Comparing Multistep Methods Within Parametric Classes to Determine Viability in Solver Applications

Michael Kraut

There are many colorful and densely inked graphs in this paper.
Please consider sparing your printing resources by viewing it digitally.

Contents

Abstract

In the 2017 paper by Arévalo and Söderlind [2] a framework was established for creating linear multistep methods of various classes based on a polynomial formulation which includes variable step size adaptivity. These classes are: k step explicit and implicit methods of order k and $k+1$ respectively for nonstiff problems (such as Adams methods), and k step implicit methods of order k for stiff problems (such as BDF methods). For each method class and order, all multistep methods of maximal order, including those which lack zero stability, are given by a parametrization depending on the method class and order. In this paper we conduct a pre-study on low order methods, comparing the properties of methods of the same class and order, and present experimental results when these methods are applied to simple test problems. We are motivated by the possibility of using method changes as a primary means of error control in solvers alongside traditional error control tools such as step size variability and order control. This paper also discusses some of the difficulties encountered during the research and concludes with questions for future study.

Chapter 1

Introduction

The study of ordinary differential equations (ODEs) and initial value problems (IVPs) has been around since at least the 1700s. IVPs arise naturally in many different fields of science and engineering and understanding how to solve them is necessary for the development of these subjects. However, most IVPs have no readily available theoretical solution. Therefore it is important to develop numerical methods that can give a "sufficiently good" approximation of the solution.

The simplest numerical method for solving IVPs is the Explicit Euler method, dating back to the 1768 work by Leonhard Euler [7]. This is a *time-stepping scheme* where solution values are approximated at grid points in an iterative fashion, using data from the previous time point to approximate the solution value at the next. In order to work, this procedure only requires the ODE and initial value, as well as a choice of step size. In theory, this is the only information that should be required, since Picard's Theorem states that any IVP satisfying some mild continuity requirements will have a unique solution [20, Thm 12.1]. However, since this is a numerical scheme, some amount of error is incurred in each step, and so mathematicians have sought to develop methods to improve the quality of the results. The two main types of numerical methods are *Runge-Kutta methods*, which take *stage derivatives* between the grid points in order to approximate the average rate of change more accurately, and *linear multistep methods*, which use some of the previous time steps to more accurately interpolate the objective function. This paper focuses on the latter.

There is a distinction to be made between a numerical *method* and a numerical *solver*. A numerical *method* is simply a formula for approximating the solution at the next time point. However, this approximation can vary in quality, depending on the size of the step, the quality of the previous estimates, and the properties of the IVP itself. Thus, it is necessary to develop numerical *solvers*, which attempt to take an *error estimate* at each step, and use various heuristic schemes to control this error.

Since the choice of step size is somewhat free, most solvers use dynamic step size changes to control the quality of the approximation. It is important

to point out the difference between *variable step size multistep methods* which can account for non equidistant grids, and *solvers using fixed step-size methods* which can vary the grids. A solver using the former can change the step size, in any step if desired, with some degree of liberty, while a solver using the latter implements a fixed step size method locally, then must perform some type of re-gridding and interpolation when a step size change is deemed necessary. An overview of how step size control works is presented in [3].

It is also common for solvers to perform order control, see [9, 13, 17]. This means selecting a *different* method of a different order, although it may be a method from the same *family*, such as Adams or BDF. Since the goal is usually to maximize the step size, therefore minimizing the total number of steps and work to be done, while maintaining a desired degree of accuracy, the typical scheme for order control is: when the solver is ready to consider an order change, it will simulate the next step with a method of one order higher, and a method of one order lower. Then the solver will estimate errors for all methods (current, higher, and lower order) and determine the next step size for each. Whichever method is able to take the largest step is chosen.

While a solver implementing a variable step size multistep method can change step size continuously, many solvers choose to keep the step size fixed until a change is deemed necessary, for example see [13, 17]. This usually happens when the error estimate becomes too large or sufficiently small, but it can also be triggered by issues in other parts of the integration, such as a nonlinear iteration failing to converge. It has been observed that this scheme can lead to unpredictable work-accuracy results: sometimes, a small change in solver parameters or IVP properties can lead to unpredictable changes in total steps and performance, see [23, 24]. Furthermore, method changes are usually carried out only in the form of order changes, which are therefore discrete. Thus a solver will in practice only ever implement a small handful of methods.

Until recently, variable step size multistep methods were limited to specific implementations, usually Adams-like implicit methods for nonstiff problems and BDF-like implicit methods for stiff problems, see [4, 5, 11, 15, 17]. However, the new work by Arévalo and Söderlind in [2] generalizes this idea to allow construction of variable step size implementations of all relevant multistep methods of maximal order for a given method class.

The purpose of this paper is to study these new classes of methods and determine some basic properties about them in order to motivate the possibility of using method changes through a continuous parametrized class as a primary means of error control. In Chapter 2 we present the mathematical background of linear multistep methods and describe the basic properties of solvers which utilize them. In Chapter 3 we introduce the parametrized classes of methods and discuss the MODES software by Arévalo et al., a Matlab software pack implementing these methods for the solution of IVPs. In Chapter 4 we present properties of several low order method classes and experimental results of these methods applied to test problems. Finally in Chapter 5 we discuss some of the difficulties which arose during the course of this research and present questions for future study.

Chapter 2

Background

2.1 Linear Multistep Methods

An *ordinary differential equation* (ODE) is an equation of the form

$$\dot{y} = f(t, y). \tag{2.1}$$

An *initial value problem* (IVP) is a problem of finding a differentiable function y satisfying an ODE and initial value of the form

$$\dot{y} = f(t, y), \qquad y(t_0) = y_0, \qquad t \in [t_0, t_\mathrm{f}]. \tag{2.2}$$

With some simple restrictions on the function f, the IVP will have a unique solution [20, Thm. 12.1].

A *linear multistep method* (LMM) is a type of formula used to compute a sequence of pointwise approximations $\{y_n\}_{n=0}^N$ to the solution of (2.2) where $y_n \approx y(t_n)$ is an approximation to the solution at each time point and $\{t_n\}_{n=0}^N$ is some gridding of the interval $[t_0, t_\mathrm{f}]$. An LMM uses k previous estimates and their function evaluations to compute an estimate for the next time point. In this case it is called a k step method. If $t_n - t_{n-1} = h$ for all n and some fixed h, we say that the method is *fixed step size*, in which case it is usually represented as

$$\sum_{i=0}^{k} \alpha_{k-i} y_{n-i} - h \sum_{i=0}^{k} \beta_{k-i} y'_{n-i} = 0 \tag{2.3}$$

For variable step size methods, $h_{n-i-1} = t_{n-i} - t_{n-i-1}$ will be inside the sum and the α and β coefficients will depend on the previous k step size ratios h_{n-i}/h_{n-i-1}. Here $y'_i = f(t_i, y_i)$ is a sample of the vector field defining the ODE. This is different from the time derivative of the solution, since the numerical solution often does not lie on a single trajectory. Note that the indexing convention chosen here may differ from other literature. The polynomials

defined by

$$\rho(z) = \sum_{j=0}^{k} \alpha_j z^j \tag{2.4}$$

$$\sigma(z) = \sum_{j=0}^{k} \beta_j z^j. \tag{2.5}$$

are called the *characteristic polynomials* of the method and will be useful for analysis.

The normalization $\alpha_k = 1$ is often used. If $\beta_k = 0$ the method is called *explicit*. In this case, the values $y_{n-k}, \ldots, y_{n-1}$ and $y'_{n-k}, \ldots, y'_{n-1}$ are "plugged in" to the equation to solve for y_n in each step. If $\beta_k \neq 0$ then the method is called *implicit*, in which case a nonlinear equation for y_n must be solved in each step since y'_n depends on y_n. This is often solved using *fixed-point iteration* or *Newton iteration* [6]. Some solvers like CVODE have several choices of nonlinear solvers and many options for their parameters [13].

2.1.1 Error

Obviously it is important that the values $\{y_n\}$ returned by the method are a reasonable approximation to the function $y(t)$, the actual solution to the IVP. To measure how good of an approximation it is, we measure the *error*. For a given method and problem, this is formulated as a function of the (newly calculated) time point (t_n, y_n) and the step size $h_n = t_n - t_{n-1}$. There are three different types of errors:

- **Global Error -** $E_n = |y_n - y(t_n)|$ is the difference between the computed solution and the exact solution at point t_n. Ideally, the goal is to minimize this, however if the exact solution is not known, this is impossible to evaluate. Therefore the global error can only be used as a research tool to study the quality of numerical methods on IVPs with known solutions.
- **Local Error -** $L_n = |y_n - \tilde{y}(t_n)|$ where $\tilde{y}$ is the solution to the IVP $\dot{y} = f(t, y),\ y(t_{n-1}) = y_{n-1}$. This means the error at the newly computed point, assuming the solution at the last point was exact. Since the global error is an accumulation of local errors, the global error can in theory be controlled by controlling the local errors.
- **Local Truncation Error -** $T_n = |y_n - \tilde{y}(t_n)|$ where $\tilde{y}$ is as before, but now y_n is the result of applying the k-step LMM to the points $\{y_{n-k}, \ldots, y_{n-1}\}$. The distinction is that since the points are themselves approximations, they often do not lie on the same solution curve. Therefore this is the error of the LMM assuming that the last k points are all exact. This error can be estimated, and so it is the most useful quantity to consider when discussing error control. Therefore throughout the rest of this paper, when discussing error, the local truncation error will always be used unless specified otherwise.

Sometimes *error per unit step* is used instead and these values are divided by h_n.

2.1.2 Order of Consistency

The following derivations are from [20, Ch. 12] but multiplied through by h to use error instead of error per unit step. Alternate derivations can be found in [6, Ch 4.1.4], [9, Ch 8.1] or [10, Ch III.2]. A method of form (2.3) is said to be *consistent of order* p if p is the largest positive integer such that, for any sufficiently smooth solution $y(t)$ of (2.2),

$$T_n = \mathcal{O}(h^{p+1}) \tag{2.6}$$

for the truncation error produced by the method at any point t_n. This is equivalent to the method being exact whenever the solution $y(t)$ is a polynomial of at most degree p. A method is said to be *consistent* if it is consistent of at least order one. To determine the order of a method, we write the truncation error (neglecting absolute value) as

$$T_n = y_n - \tilde{y}(t_n) \tag{2.7a}$$

$$= \frac{\sum_{j=0}^{k} \alpha_j y(t_{n-k} + jh) - h \sum_{j=0}^{k} \beta_j \dot{y}(t_{n-k} + jh)}{\sum_{j=0}^{k} \beta_j}. \tag{2.7b}$$

Note that the indices are now reversed, since the next step is to expand this formula in a Taylor series around the point t_{n-k}, the furthest point back. This results in a series of the form

$$T_n = \frac{C_0 y(t_{n-k}) + C_1 h\dot{y}(t_{n-k}) + C_2 h^2 \ddot{y}(t_{n-k}) + \cdots}{\sum_{j=0}^{k} \beta_j}. \tag{2.8}$$

The constants C_q can now be given explicitly as

$$C_0 = \sum_{j=0}^{k} \alpha_j, \tag{2.9a}$$

$$C_1 = \sum_{j=0}^{k} j\alpha_j - \beta_j, \tag{2.9b}$$

$$C_q = \frac{1}{q!} \sum_{j=0}^{k} j^q \alpha_j - q j^{q-1} \beta_j. \tag{2.9c}$$

Then it is clear that a method is of order p if and only if $C_q = 0$ for $q = 0, \ldots, p$ and $C_{p+1} \neq 0$. This means that for a method to be consistent, it must be the case that $C_0 = C_1 = 0$, and therefore $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$ $(\neq 0)$.

Since $\sum_{j=0}^{k} \beta_j = \sigma(1)$, the error can be written as

$$T_n = \frac{C_{p+1}}{\sigma(1)} h^{p+1} y^{(p+1)}(t_{n-k}) + \mathcal{O}(h^{p+2}) \tag{2.10}$$

The constant $\frac{C_{p+1}}{\sigma(1)}$ is called the *error coefficient of order p*. For an order p method, this is normally just called the "error coefficient" or "error constant." We make a slight distinction here since we will typically be dealing with k step methods of orders depending on k: for explicit methods, $p = k$, and for implicit methods $p = k$ or $p = k+1$. However there will be a few instances where methods which are expected to have a certain order will attain higher order, and in these situations we would like to consider these methods as having zero error coefficient for their "designated" order, while having some other error coefficient for the higher order. Although this coefficient can be negative depending on how it is defined, we will always refer to its absolute value.

2.1.3 Stability

A method is said to be stable if small perturbations in starting values do not cause the results of the numerical method to change without bound. Formally, a method is said to be *stable* if for every IVP (2.2), there exists a constant K such that if $(x_0, \ldots, x_{k-1})$ and $(y_0, \ldots, y_{k-1})$ are two different sequences of starting values generating the sequences $(x_n), (y_n)$, then

$$|x_n - y_n| \leq K \cdot \max\{|x_0 - y_0|, \ldots, |x_{k-1} - y_{k-1}|\} \tag{2.11}$$

for $t_n \leq t_{\mathrm{f}}$ and K not depending on h [20, Def. 12.3].

This condition is known as *zero stability* since it is sufficient to prove that it holds for the trivial equation $y' = 0$. Fortunately, there is an easy way to check whether or not a method is zero stable. The following theorem is known as the *Root Condition*

Theorem 2.1.1. *A linear multistep method is zero stable if and only if the roots of the characteristic polynomial $\rho(z)$ all have magnitude less than or equal to one, with those having magnitude one being simple roots.*

A proof of this theorem can be found in [20, Thm. 12.4]. Since $\rho(1) = 0$ for consistent methods, there is always at least one root of magnitude one. A method is called *strongly stable* if this is the only such root, and *weakly stable* if there is at least one other (simple) root of magnitude one. In this paper, when referring to stable methods we always mean strongly stable, except when stated otherwise. We will see in the next section that zero stability is extremely important.

Another important concept is the *stability region* of a method. This is determined by applying the method to the Linear Test Equation $y' = \lambda y$ with step size h. Doing so produces the linear recurrence relation

$$\sum_{i=0}^{k} \alpha_{k-i} y_{n-i} - h\lambda \sum_{i=0}^{k} \beta_{k-i} y_{n-i} = 0, \tag{2.12}$$

which can be written as

$$AY_n = Y_{n+1}, \tag{2.13}$$

where Y_n is the vector of points, $Y_n = (y_n, \ldots, y_{n-k})^T$. Then the top row of this matrix is $(\alpha_k - h\lambda\beta_k, \ldots, \alpha_0 - h\lambda\beta_0)$ and everything underneath is a shift matrix consisting of ones below the main diagonal and zeros elsewhere. This matrix has characteristic polynomial

$$\pi(z; h\lambda) = \rho(z) - h\lambda\sigma(z) = 0 \tag{2.14}$$

and therefore the iteration is stable precisely when this polynomial satisfies the Root Condition. The set of $h\lambda \in \mathbb{C}$ for which the method is stable is called the *stability region* of the method. A method which is stable in the entire negative half plane is called *A-stable*. Zero stability is equivalent to zero being in the stability region of the method.

2.1.4 Convergence

A method is said to be convergent if the *global* error at every point in the approximation goes to zero as the step size decreases. This means that the approximation begins to approach the true solution as the step size decreases and more grid points are used.

The *Dahlquist Equivalence Theorem* says that a method is convergent if and only if it is both consistent and stable [20, Th. 12.5]. Furthermore, a method which is consistent of order p has global error $E_n = \mathcal{O}(h^p)$.

The *Dahlquist Barrier Theorem* states that the maximum order a convergent k step method can attain is:

$$\begin{cases} k & \text{for explicit methods} \\ k+1 & \text{for implicit methods with } k \text{ odd} \\ k+2 & \text{for implicit methods with } k \text{ even.} \end{cases} \tag{2.15}$$

A proof of this theorem can be found in [10, p. 385].

2.2 Solver Properties

While a multistep method gives a scheme for computing a sequence of pointwise approximations to the solution of a differential equation, a *solver* is a standalone algorithm whose job is to return "good" approximations to IVPs. What makes for a "good" solution involves several components and tradeoffs must be made to balance each consideration. We present here some of the fundamental properties that a solver should satisfy.

2.2.1 Error Control

Fundamentally, the job of a solver is to return a solution which is an accurate approximation to the solution of the IVP. It is important that the user should be able to expect the global error to be sufficiently small. To do this, the solver must take as input a specified tolerance `TOL`. The solver then *estimates* the

truncation error and then makes adjustments to the integration in an attempt to keep this error reasonably close to `TOL`. Although the solver may not be able to actually keep the truncation error less than `TOL`, it is usually the case that the global error can be kept approximately proportional to it, and therefore by changing the tolerance by a factor of M, the global error will also change by approximately a factor of M. This principle is called *tolerance proportionality.* The purpose of tolerance proportionality is to give the user the choice between a faster, less accurate solution, or a slower, more accurate solution.

Usually, the primary control scheme is the step size. If the estimated error is too large, the solver will reduce the step size in order to reduce the error. Similarly if the estimated error is sufficiently small, the solver will increase the step size in order to speed up the computation.

Another common control scheme is order control, which allows the solver to select a higher or lower order method, if such a change can produce better results. Furthermore, if there is a problem at any step, such as the error controller suggestic a drastic step size change, or a nonlinear convergence failure, the order of the method can be reduced to one in an attempt to overcome problematic areas in the IVP.

2.2.2 Stability

Generally speaking, a stable solver is one which is able to return a "well-behaved" solution to a "well-behaved" problem. This means that small deviations encountered during the integration process should not cause large deviations in the results. This requires all parts of the solver to work together: for example, the solver must use stable methods in order to be convergent, but it must also choose step size sequences which are within the stability regions of the methods. It should also be the case that a small change in the step size sequence does not cause a large change in the solution. This could occur if the starting values are slightly changed, or if a different error controller is chosen.

However, stability clearly depends on the properties of the IVP itself. For example, if a step size is chosen which is a multiple of an oscillatory frequency in the problem, then the solver will encounter *resonance* and this component of the solution will grow instead of oscillating. It is the job of the solver to try to avoid these pitfalls wherever possible.

It is extremely important that solvers are stable since errors will be incurred during the integration process. If a small error can lead to a large change in the accuracy of the solution, then the results of the solver are unreliable.

2.2.3 Computational Efficiency

It is important for a solver to be able to perform its task with a reasonable amount of computational resources. If a solver is more efficient, it can produce higher quality results faster. Even assuming software is running on a "modern" computer, the solution of an IVP can take milliseconds or days, depending on the difficulty of the problem and the efficiency of the solver. It is also usually

important for a solver to be able to halt computation when the workload is too much. This is usually done by checking if a maximum number of steps has been reached, or if a designated amount of time has passed. A solver may also give up if results become too large. This may be the result of instability or a convergence failure. These limits are clearly situational and therefore must be controllable by the user.

2.2.4 Stiff Tolerance

The concept of *stiffness* has multiple definitions. One definition is the presence of factors which decay many orders of magnitude faster than the time scale of the problem. Stiff problems challenge the stability and efficiency of solvers. Not all methods nor all solvers are suitable for stiff problems, but stiff problems arise in real life scenarios and so stiff tolerant solvers are necessary for practical computations.

Chapter 3

Trigonometric Parametrization of Methods

Following the construction in [2], we build k-step methods as polynomial methods. This means that in a typical step from t_{n-1} to t_n we construct a polynomial P_n which approximates the solution point by

$$y_n = P_n(t_n). \tag{3.1}$$

A polynomial of degree p will define a method of at most order p. By the Barrier Theorem, the degree of a k step method is limited, and so they will be constructed to achieve at most order $k+1$ if the method is implicit, or order k if the method is explicit. Therefore instead of interpolating at all previous points we leave a "slack" at a few of them characterized in the following way:

For given sequences $\{(t_{n-i}, y_{n-i})\}_{i=1}^{k}$ and $\{y'_{n-i}\} = \{f(t_{n-i}, y_{n-i})\}$, and polynomial P_n, the *state slack* s_{n-i} and *derivative slack* s'_{n-i} at t_{n-i} are defined by

$$s_{n-i} = P_n(t_{n-i}) - y_{n-i}, \qquad s'_{n-i} = \dot{P}_n(t_{n-i}) - y'_{n-i}. \tag{3.2}$$

For a given θ_{k-i}, the polynomial P_n is said to satisfy the *slack-balance condition* at t_{n-i} if

$$s_{n-i}\cos(\theta_{k-i}) + h_{n-i}s'_{n-i}\sin(\theta_{k-i}) = 0 \tag{3.3}$$

where $h_{n-i} = t_{n+1-i} - t_{n-i}$

In addition, we have the *interpolation condition* $s_{n-1} = 0$ and the *explicit collocation condition* $s'_{n-1} = 0$, as well as the *implicit collocation condition* $\dot{P}_n(t_n) = f(t_n, P_n(t_n))$, which can be considered as $s'_n = 0$.

These conditions can be used to describe one step methods. The unique one step explicit method is the Explicit Euler method, where the (linear) polynomial used to determine the next point satisfies the interpolation and explicit collocation conditions. By allowing the method to be implicit, the order can be raised to two by requiring the polynomial to also satisfy the implicit collocation

condition: this is the Trapezoidal Rule. By removing the order two requirement, the interpolation and explicit collocation conditions can be replaced by a slack-balance condition, giving the one step implicit Theta Methods. When the slack-balance condition degenerates to only an interpolation condition, the Implicit Euler method is obtained.

