MA3227 Numerical Analysis II

Lecture 6: Runge-Kutta Methods

Simon Etter

Semester II, AY 2020/2021

Runge-Kutta Methods

Problem statement

Given

$$f : \mathbb{R}^n \to \mathbb{R}^n, \qquad y_0 \in \mathbb{R}^n \qquad \text{and} \qquad T > 0,$$

determine $y : [0, T) \to \mathbb{R}^n$ such that

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

$\dot{y}$ is a shorthand notation for $\dot{y} = \frac{dy}{dt}$.

Terminology: Ordinary differential equations (ODEs)

Problems of the above form are known as *ordinary differential equations*.

ODEs are typically used to model time-evolution phenomena. For this reason, y_0 is called the *initial condition*, and T is called the *final time*.

Outlook

The following slides will illustrate the above problem statement by discussing several example 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

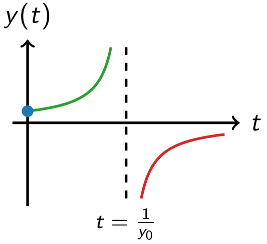
$$y(0) = \frac{y_0}{1 - y_0\, 0} = y_0 \qquad \text{and} \qquad \dot{y}(t) = \frac{y_0^2}{(1 - y_0 t)^2} = y(t)^2.$$

Runge-Kutta Methods

Example 2 (continued)

Note that the above solution diverges as t approaches $\frac{1}{y_0}$,

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



Example 2 is hence different from Example 1 in that the above $y(t)$ diverges already after a finite time $\frac{1}{y_0}$ while the solution in Example 1, namely

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

is finite for all $t \in [0, \infty)$.

Runge-Kutta Methods

Example 3

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

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

The solution to this equation is given by

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

since this function satisfies

$$x(0) = \cos(0) = 1, \quad \dot{x}(0) = -\sin(0) = 0$$

and

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

Runge-Kutta Methods

Example 3 (continued)

The ODE $\ddot{x} = -x$ is not of the form $\dot{y} = f(y)$, but it can be reduced to this form by setting

$$y = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad \text{and} \qquad f(y) = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix}$$

since then

$$\dot{y} = \begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix} = f(y).$$

This trick can be generalised to reduce ODEs with arbitrarily high derivatives to a system of ODEs involving only first-order derivatives.

Runge-Kutta Methods

Real-world example: Newton's law of motion

A famous real-world example of an ODE is Newton's law of motion

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

This equation relates the acceleration $\ddot{x}$ of a point particle of mass m to the forces $F\big(x(x)\big)$ acting on the particle at the current position $x(t)$.

Runge-Kutta Methods

ODEs vs PDEs

ODEs are similar to PDEs in the sense that the problem is to find a function $y(t)$ given an equation in terms of $y(t)$ and its derivatives.

Formally, the difference between ODEs and PDEs is the following.

- In an ODE, the unknown $y(t)$ depends on a single scalar variable and hence the derivatives are *ordinary* derivatives.
- In a PDE, the unknown $u(x_1, \dots, x_d)$ depends on several variables and hence the derivatives are *partial* derivatives.

The terms "ODE" and "PDE" are hardly ever used in this way, 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 closer in spirit 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

Outlook

Our main goal in this lecture is of course to develop numerical algorithms for evaluating the ODE 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, before doing so it is advisable to first study the conditions under which this map is well defined, i.e. the conditions which guarantee that the above problem has precisely one solution.

It turns out that these conidtions are fairly simple: all we need is that $f(y)$ is Lipschitz continuous. The following slides will explain further.

Runge-Kutta Methods

Def: (Global) Lipschitz continuity

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

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

A function which is Lipschitz continuous with some unspecified Lipschitz constant $L > 0$ is called simply *Lipschitz continuous*.

Picard-Lindelöf theorem, global version

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous and $T > 0$. 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.

Runge-Kutta Methods

The Picard-Lindelöf theorem indicates that understanding Lipschitz continuity is important for understanding ODEs.

The following result provides a convenient tool for establishing Lipschitz continuity of a given function $f(y)$.

Thm: Global Lipschitz continuity and differentiability

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous if $\|\nabla f\|$ is bounded.

Proof (not examinable). Immediate corollary of the result on the next slide (which I will skip in class).

Example

Recall from Example 1 on slide 3 the ODE

$$y(0) = y_0, \qquad \dot{y} = \lambda y.$$

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

Lemma: Lipschitz constants and derivatives (not examinable)

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ 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.

$$\|f(y_1) - f(y_2)\| \underset{\uparrow}{=} \left\| \int_0^1 \frac{d}{dt}\Big(f\left(y_1 + t\left(y_2 - y_1\right)\right)\Big)\,dt \right\|$$

(fundamental theorem of calculus)

$$\underset{\uparrow}{=} \left\| \int_0^1 \nabla f\left(y_1 + t\left(y_2 - y_1\right)\right)\left(y_2 - y_1\right) dt \right\|$$

(chain rule)

$$\underset{\uparrow}{\leq} \int_0^1 \left\|\nabla f\left(y_1 + t\left(y_2 - y_1\right)\right)\right\| \left\|y_2 - y_1\right\| dt$$

($\| \int f(t)\,dt\| \leq \int \|f(t)\|\,dx$ and matrix norm definition)

$$\leq \left(\sup_{y \in D} \|\nabla f(y)\|\right) \left\|y_2 - y_1\right\|.$$

Runge-Kutta Methods

The Picard-Lindelöf theorem on slide 12 assumes that $f(y)$ is globally Lipschitz but in return guarantees existence and uniqueness of the solution for all $t \geq 0$. There is also a version of the Picard-Lindelöf theorem which assumes that $f(y)$ is only *locally Lipschitz continuous* (see below) but in return guarantees existence and uniqueness of the solution only over some potentially finite interval $[0, T)$.

