Applied Mathematics 225

Unit 1: Advanced ODE integration methods

Lecturer: Chris H. Rycroft

Overview

In this unit, we will look at methods to solve the ODE Initial Value Problem (IVP)

$$y' = f(x, y), \qquad y(x_0) = y_0$$

where $y(x) \in \mathbb{R}^n$ is a vector function of unknowns.

ODE IVPs have a huge range of applications in many different domains

- n **could be small** – epidemiological models, physical dynamics models, electric circuits, ...
- n **could be billions or trillions** – gravitational interactions between stars, timestepping spatial discretizations of PDEs, ...

Topics covered in AM205

- Simple low-order timestepping methods (explicit Euler, implicit Euler, *etc.*)
- Local error, global error, truncation error of a method
- Theoretical bounds on error
- Basic Runge–Kutta methods, Butcher tableaus
- Error estimation, Richardson extrapolation
- Multistep Adams–Bashforth methods

For a review, consult Unit 3, part 2 from AM 205.

Questions to answer in this Unit

AM 205[1] states Runge–Kutta methods from thin air—where do they come from? What's the mathematics behind this?

How do we design efficient and practical timestepping schemes? How do we compare methods against each other?

Can we design special methods for certain problems, *e.g.* high-order methods, symplectic methods for energy-conserving systems?

How do we deal with stiff ODE systems? What can we say in general about stability of a method?

Can we exploit parallelism?

[1] This is true for many scientific computing courses.

Books

Two excellent books on the subject:

- E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 1993.
- E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential–Algebraic Problems*. Springer, 1996.

We will follow their notational conventions

Their notation is slightly different from AM205, but the main principles are interchangeable.

The Runge–Kutta methods

Consider taking a step from y_0 to y_1 of size h. Compute the sequence of intermediate steps

$$k_i = f\left(x_0 + c_i h, y_0 + h \sum_{j=1}^{i-1} a_{ij} k_j\right)$$

for $i = 1, \ldots, s$, after which the solution is given by

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i k_i.$$

Coefficients in the method are described in a Butcher tableau

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \ldots & b_{s-1} & b_s
\end{array}$$

Low-order methods

Explicit Euler

0	
	1

Ralston's method

0		
$2/3$	$2/3$	
	$1/4$	$3/4$

Heun's 3rd order method

0			
$1/3$	$1/3$		
$2/3$	0	$2/3$	
	$1/4$	0	$3/4$

4th order Runge–Kutta method

0				
$1/2$	$1/2$			
$1/2$	0	$1/2$		
1	0	0	1	
	$1/6$	$1/3$	$1/3$	$1/6$

Simplifying assumption

Note that all methods on the previous slide satisfy

$$c_i = \sum_{j=1}^{i-1} a_{ij}.$$

This condition expresses that all points where f is evaluated are first-order approximations to the solution. It greatly simplfies the derivation of high-order methods.

For low orders, this assumption is not necessary.

Method comparison

Standard test problem in Hairer *et al.* is the Brusselator for $(y_1(x), y_2(x))$,