Interestingly, all higher order methods can be constructed from this template by adding slack-balance conditions to the points $t_{n-2}, ..., t_{n-k}$. This leads to the definitions of the following three classes of multistep methods:

- E_k **methods -** Explicit methods of at least order k, $n_\theta = k - 1$. The polynomial P_n is uniquely determined by the conditions:

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-i}\cos(\theta_{k-i}) + h_{n-i}s'_{n-i}\sin(\theta_{k-i}) = 0 \quad i = 2, \ldots, k \end{cases} \tag{3.4}$$

- I_k **methods -** Implicit methods of at least order k, $n_\theta = k$. The polynomial P_n is uniquely determined by the conditions:

$$\begin{cases} \dot{P}_n(t_n) = f(t_n, P_n(t_n)) \\ s_{n-1}\cos(\theta_{k-1}) + h_{n-1}s'_{n-1}\sin(\theta_{k-1}) = 0 \\ s_{n-i}\cos(\theta_{k-i}) + h_{n-i}s'_{n-i}\sin(\theta_{k-i}) = 0 \qquad i = 2, \ldots, k \end{cases} \tag{3.5}$$

- I_k^+ **methods -** Implicit methods of at least order $k + 1$, $n_\theta = k - 1$. The polynomial P_n is uniquely determined by the conditions:

$$\begin{cases} \dot{P}_n(t_n) = f(t_n, P_n(t_n)) \\ s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-i}\cos(\theta_{k-i}) + h_{n-i}s'_{n-i}\sin(\theta_{k-i}) = 0 \quad i = 2, \ldots, k \end{cases} \tag{3.6}$$

In practice, θ_{k-i} can be chosen in whichever interval of length π is most convenient. In this paper, we choose $\theta_{k-i} \in [0, \pi]$, with redundancy at the endpoints.

In [2] it is shown that all I_k^+ methods are at least order $k + 1$ and proofs that methods from the other two classes are at least order k can be constructed similarly. However, although all methods achieve at least the designated order, not all methods generated in this fashion are zero stable.

3.1 Parametric Equivalence Theorem

In [1, Thm 2.1] the following theorem is given:

Theorem 3.1.1 (Parametric Equivalence)**.** *For a particular method with polynomial formulation* (3.4)*,* (3.5)*, or* (3.6)*, and constant step-size formula* (2.3)*, the following parametric relations hold:*

$$\tan(\theta_j) = -\frac{\beta_j}{\alpha_j}, \qquad j = 0, \ldots, K, \tag{3.7}$$

with $K = k - 2$ for E_k and I_k^+ methods, or $K = k - 1$ for I_k methods.

Thus, given a k-step method of proper order in fixed step size form (2.3), one can determine the method class and θ parametrization using this theorem. To go the other way however is slightly more complicated; given a method class and θ vector, one must solve the consistency equations to determine the fixed step size formula. This means setting $C_0 = \cdots = C_p$ in Equation (2.9) and solving for the α and β coefficients, which is a system of size $p + 1$.

3.2 Common Methods

There are several common multistep methods which have been developed and used in commercial solver software. The following derivations are detailed in [8], other good references include [10, 14, 18]. We use the notation $f_{n-i} = f(t_{n-i}, y_{n-i})$ to denote function evaluations of previous time points, whether computed via a numerical scheme or given as initial data.

3.2.1 One Step Methods

Discussions of multistep methods usually begin with the Explicit and Implicit Euler methods. The *Explicit Euler Method* is given by the formula:

$$y_n = y_{n-1} + hf_{n-1}. \tag{3.8}$$

This is a one step explicit method, which only uses the approximation at the previous time point and its function value to calculate the next point. Replacing the function value at the previous point with the function value at the next point, we obtain the *Implicit Euler Method*:

$$y_n = y_{n-1} + hf_n. \tag{3.9}$$

As this is now an implicit method, a nonlinear system must be solved in each step for y_n. As mentioned before, this is usually done through an iterative method. The Implicit and Explicit Euler methods are both order one methods. However, by taking an average of function values one can obtain the *Trapezoidal Rule:*

$$y_n = y_{n-1} + \frac{h}{2}(f_{n-1} + f_n). \tag{3.10}$$

The Trapezoidal Rule is the unique one step method of order two.

3.2.2 Adams-Bashforth Methods

Moving on to methods which are more traditionally considered "multistep," we have the family of explicit *Adams-Bashforth* methods. These are derived by writing (2.1) as an integral from t_{n-1},

$$y(t) = y_{n-1} + \int_{t_{n-1}}^{t} f(\tau, y(\tau))d\tau, \tag{3.11}$$

and then substituting a polynomial approximation for the integrand. To calculate a given y_n, we assume the previous k approximations $\{y_{n-k}, \ldots, y_{n-1}\}$ and time points $\{t_{n-k}, \ldots, t_{n-1}\}$ are given. We can then calculate the respective function values $\{f_{n-k}, \ldots, f_{n-1}\}$ and form the unique interpolating polynomial π_{k-1} of degree $k-1$ with the property $\pi_{k-1}(t_{n-i}) = f_{n-i}$ for $i = 1, \ldots, k$. We write this polynomial using the *Lagrange basis polynomials*, defined by

$$l^i_{k-1}(t) = \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{t - t_{n-j}}{t_{n-i} - t_{n-j}}, \tag{3.12}$$

with the property $l^i_{k-1}(t_{n-j}) = \delta_{ij}$ (Kronecker delta), so that π_{k-1} can be written as

$$\pi_{k-1}(t) = \sum_{i=1}^{k} l^i_{k-1}(t) f_{n-i}. \tag{3.13}$$

Putting this approximation into Equation (3.11), we have:

$$y(t) \approx y_{n-1} + \sum_{i=1}^{k} \left(\int_{t_{n-1}}^{t} l^i_{k-1}(t) \right) f_{n-i}. \tag{3.14}$$

The right hand term is a degree k polynomial which we evaluate at t_n to approximate y_n, so it is of the form (3.1). This polynomial satisfies

$$P_n(t_{n-1}) = y_{n-1}, \tag{3.15a}$$

$$\dot{P}_n(t_{n-1}) = f_{n-1}, \tag{3.15b}$$

$$\dot{P}_n(t_{n-i}) = f_{n-i}, \qquad i = 2, \ldots, k, \tag{3.15c}$$

and therefore the Adams-Bashforth methods are the family of E_k methods (3.4) with $\theta_{k-i} = \pi/2$ for $i = 2, \ldots, k$.

For a fixed step size $h = t_{n+1-i} - t_{n-i}$ the basis polynomials can be integrated directly to give the formulas for the *Adams-Bashforth k step Methods* (AB-k).

Below are the formulas for $1 \leq k \leq 5$:

$$y_n = y_{n-1} + hf_{n-1} \qquad \text{(Explicit Euler)} \tag{3.16a}$$

$$y_n = y_{n-1} + h\left(\frac{3}{2}f_{n-1} - \frac{1}{2}f_{n-2}\right) \tag{3.16b}$$

$$y_n = y_{n-1} + h\left(\frac{23}{12}f_{n-1} - \frac{16}{12}f_{n-2} + \frac{5}{12}f_{n-3}\right) \tag{3.16c}$$

$$y_n = y_{n-1} + h\left(\frac{55}{24}f_{n-1} - \frac{59}{24}f_{n-2} + \frac{37}{24}f_{n-3} - \frac{9}{24}f_{n-4}\right) \tag{3.16d}$$

$$y_n = y_{n-1} + h\left(\frac{1901}{720}f_{n-1} - \frac{2774}{720}f_{n-2} + \frac{2616}{720}f_{n-3} - \frac{1274}{720}f_{n-4} + \frac{251}{720}f_{n-5}\right). \tag{3.16e}$$

Since the Adams-Bashforth methods are the unique k step explicit methods of order k with $\alpha = (1, -1, 0, \ldots, 0)$, the θ vector can also be determined by the Parametric Equivalence Theorem 3.1.1.

3.2.3 Adams-Moulton Methods

By removing the explicitness requirement and including t_n as an interpolating point, we can raise the order of the method by one to obtain the *Adams-Moulton* methods. We now include t_n in the basis polynomials as

$$l_k^i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{t - t_{n-j}}{t_{n-i} - t_{n-j}} \tag{3.17}$$

and similarly define π_k as

$$\pi_k(t) = \sum_{i=0}^{k} l_k^i(t) f_{n-i}. \tag{3.18}$$

Similar to Equation (3.14), the basis polynomials can be integrated to create the formula

$$y(t) \approx y_{n-1} + \sum_{i=0}^{k} \left(\int_{t_{n-1}}^{t} l_k^i(t)\right) f_{n-i}. \tag{3.19}$$

This is now an implicit scheme and so the formula for y_n must be solved iteratively. The right hand side is a degree $k+1$ polynomial of the form (3.1) satisfying

$$\dot{P}_n(t_n) = y_n, \tag{3.20a}$$

$$P_n(t_{n-1}) = y_{n-1}, \tag{3.20b}$$

$$\dot{P}_n(t_{n-1}) = f_{n-1}, \tag{3.20c}$$

$$\dot{P}_n(t_{n-i}) = f_{n-i}, \qquad i = 2, \ldots, k, \tag{3.20d}$$

and therefore the Adams-Moulton methods are the family of I_k^+ methods (3.6) with $\theta_{k-i} = \pi/2$ for $i = 2, \ldots, k$.

For a fixed step size, we obtain the *Adams-Moulton k step methods* (AM-k). Below are the formulas for $1 \leq k \leq 4$:

$$y_n = y_{n-1} + h\left(\frac{1}{2}f_n - \frac{1}{2}f_{n-1}\right) \qquad \text{(Trapezoidal Rule)} \tag{3.21a}$$

$$y_n = y_{n-1} + h\left(\frac{5}{12}f_n + \frac{8}{12}f_{n-1} - \frac{1}{12}f_{n-2}\right) \tag{3.21b}$$

$$y_n = y_{n-1} + h\left(\frac{9}{24}f_n + \frac{19}{24}f_{n-1} - \frac{5}{24}f_{n-2} + \frac{1}{24}f_{n-3}\right) \tag{3.21c}$$

$$y_n = y_{n-1} + h\left(\frac{251}{720}f_n + \frac{646}{720}f_{n-1} - \frac{264}{720}f_{n-2} + \frac{106}{720}f_{n-3} - \frac{19}{720}f_{n-4}\right). \tag{3.21d}$$

In an identical situation to the Adams-Bashforth methods, the Adams-Moulton methods are the unique k step implicit methods of order $k+1$ with $\alpha = (1, -1, 0, \ldots, 0)$, and therefore θ can be determined from the Parametric Equivalence Theorem 3.1.1.

3.2.4 BDF Methods

By instead considering an interpolation polynomial on the solution points, we derive the following scheme to construct the *Backwards Differentiation Formulas*. We would like to construct a degree k polynomial π_k satisfying the following conditions:

$$\pi_k(t_{n-i}) = y_{n-i}, \quad i = 0, \ldots, k \tag{3.22a}$$

$$\dot{\pi}_k(t_n) = \dot{y}_n = f_n, \tag{3.22b}$$

where y_n is computed by (implicitly) evaluating $\pi_k(t_n) = y_n$. We start by writing the interpolation polynomial

$$\pi_k(t) = \sum_{i=0}^{k} l_k^i(t) y_{n-i}, \tag{3.23}$$

and then differentiating the Lagrange basis polynomials to get the equation

$$\dot{\pi}_k(t) = \sum_{i=0}^{k} \dot{l}_k^i(t) y_{n-i}, \tag{3.24}$$

and then evaluating at t_n to obtain the formula

$$f_n = \sum_{i=0}^{k} \dot{l}_k^i(t_n) y_{n-i}. \tag{3.25}$$

This formula is again implicit and requires an iterative scheme to determine the new point y_n. The polynomial π_k itself now fits the form (3.1), and so $P_n = \pi_k$ satisfies:

$$\dot{P}_n(t_n) = y_n, \tag{3.26a}$$
$$P_n(t_{n-1}) = y_{n-1}, \tag{3.26b}$$
$$P_n(t_{n-i}) = y_{n-i}, \qquad i = 2, \ldots, k, \tag{3.26c}$$

and therefore the BDF methods are the family of I_k methods (3.5) with $\theta_{k-i} = 0$ for $i = 1, \ldots, k$.

For a fixed step size h, the Lagrange basis polynomials are differentiated to obtain the *Backwards Differentiation k-step Formulas* (BDF-k) for $1 \leq k \leq 6$.

$$hf_n = y_n - y_{n-1} \qquad \text{(Implicit Euler)} \tag{3.27a}$$
$$hf_n = \frac{3}{2}y_n - 2y_{n-1} + \frac{1}{2}y_{n-2} \tag{3.27b}$$
$$hf_n = \frac{11}{6}y_n - 3y_{n-1} + \frac{3}{2}y_{n-2} - \frac{1}{3}y_{n-3} \tag{3.27c}$$
$$hf_n = \frac{25}{12}y_n - 4y_{n-1} + 3y_{n-2} - \frac{4}{3}y_{n-3} + \frac{1}{4}y_{n-4} \tag{3.27d}$$
$$hf_n = \frac{137}{60}y_n - 5y_{n-1} + 5y_{n-2} - \frac{10}{3}y_{n-3} + \frac{5}{4}y_{n-4} - \frac{1}{5}y_{n-5} \tag{3.27e}$$
$$hf_n = \frac{147}{60}y_n - 6y_{n-1} + \frac{15}{2}y_{n-2} - \frac{20}{3}y_{n-3} + \frac{15}{4}y_{n-4} - \frac{6}{5}y_{n-5} + \frac{1}{6}y_{n-6}. \tag{3.27f}$$

The BDF methods are the unique implicit methods of order k with $\beta = (\beta_k, 0, \ldots, 0)$. The normalization of the highest term depends on the choice to set $\alpha_k = 1$, which it is not in the above formulas, but the conclusion about θ from the Parametric Equivalence Theorem 3.1.1 remains the same.

The BDF methods are only zero stable for $k \leq 6$. The advantage to using these BDF methods is that they are *stiff-stable:* for each of these methods, the stability region contains the entire negative real axis.

These are the most common multistep methods. They are used in industrial solver software such as ODEPACK, SUNDIALS, and other solvers [4, 5, 11, 12, 13, 15, 17]. They also appear in just about every academic text which discusses multistep methods [6, 8, 9, 10, 14, 18, 19, 20]. Their ubiquity makes them excellent reference methods from each class, and therefore it is valuable to consider the performance of other multistep methods in comparison to the corresponding Adams or BDF method of the same step number and order.

3.3 The MODES software

MODES is a Matlab software package for solving IVPs using parametrized multistep methods of variable step size and order. It is documented in [1] and can be downloaded here: `https://github.com/mss1972/MODES_v1.0`. The main

feature of MODES is the `modes` function which takes as input a right-hand side function `f`, an initial value `y0` and a time interval `tSpan` and returns the result of the simulation as two vectors `t` and `y`. The `modes` function has a number of optional parameters, most importantly the `'methodclass'` variable which can be `'E'`, `'I'` or `'Ip'`, designating the class of method to use. This variable is set in conjunction with `'methodfunc'`, which should be a function of the variable `p`, the order, that returns the θ vector of the appropriate length, determining the method for each order.

Since this paper focuses on comparing methods from the same class and order, the order control feature is not used, and instead a fixed order is set during integration with `methodfunc` being set to a function returning a constant value. There is also the possibility to choose `'methodname'` which selects a named family of methods for various orders, such as Adams-Bashforth, Adams-Moulton, BDF, or other named families such as EDF, dcBDF, or Nyström methods.

MODES has the option to set a filter for step size control, as well as error weight parameters and norm function. The theory behind MODES' step size control features is discussed in [1] as well as [3, 21, 22]. Most importantly, the step size is changed in every step, with the step size ratio being limited to the interval $[0.8, 1.2]$ by default. The purpose of this is to try to keep the step size sequence relatively "normal" and avoid drastic changes in step size. Since the step size controller itself is not the focus of this paper, it is left on default settings when step size adaptivity is required. It is assumed that the step size controller does not influence the results "too much" and that the comparisons of method performance are not significantly affected by this choice, but this is a confounding factor that could be subject to future investigation.

MODES uses a Newton iteration to solve implicit equations. At the time of writing, there is currently no option to choose any other nonlinear solver or set nonlinear solver options. Similarly to the assumption made for the step size controller, it is assumed that the choice of nonlinear solver does not significantly influence results.

The MODES package also has several useful tools. The first is `theta2coeff`, which converts a method class and θ vector to α and β coefficients. Next is `errorConstant` which takes as input α and β vectors as well as number of steps k and order p and returns the error coefficient of order p. Perhaps most useful is the tool `stabRegion`, which draws the outline of the stability region for a given method class and θ vector. To do this, θ is converted into α and β coefficients and the characteristic ρ and σ polynomials are determined. The boundary of the stability region is $h\lambda \in \mathbb{C}$ where the stability polynomial $\rho(z) - h\lambda\sigma(z)$ has a root of magnitude one. Therefore the points $h\lambda = \rho(z)/\sigma(z)$ are plotted for z in the complex unit circle. This tool only works for zero stable methods. These tools are helpful for determining method properties and creating graphs to compare.

Chapter 4

One and Two Parameter Classes

In this chapter we examine six low order method classes, and for each class we compare the properties of methods within that class. We will begin by finding explicit formulas to compute the fixed step size coefficients α and β from the method class and θ vector. From these coefficients we compute the error coefficient, and take roots of the ρ polynomial to determine stability. This will allow us to make predictions about how the methods will compare against each other when solving ODEs. Although this can be done using the MODES tools mentioned earlier, it is a straightforward process for low order methods to determine these formulas directly, and the calculations are much faster when the explicit formulas for the coefficients are used instead. Finally, we simulate some basic test problems to see how the methods compare in practice. Our primary goal is to motivate the possibility of changing methods within the same class and order as a means of improving solver results, at least in some simple scenarios.

Throughout this chapter we use the shorthand notations $T_1 = \tan(\theta_1)$ and $T_0 = \tan(\theta_0)$. Furthermore, all coefficients within a method class have a common denominator, which is the determinant of the consistency matrix. This will be denoted D and factored out. This should make the formulas more pleasant to look at.

4.1 Determining Method Properties

First, to analyze the theoretical properties of each method, we use the fixed step size formula and consistency conditions for each method class to determine explicit formulas for the coefficients in terms of the θ parameters.

For each given method class and parameters $\{\theta_i\}$, we determine the α and β coefficients of (2.3) by solving the system of order conditions. Although the multistep methods are not fixed step size, and the coefficients should in

theory also depend on the step size sequence $\{h_n\}$, this analysis will help us to determine which methods are zero stable and compute error coefficient. We determine zero stability from the roots of the characteristic polynomials, and error coefficient from Formula (2.10). The set of θ for which the methods are zero stable will be called the *zero stability region* of the method class; this is not to be confused with the *stability region* of a method, as defined in Section 2.1.3.

We begin with three lemmas, which when used in conjunction with the Root Condition, Theorem 2.1.1, will help us to quickly identify when methods of three or less steps are zero stable. The first lemma regarding zero stability of one step methods is rather trivial:

Lemma 4.1.1. *All consistent one step methods are zero stable.*

Proof. For a one step method, the characteristic ρ polynomial is linear, and is therefore of the form $\rho(z) = z - c$. Since the method is consistent, $\rho(1) = 0$ and therefore $\rho(z) = z - 1$. Since this polynomial satisfies the Root Condition, the method is zero stable. □

For a two step method, determining zero stability is similarly easy:

Lemma 4.1.2. *For a consistent two step method, the roots of the characteristic ρ polynomial are one and α_0.*

Proof. The characteristic ρ polynomial for a two step method is given by: $\rho(z) = z^2 + \alpha_1 z + \alpha_0$. Since the method is consistent, $\rho(1) = 0$ and therefore this will factor as $\rho(z) = (z - 1)(z - \alpha_0)$. □

As a corollary, $\alpha_1 = -(\alpha_0 + 1)$.

Lemma 4.1.3. *For a consistent three step method, the roots of the characteristic ρ polynomial are one and*

$$\frac{(\alpha_1 + \alpha_0) \pm \sqrt{(\alpha_1 + \alpha_0)^2 + 4\alpha_0}}{2}.$$

Proof. Since $\rho(1) = 0$, the polynomial factors:

$$\rho(z) = z^3 + \alpha_2 z^2 + \alpha_1 z + \alpha_0$$

$$= (z - 1)(z^2 + cz - \alpha_0).$$

By examining the quadratic term, we can determine that $c = (\alpha_2 + 1)$. By examining the linear term, we can conclude that

$$\alpha_0 + \alpha_1 = -(\alpha_2 + 1).$$

The conclusion then follows from the quadratic formula on the second factor. □

4.2 Experiments

In order to compare method performance in practice, we focus on only two simple problems: The first problem is the Linear Test Equation,

$$y' = \lambda y, \qquad y(0) = 1, \qquad t \in [0, 10], \tag{4.1}$$

with solution $y(t) = e^{\lambda t}$. λ is chosen to be negative so that the solution is decaying. Although this ODE is extremely basic, it will help us determine how stability and error coefficient affect method performance when comparing methods within the same class. Choosing λ to be large negative allows us to create a stiff problem and determine how method behavior changes between stiff and non-stiff problems. Another property of the Linear Test Equation is that Formula (2.10) simplifies to

$$T_n = EC * (h\lambda)^{p+1} y_{n-k} + \mathcal{O}(h^{p+2}), \tag{4.2}$$

which makes perfect sense because scaling the step size is proportional to scaling the decay coefficient. The other test problem we use is the Prothero-Robinson problem,

$$y' = \lambda(y - F(t)) + F'(t), \qquad y(0) = 10, \qquad t \in [0, 2] \tag{4.3}$$

with solution $y(t) = F(t) + (y(0) - F(0))e^{\lambda t}$. This problem can be viewed as "the Linear Test Equation with forcing." We choose the forcing function $F(t) = C\sin(\omega\pi t)$. The y dependence in the ODE remains linear, but the function F, which we have chosen here to be the sine function, prevents the solution from decaying to zero, which effectively prevents the error from decaying away and challenges the step size controller. The stiffness can again be controlled by λ, and C and ω can be chosen to challenge the solver, primarily the step size controller, to see how the method reacts to a more difficult problem numerically, even though it is still theoretically simple. In the following experiments, λ is fixed, since the effects of varying λ are examined in the previous problem. The values of C and ω are chosen to try to produce "interesting results." This choice and goal are clearly somewhat arbitrary. However it is guided by the idea that choosing very small values will cause the results to look similar to the Linear Test Equation, while choosing very large values will cause the solver to take many steps and possibly fail. Therefore we attempt to design our experiments to land between these two extremes.

