MA5233 Computational Mathematics

Lecture 10: Runge-Kutta Methods

Simon Etter

Semester I, AY 2020/2021

Runge-Kutta Methods

Introduction

The aim of this lecture is to develop numerical methods for solving problems of the following form.

Def: Ordinary differential equation (ODE)

Given $f : \mathbb{R}^n \to \mathbb{R}^n$, $y_0 \in \mathbb{R}^n$ and $T > 0$, find $y : [0, T) \to \mathbb{R}^n$ such that

$$y(0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = f\big(y(t)\big) \quad \text{for all } t \in [0, T).$$

$\dot{y} = y' = \frac{dy}{dt}$ is a shorthand notation for taking derivatives of a function $y : \mathbb{R} \to \mathbb{R}^n$.

Outlook

The following slides illustrate the above definition by discussing three example ODEs. These ODEs will reappear later in this lecture, so pay attention even if you are already familiar with ODEs.

Runge-Kutta Methods

Example 1

Consider the problem of finding $y : [0, \infty) \to \mathbb{R}$ such that

$$y(0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = \lambda\, y(t)$$

for some given $y_0, \lambda \in \mathbb{R}$.

The solution to this problem is given by

$$y(t) = y_0 \, \exp(\lambda t)$$

because this function satisfies

$$y(0) = y_0 \, \exp(\lambda 0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = y_0 \, \exp(\lambda t)\, \lambda = \lambda\, y(t).$$

Runge-Kutta Methods

Example 2

Consider the problem of finding $y : [0, \frac{1}{y_0}) \to \mathbb{R}$ such that

$$y(0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = y(t)^2$$

for some given $y_0 \in \mathbb{R}$.

The solution to this problem is given by

$$y(t) = \frac{y_0}{1 - y_0 t}$$

because this function satisfies

$$y(0) = \frac{y_0}{1 - y_0\, 0} = y_0 \qquad \text{and} \qquad \dot{y}(t) = \frac{y_0^2}{(1 - y_0 t)^2} = y(t)^2.$$

Note that this $y(t)$ is undefined for $t = \frac{1}{y_0}$. This shows that unlike the solution of Example 1, the solution to this ODE has a finite domain of definition $[0, \frac{1}{y_0})$.

The formula for $y(t)$ is well defined for $t > \frac{1}{y_0}$, but this part of the solution has no mathematical meaning since it is disconnected from the starting point $t = 0$.

Runge-Kutta Methods

Example 3

Consider the problem of finding $x : [0, \infty) \to \mathbb{R}$ such that

$$x(0) = 1 \qquad \text{and} \qquad \ddot{x} = -x.$$

The solution to this equation is given by

$$x(t) = \cos(t)$$

since this function satisfies $x(0) = \cos(0) = 1$ and

$$\ddot{x}(t) = \frac{d^2}{dt^2}\cos(t) = -\frac{d}{dt}\sin(t) = -\cos(t) = -x(t).$$

$\ddot{x} = -x$ is not of the form $\dot{y} = f(y)$, but it can easily be reduced to this form: if we set

$$y = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad \text{and} \qquad f(y) = \begin{pmatrix} y[2] \\ -y[1] \end{pmatrix},$$

then this $y(t)$ and $f(y)$ indeed satisfy

$$\dot{y} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} y[2] \\ -y[1] \end{pmatrix} = f(y).$$

This shows that considering ODEs involving only first-order derivatives is enough to cover ODEs involving arbitrarily high derivatives.

Runge-Kutta Methods

Real-world example: Newton's law of motion

Consider a point particle of mass $m > 0$ subject to a force $F(x) \in \mathbb{R}^3$ which depends on the location $x \in \mathbb{R}^3$ of the particle.

Newton's law of motion then states that the trajectory $x(t) \in \mathbb{R}^3$ of this particle satisfies

$$m\,\ddot{x}(t) = F\big(x(t)\big).$$

This equation is again not of the form $\dot{y} = f(y)$ but can reduced to this form by setting

$$y = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}, \qquad f(y) = \begin{pmatrix} y[2] \\ \frac{1}{m}\,F(y[1]) \end{pmatrix}$$

such that

$$\dot{y} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \frac{1}{m}\,F(x) \end{pmatrix} = \begin{pmatrix} y[2] \\ \frac{1}{m}\,F(y[1]) \end{pmatrix} = f(y).$$

Runge-Kutta Methods

Time-dependent ODEs

Many textbooks allow the ODE-defining function $f(y, t)$ to also depend on the time variable t. I omit this generality here because it is rarely needed in applications and it significantly complicates the analysis.

Runge-Kutta Methods

ODEs vs PDEs

ODEs are similar to PDEs in that the problem is to find a function $y(t)$ given an equation in terms of $y(t)$ and its derivatives which is to hold at every point in the domain.

Historically / formally, the defining property of an ODE is that the unknown $y(t)$ depends on only a single variable and hence the derivatives in the differential equations are *ordinary* derivatives.

By contrast, the unknown $u(x_1, \ldots, x_d)$ in a PDE depends on several variables and hence the derivatives are *partial* derivatives.

The terms "ODE" and "PDE" are hardly ever used in this way anymore, however.

In modern terminology, the defining property of an ODE is that fixed values for $y(t)$ and its derivatives are specified at a single point t_0. For this reason, ODEs are also called *initial value problems*.

The defining property of a PDE is that fixed values of $u(x)$ and its derivatives are specified at two or more points $x \in \partial\Omega$. For this reason, PDEs are also called *boundary value problems*.

Runge-Kutta Methods

Example: ODEs vs PDEs

The one-dimensional Poisson equation $-u''(x) = f(x)$ is an ODE in the formal sense because there is only a single independent variable x.

However, this equation is usually called a PDE because it is almost always paired with boundary conditions rather than initial conditions and hence it is much more similar to e.g. the higher-dimensional Poisson equation $-\Delta u = f$ than to Newton's law of motion $m\ddot{x} = F(x)$.

Runge-Kutta Methods

Discussion

As mentioned earlier, our goal in this lecture is to develop numerical algorithms which approximate the map

$$\big(f(y), y_0, T\big) \quad \mapsto \quad y(t) \quad \text{such that} \quad y(0) = y_0, \quad \dot{y}(t) = f(y(t)).$$

However, it is advisable to first establish that this map is well defined and well conditioned, since otherwise we might end up constructing (and trusting!) numerical solutions $\tilde{y}(t)$ which have no meaning.

It turns out that the key condition guaranteeing the existence of solutions $y(t)$ is Lipschitz continuity as defined on the next slide.

Runge-Kutta Methods

Def: (Global) Lipschitz continuity

A function $f : D \to \mathbb{R}^n$ with $D \subset \mathbb{R}^n$ is called *(globally) Lipschitz continuous with Lipschitz constant* $L > 0$ if for all $y_1, y_2 \in D$ we have

$$\left\| f(y_1) - f(y_2) \right\| \leq L \left\| y_1 - y_2 \right\|.$$

Note that whether a function $f(y)$ is Lipschitz continuous does not depend on the choice of norm due to norm equivalence in finite dimensions. However, the Lipschitz constant L may depend on the choice of norm.

Def: Local Lipschitz continuity

A function $f : \mathbb{R}^n \to \mathbb{R}^n$ is called *locally Lipschitz continuous* if for every $y_0 \in \mathbb{R}^n$ there exists a pair $\delta, L > 0$ such that for all $y_1, y_2 \in \mathbb{R}^n$ we have

$$\left\| y_k - y_0 \right\| \leq \delta \qquad \Longrightarrow \qquad \left\| f(y_1) - f(y_2) \right\| \leq L \left\| y_1 - y_2 \right\|.$$

Runge-Kutta Methods

Discussion

Lipschitz continuity of a function $f(y)$ is usually most conveniently verified using the following result.

Corollary

Assume $D \subset \mathbb{R}^n$ is convex and $f : D \to \mathbb{R}^n$ is differentiable everywhere in D. Then, $f(y)$ is locally Lipschitz continuous, and it is globally Lipschitz continuous if $\|\nabla f\|$ is bounded.

Proof. Immediate corollary of the result on the next slide.

Runge-Kutta Methods

Lemma: Lipschitz constants and derivatives

Assume $D \subset \mathbb{R}^n$ is convex and $f : D \to \mathbb{R}^n$ has a bounded derivative. Then,

$$\|f(y_1) - f(y_2)\| \leq L \|y_1 - y_2\| \qquad \text{where} \qquad L = \sup_{y \in D} \|\nabla f(y)\|$$

Proof. According to the chain rule, we have that

$$\frac{d}{dt}\Big(f\big(y_1 + t\,(y_2 - y_1)\big)\Big) = \nabla f\big(y_1 + t\,(y_2 - y_1)\big)\,(y_2 - y_1)$$

and hence we conclude using the fundamental theorem of calculus that

$$\begin{aligned}
\|f(y_1) - f(y_2)\| &= \left\| \int_0^1 \nabla f\big(y_1 + t\,(y_2 - y_1)\big)\,(y_2 - y_1)\,dt \right\| \\
&\leq \int_0^1 \left\| \nabla f\big(y_1 + t\,(y_2 - y_1)\big) \right\| \left\| y_2 - y_1 \right\| dt \\
&\leq \left(\sup_{y \in D} \|\nabla f(y)\| \right) \left\| y_2 - y_1 \right\|.
\end{aligned}$$

Runge-Kutta Methods

Discussion

Armed with the definition of Lipschitz continuity, we can now establish that the ODE function $\big(f(y), y_0, T\big) \mapsto y(t)$ is indeed well defined if $f(y)$ is Lipschitz continuous.