Def: Local Lipschitz continuity

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

$$\|y_1 - y_2\| \leq \delta \qquad \Longrightarrow \qquad \big\|f(y_1) - f(y_2)\big\| \leq L\,\|y_1 - y_2\|.$$

Picard-Lindelöf theorem, local version

Assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous and $y_0 \in \mathbb{R}^n$. 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.

Runge-Kutta Methods

Local Lipschitz continuity is again related to differentiability.

Thm: Local Lipschitz continuity and differentiability
A differentiable function $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous.

Proof (not examinable). Corollary of the result on slide 14.

Comparing the above against the analogous theorem for global Lipschitz continuity on slide 13, we conclude that the difference between local and global Lipschitz continuity is whether $\|\nabla f(y)\|$ is bounded.

Runge-Kutta Methods

Example

Recall from Example 2 on slide 4 the ODE

$$y(0) = y_0, \qquad \dot{y} = y^2.$$

Since $f'(y) = 2y$ exists but is unbounded, $f(y) = y^2$ is locally but not globally Lipschitz continuous and hence solutions may exist only over some finite interval $[0, T]$.

Indeed, we have observed previously that the solution

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

exists as a function $y : \mathbb{R} \to \mathbb{R}$ only on $[0, \frac{1}{y_0})$.

Runge-Kutta Methods

Continuity of the ODE map

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 the initial conditions; otherwise any small perturbation in y_0 (e.g. rounding errors) may lead to arbitrarily large errors in the solution $y(t)$.

Fortunately, it turns out that Lipschitz continuity of $f(y)$ guarantees not existence and uniqueness of solutions but also that these solutions are a Lipschitz continuous function of the initial conditions y_0.

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 $y_1, y_2 : [0, T) \to \mathbb{R}^n$ are two solutions to the same ODE $\dot{y}_k = f(y_k)$ but with different initial conditions $y_1(0)$ and $y_2(0)$. Then,

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

Proof (not examinable). See the following slides (which I will skip in class).

Runge-Kutta Methods

Lipschitz continuity of the ODE map is a consequence of the following auxiliary result.

Lemma: Gronwall's inequality (not examinable)

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

Proof (not examinable).
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

Proof of the theorem on slide 18 (not examinable).

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

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

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

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

The claim then follows by Gronwall's inequality.

Runge-Kutta Methods

Interpretation of the ODE continuity

In the above bound

$$\|y_1(t) - y_2(t)\| \ \le \ \exp(Lt)\ \|y_1(0) - y_2(0)\|,$$

$y_1(t)$ typically represents the exact solution and $y_2(t)$ represents an approximation to $y_1(t)$ resulting from a slightly perturbed initial condition $y_2(0)$. In this context, the above bound is both a blessing and a curse:

- Good: The error at time t is proportional to the error at time 0.
- Bad: The constant of proportionality is $\exp(Lt)$ and hence grows very rapidly once $t > \frac{1}{L}$.

The second point implies that solving ODEs over time spans longer than one over the Lipschitz constant is essentially impossible.

It is the mathematical foundation of phenomena like the butterfly effect, i.e. the claim that the occurrence of a tornado may depend on whether a butterfly flaps its wings.

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 solving ODEs.

It turns out that solving ODEs is in some sense equivalent to evaluating integrals, namely we have

$$y(0) = y_0, \quad \dot{y} = f(y) \qquad \Longleftrightarrow \qquad y(t) = y_0 + \int_0^t f(y(\tau))\, d\tau.$$

Numerically computing integrals is known as *quadrature*, and it is a problem that we already know how to solve. The following slides will recapitulate the basics.

Runge-Kutta Methods

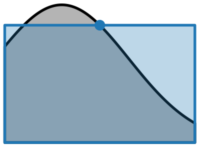
Def: Quadrature rule

A formula of the form

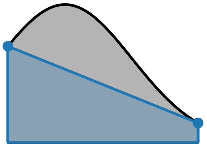
$$\sum_{k=1}^{n} f(x_k)\, w_k \approx \int_a^b f(x)\, dx$$

is called a *quadrature rule for* $[a, b]$. The parameters $(x_k \in \mathbb{R})_{k=1}^{n}$ and $(w_k \in \mathbb{R})_{k=1}^{n}$ are called *quadrature points* and *quadrature weights*, respectively.

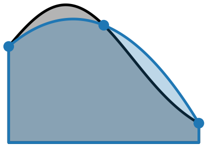
Example quadrature rules $\left(m = \frac{a+b}{2}\right)$


Midpoint rule

$(b-a)\, f(m)$


Trapezoidal rule

$\frac{b-a}{2}\,\big(f(a) + f(b)\big)$


Simpson's rule

$\frac{b-a}{6}\,\big(f(a) + 4\, f(m) + f(b)\big)$

Runge-Kutta Methods

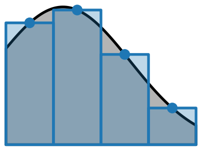
Composite quadrature

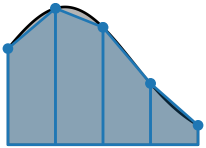
Quadrature rules generally become more accurate the larger the number of quadrature points. A simple way to construct quadrature rules with many points is to split the original integral into many small integrals,

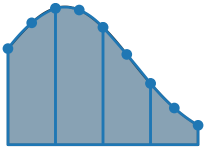
$$\int_a^b f(x)\,dx \;=\; \sum_{k=1}^{n} \int_{c_{k-1}}^{c_k} f(x)\,dx$$

and then apply a simple quadrature rule to each of these small integrals. This trick is known as *composite quadrature*.