For each problem and method, we run two tests:

In the first test, we set a fixed step size and measure how accurate the results are to the theoretical solution. For the Linear Test Equation we use a step size of $h = 0.01$ and for the Prothero-Robinson problem we use a step size of $h = 0.002$ for a total of 1000 steps in both cases. To measure error, we use a weighted root mean square (wRMS) norm, which is a discrete approximation to the L^2 norm using the Trapezoidal Quadrature Rule:

$$||y||^2 = \sum_{i=1}^{N} (t_i - t_{i-1}) \left(\frac{y_i^2 + y_{i-1}^2}{2} \right). \tag{4.4}$$

This accounts for all points in the simulation. For this experiment, we consider the "best" method to be the one which achieves the smallest error from the true solution.

In the second test, we set a fixed tolerance and allow the step size to vary according to MODES' default step size controller. The tolerance must be determined based on the method class. We also measure the error between the computed solution and the theoretical solution using the same norm as before. For this experiment we are looking for which methods can take fewer steps while still maintaining a reasonable amount of accuracy.

4.3 One Parameter Classes

The one parameter classes are the E_2, I_1 and I_2^+ methods. Although low order methods may not be the most interesting for applications in solving real world problems, they illustrate some trends which may generalize to methods of higher order.

For all method classes, the goal is to illustrate the landscape of the parameter space in order to highlight the strengths and weaknesses of each method. It is possible to test a large number of different methods by breaking the interval $[0, \pi]$ into some number of equally spaced points and running tests with every method which lands on one of these points. The denominator of 144 is chosen for its sufficient size and high divisibility: meaning θ_0 is chosen in increments of $\pi/144$, because this choice makes smooth graphs and gives us many "nice" fractions like $3\pi/4$, or $\pi/6$ for example.

For all three method classes, we first establish the explicit formulas for the α and β coefficients, which allows us to easily determine zero stability and error coefficients. The roots of the ρ polynomial and the error coefficient are then plotted. Next, we simulate the Linear Test Equation with different values of λ in order to get a basic comparison between these methods and to see how stiffness changes performance. Next, we simulate the Prothero-Robinson problem with λ fixed at -1 and different values of C and ω in order to determine how the methods handle a slightly more difficult problem. For situations which warrant extra investigation, we run more experiments. For the explicit and implicit-plus methods, we examine the effect of a tighter tolerance. For the implicit methods, we examine the effect of increased stiffness.

4.3.1 E_2 Methods

We begin with the explicit two step methods. The fixed step size formula is given by

$$y_n + \alpha_1 y_{n-1} + \alpha_0 y_{n-2} = h(\beta_1 y'_{n-1} + \beta_0 y'_{n-2}). \tag{4.5}$$

For consistency and order two, we require

$$\alpha_1 + \alpha_0 = -1 \tag{4.6a}$$
$$\beta_1 + \beta_0 = 2 + \alpha_1 \tag{4.6b}$$
$$2(\beta_1) = 4 + \alpha_1. \tag{4.6c}$$

Using the Parametric Equivalence Theorem to substitute out $\beta_0 = -T_0\alpha_0$, the coefficients can be determined by solving the system

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & T_0 & -1 \\ 1 & 0 & -2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ -4 \end{pmatrix}. \tag{4.7}$$

For $T_0 \neq 1/2$ the matrix can be inverted to give

$$\frac{1}{2T_0 - 1} \begin{pmatrix} 2T_0 & -2 & 1 \\ -1 & 2 & -1 \\ T_0 & -1 & 1 - T_0 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ -4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_0 \\ \beta_1 \end{pmatrix}. \tag{4.8}$$

Therefore we can solve the coefficients explicitly,

$$D = 2T_0 - 1 \tag{4.9a}$$
$$D\alpha_1 = -2T_0 \tag{4.9b}$$
$$D\alpha_0 = 1 \tag{4.9c}$$
$$D\beta_1 = 3T_0 - 2 \tag{4.9d}$$
$$D\beta_0 = -DT_0\alpha_0 = -T_0. \tag{4.9e}$$

According to Lemma 4.1.2, these methods are strongly stable when $|\alpha_0| = |\frac{1}{D}| = |\frac{1}{2T_0 - 1}| < 1$, which occurs when $\theta_0 \in (\pi/4, \pi)$. At $\theta_0 = \pi/2$ we have the Adams-Bashforth method AB-2.

The error constant for the E_2 methods can be calculated from:

$$C_3 = \frac{1}{3!}(\alpha_1 + 8) - \frac{1}{2!}(\beta_1) \tag{4.10a}$$
$$EC = \frac{C_3}{\beta_1 + \beta_0}. \tag{4.10b}$$

This gives:

$$C_3 = \frac{5T_0 - 2}{6(2T_0 - 1)}, \tag{4.11a}$$
$$EC = \frac{5T_0 - 2}{12(T_0 - 1)}. \tag{4.11b}$$

In Figure 4.1 we plot the absolute values of the second root of the ρ polynomial, which shows which methods are zero stable, and the error coefficient.

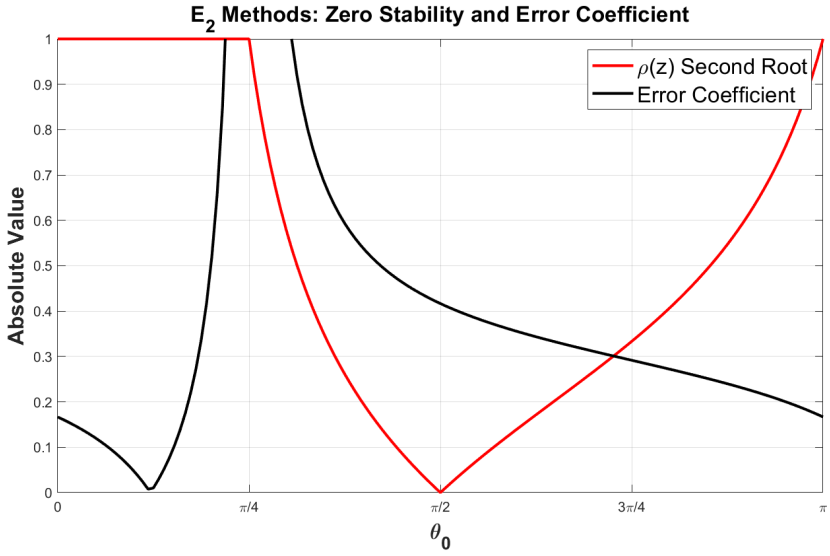

Figure 4.1: E_2 Methods: The methods are zero stable where the second root of $\rho(z)$ is less than one, which is between $\pi/4$ and π. Within this interval, the error coefficient is decreasing, with a minimum of $1/6$ at $\theta_0 = \pi$. At $\theta_0 = \pi/2$ is the AB-2 method.

The plot of the second root verifies that the method is zero stable for $\theta_0 \in (\pi/4, \pi)$. In this region, the error coefficient decreases as θ_0 becomes larger. Within the zero stability region, the error coefficient is bounded below by $1/6$. For $\theta_0 = \arctan(2/5)$ the error coefficient vanishes and the method becomes order three, but it is not zero stable. This is to be expected due to the Dahlquist Barrier Theorem.

We would like to see how the stability regions change as θ_0 changes. For some values of $\theta_0 \in [\pi/4, \pi]$, the corresponding stability regions are shown in Figure 4.2. We see that as θ_0 becomes larger the stability region shrinks. Note that $\theta_0 = \pi/4$ and $\theta_0 = \pi$ are both essentially "invalid" methods, since $\theta_0 = \pi/4$ corresponds to a method with infinite error coefficient, while $\theta_0 = \pi$ corresponds to a method which is nowhere stable.

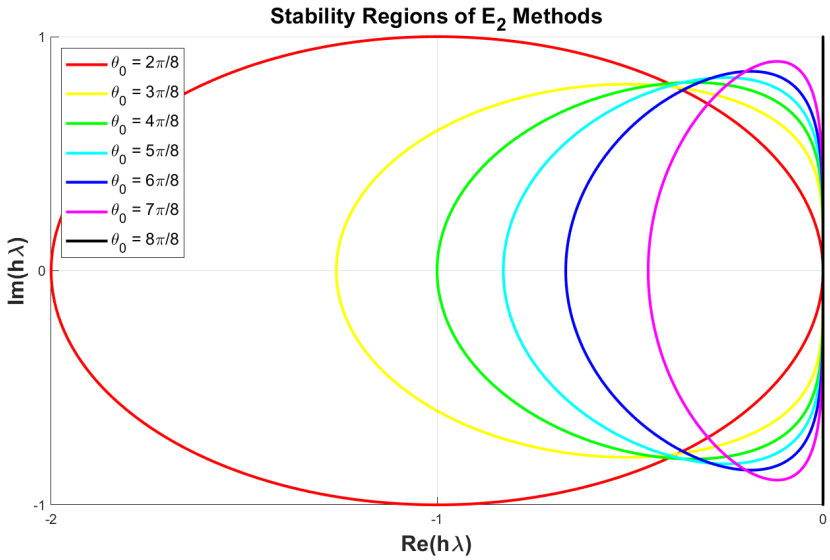

Figure 4.2: E_2 Methods: stability regions for different values of θ_0. The stability region is becoming smaller as θ_0 increases from $\pi/4$ to π. At $\theta_0 = \pi/4$ the stability region is largest, but the error coefficient approaches infinity asymptotically. At $\theta_0 = \pi$ the error coefficient is minimized, but the stability region vanishes.

This demonstrates an important analytic property for this class of methods: there is a trade-off between methods with a large stability region and those with a small error coefficient.

Next, we run simulations using the Linear Test Equation. In the fixed step size experiment, we choose $\lambda = -1, -2, -4, -8, -16, -32$. We choose the zero stable methods with $\pi/4 < \theta_0 < \pi$, omitting endpoints. The accuracy measurements are plotted in Figure 4.3. We find three interesting trends:

1. The simulations become less accurate as λ is increased (more negative). This may be because λ directly affects the error term as given in Equation (4.2).

2. The lines slope downward, following the trend in error coefficient. Therefore, a smaller error coefficient is better, *so long as the method remains stable,* which leads directly to the next item:

3. As λ becomes larger, methods near $\theta_0 = \pi$ become unstable. This is because the stability region of these methods is too small to contain $h\lambda$ for the given step size.

Therefore, in this experiment the conclusion is: the "best" method is the one furthest to the right, such that the stability region is large enough to include $h\lambda$ for the given problem and step size.

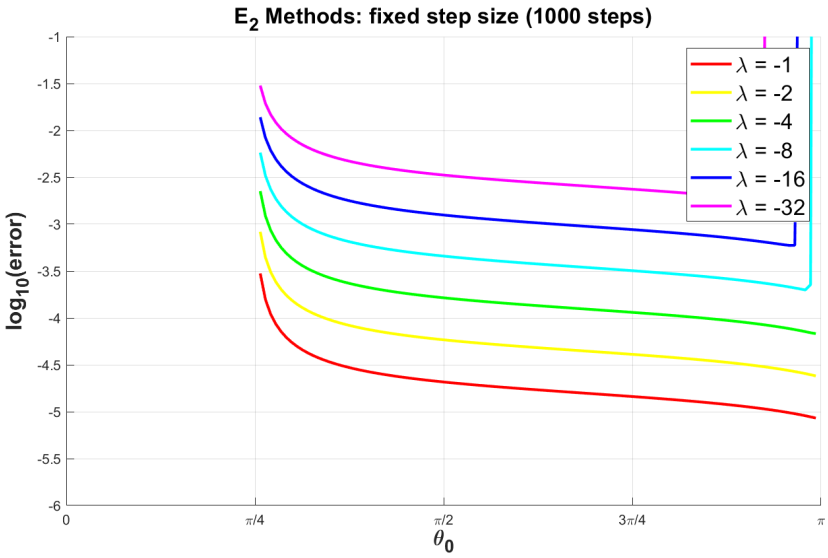

Figure 4.3: E_2 Methods: Accuracy of fixed step size simulation of Linear Test Equation (4.1) for different values of λ. Step size is $h = 0.01$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

Next, we simulate the Linear Test Equation with variable step size. A tolerance of 10^{-4} is chosen. The number of steps are plotted in Figure 4.4. In this experiment we find that for the larger (more negative) values of λ, especially $\lambda = -16$ and $\lambda = -32$, the conclusion from the previous experiment is actually reversed, and methods for smaller values of θ_0 perform better. It seems that when the problem is more stiff, the need for a larger stability region is greater than the benefit gained from the smaller error coefficient. When the problem is less stiff, the curve is more flat and the methods perform more similarly to each other.

In Figure 4.5 we see that all methods are achieving a reasonable amount of accuracy. For all cases, methods on the right are slightly more accurate. However, this isn't an issue: the global error is not expected to be less than the tolerance, only reasonably small with respect to it. The reason that the stiffer problems achieve higher accuracy is likely because of the fact that they decay to zero faster. The "squiggles" that occur between neighboring methods may be an effect of the error controller, but this is not entirely clear.

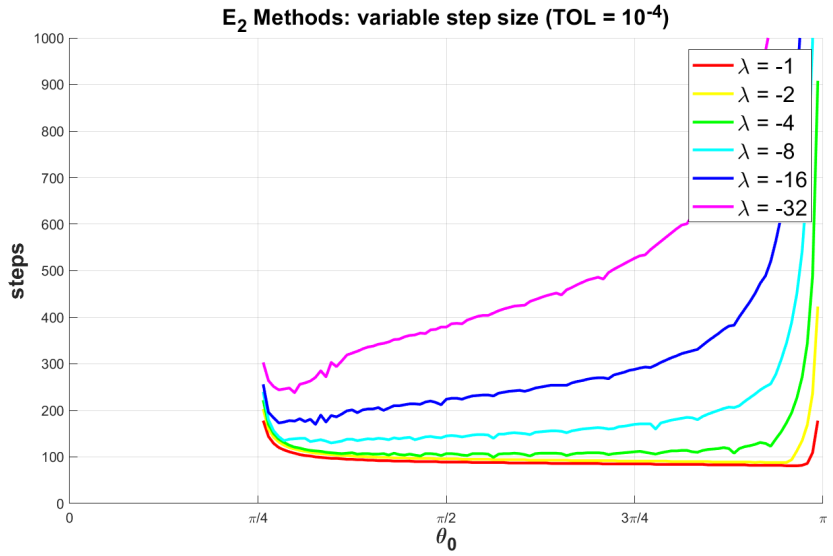

Figure 4.4: E_2 Methods: Number of steps in variable step size simulation of Linear Test Equation (4.1) for different values of λ with a tolerance of 10^{-4}. Step size controller is MODES' default.

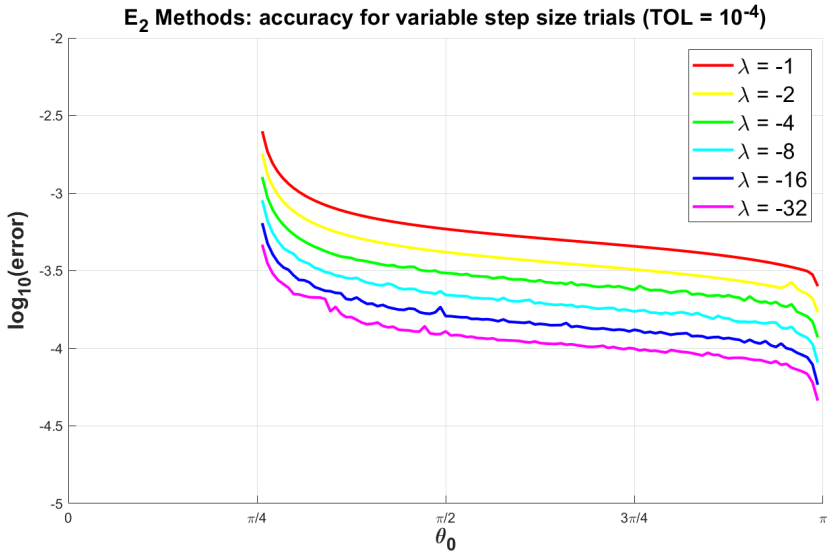

Figure 4.5: E_2 Methods: Accuracy of variable step size simulation of Linear Test Equation (4.1) depicted in Figure 4.4, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

We also consider the effect of tightening the tolerance. In Figure 4.6 we show the same simulation with a tolerance of 10^{-8}. In this case we find that for smaller values of λ there is a clear preference for methods with larger θ_0. This is likely because a small step size is already required in order to achieve the desired accuracy, and therefore there is no benefit to gain from a large stability region. Instead, the solver benefits from a smaller error coefficient. For larger λ the solver again begins to need a larger stability region but the tolerance still limits the maximum step size and so the best method is found somewhere in the middle.

Accuracy is plotted in Figure 4.7. Methods on the right are slightly more accurate, with a range of accuracies similar to the previous experiment.

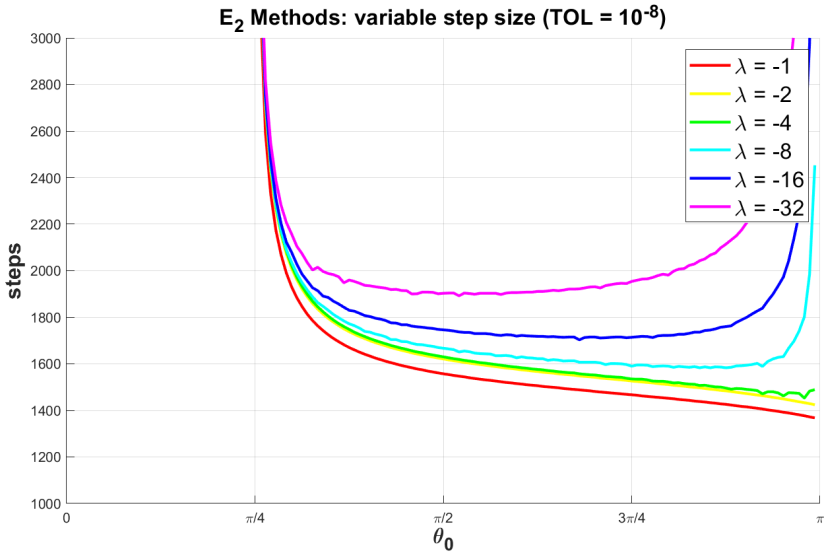

Figure 4.6: E_2 Methods: Number of steps in variable step size simulation of Linear Test Equation (4.1) for different values of λ with a tolerance of 10^{-8}. Step size controller is MODES' default.

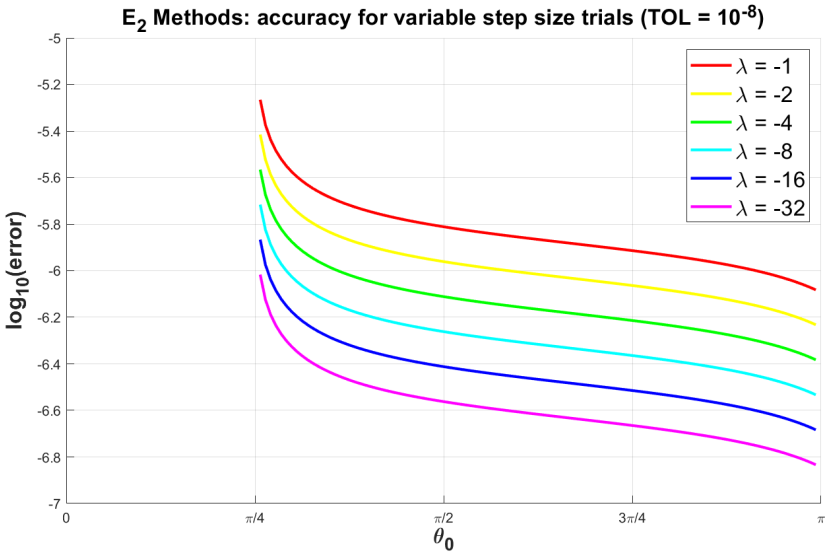

Figure 4.7: E_2 Methods: Accuracy of variable step size simulation of Linear Test Equation depicted in Figure 4.6, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

Another pattern to note in Figures 4.3, 4.5, and 4.7 is that the lines are equally spaced. Since the values of λ are chosen in powers of two and the log of the error is plotted, the equal spacing demonstrates that the error is proportional to a power of λ. In the error formula for the Linear Test Equation, Equation (4.2), we see that this should be the case for the leading error term. This is simply a property of the Linear Test Equation and is not unique to this method class.

Finally we simulate the Prothero-Robinson problem without stiffness ($\lambda = -1$) but with various amplitudes and frequencies of the forcing sine function. The comparisons between the methods is similar to the nonstiff Linear Test Equation case. We note that although the problem $C = 5, \omega = 1$ achieves higher accuracy in the fixed step size trial and fewer steps in the variable step size trial than the problem $C = 1, \omega = 10$, it is actually less accurate in the variable step size trial.

