Concerning the Stability of Complex Time Steppers

Cian J. Duggan

July 30, 2024

Contents

1 Introduction

A paper by Lloyd N. Trefethen [1] offers two definitions of numerical analysis:

> *The study of rounding errors.*

> *The study of algorithms for the problems of continuous mathematics.*

Trefethen argues that the first definition, though an accurate summation of the field's history, does not serve to entice the curious to explore the field. He proposes the second as a more compelling definition, one that emphasises the field's role in solving real-world problems, and inspires curiosity.
He offers an optimistic view of the field, one that is not bogged down by the minutiae of rounding errors, but rather one that is focused on the algorithms that make numerical analysis possible. More to ground, the majoirity of numerical analysis is concerned with the speed of convergence and minimization of error.
Through my work on this paper, I have honed my own definition:

> *Numerical analysis is the field which attempts to straddle the gaps between the discrete and the continuous.*

In this paper we will take a look at the stability of various numerical methods when applied to problems of differential calculus. In particular, we wish to compare the use of complex time steps with real time steps for these methods.

2 Definitions and Concepts

2.1 Time Steps

Let's take a step back to when we first studied derivatives.
Newton's Difference Quotient should be familiar: $f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$
Viewing Newton's Difference Quotient in the context of numerical methods, we call h a *time step*.
Due to computational constraints, we cannot take h to be infinitesimal. (A computer's memory is as small as it is big)
Instead, we take h to be a small, finite number.
This defines the concept of a *step size*.

In a more general sense, differential equations may be represented as $y'(t) = \phi(t, y(t))$.
In this case, t is an independent variable and $y(t)$ is a dependent variable.
The function $\phi(t, y(t))$ is the derivative of $y(t)$ with respect to t, and is a function of both t and $y(t)$.
In the context of numerical methods, we approximate $y'(t)$ by $\frac{y(t+\Delta_t)-y(t)}{\Delta_t}$.
Here, Δ_t is the time step.
This gives $y(t + \Delta_t) \approx y(t) + \Delta_t y'(t)$; a first-order approximation of y a small time step Δ_t in the future.

2.2 Complex-Variable Method

Take a function $f\colon \mathbb{R} \longrightarrow \mathbb{R},\ x \longmapsto f(x)$.
Let f be *holomorphic* on its domain: f is complex differentiable in a neighbourhood of any $x \in \mathbb{R}$.
$\forall x$, f is complex differentiable on $B_\beta(x) = \{z \in \mathbb{C} \mid |z| - |x| < \beta_{\text{small}}\}$ for some $\beta_{\text{small}} > 0$.
In this case, we get the following fact:

$$f'(x) = \frac{Im\big(f(x+ih)\big)}{h} + O(h^2), \quad \text{where } h \in \mathbb{R} \text{ and } h \neq 0$$

Combining this with the first order approximation above, we get $y(t + \Delta_t) \approx y(t) + Im\big(y(t + i\Delta_t)\big)$.

2.3 Stability

In numerical analysis, a method is said to be *stable* if small deviations in the input do not lead to large perturbtions in the output. In the context of differential equations, a method is said to be stable if the solution does not grow to be unbounded as the number of time steps increases.
When solving differential equations numerically, the choice of time step is crucial. If the step size is too large, the solution may become unstable; the numerical solution will diverge from the analytic solution in proportion with the number of steps. If the step size is too small, the solution may be accurate and stable, but the computation may be too slow.
This tradeoff between accuracy and stability is a common theme in numerical analysis, and depends entirely on the choice of numerical method.

3 Stability Analysis

3.1 Introduction

In this section we will lay out a framework for analysing the stability of a numerical method dependant on time steps. To begin, we will consider a simple toy problem to illustrate the concept of stability. We will introduce the concept of the stability function and show how it can be used to analyse the stability of a numerical method. Finally, we will extend this framework to view the stability of a numerical method as a function of complex time steps.