Example composite quadrature rules


Composite midpoint


Composite trapezoidal


Composite Simpson

Runge-Kutta Methods

Solving ODEs using quadrature (continued)

Let us now return to the problem of evaluating the ODE integral

$$y(t) = y_0 + \int_0^t f(y(\tau))\,d\tau.$$

The first challenge that we face when trying to apply quadrature to this integral is that the integration interval $[0, t]$ is of variable rather than fixed width. This circumstance can be remedied by substituting $\tau = xt$, which yields

$$y(t) = y_0 + \int_0^{1} f(y(xt))\,t\,dx.$$

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

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

Runge-Kutta Methods

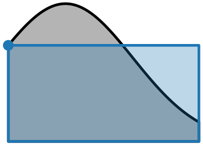
Solving ODEs using quadrature (continued)

The second challenge is that the highlighted values in

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

are available only if $x_k = 0$.

The only admissible quadrature rule is hence the *left-point rule* given by

$$x_1 = 0, \qquad w_1 = 1 \qquad \longleftrightarrow$$  .

This quadrature rule leads us to the approximation

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

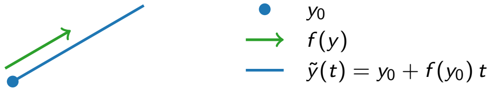
which is known as an *Euler step*.

Runge-Kutta Methods

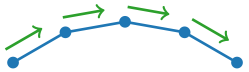
Visual interpretation of Euler's method

The right-hand side $f(y)$ in the ODE $\dot{y} = f(y)$ can be interpreted as the direction in which $y(t)$ should move given its current position.

In this mental model, the Euler step formula corresponds to looking at the direction once and then moving in this direction forever.



It is clear that this procedure will not lead to good approximations. We can obtain better approximations by using the Euler step formula to extrapolate only some small distance into the future and then update the direction $f(y)$.



This procedure is analogous to the composite quadrature idea and known as *Euler's method*.

Runge-Kutta Methods

Def: Euler's method

Approximating the solution to $y(0) = y_0$, $\dot{y} = f(y)$ using

$$\tilde{y}(0) = y_0, \qquad \tilde{y}\big(t_k\big) = \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,\big(t_k - t_{k-1}\big)$$

is known as *Euler's method*.

The t_k in this formula refer to a sequence of time points

$$0 = t_0 < t_1 < \ldots < t_{n-1} < t_n = T.$$

Such a sequence is called a *temporal mesh*.
For much of this lecture, I will be using the equispaced temporal mesh

$$\Big(\ t_k = \tfrac{T}{n}\,k\ \Big)_{k=0}^{n},$$

but it will occasionally be useful to also consider more general meshes.

Numerical demonstration

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

Runge-Kutta Methods

Runtime and Convergence of Euler's method

In addition to the ODE parameters $f(y)$, y_0, and T, Euler's method requires us to also choose a temporal mesh $(t_k)_{k=0}^n$.

We intuitively expect that choosing a mesh with a larger number of points n will lead to smaller errors but longer runtimes. Choosing a temporal mesh hence amounts to striking a balance between accuracy and performance.

As usual, the best way to quantify this error is to analyse separately how the runtime and accuracy depends on the number of time steps n.

This will be tackled on the following slides.

Runge-Kutta Methods

Thm: Runtime of Euler's method
n steps of Euler's method require n evaluations of $f(y)$ and $O(n)$ other operations.

Proof. Immediate consequence of the recursion equation

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

Note that I count exactly the number of ODE-right-hand-side evaluations $f(y)$ but I count only in the big O sense the number of other operations. The reason for this is that evaluating $f(y)$ is usually by far the most time-consuming part of any ODE solver; hence it is usually fair to say that an algorithm which performs x times as many $f(y)$ evaluations is also x times slower.

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,

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

assuming $f(y)$ is Lipschitz continuous with Lipschitz constant L.
Proof. See the following slides.

Numerical demonstration
See `convergence()`.

Runge-Kutta Methods

Verifying the above result requires us to first put into place quite a bit of mathematical machinery.

For starters, we need the notion of time propagators defined as follows.

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

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

Simply put, time propagators are functions which take in the current state y_0 and a time span t, and produce the (approximate) solution at time t in the future.

Runge-Kutta Methods

We have seen on slide 18 that the ODE solution $y(t)$ is a Lipschitz continuous function of the initial condition y_0.

This fact can be expressed in terms of the time propagator as follows.

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.

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

Proof. Immediate consequence of the result on slide 18.

Runge-Kutta Methods

Shorthand notations

I will use the shorthand notations

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

and

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

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

With this, we now have all the pieces in place to derive an error estimate for Euler's method.

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 temporal mesh $(t_k)_{k=0}^{n}$. Then,

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

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

Proof.

