MA3227 Numerical Analysis II

Lecture 16: Explicit Runge-Kutta Methods

Simon Etter

2019/2020

Explicit Runge-Kutta Methods

Introduction

The last lecture introduced the basic theory for ODEs: if $f(y)$ is Lipschitz-continuous, then $\dot{y} = f(y)$ has a unique solution and the solution is a Lipschitz-continuous function of the initial conditions. Our aim in this lecture is to compute these solutions numerically.

We have already noted previously that

$$y(0) = y_0, \qquad \dot{y}(t) = f(y(t))$$

is equivalent to

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

All ODE solvers in this module will be derived by applying quadrature to the above integral. This lecture will therefore first review some definitions and results regarding quadrature and then explain how to apply them to ODEs.

Explicit Runge-Kutta Methods

Terminology

Quadrature refers to computing an integral numerically.
Solving an ODE is sometimes referred to as *numerical integration*.

These two terms are easily confused, so try your best to remember this.

Quadrature rule

A quadrature rule is a collection of quadrature points $x_k \in [a, b]$, and quadrature weights $w_k \in \mathbb{R}$ such that

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

Explicit Runge-Kutta Methods

Mapping of quadrature rules

A quadrature rule (x_k, w_k) for an interval $[a, b]$ can be mapped to a quadrature rule $(\hat{x}_k, \hat{w}_k)$ for another interval $[\hat{a}, \hat{b}]$ using the formula

$$\hat{x}_k = \phi(x_k), \qquad \hat{w}_k = \tfrac{\hat{b}-\hat{a}}{b-a}\, w_k,$$

where

$$\phi(x) = \hat{a}\,\tfrac{b-x}{b-a} + \hat{b}\,\tfrac{x-a}{b-a}, \qquad \phi'(x) = \tfrac{\hat{b}-\hat{a}}{b-a}.$$

This follows immediately from the integration by substitution formula

$$\int_{\hat{a}}^{\hat{b}} f(\hat{x})\, d\hat{x} = \int_a^b f(\phi(x))\, \phi'(x)\, dx.$$

Explicit Runge-Kutta Methods

Composite quadrature

Instead of applying a quadrature rule (x_k, w_k) to the whole integral of interest, we can split that integral into smaller parts and apply the quadrature rule mapped according to the result on the previous slide to each partial integral. This is called composite quadrature.

For example, if (x_k, w_k) is a quadrature rule for $[0, 1]$, we can write

$$\int_0^1 f(x)\,dx = \sum_{k=1}^{n} \int_{(k-1)/n}^{k/n} f(x)\,dx = \sum_{k=1}^{n} \int_0^1 f\left(\tfrac{k-1+x}{n}\right) \tfrac{1}{n}\,dx$$
$$\approx \sum_{k=1}^{n} \sum_{\ell=1}^{p} f\left(\tfrac{k-1+x_\ell}{n}\right) \tfrac{w_\ell}{n}.$$

Explicit Runge-Kutta Methods

Error estimates for composite quadrature

Error estimates for composite quadrature rules typically take the form

$$\left| \sum_{k=1}^{n} \sum_{\ell=1}^{p} f\left(\phi_k(x_\ell)\right) \phi'(x_k)\, w_\ell - \int_a^b f(x)\, dx \right| = \mathcal{O}\left(n^{-d}\right)$$

for some $d \in [p, 2p]$, assuming $f(x)$ has d derivatives.

d is just a parameter of the quadrature rule which you can look up in textbooks / on Wikipedia along with the quadrature points x_k and weights w_k.