3.2 Exponential Decay Problem

We would like a simple toy problem on which we can build a framework for the analysis of the stability of a numerical method.
A common example across numerical analysis is the ODE exponential decay problem:

$y'(t) = \lambda y(t) \quad \text{with} \quad y(0) = 1 \text{ and } Re(\lambda) < 0$

Where the constant $\lambda \in \mathbb{C}$ can be interpreted as the decay rate of the solution.
This has the exact solution $y(t) = e^{\lambda t}$.

3.2.1 Proponents of the Exponential Decay Problem

There are a few pros to this choice of problem:

- The exact solution is known and easily computed; it can be used to evaluate a numerical solution.
- The solution is a straigtforward and well understood function.
  This simplicity allows us to focus on the numerical method.
- The problem is linear in $y(t)$, making the problem simple to work with.
  Again, our focus can stay on the numerical method.
- The graph of the exact solution is easy to visualise; the more chaotic outputs of the numerical methods can be compared to the smooth curve of the exact solution easily.
- Exponential decay is a model for many physical processes, including radioactive decay, population decay, and capacitor discharge.
  This means readers from many different disciplines already posses an intuition for the problem.
- The problem is simple to generalise to complex numbers, allowing us to explore the stability of numerical methods in the complex plane.
- Stiffness. WRITE MORE
- Generalisation. WRITE MORE

3.2.2 Additional Details

The Taylor series for the exact solution is

$$y(t) = \sum_{n=0}^{\infty} \frac{(t)^n}{n!}$$

The solution decays to zero as $t \to \infty$ when $Re(\lambda) < 0$.

Normally, when using the exponential decay problem in a numerical analysis context, we would allow $\lambda \in \mathbb{C}$ with $Re(\lambda) < 0$.
However, for the purposes of this paper, we will restrict λ to $\mathbb{R}$ as we will be looking at $h \in \mathbb{C}$ and we don't want to overcomplicate things.
Unless otherwise mentioned, we will assume that $\lambda \in \mathbb{R}$ and $\lambda < 0$ for the rest of this paper.

$\phi(t, y) = y'(t) = \lambda y(t)$ is linear in $y(t)$

A lot of the findings in this paper will be represented graphically.
Let's start with the most basic.

Here's a graph of the exact solution for the exponential decay problem with $\lambda = -1$:

`Exponential Decay/Exact with -1.png`

3.3 Numerical Method: Forward Euler

We will now consider a simple numerical method applied to the exponential decay problem. The Forward Euler method is a first-order numerical method for solving ODEs with a given initial condition. Its algorithm can be defined as follows:

$$y(t_{j+1}) = y(t_j) + hy'(t_j)$$

Or equivalently:

$$y_{j+1} = y_j + h\phi(t_j, y_j)$$

Where h is the time step and y_j is the numerical solution at time t_j.
For the exponential decay problem, the Forward Euler method can be written as:

$$y_{j+1} = y_j + h\lambda y_j \quad \text{where} \quad y_0 = 1$$

This gives us the following algorithm:

$$y_{j+1} = (1 + h\lambda)y_j$$

This gives an approximation of the exact solution at time t_j as follows:

$$y_j \approx (1 + h\lambda)^j y_0$$

We can see that the Forward Euler method is stable if $|1 + h\lambda| < 1$; both the exact solution and our approximation will decay to zero as $t \to \infty$. The stability is dependent on the time step h and the value of λ. We can write this as $s(\lambda, h) = 1 + h\lambda$. By analysing s for different values of λ and h, we can infer the stability of the Forward Euler method for the exponential decay problem. We call s the *stability function* of the Forward Euler method.