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

Runge-Kutta Methods

The factor $\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$ on the right-hand side of the above error estimate is known under several names.

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

The above bound

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

can hence 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 $\Phi(y, t)$ (see slide 18).

Runge-Kutta Methods

The above bound shows that the total error $\|\tilde{y}_n - y_n\|$ vanishes if all the local errors $\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\|$ vanish as the time steps $t_k - t_{k-1}$ go to zero. The following result shows that this is indeed the case.

Lemma: Consistency of Euler time propagator

The Euler 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.

$$\begin{aligned} \Phi(y_0, t) &= y(0) + \dot{y}(0)\, t + O(t^2) && \text{(apply Taylor)} \\ &= y_0 + f(y_0)\, t + O(t^2) && \text{(use } y(0) = y_0 \text{ and } \dot{y} = f(y)\text{)} \\ &= \tilde{\Phi}(y_0, t) + O(t^2) && \text{(definition of } \tilde{\Phi}(y_0, t)\text{)} \end{aligned}$$

Runge-Kutta Methods

The above property has a special name.

Terminology: pth-order consistency

A numerical time propagator $\tilde{\Phi}(y_0, t)$ such that

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

is said to be pth-order consistent.

We can now finally verify the claimed order of convergence of Euler's method by combining the above error bound and consistency result.

Proof of convergence of Euler's method (slide 31).

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

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 $O\big(n^{-2}\big)$; 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

$$\big\|\tilde{y}(T) - y(T)\big\| = O\Big(\exp(LT)\,\tfrac{T^2}{n}\Big) \leq \tau$$

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

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

Improving Euler's method

The theorem on slide 31 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

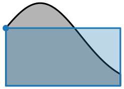
$$\int_0^t f(y(\tau))\,d\tau \;\approx\; f(y(0))\,t;$$

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

Runge-Kutta Methods

Improving Euler's method (continued)

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

$$x_1 = 0, \qquad w_1 = 1 \qquad \longleftrightarrow$$  .

on slide 26 was that we assumed that we do not know $y(t)$ at times $t > 0$ and hence we cannot have quadrature points $x_k > 0$.

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

For example, we can use an Euler step to estimate

$$\tilde{y}_2(t) \;=\; y(0) + f\big(y(0)\big)\, t,$$

and then we can use a trapezoidal rule approximation

$$\tilde{y}(t) = y(0) + \Big(f\big(y(0)\big) + f\big(\tilde{y}_2(t)\big)\Big)\, \tfrac{t}{2},$$

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

Runge-Kutta Methods

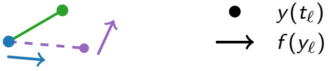
Def: Trapezoidal rule

Approximating the solution to $y(0) = y_0$, $\dot{y} = f(y)$ using

$$\begin{aligned}
\tilde{y}(0) &= y_0 \\
\tilde{y}_2(t_k) &= \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,\big(t_k - t_{k-1}\big), \\
\tilde{y}(t_k) &= \tilde{y}(t_{k-1}) + \Big(f\big(\tilde{y}(t_{k-1})\big) + f\big(\tilde{y}_2(t_k)\big)\Big)\,\tfrac{t_k - t_{k-1}}{2},
\end{aligned}$$

is known as the trapezoidal rule method.

The pictorial interpretation of a trapezoidal step is as follows.



Runge-Kutta Methods

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 the "trapezoidal rule".

This is not a problem since we can usually deduce 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 $\tilde{y}_2(t_k)$ and $\tilde{y}(t_k)$ to the same 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

Runtime and convergence of the trapezoidal rule

We now have two algorithms for solving the same problem, namely Euler's method and the trapezoidal rule.

To compare these methods, I will next repeat for the trapezoidal rule what we have already done for Euler's method, namely I will study the scaling of the runtime and accuracy as a function of the number of time steps n.

Thm: Runtime of the trapezoidal rule

n steps of the trapezoidal rule require $2n$ evaluations of $f(y)$ and $O(n)$ other operations.

Proof. Immediate consequence of the recursion equations

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

Runge-Kutta Methods

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,

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

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

Proof. See the following slides.

Runge-Kutta Methods

Proof of convergence of the trapezoidal rule.

We have seen on slides 31 and following that the first-order convergence of Euler's method is a consequence of the error bound

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

and the second-order consistency of the Euler time propagator.

Looking back at the proofs of these results, we realise that both the above bound and the claim

$$(p+1)\text{th order consistency} \quad \Longrightarrow \quad p\text{-order convergence}$$

are in fact not specific to Euler's method but rather hold for any numerical time propagator $\tilde{\Phi}_k(y_0, t)$.

All that is needed to show second-order convergence of the trapezoidal rule is hence to introduce the numerical time propagator associated with the trapezoidal rule and show that this propagator is third-order consistent.

This will be tackled on the following slides.

Runge-Kutta Methods

Def: Trapezoidal rule time propagator

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

$$\tilde{\Phi}(y_0, t) = y_0 + \big(f(y_0) + f(y_2)\big)\,\tfrac{t}{2} \qquad \text{where} \qquad y_2 = y_0 + f(y_0)\,t,$$

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

Lemma: Consistency of trapezoidal rule

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

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

Proof. As on slide 37, 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

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

and conclude that we indeed have

$$\tilde{y}(0) = y_0 = y(0), \quad \dot{\tilde{y}}(0) = f(y_0) = \dot{y}(0), \quad \ddot{\tilde{y}}(0) = f'(y_0)\,f(y_0) = \ddot{y}(0),$$

where for the second-order term I used that

$$\dot{y}(t) = f(y(t)) \quad \Longrightarrow \quad \ddot{y}(t) = f'\big(y(t)\big)\,\dot{y}(t) = f'\big(y(t)\big)\,f\big(y(t)\big).$$

Runge-Kutta Methods

Euler vs trapezoidal method

We can summarise the performance characteristics of Euler's method and the trapezoidal rule as follows.