Explicit Runge-Kutta Methods

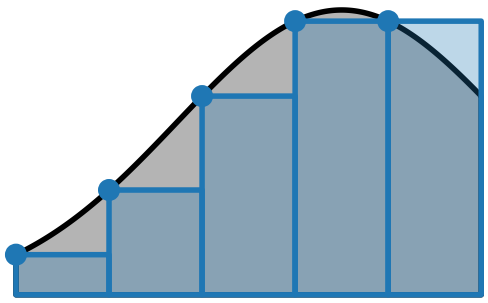
Important quadrature rules

Left-point: $$\int_0^1 f(x)\,dx = \sum_{k=1}^{n} f\left(\tfrac{k-1}{n}\right)\tfrac{1}{n} + \mathcal{O}\left(n^{-1}\right)$$

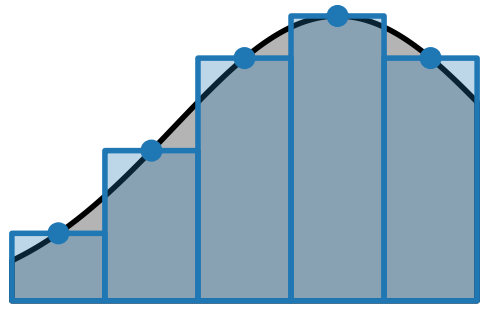
Midpoint: $$\int_0^1 f(x)\,dx = \sum_{k=1}^{n} f\left(\tfrac{k-1/2}{n}\right)\tfrac{1}{n} + \mathcal{O}\left(n^{-2}\right)$$

Trapezoidal: $$\int_0^1 f(x)\,dx = \left(f(0) + 2\sum_{k=1}^{n-1} f(\tfrac{k}{n}) + f(1)\right)\tfrac{1}{2n} + \mathcal{O}\left(n^{-2}\right)$$

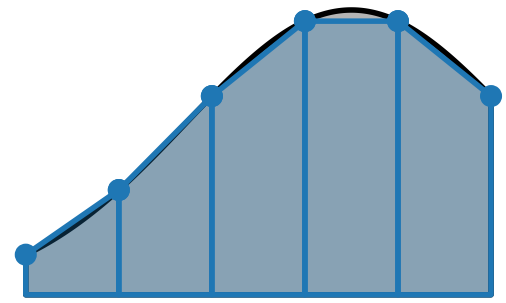
Left-point



Midpoint



Trapezoidal



Explicit Runge-Kutta Methods

Quadrature for ODEs

Our aim is to solve ODEs by applying quadrature to

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

Given a quadrature formula (θ_k, w_k) for $[0, 1]$, we can easily write down

$$\tilde{y}(t) = y_0 + \sum_{k=1}^{p} f(y(\theta_k\, t))\, t\, w_k,$$

but this formula is not practical because we do not know $y(t)$ for $t > 0$.

An easy solution is to simply avoid evaluating $y(t)$ for $t > 0$ by using the left-point rule:

$$\tilde{y}(t) = y_0 + f\left(y_0\right) t.$$

This formula is known as Euler's method. We will see that it works but does not give good accuracy since it is based on a very low-order quadrature rule.

Explicit Runge-Kutta Methods

Quadrature for ODEs (continued)

Better accuracy can be achieved by using the left-point rule to estimate $y(\frac{t}{2})$ and then using the midpoint rule:

$$\tilde{y}(t) = y_0 + f\left(\tilde{y}(\tfrac{t}{2})\right) t, \qquad \tilde{y}(\tfrac{t}{2}) = y_0 + f\left(y_0\right) \tfrac{t}{2}.$$

The same idea can also be used for the trapezoidal rule and higher-order quadrature rules.

For large t, all of the above methods will deliver poor accuracy since they apply a single quadrature rule to the whole interval. The obvious remedy is to use composite quadrature, which in the context of ODEs works as follows.

- Fix a small interval $[0, t_1]$ and compute $\tilde{y}(t_1)$ using any of the above formulae.
- Fix another small interval $[t_1, t_2]$ and compute $\tilde{y}(t_2)$ using any of the above formulae and initial value $\tilde{y}(t_1)$.
- Repeat until we hit the desired final time T.

ODE solvers of this form are called *single-step* or *Runge-Kutta* methods. See `16_explicit_runge_kutta_methods.jl` for algorithmic details.

Explicit Runge-Kutta Methods

Discussion

As usual, once we have a numerical method which appears to work, we would like to study how the error changes as a function of the parameters of the method.