3.4 The Stability Function and corresponding Stability Region

By the same methodology, we can define the stability function for any numerical method:
Write the algorithm in the form $y_{j+1} = s(\lambda, h)y_j$.
$s(\lambda, h)$ is the *stability function* of the numerical method with a time step h.
Note: This is equivalent to $y_j = s(\lambda, h)^j y_0$
The *stability region* of a numerical method is the set $S = \left\{(\lambda, h) \;\middle|\; |s(\lambda, h)| < 1\right\}$
This follows from the definition of stability for our exponential decay problem; a method is stable if the numerical solution decays to zero as $t \to \infty$.
Clearly, $y_j = s(\lambda, h)^j y_0$ will decay to zero as $j \to \infty$ if $|s(\lambda, h)| < 1$.

3.4.1 Stability Region for Euler's Forward Method

Euler's Forward Method has the stability function

$$s(\lambda, h) = 1 + \lambda h$$

The corresponding stability region is

$$S = \left\{(\lambda, h) \;\middle|\; |1 + \lambda h| < 1\right\}$$

We can see this region plotted in red on the right; a unit circle centred at -1.
For a given $\lambda \in \mathbb{C}$, the method is stable for any step-size $h \in \mathbb{R}$ such that $|1 + \lambda h| < 1$.
Expanding $\lambda = a + bi$, we get the restriction

$$|1 + (a + bi)h| < 1 \qquad \implies \qquad a^2h^2 + 2ah + b^2h^2 < 0$$

As a, b and $h \in \mathbb{R}$, $(a^2 + b^2)h^2 > 0$
$\implies 2ah < 0$ with $|2ah| > (a^2 + b^2)h^2$

`Stability Regions/Graphs/Real 1-Step/Euler's Forward.png`

3.4.2 Stability Region for Euler's Backward Method

Stability Regions/Graphs/Real 1-Step/Euler's Backward.png

Euler's Backward Method can be written as

$y_{j+1} = y_j + h.\phi(t_{j+1}, y_{j+1})$

$\implies y_{j+1} = y_j + h(\lambda y_{j+1}) \implies y_{j+1} = \dfrac{1}{1-\lambda h} y_j$

Thus, the stability function is

$s(\lambda, h) = \dfrac{1}{1-\lambda h}$

The corresponding stability region is

$$S = \left\{(\lambda, h) \;\middle|\; \left|\frac{1}{1-\lambda h}\right| < 1\right\}$$

This is plotted in red on the left; the region outside a unit circle centred at 1.
The white region of instability is exactly the stability region for Euler's Forward Method, flipped about Imaginary axis.

3.4.3 Stability Region for Runge-Kutta 4

Runge-Kutta 4 can be written as

$y_{j+1} = y_j + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Where

$k_1 = \phi(t_j, y_j) \qquad k_2 = \phi(t_j + \dfrac{h}{2}, y_j + \dfrac{h}{2}k_1)$

$k_3 = \phi(t_j + \dfrac{h}{2}, y_j + \dfrac{h}{2}k_2) \qquad k_4 = \phi(t_j + h, y_j + hk_3)$

For the Exponential Decay Problem, we get

$y_{j+1} = (1 + \lambda h + \dfrac{(\lambda h)^2}{2} + \dfrac{(\lambda h)^3}{6} + \dfrac{(\lambda h)^4}{24})y_j$

The stability function is $s(\lambda, h) = \sum_{m=0}^{4} \frac{(\lambda h)^m}{m!}$
The corresponding stability region is

$$S = \left\{(\lambda, h) \;\middle|\; \left|\sum_{m=0}^{4} \frac{(\lambda h)^m}{m!}\right| < 1\right\}$$

Stability Regions/Graphs/Real 1-Step/Runge-Kutta 4.png

3.5 Interpretation of Stability Regions

In the first graph, shaded in grey, is the region of stability for Runge-Kutta 4.
This region is the set of λh values for which the method is stable.

In all other graphs, in black, is a plot of the exact solution to the Exponential Decay Problem for a given λ. For this curve to appear smooth, a step-size in t of $h = \frac{1}{1000}$ was used.

