MA3227 Numerical Analysis II

Lecture 18: Implicit Runge-Kutta Methods

Simon Etter

National University of Singapore

2019/2020

Implicit Runge-Kutta Methods

Introduction

Recall from Lecture 17 that adaptive time-stepping applied to the ODE $\dot{y} = \lambda y$ with $\lambda < 0$ failed to increase the time step Δt beyond a certain upper bound even though theory tells us that we should be able to choose Δt arbitrarily large for t large enough.

The ODE $\dot{y} = \lambda y$ is simple enough that we can determine explicit formulae for the Euler and midpoint steps:

- Euler: $$\tilde{y}(t) = y_0 + f(y_0)\, t = y_0 + \lambda y_0 t = (1 + \lambda t)\, y_0.$$
- Midpoint: $$\begin{aligned}\tilde{y}(t) &= y_0 + f\left(y_0 + f(y_0)\,\tfrac{t}{2}\right) t \\ &= y_0 + \lambda \left(y_0 + \tfrac{1}{2}\lambda y_0 t\right) t \\ &= \left(1 + \lambda t + (\lambda t)^2\right) y_0.\end{aligned}$$

Conclusion: after k steps with constant step size Δt, the Runge-Kutta solution is given by

$$\tilde{y}(k\,\Delta t) = R(\lambda\,\Delta t)^k\, y(0) \quad \text{where} \quad R(z) = \begin{cases} 1 + z & \text{(Euler)}, \\ 1 + z + \frac{z^2}{2} & \text{(midpoint)}. \end{cases}$$

Implicit Runge-Kutta Methods

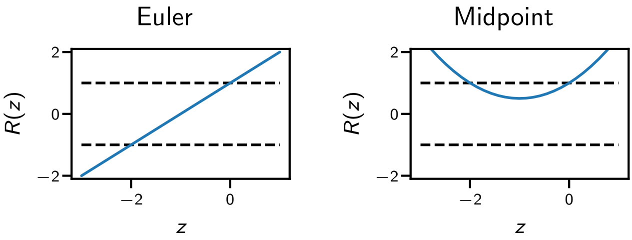
Introduction (continued)

We know that since $\lambda < 0$, the exact solution satisfies

$$\lim_{k\to\infty} y(k\,\Delta t) = \lim_{k\to\infty} \exp(\lambda\, k\,\Delta t) = 0.$$

It follows from the equation on the previous slide that the numerical solution $\tilde{y}(k\,\Delta t)$ has the same limit if and only if $|R(\lambda\,\Delta t)| < 1$, and plots of $R(z)$ reveal that

$$|R(\lambda\,\Delta t)| < 1 \quad\Longleftrightarrow\quad -2 < \lambda\,\Delta t < 0 \quad\Longleftrightarrow\quad \Delta t < \tfrac{2}{-\lambda}.$$



Implicit Runge-Kutta Methods

Introduction (continued)

The above explains our observations in Lecture 17: if $\Delta t > \frac{2}{-\lambda}$, then the numerical solution diverges from the exact solution. The step-size control detects this and makes sure Δt never exceeds $\frac{2}{-\lambda}$.

The discrepancy between our expectations and the numerical results arises because our expectations are based on a wrong interpretation of Taylor series. We have seen that Euler's method satisfies

$$\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1}) = \ddot{y}(t_{k-1})\,\frac{\Delta t_k^2}{2} + \mathcal{O}\big(\Delta t_k^3\big),$$

and we have assumed that if the Δt_k^2-term is small, then all higher-order terms must be even smaller. This is indeed the case if Δt_k is small, but in adaptive time-stepping we are interested in making Δt_k as large as possible, so eventually this Taylor-series argument will break down.

Implicit Runge-Kutta Methods

Linearisation of ODEs

The discussion so far was specific to the ODE $\dot{y} = \lambda y$

However, it turns out that our conclusions are relevant for generic ODEs $\dot{y} = f(y)$ as long as there is an attractive fixed-point, i.e. a y_F such that $f(y_F) = 0$ and $\nabla f(y_F)$ has at least one eigenvalue λ with $\mathrm{Re}(\lambda) < 0$.