Picard-Lindelöf theorem

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous. Then, there exists a unique function $y : [0, \infty) \to \mathbb{R}^n$ such that

$$y(0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = f\big(y(t)\big) \quad \text{for all } t \in [0, \infty).$$

Proof. Beyond the scope of this module.

Example

Consider the function $f(y) = \lambda y$ from the example on slide 3. Since $f'(y) = \lambda$ is bounded for all y, this function is globally Lipschitz continuous and hence the solution $y(t) = y_0 \exp(\lambda t)$ exists for all $t \geq 0$.

Runge-Kutta Methods

Discussion

The above result assumes global Lipschitz continuity of $f(y)$ but in returns guarantees that the solution is well defined for all times $t \geq 0$.

If $f(y)$ is only local Lipschitz, then the following result establishes that the solution $y(t)$ is well defined at least on some interval $[0, T] \subset [0, \infty)$.

Picard-Lindelöf theorem, local version

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous. Then, there exists a $T > 0$ and a unique function $y : [0, T) \to \mathbb{R}^n$ such that

$$y(0) = y_0 \qquad \text{and} \qquad \dot{y}(t) = f\big(y(t)\big) \quad \text{for all } t \in [0, T).$$

Proof. Beyond the scope of this module.

Example

Consider the function $f(y) = y^2$ from the example on slide 4. Since $f'(y) = 2y$ exists but is unbounded $\mathbb{R}$, this function is locally Lipschitz but not globally Lipschitz. Correspondingly, the solution $y(t) = \frac{y_0}{1 - y_0 t}$ is defined only on the finite interval $[0, \frac{1}{y_0})$ but not on all of $[0, \infty)$.

Runge-Kutta Methods

Discussion

In order to approximate the ODE map $\big(f(y), y_0, T\big) \mapsto y(t)$ numerically, we need this map to be not only well defined but also continuous with respect to y_0; if this is not the case, then any small perturbation in y_0 (e.g. floating-point rounding) may lead to arbitrarily large errors in the solution $y(t)$.

We will see on the next slide that continuity is a simple consequence of the following auxiliary result.

Gronwall's inequality

$$\dot{y}(t) \leq \lambda\, y(t) \qquad \Longrightarrow \qquad y(t) \leq \exp(\lambda t)\, y(0).$$

Proof. Consider $z(t) = \exp(-\lambda t)\, y(t)$. Then, $z(0) = y(0)$ and

$$\begin{aligned} \dot{z}(t) &= -\lambda\, \exp(-\lambda t)\, y(t) + \exp(-\lambda t)\, \dot{y}(t) \\ &\leq -\lambda\, \exp(-\lambda t)\, y(t) + \exp(-\lambda t)\, \lambda\, y(t) = 0; \end{aligned}$$

hence $z(t) \leq y(0)$ and thus $y(t) \leq y(0)\, \exp(\lambda t)$.

Runge-Kutta Methods

Thm: Lipschitz continuity of the ODE map

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous with Lipschitz constant L, and assume $y_1, y_2 : [0, T) \to \mathbb{R}^n$ satisfy $\dot{y}_k = f(y_k)$. Then,

$$\|y_1(t) - y_2(t)\| \;\leq\; \exp(Lt)\,\|y_1(0) - y_2(0)\| \qquad \text{for any } t < T.$$

Proof. We have for any $y : \mathbb{R} \to \mathbb{R}^n$ that

$$\tfrac{d}{dt}\|y(t)\| = \lim_{\tilde{t}\to t} \tfrac{\|y(\tilde{t})\| - \|y(t)\|}{\tilde{t}-t} \leq \lim_{\tilde{t}\to t} \tfrac{\|y(\tilde{t}) - y(t)\|}{\tilde{t}-t} = \|\dot{y}(t)\|.$$

Combining the above with the Lipschitz continuity of $f(y)$, we obtain

$$\begin{aligned} \tfrac{d}{dt}\|y_2(t) - y_1(t)\| &\leq \|\dot{y}_2(t) - \dot{y}_1(t)\| \\ &= \|f(y_2(t)) - f(y_1(t))\| \\ &\leq L\,\|y_2(t) - y_1(t)\| \end{aligned}$$

from which the claim follows by Gronwall's inequality.

Runge-Kutta Methods

Discussion

The bound on the previous slide is a two-sided coin:

- Good: error at time t is proportional to error at time 0.
- Bad: constant of proportionality is $\exp(Lt)$.

The exponential function has a noticeable two-stage character:

- For $t \lesssim \frac{1}{L}$, $\exp(Lt)$ is about 1.
- For $t \gtrsim \frac{1}{L}$, $\exp(Lt)$ grows very quickly.

In practice, this means that there is often a characteristic time-scale $\frac{1}{L}$ beyond which numerical simulations become unreliable.

For example, we can predict the weather fairly accurately for the next 2-3 days, but predictions beyond one week are virtually impossible.

Runge-Kutta Methods

Solving ODEs using quadrature

We have now established that if $f(y)$ is Lipschitz continuous, then $\dot{y} = f(y)$ has a unique solution and this solution is a Lipschitz continuous function of the initial conditions y_0.

Let us now move on to discuss numerical methods for approximating $y(t)$.

According to the fundamental theorem of calculus, we have

$$\left.\begin{aligned} y(0) &= y_0 \\ \dot{y} &= f(y) \end{aligned}\right\} \qquad \Longleftrightarrow \qquad y(t) = y_0 + \int_0^t f(y(\tau))\, d\tau.$$

Given a quadrature rule $(\theta_k, w_k)_{k=1}^s$ for $[0,1]$, we can hence compute a numerical approximation to $y(t)$ using the formula

$$y(t) \approx y_0 + \sum_{k=1}^{s} f\big(y(\theta_k t)\big)\, w_k t.$$

The quadrature points θ_k and weights w_k are multiplied by t because we need to scale the quadrature rule from $[0,1]$ to $[0,t]$.

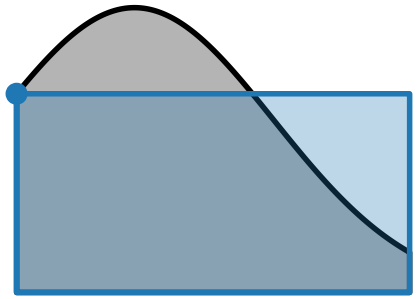
However, we need to be careful because we do not know $y(t)$ for $t > 0$ and hence we cannot evaluate $f\big(y(\theta_k t)\big)$ for quadrature points $\theta_k > 0$.

Runge-Kutta Methods

Solving ODEs using quadrature (continued)

The above suggests that we solve ODEs using the 1-point quadrature rule

$$\theta_1 = 0, \qquad w_1 = 1 \qquad \longleftrightarrow$$



.

This yields the following ODE solver.

Def: Euler step

Approximate the solution to $\dot{y} = f(y)$ using

$$y(t) \approx y(0) + f\big(y(0)\big)\, t.$$

Discussion

The error of this solver is easily estimated using Taylor's theorem.

Runge-Kutta Methods

Thm: Error estimate for Euler step

Assume $f(y)$ is differentiable, $y(t)$ satisfies $\dot{y} = f(y)$, and $t > 0$. Then, there exists $\xi \in (0, t)$ such that

$$y(t) \;=\; \underbrace{y(0) + f(y(0))\, t}_{\text{Euler step}} \;+\; \underbrace{\tfrac{1}{2}\, \nabla f(\xi)\, \dot{y}(\xi)\, t^2}_{\text{error}}.$$

Proof. According to Taylor's theorem, there exists $\xi \in (0, t)$ such that

$$y(t) = y(0) + \dot{y}(0)\, t + \tfrac{1}{2}\, \ddot{y}(\xi)\, t^2.$$

The claim follows by noting that since $\dot{y}(t) = f(y(t))$, we have

$$\dot{y}(0) = f(y(0)) \qquad \text{and} \qquad \ddot{y}(\xi) = \nabla f(y(\xi))\, \dot{y}(\xi).$$

Runge-Kutta Methods

Discussion

The error estimate

$$y(t) = \underbrace{y(0) + f(y(0))\, t}_{\text{Euler step}} + \underbrace{\tfrac{1}{2} \nabla f(\xi)\, \dot{y}(\xi)\, t^2}_{\text{error}}$$

gives us two important pieces of information.

- The error is smaller the closer $f(y)$ and $y(t)$ are to being constant (i.e. the smaller their derivative).

  This is not surprising since the quadrature rule $\theta_1 = 0$, $w_1 = 1$ is exact for constant integrands.

- The error is smaller for smaller times t.

  This is again not surprising because for smaller t, the integral $\int_0^t f(y(\tau))\, d\tau$ is both smaller and the integrand $f(y(\tau))$ is closer to being constant over the interval $[0, t]$.

Both the ODE function $f(y)$ and the time interval $[0, T]$ are imposed on us by the application; hence the Euler step presented above gives us no means to improve the accuracy by investing more compute time.

This circumstance can be remedied using the composition idea presented on the next slide.

Runge-Kutta Methods

Def: Euler's method

Approximate the solution to $\dot{y} = f(y)$ using

$$\tilde{y}(0) = y(0), \qquad \tilde{y}(t_k) = \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,(t_k - t_{k-1})$$

where $(t_k)_k$ denotes a given temporal mesh.