	$f(y)$ evals	Error
Euler	n	$O(n^{-1})$
Trapezoidal	$2n$	$O(n^{-2})$

In practice, this means that Euler's method tends to be faster for moderate accuracy requirements but is eventually outperformed by the trapezoidal rule as the error tolerance decreases.

See `convergence()` (uncomment the `"trapezoidal"` line).

Runge-Kutta Methods

General Runge-Kutta methods

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 procedure which led us to the trapezoidal rule.

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

Runge-Kutta Methods

Def: Runge-Kutta method

An algorithm which approximates the solution to $y(0) = y_0$, $\dot{y} = f(y)$ using

```
1: $\tilde{y}(0) = y_0$,
2: for $k = 1, \dots, n$ do
3:     for $i = 1, \dots, s$ do
4:         $\tilde{y}_i = \tilde{y}(t_{k-1}) + \sum_{j=1}^{s} f(\tilde{y}_j) W_{ij} (t_k - t_{k-1})$
5:     end for
6:     $\tilde{y}(t_k) = \tilde{y}(t_{k-1}) + \sum_{j=1}^{s} f(\tilde{y}_j) w_j (t_k - t_{k-1})$
7: end for
```

is called an *s-stage Runge-Kutta method*.

The parameters $w \in \mathbb{R}^s$ and $W \in \mathbb{R}^{s \times s}$ can be interpreted as the quadrature weights of the following quadrature rules:

- $(x_j, w_j)_{j=1}^{s}$ is a quadrature rule for $[0, 1]$.
- $(x_i, W_{ij})_{i=1}^{s}$ is a quadrature rule for $[0, x_j]$.

Runge-Kutta Methods

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

Def: Butcher tableau

The representation of the Runge-Kutta parameters $x \in \mathbb{R}^s$, $W \in \mathbb{R}^{s \times s}$ and $w \in \mathbb{R}^s$ given by

$$\begin{array}{c|ccc} x_1 & W_{11} & \dots & W_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ x_s & W_{s1} & \dots & W_{ss} \\ \hline & w_1 & \dots & w_s \end{array}$$

is called a *Butcher tableau*.

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

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

Example 2: Trapezoidal rule

Tableau

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

Interpretation

$$\begin{array}{r} \tilde{y}_1(t_{k-1}) = \tilde{y}(t_{k-1}) \\ \tilde{y}_2(t_k) = \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,(t_k - t_{k-1}) \\ \hline \tilde{y}(t_k) = \tilde{y}(t_{k-1}) + f(\tilde{y}_1)\,\frac{t_k - t_{k-1}}{2} + f(\tilde{y}_2)\,\frac{t_k - t_{k-1}}{2} \end{array}$$

Runge-Kutta Methods

Optimal Runge-Kutta methods

The above definition allows us to turn any set of parameters x, 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 same number of $f(y)$ evaluations per time step but better accuracy.

Combining this insight with the observation that the number of $f(y)$ evaluation per step is equal to the number of stages s, we conclude that a particular Runge-Kutta method is interesting only 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 35.
- The order of convergence is thus maximised if we choose x, 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 can be 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 time-stepping

The definition of Runge-Kutta methods on slide 52 depends on the physical / mathematical parameters $f(y)$ and y_0, but in addition it also depends on two types of numerical parameters, namely

- the quadrature weights $W \in \mathbb{R}^{s \times s}$ and $w \in \mathbb{R}^s$, and
- the temporal mesh $(t_k)_{k=0}^n$.

The above slides discussed the various criteria affecting our choice of quadrature weights. Next, I would like to turn our attention towards the temporal mesh $(t_k)_{k=0}^n$, and specifically I would like to study how the accuracy depends on the mesh points t_k for a fixed mesh size n.

This study complements our earlier convergence results where we fixed the mesh $t_k = \frac{T}{n}\,k$ to be equispaced and instead studied the limit of increasingly larger mesh sizes n.

Runge-Kutta Methods

Adaptive time-stepping (continued)

The motivation for studying accuracy as a function of the mesh points t_k is that the runtime of Runge-Kutta methods depends on only the number of time steps n but not on their spacing, and thus varying the mesh points t_k without increasing their number n gives us a means to improve accuracy without increasing the computational workload.

As soon as we understand how the temporal mesh affects the accuracy, we will therefore also be looking into how we can optimise the mesh such that the error is as small as possible for the given number of mesh points.

The algorithm which will result from this analysis is known as *adaptive time-stepping*.