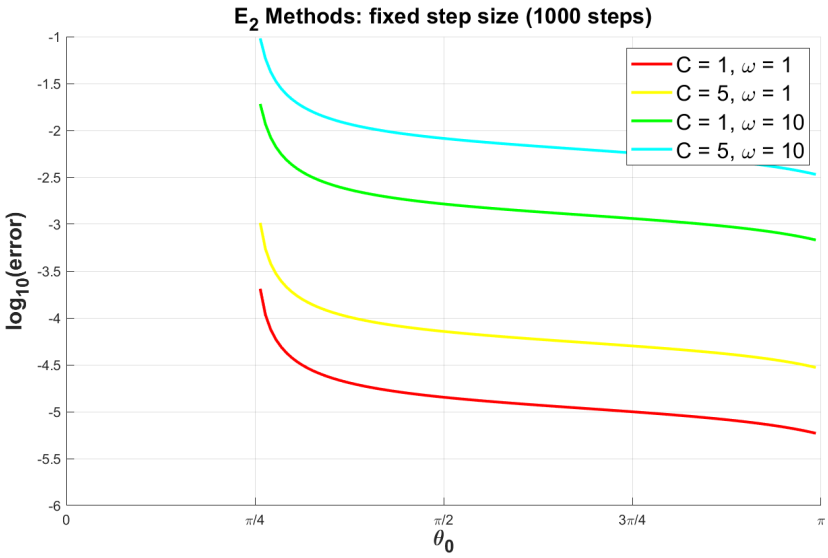

Figure 4.8: E_2 Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω. Step size is $h = 0.002$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

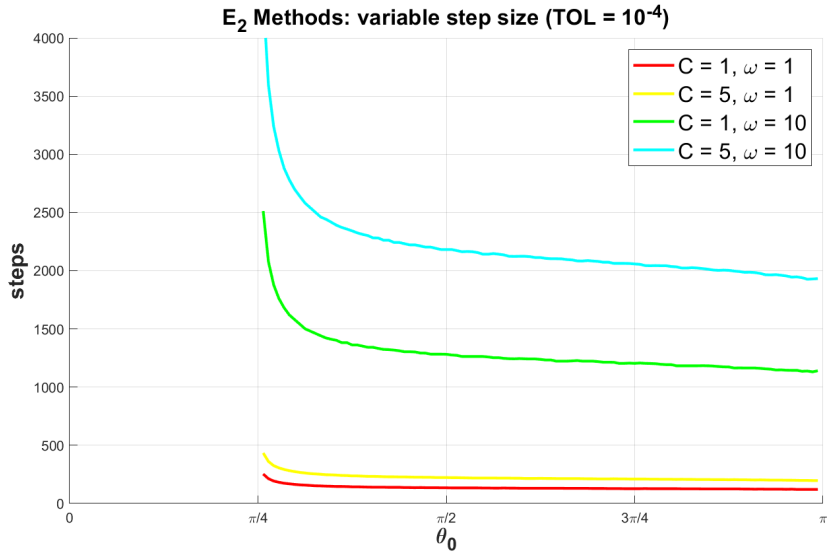

Figure 4.9: E_2 Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω with a tolerance of 10^{-4}. Step size controller is MODES' default.

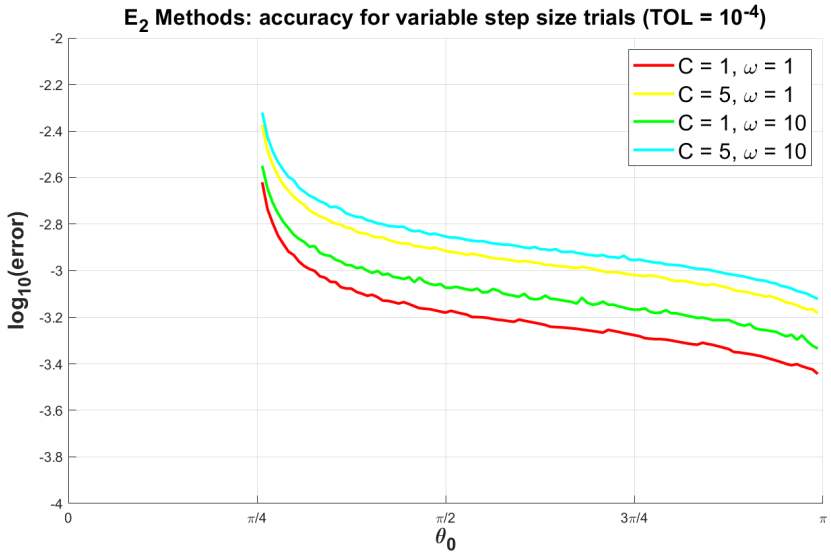

Figure 4.10: E_2 Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.9, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

4.3.2 I_1 Methods

The implicit one step methods are the familiar "Theta Methods" and are usually parametrized as

$$y_n = y_{n-1} + h(\theta f_{n-1} + (1-\theta)f_n) \tag{4.12}$$

for $0 \leq \theta \leq 1$. To fit into the scheme presented in Chapter 3, they will be parametrized as

$$y_n = y_{n-1} + h(\tan(\theta_0) f_{n-1} + (1-\tan(\theta_0))f_n) \tag{4.13}$$

with $0 \leq \theta_0 \leq \pi$. Clearly this is a superset since choosing $\theta_0 > \pi/4$ corresponds to a choice of $\theta \notin [0,1]$. However this parametrization includes all consistent one step implicit methods.

By Lemma 4.1.1 the I_1 methods are all zero stable, so the only thing to compute is the error coefficient.

The α and β coefficients can be read off from Equation 4.13. The error coefficient is

$$EC = \tan(\theta_0) - \frac{1}{2}. \tag{4.14}$$

The absolute values of the error coefficients are plotted in Figure 4.11.

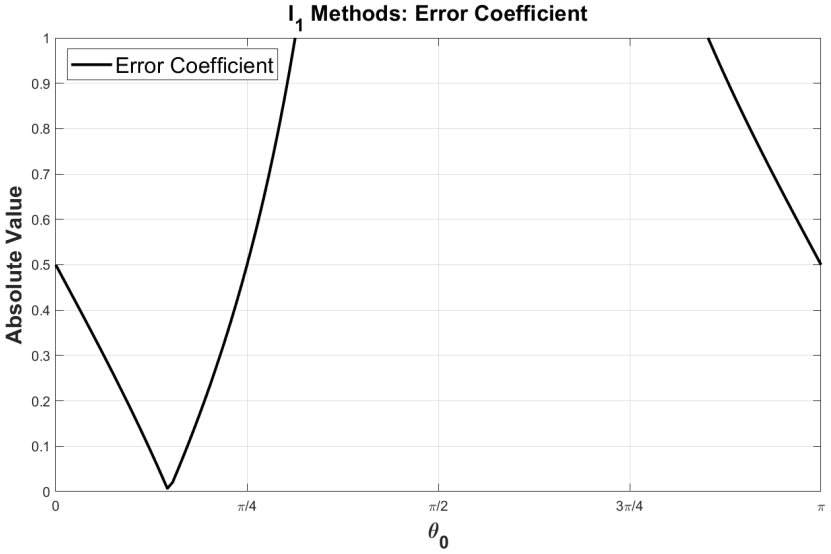

Figure 4.11: I_1 Methods: Absolute values of error coefficients. All methods are zero stable, but no method is defined for $\theta_0 = \pi/2$. For $\theta_0 = 0$, $\arctan(1/2)$, and $\pi/4$ we have the Implicit Euler, Trapezoidal Rule, and Explicit Euler methods, with error coefficients of $1/2$, 0, and $1/2$ respectively.

There is an asymptote at $\theta_0 = \pi/2$ where the α and β coefficients tend to infinity. On the interval $(\pi/2, \pi] \cup [0, \arctan(1/2)]$ the methods are A-stable

and the error coefficient is decreasing. $\theta_0 = 0$ corresponds to the Implicit Euler method. $\theta_0 = \arctan(1/2)$ is the Trapezoidal Rule, where the error coefficient vanishes and the method attains order two. On the interval $[\arctan(1/2), \pi/2)$ the methods have bounded stability regions, with the stability region shrinking and error coefficient increasing for larger θ_0. Furthermore, at $\theta_0 = \pi/4$, a degeneracy occurs and the implicit term disappears and the method becomes the Explicit Euler method.

Beginning with the Linear Test Equation, when the problem is not stiff we see in the fixed step size trial in Figure 4.12 that the performance is nearly directly associated with the error coefficient. The Trapezoidal rule is the most accurate and the accuracy decreases in both directions as θ_0 approaches $\pi/2$. For $\lambda = -32$ the minimum begins to shift to the left, away from $\theta_0 = \arctan(1/2)$. Since the advantage to this method class is its ability to handle stiff problems, we will investigate shortly how these methods compare against each other when stiffer problems are chosen.

However, we first look at what happens when the step size is allowed to change. In the variable step size trial we see in Figure 4.13 that the minimum is shifted to the Explicit Euler method, which is taking less steps than all other methods. When we measure accuracies, there is a significant decrease in accuracy at $\theta_0 = \pi/4$ suggesting that there must be an issue with the error estimator. This is shown in Figure 4.14.

The reason why this happens is because of how the error estimator for I_k methods is constructed, which is detailed in [16, Theorem II.6.2]. For an I_k method constructed from a parameter vector θ, the error is estimated as $||x_n - x^{\text{pred}}||$, where x^{pred} is calculated by an E_k method constructed from $\hat{\theta}$, where $\hat{\theta}$ is the same as the first $k-1$ components of θ. This means that whenever an I_k method degenerates to an explicit method, x^{pred} will actually be calculated using an E_k method which is the same method, and so x_n will equal x^{pred} and the error estimate will be zero. For the I_1 methods, x^{pred} is determined using the unique order one explicit method, which is the Explicit Euler method. Since the error estimator is measuring zero error, the step size is ramping up by the maximum step size ratio in every step, which is 1.2 by default. This clearly causes a significant loss in accuracy.

Aside from this issue, we find that with the A-stable methods, the methods with θ_0 closer to $\arctan(1/2)$ are all performing better, achieving less steps and better accuracy. Furthermore the choice of problem is not greatly affecting performance, with stiffer problems requiring only slightly more steps, and achieving greater overall accuracy for likely the same reason as in the explicit case: simply because the problem is decaying to zero. For the non A-stable methods, the performance is hindered greatly by the increased stiffness, and the results are generally poor, so it is probably a good idea to simply not use these methods.

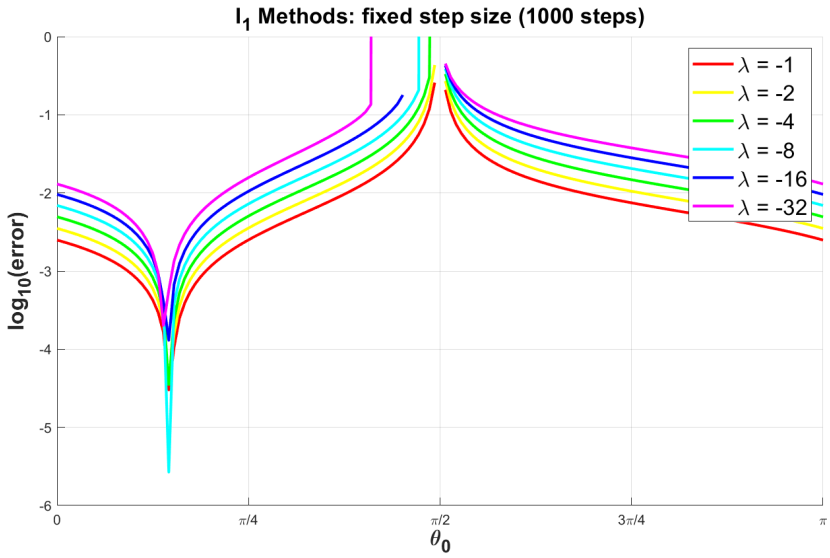

Figure 4.12: I_1 Methods: Accuracy of fixed step size simulation of Linear Test Equation (4.1) for different values of λ. Step size is $h = 0.01$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

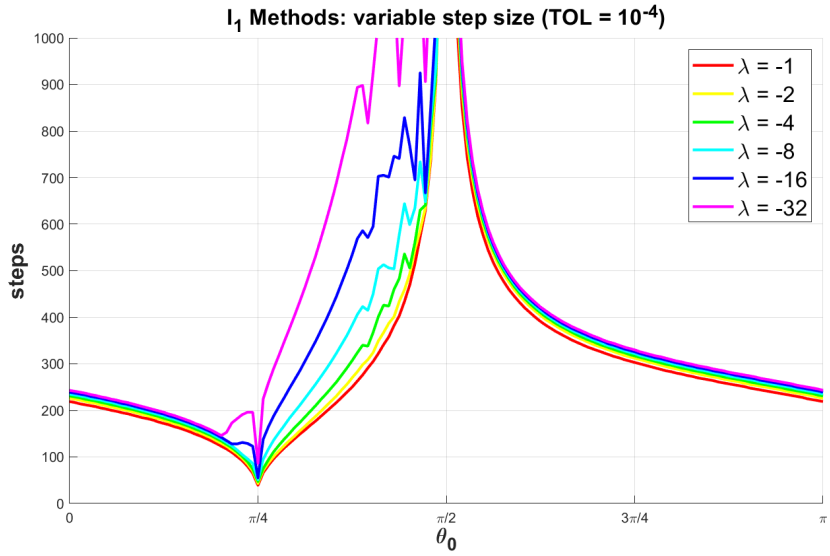

Figure 4.13: I_1 Methods: Number of steps in variable step size simulation of Linear Test Equation (4.1) for different values of λ with a tolerance of 10^{-4}. Step size controller is MODES' default.

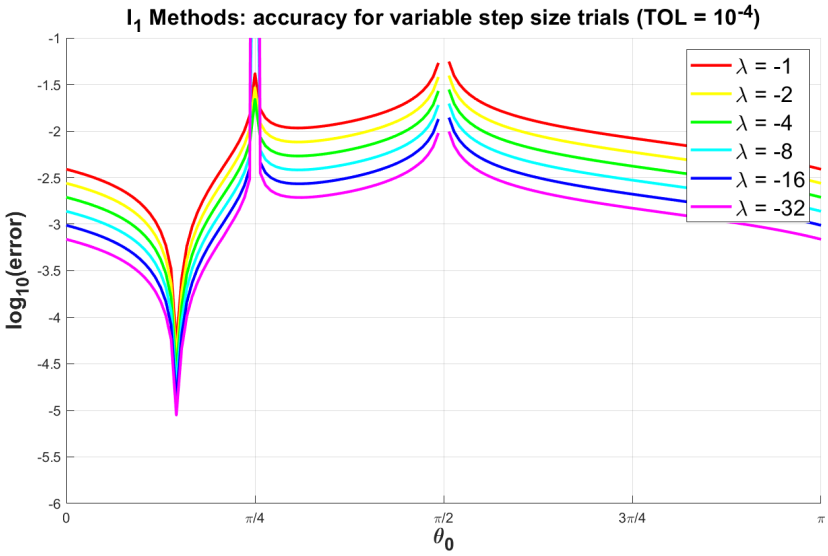

Figure 4.14: I_1 Methods: Accuracy of variable step size simulation of Linear Test Equation (4.1) depicted in Figure 4.13, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

Just like with the E_2 method class, we observe that the lines in Figures 4.12 and 4.14 are equally spaced since the values of λ are chosen as powers of two and the principle error term in Equation 4.2 is a power of λ.

When the stiffness is increased, we find a very interesting trend. For the fixed step size trial, in Figure 4.15 we see that the minimum shifts from $\arctan(1/2)$ to 0 as the stiffness increases. This supports the idea that the Implicit Euler method is better than the Trapezoidal Rule for extremely stiff problems due to its greater numerical damping. This also brings up the possibility that a method choice somewhere between the two might be optimal depending on the stiffness of the problem. These are simply the Theta Methods in Equation (4.12) with $0 \leq \theta \leq 1/2$. As expected, the methods which are not A-stable quickly become useless.

For the variable step size trial, Figure 4.16 shows that the A-stable methods all perform similarly well, with the increased stiffness not greatly affecting performance, except in a small region near $\pi/2$. In fact, the step counts are even comparable to the nonstiff case in Figure 4.13. The accuracy calculations in Figure 4.17 show that the Trapezoidal Rule is still achieving higher accuracy. Therefore, the effectiveness of the Trapezoidal Rule should not be ignored, even for extremely stiff problems.

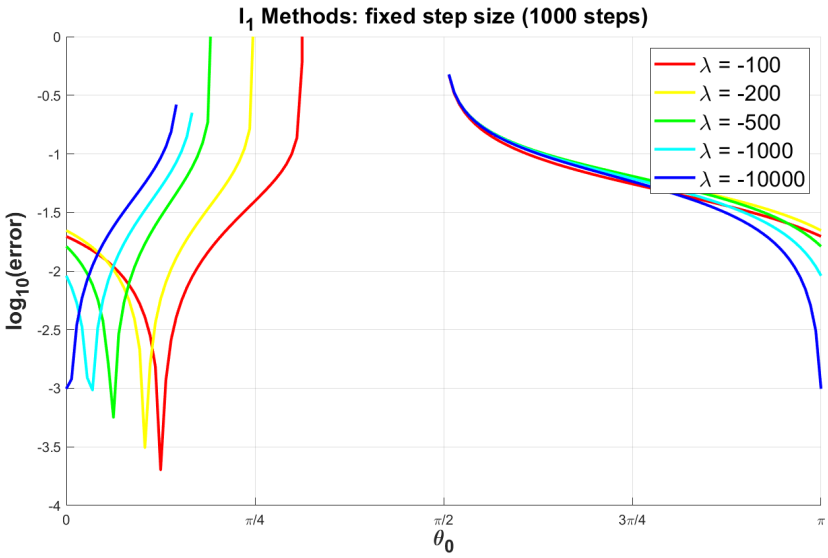

Figure 4.15: I_1 Methods: Accuracy of fixed step size simulation of stiff Linear Test Equation (4.1) for different (large negative) values of λ. Step size is $h = 0.01$, global error measured using wRMS norm (4.4).

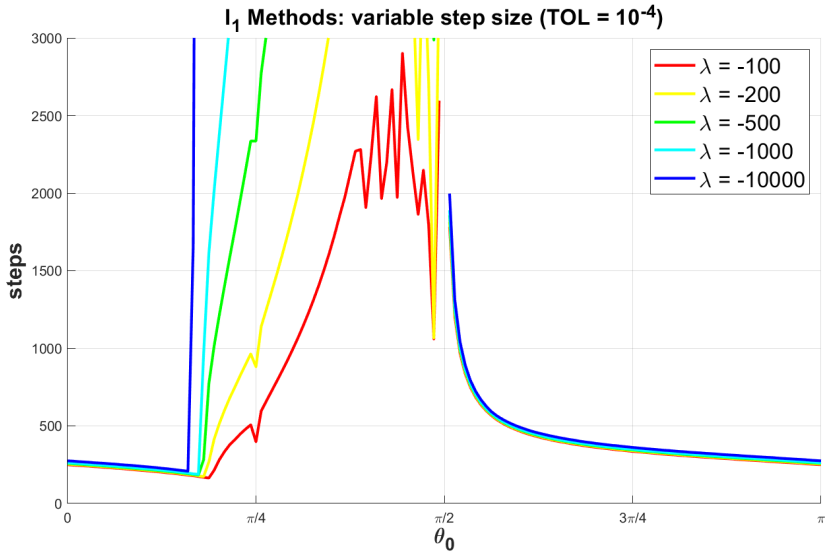

Figure 4.16: I_1 Methods: Number of steps in variable step size simulation of stiff Linear Test Equation (4.1) for different (large negative) values of λ with a tolerance of 10^{-4}. Step size controller is MODES' default.

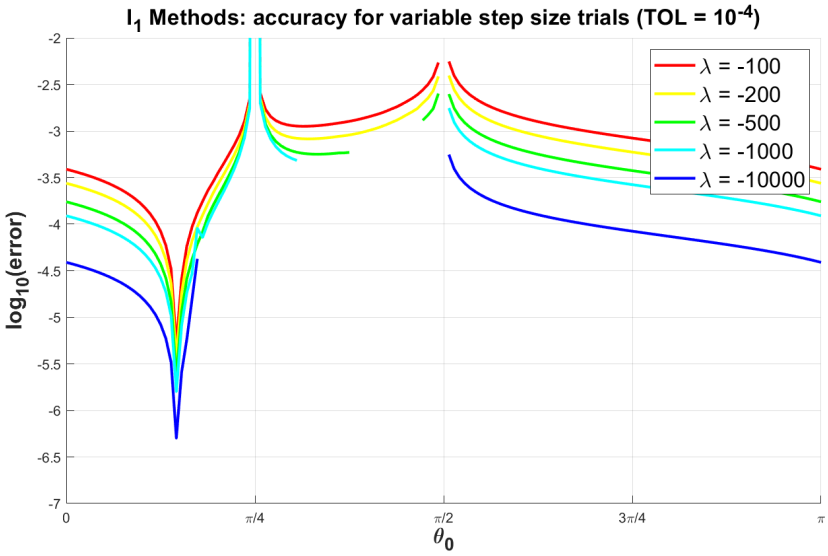

Figure 4.17: I_1 Methods: Accuracy of variable step size simulation of stiff Linear Test Equation (4.1) depicted in Figure 4.16, error measured using wRMS norm (4.4).

Finally we examine the Prothero-Robinson problem. As a comparison between methods, the results are not extremely different from those found in the Linear Test Equation. However, we find that while increasing the frequency ω does not cause a great loss of accuracy in the fixed step size case, it does cause the solver to require many more steps in the variable step size case.