Def: Temporal mesh

An ordered sequence of points $0 = t_0 < t_1 < \ldots$.

Numerical demonstration

See `euler_step()`, `integrate()` and `example()`.

Discussion

We intuitively expect that composing Euler steps into Euler's method yields a numerical approximation $\tilde{y}(t)$ which converges to the exact solution $y(t)$ as the mesh width $\max_k |t_k - t_{k-1}|$ goes to 0.

I will next show that this is indeed the case, and I will do so using the notions of exact and Euler time propagators introduced on the next slide.

Runge-Kutta Methods

Def: (Exact) time propagator

The *(exact) time propagator* associated with $f : \mathbb{R}^n \to \mathbb{R}^n$ is the function

$$\Phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n, \qquad \Phi(y_0, t) = y(t)$$

where $y(t)$ is the solution to $\dot{y} = f(y)$, $y(0) = y_0$.

Def: Euler time propagator

The *Euler time propagator* associated with $f : \mathbb{R}^n \to \mathbb{R}^n$ is the function

$$\tilde{\Phi} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n, \qquad \tilde{\Phi}(y_0, t) = y_0 + f(y_0)\, t.$$

Runge-Kutta Methods

Discussion

It will be important in the following that the exact time propagator $\Phi(y_0, t)$ is a Lipschitz continuous function of y_0. We have already seen on slide 17 that this is indeed the case, so all we have to do is to translate that result into the new notation.

Thm: Lipschitz continuity of the exact time propagator

If $f(y)$ is Lipschitz continuous with Lipschitz constant L, then $\Phi(y_0, t)$ is Lipschitz continuous in y_0 with Lipschitz constant $\exp(Lt)$, i.e.

$$\left\|\Phi(y_1, t) - \Phi(y_2, t)\right\| \;\leq\; \exp(Lt)\left\|y_1 - y_2\right\|.$$

Proof. Immediate consequence of the result on slide 17.

Runge-Kutta Methods

Shorthand notations

In order to keep the statement and proof of the error estimate for Euler's method manageable, I will use the shorthand notations

$$y_k = y(t_k), \qquad \tilde{y}_k = \tilde{y}(t_k),$$

and

$$\Phi_k(y_0) = \Phi\big(y_0, t_k - t_{k-1}\big), \qquad \tilde{\Phi}_k(y_0) = \tilde{\Phi}\big(y_0, t_k - t_{k-1}\big).$$

As usual, $(t_k)_k$ denotes some temporal mesh specified by the context.

With this, we now have all the pieces in place show that Euler's method indeed converges.

Runge-Kutta Methods

Thm: Error estimate for Euler's method

Denote by $y(t)$ the solution to $\dot{y} = f(y)$ and by $\tilde{y}(t)$ the numerical approximation obtained by Euler's method with mesh $(t_k)_{k=0}^n$.

Then,

$$\|\tilde{y}_n - y_n\| \leq \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\, \|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\|,$$

assuming $f(y)$ is Lipschitz continuous with Lipschitz constant L.

Proof.

$$\begin{aligned}
\|\tilde{y}_n - y_n\| &= \|\tilde{\Phi}_n(\tilde{y}_{n-1}) - \Phi_n(y_{n-1})\| && \text{(definition of time propagators)} \\
&\leq \|\tilde{\Phi}_n(\tilde{y}_{n-1}) - \Phi_n(\tilde{y}_{n-1})\| + \|\Phi_n(\tilde{y}_{n-1}) - \Phi_n(y_{n-1})\| && \text{(triangle ineq.)} \\
&\leq \|\tilde{\Phi}_n(\tilde{y}_{n-1}) - \Phi_n(\tilde{y}_{n-1})\| + \exp\big(L\,(t_n - t_{n-1})\big)\, \|\tilde{y}_{n-1} - y_{n-1}\| && (\Phi \text{ Lipschitz}) \\
&\leq \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\, \|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\|. && \text{(recursion)}
\end{aligned}$$

Runge-Kutta Methods

Terminology

$\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$ is called the *local*, *consistency* or *truncation error* of the numerical time propagator $\tilde{\Phi}$.

Remark

The bound

$$\left\|\tilde{y}_n - y_n\right\| \leq \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\ \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$$

can be put into words as follows.

> The error at the final time t_n is upper-bounded by the sum of the errors introduced in the time steps $(t_{k-1} \to t_k)_{k=1}^{n}$ multiplied by the Lipschitz constant of $\Phi(y,t)$ derived on slide 17.

Runge-Kutta Methods

Discussion

The above bound shows that if

$$\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| \to 0 \qquad \text{in the limit} \qquad |t_k - t_{k-1}| \to 0, \quad (1)$$

then

$$\max_k |t_k - t_{k-1}| \to 0 \qquad \text{implies that} \qquad \|\tilde{y}_n - y_n\| \to 0.$$

We have already seen on slide 21 that (1) is indeed the case, so once again all we have to is to translate the old result into the new notation.

Lemma: Consistency of Euler time propagator

The Euler rule time propagator $\tilde{\Phi}(y_0, t)$ satisfies

$$\left\|\tilde{\Phi}(y_0, t) - \Phi(y_0, t)\right\| = O(t^2) \qquad \text{for } t \to 0.$$

Proof. Immediate consequence of the result on slide 21.

Runge-Kutta Methods

Thm: Convergence of Euler's method

Denote by $y(t)$ the solution to $\dot{y} = f(y)$ and by $\tilde{y}(t)$ the numerical approximation obtained using Euler's method with the equispaced temporal mesh $(t_k = \frac{k}{n}\, T)_{k=0}^{n}$.

Then,

$$\left\|\tilde{y}(T) - y(T)\right\| = O\left(\exp(LT)\,\frac{T^2}{n}\right) \qquad \text{for both } n, T \to \infty,$$

assuming $f(y)$ is Lipschitz continuous with Lipschitz constant L.

Proof.

$$\begin{aligned}
\left\|\tilde{y}_n - y_n\right\| &\leq \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\, \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| \\
&\leq n\, \exp\big(L\,(t_n - t_0)\big)\, O\left(\left(\tfrac{T}{n}\right)^2\right) \\
&= \exp\big(L\,(t_n - t_0)\big)\, O\left(\tfrac{T^2}{n}\right)
\end{aligned}$$

Numerical demonstration

See `convergence()`.

Runge-Kutta Methods

Remark 1

Note that Euler's method is second-order consistent but only first-order convergent. The heuristic reason for this is that Euler's method makes n errors of magnitude $O\left(n^{-2}\right)$; hence the total error is $n\,O(n^{-2}) = O(n^{-1})$.

Remark 2

The above convergence estimate indicates that the number of steps required to reach a fixed error tolerance

$$\left\|\tilde{y}(T) - y(T)\right\| = O\left(\exp(LT)\,\frac{T^2}{n}\right) \leq \tau$$

is $n = O\left(\tau^{-1}\,T^2\,\exp(LT)\right)$.

This once again indicates that it is difficult to solve ODEs on time-scales larger than one over the Lipschitz constant.

See `nsteps()` for numerical demonstration.

Runge-Kutta Methods

Discussion

The theorem on slide 30 establishes that Euler's method

$$\tilde{y}(0) = y(0), \qquad \tilde{y}\big(t_k\big) = \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,\big(t_k - t_{k-1}\big)$$

converges to the exact solution $y(t)$, but it also shows that the rate of convergence is only

$$\big\|\tilde{y}(T) - y(T)\big\| = O\big(n^{-1}\big).$$

This slow convergence is ultimately due to the fact that Euler's method is based on the very poor quadrature approximation

$$\int_0^t f(y(\tau))\,d\tau \;\approx\; f(y(0))\,t;$$

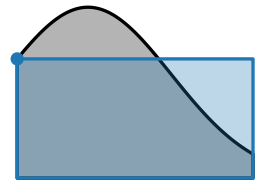
hence the question arises whether we can improve the speed of convergence by choosing a more accurate quadrature rule.

Runge-Kutta Methods

Discussion (continued)

The reason why we settled for the "left-point rule"

$$\theta_1 = 0, \qquad w_1 = 1 \qquad \longleftrightarrow$$



.

on slide 20 was that we assumed that we do not know $y(t)$ at times $t > 0$ and hence we cannot have quadrature points $\theta_k > 0$.

However, this constraint no longer applies now that we know how to (approximately) propagate $y(0)$ into the future using Euler's step.

For example, we can use Euler's step to obtain an estimate

$$\tilde{y}_1(t) \;=\; y(0) + f(y(0))\, t,$$

and then we can use a trapezoidal rule approximation

$$\tilde{y}(t) = y(0) + \Big(f\big(y(0)\big) + f\big(\tilde{y}_1(t)\big) \Big)\, \tfrac{t}{2},$$

to improve this estimate. This idea is known as the trapezoidal rule.

Runge-Kutta Methods

Def: Trapezoidal rule

Approximate the solution to $\dot{y} = f(y)$ using $\tilde{y}(0) = y(0)$ and

$$\tilde{y}_1(t_k) = \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,\big(t_k - t_{k-1}\big),$$

$$\tilde{y}(t_k) = \tilde{y}(t_{k-1}) + \Big(f\big(\tilde{y}(t_{k-1})\big) + f\big(\tilde{y}_1(t_k)\big)\Big)\,\tfrac{t_k - t_{k-1}}{2},$$

where $(t_k)_k$ denotes a given temporal mesh.