As we have fixed h, the stability region gives the set of values $\frac{\lambda}{1000}$ for which the method is stable.
From this we can derive stable λ values.
Highlighted on the left are n coloured points, representing n different values of $\frac{\lambda}{1000}$. The green point is -1, and sits inside the stability region. This tells us that the method is stable for $\lambda = -1000$.
We can see this on the right, as the green curve of RK4 with $\lambda = -1000$ decays to zero and closely follows the exact solution in black.

MORE DETAIL HERE

3.6 Numerical Method: 2-step Abysmal Kramer-Butler Method

This method was designed to be unstable for the sake of demonstration.
The algorithm is as follows:

$$y_{j+1} = y_{j-1} + h(4\phi(t_j, y_j) - 2\phi(t_{j-1}, y_{j-1}))$$

As we know the exact solution to our Exponential Decay problem is $y(t) = e^{\lambda t}$, we know $\phi(t, y) = \lambda y$.
This gives:

$$y_{j+1} = y_{j-1} + h(4\lambda y_j - 2\lambda y_{j-1})$$

Simplified:

$$y_{j+1} = 4\lambda h y_j + (1 - 2\lambda h) y_{j-1}$$

4 Complex Time-Steps

So far, we have restricted ourselves to time-steps in the real domain.
It has often proven useful to extend the analysis of a problem to the complex domain.
Motivated by a paper by *George, Yung and Mangan*[2], we will have a look at the stability of numerical methods when the chosen time-step is complex.
They state:

> *Most numerical methods for time integration use real time steps. Complex time steps provide an additional degree of freedom, as we can select the magnitude of the step in both the real and imaginary directions. By time stepping along specific paths in the complex plane, integrators can gain higher orders of accuracy or achieve expanded stability regions.*

4.1 Complex 2-Step

We want to take an overall step of size $\lambda h \in \mathbb{R}$ comprised of two steps $z_1, z_2 \in \mathbb{C}$ such that $z_1 + z_2 = \lambda h$.
We can set $z_1 = a + bi$ and $z_2 = \lambda h - a - bi$.
For analysing our numerical methods, we require a comparable implementation of the real step setup.

A real 2-step method where we take a step of size $\frac{\lambda h}{2}$ allows us to take two steps and go the same distance as the complex 2-step method.
We can write this real 2-step method as $y_{j+1} = s(\lambda, \frac{h}{2})^2 y_j$.
Note: We can write this explicitly as

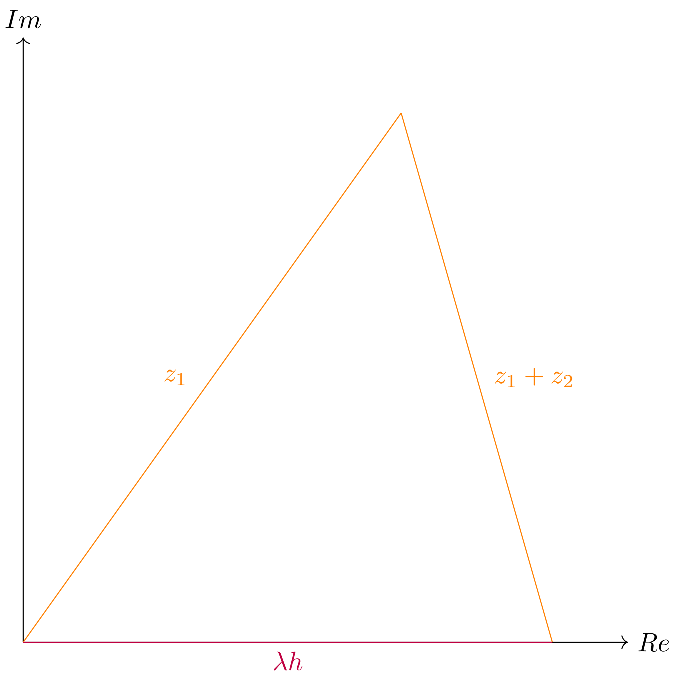
$$y_{j+1} = s(\lambda, \frac{h}{2})y_{j+\frac{1}{2}} = s(\lambda, \frac{h}{2})s(\lambda, \frac{h}{2})y_j$$