$$y_1' = 1 + y_1^2 y_2 - 4y_1, \qquad y_2' = 3y_1 - y_1^2 y_2$$

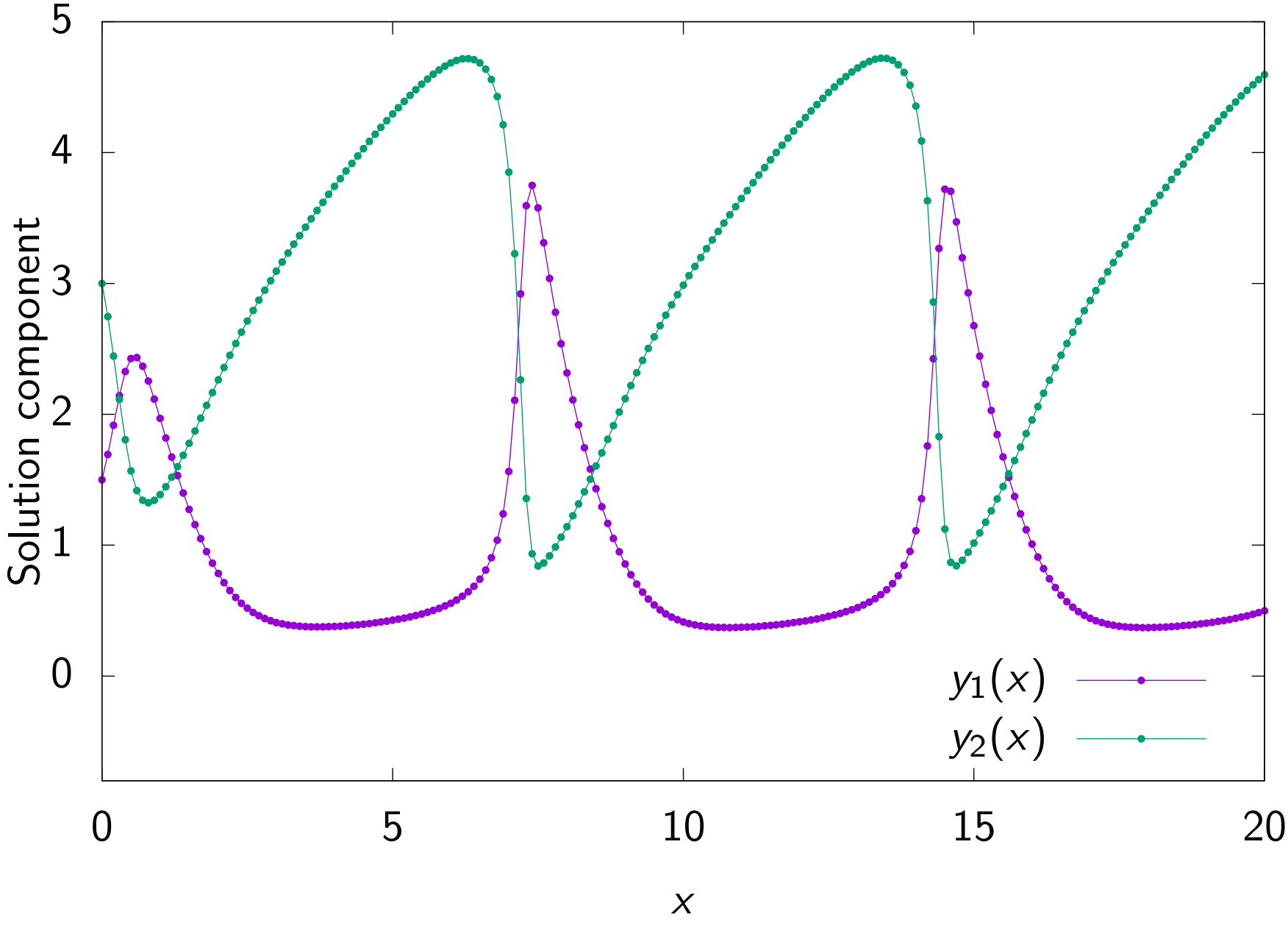
with initial conditions

$$y_1(0) = 1.5, \qquad y_2(0) = 3.$$

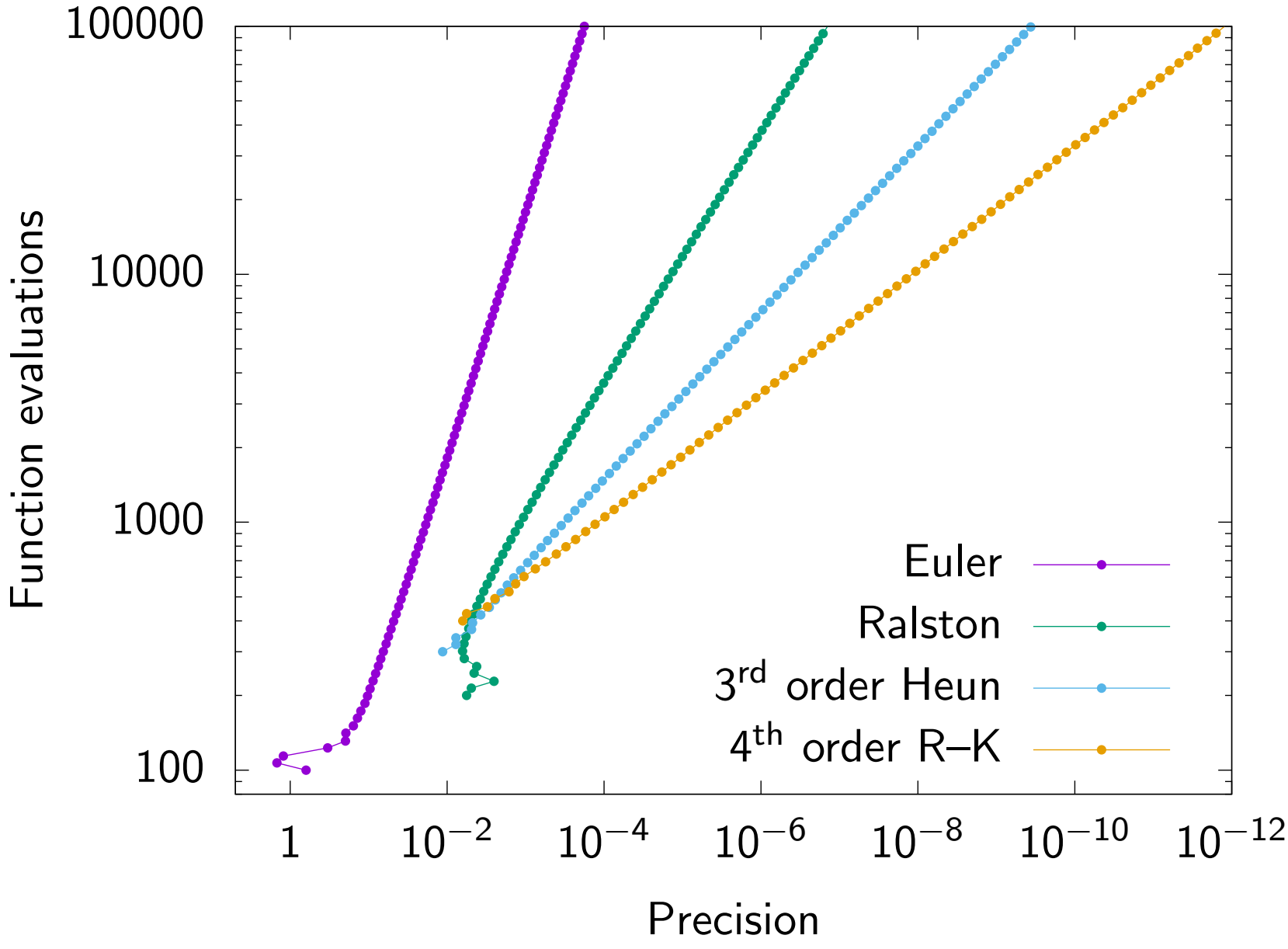
Simple model of chemical kinetics. Good test since the smoothness of the solution varies over time.

Computer demo: Four black-box solvers for the Brusselator problem

Brusselator results (Heun method)



Precision–work diagram



Order conditions for Runge–Kutta methods

The explicit Euler method is a one-step Runge-Kutta method. Consider taking a step from y_0 to y_1 of size h:

$$k_1 = f(x_0, y_0), \qquad y_1 = y_0 + hk_1 = y_0 + hf(x_0, y_0). \tag{1}$$

Compare to Taylor series expansion of the solution

$$y_1 = y_0 + hy'(x_0) + O(h^2).$$

By substituting in $y' = f(x, y)$ this yields

$$y_1 = y_0 + hf(x_0, y_0) + O(h^2). \tag{2}$$

Comparing (2) to (1) shows that the two agree up to $O(h^2)$. Hence the explicit Euler method is first-order accurate.

Extension to second order

Consider two-step method with Butcher tableau

$$\begin{array}{c|cc} 0 & & \\ \alpha & \beta & \\ \hline & a & b \end{array}$$

By comparing to second order, it can be shown[2] that if

$$a + b = 1, \qquad \alpha b = \beta b = 1/2$$

then the method is second-order accurate.

[2] https://courses.seas.harvard.edu/courses/am205/notes/am205_rk2_multi.pdf

Extension to fourth order

Repeated Taylor expansions show that a general four-step method must satisfy

$$
\begin{aligned}
b_1 + b_2 + b_3 + b_4 &= 1, \\
b_2 c_2 + b_3 c_3 + b_4 c_4 &= 1/2, \\
b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 &= 1/3, \\
b_3 a_{32} c_2 + b_4 (a_{42} c_2 + a_{43} c_3) &= 1/6, \\
b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 &= 1/4, \\
b_3 c_3 a_{32} c_2 + b_4 c_4 (a_{42} c_2 + a_{43} c_3) &= 1/8, \\
b_3 a_{32} c_2^2 + b_4 (a_{42} c_2^2 + a_{43} c_3^2) &= 1/12, \\
b_4 a_{43} a_{32} c_2 &= 1/24
\end{aligned}
$$

to be fourth-order accurate. As stated by Hairer *et al.*,

> *These computations . . . are very tedious. And they grow enormously with higher orders.*

First simplification

Suppose $y \in \mathbb{R}^n$. Rewrite the equation $y'(x) = f(x, y)$ as the augmented problem

$$\begin{pmatrix} x \\ y \end{pmatrix}' = \begin{pmatrix} 1 \\ f(x, y) \end{pmatrix}$$

and define a new variable $Y = (x, y) \in \mathbb{R}^{n+1}$ such that the equation becomes

$$Y' = F(Y).$$

Explicit x dependence is removed—this is referred to as autonomous form.

Hang onto your hats

The reader is now asked to take a deep breath, take five sheets of reversed computer paper, remember the basic rules of differential calculus, and begin the following computations.

– Hairer *et al.* (1993)

It is difficult to keep a cool head when discussing the various derivatives . . .

– S. Gill (1956)

Beginning notation

Consider autonomous ODE $y' = f(y)$ where $y(x) \in \mathbb{R}^n$. Use capital superscript indices for vectors, so that

$$(y^J)' = f^J(y^1, y^2, \ldots, y^n), \qquad J = 1, \ldots, n.$$

Rather than work with the RK steps k_i directly, we work with their arguments g_i such that $k_i = f(g_i)$. Then

$$g_i^J = y_0^J + \sum_{j=1}^{i-1} a_{ij} h f^J(g_j^1, \ldots, g_j^n), \qquad i = 1, \ldots s,$$

$$y_1^J = y_0^J + \sum_{j=1}^{s} b_j h f^J(g_j^1, \ldots, g_j^n).$$

Main result (see notes for definitions)

A Runge–Kutta method is of order p if and only if

$$\sum_{j=1}^{s} b_j \Phi_j(t) = \frac{1}{\gamma(t)}$$

for all trees t of order $\leq p$.

The number of conditions grows rapidly with the order p.

Order p	1	2	3	4	5	6	7	8	9	10
# trees	1	1	2	4	9	20	48	115	286	719
# conditions	1	2	4	8	17	37	85	200	486	1205

Principal error term

If the Runge–Kutta method is order p and f is $(p+1)$-times continuously differentiable, then

$$y^J(x_0+h) - y_1^J = \frac{h^{p+1}}{(p+1)!} \sum_{t \in T_{p+1}} \alpha(t) e(t) F^J(t)(y_0) + O(h^{p+2})$$

where

$$e(t) = 1 - \gamma(t) \sum_{j=1}^{s} b_j \Phi_j(t)$$

For 4^{th} order Runge–Kutta method, we must consider the nine trees of order 5. Coefficients $e(t)$ are given by

$$\left(-\frac{1}{24}, -\frac{1}{24}, \frac{1}{16}, -\frac{1}{4}, -\frac{2}{3}, \frac{1}{6}, \frac{1}{6}, -\frac{1}{4}, 1 \right)$$

Good methods aim to minimize these numbers.

Butcher barriers

For a five-step method, there are ten available a_{jk} coefficients and five available b_j coefficents

For order $p = 5$ there are 17 constraints. Kutta hypothesized that there might still be a solution, but this was later disproved:

Theorem: For $p \geq 5$ no explicit Runge–Kutta method exists of order p with $s = p$ stages.

Hence we must use $s > p$ stages to reach higher orders

Error estimation

To estimate error, can derive Butcher tableaus with a second estimate for the solution $\hat{y}_1$ with order $\hat{p}$. For example, Zonneveld's 4(3) method with $s = 5$ is

0					
$1/2$	$1/2$				
$1/2$	0	$1/2$			
1	0	0	1		
$3/4$	$5/32$	$7/32$	$13/32$	$-1/32$	
y_1	$1/6$	$1/3$	$1/3$	$1/6$	
$\hat{y}_1$	$-1/2$	$7/3$	$7/3$	$13/6$	$-16/3$

Solution y_1 is of order $p = 4$. Solution $\hat{y}_1$ is of order $\hat{p} = 3$. Use $y_1 - \hat{y}_1$ as an error estimate.

