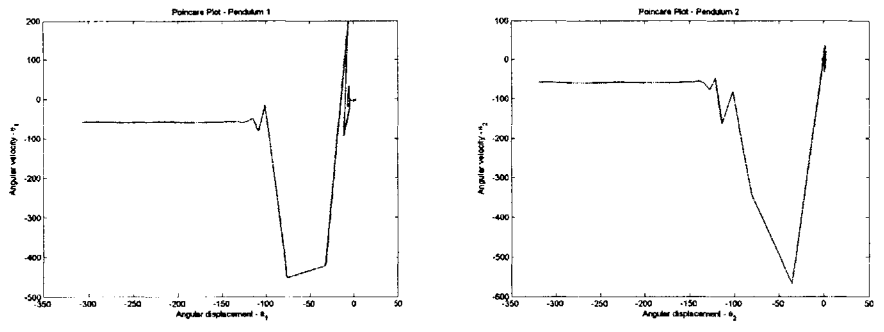
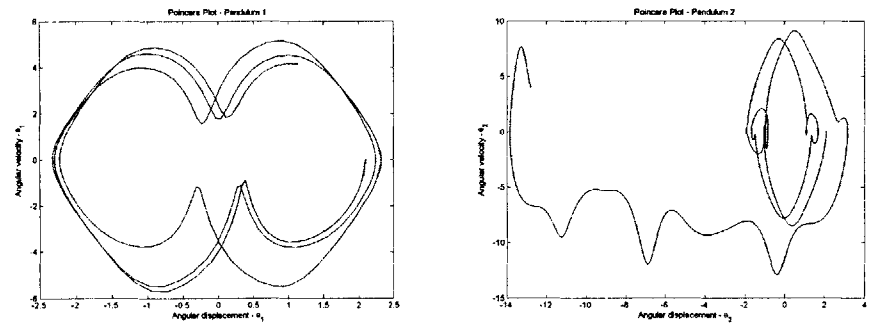


Figure 4-9: $\theta_{10} = \theta_{20} = \frac{2\pi}{3}, \dot{\theta}_{10} = \dot{\theta}_{20} = 0, dt = 0.04$



(a) Pendulum 1 - Trapezoidal scheme (b) Pendulum 2 - Trapezoidal scheme



(c) Pendulum 1 - TR-BDF2 scheme (d) Pendulum 2 - TR-BDF2 scheme

Figure 4-10: Poincare plots with the different methods

Test 2 - Initial data: $\theta_{10} = \theta_{20} = \frac{3\pi}{4}, \dot{\theta}_{10} = \dot{\theta}_{20} = 0, dt = 0.08$

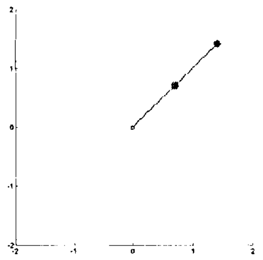
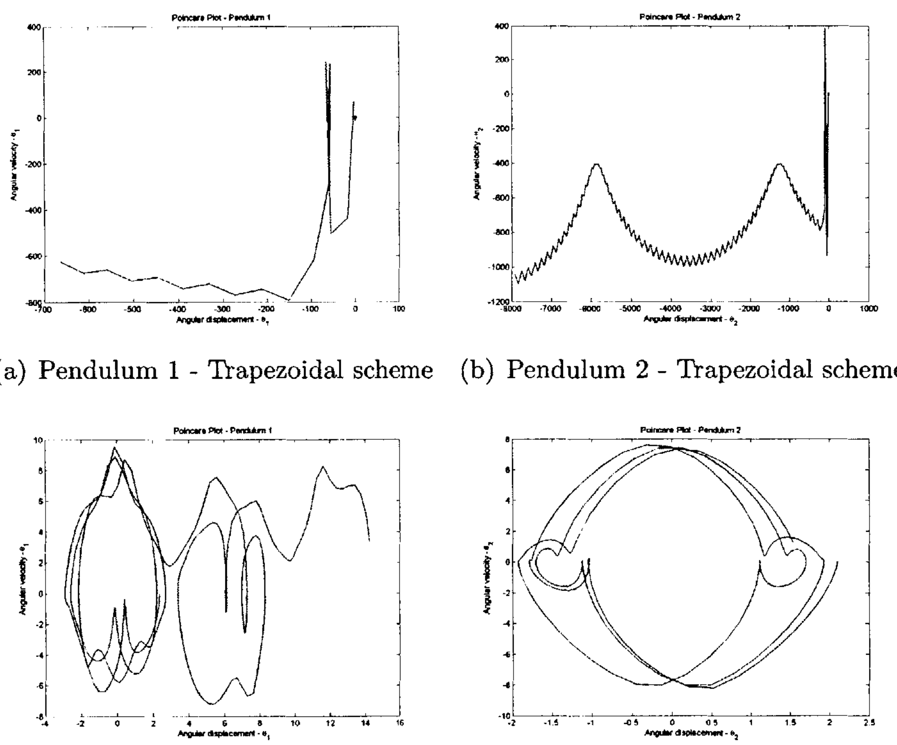