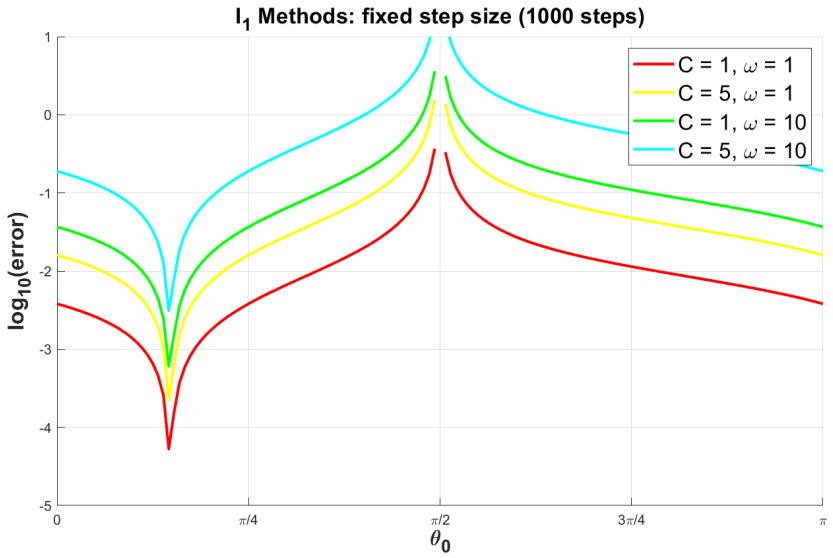

Figure 4.18: I_1 Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω. Step size is $h = 0.002$, global error measured using wRMS norm (4.4).

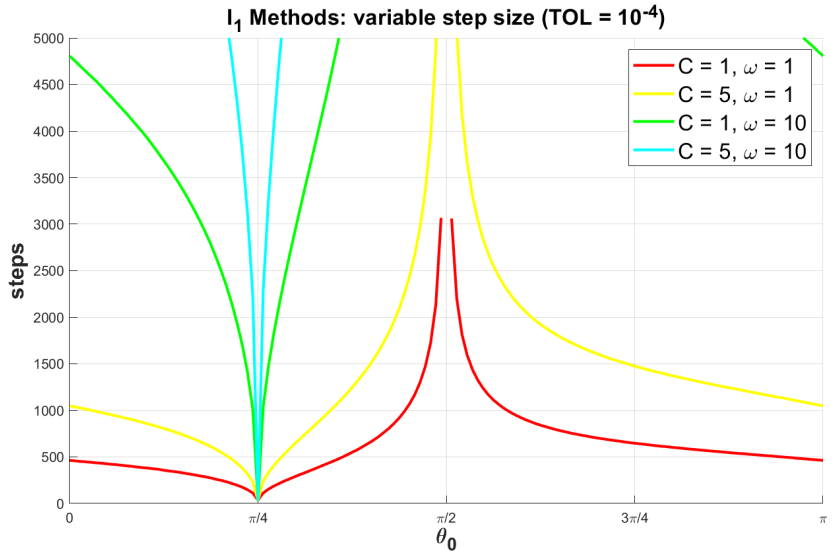

Figure 4.19: I_1 Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω with a tolerance of 10^{-4}. Step size controller is MODES' default.

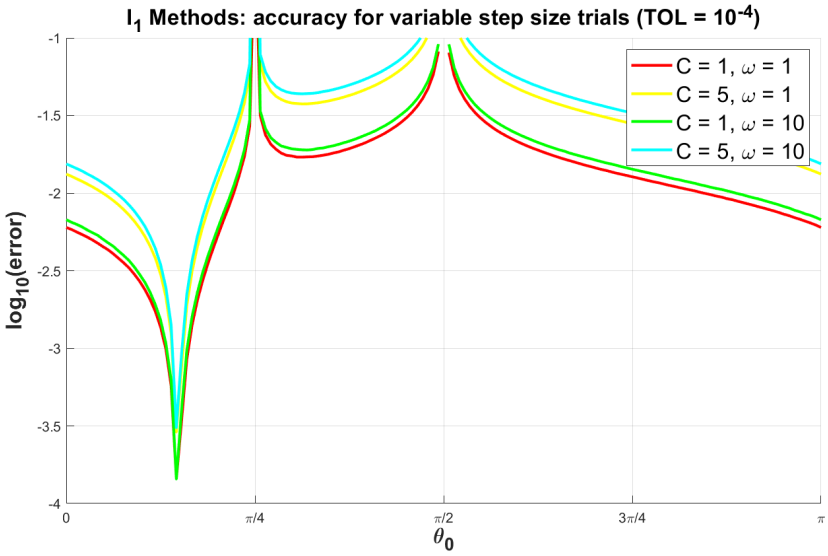

Figure 4.20: E_2 Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.19, error measured using wRMS norm (4.4).

4.3.3 I_2^+ Methods

The Implicit Plus methods have more variables and order conditions. For the I_2^+ methods, the order conditions lead to the system

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & T_0 & -1 & -1 \\ 1 & 0 & -4 & -2 \\ 1 & 0 & -12 & -3 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_0 \\ \beta_2 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ -4 \\ -8 \end{pmatrix}. \tag{4.15}$$

Inversion gives:

$$\frac{1}{12T_0 - 5} \begin{pmatrix} 12T_0 & -12 & 9 & -2 \\ -5 & 12 & -9 & 2 \\ -T_0 & 1 & 3T_0 - 2 & 1 - 2T_0 \\ 8T_0 & -8 & 11 - 12T_0 & 4T_0 - 3 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ -4 \\ -8 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_0 \\ \beta_2 \\ \beta_1 \end{pmatrix}, \tag{4.16}$$

which leads to the explicit formulas for the coefficients:

$$D = 12T_0 - 5 \tag{4.17a}$$
$$D\alpha_1 = 4 - 12T_0 \tag{4.17b}$$
$$D\alpha_0 = 1 \tag{4.17c}$$
$$D\beta_2 = 5T_0 - 2 \tag{4.17d}$$
$$D\beta_1 = 8T_0 - 4 \tag{4.17e}$$
$$D\beta_0 = -DT_0\alpha_0 = -T_0. \tag{4.17f}$$

Using Lemma 4.1.2, we determine that methods are zero stable when $|\frac{1}{12T_0-5}| < 1$, which is exactly when $\theta_0 \notin [\arctan(1/3), \arctan(1/2)]$.

The error coefficient can be determined from the formulas:

$$C_4 = \frac{1}{4!}(16 + \alpha_1) - \frac{1}{3!}(8\beta_2 + \beta_1), \tag{4.18a}$$

$$EC = \frac{C_4}{\beta_2 + \beta_1 + \beta_0}, \tag{4.18b}$$

which leads to the computations:

$$C_4 = \frac{-3T_0 + 1}{6(12T_0 - 5)}, \tag{4.19a}$$

$$EC = \frac{-3T_0 + 1}{36(2T_0 - 1)}. \tag{4.19b}$$

In Figure 4.21 we plot the absolute values of the second root of the ρ polynomial and the error coefficient.

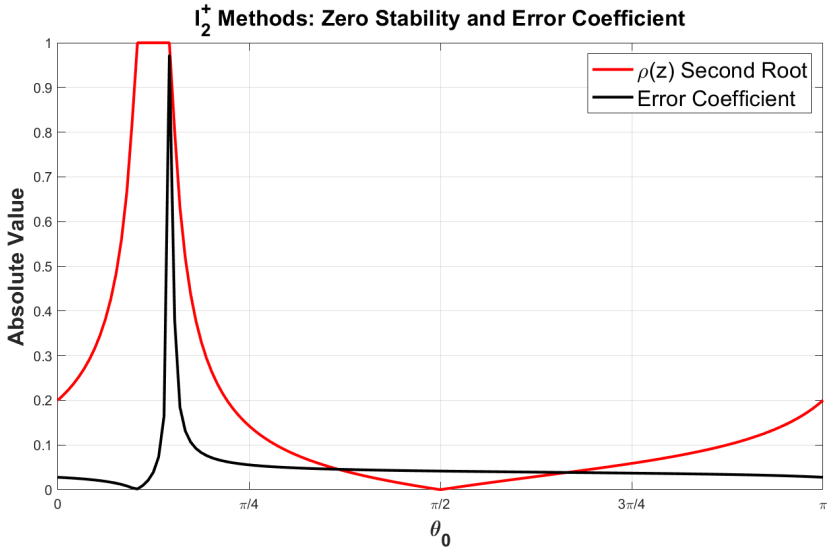

Figure 4.21: I_2^+ Methods: The methods are zero stable where the second root of $\rho(z)$ is less than one, which is from $\arctan(1/2)$ wrapping around to $\arctan(1/3)$. Within this interval, the error coefficient is decreasing but mostly flat, reaching zero at $\theta_0 = \arctan(1/3)$, giving a weakly stable order four method. At $\theta_0 = \pi/2$ is the AM-2 method.

For this method class the zero stability region is quite large and the error coefficient curve is quite flat, but decreasing for larger θ_0. At $\theta_0 = \pi/2$ is the Adams-Moulton method AM-2. At $\theta_0 = \arctan(1/3)$ is a weakly stable order

four method with $\alpha = (1, 0, -1)$, $\beta = (1/3, 4/3, 1/3)$ and fourth order error coefficient $1/180$.

We examine how the stability region changes as θ_0 changes. In Figure 4.22 we see that the stability region also shrinks for larger θ_0, therefore giving a trend similar to that witnessed in the E_2 methods where there is a trade-off between methods with large stability regions and those with small error coefficients.

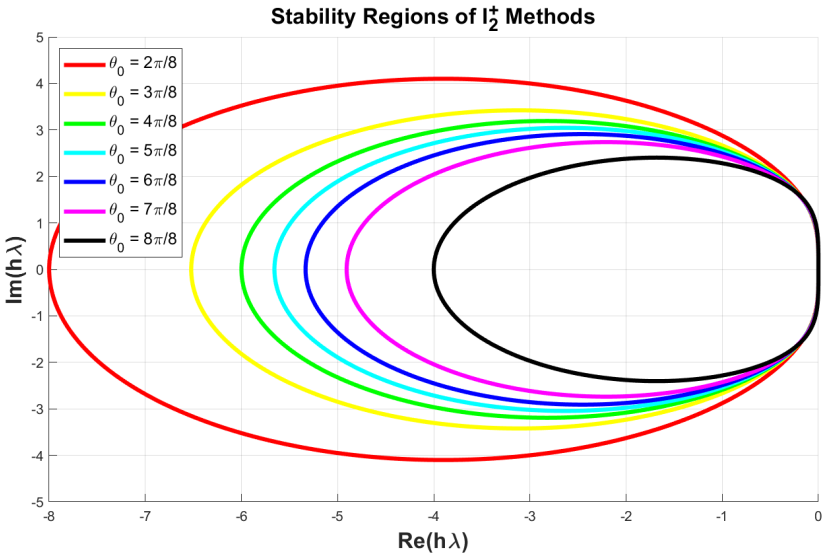

Figure 4.22: I_2^+ Methods: stability regions for different values of θ_0. The stability region is becoming smaller as θ_0 increases from $\pi/4$ to π.

Next, we simulate the Linear Test Equation. In Figure 4.23 we see that method performance is relatively flat for each given problem, but that the increased stiffness actually causes a significant decrease in accuracy overall. One likely explanation for this is that in Equation 4.2 the power of p is greater since these methods are higher order ($p = 3$), and so the stiffness λ has a greater effect on the error. The step size also has a greater influence on the error and this is to be expected from higher order methods: they converge faster as h becomes small, but can be outperformed by lower order methods when the step size is large. For all problems, the method achieving highest accuracy seems to be near $\arctan(1/3)$, which is where the error coefficient is smallest, and the zero stability region ends.

When the step size is allowed to vary, we see that for a tolerance of 10^{-4} in Figure 4.24 that most methods are able to simulate the problem in very few steps, but methods near $\theta_0 = \pi$ start to experience poor performance, and for those past $\theta_0 = 0$, the simulation fails. In these cases `modes` returns an error that the polynomial coefficients are not finite. Furthermore, in Figure 4.25 we see that the pattern of poor performance extends to the achieved accuracy.

The reason for this is not entirely clear: Although the stability region becomes smaller towards the right, it does not vanish. If, for example, we look at the method for $\theta_0 = 0$ we have the coefficients

$$\alpha = (1, -0.8, -0.2)$$
$$\beta = (0.4, 0.8, 0)$$

The error coefficient is 1/36 and the stability region is large enough to touch -4 (it is depicted in Figure 4.22). The reason why these methods fail is a question left to future research.

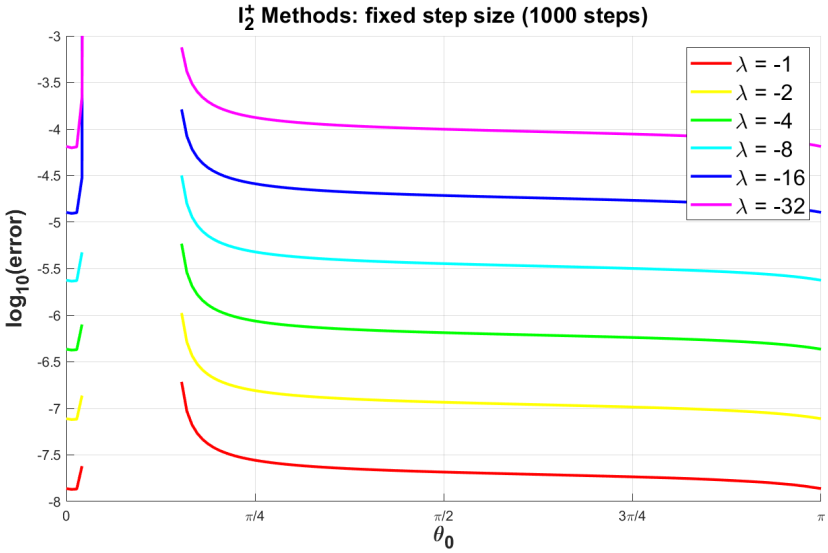

Figure 4.23: I_2^+ Methods: Accuracy of fixed step size simulation of Linear Test Equation (4.1) for different values of λ. Step size is $h = 0.01$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

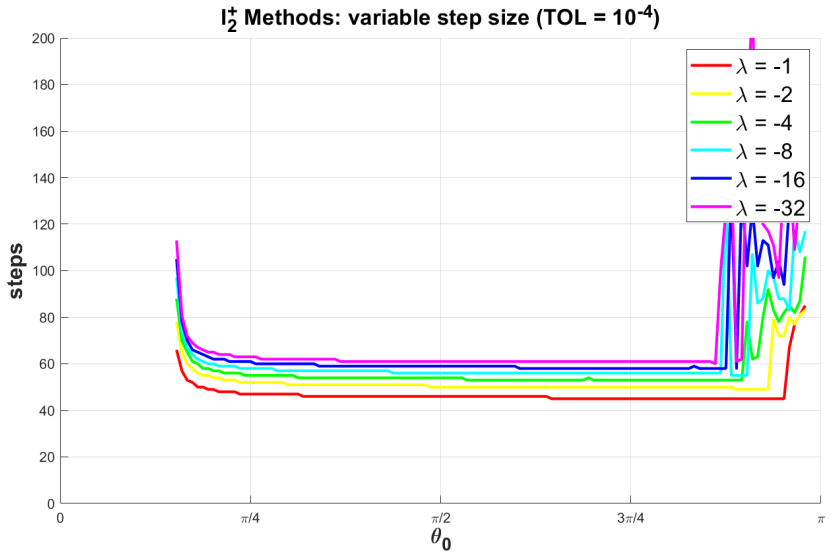

Figure 4.24: I_2^+ Methods: Number of steps in variable step size simulation of Linear Test Equation (4.1) for different values of λ with a tolerance of 10^{-4}. Step size controller is MODES' default.

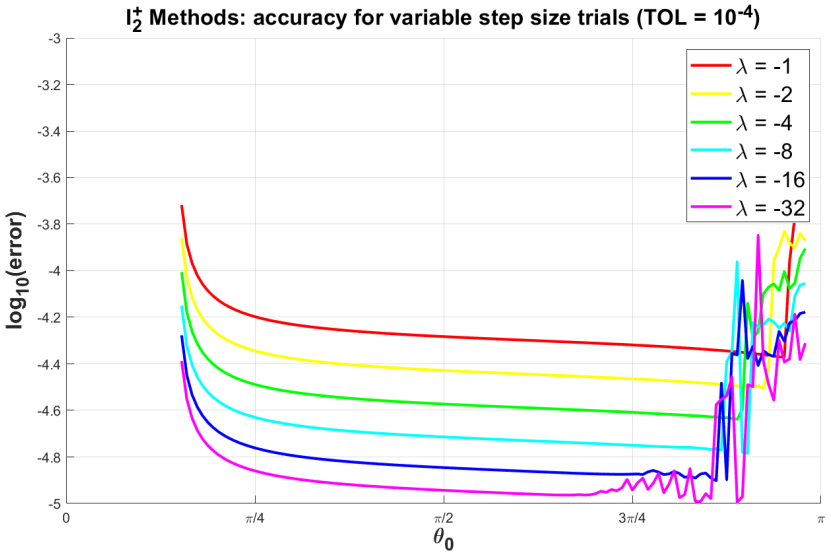

Figure 4.25: I_2^+ Methods: Accuracy of variable step size simulation of Linear Test Equation (4.1) depicted in Figure 4.24, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

We also test the implicit-plus methods with tighter tolerance. When a tolerance of 10^{-8} is chosen we see in Figure 4.26 that all methods take more steps, and the "bad behavior" experienced near $\theta_0 = \pi$ is slightly "smoothed out." In fact, for $\lambda = -1$ there are even a few methods past $\theta_0 = 0$ which are able to successfully simulate. We also see in Figure 4.27 that all methods are achieving a reasonably good accuracy that does not vary too much.

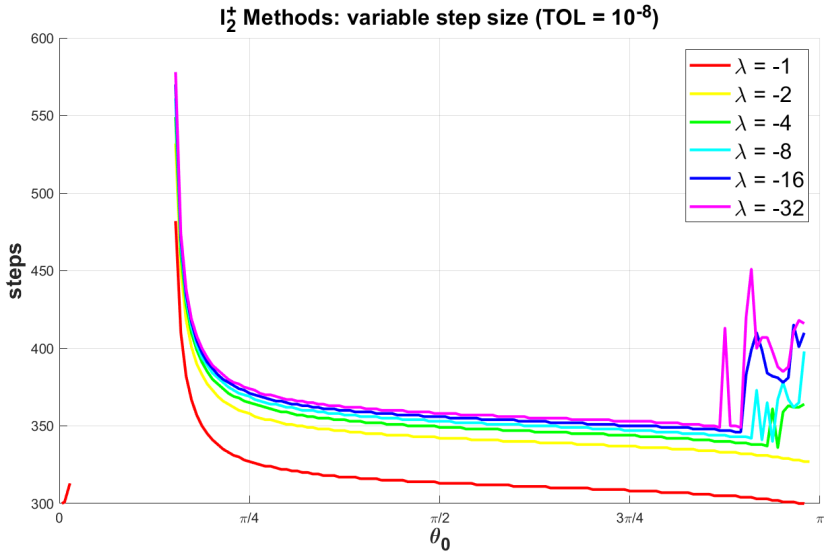

Figure 4.26: I_2^+ Methods: Number of steps in variable step size simulation of Linear Test Equation (4.1) for different values of λ with a tolerance of 10^{-8}. Step size controller is MODES' default.

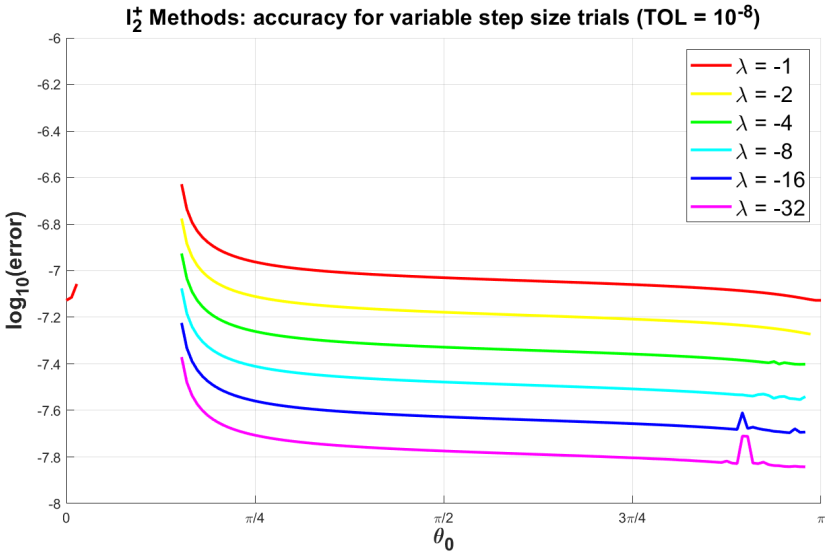

Figure 4.27: I_2^+ Methods: Accuracy of variable step size simulation of Linear Test Equation depicted in Figure 4.6, error measured using wRMS norm (4.4).

In a similar fashion to the trends observed in both the E_2 and I_1 method classes, the lines in Figures 4.23, 4.25 and 4.27 are equally spaced, since λ in the Linear Test Equation is chosen in powers of two, and the principle error term in Equation 4.2 is proportional to a power of λ.

Finally, we run the I_2^+ methods on the Prothero-Robinson problem. For the fixed step size trial, since the stiffness is chosen as $\lambda = -1$, the methods compare similarly against each other as they did with the Linear Test Equation. Adjusting the forcing parameters does sufficiently increase the difficulty which causes the simulation to be less accurate. In the variable step size case, the number of steps required is nearly smooth, however the achieved accuracy is oddly chaotic. This disruption is even visible at $\theta_0 = \pi/2$, the Adams-Moulton method, a method which has many different variable step size implementations, and so it's even more unclear what causes this and whether such a thing is potentially problematic.