FSAL (First Same As Last)

For $s = 4$ stages, it is impossible to find a pair of order 4(3). But y_1 can be added as a fifth stage, and we can search for a third order method that uses all function values. One such method is

0					
$1/3$	$1/3$				
$2/3$	$-1/3$	1			
1	1	-1	1		
1	$1/8$	$3/8$	$3/8$	$1/8$	
y_1	$1/8$	$3/8$	$3/8$	$1/8$	
$\hat{y}_1$	$1/12$	$1/2$	$1/4$	0	$1/6$

The computation of k_5 is used to evaluate $\hat{y}_1$, but is then re-used as k_1 in the next integration step.

Fehlberg's order 4(5) method, RKF45

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$\frac{-8}{27}$	2	$\frac{-3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
y_1	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
$\hat{y}_1$	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Coefficients are chosen to minimize the error on the fourth-order solution y_1. Other formula $\hat{y}_1$ is order 5. Use $y_1 - \hat{y}_1$ to estimate error.

Practical step size selection

We want to write a code to automatically adjust the step size. Aim to satisfy

$$|y_{1i} - \hat{y}_{1i}| < sc_i, \qquad sc_i = Atol_i + Rtol_i \max\{|y_{0i}|, |y_{1i}|\}$$

where $Atol_i$ and $Rtol_i$ are the absolute and relative error tolerances, respectively. A scaled measure of error is then

$$err = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_{1i} - \hat{y}_{1i}}{sc_i}\right)^2}.$$

Expect $err \approx Ch^{q+1}$ where $q = \min\{p, \hat{p}\}$. Aim for $err < 1$ for an acceptable step. Thus if the current step is of size h, then the optimal step size is

$$h_{\text{opt}} = h(1/err)^{1/(q+1)}.$$

Practical step size selection

Since we want the next step to be selected with high probability, we decrease h_{opt} by a safety factor fac. Also, we do not want the step size to increase or decrease too much. Hence define

$$h_{\text{new}} = h \min\{facmax, \max\{facmin, fac(1/err)^{1/(q+1)}\}\}.$$

Reasonable parameters are

$$fac = 0.9, \qquad facmax = 3, \qquad facmin = 1/3.$$

Now, if $err < 1$ then the current step is accepted and a new step of size h_{new} is tried. Otherwise the current step is rejected and the code tries again with h_{new}. From the formula, h_{new} will be smaller than h in the case of a rejection.

A note about $p(\hat{p})$ methods

In Fehlberg's 4(5) method, the error coefficients are minimized on the fourth order solution y_1. Since the other solution $\hat{y}_1$ is of order five, the value $y_1 - \hat{y}_1$ is an estimate of the local error of y_1.

Thus by using y_1 for integration, we get an accurate fourth-order solution, and local error estimates

Wouldn't it be natural to use the high order method for integration?

You can do this! You can still use $y_1 - \hat{y}_1$ for step size selection, but the concept of error estimation is abandoned.

Not clear the local error estimation is very useful for predicting global errors, anyway.

Dormand–Prince 5(4) method (DOPRI5)

Minimizes the error terms on the higher order result, the opposite of RKF45. Fifth-order y_1 intended to be used for integration.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
y_1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{y}_1$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

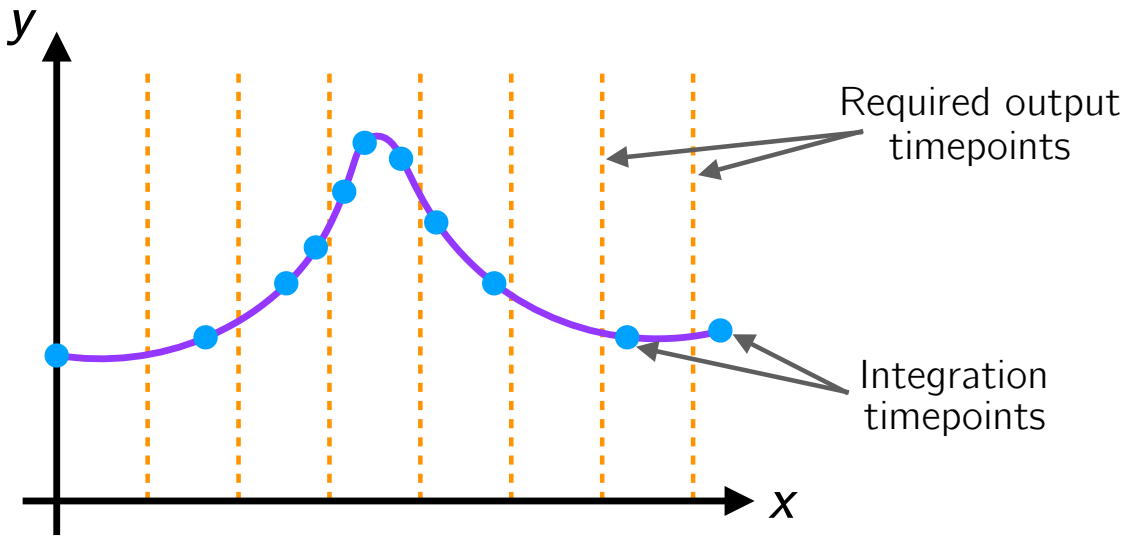
Uses FSAL approach for error estimation in $\hat{y}_1$.

Cash–Karp method 5(4,3,2,1)

Contains embedded formulae for all lower orders. Lower order formulae can be used to quit early when there are unacceptable errors, without evaluating all steps.

0						
$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	
(Order 5) y_1	$\frac{37}{378}$	0	$\frac{250}{621}$	$\frac{125}{594}$	0	$\frac{512}{1771}$
(Order 4) $\hat{y}_1$	$\frac{2825}{27648}$	0	$\frac{18575}{48384}$	$\frac{13525}{55296}$	$\frac{277}{14336}$	$\frac{1}{4}$
(Order 3) $\hat{y}_1$	$\frac{19}{54}$	0	$-\frac{10}{27}$	$\frac{55}{54}$	0	0
(Order 2) $\hat{y}_1$	$-\frac{3}{2}$	$\frac{5}{2}$	0	0	0	0
(Order 1) $\hat{y}_1$	1	0	0	0	0	0

Dense output



Adaptive high-order RK methods require infrequent, intermittent timesteps. But often we need to output the solution at frequent, specific times.

Simple solution: decrease timestep to exactly match the output times. Requires more timesteps and function evaluations.

Better solution: create polynomial interpolant of order p^* over each timestep, then cheaply evaluate the solution over the entire interval.

Dense output

Consider step of size h from (x_0, y_0) to (x_1, y_1). Aim to find a polynomial function u of degree p^* such that

$$y(x_0 + h\theta) = u(\theta)$$

Hermite interpolation can be used for $p^* = 3$:

- Know function values y_0 and y_1.
- Know derivatives $f_0 = f(x_0, y_0)$ and $f_1 = f(x_0 + h, y_1)$.

Four constraints for four unknowns in the cubic. Hermite interpolant is

$$\begin{aligned} u(\theta) = (1-\theta)y_0 + \theta y_1 \\ + \theta(\theta-1)\left((1-2\theta)(y_1-y_0) + (\theta-1)hf_0 + \theta h f_1\right). \end{aligned}$$

Dense output: required accuracy

Fourth-order methods (*e.g.* the classic 4th-order Runge–Kutta method) are popular. Is a cubic polynomial good enough?

Consider p^{th} order method, and a dense output polynomial u of order p^*

Consider interval away from initial value, $[x_n, x_{n+1}]$. Denote $z(x)$ to be the local solution starting from (x_n, y_n). Difference between true solution and dense output is

$$u(\theta) - y(x_n + \theta h) = \underbrace{(u(\theta) - z(x_n + \theta h))}_{\text{Polynomial error, } O(h^{p^*+1})} + \underbrace{(z(x_n + \theta h) - y(x_n + \theta h))}_{\text{Global error, } O(h^p)}$$

Thus obtaining a polynomial with $p^* = p - 1$ gives commensurate error terms.

Bootstrapping to higher order

Suppose we have a third order approximation available. Fix $\alpha \in (0, 1)$ and denote the third-order approximation by y_α. Then $hf(x_0 + \alpha h, y_\alpha)$ is a fourth-order approximation to $hy'(x_0 + \alpha h)$.

Find quartic polynomial $u(\theta)$ such that

$$u(0) = y_0, \qquad u(1) = y_1, \qquad u'(0) = hf(x_0, y_0),$$

$$u'(1) = hf(x_0 + h, y_1), \qquad u'(\alpha) = hf(x_0 + \alpha h, y_\alpha)$$

Can be generalized to higher orders.

