Simulation Methods

Numerical Methods for Ordinary Differential Equations

Joan Carles Tatjer

Departament de Matemàtiques i Informàtica

Universitat de Barcelona

The first Runge-Kutta methods I

The Euler's method has low accurancy. We can obtain better methods: We can transform the Cauchy problem

$$x' = f(t, x), \qquad x(t_0) = x_0$$

into the integral equation

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, ds.$$

Then, using the midpoint rule to approximate the integral, we obtain

$$\begin{aligned}
x(t_0 + h_0) &\approx x_1 = x_0 + h_0 f\left(t_0 + \frac{h_0}{2}, x\left(t_0 + \frac{h_0}{2}\right)\right), \\
x(t_1 + h_1) &\approx x_2 = x_1 + h_1 f\left(t_1 + \frac{h_1}{2}, x\left(t_1 + \frac{h_1}{2}\right)\right), \\
&\dots \\
x(T) &\approx X = x_{n-1} + h_{n-1} f\left(t_{n-1} + \frac{h_{n-1}}{2}, x\left(t_{n-1} + \frac{h_{n-1}}{2}\right)\right),
\end{aligned}$$

The first Runge-Kutta methods II

Comment

The global error is known to be bounded by Ch^2 *if* $f(t,x) = f(x)$

If f depends on t, we compute $x(t_i + \frac{h_i}{2})$ using one small Euler step with step size $h/2$. Then we can write the method as

$$\left.\begin{array}{rcl} k_1 & = & f(t_0, x_0), \\ k_2 & = & f\left(t_0 + \frac{h}{2}, x_0 + \frac{h}{2}k_1\right), \\ x_1 & = & x_0 + hk_2. \end{array}\right\} \quad \textbf{(modified Euler's method)}$$

Again to obtain the order we compute the Taylor expansion of x_1 as a function of h.

Order of covergence of the modified Euler's method I

$$x_1 = x_0 + h \cdot f\left(t_0 + \frac{h}{2}, x_0 + \frac{h}{2} f(t_0, x_0)\right) =$$

$$= x_0 + hf(t_0, x_0) + \frac{h^2}{2}(f_t + f_x f)(t_0, x_0) + \frac{h^3}{8}(f_{tt} + 2f_{tx} f + f_{xx} f^2)(t_0, x_0) + \cdots$$

and compare with

$$x(t_0 + h) = x_0 + hf(t_0, x_0) + \frac{h^2}{2}(f_t + f_x f)(t_0, x_0) +$$

$$+ \frac{h^3}{6}(f_{tt} + 2f_{tx} f + f_{xx} f^2 + f_x f_t + f_x^2 f)(t_0, x_0) + \cdots .$$

Substracting these two equations, we obtain for the error of the first step

$$x(t_0 + h) - x_1 = \frac{h^3}{24}(f_{tt} + 2f_{tx} f + f_{xx} f^2 + 4(f_x f_t + f_x^2 f))(t_0, x_0) + \cdots$$

Order of covergence of the modified Euler's method II

When all the second derivatives are bounded, we thus obtain:

$$|x(t_0 + h) - x_1| \leq Kh^3,$$

and the method has global order of convergence equal to **2**.

Heun method

The analogous extension of the trapezoidal rule gives the **Heun method:** In this case we have the map

$$\tilde{\varphi}(h;t,x) = x + \frac{h}{2}(f(t,x) + f(t+h, x+hf(t,x))).$$

It is easy to prove that this method is of global order of convergence **2** because

$$\tilde{\varphi}(h;t,x) = x + hf(t,x) + \frac{h^2}{2}(f_t + f_x f)(t,x) + \frac{h^3}{4}(f_{tt} + 2f_{tx}f + f_{xx}f^2)(t,x) + \cdots .$$

Generalization of the modified Euler's and Heun methods I

At this moment, we observe that both methods have functions of the following form:

$$\tilde{\varphi}(h;t,x) = x + h(b_1 f(t,x) + b_2 f(t + c_2 h, x + a_{21} h f(t,x))).$$

If we impose that this method is of order two, we have

$$x(t_0 + h) - x_1 = h(1 - b_1 - b_2) f(t_0, x_0) +$$

$$+ \frac{h^2}{2}((1 - 2b_2 c_2) f_t + (1 - 2b_2 a_{21}) f_x f) + O(h^3).$$

Therefore,

$$b_1 + b_2 = 1, \qquad b_2 c_2 = \frac{1}{2}, \qquad b_2 a_{21} = \frac{1}{2}.$$

As a particular cases we obtain for $b_1 = b_2 = 1/2$, $c_2 = a_{21} = 1$ the Heun's method and for $b_1 = 0$, $b_2 = 1$, $c_2 = a_{21} = 1/2$ the modified Euler's method.