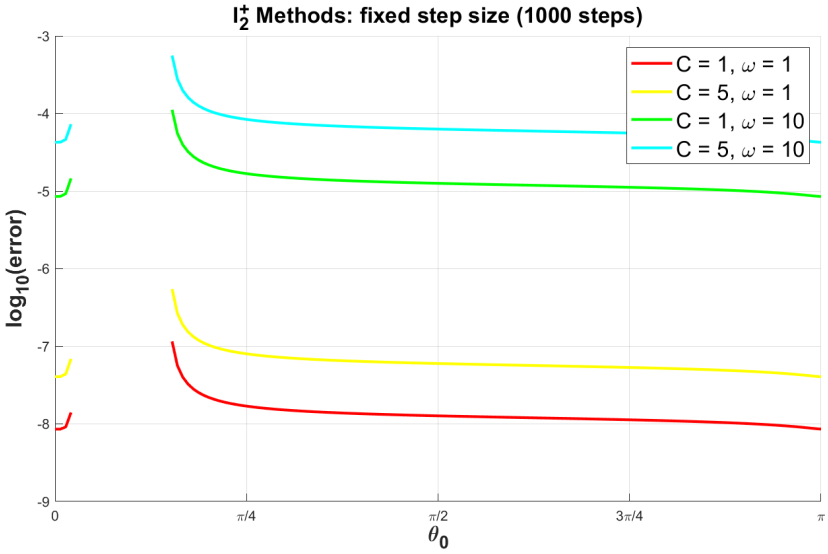

Figure 4.28: I_2^+ Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω. Step size is $h = 0.002$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

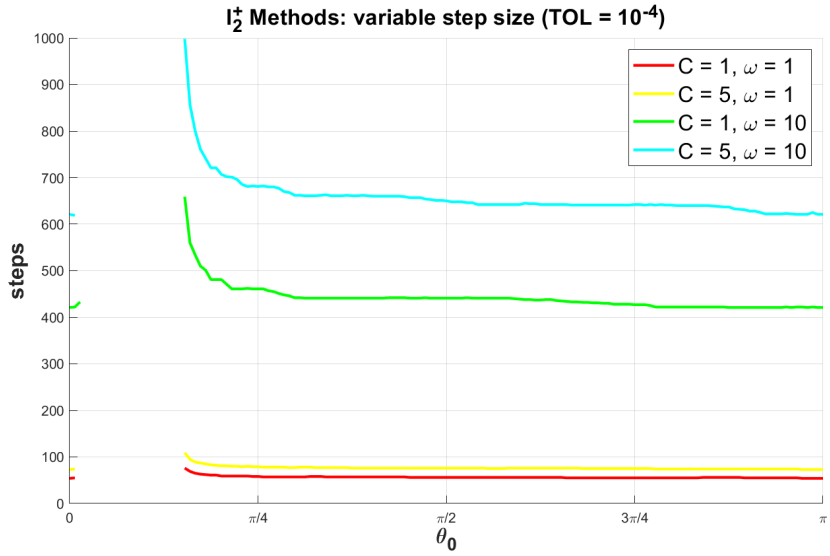

Figure 4.29: I_2^+ Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -1$, $F(t) = C\sin(\omega t)$ and different values of C and ω with a tolerance of 10^{-4}. Step size controller is MODES' default.

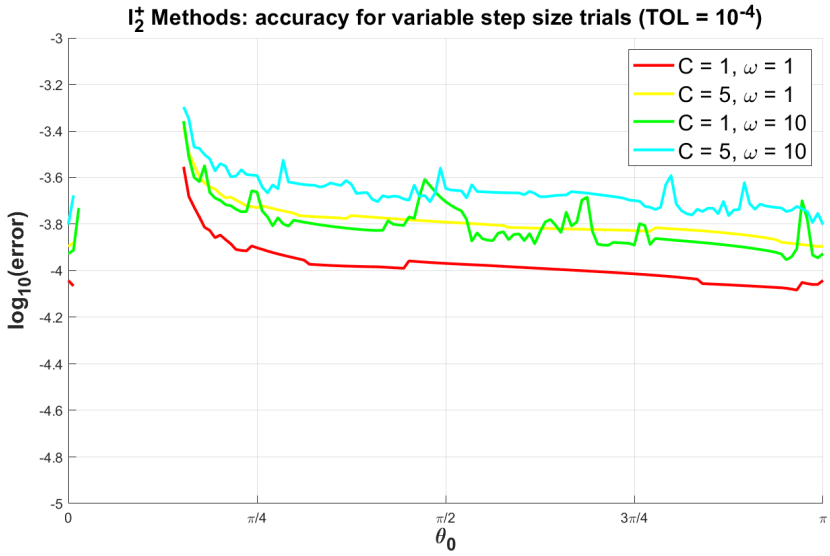

Figure 4.30: I_2^+ Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.9, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

4.4 Two Parameter Classes

There are several difficulties which must be overcome in order to analyze the methods of two parameters. First, for the one parameter methods, the interval $[0, \pi]$ is broken up into 145 points (including both endpoints) to give a selection of methods. For each method we compute its ρ polynomial and its roots as well as the error coefficient, and apply the method to a few test problems. In the two dimensional case, copying this strategy means breaking the square $[0, \pi] \times [0, \pi]$ into a 145×145 grid, which means the amount of work required is effectively squared.

In early experiments the zero stability of each method was calculated using MODES' `theta2coeff` tool and Matlab's `root` function, and error coefficient was calculated using MODES' `errorConstant` tool. In the one dimensional case computations took only a few minutes but in the two dimensional case they could take several hours. By instead determining explicit formulas for the coefficients in terms of the tangents of the θs and using the lemmas, the computation time required to produce all these graphs was reduced to just a few seconds.

However there is no shortcut to conducting experimental simulations. Excepting methods which are not zero stable, each method must be chosen and a MODES simulation must be run for a fixed step size and variable step size trial. Furthermore, since two dimensional plots are required to display the data (contour plots were chosen in this case), there is no longer an easy way to compare the results of simulating different problems on the same graph. This motivates the choice of a single problem for each method class that is "suitably difficult" for the solver so that the differences between methods from that class can be highlighted. If the problem is "too easy," the solver is simply able to ramp up the step size and many trials resolve in the exact same number of steps, making comparisons less meaningful. If the problem is "too hard," the computation time is increased and a larger proportion of methods lead to bad computations.

It is important to note that the choice of problem variables is essentially arbitrary. The Prothero-Robinson problem offers the chance to increase the complexity of the problem by increasing the stiffness and forcing function to a point where the solver must accumulate some error and perform some step size control. We choose the interval $t \in [0, 2]$ with initial value $y_0 = 10$ and $\lambda = -5$ so that some mild stiffness and damping is incurred. Since the two parameter methods are each one order higher than their one parameter counterparts, the problem must be made a little tougher, so the forcing function $F(t) = 5\sin(5\pi t)$ is chosen.

In the fixed step size trial, we would again like to measure the accuracy of a simulation of 1000 steps, and therefore the step size of 0.002 is chosen. In the variable step size trial, we would like the solver to work similarly hard, and the final control tool is the tolerance. Through experimentation the following tolerances were chosen:

- 10^{-6} for the E_3 methods.

- 10^{-4} for the I_2 methods.
- 10^{-8} for the I_3^+ methods.

It was found that these tolerances, combined with this problem, cause most stable simulations to take somewhere between 800 and 2000 steps. This is large enough to make meaningful comparisons between methods but small enough that the computer is not made to work unreasonably hard. Methods which begin to encounter stability problems can have their step counts increase rapidly, or integration can fail completely. Therefore, in order to more accurately display the differences between the more stable methods, simulation is halted at 2000 steps.

Another difficulty encountered in this section is the question of how the stability regions of methods change as θ changes continuously. Such an analysis is difficult because there is no clear way to visualize the change, since θ varies in two dimensions. One idea might be to simply measure the area of the stability region - this would make it possible to determine which stability regions are larger, but gives no further qualitative results. This could be done using integration to determine the area inside $\rho(z)/\sigma(z)$ for z in the complex unit circle, but these curves are not always simple, and when they separate the plane into multiple regions, some analysis is required to determine which region is the correct stability region. Alas, considerations of the size of the stability region will be mentioned in each section and some hypotheses will be made regarding how the stability region might change and what effects this might have, but definitive conclusions may not be reached.

4.4.1 E_3 Methods

For the E_3 methods, the coefficients can be determined by solving the four dimensional system:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 1+T_1 & T_0 & -1 \\ 4 & 1+2T_1 & 0 & -4 \\ 8 & 1+3T_1 & 0 & -12 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ \alpha_1 \\ \alpha_0 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \\ -9 \\ -27 \end{pmatrix}. \tag{4.20}$$

The coefficients are:

$$D = -3T_1T_0 + T_1 + 2T_0 - 1 \tag{4.21a}$$
$$D\alpha_2 = 12T_1T_0 - 9T_1 + 8T_0 - 6 \tag{4.21b}$$
$$D\alpha_1 = -4T_0 + 3 \tag{4.21c}$$
$$D\alpha_0 = 5T_1 - 2 \tag{4.21d}$$
$$D\beta_2 = -23T_1T_0 + 6T_1 + 22T_0 - 12 \tag{4.21e}$$
$$D\beta_1 = -DT_1\alpha_1 = 4T_1T_0 - 3T_1 \tag{4.21f}$$
$$D\beta_0 = -DT_0\alpha_0 = -5T_1T_0 + 2T_0. \tag{4.21g}$$

Using Lemma 4.1.3 we can determine when the method is zero stable from the coefficients a_1 and a_0. The roots of the ρ polynomial are one and:

$$\frac{(\alpha_1 + \alpha_0) \pm \sqrt{(\alpha_1 + \alpha_0)^2 + 4\alpha_0}}{2}. \tag{4.22a}$$

The method is stable when both these roots are less than one in magnitude. The explicit formulas for these roots and the error coefficient are messy, and so they are omitted. The second largest root of ρ is plotted in Figure 4.31; where it is one, the method is not zero stable. The error coefficients are plotted in Figure 4.32. Those which are greater than 1 are truncated.

By overlaying these two plots, we see that within the zero stability region, the error coefficient is reduced for methods towards the lower-right corner, "wrapping around" towards the sliver of zero stable methods which exist in the lower-left corner. The lower bound for the error coefficient of zero stable methods was not determined, but experimental evidence suggests that in this corner, it is around 0.11. There are also a couple small "islands" of zero stability past this corner where the error coefficient is slightly smaller, possibly becoming as small as 0.09. These islands likely appear as a result of "roundoff error" due to the gridding scheme described at the beginning of the chapter. The zero stability plot was recreated in higher resolution with θ chosen in increments of $\pi/1000$ and this area appeared as a single, thin island which does not quite touch the corner, although this plot is not shown.

Similarly to how there exists an E_2 method with zero error coefficient which is not zero stable, there is a one-dimensional set of E_3 methods with zero error coefficient, none of which are zero stable. It is possible to determine the set of θs for which this occurs by setting the error coefficient formula equal to zero:

$$EC = \frac{1}{24 \sum \beta_j} \left(3^4 + 2^4\alpha_2 + \alpha_1 - 4 \cdot 2^3\beta_2 - 4\beta_1\right) = 0. \tag{4.23}$$

The fraction on the outside can be multiplied out, and the formulas for the coefficients in terms of θ can be substituted in. The coefficients have a common denominator which can be multiplied through and after collecting like terms the formula simplifies to the much more manageable

$$27T_1T_0 - 9T_1 - 14T_0 - 6 = 0, \tag{4.24}$$

which we can factor and write as

$$T_0 = \frac{9T_1 - 6}{27T_1 - 14}. \tag{4.25}$$

By using Lemma 4.1.3 we can now find out "how far away" these methods are from methods which are zero stable. We substitute out T_1 in the formulas for α_1 and α_0 and through a miracle of cancellation we find:

$$\alpha_1 = -9$$
$$\alpha_0 = \frac{1}{3T_0 - 1}.$$

Although not particularly useful, it is quite interesting to find that the α_1 coefficient is a constant -9! The nontrivial roots of the ρ polynomial are therefore minimized for $\alpha_0 = -\alpha_1 = 9$, where they are ± 3.

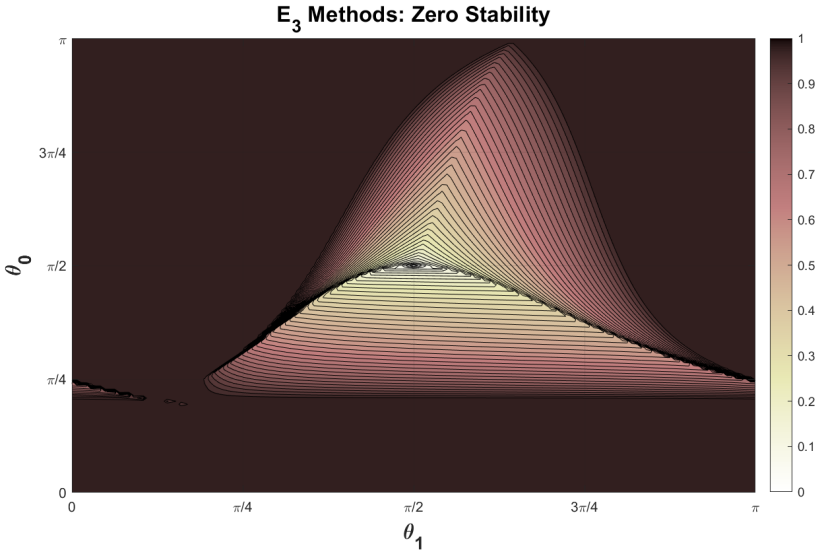

Figure 4.31: E_3 Methods, Zero Stability: The second largest root of $\rho(z)$ is plotted, methods are stable where it is less than one. At $\theta = (\pi/2, \pi/2)$ is the AB-3 method.

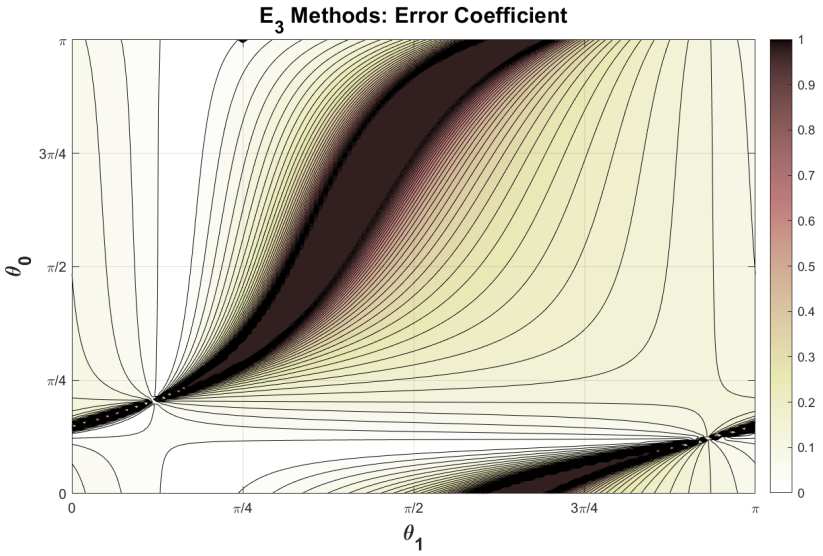

Figure 4.32: E_3 Methods, Error Coefficient: The absolute value of the error coefficient is plotted where it is less than one. At $\theta = (\pi/2, \pi/2)$ is the AB-3 method.

The next task is to simulate the Prothero-Robinson test problem for the E_3 methods. In the fixed step size experiment, we see in Figure 4.33 that the accuracy is correlated to the error coefficient, with methods towards the lower-right corner of the zero stability region achieving higher accuracy. Methods with total error greater than one are omitted.

Next, the E_3 methods are applied to the test problem with a variable step size and tolerance of 10^{-6}. We find in Figure 4.34 for these parameters that there is a correlation between step count and error coefficient. In Figure 4.35 it is shown that the methods with smaller error coefficients also achieve better accuracy.

We hypothesize for the E_3 methods that a general trend exists where methods with a smaller error coefficient also have smaller stability regions, similar to the trend in E_2 methods. However for this experiment it is inconclusive whether or not this is true. Since there are now two dimensions to choose from, it might be possible to find a method with larger stability region and smaller error coefficient than another given method. It may be the case that choosing a stiffer problem, or a simpler problem with a lower tolerance, could cause methods with large stability regions to experience better performance. This could be one way to observe what trends exist for how the stability region changes. However, no such definitive analysis has yet been made. Such is a relevant topic for future research.

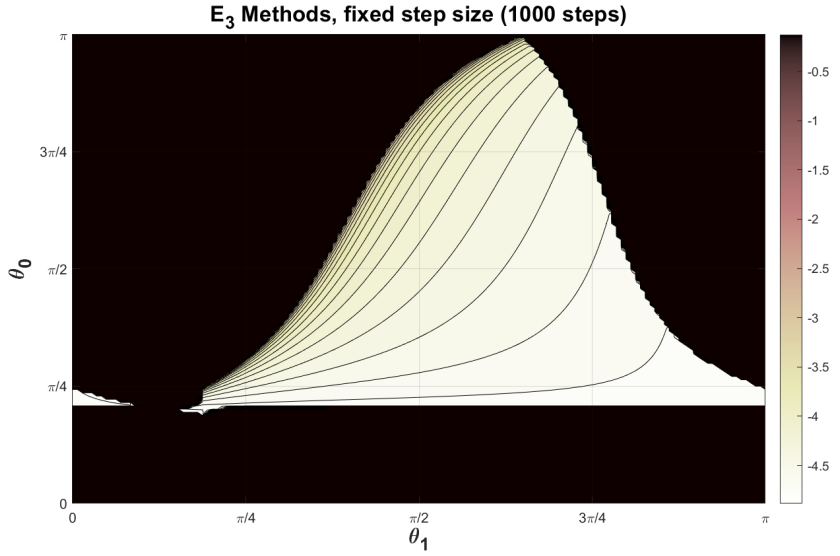

Figure 4.33: E_3 Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$. Step size is $h = 0.002$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

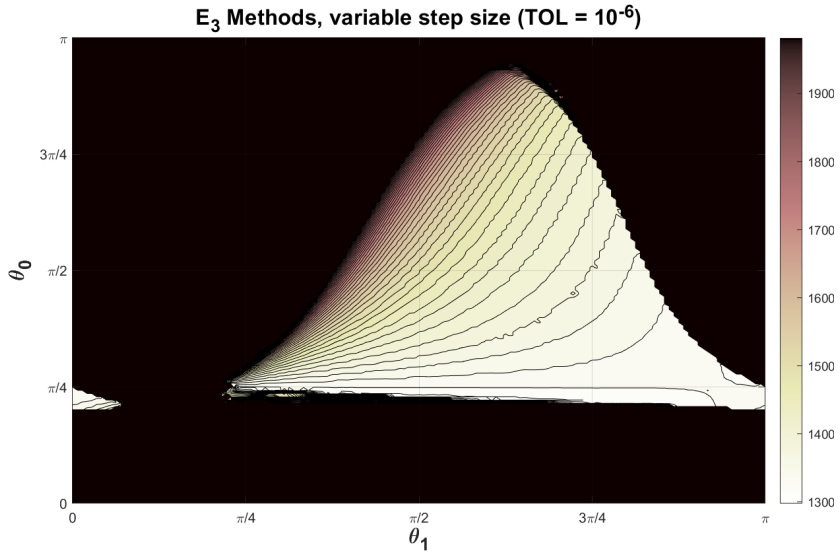

Figure 4.34: E_3 Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$ with a tolerance of 10^{-6}. Step size controller is MODES' default.

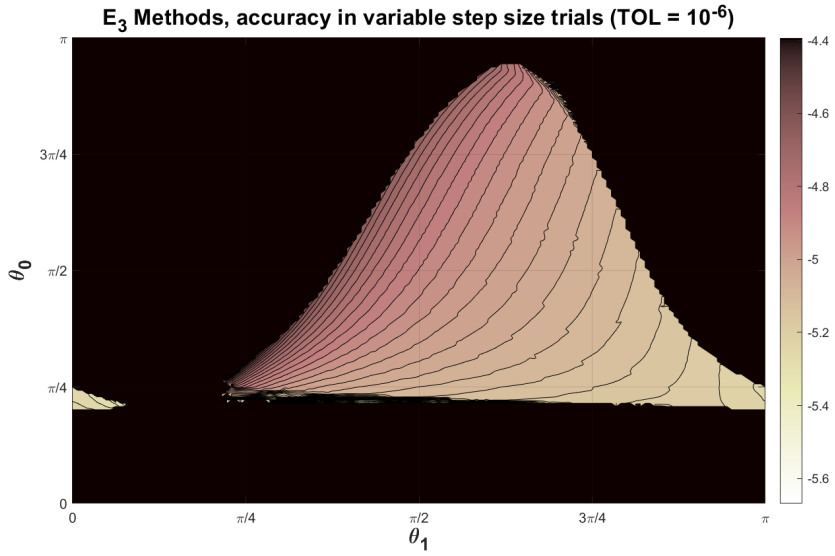

Figure 4.35: E_3 Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.34, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

4.4.2 I_2 Methods

We now turn our attention to the I_2 methods. As these are order two methods, the formulas for the coefficients can be determined by solving a three dimensional system:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1+T_1 & T_0 & -1 \\ 1+2T_1 & 0 & -4 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_0 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ -4 \end{pmatrix}. \tag{4.26}$$

After inversion, we compute the coefficients:

$$D = 2T_1 - 4T_0 + 3 \tag{4.27a}$$
$$D\alpha_1 = 4T_0 - 4 \tag{4.27b}$$
$$D\alpha_0 = -2T_1 + 1 \tag{4.27c}$$
$$D\beta_2 = 2T_1T_0 - 3T_0 + 2 \tag{4.27d}$$
$$D\beta_1 = -DT_1\alpha_1 = -4T_1T_0 + 4T_1 \tag{4.27e}$$
$$D\beta_0 = -DT_0\alpha_0 = 2T_1T_0 - T_0. \tag{4.27f}$$

To determine zero stability, by Lemma 4.1.2 we require the absolute value of α_0. This is plotted in Figure 4.36 where it is less than one. The error coefficient is plotted in Figure 4.37.