"Proof". For y close to y_F, we obtain

$$\begin{aligned} \tfrac{d}{dt}\big(y(t) - y_F\big) &= f\big(y(t)\big) \\ &= \underbrace{f(y_F)}_{=0} + \nabla f(y_F)\,\big(y(t) - y_F\big) + \mathcal{O}\big(\|y(t) - y_F\|^2\big). \end{aligned}$$

Let us assume that $\nabla f(y_F)$ has eigendecomposition $\nabla f(y_F) = V \Lambda V^{-1}$. Ignoring the $\mathcal{O}$-term and introducing $w(t) = V^{-1}\big(y(t) - y_F\big)$, we then obtain

$$V\dot{w}(t) = \tfrac{d}{dt} V w(t) = \nabla f(y_F)\, V w(t) = V \Lambda w(t) \quad \iff \quad \dot{w} = \Lambda w.$$

The last equation is a system of n decoupled ODEs of precisely the form $\dot{w}_i = \lambda\, w_i$; hence the above discussion applies for $y(t)$ close enough to y_F.

Implicit Runge-Kutta Methods

Discussion

The discussion on the previous slide shows that for general $f(y)$, we are interested in the behaviour of Runge-Kutta methods applied to the ODE $\dot{y} = \lambda y$ where λ is an eigenvalue of $\nabla f(y_F)$.

The Jacobian $\nabla f(y_F)$ is generally a non-symmetric matrix, so the eigenvalues λ can be complex. The solution to $\dot{y} = \lambda y$ is still

$$y(t) = y_0 \, \exp(\lambda t) = y_0 \, \exp\big(\mathrm{Re}(\lambda) t\big) \, \Big(\cos\big(\mathrm{Im}(\lambda)\big) + \iota \sin\big(\mathrm{Im}(\lambda)\big)\Big);$$

hence we conclude that $\mathrm{Re}(\lambda)$ indicates whether $y(t)$ converges to zero ($\mathrm{Re}(\lambda) < 0$) or diverges ($\mathrm{Re}(\lambda) > 0$), and $\mathrm{Im}(\lambda)$ indicates whether the solution oscillates.

Similary, the Runge-Kutta solutions still satisfy $\tilde{y}(k\,\Delta t) = R(\lambda\,\Delta t)^k$, and we conclude that these solutions have the right convergence / divergence behaviour if

$$\mathrm{Re}(z) < 0 \quad \iff \quad |R(z)| < 1.$$

This motivates the definitions on the next slide.

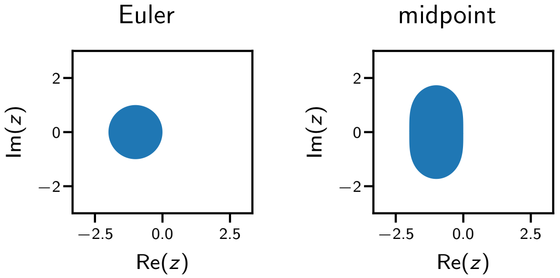
Implicit Runge-Kutta Methods

Terminology

- The functions $R(z)$ introduced above are called the stability function of the Runge-Kutta method.
- The set $\{z \in \mathbb{C} \mid |R(z)| < 1\}$ is called the stability domain of the Runge-Kutta method.

Example

The stability domains of the Euler and midpoint methods are:



Implicit Runge-Kutta Methods

Example

Consider the ODE $\ddot{x} = -x$, $x(0) = 1$, $\dot{x}(0) = 0$, or equivalently

$$\dot{y} = \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix} = f(y) \qquad \text{with} \qquad \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}.$$

We have $\nabla f = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix} \right)$ with eigenvalues $\lambda = \pm\iota$.

These eigenvalues are purely imaginary; hence the solution oscillates but does neither converge nor diverge. This is indeed the case since the exact solution is $x(t) = \cos(t)$.

We conclude from the domains of stability shown above that both Euler and midpoint methods have $|R(z)| > 1$ for purely imaginary z; hence they diverge when applied to the above ODE.

Moreover, knowing $R(z)$ allows us to precisely predict the rate with which they diverge, see `harmonic_oscillator_divergence()`.