We can write the complex 2-step method as

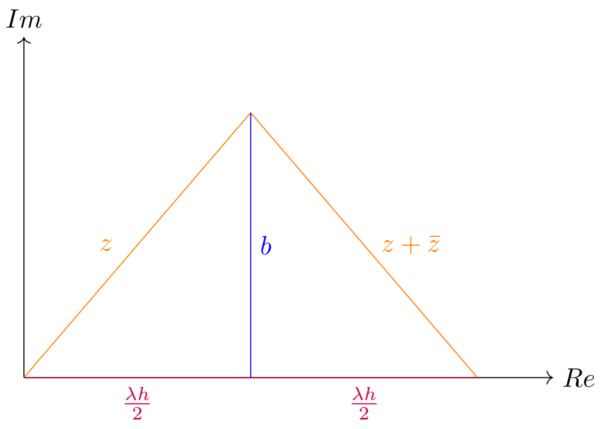
$$y_{j+1} = s(\lambda, z_1)s(\lambda, z_2)y_j$$

Note: We can write this explicitly as

$$y_{j+1} = s(\lambda, z_1)y_{j+\frac{1}{2}} = s(\lambda, z_1)s(\lambda, z_2)y_j$$



4.2 Complex Conjugate Pairs



When we restrict the exponential decay problem by $\lambda \in \mathbb{R}$, we also find that $s(\lambda, z_1)s(\lambda, z_2) \in \mathbb{R}$
This is because any value attained by $e^{\lambda t}$ is real.
This restricts the values of z_1 and z_2, as any imaginary terms in $s(\lambda, z_1)s(\lambda, z_2)$ must cancel out.

One particularly simple family of such complex pairs is defined by setting $a = \frac{\lambda h}{2}$
This necessitates $z_2 = \bar{z_1}$
We will refer to this family as the *complex conjugate step pairs*.

4.3 Stability of 2-Step Methods for Complex Conjugate Step Pairs

In this section we will compare the stability regions for the real and complex 2-step methods.
The real step size is given by $\frac{\lambda h}{2}$
The complex conjugate step pair is given by $z = \frac{\lambda h}{2} + bi$ and $\bar{z} = \frac{\lambda h}{2} - bi$
The plots in this section show the regions of stability $S = \{h\colon |s(\lambda, h)| \leq 1\}$ for the case where $b = a = \frac{\lambda h}{2}$
We will have a look at varying b in a subsequent section.

4.3.1 Euler's Forward 2-Step

Stability Regions/Graphs/Real VS Complex Comparison/Euler's Forward.png

Euler's Forward 2-Step $\mathbb{R}$

$$y_{j+1} = \left(1 + \frac{\lambda h}{2}\right)^2 y_j$$

$$\implies s_{\mathbb{R}}(\lambda, h) = 1 + \lambda h + \frac{(\lambda h)^2}{4}$$

Euler's Forward 2-Step $\mathbb{C}$

$$\begin{aligned} y_{j+1} &= \left(1 + z\right)\left(1 + \bar{z}\right) y_j \\ &= \left(1 + \frac{\lambda h}{2} + bi\right)\left(1 + \frac{\lambda h}{2} - bi\right) y_j \\ &= \left(\left(1 + \tfrac{\lambda h}{2}\right)^2 + b^2\right) y_j \end{aligned}$$

$$\implies s_{\mathbb{C}}(\lambda, h) = 1 + \lambda h + \frac{(\lambda h)^2}{4} + b^2$$

$$\implies s_{\mathbb{C}}(\lambda, h) = s_{\mathbb{R}}(\lambda, h) + b^2$$

As $b \in \mathbb{R}$, $s_{\mathbb{C}}(\lambda, h) \geq s_{\mathbb{R}}(\lambda, h)$
This means the stability region for the complex 2-step method is smaller than that of the real 2-step method; there are less values of λh for which the complex 2-step method is stable.
$S_{\mathbb{C}}$ is smaller than $S_{\mathbb{R}}$
The diagram above illustrates the case where $b = a = \frac{\lambda h}{2}$

We can vary the b value to see how the stability region changes.
This series of diagrams shows the stability regions for the complex 2-step method with varying b values.