Runge-Kutta Methods

Adaptive time-stepping (continued)

It turns out that all we have to do to understand the impact of the temporal mesh on accuracy is to repeat the earlier convergence analysis but fill in a bit more detail. Specifically:

1. On slide 35, we have seen that the error of a Runge-Kutta scheme $\tilde{y}(t)$ with associated time propagator $\tilde{\Phi}(y_0, t)$ is bounded by

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

2. If we had used the Taylor theorem with remainder in our consistency analyses on slides 37 and 49, then we would have found that the local errors are bounded by

$$\left\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\right\| \;\le\; \tfrac{1}{p!}\,\bigl(t_k - t_{k-1}\bigr)^p \max_{t \in [t_{k-1}, t_k]} \left\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\right\|,$$

   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 time-stepping (continued)

The above results readily answer the question of how the step lengths impact accuracy, but they are a bit inconvenient to work with. Let us therefore make the following simplifying assumptions:

- $\exp\big(L\,(t_n - t_k)\big) \approx 1.$

  Justification: We have seen earlier that solving ODEs over time spans $T \gg \frac{1}{L}$ is generally not advisable

- $\max_{t \in [t_{k-1}, t_k]} \big\|\tilde{y}^{(p)}(t) - y^{(p)}(t)\big\| \;\approx\; \big\|\tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1})\big\|.$

  Justification: The step sizes $t_k - t_{k-1}$ are typically small enough such that the objective function is essentially constant over $[t_{k-1}, t_k]$.

We then obtain the approximate error bound

$$\big\|\tilde{y}(t_n) - y(t_n)\big\| \;\lesssim\; \sum_{k=1}^{n} \tfrac{1}{p!}\,\big(t_k - t_{k-1}\big)^p\,\big\|\tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1})\big\|.$$

This answers the question of how the temporal mesh impacts the error.

Runge-Kutta Methods

Adaptive time-stepping (continued)

Next, let us consider how we can optimise the mesh such that the error is as small as possible, i.e. let us consider the constrained optimisation problem

$$\min_{(\Delta t_k)_{k=1}^n} \sum_{k=1}^{n} c_k \, \Delta t_k^p \quad \text{subject to} \quad \sum_{k=1}^{n} \Delta t_k - T = 0$$

where I introduced the abbreviations

$$\Delta t_k = t_k - t_{k-1} \qquad \text{and} \qquad c_k = \tfrac{1}{p!} \left\| \tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1}) \right\|.$$

According to the Lagrangian multiplier trick, the solution to this problem is such that the derivatives of the objective function

$$F\big((\Delta t_k)_k, \lambda\big) = \sum_{k=1}^{n} c_k \, \Delta t_k^p \; + \; \lambda \left(\sum_{k=1}^{n} \Delta t_k - T \right)$$

with respect to all Δt_k and λ are zero.

Runge-Kutta Methods

Adaptive time-stepping (continued)

This in particular implies that

$$\frac{\partial F}{\partial \Delta t_k} \;=\; p\, c_k\, \Delta t_k^{p-1} + \lambda \;=\; 0;$$

hence the mesh $(t_k)_k$ is optimal if

$$\Delta t_k^{p-1} \left\| \tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1}) \right\| \;=\; \tau,$$

where $\tau \;=\; -(p-1)!\,\lambda \;\in\; [0,\infty)$ represents some local error tolerance independent of k.

This condition may look cryptic at first, but actually its meaning is both simple and intuitive, see next slide.

Runge-Kutta Methods

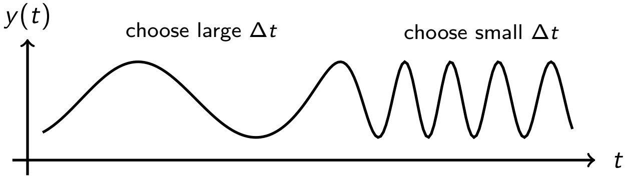
Adaptive time-stepping (continued)

It turns out that for many Runge-Kutta methods, we have

$$\left\|\tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1})\right\| \approx \left\|y^{(p)}(t_{k-1})\right\|.$$

The right-hand side in this approximate identity can be interpreted as a measure for the curvature in $y(t)$, and hence the above finding can be summarised as follows.

The optimal step size Δt_k is small if the curvature is large, and large if the curvature is small.



Runge-Kutta Methods

Adaptive time-stepping (continued)

The condition

$$\Delta t_k^{p-1} \left\| \tilde{y}^{(p)}(t_{k-1}) - y^{(p)}(t_{k-1}) \right\| = \tau$$

tells us on a theoretical level what a good mesh should like, but it does not quite tell us yet how to determine such a mesh in practice because we are missing a tool to estimate the purple factor.

This missing tool is provided by the result on the next slide.

Runge-Kutta Methods

Thm: Estimating curvature by comparing methods

Let $\tilde{\Phi}(y_0, t)$, $\tilde{\Phi}_{\text{ref}}(y_0, t)$ be two Runge-Kutta time propagators with orders of consistency p and $p_{\text{ref}} > p$, respectively. We then have

$$\left\|\tilde{\Phi}(y_0, t) - \tilde{\Phi}_{\text{ref}}(y_0, t)\right\| = \left\|\tilde{y}^{(p)}(0) - y^{(p)}(0)\right\| \tfrac{t^p}{p!} + O(t^{p+1}),$$

i.e. we can estimate the curvature by comparing in each time step two approximate solutions $\tilde{y}(t)$ and $\tilde{y}_{\text{ref}}(t)$ computed using two Runge-Kutta steps of different orders of consistency $p < p_{\text{ref}}$.