Figure 4-11: $\theta_{10} = \theta_{20} = \frac{3\pi}{4}, \dot{\theta}_{10} = \dot{\theta}_{20} = 0, dt = 0.04$



(a) Pendulum 1 - Trapezoidal scheme (b) Pendulum 2 - Trapezoidal scheme

(c) Pendulum 1 - TR-BDF2 scheme (d) Pendulum 2 - TR-BDF2 scheme

Figure 4-12: Poincare plots with the different methods

In all of the examples considered we have chosen our initial conditions close to the unstable equilibrium position of the system: the inverted configuration. The exact solutions have *not* been overlaid on these plots, because for the TR-BDF2 numerical solution to even slightly resemble the exact one, a time step of $dt = 0.005$ must be used, which does not expose the inherent weakness of the Trapezoidal rule. However, once again as seen from the previous example of the elastic pendulum, stability was apparent only with the TR-BDF2 scheme.

The resulting data was used to produce an animated movie (available from `http://www-math.mit.edu/18085/DoublePendulum.html`) for both the Trapezoidal and the TR-BDF2 scheme. We see from these animations that when the $\dot{\theta}_1$ and the $\dot{\theta}_2$ components were too high (relative to the time step) the numerical solution from the Trapezoidal rule blew up.

In the first example, Figure 4-10, the initial condition was such that the upper pendulum did not do a flip. The resulting motion was quite well resolved (no apparent discontinuities) despite the somewhat large time step of $dt = 0.04$ while the Trapezoidal rule blew up almost immediately (similarly for pendulum two).

Similar results were obtained for Figure 4-12, an initial condition that was even closer to the unstable equilibrium position, with an even larger time step of $dt = 0.08$. It is worth noting that now we see a spurious 'ringing[2]' effect for the second pendulum in the case of the Trapezoidal rule. Since at each iteration, Newton-Raphson iterations were performed to solve the nonlinear equations to an accuracy of 10^{-8}, this meant we were doing an excessive amount of computation to compute a solution that is clearly wrong. Again, despite the larger time step (and hence larger inaccuracy) the TR-BDF2 scheme remained stable unlike the Trapezoidal rule. Simulations confirmed that even larger values of dt were possible (up to 0.17) without stability being an issue.

[2]Although there is no technical definition for ringing, in this case it implies a banded oscillation of our numerical solution.

4.4 The Advection-Reaction-Diffusion Equation

The advection-reaction-diffusion (ADR) equation is a very significant one in nature. It can be used to model complex systems where quantities of interest can move around while simultaneously reacting with each other. There are numerous examples of such systems: the concentration of bacteria in a flowing culture solution, or tracking a pollutant released into the atmosphere (the former will be the the study of this section).