More general dense output formulae

The connection between the RK stages k_i and the polynomial interpolant is not straightforward

For some higher-order RK method with s stages, we can add $s^* - s$ new stages and then evaluate

$$u(\theta) = y_0 + h \sum_{i=1}^{s^*} b_i(\theta) k_i$$

for some polynomials $b_i(\theta)$.

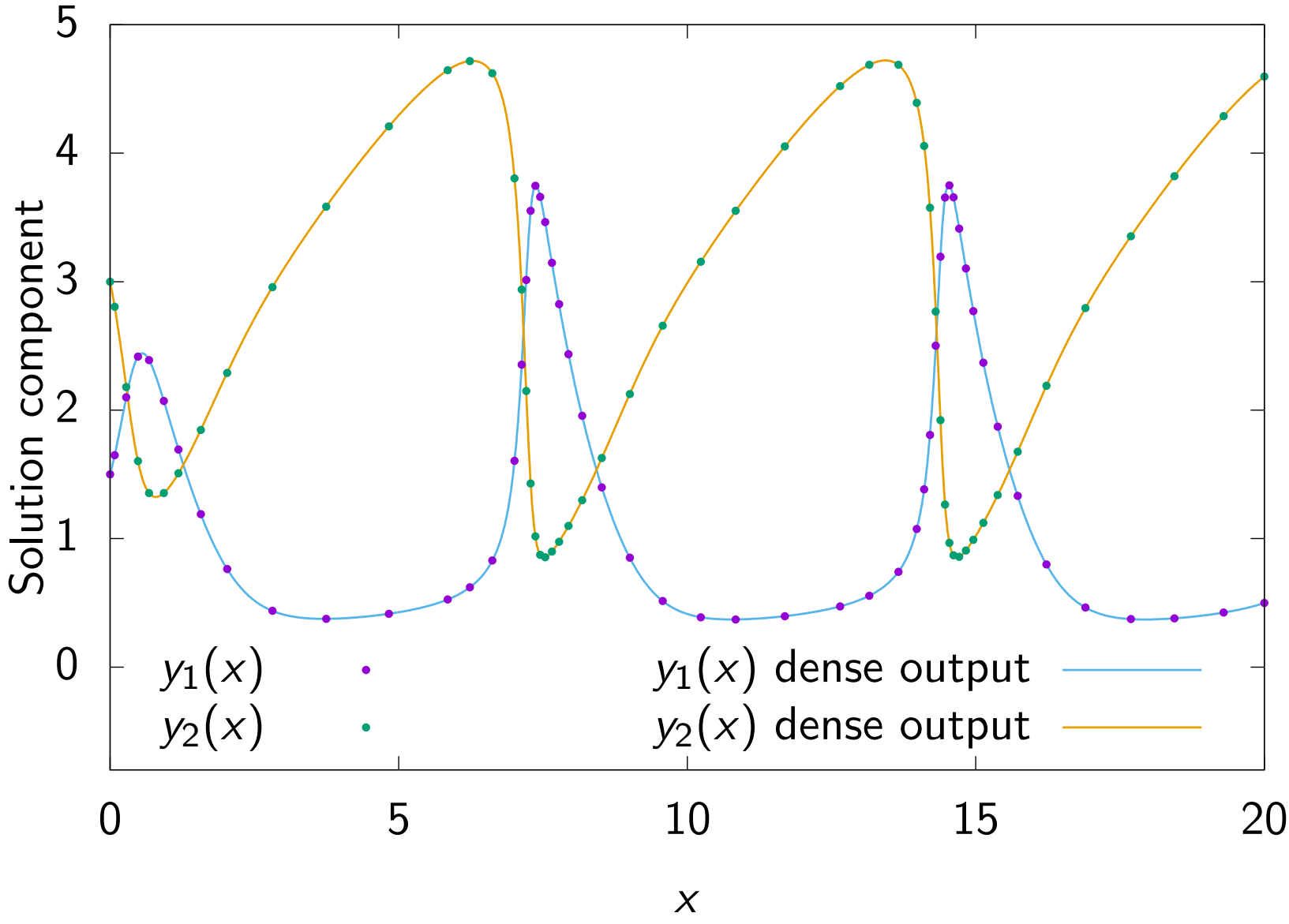
Dormand–Prince 8(5,3) (DOP853)

Eigth-order method with $s = 13$ steps and the FSAL property. Contains embedded formulae of orders 5 and 3 for adaptive step size control.

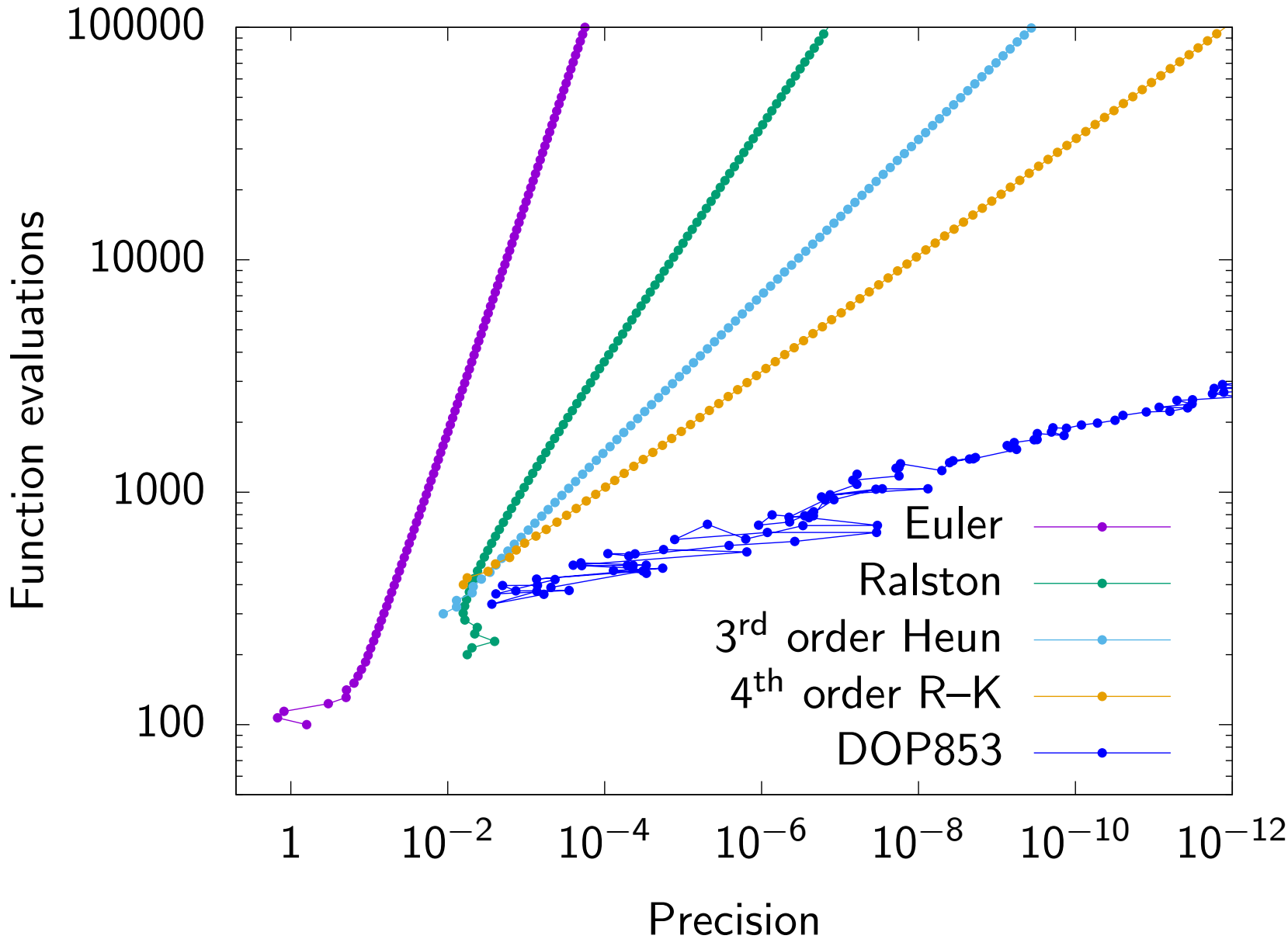
$i=2$	1												
$i=3$	1	1											
$i=4$	1	0	1										
$i=5$	1	0	1	1									
$i=6$	1	0	0	1	1								
$i=7$	1	0	0	1	1	1							
$i=8$	1	0	0	1	1	1	1						
$i=9$	4	0	0	3	3	4	4	4					
$i=10$	4	0	0	3	3	4	4	4	4				
$i=11$	4	0	0	3	3	4	4	4	4	2			
$i=12$	4	0	0	3	3	4	4	4	4	2	2		
$i=13$	1	0	0	0	0	1	1	1	1	1	1	1	
$i=14$	5	0	0	0	0	0	5	5	5	5	5	5	5
$i=15$	5	0	0	0	0	5	5	5	0	0	5	5	5	5
$i=16$	5	0	0	0	0	5	5	5	5	0	0	0	5	5	5

Fig. 5.3. Steps in the construction of an 8th order RK method; the entries 0 indicate vanishing coefficients; the stages $i = 14, 15, 16$ will be used for dense output, see II.6.