Proof. According to Taylor's theorem, we have

$$\begin{aligned}
\tilde{\Phi}(y_0, t) &= y(0) + \ldots + y^{(p-1)}(0)\,\tfrac{t^{p-1}}{(p-1)!} + \tilde{y}^{(p)}(0)\,\tfrac{t^p}{p!} + O\big(t^{p+1}\big),\\
\tilde{\Phi}_{\text{ref}}(y_0, t) &= y(0) + \ldots + y^{(p-1)}(0)\,\tfrac{t^{p-1}}{(p-1)!} + y^{(p)}(0)\,\tfrac{t^p}{p!} + O\big(t^{p+1}\big),\\
\Phi(y_0, t) &= y(0) + \ldots + y^{(p-1)}(0)\,\tfrac{t^{p-1}}{(p-1)!} + y^{(p)}(0)\,\tfrac{t^p}{p!} + O\big(t^{p+1}\big).
\end{aligned}$$

Therefore,

$$\tilde{\Phi}(y_0, t) - \tilde{\Phi}_{\text{ref}}(y_0, t) \;=\; \Big(\tilde{y}^{(p)}(0) - y^{(p)}(0)\Big)\,\tfrac{t^p}{p!} + O\big(t^{p+1}\big).$$

Runge-Kutta Methods

Adaptive Runge-Kutta methods (continued)

We now have all the mathematical tools needed to determine good temporal meshes. What is needed next is some clever mathematical engineering to translate these tools into a simple and fast adaptive time-stepping algorithm.

The next slide will present a big-picture outline of this algorithm, and the slides after that will fill in a few details.

Runge-Kutta Methods

The adaptive time-stepping algorithm

Set $t_0 = 0$, $\tilde{y}(t_0) = y_0$, and until we have $t_k = T$, repeat the following:

1. Compute the two Runge-Kutta estimates $\tilde{y}$ and $\tilde{y}_{\text{ref}}$ for the solution at the next time point t_k using a trial step size Δt.
2. Estimate what the optimal step size would have been by solving for Δt_{opt} the two equations

   $$\|\tilde{y} - \tilde{y}_{\text{ref}}\| = \|\tilde{y}^{(p)} - y^{(p)}\| \, \frac{\Delta t^p}{p!} \quad \text{and} \quad \|\tilde{y}^{(p)} - y^{(p)}\| \, \frac{\Delta t_{\text{opt}}^{p-1}}{p!} = \tau,$$

   i.e. by computing

   $$\Delta t_{\text{opt}} = \left(\frac{\tau \, \Delta t^p}{\|\tilde{y} - \tilde{y}_{\text{ref}}\|} \right)^{1/(p-1)} .$$

3. If the trial step size is less than the optimal step size, $\Delta t < \Delta t_{\text{opt}}$, then we set $t_k = t_{k-1} + \Delta t$ and $\tilde{y}(t_k) = \tilde{y}_{\text{ref}}$ and move on to the next step $t_k \to t_{k+1}$.

   Otherwise, we update the trial step size to $\Delta t = 0.9 \, \Delta t_{\text{opt}}$ and restart from step 1.

   The 0.9 in this formula is a safety factor intended to increase the chances that we accept the new trial step size.

Runge-Kutta Methods

As promised, I will next look at a few details of the above algorithm.

Embedded Runge-Kutta schemes

At first sight, it may seem that computing both $\tilde{y}$ and $\tilde{y}_{\text{ref}}$ in step 1 of the above algorithm would take roughly twice as long as computing just one of these trial solutions.

This factor of 2 can be avoided by choosing the two Runge-Kutta schemes such that the lower-order method uses only $f(y)$ values which must be computed anyway when evaluating the higher-order method.

Such pairs of Runge-Kutta schemes are called *embedded*.

Example. The trapezoidal step

$$\begin{aligned}\tilde{y}_{\text{Euler}}(t_k) &= \tilde{y}(t_{k-1}) + f\big(\tilde{y}(t_{k-1})\big)\,\big(t_k - t_{k-1}\big),\\ \tilde{y}_{\text{trap}}(t_k) &= \tilde{y}(t_{k-1}) + \Big(f\big(\tilde{y}(t_{k-1})\big) + f\big(\tilde{y}_{\text{Euler}}(t_k)\big)\Big)\,\tfrac{t_k - t_{k-1}}{2}\end{aligned}$$

already evaluates the Euler step formula; thus the runtime of using the trapezoidal rule to estimate the accuracy of an Euler step is exactly equal to that of just evaluating a trapezoidal step. See `embedded_ET_step()`.

Runge-Kutta Methods

Embedded Runge-Kutta schemes (continued)

Further examples of embedded Runge-Kutta schemes can be found under

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

Implementation of adaptive time-stepping

See `propagate_adaptive()` and `step_example()`.

Note that this code determines a good temporal mesh for the less accurate Runge-Kutta scheme $\tilde{\Phi}$ but then returns the approximations computed according to the more accurate scheme $\tilde{\Phi}_{\mathrm{ref}}$ as “the” ODE solution (see the `push!(y,yref)` statement).

The rationale for this abuse of the higher-order scheme is that it seems silly to compute a more accurate result and then not use it.

Runge-Kutta Methods

Adaptive time-stepping applied to $\dot{y} = \lambda y$

Encouraged by our observations in `step_example()`, I next apply adaptive time-stepping to the ODE

$$y(0) = 1, \quad \dot{y} = \lambda y \quad \Longleftrightarrow \quad y(t) = \exp(\lambda\, t) \quad \text{where} \quad \lambda < 0$$

in `decay_example()`.

Since the solution to this ODE converges to a constant, one might expect that our adaptive time-stepping scheme would choose increasingly larger step sizes as $t \to \infty$, but we empirically observe that this is not the case.