For Runge-Kutta methods, these parameters are:

- The local stepping function, e.g. Euler vs. midpoint.
- The local step length. In the following, I will assume that we have a temporal mesh

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

  where on each interval $[t_{k-1}, t_k]$ we propagate the solution using the local stepping function.

Our next aim will therefore be to estimate the error $\|\tilde{y}(T) - y(T)\|$ as a function of the local stepping function and the temporal mesh $(t_k)_{k=0}^n$.

The following slide introduces the terminology and notation required to formulate and prove such a result.

Explicit Runge-Kutta Methods

Time propagators

An ODE $\dot{y} = f(y)$ implicitly defines a function $\Phi : (y_0, T) \mapsto y(T)$.
Similarly, Runge-Kutta methods define functions

$$\tilde{\Phi} : \big(y_0, (t_k)_{k=0}^n\big) \mapsto \tilde{y}(T).$$

These functions are called exact and numerical time propagators, respectively.

The above notation for the time propagators is quite cumbersome. Let us simplify the notation by introducing the abbreviations

$$\Phi_k(y) = \Phi(y, t_k - t_{k-1}), \qquad y_k = \Phi(y_0, t_k)$$
$$\tilde{\Phi}_k(y) = \tilde{\Phi}\big(y, (t_{k-1}, t_k)\big), \qquad \tilde{y}_k = \tilde{\Phi}\big(y_0, (t_{k'})_{k'=0}^k\big).$$

We then have the recurrence formulae

$$y_k = \Phi_k(y_{k-1}), \qquad \tilde{y}_k = \tilde{\Phi}_k\big(\tilde{y}_{k-1}\big),$$

and the stability estimate from Lecture 15 implies

$$\|\Phi_k(y_1) - \Phi_k(y_2)\| \leq \exp\big(L\,(t_k - t_{k-1})\big)\,\|y_1 - y_2\|.$$

Explicit Runge-Kutta Methods

Lemma: Error bound for Runge-Kutta methods

Assume $f(y)$ has Lipschitz constant L. Then,

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

Proof.

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

Explicit Runge-Kutta Methods

Discussion

The factors $\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\|$ appearing in the above bound are called local, consistency or truncation error. They are the errors which are newly introduced in step k, in addition to the errors which are carried over from steps $k' < k$.

We will next use the above error bound to show that Runge-Kutta methods converge to the exact solution when applied to a sequence of temporal meshes of increasing fineness, i.e. such that $\max_k |t_k - t_{k-1}| \to 0$. To keep the discussion simple, we will do so assuming a uniformly spaced mesh $t_k = T\,\frac{k}{n}$.

The above bound shows that we get convergence if

$$t_k - t_{k-1} \to 0 \quad \Longrightarrow \quad \|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\| \to 0.$$

This motivates the following definition.

Def: Consistency of numerical time-propagator

We say that $\tilde{\Phi}\big(y_0, (0,t)\big)$ is $(p+1)$th-order consistent with $\Phi(y_0, t)$ if

$$\big\|\tilde{\Phi}\big(y_0, (0,t)\big) - \Phi_t(y_0, t)\big\| = \mathcal{O}\big(t^{p+1}\big).$$

Explicit Runge-Kutta Methods

Thm: Convergence of Runge-Kutta methods

Assume $\tilde{y}(T)$ is obtained by applying a $(p+1)$th-order consistent Runge-Kutta method on an equispaced temporal mesh $t_k = T\,\frac{k}{n}$ to the ODE $\dot{y} = f(y)$ where $f(y)$ has Lipschitz constant L. Then,

$$\|\tilde{y}(T) - y(T)\| = \mathcal{O}\big(\exp(LT)\,\tfrac{T^{p+1}}{n^p}\big).$$

This is called pth-order convergence.