Implicit Runge-Kutta Methods

Discussion

The above shows that the constraint $|R(\lambda\,\Delta t)| < 1$ imposes a limit on how large the step size Δt can be. Our next goals are therefore to determine a formula for $R(z)$ for arbitrary Runge-Kutta methods, and figuring out how to construct Runge-Kutta methods which do not have a step size constraint.

Implicit Runge-Kutta Methods

Stability function for abstract Runge-Kutta method

Consider a general Runge-Kutta scheme with Butcher tableau

$$\left(\begin{array}{c|c} \theta & V \\ \hline & w^T \end{array}\right).$$

When applied to the ODE $\dot{y} = \lambda y$, the numerical solution $\tilde{y}(t)$ after a single step is given by

$$\tilde{y}(t) = y_0 + w^T \mathbf{f}\, t \quad \text{where} \quad \mathbf{f} = \lambda \left(y_0 + V\, \mathbf{f}\, t \right).$$

You can verify the above formula by comparing it against the formula provided in the summary of Lecture 16.

Solving the second formula for $\mathbf{f}$ yields ($\mathbf{1}$ denotes the vector of all ones)

$$\mathbf{f} = \lambda \left(1 - \lambda t\, V \right)^{-1} \mathbf{1}\, y_0,$$

and inserting this expression into the formula for $\tilde{y}(t)$ yields

$$\tilde{y}(t) = \Big(1 + \lambda t\, w^T \left(I - \lambda t\, V \right) \mathbf{1} \Big)\, y(0).$$

Stability function for abstract Runge-Kutta method (continued)

Replacing all instances of λt with z in the above formula, we conclude that the stability function for the abstract Runge-Kutta scheme

$$\left(\begin{array}{c|c} \theta & V \\ \hline & w^T \end{array}\right)$$

is given by

$$R(z) = 1 + z\, w^T \big(I - z\, V\big)\, \mathbf{1}.$$

Example: stability function for Euler method

Butcher tableau:

$$\left(\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}\right)$$

Stability function:

$$R(z) = 1 + z\, 1 (1 - 0) 1 = 1 + z.$$

Implicit Runge-Kutta Methods

Example: stability function for midpoint method

Butcher tableau:

$$\left(\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}\right)$$

Stability function:

$$\begin{aligned} R(z) &= 1 + z \begin{pmatrix} 0 & 1 \end{pmatrix} \left(I - z \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= 1 + z \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{z}{2} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= 1 + z \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 + \frac{z}{2} \end{pmatrix} \\ &= 1 + z \left(1 + \frac{z}{2} \right) \\ &= 1 + z + \frac{z^2}{2}. \end{aligned}$$

Implicit Runge-Kutta Methods

Discussion

Recall the formula for the stability function given above:

$$\left(\begin{array}{c|c} \theta & V \\ \hline & w^T \end{array}\right) \quad \longrightarrow \quad R(z) = 1 + z\, w^T \big(I - z\, V\big)^{-1} \mathbf{1}.$$

We observe that $R(z)$ is a rational function for all V and w, and one can easily show that $R(z)$ is a polynomial if V is strictly lower triangular using Cramer's rule.

All Runge-Kutta methods that we have seen so far have a strictly lower-triangular V. This is for a good reason, as the example on the next slide shows.

Implicit Runge-Kutta Methods

Example: Implicit Euler

Consider the Runge-Kutta method with Butcher tableau

$$\left(\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}\right)$$

The corresponding one-step equations are

$$\tilde{y}(t) = y_0 + f_1\, t, \quad f_1 = f\big(y_0 + f_1\, t\big) \quad \iff \quad \tilde{y}(t) = y_0 + f\big(\tilde{y}(t)\big)\, t.$$

This method is called the implicit Euler method because the one-step equation has the same form as the (explicit) Euler method that we have seen before, but the argument to $f(y)$ is now $\tilde{y}(t)$ rather than y_0.

In quadrature terms, explicit Euler corresponds to a left-point rule while implicit Euler corresponds to a right-point rule.

Implicit Runge-Kutta Methods

Discussion

We conclude that if V is not strictly lower triangular, then the one-step equations become implicit, i.e. $\tilde{y}(t)$ can no longer be computed by simply evaluating a given formula, but rather we have to solve a potentially nonlinear equation or maybe even system of equations.

This motivates the following terminology.