Terminology: Quadrature rules vs. ODE solvers

It is common to refer to both the above ODE solver as well as the quadrature rule from which it is derived as "trapezoidal rule". This is not a problem since it is always clear from context whether the ODE solver or the quadrature rule is meant.

Notation: Distinguishing approximations to the same value

Note that the trapezoidal rule computes two different numerical approximation $\tilde{y}_1(t_k)$ and $\tilde{y}(t_k)$ to the same exact value $y(t_k)$. Here and throughout this lecture, I will distinguish these approximations by writing $\tilde{y}_i(t_k)$, $i = 1, 2, 3, \ldots$, for the auxiliary approximations, and $\tilde{y}(t_k)$ without a subscript for the final approximation.

Runge-Kutta Methods

Implementation of the trapezoidal rule

See `trapezoidal_step()` and `integrate()`.

Note that `f1 = t * f(y0)` is used both in

`f2 = t * f(y0 + f1)` and `y0 + (f1 + f2)/2` .

Introducing `f1` hence avoids having to evaluate `t * f(y0)` twice.

Runge-Kutta Methods

Terminology: Bootstrapping

Recall that we derived the trapezoidal rule ODE solver by combining the trapezoidal quadrature rule

$$\int_0^t f(y(\tau))\,d\tau \approx \Big(f(y(0)) + f(y(t))\Big)\,\tfrac{t}{2}$$

with an Euler step to produce an estimate for $y(t)$.

This idea of using a simple procedure to fill in some blanks in a more complicated procedure appears repeatedly in applied mathematics, statistics and computer science and is known as *bootstrapping*.

This terminology is derived from the idiom "to pull oneself over the fence by one's bootstrap" which is used to mock claims of having achieved obviously impossible feats.

Example: "I completed the Computational Mathematics assignment in just one hour" - "Yeah right, and I pulled myself over the fence by my bootstraps."

Picture of a bootstrap: `https://i.stack.imgur.com/qjyj6.png`

Runge-Kutta Methods

Convergence of the trapezoidal rule

We have derived on slide 27 the estimate

$$\left\|\tilde{y}_n - y_n\right\| \;\leq\; \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\; \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$$

which bounds the total error in terms of the local errors. Looking back at the proof, we realise that the same bounds also holds for the trapezoidal rule if we just replace the Euler time propagator

$$\tilde{\Phi}(y_0, t) = y_0 + f(y_0)\, t$$

with the following.

Def: Trapezoidal rule time propagator

The *trapezoidal rule time propagator* associated with $f : \mathbb{R}^n \to \mathbb{R}^n$ is

$$\tilde{\Phi}(y_0, t) = y_0 + \big(f(y_0) + f(y_1)\big)\,\tfrac{t}{2} \qquad \text{where} \qquad y_1 = y_0 + f(y_0)\, t,$$

or equivalently,

$$\tilde{\Phi}(y_0, t) = y_0 + f(y_0)\,\tfrac{t}{2} + f\big(y_0 + f(y_0)\, t\big)\,\tfrac{t}{2}.$$

Runge-Kutta Methods

Convergence of the trapezoidal rule (continued)

All we have to do to obtain a convergence estimate for the trapezoidal rule method is hence to estimate the order of convergence of the trapezoidal rule consistency error.

Lemma: Consistency of trapezoidal rule

The trapezoidal rule time propagator $\tilde{\Phi}(y_0, t)$ satisfies

$$\left\|\tilde{\Phi}(y_0, t) - \Phi_k(y_0, t)\right\| = O(t^3) \qquad \text{for } t \to 0.$$

Proof. The proof amounts to showing that the Taylor series of $\tilde{y}(t) = \tilde{\Phi}(y_0, t)$ agrees with the Taylor series of $y(t) = \Phi(y_0, t)$ up to (and including) the second-order term.

I therefore compute

$$\begin{aligned}
\tilde{y}(t) &= y_0 + f(y_0)\,\tfrac{t}{2} + f\left(y_0 + f(y_0)\,t\right)\tfrac{t}{2} \\
\dot{\tilde{y}}(t) &= f(y_0)\,\tfrac{1}{2} + f'\left(y_0 + f(y_0)\,t\right) f(y_0)\,\tfrac{t}{2} + f\left(y_0 + f(y_0)\,t\right)\tfrac{1}{2} \\
\ddot{\tilde{y}}(t) &= f''\left(y_0 + f(y_0)\,t\right) f(y_0)^2\,\tfrac{t^2}{4} + 2\,f'\left(y_0 + f(y_0)\,t\right) f(y_0)\,\tfrac{1}{2}
\end{aligned}$$

and conclude that we indeed have

$$\tilde{y}(0) = y_0 = y(0), \quad \dot{\tilde{y}}(t) = f(y_0) = \dot{y}(0), \quad \ddot{\tilde{y}}(t) = f'(y_0)\,f(y_0) = \ddot{y}(0).$$

Runge-Kutta Methods

Remark

The above proof is analogous to the proof on slide 21 which showed that Euler's step is second-order consistent.

Thm: Convergence of the trapezoidal rule

Denote by $y(t)$ the solution to $\dot{y} = f(y)$ and by $\tilde{y}(t)$ the numerical approximation obtained using the trapezoidal rule with the equispaced temporal mesh $(t_k = \frac{k}{n}\, T)_{k=0}^{n}$.

Then,

$$\left\|\tilde{y}(T) - y(T)\right\| = O\left(\exp(LT)\, \frac{T^3}{n^2}\right) \qquad \text{for both } n, T \to \infty,$$

assuming $f(y)$ is Lipschitz continuous with Lipschitz constant L.

Proof. Analogous to the one on slide 30.

Numerical demonstration

See `convergence()`.

Runge-Kutta Methods

Euler's method vs. the trapezoidal rule

The above result shows that the trapezoidal rule indeed achieves our goal of improving on the $O(n^{-1})$ convergence of Euler's method.

However, comparing the time propagation equations

$$\text{Euler:} \quad \tilde{y}(t_k) = \tilde{y}(t_{k-1}) + f\left(\tilde{y}(t_{k-1})\right)(t_k - t_{k-1}),$$

$$\text{Trapezoidal:} \quad \tilde{y}(t_k) = \tilde{y}(t_{k-1}) + \left(f\left(\tilde{y}(t_{k-1})\right) + f\left(\tilde{y}_1(t_k)\right)\right)\frac{t_k - t_{k-1}}{2}$$

$$\text{where} \qquad \tilde{y}_1(t_k) = \tilde{y}(t_{k-1}) + f\left(\tilde{y}(t_{k-1})\right)(t_k - t_{k-1}),$$

we also observe that the improved speed of convergence of the trapezoidal rule comes at the price of more computations per time step.

It is therefore not clear whether the trapezoidal rule is indeed faster than Euler's method.

Runge-Kutta Methods

Euler's method vs. the trapezoidal rule (continued)

We can answer this question by observing that if we have a convergence estimate

$$\|\tilde{y}(T) - y(T)\| = O(n^{-p})$$

and need to achieve an error tolerance

$$\|\tilde{y}(T) - y(T)\| \leq \tau,$$

then we need to take $O(\tau^{-1/p})$ time steps and hence we incur an $O(w\,\tau^{-1/p})$ runtime where w denotes the runtime per time step.

This shows that increasing the order of convergence p at the expense of a larger runtime per time step w pays off whenever the error tolerance τ is small enough.

Runge-Kutta Methods

Discussion

Given the above, one may wonder whether we can invest even more work per time step to achieve an even better order of convergence and hence a smaller runtime in the limit $\tau \to 0$.

The answer is yes, and in fact we can do so simply by continuing the bootstrapping procedure which lead us to the trapezoidal rule.

Algorithms derived from this idea are known as *Runge-Kutta methods*.

Runge-Kutta Methods

Def: Runge-Kutta method

Approximate the solution to $\dot{y} = f(y)$ using

1: $\tilde{y}(0) = y(0)$,
2: **for** $k = 1, 2, \ldots$ **do** (time stepping loop)
3: **for** $i = 1, \ldots, s$ **do** (stage loop)
4: $$\tilde{y}_i = \tilde{y}(t_{k-1}) + \sum_{j=1}^{i-1} f\left(\tilde{y}_j\right) W_{ij} \left(t_k - t_{k-1}\right) \quad \approx y(t_{k-1} + \theta_i (t_k - t_{k-1}))$$
5: **end for**
6: $$\tilde{y}(t_k) = \tilde{y}(t_{k-1}) + \sum_{j=1}^{s} f\left(\tilde{y}_j\right) w_j \left(t_k - t_{k-1}\right)$$
7: **end for**

where

- $(t_k)_k$ denotes a given temporal mesh,
- $(\theta_j, W_{ij})_{j=1}^{i-1}$ with $i \in \{1, \ldots, s\}$ denotes a sequence of quadrature rules for the intervals $[0, \theta_i]$, and
- $(\theta_i, w_i)_{i=1}^{s}$ is a quadrature rule for $[0, 1]$,
- $s \in \mathbb{N}$ is called the *number of stages* of the Runge-Kutta scheme.

Runge-Kutta Methods

Discussion

The parameters $\theta \in \mathbb{R}^s$, $W \in \mathbb{R}^{s \times s}$ and $w \in \mathbb{R}^s$ introduced in the above definition are most conveniently represented in a tabular form known as *Butcher tableau*.

Def: Butcher tableau

$$\begin{array}{c|ccc} \theta_1 & w_{11} & \dots & w_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_s & w_{s1} & \dots & w_{ss} \\ \hline & w_1 & \dots & w_s \end{array}$$

The examples on the next slide might help you to better understand the definition of Runge-Kutta schemes and Butcher tableaus.

