The Stability of Complex Time Steppers

Cian J. Duggan

Trinity College Dublin
The University of Dublin

Introduction

This poster looks at the stability of numerical methods when applied to problems of differential calculus.
This project aims to add a visual understanding to the concepts of stability.
It also aims to explore the use of complex time steps in numerical methods.

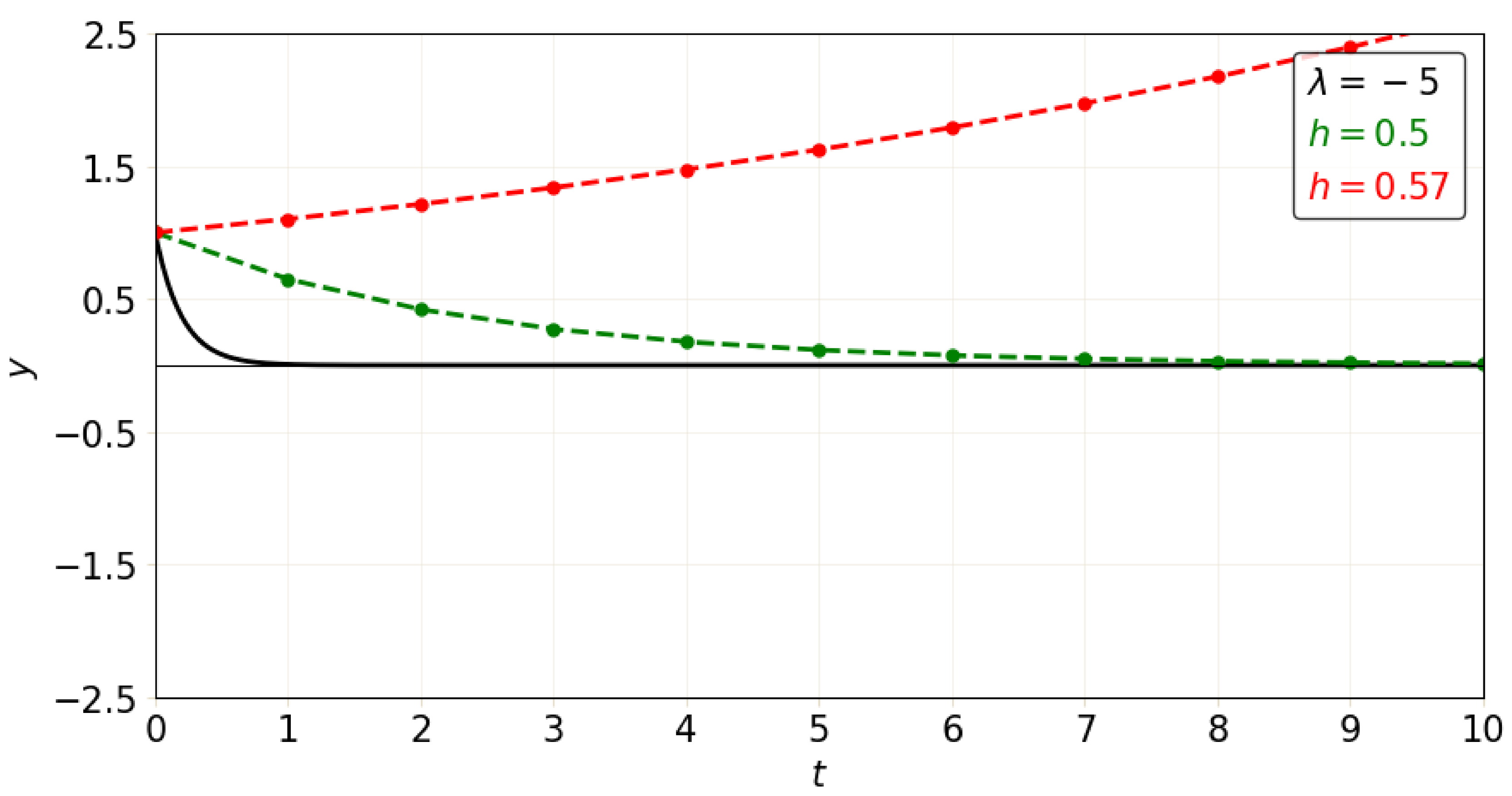
Key Terminology

Numerical Methods are iterative algorithms used to approximate solutions to problems that cannot be solved exactly.
These methods are often employed to solve differential equations.

The infinitesimal h in $y'(t) = \lim_{h\to 0} \frac{y(t+h)-y(t)}{h}$ cannot be represented on a computer.
Instead, a finite **time step** h is used to approximate the derivative.