- A RK scheme is called *explicit* if V is strictly lower triangular.
- A RK scheme is called *implicit* if V is not strictly lower triangular.

It depends on the context whether implicit one-step equations are a problem. For example, if $f(y) = Ay$ for some $A \in \mathbb{R}^{n \times n}$, the implicit Euler equation becomes

$$\tilde{y}(t) = y_0 + A\tilde{y}(t)\,t \quad \iff \quad \tilde{y}(t) = (I - At)^{-1}\,y_0.$$

If n is small, we can solve $(I - At)^{-1}\,y_0$ reliably and cheaply using the LU factorisation. If n is large, we may have to use iterative methods like Krylov or multigrid, which can become expensive since we have to solve a new linear system $(I - At)^{-1}\,y_0$ for every time step.

Implicit Runge-Kutta Methods

Implicit Runge-Kutta methods (continued)

Also, recall that iterative methods may fail to converge, which can be frustrating if your ODE solver breaks down due to a failure of the nested iterative solver.

Of course, the above remarks regarding iterative linear solvers also apply if $f(y)$ is nonlinear and we have to solve

$$\tilde{y}(t) - y_0 - f\left(\tilde{y}(t)\right) t = 0$$

using an iterative nonlinear solver like Newton's method.

We conclude that going from explicit to implicit Runge-Kutta methods may or may not introduce difficulties depending on the properties of $f(y)$. Next, let us look into why we are interested in implicit Runge-Kutta methods in the first place.

Implicit Runge-Kutta Methods

Stability functions of explicit Runge-Kutta methods

Recall the formula for the stability function given above:

$$\left(\begin{array}{c|c} \theta & V \\ \hline & w^T \end{array}\right) \quad \longrightarrow \quad R(z) = 1 + z\, w^T \big(I - z\, V\big)^{-1} \mathbf{1}.$$

We have seen:

- Explicit Runge-Kutta methods have a strictly lower-triangular V.
- Strictly lower-triangular V implies $R(z)$ is a polynomial.

For polynomial $R(z)$, we necessarily have

$$|z| \to \infty \quad \Longrightarrow \quad |R(z)| \to \infty.$$

This implies that the stability domain $\big\{z \in \mathbb{C} \mid |R(z)| < 1\big\}$ is bounded, which in turn implies that all explicit Runge-Kutta methods have a stability-induced step-size constraint.

By contrast, $R(z)$ is a rational function if V is arbitrary, and rational functions can be bounded for $|z| \to \infty$. This is the one and only reason why one would ever consider implicit Runge-Kutta methods.

Implicit Runge-Kutta Methods

Remark: determining stability functions

Stability functions can be determined in either of two ways.

- Write down the Butcher tableau use the formula given above,

$$\left(\begin{array}{c|c} \theta & V \\ \hline & w^T \end{array}\right) \quad \longrightarrow \quad R(z) = 1 + z\, w^T \big(I - z\, V\big)^{-1}\, \mathbf{1}.$$

- Write down the one-step equations for $\dot{y} = y$ and rearrange them into the form $\tilde{y}(t) = R(t)\, y_0$.

The first approach is convenient if you have a computer to do the linear algebra for you, the second approach is easier if you have to do the calculations by hand.

Implicit Runge-Kutta Methods

Example: stability function for the implicit Euler method

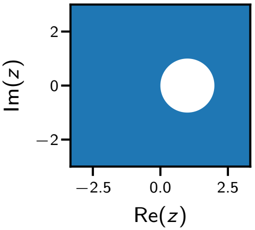
The one-step equation for implicit Euler is $\tilde{y}(t) = y_0 + f\left(\tilde{y}(t)\right) t$.
Inserting $f(y) = y$ yields

$$\tilde{y}(t) = y_0 + \tilde{y}(t)\, t \quad \iff \quad \tilde{y}(t) = \frac{y_0}{1-t};$$

hence the stability function is $R(z) = \frac{1}{1-z}$, and the stability domain is

$$\left\{z \mid |R(z)| < 1\right\} = \left\{z \mid \tfrac{1}{|1-z|} < 1\right\} = \left\{z \mid |1-z| > 1\right\}$$

which is the complement of the ball of radius 1 around $z = 1$.



Implicit Runge-Kutta Methods

[To be continued]