We can do the same kind of analysis for other Numerical Methods.
Below are the same derivations and diagrams for the Backward Euler and Runge-Kutta 4 methods.

4.3.2 Euler's Backward 2-Step

Stability Regions/Graphs/Real VS Complex Comparison/Euler's Backward.png

Euler's Backward 2-Step $\mathbb{R}$

$$y_{j+1} = \left(\frac{1}{1-\frac{\lambda h}{2}}\right)^2 y_j$$

$$\implies s_{\mathbb{R}}(\lambda, h) = \frac{1}{1-\lambda h+\frac{(\lambda h)^2}{4}}$$

Euler's Backward 2-Step $\mathbb{C}$

$$y_{j+1} = \left(\frac{1}{1-\frac{\lambda h}{2}+bi}\right)\left(\frac{1}{1-\frac{\lambda h}{2}-bi}\right) y_j$$

$$\implies s_{\mathbb{C}}(\lambda, h) = \frac{1}{1-\lambda h+\frac{(\lambda h)^2}{4}+b^2}$$

Again, $b \in \mathbb{R}$, so this time, $s_{\mathbb{C}}(\lambda, h) \leq s_{\mathbb{R}}(\lambda, h) \implies S_{\mathbb{C}}$ is larger than $S_{\mathbb{R}}$

4.3.3 Runge-Kutta 4 2-Step

Stability Regions/Graphs/Real VS Complex Comparison/Runge-Kutta 4.png

Runge-Kutta 4 2-Step $\mathbb{R}$

$$y_{j+1} = \left(1+z+\frac{z^2}{2}+\frac{z^3}{6}+\frac{z^4}{24}\right)^2 y_j$$

$$y_{j+1} = \Bigg(1+\lambda h+(\lambda h)^2+\frac{(\lambda h)^3}{6}+\frac{(\lambda h)^4}{24} + \frac{(\lambda h)^5}{128}+\frac{5(\lambda h)^6}{4608}+\frac{(\lambda h)^7}{9216}+\frac{(\lambda h)^8}{147456}\Bigg) y_j$$

$$\implies s_{\mathbb{R}}(\lambda, h) = 1+\lambda h+(\lambda h)^2+\frac{(\lambda h)^3}{6}+\frac{(\lambda h)^4}{24} + \frac{(\lambda h)^5}{128}+\frac{5(\lambda h)^6}{4608}+\frac{(\lambda h)^7}{9216}+\frac{(\lambda h)^8}{147456}$$

Computer Calculated Complex; should probably be checked
$1+\lambda h+\frac{\lambda h^2}{2}+\frac{\lambda h^3}{6}+\frac{\lambda h^4}{24}+\frac{\lambda h^5}{128}+\frac{5\lambda h^6}{4608}+\frac{\lambda h^7}{9216}+\frac{\lambda h^8}{147456}+\frac{b^2\lambda h^3}{48}+\frac{b^2\lambda h^4}{128}+\frac{b^2\lambda h^5}{768}+\frac{b^2\lambda h^6}{9216}+\frac{b^4\lambda h}{24}-\frac{b^4\lambda h^2}{96}+\frac{b^4\lambda h^3}{192}+\frac{b^4\lambda h^4}{1536}-\frac{b^6}{72}+\frac{b^6\lambda h}{144}+\frac{b^6\lambda h^2}{576}+\frac{b^8}{576}$

References

[1] Llyod N. Trefethen, *The definition of numerical analysis*, Cornell University, 1992.

[2] Jithin D. George, Samuel Y. Jung, and Niall M. Mangan, *Walking into the complex plane to 'order' better time integrators*, `https://arxiv.org/abs/2110.04402`, 2021.

[3] Next reference.