The partial differential equation for such this process is given by

$$\frac{\partial c}{\partial t} + \nabla \cdot (\mathbf{u}c) = \nabla \cdot (D\nabla c) + R(c), \quad c = c(\mathbf{u}, t) \tag{4.32}$$

Individually, advection, reaction and diffusion constitute some of the most basic processes in nature. Examples are numerous and for clarity we highlight some of the major ones, since an understanding of the physical significance of these effects will be crucial to interpreting the results later in this section.

Advection

Advection describes the transport of a material in a fluid medium - concentration of chemical released into a river for example. The transport of the chemical is dominated by the flowing water, as opposed to diffusion (spreading out along a concentration gradient). For a scalar quantity c, (here it is the concentration of the chemical) this can be represented as

$$\frac{\partial c}{\partial t} + \nabla \cdot (\mathbf{u}c) = 0 \tag{4.33}$$

Reaction

The reaction term expresses an interaction between two or more variables of interest - for example, bacteria in a culture solution, feeding on a nutrient supply. If b represents the bacteria density at a point (a scalar) and c represents the concentration of the nutrient, this can be represented as

$$\frac{\partial c}{\partial t} = -bc \tag{4.34}$$

Diffusion

A diffusive term describes how matter moves from a region of high concentration, to a region of lower concentration. The classic example is how the heat profile u of a point source evolves in time. In 3 dimensions for an isotropic, homogeneous medium with parameter κ (dependent on the density, specific heat capacity and the conductivity of the medium)

$$\frac{\partial u}{\partial t} = \kappa \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \tag{4.35}$$

A simplified model

In this section for simplicity we consider the following coupled ADR model with the minimum number of terms:

$$\frac{\partial b}{\partial t} = D_b \nabla^2 b - \nabla(b \nabla c) \tag{4.36}$$

$$\frac{\partial c}{\partial t} = D_c \nabla^2 c - bc \tag{4.37}$$

A scenario where such a model would be relevant might be in a bacteria-culture system. The density of bacteria b, can diffuse with diffusivity constant D_b while simultaneously being advected along a concentration gradient determined by the concentration of the nutrients c. The nutrients also diffuse, with diffusivity constant D_c but instead of being advected along like the bacteria, the reaction term $-bc$ implies nutrients are *consumed* by the bacteria.

One way of solving this system of equations would be to consider all the terms at once: discretize the equations using a difference scheme to replace all the derivative terms and then simply march forward in time. This simple approach has its drawbacks however: an implicit method would probably need to be used because of the nature of the diffusive terms which would otherwise impose a severely small (impractical) time step restrictions.

Another reason why we may have no choice but to use an implicit scheme could be the nature of the reaction term. If these terms are "stiff" (meaning the process being modeled evolves in time on a much quicker scale than the others) then once again we have to drastically reduce our time step to make sure our numerical solution remains stable. Despite the bleak setup, there exists a very clever technique known as "operator splitting" to overcome these difficulties which we will employ to solve our ADR model in an efficient manner.

Splitting methods

The observation that almost all partial differential equations (PDEs) which evolve in time, split additively into (possible coupled) subprocesses led to the notion of operator splitting/time splitting. These subprocesses are in general simpler PDEs than our original one and the key idea of operator splitting is that we can step these individual subprocesses in time (while maintaining an order of appearance) to time march the solution of our original PDE.

The basis of these methods have been well developed over the last few years [11] and a formal analysis would be beyond the scope of this thesis - for our needs, it suffices to show how we can use these ideas to efficiently compute solutions of the advection-diffusion-reaction.

The underlying strength of splitting theory is that we can apply *fractional step methods* to our problem - we can decouple the processes involved and apply a numerical technique we deem most suitable to each one. Given some of the excellent explicit methods available for solving hyperbolic PDEs (together with their associated conservation laws) we can see how the splitting scheme will be advantageous over the naïve approach described earlier (where we directly discretized and used finite differences, then marched in time). For example, we will be able to treat the advective subprocess very accurately (and rapidly) using a high order explicit method, such as Fromm's scheme[1].