Generalization of the modified Euler's and Heun methods II

The the global error of all these methods is bounded by Ch^2. Then they are improvements on the Euler method. For high precision computations we need to find still better methods; this will be the task of what follows.

General formulation of Runge-Kutta methods

Definition

Let s be an integer (the number of stages) and

$$a_{21}, a_{31}, a_{32}, \ldots, a_{s1}, a_{s2}, \ldots, a_{s,s-1}, b_1, \ldots, b_s, c_2, \ldots, c_s$$

be real coefficients. Then the method

$$\begin{aligned}
k_1 &= f(t_0, x_0) \\
k_2 &= f(t_0 + c_2 h, x_0 + h a_{21} k_1) \\
k_3 &= f(t_0 + c_3 h, x_0 + h(a_{31} k_1 + a_{32} k_2)) \\
&\ldots \\
k_s &= f(t_0 + c_s h, x_0 + h(a_{s1} k_1 + \cdots + a_{s,s-1} k_{s-1})) \\
x_1 &= x_0 + h(b_1 k_1 + \cdots + b_s k_s)
\end{aligned}$$

is called an **s-stage explicit Runge-Kutta method** *(ERK) for the Cauchy problem $x' = f(t, x)$, $x(t_0) = x_0$.*

Comment

Usually, the c_i satisfy the conditions

$$c_2 = a_{21}, c_3 = a_{31} + a_{32}, \ldots, c_s = a_{s1} + \cdots + a_{s,s-1}.$$

These conditions express that all the points where f is evaluated are first order approximations to the solution.

Definition

A Runge-Kutta method has order p if for sufficiently smooth problems,

$$\|x(t_0 + h) - x_1\| \leq Kh^{p+1}.$$

We represent a RK method with the **Butcher table:**

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

represents

$$
\begin{aligned}
k_1 &= f(t_0, x_0) \\
k_2 &= f(t_0 + c_2 h, x_0 + h a_{21} k_1) \\
k_3 &= f(t_0 + c_3 h, x_0 + h(a_{31} k_1 + a_{32} k_2)) \\
&\cdots \\
k_s &= f(t_0 + c_s h, x_0 + h(a_{s1} k_1 + \cdots + a_{s,s-1} k_{s-1})) \\
x_1 &= x_0 + h(b_1 k_1 + \cdots + b_s k_s)
\end{aligned}
$$

Example

The **classical Runge-Kutta method** (order four):

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

It has an associated scheme:

$$\tilde{\varphi}(h;t,x) = x + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

with

$$\begin{aligned} k_1 &= f(t,x), \\ k_2 &= f(t + \frac{1}{2}h, x + \frac{1}{2}hk_1), \\ k_3 &= f(t + \frac{1}{2}h, x + \frac{1}{2}hk_2), \\ k_4 &= f(t + h, x + hk_3) \end{aligned}$$

Stability and implicit methods I

Let $\lambda \in \mathbb{R}$. The solution of Cauchy problem

$$\left.\begin{array}{rcl} \dot{x} & = & \lambda x \\ x(0) & = & x_0 \end{array}\right\}$$

is $x(t) = x_0 e^{\lambda t}$.

If $\lambda < 0$ the origin is **asymptotically stable**, in the sense that given any solution $x(t)$ of the differential equation, we have that $\lim_{t\to\infty} x(t) = 0$.

Definition

A numerical method is **stable** *for a step h and a constant $\lambda \in \mathbb{C}$ if $\tilde{x}_n$ is bounded for all $n \in \mathbb{N}$.*

Stability and implicit methods II

Example (The explicit Euler method 1)

$$\left.\begin{array}{rcl} t_{n+1} & = & t_n + h, \\ \tilde{x}_{n+1} & = & \tilde{x}_n + \lambda h \tilde{x}_n \end{array}\right\}$$

Then

$$\tilde{x}_n = (1 + \lambda h)^n x_0.$$

This means that the method is stable if $|1 + \lambda h| \leq 1$. *If* $\lambda < 0$ *then* $h \leq 2/|\lambda|$.