DOP853 on the Brusselator (Atol $= 10^{-6}$)



Updated work–precision plot



Implicit methods

Implicit methods are more complicated, but frequently have better stability properties that allow for larger timesteps

Simple implicit method is the backward Euler method,

$$y_1 = y_0 + hf(x_1, y_1).$$

Another is the implicit midpoint method,[3]

$$y_1 = y_0 + hf\left(x_0 + \frac{h}{2}, \frac{y_0 + y_1}{2}\right)$$

If rewritten as

$$k_1 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{hk_1}{2}\right), \qquad y_1 = y_0 + hk_1,$$

then it starts to look like a Runge–Kutta method.

[3] This featured on an AM205 homework assignment.

A more general Runge–Kutta definition

The method given by

$$k_i = f\left(x_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

for $i = 1, \ldots, s$, and

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i k_i$$

is called an s-stage Runge–Kutta method. Furthermore

- If $a_{ij} = 0$ for all $i \leq j$ we have an explicit (ERK) method.
- If $a_{ij} = 0$ for $i < j$ and at least one $a_{ii} \neq 0$, we have a diagonal implicit Runge–Kutta (DIRK) method.
- Otherwise we have an implicit Runge-Kutta (IRK) method.

Low-order methods

Implicit Euler
(DIRK, order 1)

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

Implicit midpoint rule
(DIRK, order 2)

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

Hammer & Hollingsworth #1
(DIRK)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 2/3 & 1/3 & 1/3 \\ \hline & 1/4 & 3/4 \end{array}$$

Hammer & Hollingsworth #2
(IRK, order 4)

$$\begin{array}{c|cc} 1/2 - \sqrt{3}/6 & 1/4 & 1/4 - \sqrt{3}/6 \\ 1/2 + \sqrt{3}/6 & 1/4 + \sqrt{3}/6 & 1/4 \\ \hline & 1/2 & 1/2 \end{array}$$

Connection with Gaussian quadrature

Consider applying Hammer & Holligsworth #2 (HH2) to the simplified problem

$$y' = f(x)$$

with initial condition $y(0) = 0$. This has an integral solution

$$y(x) = \int_0^x f(t)dt.$$

One step of size h with HH2 yields

$$y_1 = y_0 + \frac{h}{2}(f\left(h(1/2 - \sqrt{3}/6)\right) + f\left(h(1/2 + \sqrt{3}/6)\right)).$$

This is exactly equivalent to two-point Guassian quadrature!

HH2 is fourth-order accurate, even when applied to the general case $y' = f(x, y)$.

Connection with Gaussian quadrature

Similarly, the order 2 implicit midpoint rule is equivalent to one-point Gaussian quadrature.

Kuntzmann and Butcher showed it is possible to compute s-stage IRK methods of order $2s$ based on Gaussian quadrature, for any s. Three point Gaussian quadrature leads to the following sixth-order scheme

$$
\begin{array}{c|ccc}
1/2 - \sqrt{15}/10 & 5/36 & 2/9 - \sqrt{15}/15 & 5/36 - \sqrt{15}/30 \\
1/2 & 5/36 + \sqrt{15}/24 & 2/9 & 5/36 - \sqrt{15}/24 \\
1/2 + \sqrt{15}/10 & 5/36 + \sqrt{15}/30 & 2/9 + \sqrt{15}/15 & 5/36 \\
\hline
 & 5/18 & 4/9 & 5/18
\end{array}
$$

Other quadrature schemes (*e.g.* Gauss–Lobatto) also lead to IRK schemes.

Existence of a solution

Solving an IRK will require root finding in general. Is a solution guaranteed? Yes, under certain conditions.

Theorem: Let $f(x, y)$ satisfy a Lipschitz condition[4] with constant L. If

$$h < \frac{1}{L \max_i \sum_j |a_{ij}|}$$

then there exists a unique solution which can be obtained by iteration.

[4] With respect to y.

Existence of a solution

Proof: Consider an iterative process where superscript (m) marks the m^{th} iteration. Then

$$k_i^{(m+1)} = f\left(x_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij} k_j^{(m)}\right)$$

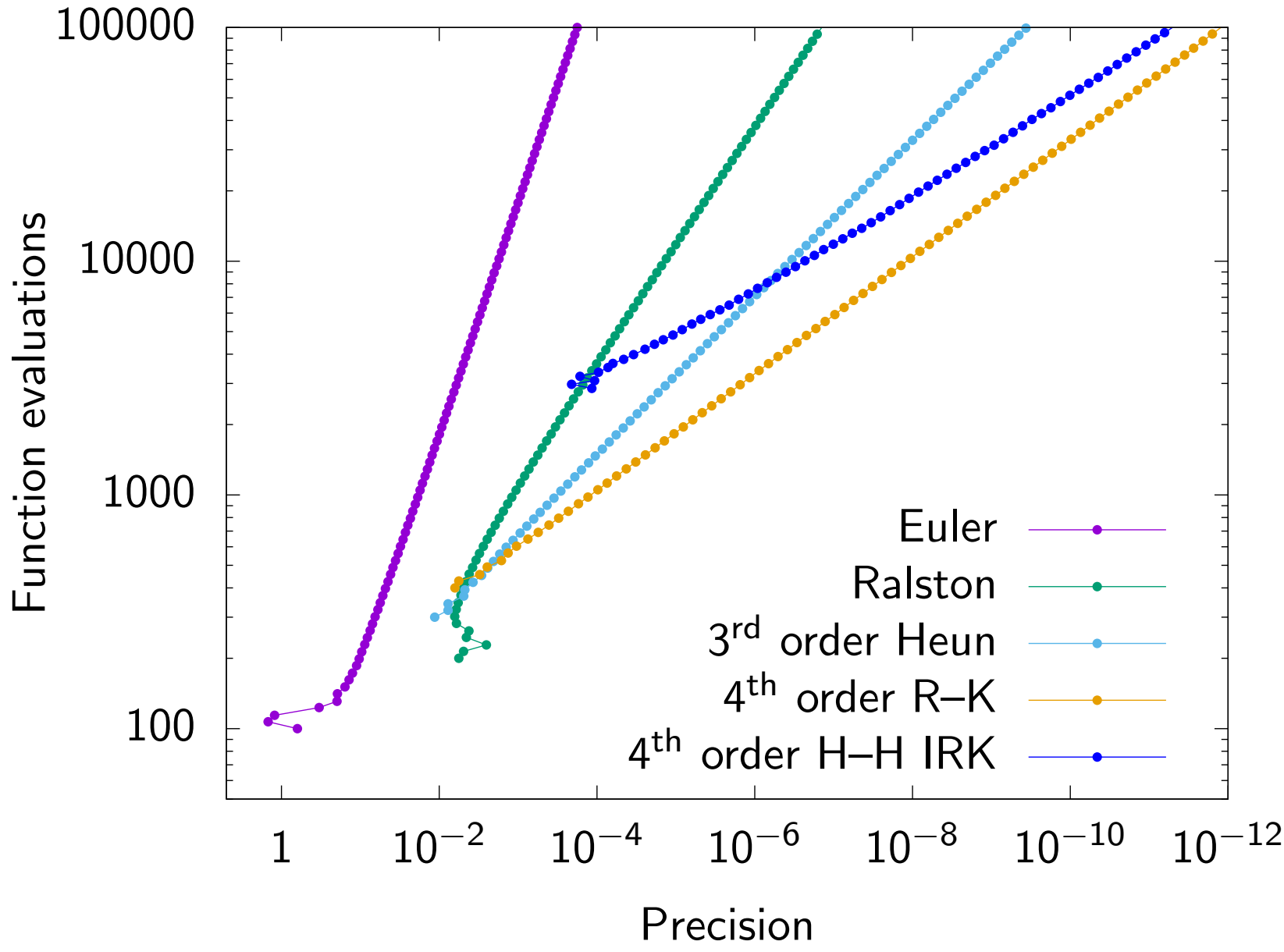
Define $K \in \mathbb{R}^{sn}$ as $K = (k_1, k_2, \ldots, k_s)^{\mathsf{T}}$ and use the norm $\|K\| = \max_i \|k_i\|$. Then

$$F_i(K) = f\left(x_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

for $i = 1, \ldots s$. Then the Lipschitz condition and the triangle inequality show that

$$\|F(K_1) - F(K_2)\| \leq hL \max_i \sum_{j=1}^{s} |a_{ij}| \|K_1 - K_2\|$$

Work–precision plot with Hammer–Hollingsworth



Hammer–Hollingsworth performance[5]

Hammer–Hollingsworth code exhibits asympototically fourth-order convergence

Larger prefactor than the classic fourth-order RK scheme, since each step requires multiple (roughly 5–10) fixed-point iterations to reach convergence

Could be sped up by using faster root-finding methods (*e.g.* Newton–Raphson)

Since it is implicit, Hammer–Hollingsworth is better suited to stiff problems

[5] Note: the Hammer–Hollingsworth code is added to the "low order ODE" example codes.