Runge-Kutta Methods

Example 1: Euler's method

Tableau

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

Interpretation

$$\begin{aligned} \tilde{y}_1(t_{k-1}) &= \tilde{y}(t_{k-1}) \\ \hline \tilde{y}(t_k) &= \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,(t_k - t_{k-1}) \end{aligned}$$

Example 2: Trapezoidal rule

Tableau

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

Interpretation

$$\begin{aligned} \tilde{y}_1(t_{k-1}) &= \tilde{y}(t_{k-1}) \\ \tilde{y}_2(t_k) &= \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,(t_k - t_{k-1}) \\ \hline \tilde{y}(t_k) &= \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,\tfrac{t_k - t_{k-1}}{2} + f(\tilde{y}_2)\,\tfrac{t_k - t_{k-1}}{2} \end{aligned}$$

Runge-Kutta Methods

Remark

Note that the quadrature weight matrix $W \in \mathbb{R}^{s \times s}$ associated with the auxiliary quantities $\tilde{y}_i$ must be strictly lower triangular, i.e. we must have

$$i \leq j \implies W_{ij} = 0.$$

If this was not the case, then we would not be able to compute the auxiliary variables using simple forward substitution.

Example

Tableau		Interpretation
1	1	$\tilde{y}_1(t_k) = \tilde{y}(t_{k-1}) + f\left(\tilde{y}_1(t_k)\right) t$
	1	$\tilde{y}(t_k) = \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,(t_k - t_{k-1})$

Computing $\tilde{y}_1(t_k)$ would require solving

$$\tilde{y}_1(t_k) = \tilde{y}(t_{k-1}) + f\left(\tilde{y}_1(t_k)\right) t.$$

Depending on the ODE, this could be a high-dimensional nonlinear system of equations!

Runge-Kutta Methods

Discussion

The above definition allows us to turn any set of parameters θ, W and w into a Runge-Kutta method, but of course not all methods constructed in this way are equally interesting.

Specifically, there is no point considering a particular Runge-Kutta method if there is another method with a comparable runtime per time step but better accuracy.

Combining this insight with the observation that the runtime per time step is mainly determined by the number of stages s, we conclude that a particular Runge-Kutta method is only interesting if its order of convergence is as large as possible for the given s.

Runge-Kutta Methods

Convergence theory of Runge-Kutta methods

Determining Runge-Kutta methods with maximal order of convergence is straightforward.

- Like Euler's method and the trapezoidal rule, Runge-Kutta methods construct an approximate solution $\tilde{y}(t)$ by repeatedly applying a particular numerical time propagator $\tilde{\Phi}(y_0, t)$;
- Any Runge-Kutta solution $\tilde{y}(t_n)$ hence satisfies the bound

$$\left\|\tilde{y}(t_n) - y(t_n)\right\| \leq \sum_{k=1}^{n} \exp\bigl(L\,(t_n - t_k)\bigr)\, \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$$

  derived on slide 27.
- The order of convergence is thus maximised if we choose θ, W and w such that the Taylor series of the Runge-Kutta time propagator $\tilde{\Phi}(y_0, t)$ agrees with the one of the exact time propagator $\Phi(y_0, t)$ as far as possible.

Runge-Kutta Methods

Convergence theory of Runge-Kutta methods (continued)

The only obstacle in the above procedure is that computing derivatives of $\Phi(y_0, t)$ and $\tilde{\Phi}(y_0, t)$ becomes more and more tedious as the order of the derivative and the number of stages of the Runge-Kutta method increase.

Fortunately, you will hardly ever have to do these computations yourself, because other people have already done them and they have catalogued their findings for us. See

`https://en.wikipedia.org/wiki/List_of_Runge-Kutta_methods`.

In practice, determining a suitable Runge-Kutta method is hence as simple as skimming the appropriate Wikipedia list.

Runge-Kutta Methods

Example

Runge-Kutta ODE solvers are named after two German mathematicians who proposed the following scheme and showed that it is a fourth-order method.

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & & \frac{1}{2} & & \\
1 & & & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

I demonstrate in `rk4_step()` how to implement this scheme and in `convergence()` that it is indeed a fourth-order method.

Runge-Kutta Methods

Number of stages vs. order of convergence

We have now seen three particular Runge-Kutta methods.

Method	# stages	Convergence
Euler	1	$O(n^{-1})$
Trapezoidal	2	$O(n^{-2})$
RK4	4	$O(n^{-4})$

Based on this table, you might expect but that optimal s-stage Runge-Kutta schemes achieve $O(n^{-s})$ convergence, but this is not true; it has been shown that the optimal order of convergence p as a function of the number of stages s is given by

s	1	2	3	4	5	6	7	8	9	10	...
p	1	2	3	4	4	5	6	6	7	8	...

.

For this reason, Runge-Kutta schemes with $s > 4$ stages are rarely used.

Runge-Kutta Methods

Adaptive Runge-Kutta methods

So far, I have used equispaced temporal meshes $(t_k = \frac{k}{n}\, T)_{k=0}^{n}$ in my numerical experiments and when estimating the rate of convergence of Runge-Kutta methods.

Equispaced meshes are convenient due to their simplicity, but they do not deliver optimal performance: we have seen that the global error is upper-bounded by

$$\left\|\tilde{y}(t_n) - y(t_n)\right\| \;\leq\; \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\; \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$$

and that the local error satisfies

$$\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1}) = \tfrac{1}{p!} \left(\tilde{y}^{(p)}(\xi) - y^{(p)}(\xi)\right) \left(t_k - t_{k-1}\right)^p$$

for some $\xi \in (t_{k-1}, t_k)$, where p denotes the order of consistency (e.g. $p = 2$ for Euler's method and $p = 3$ for the trapezoidal rule).

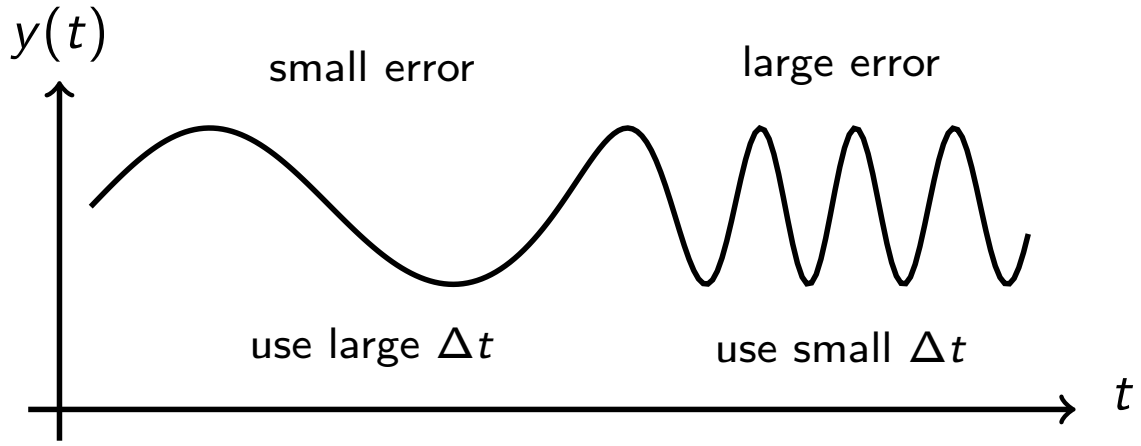
Runge-Kutta Methods

Adaptive Runge-Kutta methods (continued)

If we choose the same time step $t_k - t_{k-1}$ for all k, then the local error will therefore be small in areas where $\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\|$ is small and large in areas where $\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\|$ is large.

Since the global error is just a weighted sum of the local errors, we therefore conclude that we could have increased the step size $t_k - t_{k-1}$ in areas where $\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\|$ is small without significantly increasing the overall error since the overall error is dominated by the errors in the areas where $\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\|$ is large.

I assume here that $t_n \lesssim \frac{1}{L}$ such that all exponential prefactors are $O(1)$. This is justified since we have already seen that there is little hope to compute $\tilde{y}(t_n)$ accurately if $t_n \gg \frac{1}{L}$.



Runge-Kutta Methods

Adaptive Runge-Kutta methods (continued)

The above indicates that we would ideally choose the temporal mesh $(t_k)_{k=0}^n$ such that the time steps $t_k - t_{k-1}$ are roughly proportional to the local curvature $\|\tilde{y}^{(p)} - y^{(p)}\|_{[t_{k-1},t_k]}$.

See `adaptive_example_1()` for a concrete example.

Determining such a mesh requires two procedures.

- A way to estimate the local curvature $\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\|$, or equivalently the local error $\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\|$.
- A rule for turning this information into step sizes $t_k - t_{k-1}$.

Of course, both of these procedures should be computationally feasible, i.e. they should assume that

- we do not know $t(t)$ for $t > t_{k-1}$, and
- we can only evaluate $f(y)$ at a finite number of points but have no other information about $f(y)$.

Unfortunately, it is not possible to provide a rigorous bound on the local error under these assumptions, see next slide.

Runge-Kutta Methods

"Thm:" Bounding the local error is impossible

No algorithm can compute a bound on $\|\tilde{\Phi}(y_0, t) - \Phi(y_0, t)\|$ using only a finite number of evaluations of $f(y)$.