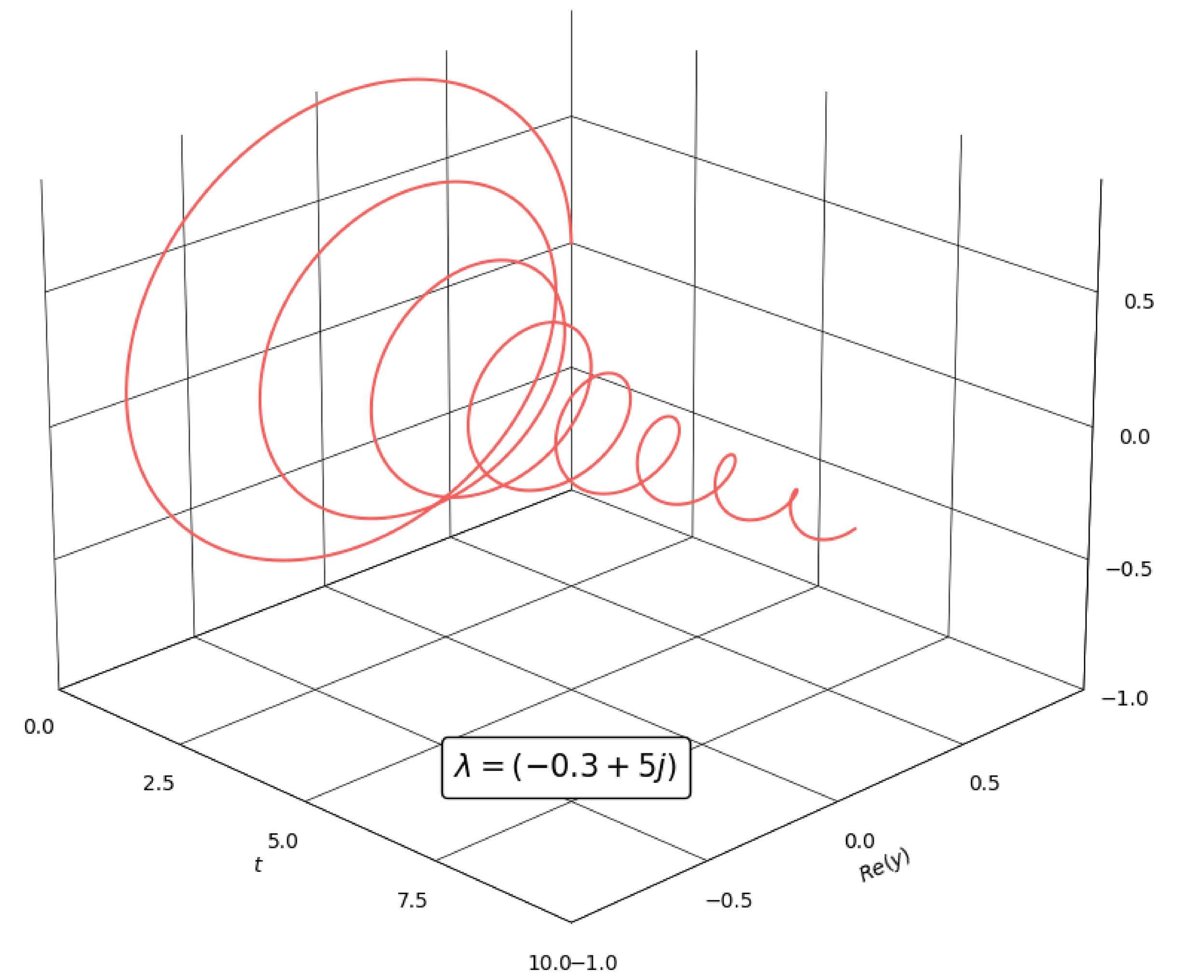
A numerical method is **stable** for a given size of time step if the solution does not diverge from the exact solution.
The choice of step size is crucial; too large, and the solution will diverge, too small, and the solution will cost computational resources.

Shown below is **Runge-Kutta 4** applied to the Exponential Decay Problem.
The exact solution is known, and shown in black.
Stable and unstable choices of h are shown in green and red respectively.