Proof.

$$\begin{aligned}
\big\|\tilde{y}_n - y_n\big\| &\leq \sum_{k'=1}^{n} \exp\big(L\,(t_n - t_{k'})\big)\,\|\tilde{\Phi}_{k'}(\tilde{y}_{k'-1}) - \Phi_{k'}(\tilde{y}_{k'-1})\| \\
&\leq n\,\exp\big(L\,(t_n - t_0)\big)\,\mathcal{O}\Big(\big(\tfrac{T}{n}\big)^{p+1}\Big) \\
&= \exp\big(L\,(t_n - t_0)\big)\,\mathcal{O}\big(\tfrac{T^{p+1}}{n^p}\big)
\end{aligned}$$

Explicit Runge-Kutta Methods

Remark 1

Note that $(p+1)$th-order consistency is required for pth-order convergence. The heuristic reason for this is that a $(p+1)$-order consistent Runge-Kutta method makes n errors of magnitude $\mathcal{O}\left(n^{-p-1}\right)$; hence the overall error is $\mathcal{O}\left(n\, n^{-p-1}\right) = \mathcal{O}\left(n^{-p}\right)$.

Remark 2

The above convergence estimate once again indicates that it may be difficult to solve ODEs on time-scales larger than one over the Lipschitz constant; see `error_bound()`.

However, it is also worth pointing out that the error estimate is a worst-case bound, which is achieved in `error_bound()` because the function itself blows up exponentially. The situation is more benign if the solution converges to a fixed point or is oscillatory. Set λ `= -1.0` or λ `= 1.0im`, respectively, for numerical demonstration.

Discussion

Thanks to the above theorem, all that remains to do to determine the order of convergence of a concrete Runge-Kutta method is to determine its order of consistency. The following slide demonstrates how to do this.

Explicit Runge-Kutta Methods

Consistency of Euler and midpoint methods

Assuming y (or equivalently f) has sufficiently many derivatives, the consistency error $\tilde{\Phi}\big(y_0,(0,t)\big) - \Phi(y_0,t) = \tilde{y}(t) - y(t)$ can be estimated using Taylor series as follows.

- Euler's method:

$$\tilde{y}(t) = y(0) + f\big(y(0)\big)\, t$$
$$y(t) = y(0) + \dot{y}(0)\, t + \mathcal{O}(t^2)$$

  Since $\dot{y}(0) = f\big(y(0)\big)$, we have $\tilde{y}(t) - y(t) = \mathcal{O}(t^2)$.

- Midpoint method:

$$\begin{aligned}\tilde{y}(t) &= y(0) + f\big(y(0) + f(y(0))\tfrac{t}{2}\big)\, t \\ &= y(0) + f\big(y(0)\big)\, t + f'\big(y(0)\big)\, f\big(y(0)\big)\, \tfrac{t^2}{2} + \mathcal{O}(t^3) \\ y(t) &= y(0) + \dot{y}(0)\, t + \ddot{y}(0)\, \tfrac{t^2}{2} + \mathcal{O}(t^3)\end{aligned}$$

  Since $\dot{y}(0) = f\big(y(0)\big)$ and $\ddot{y}(0) = f'\big(y(0)\big)\, \dot{y}(0) = f'\big(y(0)\big)\, f\big(y(0)\big)$, we have $\tilde{y}(t) - y(t) = \mathcal{O}(t^3)$.

Explicit Runge-Kutta Methods

Convergence of Euler and midpoint methods

Putting all of the above together, we conlude that the error in Euler's method is $\mathcal{O}\left(n^{-1}\right)$ and the error in the midpoint method is $\mathcal{O}\left(n^{-2}\right)$, i.e. Euler is first-order convergent and midpoint is second-order convergent. See `convergence()`.

Discussion

Recall: the midpoint method achieves a higher order of convergence by using a high-order quadrature rule (the midpoint rule) to compute $\tilde{y}(t)$ and a low-order quadrature rule (the left-point rule) to compute the $y(t\tau_k)$ required by the high-order quadrature rule.

This procedure can be iterated even further to obtain even higher convergence orders; see next slide.