"Proof." Consider an arbitrary algorithm which bounds $\|\tilde{\Phi}(y_0, t) - \Phi(y_0, t)\|$ by sampling $f(y)$ at a collection of points y_k.

If we pass this algorithm a function $f(y)$ such that $f(y_k) = 0$ for all k but $f(y) > 0$ otherwise, then the algorithm must assume that $f(y) = 0$ for all y and hence $\tilde{\Phi}(y_0, t) = \Phi(y_0, t) = y_0$ for all t, but actually we have $\tilde{\Phi}(y_0, t), \Phi(y_0, t) > y_0$ and in general $\tilde{\Phi}(y_0, t) \neq \Phi(y_0, t)$.

Hence this algorithm would predict that the local error is zero when actually it is positive.

I have put "Thm" and "Proof" in quotation marks because neither the statement nor the proof are fully rigorous. They convey the correct message, however, and that is what matters here.

Runge-Kutta Methods

Estimating the local error

Since computing a rigorous bound on the local error is impossible, we must instead rely on heuristics.

The most commonly employed heuristic is to use two Runge-Kutta methods of different orders of convergence and treat the difference between their solutions as an approximation of the error in the less accurate solution. This heuristic is justified by the following result.

Thm: Estimating the local error by comparing methods

Let $\tilde{\Phi}_1(y_0, t)$, $\tilde{\Phi}_2(y_0, t)$ be two Runge-Kutta time propagators such that the local errors satisfy

$$\left\|\tilde{\Phi}_\ell(y_0, t) - \Phi(y_0, t)\right\| = O(t^{p_\ell})$$

for two different exponents $p_1 < p_2$.

We then have

$$\left\|\tilde{\Phi}_1(y_0, t) - \tilde{\Phi}_2(y_0, t)\right\| = \left\|\tilde{\Phi}_1(y_0, t) - \Phi(y_0, t)\right\| + O(t^{p_1+1}).$$

Proof. See next slide.

Runge-Kutta Methods

Proof. According to Taylor's theorem, we have

$$\tilde{\Phi}_1(y_0,t) = y(0) + \dot{y}(0)\,t + \ldots + \frac{y^{(p_1)}(0)}{(p_1-1)!}\,t^{p_1-1} + C\,t^{p_1} + O\big(t^{p_1+1}\big)$$

$$\tilde{\Phi}_2(y_0,t) = y(0) + \dot{y}(0)\,t + \ldots + \frac{y^{(p_1)}(0)}{(p_1-1)!}\,t^{p_1-1} + \frac{y^{(p_1)}(0)}{p_1!}\,t^{p_1} + O\big(t^{p_1+1}\big)$$

$$\tilde{\Phi}(y_0,t) = y(0) + \dot{y}(0)\,t + \ldots + \frac{y^{(p_1)}(0)}{(p_1-1)!}\,t^{p_1-1} + \frac{y^{(p_1)}(0)}{p_1!}\,t^{p_1} + O\big(t^{p_1+1}\big)$$

for some constant $C \neq \frac{y^{(p_1)}(0)}{p_1!}$. We therefore conclude that

$$\begin{aligned}\tilde{\Phi}_1(y_0,t) - \tilde{\Phi}_2(y_0,t) &= \left(C - \frac{y^{(p_1)}(0)}{p_1!}\right) t^{p_1} + O\big(t^{p_1+1}\big) \\ &= \tilde{\Phi}_1(y_0,t) - \tilde{\Phi}(y_0,t) + O\big(t^{p_1+1}\big).\end{aligned}$$

Runge-Kutta Methods

Embedded Runge-Kutta methods

In addition to being exact to leading order, estimating local errors by comparing Runge-Kutta methods has the advantage that it often incurs little or no additional cost.

For example, the trapezoidal rule step

$$\begin{aligned} \tilde{y}_1(t) &= y(0) + f(y(0))\, t, \\ \tilde{y}(t) &= y(0) + \Big(f\big(y(0)\big) + f\big(\tilde{y}_1(t)\big) \Big) \, \tfrac{t}{2} \end{aligned}$$

already computes the Euler solution $\tilde{y}_1(t)$ and thus we can compute

$$\big\| \tilde{y}_1(t) - \tilde{y}(t) \big\| \approx \big\| \tilde{y}_1(t) - y(t) \big\|$$

without any additional cost compared to just evaluating the trapezoidal solution $\tilde{y}(t)$, see `embedded_ET_step()`.

Pairs of Runge-Kutta methods where the lower-order method uses only intermediate quantities of the higher-order method are called *embedded*.

Embedded Runge-Kutta methods of various orders are known, see

https://en.wikipedia.org/wiki/List_of_Runge-Kutta_methods#Embedded_methods.

Runge-Kutta Methods

Choosing the step size

Now that we know how to estimate the local error, we can address the problem of choosing the step sizes $t_k - t_{k-1}$ such that the local errors are approximately the same everywhere, i.e. such that

$$\left\|\tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| \approx \tau \text{ independent of } k.$$

This problem can be solved in many ways, but the most efficient is to note that

$$\left\|\tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| = O\big((t_k - t_{k-1})^p\big)$$

implies that

$$\left\|\tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \Phi(\tilde{y}_{k-1})\right\| \approx C_k\,(t_k - t_{k-1})^p$$

for some $C_k > 0$ as long as $t_k - t_{k-1} > 0$ is small enough, and thus

$$\left\|\tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| \approx \tau \qquad \Longleftrightarrow \qquad t_k - t_{k-1} = \left(\tfrac{\tau}{C}\right)^{1/p}.$$

This formula for $t_k - t_{k-1}$ can be evaluated as soon as we have an estimate for C_k, and such an estimate can be obtained by solving for C_k the error estimate for the previous time step, which is

$$\left\|\tilde{\Phi}_{1,k-1}(\tilde{y}_{k-2}) - \Phi_{2,k-1}(\tilde{y}_{k-2})\right\| \approx C_k\,(t_{k-1} - t_{k-2})^{p_1}.$$

Runge-Kutta Methods

Choosing the step size (continue)

Putting all of the above together, we obtain the formula

$$t_k - t_{k-1} = \left(t_{k-1} - t_{k-2}\right) \left(\frac{\tau}{\left\| \tilde{\Phi}_{1,k-1}(\tilde{y}_{k-2}) - \Phi_{2,k-1}(\tilde{y}_{k-2}) \right\|} \right)^{1/p_1} . \quad (2)$$

This formula provides a first practical rule for determining a temporal mesh $(t_k)_{k=0}^n$ such that all local errors are approximately equal to τ, but it is usually not quite accurate enough on its own.

Most adaptive ODE solvers therefore check whether the above choice for the step width $t_k - t_{k-1}$ indeed yields

$$\left\| \tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \tilde{\Phi}_{2,k}(\tilde{y}_{k-1}) \right\| \ \leq \ \tau,$$

and if not then (2) is iterated until we find a step which such that the local error tolerance is satisfied. Furthermore, since rejecting and recomputing steps is expensive, we usually multiply (2) by a "safety factor" (e.g. 0.9) to increase our chances of accepting $t_k - t_{k-1}$.

Code describes this better than words, so I strongly recommend that you have a look at `propagate_adaptively()`.

Runge-Kutta Methods

Adaptive step selection applied to $\dot{y} = -y$

`adaptive_example_1()` demonstrates that the heuristics encoded in `propagate_adaptively()` work fairly well for this particular example.

Encouraged by this finding, I apply the same code to the simple ODE $\dot{y} = -y$, $y(0) = 1$ in `adaptive_example_2()`. Since the exact solution is

$$y(t) = \exp(-t) \qquad \text{and thus} \qquad y^{(p)}(t) = (-1)^p \exp(-t),$$

and our step selection routine tries to find step sizes $t_k - t_{k-1}$ such that

$$\left\| \tilde{\Phi}_{1,k}(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1}) \right\| \;\approx\; \tfrac{1}{p!} \left\| \tilde{y}^{(p)}(t) - y^{(p)}(t) \right\| (t_k - t_{k-1})^p \;\approx\; \tau,$$

we expect that our step size selection routine should result in a temporal mesh $(t_k)_{k=0}^n$ such that

$$t_k - t_{k-1} \;\approx\; (p!\,\tau)^{1/p} \exp(t/p),$$

i.e. the mesh width should increase exponentially with t. However, we observe that this is true only up to $t \approx 15$ and after that the step size remains constant. See also `stepsize()` for further numerical evidence.

Runge-Kutta Methods

Adaptive step selection applied to $\dot{y} = -y$ (continued)

This discrepancy between our expectations and the numerical results is the result of a wrong interpretation of Taylor series.

We have seen on slide 21 that Euler's step satisfies

$$\tilde{\Phi}(y_0, t) - \Phi(y_0, t) \;=\; \tfrac{1}{2}\,\ddot{y}(t)\,t^2 + O(t^3),$$

and throughout the above analysis we assumed that t is small and hence the $O(t^3)$ term is negligible.

In `adaptive_example_2()`, this assumption is satisfied at the very left end of the plot, but as we move towards the right the step size gradually increases and thus the $O(t^3)$ may no longer be negligible.

Fortunately, our step selection routine detects that at some point it can no longer increase the step size without increasing the error and therefore keeps the step size constant, but the flip side of this coin is that doing so results in a loss of performance: even though $y(t \geq 10) = 0$ up to the resolution of the plot, we are nevertheless not allowed to choose large step sizes because doing so would result in large errors.