There are several "degenerate" sets of methods living in the I_2 class. For $T_1 = 1/2 \Rightarrow \theta_1 = \arctan(1/2)$ the α_0 and β_0 coefficients disappear and the

method degenerates to a one step method. In fact, the formulas for the other coefficients simplify and become $\alpha_1 = -1$, $\beta_2 = 1/2$, $\beta_1 = 1/2$. This is the Trapezoidal Rule, disguised as a two step method which ignores the furthest point. The fact that all these degenerate one step methods must be the Trapezoidal Rule is also provable from the fact that all methods produced from the I_2 parametrization must be order two.

In the I_1 methods, the Trapezoidal Rule is the single method with zero error coefficient which attains order two. This method can be thought of as the unique I_1^+ method. In the I_2 methods, there is a one dimensional set of methods with zero error coefficient, therefore attaining order three. Unlike in the explicit classes, some of these methods are zero stable. These are I_2^+ methods, hidden in the I_2 class. To find these methods, we again set the error coefficient equal to zero. Removing the extra factors, this gives the equation

$$2^3 + \alpha_1 - 3 \cdot 2^2 \beta_2 - 3\beta_1 = 0. \tag{4.28}$$

The formulas for the coefficients are substituted in, and the common denominator is multiplied through. After like terms are collected, we arrive at the harmless equation

$$3T_1T_0 - T_1 - 2T_0 - 1 = 0, \tag{4.29}$$

which can be factored and rewritten as

$$T_0 = \frac{T_1 - 1}{3T_1 - 2}. \tag{4.30}$$

Another interesting phenomenon is that there is a degenerate set of explicit methods that lie within the I_2 class. This can be compared to how the Explicit Euler method, the unique E_1 method, was present as a degenerate I_1 method. Finding these explicit methods is relatively easy: all we have to do is set $\beta_2 = 0$. From the coefficient formula this means

$$2T_0T_1 - 3T_0 + 2 = 0. \tag{4.31}$$

This can also be factored and written as

$$T_0 = \frac{-2}{2T_1 - 3}. \tag{4.32}$$

Part of this line also lies within the zero stability region. Bear in mind that due to the gridding scheme described at the beginning of this chapter, the I_2^+ and E_2 methods are not being depicted exactly in these figures, but rather nearby methods in multiples of $\pi/144$ are being chosen.

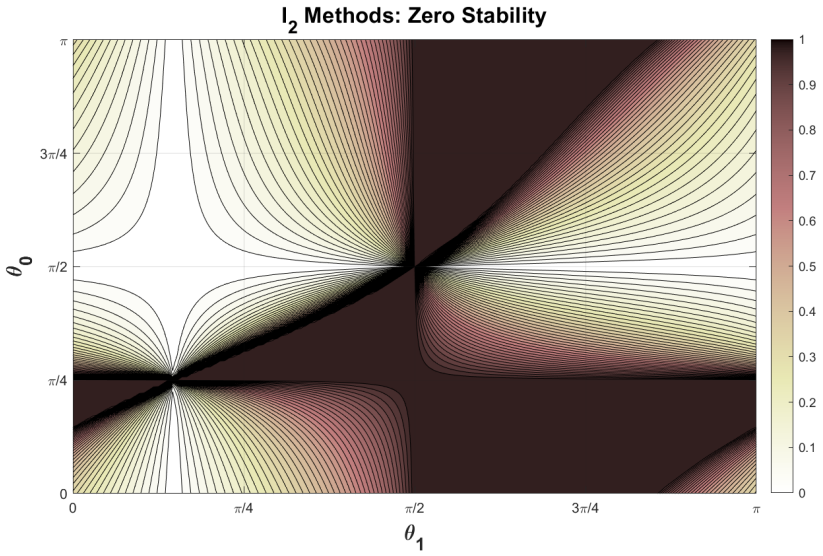

Figure 4.36: I_2 Methods, Zero Stability: The second largest root of $\rho(z)$ is plotted, methods are stable where it is less than one. At $\theta = (0, 0)$ is the BDF-2 method.

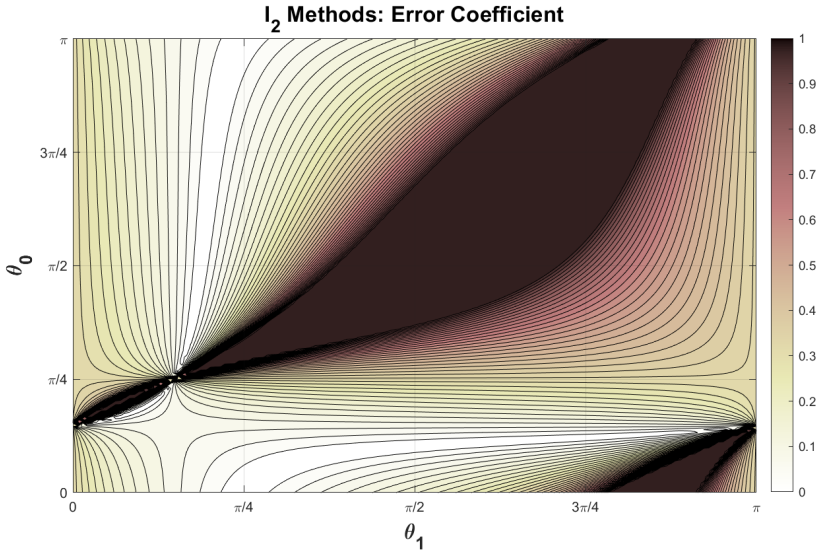

Figure 4.37: I_2 Methods, Error Coefficient: The absolute value of the error coefficient is plotted where it is less than one. At $\theta = (0, 0)$ is the BDF-2 method.

Simulations with the I_2 methods for fixed step size are shown in Figure 4.38. For the fixed step size trials, the results are quite well behaved and match error coefficient calculations. There is a curve on the left side where the methods perform exceptionally well. These are the (neighbors of the) I_2^+ methods that were identified earlier.

In the variable step size trials in Figure 4.39, we see a region near the center-top where the methods achieve very few steps. It turns out that, in a similar situation to the I_1 case, these are the (neighbors of the) degenerate explicit methods. This means that the error estimator is failing for these methods and measuring zero or near-zero error, causing the step size to be ramped up in every step. This happens for the same reason given in Section 4.3.2. When we measure and plot the achieved accuracies in Figure 4.40, we see in a similar fashion to the I_1 class that the higher order methods are achieving small errors, while the degenerate explicit methods have high errors.

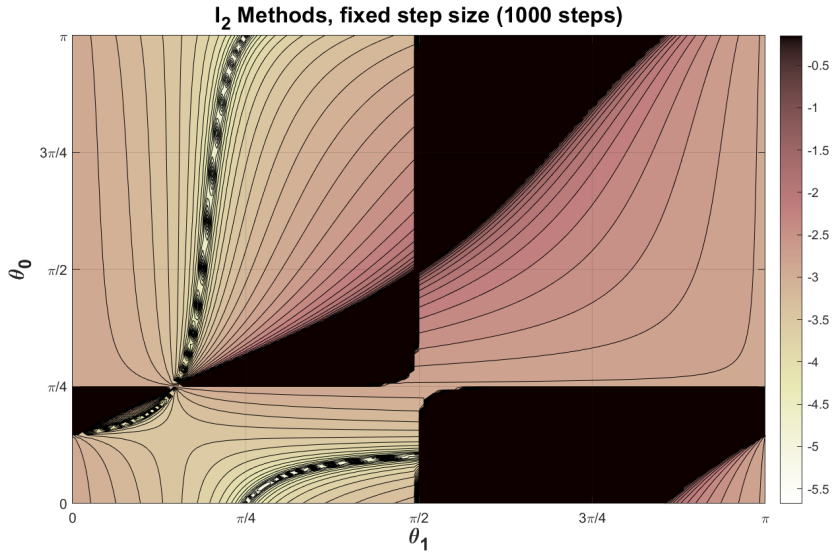

Figure 4.38: I_2 Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$. Step size is $h = 0.002$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

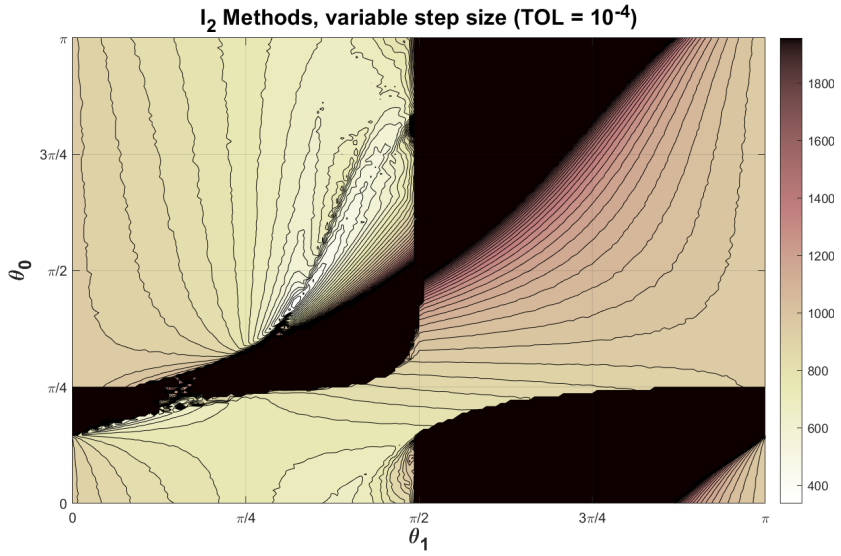

Figure 4.39: I_2 Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$ with a tolerance of 10^{-4}. Step size controller is MODES' default.

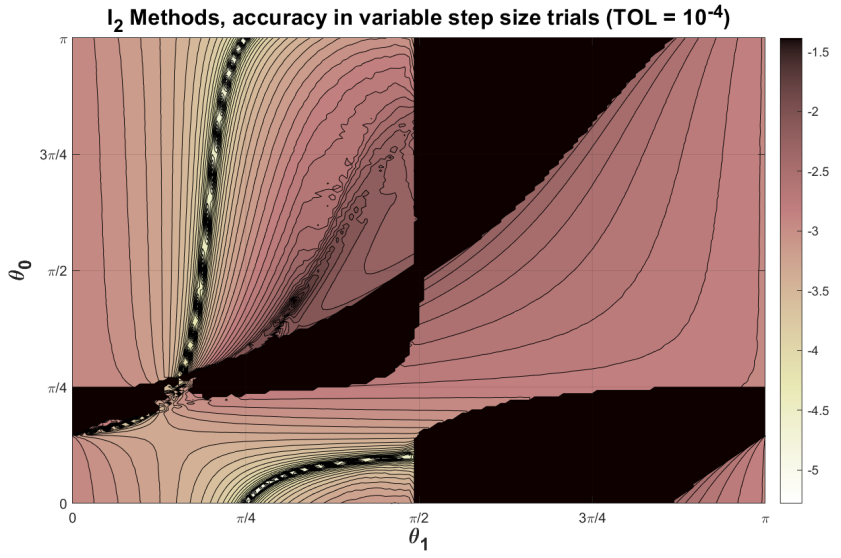

Figure 4.40: I_2 Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.39, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

One final question to answer is which I_2 methods are suitable for stiff problems. Some methods such as BDF-2 have unbounded stability regions in the left half-plane. However in the I_1 class we observed that not all methods have unbounded stability regions and it is reasonable to expect that there may also be I_2 methods which are not stiff suitable. We can determine *experimentally* which methods can be applied to stiff problems by simulating the Linear Test Equation with large negative λ with all methods. The stiffness value of $\lambda = -200$ is chosen, with $y_0 = 1$ and $t \in [0, 10]$ as before. We also perform both fixed and variable step size trials.

In the fixed step size trial in Figure 4.41 we witness a very interesting phenomenon: The accuracies attained still reflect the error coefficients, but some methods which are not zero stable are now able to successfully simulate! Recall that for the Linear Test Equation, increasing stiffness is equivalent to increasing the step size. It turns out that in the I_2 class there are methods which are not zero stable but which have stability regions in the negative half-plane which are close to, but bounded away from zero, and therefore are stable for sufficiently *large* $h\lambda$. When the stiffness of the problem is increased, some of these methods become stable for the given step size. We find, counter to intuition, that for these methods the error is actually reduced by *increasing* the step size. An example of one such method will be examined at the end of this chapter.

But before moving on, we test the stiff Linear Test Equation with variable step size. A tolerance of 10^{-4} is chosen. In Figure 4.42 we show the number of steps each method takes and in Figure 4.43 we show the accuracies of each method. We observe a "dividing line" emerging precisely along the order three methods: to the right of this line, methods are taking thousands of steps, likely indicating that the step size must be greatly reduced in order to achieve stability. This region includes the explicit methods. To the left of this line, methods take only about 100 steps, indicating that stability is no issue. We remark that the I_2^+ methods themselves should not be stiff suitable, however methods near this set seem to be able to achieve high accuracy while taking very few steps.

It is interesting to note the pattern here, which was also present in the I_1 methods. There are two separate, connected sets of methods, one being the methods which have unbounded stability regions in the negative half-plane and are suitable for stiff problems, the other being methods with bounded stability regions which are not suitable for stiff problems. These regions are separated by the one-dimension-lower set of higher order methods, and the degenerate set of explicit methods live on the side of the methods with bounded stability regions. It would be interesting to investigate this pattern for higher order I_k methods.

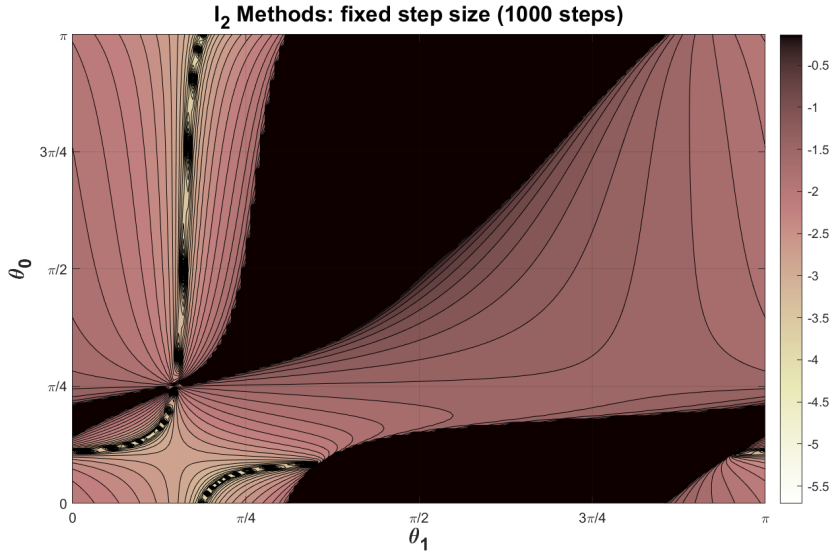

Figure 4.41: I_2 Methods: Accuracy of fixed step size simulation of stiff Linear Test Equation (4.1) for $\lambda = -200$. Step size is $h = 0.01$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

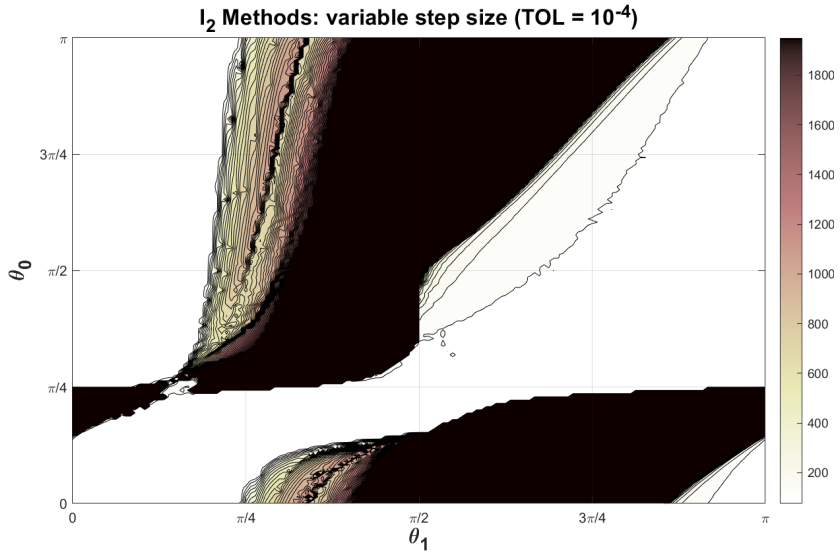

Figure 4.42: I_2 Methods: Number of steps in variable step size simulation of stiff Linear Test Equation (4.1) for $\lambda = -200$ with a tolerance of 10^{-4}. Step size controller is MODES' default.

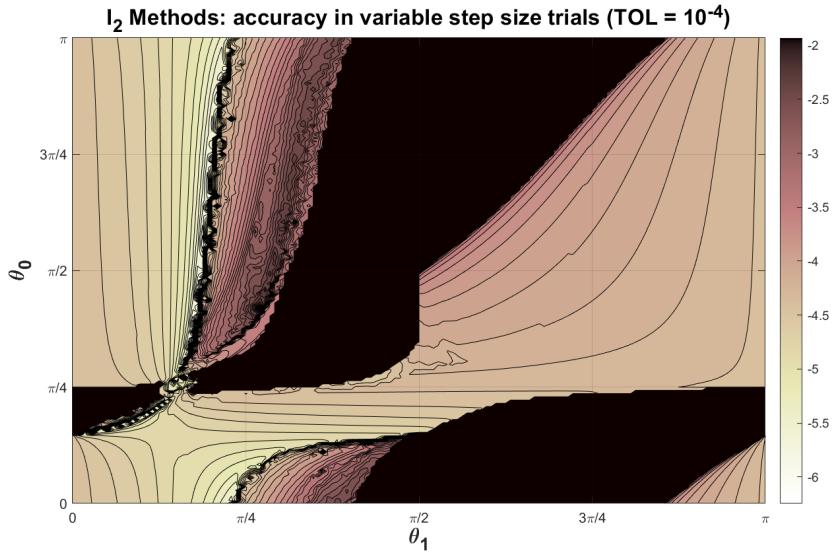

Figure 4.43: I_2 Methods: Accuracy of variable step size simulation of stiff Linear Test Equation (4.1) depicted in Figure 4.42, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

4.4.3 I_3^+ Methods

The coefficients for the I_3^+ methods are found by solving the system:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 1+T_1 & T_0 & -1 & -1 \\ 4 & 1+2T_1 & 0 & -6 & -4 \\ 8 & 1+3T_1 & 0 & -27 & -12 \\ 16 & 1+4T_1 & 0 & -108 & -32 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ \alpha_1 \\ \alpha_0 \\ \beta_3 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \\ -9 \\ -27 \\ -81 \end{pmatrix}. \tag{4.33}$$

The formulas are:

$$D = -72T_1T_0 + 24T_1 + 39T_0 - 17 \tag{4.34a}$$
$$D\alpha_2 = 72T_1T_0 - 27T_1 - 24T_0 + 9 \tag{4.34b}$$
$$D\alpha_1 = -15T_0 + 9 \tag{4.34c}$$
$$D\alpha_0 = 3T_1 - 1 \tag{4.34d}$$
$$D\beta_3 = -27T_1T_0 + 9T_1 + 14T_0 - 6 \tag{4.34e}$$
$$D\beta_2 = -57T_1T_0 + 18T_1 + 39T_0 - 18 \tag{4.34f}$$
$$D\beta_1 = -DT_1\alpha_1 = 15T_1T_0 - 9T_1 \tag{4.34g}$$
$$D\beta_0 = -DT\alpha_0 = -3T_1T_0 + T_0. \tag{4.34h}$$

The zero stability region is plotted in Figure 4.44. This region is much larger than the zero stability regions of the E_3 and I_2 methods. This means that there

are many more methods to choose from. However, we will soon see that not all of them are good quality.

The error coefficients are plotted in Figure 4.45. Since the coefficients in the denominator D are large, and since the error coefficient formula includes the reciprocal of $(p+1)!\sum\beta_j$, the error coefficients are very small for a majority of methods. In fact, most error coefficients are below 0.05, except near the curve where $D=0$. This trend can be compared to the one parameter classes, where in contrast to the explicit and implicit methods, the implicit-plus methods have a large zero stability region and small, flat error coefficients.

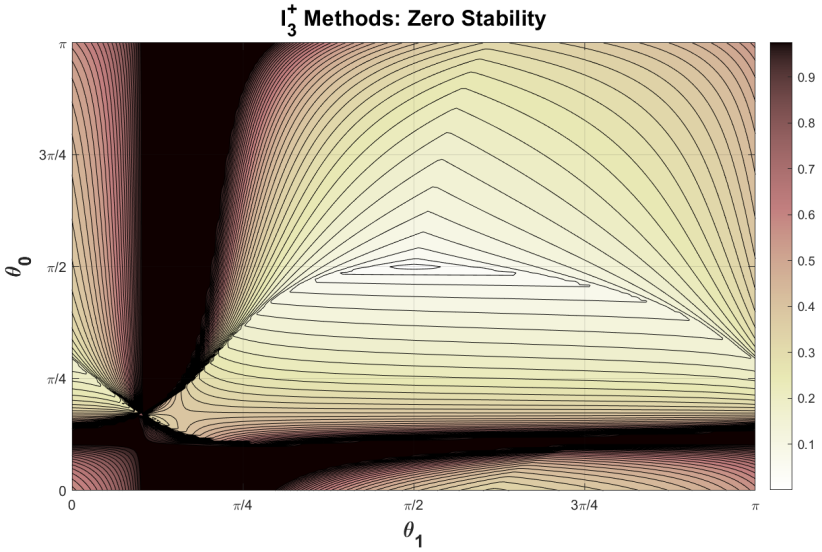

Figure 4.44: I_3^+ Methods, Zero Stability: The second largest root of $\rho(z)$ is plotted, methods are stable where it is less than one. At $\theta=(\pi/2,\pi/2)$ is the AM-3 method.