Another nice "bonus" of this method, is that is that given our full ADR PDE system,

$$\mathbf{u}_t = A(\mathbf{u}) + D(\mathbf{u}) + R(\mathbf{u}) \tag{4.38}$$

where A, D and R represent the advective, diffusive and reaction terms respectively, we can guarantee a second order accurate solution given that each subproblem is solved using a second order method, by using the symmetric Strang splitting approach [12].

$$\mathbf{u}^{n+1} = A(\Delta t/2)R(\Delta t/2)D(\Delta t)R(\Delta t/2)A(\Delta t/2)\mathbf{u}^n \tag{4.39}$$

We now proceed to discuss briefly the numerical method of choice for each subprocess and present some motivating results regarding a stability issue which arose as a result of using the standard Crank-Nicolson to discretize the diffusion step. This difficulty was resolved using the TR-BDF2 scheme and we discuss how and why this method performs better than the Crank-Nicolson method, which we be an automatic choice for most.

[1]Fromm's scheme can be thought of as the average of the Lax-Wendroff and Beam-Warming schemes.

The Advection Step

The advection subprocess to be solved is

$$\frac{\partial b}{\partial t} = -\nabla(b\nabla c)$$

$$\frac{\partial b}{\partial t} = -(ub)_x - (vb)_y$$

where the velocity field is given by

$$u = \frac{\partial c}{\partial x} \tag{4.40}$$

$$v = \frac{\partial c}{\partial y} \tag{4.41}$$

For the advection step, we use the explicit Fromm scheme which is second order accurate in space and time. As mentioned earlier, this can be regarded as the average of the well known Lax-Wendroff and Beam-Warming schemes. For the time discretization, we use a simple forwards difference and for the spacial discretization we use a second order upwind biased scheme:

$$\frac{\partial b}{\partial t} = \frac{b_i^n - b_i^{n-1}}{\Delta t}$$

$$\frac{\partial b}{\partial x} = \frac{-b_{i-2}^n + 5b_{i-1}^n - 3b_i^n - b_{i+1}^n}{4\Delta x}$$

The Reaction Step

To solve the reaction subprocess

$$\frac{\partial c}{\partial t} = -bc \tag{4.42}$$

we use an explicit method of second order (as mentioned earlier, each subprocess must be solved with a second order method to ensure the final solution is second order accurate). We consider a simple Taylor method:

$$c^{n+1} = c^n + \Delta t \left[\frac{\partial c}{\partial t}\right] + (\Delta t)^2 \left[\frac{\partial^2 c}{\partial t^2}\right] \tag{4.43}$$

The Diffusion Step

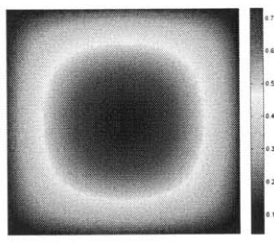
The diffusion equations to be solved are

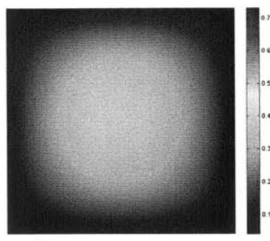
$$\frac{\partial b}{\partial t} = D_b \nabla^2 b \tag{4.44}$$

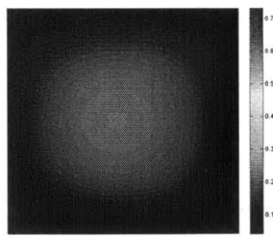
$$\frac{\partial c}{\partial t} = D_c \nabla^2 c \tag{4.45}$$

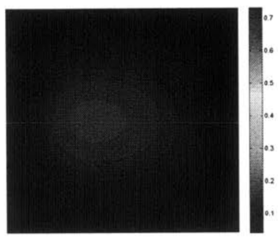
These are just the linear diffusion equations for the bacteria concentration, b and the concentration of the culture solution, c. As briefly discussed earlier, an immediate candidate method for these equations might be the second order Crank-Nicolson scheme (since it is based on the Trapezoidal rule). However, after implementing this, the simulations[2] developed oscillations that were were amplified greatly with time. This instability was more apparent as we increased diffusivity ratios. The result on the next page is the bacteria concentration using $D_b = 0.05$, $D_c = 0.01$, $\Delta t = 0.1$ and $\Delta x = 0.05$.