Runge-Kutta Methods

Runge-Kutta methods applied to $\dot{y} = -y$

To get an even better insight into what is going on in this example, let us derive explicit formulae for the Euler and trapezoidal rule propagators.

- Euler: $\tilde{y}(t) \;=\; y_0 + f(y_0)\,t \;=\; y_0 - y_0 t \;=\; (1-t)\,y_0.$
- Trapezoidal:
$$\begin{aligned}\tilde{y}(t) &= y_0 + \Big(f(y_0) + f\big(y_0 + f(y_0)\,t\big)\Big)\,\tfrac{t}{2} \\ &= y_0 + \Big(-y_0 - \big(y_0 - y_0 t\big)\Big)\,\tfrac{t}{2} \\ &= \big(1 - t + \tfrac{t^2}{2}\big)\,y_0.\end{aligned}$$

This shows that after k steps with constant step size Δt, the Euler and trapezoidal rule solutions are given by

$$\tilde{y}(k\,\Delta t) = R(-\Delta t)^k\,y(0),$$

where

$$R(z) = \begin{cases} 1 + z & \text{for Euler's method, and} \\ 1 + z + \frac{z^2}{2} & \text{for the trapezoidal rule.} \end{cases}$$

Runge-Kutta Methods

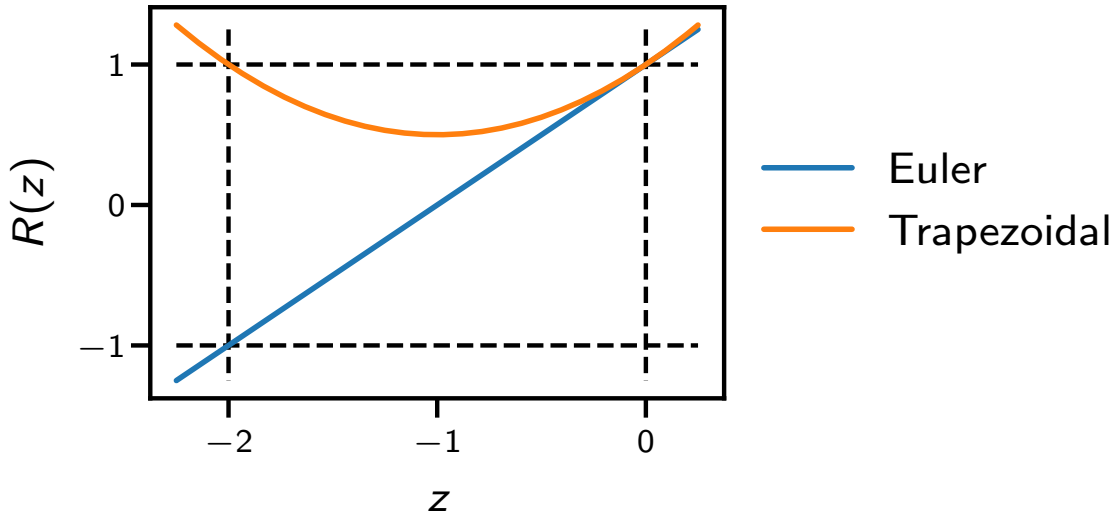
Runge-Kutta methods applied to $\dot{y} = -y$ (continued)

We conclude from the above that the Runge-Kutta solutions $\tilde{y}(t)$

- grow exponentially if $|R(-\Delta t)| > 1$,
- are constant or oscillate if $|R(-\Delta t)| = 1$, and
- decay exponentially if $|R(-\Delta t)| = 1$.

In the case $\dot{y} = -y$, we know that the exact solution $y(t) = \exp(-t)$ decays exponentially and hence we want our numerical solutions to do the same.

Combining the above with the plots of $R(z)$ shown below, we conclude that this is the case if and only if $0 < \Delta t < 2$.



Runge-Kutta Methods

Runge-Kutta methods applied to $\dot{y} = -y$ (continued)

The above shows that we must run Euler's method and the trapezoidal rule with step sizes $t_k - t_{k-1} < 2$ or else they will produce solutions $\tilde{y}(t)$ which grow exponentially in t rather than decay exponentially.

Running `stepsize()` with `show_ref = true`, we observe that our adaptive step size routine was actually able to detect this. Furthermore, `stability_example()` shows that $\tilde{y}(t)$ indeed converges to 0 only if $\Delta t < 2$.

Runge-Kutta Methods

Linearisation of ODEs

The above discussion shows that solving the ODE $\dot{y} = -y$ using Runge-Kutta methods requires quite a bit of thought when choosing the step size $t_k - t_{k-1}$.

However, we already know that

$$\dot{y} = -y \qquad \Longrightarrow \qquad y(t) = y(0)\,\exp(-t),$$

so you may wonder why I spent so much time discussing how to numerically solve an ODE whose exact solution is already known.

The reason for this is the result presented on the next slide.

Def: Attractive fixed point / steady state

A point $y_F \in \mathbb{R}^n$ is said to be a *fixed point* or a *steady state* of the ODE $\dot{y} = f(y)$ if $f(y_F) = 0$. Furthermore, y_F is said to be *attractive* if all eigenvalues of $\nabla f(y_F)$ have a negative real part.

Runge-Kutta Methods

Thm: Linearisation of ODEs

Assume $y(t)$ solves the ODE $\dot{y} = f(y)$, which has an attractive fixed point at a point $y_F \in \mathbb{R}^n$, and assume the Jacobian $\nabla f(y_F)$ has the eigenvalue decomposition $\nabla f(y_F) = V \Lambda V^{-1}$. Then,

$$\frac{d}{dt}\Big(V^{-1}\big(y(t) - y_F\big)\Big) = \Lambda\Big(V^{-1}\big(y(t) - y_F\big)\Big) + O\Big(\big\|y(t) - y_F\big\|^2\Big),$$

i.e. each of the n components of the vector $w(t) = V^{-1}\big(y(t) - y_F\big)$ satisfies

$$\dot{w}_k(t) = \lambda_k\, w_k(t) + O\big(\|w\|^2\big).$$

Proof. We have

$$\begin{aligned}\frac{d}{dt}\big(y(t) - y_F\big) &= \dot{y}(t) - \frac{d}{dt} y_F = f\big(y(t)\big) + 0 \\ &= \underbrace{f(y_F)}_{0} + \underbrace{\nabla f(y_F)}_{V\Lambda V^{-1}}\big(y(t) - y_F\big) + O\big(\|y(t) - y_F\|^2\big).\end{aligned}$$

The claim follows by multiplying from the left with V^{-1}.

Runge-Kutta Methods

Example: Velocity of a falling pebble

The above result shows that the simple ODE $\dot{y} = \lambda y$ captures the essence of the dynamics of any ODE $\dot{y} = f(y)$ which approaches a steady state y_F.

A simple real-world example of this phenomenon would be a pebble which is dropped from a very tall building. Until this pebble hits the ground, the only forces acting on it are the gravitational force $F = g$ accelerating the pebble and the drag force $F = -v^2$ slowing it down. We therefore expect that the velocity v of this pebble satisfies the ODE

$$\dot{v} = g - v^2.$$

This ODE has a fixed point at $v_F = \sqrt{g}$, and we observe that this fixed point is attractive since

$$\frac{d}{dv}\left(g - v^2\right)\Big|_{v=\sqrt{g}} = -2v\Big|_{v=\sqrt{g}} = -2\sqrt{g} < 0.$$

Runge-Kutta Methods

Example: Velocity of a falling pebble (continued)

We therefore conclude that if we want to solve this ODE using either Euler's method or the trapezoidal rule, then we must choose a time step

$$\Delta t < \frac{2}{2\sqrt{g}} = \frac{1}{\sqrt{g}}$$

to ensure that the numerical solution $\tilde{v}(t)$ does not spuriously diverge from the steady state $v_F = \sqrt{g}$.

Being subject to a maximal time step Δt_{max} is of course never desirable, but it depends on the details of the simulation whether such a constraint is indeed a problem.

If the drop is short, then the pebble will be accelerating for most of the simulation time and the time step constrain $\Delta t < \frac{1}{\sqrt{g}}$ will likely not be a problem because we need small time steps anyway in order to accurately capture the acceleration.

If the drop is long, on the other hand, then most of the simulation time will be spent observing the pebble dropping straight down at a constant velocity, and the time step constraint is a major nuisance because it forces us to spend valuable compute time to track a phenomenon which is completely uninteresting from a physics point of view.

Runge-Kutta Methods

Stiffness

ODEs like the long pebble drop example discussed above are called *stiff* in the mathematical literature.

Managing stiffness is probably the single most important source of complications when solving ODEs in practice.

I will discuss techniques for doing so shortly, but before that let me introduce some terminology.