The Exponential Decay Problem

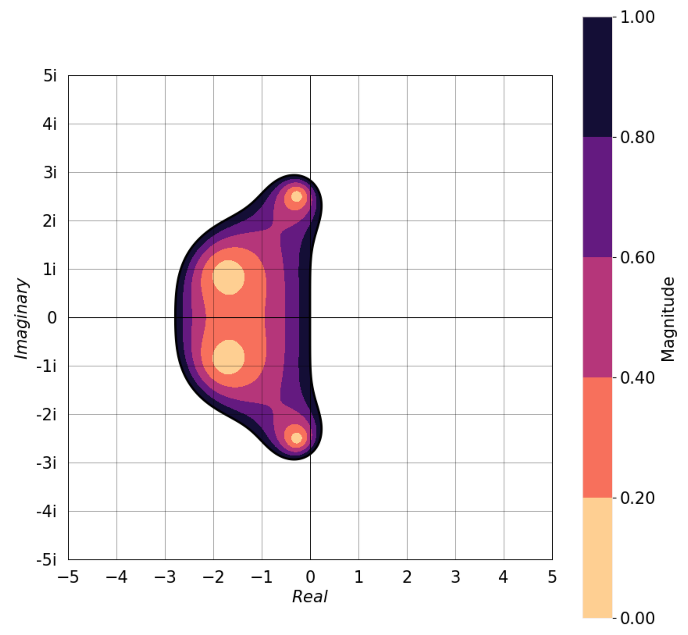
Consider a quantity y that decays relative to its current value.
The quantity at time t is $y(t)$.
The rate of decay is $\lambda \in \mathbb{C}$ with $Re(\lambda) < 0$.
The simplest case of this problem is described by the differential equation:
$y'(t) = \lambda y(t) \quad \text{with} \quad y(0) = 1$
This has the exact solution $y(t) = e^{\lambda t}$ giving the name *Exponential Decay.*