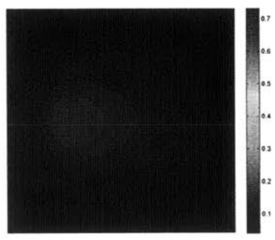
[2]The initial conditions used for both the test cases was a parabolic profile for the bacteria concentration b and a culture concentration such that $\frac{\partial c}{\partial x} = \frac{\partial c}{\partial y} = 1$ everywhere.

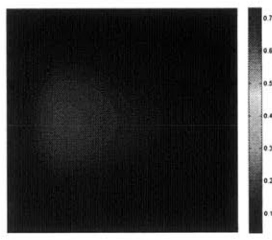

(a) t = 0

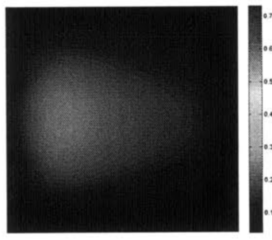

(b) t = 0.1

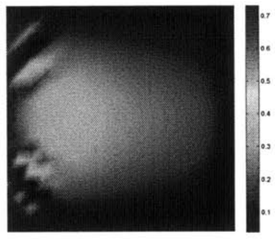

(c) t = 0.2


(d) t = 0.3


(e) t = 0.4


(f) t = 0.5


(g) t = 0.6


(h) t = 0.7

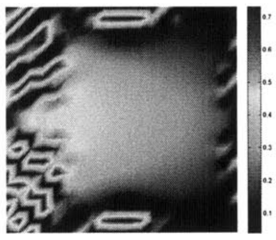

(i) t = 0.8

Figure 4-13: The bacteria concentration b, using the Crank-Nicolson scheme for the diffusion subprocess of the Strang splitting. Simulation parameters: $\Delta t = 0.1$, $\Delta x = 0.05$, $D_b = 0.05$, $D_c = 0.01$

The TR-BDF2 fix

It was seen that the Crank-Nicolson scheme did not adequately damp the oscillations. The only way to prevent the instability from growing was to use Courant numbers less than about 0.05. With such a restriction, the simulation ran extremely slowly.

The instability observed was due to the fact that the Trapezoidal rule is *not* L-stable so the oscillations from the nonlinear interaction of the advection and reaction steps were not sufficiently damped by the Crank-Nicolson diffusion step. This was rectified by implementing the the L-stable TR-BDF2 scheme to provide the necessary damping.

The analysis of the TR-BDF2 scheme was presented in an earlier section, but as a reminder, the scheme is essentially a 2-step Runge-Kutta method. The first step is done using the Trapezoidal rule, to obtain $u^{t+\Delta t/2}$, then applying the 2-step BDF to obtain $u^{t+\Delta t}$.

$$u^{t+\Delta t/2} = u^t + \frac{\Delta t}{4}\left[R(u^t) + R(u^{t+\Delta t/2})\right]$$

$$u^{t+\Delta t} = \frac{1}{3}\left[u^t + 4u^{t+\Delta t/2} + \Delta t R(u^{t+\Delta t})\right]$$

where $R(u,t)$ is our differential operator.

With this correction, we were able to simulate the model for Courant numbers very close to 1, without oscillation amplification. The screenshots on the following page are for the same simulation as before, but this time using the TR-BDF2 scheme for the diffusion step.

We were able to obtain a steady state solution to our ADR model, which was not possible earlier with the Crank-Nicolson diffusion.