Def: Stability function

The *stability function* associated with a Runge-Kutta scheme is the function $R : \mathbb{C} \to \mathbb{C}$ such that the numerical time propagator for the equation $\dot{y} = \lambda y$ is given by

$$\tilde{\Phi}(y_0, t) = R(\lambda\, t)\, y_0.$$

Def: Stability domain

The stability domain associated with a Runge-Kutta method is the set

$$\Big\{ z \in \mathbb{C} \;\Big|\; |R(z)| < 1 \Big\}.$$

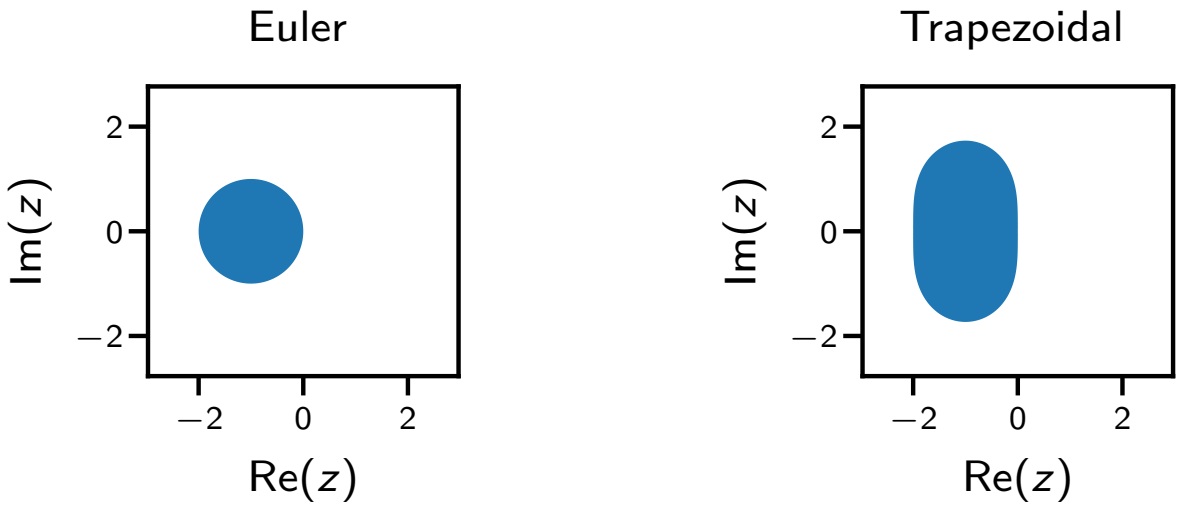
Runge-Kutta Methods

Example

We have derived on slide 63 two examples of stability functions, namely

- $R(z) = 1 + z$ for Euler's method, and
- $R(z) = 1 + z + \frac{z^2}{2}$ for the trapezoidal rule.

Plugging these functions into a plotting tool, we obtain the following stability domains.



The stability domain for Euler's method can also be determined using pen-and-paper analysis: the set

$$\{z \in \mathbb{C} \mid |R(z)| = |1 + z| < 1\}$$

is simply a ball of radius 1 centred at $z = -1$.

Runge-Kutta Methods

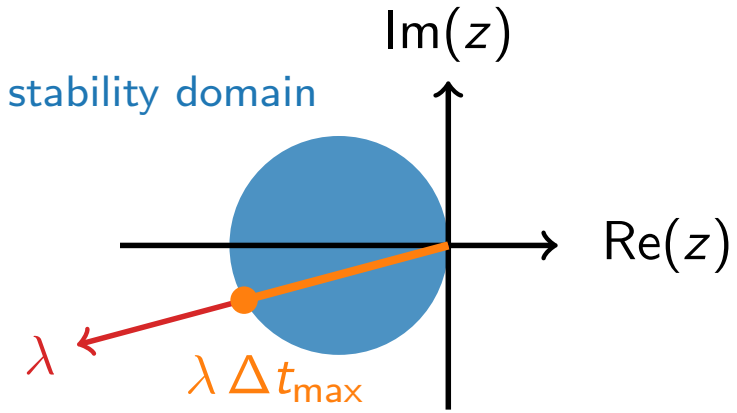
Determining step size constraints using stability domains

The purpose of the stability domain is to allow us to visually estimate the range of admissible time steps Δt such that the numerical solution

$$\tilde{y}(k\,\Delta t) = R(\lambda\,\Delta t)^k y_0$$

to $\dot{y} = \lambda y$ does not blow up: this range is proportional to the intersection between the stability domain $\{|R(z)| < 1\}$ and the ray $\lambda\,[0, \infty)$.

Furthermore, the largest admissible time step $\Delta t_{\max}$ is proportional to the point of maximum modulus in this intersection.



Runge-Kutta Methods

Discussion

We would of course like to have a Runge-Kutta method whose stability domain is the entire left half plane $\{\mathrm{Re}(z) < 0\}$ so that the numerical solution has the correct asymptotic behaviour for any $\lambda \in \mathbb{C}$.

Neither Euler's method nor the trapezoidal rule satisfy this criterion, so the question arises whether there exists a Runge-Kutta method with a stability domain as described above.

To answer this question, it is useful to have a formula for the stability function of a generic Runge-Kutta scheme. The next slide will derive such a formula.

Runge-Kutta Methods

Thm: Stability function formula

The stability function of a Runge-Kutta scheme with Butcher tableau

$$\left(\begin{array}{c|c} \theta & W \\ \hline & w^T \end{array}\right) \qquad \text{is given by} \qquad R(z) = 1 + z\, w^T \big(I - zW\big)^{-1} \mathbf{1},$$

where $\mathbf{1}$ denotes the vector of all ones.

Proof. The time propagator for the above Butcher tableau and the ODE $\dot{y} = \lambda y$ is given by

$$\tilde{\Phi}(y_0, t) = y_0 + \lambda t\, w^T \tilde{y} \qquad \text{where} \qquad \tilde{y} = y_0\, \mathbf{1} + \lambda t\, W \tilde{y}.$$

Solving the second equation for the vector of auxiliary variables $\tilde{y}$, we obtain

$$\tilde{y} = (I - \lambda t\, W)^{-1}\, y_0\, \mathbf{1},$$

and inserting this expression into the formula for $\tilde{\Phi}(y_0, t)$ yields

$$\tilde{\Phi}(y_0, t) = \Big(1 + \lambda t\, w^T (I - \lambda t\, W)^{-1} \mathbf{1}\Big)\, y_0 = R(\lambda t)\, y_0.$$

Runge-Kutta Methods

Example: Stability function for Euler's method

Butcher tableau:

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

Stability function:

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

Runge-Kutta Methods

Example: stability function for the trapezoidal rule

Butcher tableau:

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

Stability function:

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

Runge-Kutta Methods

Discussion

Now that we have a general formula for the stability function $R(z)$, let us look into what this formula tells us about the possible shapes of the stability domain.

The following result will be very useful for this endeavour.

Lemma

$$R(z) = 1 + z\, w^T \left(I - zW\right)^{-1} \mathbf{1}$$

is generally a rational function of z, and it is a polynomial in z if W is strictly lower triangular (i.e. $i \leq j \implies W_{ij} = 0$).

Proof. In principle a straightforward application of Cramer's rule to the factor $(I - zW)^{-1}\,\mathbf{1}$. Unfortunately, Cramer's rule itself is not very straightforward, so I omit the details.

Runge-Kutta Methods

Stability domains for polynomial and rational stability functions

The above result is important because if $R(z)$ is a polynomial, then we must necessarily have $\lim_{|z|\to\infty} |R(z)| = \infty$ and hence the stability domain $\left\{|R(z)| < 1\right\}$ is bounded.

[To be continued]

Runge-Kutta Methods

Summary

Theory of ODEs:

- $\dot{y} = f(y)$ has a unique (local) solution if $f(y)$ is (locally) Lipschitz continuous.
- If $f(y)$ has Lipschitz constant L, then $y(t)$ is a Lipschitz continuous function of $y(0)$, but the Lipschitz constant is $\exp(Lt)$.

Runge-Kutta Methods

Summary

Algorithm:

1: $\tilde{y}(0) = y(0)$,
2: **for** $k = 1, 2, \ldots$ **do**
3: **for** $i = 1, \ldots, s$ **do**
4: $$\tilde{y}_i^{(k)} = \tilde{y}(t_{k-1}) + \sum_{j=1}^{i-1} f\left(\tilde{y}_j^{(k)}\right) W_{ij} \left(t_k - t_{k-1}\right)$$
5: **end for**
6: $$\tilde{y}(t_k) = \tilde{y}(t_{k-1}) + \sum_{j=1}^{s} f\left(\tilde{y}_j^{(k)}\right) w_j \left(t_k - t_{k-1}\right)$$
7: **end for**

Butcher tableau representation:

$$\begin{array}{c|ccc} \theta_1 & w_{11} & \cdots & w_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_s & w_{s1} & \cdots & w_{ss} \\ \hline & w_1 & \cdots & w_s \end{array}$$

Runge-Kutta Methods

Summary

Convergence theory:

- Global error is weighted sum of local errors,

$$\left\|\tilde{y}_n - y_n\right\| \;\leq\; \sum_{k=1}^{n} \exp\big(L\,(t_n - t_k)\big)\; \left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|.$$

- Local / consistency errors $\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$ can be estimated using Taylor's theorem.

Adaptive time stepping:

- Time step formula:

$$t_k - t_{k-1} = S\,\big(t_{k-1} - t_{k-2}\big)\; \left(\frac{\tau}{\left\|\tilde{y}_1(t_k) - \tilde{y}_2(t_k)\right\|}\right)^{1/p_1}$$

where $\tilde{y}_1$, $\tilde{y}_2$ are two different Runge-Kutta solutions, $p_1 < p_2$ are their orders of consistency, and S denotes the safety factor.

Runge-Kutta Methods

Summary

Stability:

- Stability function: $R(z)$ such that

$$\dot{y} = \lambda y \quad \Longrightarrow \quad \tilde{y}(k\,\Delta t) = R(\lambda\,\Delta t)^k\, y_0.$$

- Domain of stability $\left\{z \in \mathbb{C} \mid |R(z)| < 1\right\}$.
- Pro of implicit methods: no step-size constraint.
  Con of implicit methods: we have to solve equations in every step.