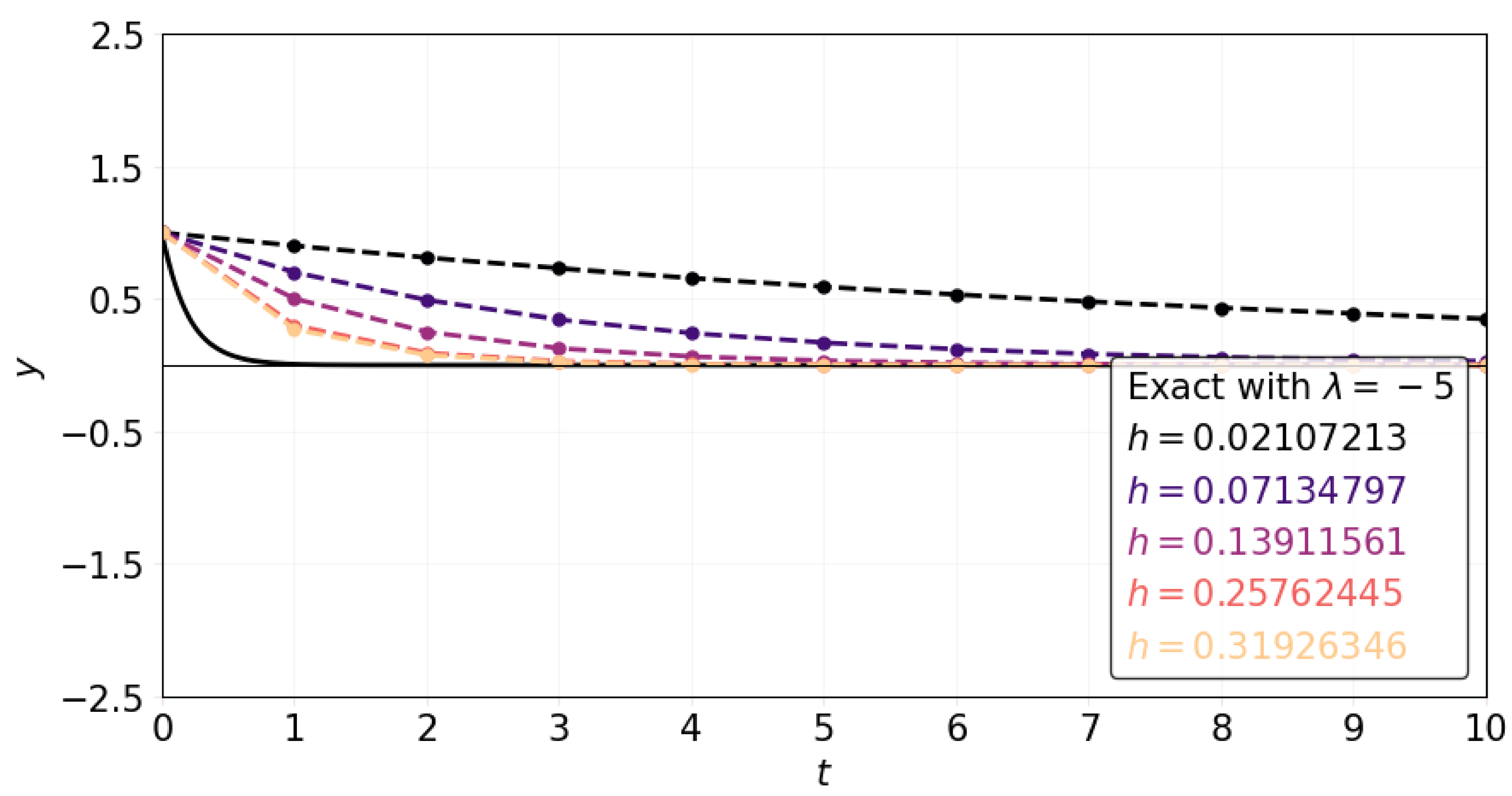


Stability

We can define the **Stability Function** $s(\lambda, h)$ for any numerical method by writing the algorithm in the form $y_{j+1} = s(\lambda, h)y_j$.
A method is stable for any pair in $S = \{(\lambda, h) \colon |s(\lambda, h)| < 1\} \in \mathbb{C}$, the **Stability Region**.
Applying **Runge-Kutta 4**: $y_{j+1} = (1 + \lambda h + \frac{(\lambda h)^2}{2} + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{24})y_j$

The stability function is
$s(\lambda, h) = \sum_{n=0}^{4} \frac{(\lambda h)^n}{n!}$
S is shown on the right.
S is divided into subregions, each corresponding to a different range of $|s(\lambda, h)|$ values.
The methods with these step sizes are shown below, for $\lambda \in \mathbb{R}^-$:

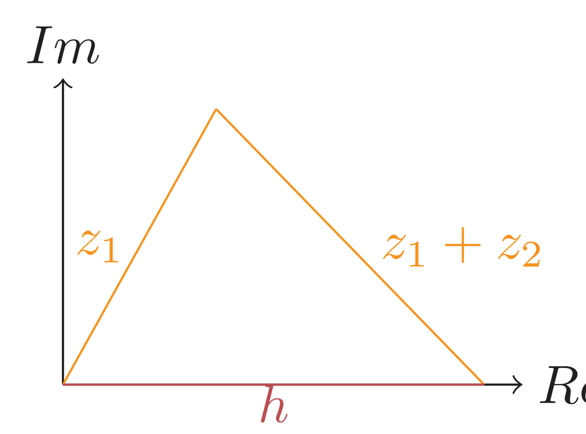


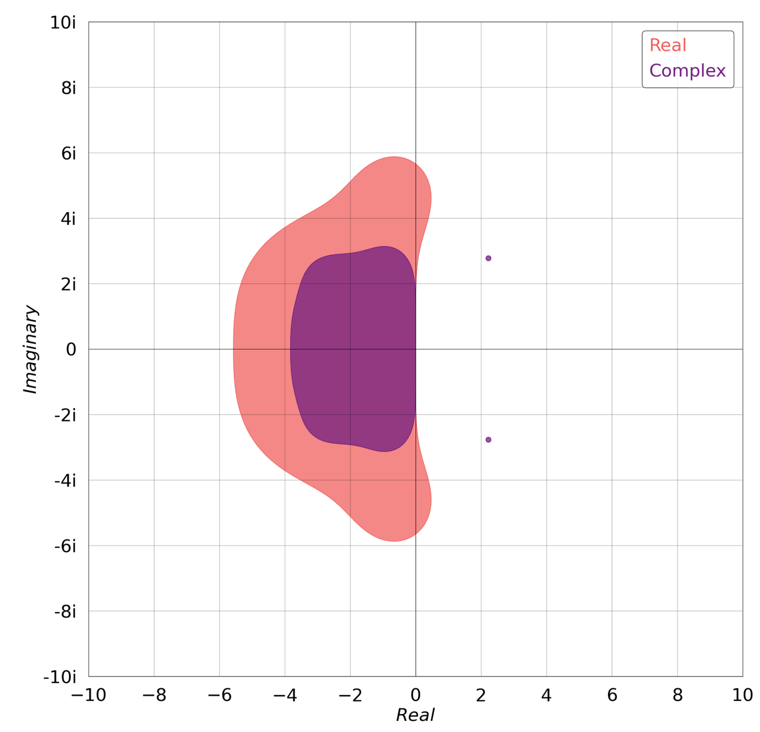


The Complex Domain

It has often proven useful to extend the analysis of a problem to the complex domain. We will have a look at the stability of numerical methods when the chosen time-step is complex.

We want to take an overall step of size $h \in \mathbb{R}$ comprised of two steps $z_1, z_2 \in \mathbb{C}$ such that $z_1 + z_2 = h$.
We compare to the case where we take 2 steps of size $\frac{h}{2} \in \mathbb{R}$.
The following diagram shows the case
$z_1 = \frac{h}{2}(1 + i)$ and $z_2 = \bar{z}_1$





George, Yung and Mangan[1] show that the error is reduced when using complex time steps.
For Runge-Kutta 4, we obtain two different stability functions
$s_{\mathbb{R}}(\lambda, h)$ and $s_{\mathbb{C}}(\lambda, h)$
They give the stability regions shown on the left.
While the error may decrease, the stability region decreases with it.

References

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

[2] The GitHub repository for this project, `https://github.com/CianJDuggan/Capstone-Project`