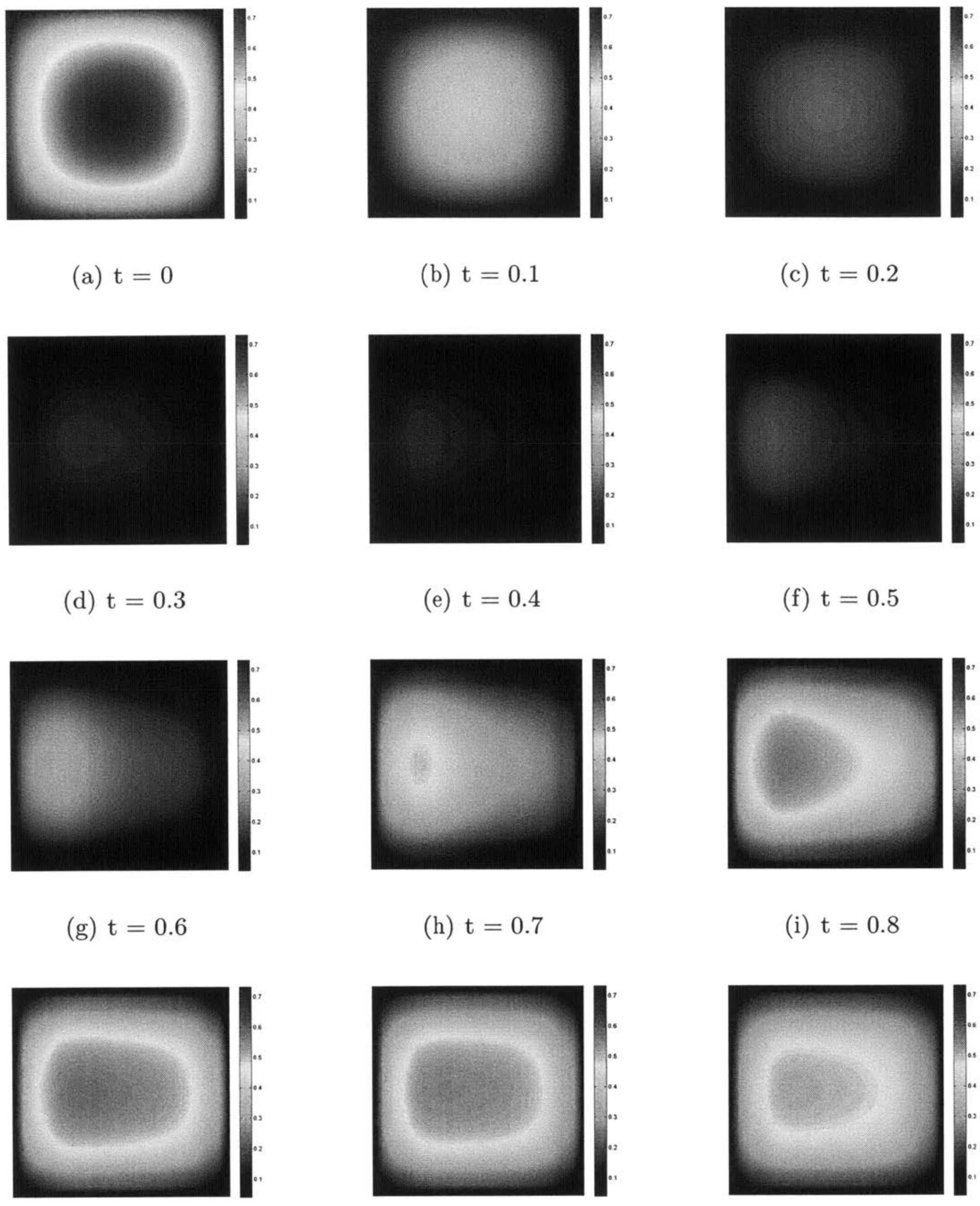


Figure 4-14: The bacteria concentration b, using the TR-BDF2 scheme for the diffusion subprocess of the Strang splitting. Simulation parameters: $\Delta t = 0.1$, $\Delta x = 0.05$, $D_b = 0.05$, $D_c = 0.01$

Conclusion

In this section we briefly presented an important application of operator splitting to solving the advection-diffusion-reaction equation for a typical model which arises in computational biology. With the splitting technique, we were not restricted to the small time step predicted by the diffusion subprocess but instead were able to apply of more state-of-the-art methods to the constituent subprocesses, namely the explicit Fromm scheme for the advection step of the ADR equations.