Choosing small step sizes even in regions where the solution $y(t)$ barely changes feels like a waste of computational resources. Let us therefore investigate why our step selection routine refuses to increase the step size beyond some upper bound Δt_{max}.

Runge-Kutta Methods

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

The key to answering the above question is to observe that both Euler's method and the trapezoidal rule applied to $\dot{y} = \lambda y$ take a very simple form: the numerical time propagator for Euler's method is given by

$$\tilde{\Phi}(y_0, t) \;=\; y_0 + f(y_0)\,t \;=\; y_0 + \lambda y_0 t \;=\; (1 + \lambda t)\,y_0, \qquad \text{(Euler)}$$

and the numerical time propagator for the trapezoidal rule is given by

$$\begin{aligned} \tilde{\Phi}(y_0, t) &= y_0 + \Big(f(y_0) + f\big(y_0 + f(y_0)\,t\big)\Big)\,\tfrac{t}{2} \\ &= y_0 + \Big(\lambda y_0 - \big(y_0 + \lambda y_0 t\big)\Big)\,\tfrac{t}{2} \qquad \text{(trapezoidal)} \\ &= \big(1 + \lambda t + \tfrac{\lambda^2 t^2}{2}\big)\,y_0. \end{aligned}$$

The numerical solutions $\tilde{y}(t)$ after k steps with constant step size Δt are hence given by

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

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 $\dot{y} = \lambda y$ (continued)

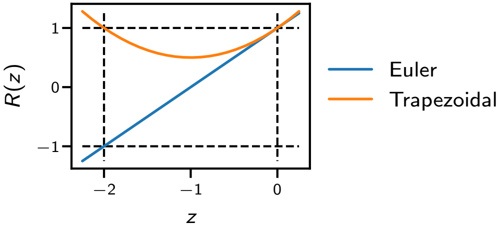
The Runge-Kutta solutions $\tilde{y}(t)$ therefore

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

In the case

$$y(0) = 1, \quad \dot{y} = \lambda y \quad \iff \quad y(t) = \exp(\lambda\,t) \quad \text{where} \quad \lambda < 0,$$

we know that the exact solution decays exponentially and hence we want our numerical solutions to do the same, i.e. we want $|R(\lambda\Delta t)| < 1$. Looking at the below plots of $R(z)$, we conclude that this is the case if and only if $0 < \lambda\,\Delta t < 2$.



Runge-Kutta Methods

We may summarise these findings as follows.

Thm: Step size constraint for $\dot{y} = \lambda y$

Euler's method and the trapezoidal rule applied to $\dot{y} = \lambda y$ with an equispaced temporal mesh $(t_k = k\,\Delta t)_{k=1}^{\infty}$ produce numerical solutions $\tilde{y}(t)$ such that $\lim_{t\to\infty} \tilde{y}(t) = 0$ if and only if $\Delta t < 2$.

Terminology: Step size constraint and largest admissible step size

Bounds of the form $\Delta t < \Delta t_{\mathrm{max}}$ are called *step size constraints*, and the upper bound Δt_{max} is called the *largest admissible step size*.

Numerical demonstration

`stepsize()` shows that our adaptive time-stepping routine was able to detect the step size constraint and choose Δt accordingly.

Furthermore, `stability_example()` shows that $\tilde{y}(t)$ indeed converges to 0 only if $\Delta t < 2$.

Runge-Kutta Methods

Why step size constraints for $\dot{y} = \lambda y$ matter

The above findings are specific to the simple ODE $\dot{y} = \lambda y$ whose solution can be determined analytically. At first sight, it may therefore seem that these findings have little practical value, but actually the opposite is true. The reason for this is the following.

The issues described above at the example of $\dot{y} = \lambda y$ arise whenever we apply Runge-Kutta methods to ODEs with attractive fixed points.

The term "fixed point" in this statement refers to the following definition.

Def: Fixed point / steady state

$y_F \in \mathbb{R}^n$ is called a *fixed point* or a *steady state* of $\dot{y} = f(y)$ if $f(y_F) = 0$.

Example. $y_F = 0$ is a fixed point of the ODE $\dot{y} = \lambda y$.

In the case $\dot{y} = \lambda y$, the "attractive" part refers to the fact that $\lambda < 0$ which implies that $y(t)$ converges to the fixed point $y_F = 0$. A more general definition of "attractive" will follow in due time.

A mathematical justification for the above claim is provided by the result on the next slide.

Runge-Kutta Methods

Thm: Linearisation of ODEs

Assume $y(t)$ solves the ODE $\dot{y} = f(y)$ which has a fixed point $y_F \in \mathbb{R}^n$, and assume the Jacobian $\nabla f(y_F)$ has the eigenvalue decomposition

$$\nabla f(y_F) = V \Lambda V^{-1}.$$

We then have

$$w(t) = V^{-1}\big(y(t) - y_F\big) \qquad \Longrightarrow \qquad \dot{w}_k = \lambda_k w_k + O\big(\|w\|^2\big).$$

Proof. We have

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

Multiplying from the left with V^{-1}, we thus have

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

Runge-Kutta Methods

The above result is known under a special name.

Terminology: Linearisation

$\dot{w} = \Lambda w$ is called the *linearisation* of $\dot{y} = f(y)$ because it is derived from the linear approximation

$$f(y) \;\approx\; f(y_F) + \nabla f(y_F)\,(y - y_F).$$

[To be continued]