Explicit Runge-Kutta Methods

s-stage Runge-Kutta methods

Assume we have quadrature points θ_i and a sequence of quadrature weights w_{ij} with $i \in \{0, \ldots, s\}$ and $j \in \{0, \ldots, i-1\}$ such that

$$\theta_0 = 0, \qquad \theta_s = 1,$$

and

$$\int_0^{\theta_i} f(\tau)\, d\tau \approx \sum_{j=0}^{i-1} w_{ij}\, f(\theta_j) \qquad \text{for all } i \in \{0, \ldots, s\}.$$

Then, we can compute an approximate solution to $\dot{y} = f(y)$ through

$$\tilde{y}(t) = y(0) + t \sum_{j=0}^{s} w_{sj}\, f_j \approx y(t)$$

where

$$f_i = f\left(y(0) + t \sum_{j=0}^{i-1} w_{ij} f_j\right) \approx f\big(y(\theta_i\, t)\big)$$

Algorithms of this form are called s-stage Runge-Kutta methods.

Explicit Runge-Kutta Methods

Butcher's tableau

Runge-Kutta methods with $s > 2$ involve many parameters. A convenient way to display these parameters is in a table as follows.

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\theta_1 & w_{10} & & & & & \\
\theta_2 & w_{20} & w_{21} & & & & \\
\vdots & \vdots & \vdots & \ddots & & & \\
\theta_{s-1} & w_{s-1,0} & w_{s-1,1} & \cdots & w_{s-1,s-2} & & \\
\hline
 & w_{s,0} & w_{s,1} & \cdots & w_{s,s-2} & w_{s,s-1}
\end{array}
$$

Such tables are called Butcher tableaus.

Examples

Euler

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

Midpoint

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

Interpretation for midpoint rule:

- $f_0 = f(y(0))$
- $f_1 = f\left(y(0) + \frac{t}{2} f_0\right) \approx f(y(\frac{t}{2}))$
- $y(t) \approx y(0) + 0\, t\, f_0 + 1\, t\, f_1$

Explicit Runge-Kutta Methods

Discussion

Given an arbitrary Runge-Kutta method, we can establish its order of convergence by showing that it is consistent and stable and then using the abstract convergence theorem presented above.

Moreover, we can use this type of analysis to determine the parameters θ_i, w_{ij} such that the Runge-Kutta method achieves the largest possible rate of convergence.

Bad news: showing consistency and stability of s-stage Runge-Kutta methods requires long and tedious calculations.

Good news: others have done the work for us, see

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

Such lists make it very easy to implement ODE solvers: simply look up a method of the desired order, translate the Butcher tableau into code and make a convergence plot to test that your implementation is correct.

Explicit Runge-Kutta Methods

Example

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

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

See `rk4_step()` regarding the implementation of this scheme, and `convergence()` for numerical demonstration that it is indeed a fourth-order method.

Explicit Runge-Kutta Methods

Remark

Any reasonable non-composite quadrature rule with s quadrature points leads to a composite quadrature rule with order of convergence in the range $[s, 2s]$. That is, the number of function evaluations per local interval and the order of convergence are proportional.

The same is not true for ODE solvers: it can be shown that the minimal number of stages s to achieve error $\mathcal{O}\left(n^{-p}\right)$ grows faster than p.

p	1	2	3	4	5	6	7	8
$\min s$	1	2	3	4	6	7	9	11

This is one of the reasons why Runge-Kutta methods with more than $s = 4$ stages are rarely used.

Explicit Runge-Kutta Methods

Summary

- Error bound for Runge-Kutta methods:

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

- Local error $\|\tilde{\Phi}_k(\tilde{y}_{k-1}) - \Phi_k(\tilde{y}_{k-1})\|$ can be estimated using Taylor series.
- Butcher-tableau representation of Runge-Kutta schemes:

$$\begin{array}{c|c} \theta_i & w_{ij} \\ \hline & w_{sj} \end{array} \qquad \longleftrightarrow \qquad \begin{aligned} f_i &= f\left(y_0 + \sum\nolimits_j w_{ij}\, t\, f_j\right) \approx f(\theta_j t) \\ \tilde{y}(t) &= y_0 + \sum\nolimits_j w_{sj}\, t\, f_j \end{aligned}$$