We were also able to demonstrate that the popular Crank-Nicolson method should be used with caution: the numerical damping provided by the method was insufficient to damp oscillations arising from certain initial conditions, while the 2-stage TR-BDF2 scheme did not present such a issue. This was due to the fact that the TR-BDF2 scheme was L-stable, while the Trapezoidal rule was not.

Chapter 5

Conclusions

As seen from the results in this section, purely A-stable integrators (of which the most well known one is probably the Trapezoidal rule) should be used with caution. A-stability of a method guarantees us *in theory* that the stability region consists of the left-half plane, but in practice because of round-off errors and possible linearization, the boundary of our stability region may not be well defined. Eigenvalues very close to the imaginary axis on the left hand side, may lead to amplification of certain modes of our solution, which may not be sufficiently damped.

The numerical examples considered demonstrate that the TR-BDF2 scheme (with the value of $\gamma = 2 - \sqrt{2}$) was efficient and overcame the difficulties encountered with the Trapezoidal rule alone. We were able to compute solutions with a reasonable value of the minimum allowable stepsize for stiff problems. As we saw in the case of the ADR equation, the oscillations produced as a result of numerical error were efficiently damped with the TR-BDF2 scheme and we were able to calculate a steady-state solution (which was not possible with the Trapezoidal rule, unless we severely restricted our stepsize).

It is only fair to mention a possible drawback of the L-stable TR-BDF2 scheme examined in this thesis: for the model problem we examined, $y' = \lambda y$ we proved that the region of convergence of the TR-BDF2 method exactly coincided with the stable parameter region of the problem. For a general problem though, this may not be the case. Essentially, because of the BDF2 step, the method dampens modes that fall into the region of convergence. So it may be difficult to determine if a system is genuinely oscillating or stably amplifying the modes in its solution.

As mentioned earlier, all the animations and source codes used to produce the images in this thesis are available online at `http://www-math.mit.edu/18085/` and it is hoped that the reader has gained a deeper appreciation of the thought that must be put into choosing an appropriate integration method: sometimes the naïve choices (Crank-Nicolson and the Trapezoidal rule) are not always the best, despite being extremely popular and easy to implement.

Bibliography

[1] A. Valdimirescu *The SPICE Book* Wiley, 1994, 412pp.

[2] R. E. Bank, W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose & R. K. Smith *Transient simulation of silicon devices and circuits. IEEE Trans. Comput. Aided Des.* CAD-4, 436451.

[3] J. D. Lambert *Numerical Methods for Ordinary Differential Systems* Wiley, 1991.

[4] C. W. Gear *Simultaneous numerical solution of differential-algebraic equations* IEEE Trans. Circ. Theory CT 18, 1971.

[5] Fredebeul, C *A-BDF: A Generalization of the Backward Differentiation Formulae* SIAM Journal on Numerical Analysis, Vol. 35, No. 5. (Oct., 1998), pp. 1917-1938.

[6] J. C. Butcher *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods* Wiley, 1987.

[7] B. Lindberg *On smoothing and extrapolation for the Trapezoidal Rule* BIT 11, 1971.

[8] `http://www.adina.com/applic/applic.shtml`

[9] `http://www.silvaco.com/tech_lib/simulationstandard/1992/apr/a1/a1.html`

[10] `http://strategis.ic.gc.ca/app/ccc/search/navigate.do?language=eng&portal=1&subPortal=&estblmntNo=123456266820&profile=completeProfile`

[11] B. Sportisse *An Analysis of Operator Splitting Techniques in the Stiff Case* Journal of Computational Physics, **161**, 140-168.

[12] G. Strang *On the construction and comparison of difference schemes* SIAM J. Numer. Anal., **5**, No. 3, pp. 506-517, 1968.