Richardson extrapolation

Suppose that y_{k+2} is the numerical result of two steps with size h of a Runge–Kutta method of order p, and w is the result of one big step with step size $2h$. Then the error of y_{k+2} can be approximated as

$$y(t_k + 2h) - y_{k+2} = \frac{y_{k+2} - w}{2^p - 1} + O(h^{p+2})$$

and

$$\hat{y}_{k+2} = y_{k+2} + \frac{y_{k+2} - w}{2^p - 1}$$

is an approximation of order $p + 1$ to $y(t_0 + 2h)$.

Extrapolation methods

Richardson extrapolation relies on the structure of local error. But global error is also structured.

Theorem (Gragg, 1964): The global error of a numerical method[6] of order p is

$$y(x) - y_h(x) = e_p(x)h^p + e_{p+1}(x)h^{p+1} + \ldots + e_N(x)h^N + E_h(x)h^{N+1}$$

where $E_h(x)$ is bounded for $x_0 \leq x \leq x_{\text{end}}$ and $0 \leq h \leq h_0$. Furthermore $e_j(x)$ are the solutions to inhomogeneous diffferential equations with $e_j(x_0) = 0$.

[6] Under mild conditions; see Hairer *et al.*, Chapter II.8 for full details.

Extrapolation methods

The extrapolation methods are a family of numerical methods that generalize Richardson extrapolation to exploit the structured nature of the error terms.

Let H be a basic step size. Introduce a seqeuence of positive integers

$$n_1 < n_2 < n_3 < \dots$$

and define corresponding step sizes of $h_i = H/n_i$. For each i, compute n_i steps of size h_i to obtain

$$y_{h_i}(x_0 + H) = T_{i,1}$$

Extrapolation methods

Define a polynomial

$$p(h) = \hat{y} - e_p h^p - e_{p+1} h^{p+1} - \ldots - e_{p+k-2} h^{p+k-2}$$

such that $p(h_i) = T_{i,1}$ for $i = j, j-1, \ldots, j-k+1$.

This gives k constraints for the k unknowns $\hat{y}, e_p, \ldots, e_{p+k-2}$.

Extrapolate to $h = 0$ and define

$$T_{j,k} = p(0) = \hat{y}$$

Then $T_{j,k}$ is numerical approximation with order $p + k - 1$.

Extrapolation methods

Obtain a family of solutions

$$
\begin{array}{ccccccc}
T_{11} & & & & & & \\
T_{21} & T_{22} & & & & & \\
T_{31} & T_{32} & T_{33} & & & & \\
T_{41} & T_{42} & T_{43} & T_{44} & & & \\
T_{51} & T_{52} & T_{53} & T_{54} & T_{55} & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}
$$

with variable order; very convenient for error estimation and designing an adaptive order method

Choices for the integer sequence $n_1 < n_2 < \ldots$

The Romberg sequence

$$1, 2, 4, 8, 16, 32, 64, 128, \ldots$$

The Bulirsch sequence[7]

$$1, 2, 3, 4, 6, 8, 12, 16, 24, 32, \ldots$$

The harmonic sequence

$$1, 2, 3, 4, 5, 6, 7, 8, \ldots$$

[7] This is made by alternately multiplying by 2 and 1.5.

Direct method to find $T_{j,k}$

System of equations is

$$H\times\begin{pmatrix} 1 & 1/n_j^p & \dots & 1/n_j^{p+k-2} \\ 1 & 1/n_{j+1}^p & \dots & 1/n_{j+1}^{p+k-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1/n_{j-k+1}^p & \dots & 1/n_{j-k+1}^{p+k-2} \end{pmatrix}\begin{pmatrix} \hat{y} \\ e_p \\ \vdots \\ e_{p+k-2} \end{pmatrix}=\begin{pmatrix} T_{j,1} \\ T_{j+1,1} \\ \vdots \\ T_{j-k+1,1} \end{pmatrix}.$$

Then $T_{j,k}=\hat{y}$.

Note that this is similar to a Vandermonde matrix problem, which occurs during polynomial interpolation of set of discrete points.

Aitken–Neville recurrence relation

We only need $\hat{y}$, not all of the e_i. The Aitken–Neville recurrence relation gives

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\frac{n_j}{n_{j-k}} - 1},$$

which does not require matrix inversion.

Extrapolation method convergence

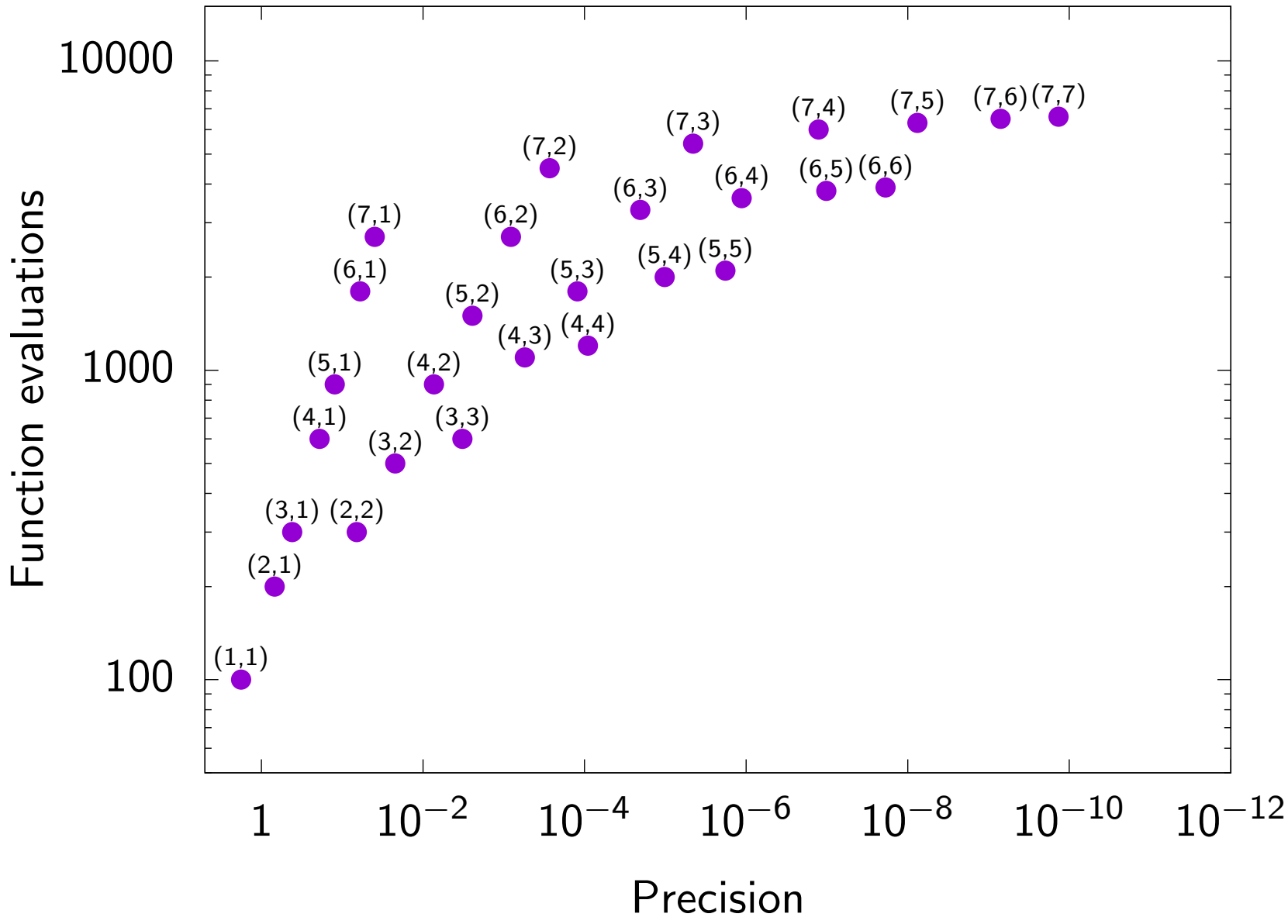
Computer demo: testing the extrapolation method on the test ODE system

$$y_1' = -xy_2$$
$$y_2' = xy_1$$

with initial conditions $y_1(0) = 1, y_2(0) = 0$. Has exact solution

$$y_1(x) = \cos\frac{x^2}{2}, \qquad y_2(x) = \sin\frac{x^2}{2}.$$

Extrapolation method convergence for (j, k) for $T_{j,k}$



A further improvement

Consider a centered finite-difference derivative of a function $f(x)$:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + e_2 h^2 + e_4 h^4 + e_6 h^6 + \dots$$

By symmetry only even powers of h are present in the asymptotic expansion.