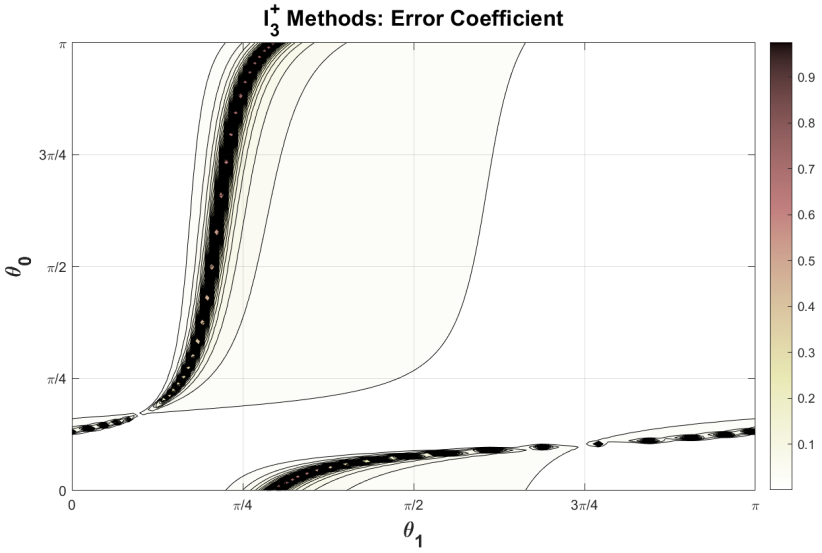

Figure 4.45: I_3^+ Methods, Error Coefficient: The absolute value of the error coefficient is plotted where it is less than one. At $\theta = (\pi/2, \pi/2)$ is the AM-3 method.

As before, we run fixed and variable step size trials. The results of the fixed step size trial are plotted in Figure 4.46. We see that the methods are far from identical. In the large central region, methods perform somewhat similarly, with high accuracies which improve slightly towards the right, wrapping around. However for methods in the center-top region, the simulation fails. The reason for this is not clear.

For example, if we look at the method $\theta = (\pi/2, 0)$, we see that the fixed step size coefficients are:

$$\alpha = (1.000, -1.125, 0.000, 0.125),$$
$$\beta = (0.375, 0.750, -0.375, 0.000).$$

The roots of $\rho(z)$ are $\{1, 0.4215, -0.2965\}$ so it is certainly zero stable. In fact, the stability region is large enough to include -2.5 and the error coefficient is 0.333. While there is nothing obvious to explain why integration fails with this particular method, even applying it to a simple problem such as the linear test equation results in the `modes` error "polynomial coefficients are not finite." When the step size is allowed to vary, the error "too small step size reached" is encountered. It is not entirely clear what causes this problem, and so we leave the question open as to why these methods are not viable, at least in this application.

In the variable step size trials, a similar pattern emerges. As shown in Figure 4.47, step counts generally decrease slightly to the right, so long as they are chosen within this "feasible" region. Measuring accuracies in Figure 4.48, it

is also hard to see any improvement in one method over another. For example, it may be possible to find improvements by selecting $\theta = (0, \pi/2)$ over the Adams-Moulton method at $\theta = (\pi/2, \pi/2)$. This method has a smaller error coefficient and has performed better in our simple experiments, but also has a smaller stability region and lacks the same amount of numerical damping.

Of all the method classes examined in this paper, the I_3^+ methods are the most difficult to draw any clear conclusions from. There is no obvious trade-off between error coefficient and stability, no clear advantage to selecting methods with one property or the other, and no indication of why some methods produce unacceptable results or fail to simulate. Perhaps there may be a way to choose methods which perform definitively better than Adams-Moulton in specific scenarios, but this question is left open to future research.

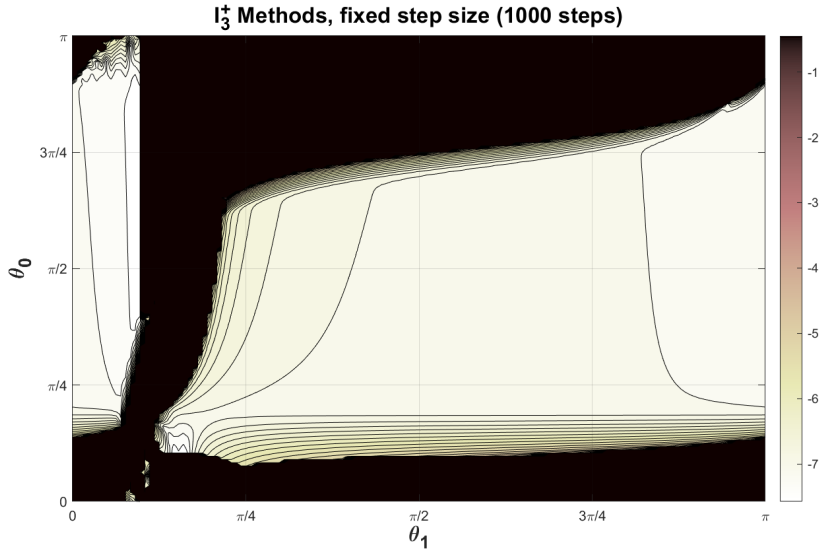

Figure 4.46: I_3^+ Methods: Accuracy of fixed step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$. Step size is $h = 0.002$, global error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

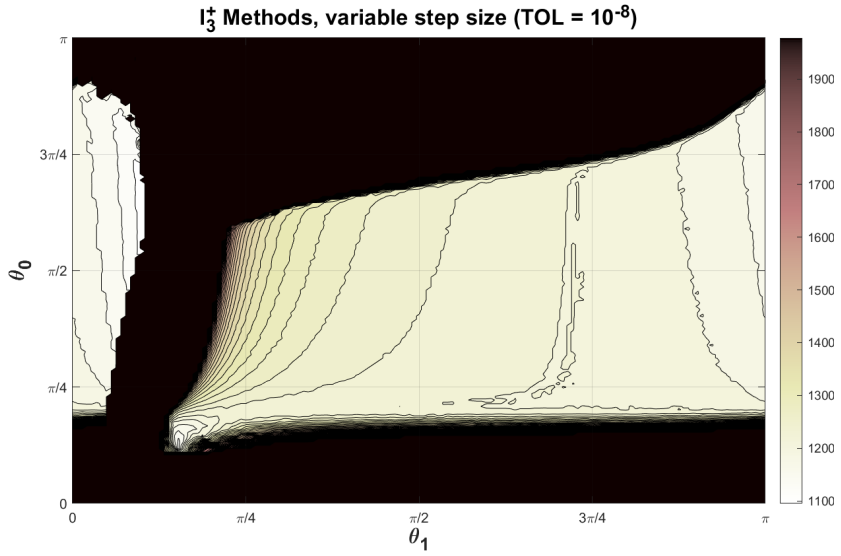

Figure 4.47: I_3^+ Methods: Number of steps in variable step size simulation of Prothero-Robinson Problem (4.3) for $\lambda = -5$, $F(t) = 5\sin(5t)$ with a tolerance of 10^{-8}. Step size controller is MODES' default.

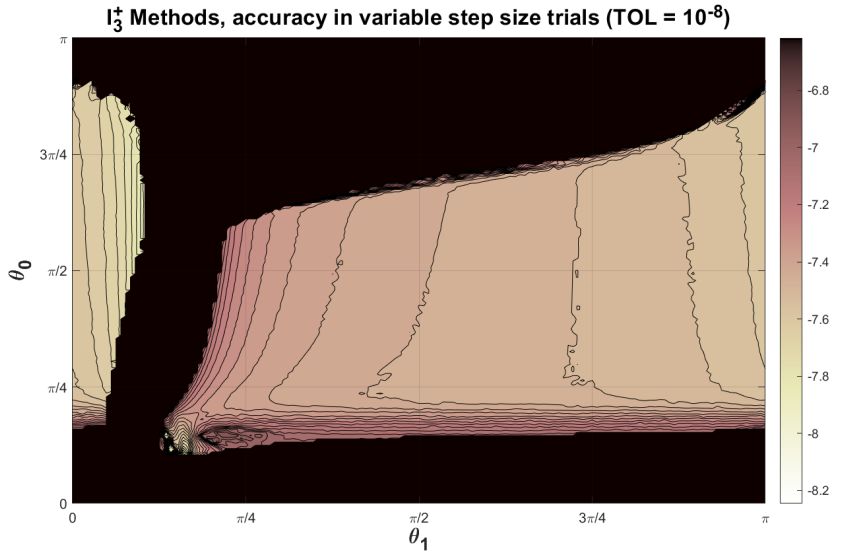

Figure 4.48: I_3^+ Methods: Accuracy of variable step size simulation of Prothero-Robinson Problem (4.3) depicted in Figure 4.47, error measured using wRMS norm (4.4), $\log_{10}$(error) plotted.

4.4.4 Methods Which Are Stable Away From Zero

Traditionally, methods which are not zero stable are never used. The reason for this is because in order to ensure the possibility of controlling error by reducing step size, it is important that the step size is allowed to become arbitrarily small without leaving the stability region of the method. However, the stability question is slightly more nuanced than this. It is possible for a method to be stable in some region of the negative half-plane which is bounded away from zero. It is possible for such a method to be stable when the step size is sufficiently large, but become unstable as the step size is decreased. In the I_2 class we witnessed that some methods became stable for the Linear Test Equation when the stiffness was increased, which is equivalent to choosing a larger step size h. In this section, we examine one such method. It has not been determined if such methods exist in E_k or I_k^+ classes, or even in I_k classes for any other k.

Consider the I_2 method given by the parameters $(\theta_1, \theta_0) = (1.5, 1.0)$. The fixed step size formula for this method has coefficients:

$$\alpha = (1.0000, 0.0893, -1.0893),$$
$$\beta = (1.6518, -1.2590, 1.6965),$$

which can be verified by the Parametric Equivalence Theorem. The characteristic polynomial for this method is then given by

$$\pi(z; h\lambda) = \sum_{k=0}^{2} (\alpha_k - h\lambda\beta_k) z^k. \tag{4.35}$$

The ρ polynomial is $\rho(z) = 1 + 0.0893z - 1.0893z^2$, which has roots $\{1, -1.0893\}$ and therefore by the Root Condition the method is not zero stable. However for $h\lambda = -0.05$ the characteristic polynomial (4.35) becomes $\pi(z; -0.05) = 1.0826 + 0.0263z - 1.0045$ with roots $\{0.9755, -0.9512\}$ and therefore the method is stable for this value of $h\lambda$. In fact, by performing a simple line search, we find that the boundary for the stability region on the negative real axis is somewhere around $h\lambda = -0.0388$, and therefore the method is stable for some step sizes to the left of this value, and unstable for step sizes to the right.

We simulate the Linear Test Equation $y' = -y$ with initial value $y(0) = 1$ on the interval $t \in (0, 10)$ with this method and fixed step size $h = 0.05$ to obtain the completely reasonable result in Figure 4.49. However when the step size is reduced to just $h = 0.025$ we see in Figure 4.50 that instability is causing rapidly accelerating oscillations as the simulation diverges.

The takeaway from this experiment is not that we should begin considering methods which are not zero stable, but rather that stability is a slightly more complex issue than is usually considered. Note that we have not examined error coefficient, and are not claiming that this method, or any other non-zero stable method, should perform better than zero stable methods. This method does not even have an unbounded stability region in the left half-plane and therefore is not suitable for highly stiff problems. Zero stable methods are necessary for

convergence, which is important for general applications in solvers which need to be able to account for a wide variety of problems and which may require the step size to become extremely small. However in experimental settings with a simple problem, with the step size being either fixed or bounded below, it is important to note that the choices of methods is not determined *precisely* by the property of zero stability, but instead by the requirement that the methods chosen are stable for the given values of $h\lambda$. Since the characteristic polynomials, and therefore stability regions of methods vary continuously with respect to the θ parameters, methods such as the one above, which is stable in a region near zero, will be located near the zero stability region in θ space, so a nearby zero stable method could be chosen anyways. However, it is interesting to discover that such methods exist, even if the plausibility of using them is dubious.

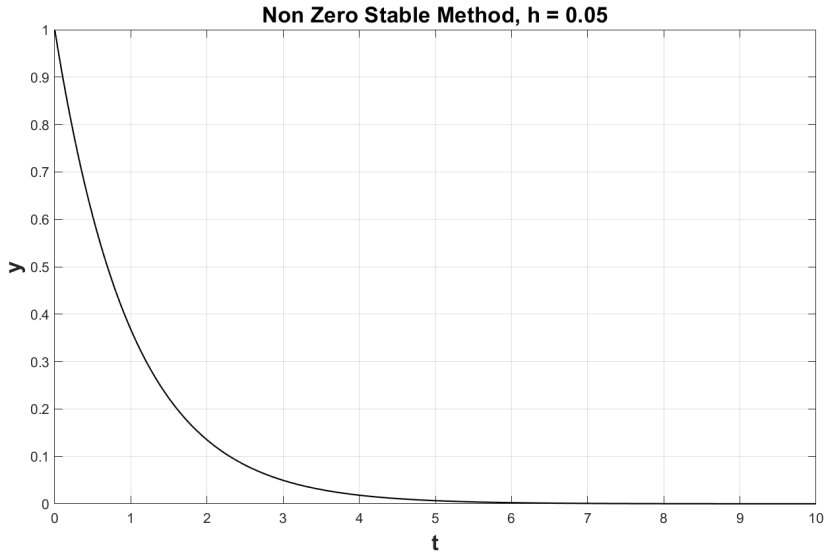

Figure 4.49: Fixed step size simulation of Linear Test Equation $y' = -y$ using the non zero stable method with step size $h = 0.05$. The step size is sufficiently large such that $h\lambda = -0.05$ is within the stability region of the method, and the simulation is stable.

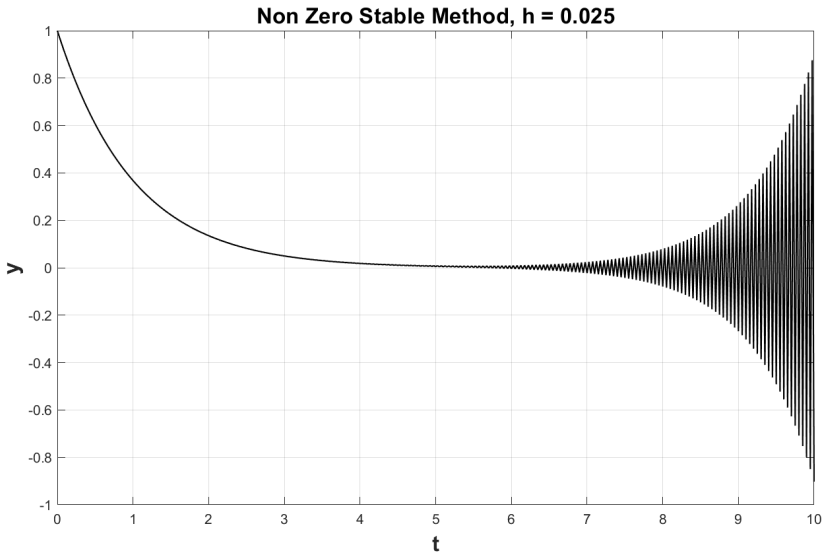

Figure 4.50: Same test as in Figure 4.49 except with a step size of $h = 0.025$. Since $h\lambda = -0.025$ is outside of the stability region of the method, the simulation is unstable.

Chapter 5

Further Study

5.1 Difficulties

In this paper many graphs are presented which show some kind of trend or make it possible to draw some experimental conclusions. In reality, many more experiments were run which were inconclusive or not illustrative. Therefore, to some extent the information presented here is "selected" in order to support conclusions. We argue that this decision is justified since these patterns emerged in many other experiments, although other experiments had either presentational or computational issues. In some cases there was no good way to show the results, in other cases only partial results would be clear. Furthermore some experiments were extremely computationally intensive, especially when looking at the methods of two parameters, since so many methods had to be tested in order to form the plots, and some experiments failed completely. This limitation meant that other factors had to be kept simple, and that small issues in codes could make it necessary to run the entire computation again.

Finally and most importantly, this research was truly experimental. Going into this subject, there was never a clear vision of what would be found, or sometimes even where to start, since there is so little known about it. The ultimate question of how to build continuous method changes into a solver has been largely circumvented because even equipped with an understanding of the theoretical properties of these methods, there is still no clear way to go about this. An error estimator which controls step size will automatically account for stability issues, but even with an error estimate, how should a solver know whether the results can be improved by choosing a method with a larger stability region, one with a smaller error coefficient, or some other property which can automatically determine method choice? In what direction should the θ vector travel, and by how much? Such a framework has yet to be established.

5.2 Higher Order Methods

The research presented here focused specifically on low order methods with few parameters because these classes were the easiest to navigate visually and computationally. While we observed certain trends in these classes and were able to make experimental conclusions regarding how methods compared to other methods of the same order and class, we were not able to generalize these results to higher orders. Since higher order methods are often useful in practical computing, it would be valuable to obtain general results regarding the higher order classes in order to assess the possibility of applying these methods to more generalized problems. For a start, it would be useful to find a quick way to determine the zero stability regions and error coefficient gradients for any method class, as well as knowing how the stability regions change. This would make it possible to navigate the θ space easily for classes of any order, at least with respect to these properties. However, even obtaining simpler results about higher order methods could be a good start. For example, BDF methods are popular implicit methods for their ability to tolerate stiff problems and produce accurate results. However, could there be other methods of orders 3 to 6 which are stiff tolerant and produce smaller errors than the BDF family? Are there similarly stiff suitable methods for higher orders?

5.3 An Algorithm Which Selects Methods Continuously

The use of parametric multistep methods opens the door to the possibility of using method choice as a primary control feature in solver performance. Classically, solvers implementing linear multistep methods will only use one or two methods for each order, and only change methods when an order change is deemed necessary. Rather than changing order or class, it may be possible to choose a new method from the same order and class to improve performance in a similar fashion to step size control. This possibility can be demonstrated by examining the E_2 methods: In this class, we saw that methods with smaller θ_0 (towards $\pi/4$) had larger stability regions, while methods with larger θ_0 (towards π) had smaller error coefficients. If a solver implementing these methods was able to detect that a problem required a larger stability region, then a method could be chosen to accommodate this necessity. If, on the other hand, the solver detected that the problem was sufficiently stable, and instead wished to decrease error for a given step size, the solver could choose a method with a smaller error coefficient. This of course is only a simple example, as E_2 methods will likely not be used as primary methods in industrial solvers. However, if more were known about the higher order classes, and if there was a clear theoretical way to navigate their qualities, then perhaps this idea could be extended to multistep solvers implementing higher order methods which must meet more realistic demands, and such a controller could be proposed which automatically selects the best method.

Bibliography

[1] Arévalo, Jonsson-Glans, Olander, Selva-Soto, Söderlind, *Transactions on Mathematical Software: A Software Platform for Adaptive High Order Multistep Methods*, 2018

[2] Arévalo, Söderlind, *Grid-Independent Consutrction of Multistep Methods*, 2017

[3] Arévalo, Söderlind, Hadjimichael, Fekete, *Local Error Estimation and Step Size Control in Adaptive Linear Multistep Methods*, 2020

[4] Brown, Byrne, Hindmarsh, *VODE, A Variable-Coefficient ODE Solver*, 1988

[5] Byrne, Hindmarsh, *A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations*, 1975

[6] Eich-Soellner, Führer, *Numerical Methods in Multibody Dynamics*, 1998

[7] Euler, *Institutionum Calculi Integralis Volumen Primum*, 1768

[8] Führer, Schroll, *Numerical Analysis - An Introduction (3rd Edition)* 2001

[9] Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, 1971

[10] Hairer, Nørsett, Wanner, *Solving Ordinary Differential Equations I Nonstiff Problems* 2008

[11] Hindmarsh, *ODEPACK, A Systematized Collection of ODE Solvers*, 1982

[12] Hindmarsh, Serban, Balos, Gardner, Reynolds, Woodward, *User Documentation for IDA v5.7.0 (SUNDIALS v5.7.0)* 2021

[13] Hindmarsh, Serban, Reynolds, *User Documentation for CVODE v5.7.0 (SUNDIALS v5.7.0)*, 2021

[14] Iserles, *A First Course in the Numerical Analysis of Differential Equations*, 2009

[15] Jackson, Sacks-Davis, *An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODEs*, 1980

[16] Olander, Jonsson-Glans, *Construction and Benchmarking of Adaptive Parametrized Linear Multistep Methods*, 2016

[17] Radhakrishnan, Hindmarsh, *Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*, 1993

[18] Shankar, *Linear Multistep Methods I: Adams and BDF Methods* 2016

[19] Shampine, Gordon, *Computer Solution of Ordinary Differential Equations*, 1975

[20] Suli, Mayers, *An Introduction to Numerical Analysis*, 2003

[21] Söderlind, *Digital Filters in Adaptive Time-Stepping*, 2003

[22] Söderlind, *Time-step selection algorithms: Adaptivity, Control, and Signal Processing*, 2006

[23] Söderlind, Wang, *Adaptive time-stepping and computational stability*, 2006

[24] Söderlind, Wang, *Evaluating numerical ODE/DAE methods, algorithms and software*, 2006