If we use a similar approach here, we can double the order of an extrapolation method . . .

. . . but we need a basic forward integration method with only even error terms.

Gragg's method

Consider Gragg's method

$$\begin{aligned} y_1 &= y_0 + hf(x_0, y_0) \\ y_{i+1} &= y_{i-1} + 2hf(x_i, y_i), \qquad i = 1, 2, \ldots, 2n \end{aligned}$$

with smoothing operator

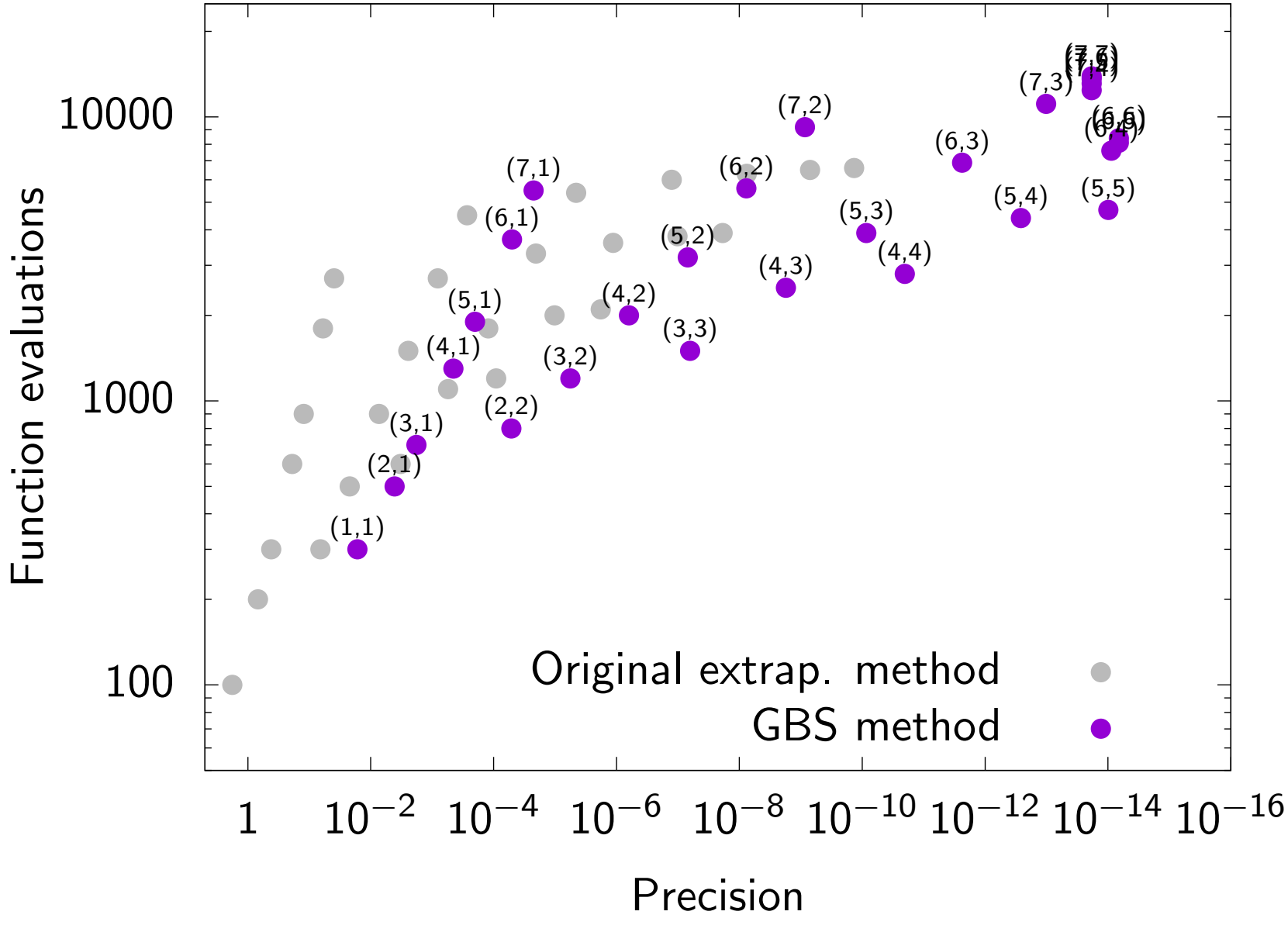
$$S_h(x) = \frac{y_{2n-1} + 2y_n + y_{2n+1}}{4}.$$

This has an asymptotic expansion in powers of h^2. Must modify recurrence relation to

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\left(\frac{n_j}{n_{j-k}}\right)^2 - 1}.$$

Using extrapolation on this yields the powerful and practical Gragg–Bulirsch–Stoer (GBS) method.

GBS convergence for different $T_{j,k}$ (Bulirsch seqence)



Comments on GBS method convergence

As expected, the GBS method doubles the order over the original extrapolation approach, leading to a very good ratio of precision to function evaluations

For $k = 5$ (tenth order) and beyond, numerical roundoff begins to dominate and there is limited practical benefit

These methods are still useful for extended-precision calculations. There are libraries for quadruple-precision float point numbers,[8] using 16 bytes each, yielding about 32 decimal digits of precision.

[8] See the QD library: http://crd-legacy.lbl.gov/~dhbailey/mpdist/

Parallelizing timestepping methods

In parallel computations, it is common practice to divide the workload in space

But there is also interest in methods to divide the workload in time, *i.e.*, devising a timestepping that processes multiple parts of the update simultaneously

Possible advantage: parallelization can be part of a black box timestepper without requiring any adjustments to the main simulation

Runge–Kutta schemes involve multiple intermediates—there is hope they can be computed in parallel

Parallelizing timestepping methods

Question: can parts of this Runge–Kutta scheme be processed simultaneously?

0				
×	×			
×	×	0		
×	×	×	0	
	×	×	×	×

Here the × symbol represents a non-zero entry

Parallelizing timestepping methods

In the previous example, k_2 and k_3 can be process simultaneously, and k_4 can begin as soon as k_2 is done.

Unfortunately, we hit a severe restriction.

Theorem: For an explicit Runge–Kutta method with σ sequential stages the order p satisfies $p \leq \sigma$ for any number of available processors.

This follows from the order condition for the "tall tree" of order p, corresponding to a long chain of p vertices. This requires at least p sequential stages to satisfy.

Parallelization: there is hope

We saw that high order methods require $s > p$ stages. Processing in parallel could feasibly reduce the number of sequential stages to p.

Extrapolation methods are highly suited to parallelization: each $T_{j,1}$ can be computed independently.

Other contemporary methods (*e.g.* spectral deferred corrections) have been shown to be well-suited to parallelization.[9]

[9] M. L. Minion, Comm. App. Math. and Comp. Sci. **5**, 265–301 (2010).

Second-order differential equations

We frequently need to solve second-order differential equations of the form

$$y'' = f(x, y', y'')$$

A simple method of solution is to write as a first-order system

$$\begin{pmatrix} y \\ y' \end{pmatrix}' = \begin{pmatrix} y' \\ f(x, y, y') \end{pmatrix}$$

with initial conditions $y(x_0) = y_0, y'(x_0) = y_0'$.

Second-order differential equations

Substituting into a standard Runge–Kutta scheme gives

$$k_i = y'_0 + h \sum_{k=1}^{s} a_{ij} k'_j$$

$$k'_i = f\left(x_0 + c_i h, y_0 + h \sum_{j=1}^{s} a_{ij} k_j, y'_0 + h \sum_{j=1}^{s} a_{ij} k'_j \right)$$

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i k_i$$

$$y'_1 = y'_0 + h \sum_{i=1}^{s} b_i k'_i$$

Second-order differential equations

Can eliminate k_i by direct substitution to obtain

$$k_i' = f\left(x_0 + c_i h, y_0 + c_i h y_0' + h^2 \sum_{j=1}^{s} \bar{a}_{ij} k_j', y_0' + h \sum_{j=1}^{s} a_{ij} k_j'\right)$$

$$y_1 = y_0 + h y_0' + h^2 \sum_{i=1}^{s} \bar{b}_i k_i'$$

$$y_1' = y_0' + h \sum_{i=1}^{s} b_i k_i'$$

where

$$\bar{a}_{ij} = \sum_{k=1}^{s} a_{ik} a_{kj}, \qquad \bar{b}_i = \sum_{j=1}^{s} b_j a_{ji}$$

Nyström methods

In the rewritten form, a_{ij} & b_i are used to find y_1, and $\bar{a}_{ij}$ & $\bar{b}_i$ are used to update y_1'. Nyström began to look for general tableaus for the two sets of coefficients that do not satisfy the algebraic constraints for $\bar{a}_{ij}$ and $\bar{b}_i$ on the previous slide.

0									
$1/2$	$1/8$		$\bar{a}_{ij}$		$1/2$		a_{ij}		
$1/2$	$1/8$	0			0	$1/2$			
1	0	0	$1/2$		0	0	1		
$\bar{b}_i \rightarrow$	$1/6$	$1/6$	$1/6$	0	$1/6$	$2/6$	$2/6$	$1/6$	$\leftarrow b_i$

Does not result in a large speedup.

Nyström methods

However we do gain a big advantage for problems with the form $y'' = f(x, y)$!

Method becomes

$$k_i' = f\left(x_0 + c_i h, y_0 + c_i h y_0' + h^2 \sum_{j=1}^{s} \bar{a}_{ij} k_j'\right),$$

$$y_1 = y_0 + h y_0' + h^2 \sum_{i=1}^{s} \bar{b}_i k_i', \qquad y_1' = y_0' + h \sum_{i=1}^{s} b_i k_i'.$$

The a_{ij} coefficients are no longer needed.

Nyström methods

An example of a very efficient four-step fifth-order[10] Nyström method has Butcher tableau

0				
$1/5$	$1/50$		$\bar{a}_{ij}$	
$2/3$	$-1/27$	$7/27$		
1	$3/10$	$-2/35$	$9/35$	
$\bar{b}_i$	$14/336$	$100/336$	$54/336$	0
b_i	$14/336$	$125/336$	$162/336$	$35/336$

[10] Specifically, $y(x_0 + h) - y_1 = O(h^{p+1})$ and $y'(x_0 + h) - y'_1 = O(h^{p+1})$. In this case $p = 5$.

Symplectic methods

Important class of ODEs arise from Hamiltonian systems given by

$$\dot{p}_i = -\frac{\partial H}{\partial q_i}(p, q), \qquad \dot{q}_i = \frac{\partial H}{\partial p_i}$$

where $p = (p_1, p_2, \ldots, p_n)$ are generalized momentum variables and $q = (q_1, q_2, \ldots, q_n)$. Represent energy-conserving physical systems where $H(p, q)$ is the energy (*e.g.* mechanical systems, orbital dynamics)

Symplectic integration methods exactly conserve $H(p, q)$. This is not true for most methods we have covered—for an order p method we could obtain global errors in H of size $O(h^p)$.

See supplemental notes and optional homework question.

Stability

An important consideration in ODE integration is stability—will the numerical scheme be well-behaved for a given step size?[11]

Consider test equation $y' = \lambda y$. For $\lambda < 0$, two solutions that start from similar initial conditions y_0 and $y_0 + \epsilon$ will stay close together. We want our numerical scheme to do the same.

For explicit Euler, timestep restriction is $-2 \leq \lambda h \leq 0$. Large λ implies small h.

[11] See AM205 unit 3 for details and definitions.

Stiffness

We frequently encounter stiff ODE systems. There is no mathematical definition of stiffness, but main principle is that the system has components that evolve on different scales, *e.g.*

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} -1000 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

Eigenvalues of $\lambda = -1, -1000$. Step size restriction is set by largest eigenvalue, and thus $0 \leq h1000 \leq 2$.

Stability analysis for Runge–Kutta methods

Let $\varphi(x)$ be a smooth solution of $y' = f(x, y)$. Then

$$y'(x) = f(x, \varphi(x)) + \frac{\partial f}{\partial y}(x, \varphi(x))(y(x) - \varphi(x)) + \dots.$$

Substituting in $\bar{y}(x) = y(x) - \varphi(x)$ gives

$$\bar{y}'(x) = \frac{\partial f}{\partial x}(x, \varphi(x)) \cdot \bar{y}(x) + \dots = J(x)\bar{y}(x) + \dots$$

where J is the Jacobian of f with respect to y. Thus assuming J is approximately constant over an interval,

$$\bar{y}'(x) = J\bar{y}$$

Thus studying the stability properties of $y' = \lambda y$, provides insight about any general nonlinear ODE system.

Definition of stability

Applying Explicit Euler to this problem gives

$$y_{m+1} = R(h\lambda)y_m$$

where $R(z) = 1 + z$.

For a general method, define $R(z)$ as the stability function, which is the numerical solution after one step of

$$y' = \lambda y, \qquad y_0 = 1, \qquad z = h\lambda,$$

which is called the Dahlquist test equation. The set

$$S = \{z \in \mathbb{C} : |R(z)| \leq 1\}$$

is the stability domain of the method.

Stability analysis for Runge–Kutta methods

For a Runge–Kutta method

$$g_i = 1 + z \sum_{j=1}^{s} a_{ij} g_j,$$

$$R(z) = 1 + z \sum_{j=1}^{s} b_j g_j.$$

For an explicit method

$$R(z) = 1 + z \sum_{j} b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + \dots$$

Stability analysis for Runge–Kutta methods

Theorem: If the Runge–Kutta method is of order p, then

$$R(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots + \frac{z^p}{p!} + O(z^{p+1})$$

Since the numerical solution of the test equation is e^z, we must have $e^z - R(z) = O(z^{p+1})$.

Stability analysis for implicit Runge–Kutta methods

Applying the implicit Euler method to the Dahlquist test equation yields

$$y_1 = 1 + h\lambda y_1 \quad \Longrightarrow \quad y_1 = \frac{1}{1 - h\lambda} \quad \Longrightarrow \quad R(z) = \frac{1}{1 - z}$$

Stability analysis for implicit Runge–Kutta methods

Quiz: what are the stability functions for the following implicit schemes?

Implicit midpoint rule

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

Hammer & Hollingsworth #1 (DIRK)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 2/3 & 1/3 & 1/3 \\ \hline & 1/4 & 3/4 \end{array}$$

Hammer & Hollingsworth #2 (IRK)

$$\begin{array}{c|cc} 1/2 - \sqrt{3}/6 & 1/4 & 1/4 - \sqrt{3}/6 \\ 1/2 + \sqrt{3}/6 & 1/4 + \sqrt{3}/6 & 1/4 \\ \hline & 1/2 & 1/2 \end{array}$$

Solutions

Implicit midpoint rule:

$$R(z) = \frac{1 + z/2}{1 - z/2}$$

Hammer–Hollingsworth #1:

$$R(z) = \frac{1 + 4z/6 + z^2/6}{1 - z/3}$$

Hammer–Hollingsworth #2:

$$R(z) = \frac{1 + z/2 + z^2/12}{1 - z/2 + z^2/12}$$

Definitions

Definition: A method is called A-stable if its stability domain satisfies

$$S \supseteq \{z \in \mathbb{C} : z \leq 0\}.$$

Also follows from properties of analytic functions: a method is A-stable if

$$R(iy) \leq 1$$

for all $y \in \mathbb{R}$, and $R(z)$ is analytic for $z < 0$.

A further definition

Some methods (*e.g.* implicit midpoint) have stability regions that exactly coincide with the left half plane.

This is not as desirable as expected. Since R is an analytic function,

$$\lim_{z\to-\infty} R(z) = \lim_{z\to\infty} R(z) = \lim_{z=iy, y\to\infty} R(iy)$$

and the final term is equal to one in this case. Very stiff components are damped out very slowly.

Definition: A method is called L-stable if it is A-stable and

$$\lim_{z\to\infty} R(z) = 0.$$

One last definition

The previous argument suggests A-stability is too weak. But in other ways it is too strong, since many good methods are ruled out. This motivates one final definition.

Definition: A method is said to be $A(\alpha)$-stable if the sector

$$S_\alpha = \{z \in C \, : \, |\arg(-z)| < \alpha, z \neq 0\}$$

is contained in the stability region.