Stability and implicit methods III

Example (The explicit Euler method 2)

Consider the system of linear equations:

$$\begin{aligned} x' &= \alpha x + \beta y \\ y' &= \beta x + \alpha y \end{aligned}$$

with $x(0) = 2$ *and* $y(0) = 0$. *The solution of this system is*

$$\begin{aligned} x(t) &= e^{(\alpha+\beta)t} + e^{(\alpha-\beta)t} \\ y(t) &= e^{(\alpha+\beta)t} - e^{(\alpha-\beta)t}. \end{aligned}$$

If we use the Euler's method we obtain

$$\begin{aligned} x_{n+1} &= x_n + h(\alpha x_n + \beta y_n) \\ y_{n+1} &= y_n + h(\beta x_n + \alpha y_n) \end{aligned}$$

with $x_0 = 2$ *and* $y_0 = 0$.

Stability and implicit methods IV

The solutions of this difference equation are

$$\begin{aligned} x_n &= (1+\alpha h+\beta h)^n + (1+\alpha h-\beta h)^n \\ y_n &= (1+\alpha h+\beta h)^n - (1+\alpha h-\beta h)^n \end{aligned}$$

If $\alpha < \beta < 0$ the solution tends to zero when $t \to \infty$. For the Euler method we need

$$|1+\alpha h+\beta h| < 1, \qquad |1+\alpha h-\beta h| < 1 \Leftrightarrow 0 < h < -2/(\alpha+\beta).$$

Example: $\alpha = -20$ and $\beta = -19$. We have linear combinations of e^{-39t} (transient function) and e^{-t}.

Stability and implicit methods V

Comment (Stiff problems)

Consider the differential equation

$$\left.\begin{array}{rcl} \dot{x} & = & \lambda x \\ \dot{y} & = & f(x,y) \end{array}\right\}$$

If $\lambda << 0$, then $x(t)$ goes very fast to zero when $t \to \infty$. In other coordinates, this behaviour could not be evident.
Facts:

- *For the Euler method we will have to take very small steps $h < 2/|\lambda|$.*
- *Very common in the numerical solution of the partial differential equations.*
- *The explicit methods are not suitable for these kind of systems.*

Stability and implicit methods VI

Example (The implicit Euler method)

In this case, we obtain

$$\left.\begin{array}{rcl} t_{n+1} & = & t_n + h \\ \tilde{x}_{n+1} & = & \tilde{x}_n + h\lambda\tilde{x}_{n+1} \end{array}\right\}$$

Then

$$\tilde{x}_{n+1} = \frac{1}{1-h\lambda}\tilde{x}_n,$$

or

$$\tilde{x}_n = \frac{x_0}{(1-h\lambda)^n}.$$

When $\lambda < 0$, $1 - \lambda h > 1$, *and* $\lim_{n\to\infty} \tilde{x}_n = 0$, *and the method is stable.*

Implicit Runge-Kutta methods: First examples I

The solution $x(t)$ of the Cauchy problem

$$x' = f(t, x), \qquad x(t_0) = x_0,$$

satisfies

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(t, x(t))\, dt.$$

- If we replace the integral by f evaluated in an intermediate value multiplied by $h = t_1 - t_0$,, we obtain the θ-**method**:

$$x_1 = x_0 + hf(t_0 + \theta h, x_0 + \Theta(x_1 - x_0)),$$

with $0 \leq \theta, \Theta \leq 1$.
a) $\theta = \Theta = 0$, explicit Euler method
b) $\theta = \Theta = 1$, implicit Euler method.

Implicit Runge-Kutta methods: First examples II

- If we use $\theta = \Theta = \frac{1}{2}$, we get

$$x_1 = x_0 + hf\left(t_0 + \frac{h}{2}, x_0 + \frac{x_1 - x_0}{2}\right),$$

or, by setting $k_1 = (x_1 - x_0)/h$:

$$\begin{aligned} k_1 &= f\left(t_0 + \frac{h}{2}, x_0 + \frac{h}{2}k_1\right), \\ x_1 &= x_0 + hk_1. \end{aligned}$$

- Approximating the integral by the trapezoidal rule:

$$x_1 = x_0 + \frac{h}{2}(f(t_0, x_0) + f(t_1, x_1)).$$

Implicit Runge-Kutta methods: First examples III

- Using the Radau rule

$$x(t_1) - x(t_0) = \int_{t_0}^{t_0+h} f(t, x(t))\, dt \approx$$

$$\approx \frac{h}{4}\left(f(t_0, x_0) + 3\left(f\left(t_0 + \frac{2}{3}h, x\left(t_0 + \frac{2}{3}h\right)\right)\right)\right).$$

Then, we approximate $x(t_0 + 2h/3)$ by the quadratic interpolation based on x_0, x_0' and $x(t_1)$,

$$x\left(t_0 + \frac{2}{3}h\right) \approx \frac{5}{9}x_0 + \frac{4}{9}x(t_1) + \frac{2}{9}hf(t_0, x_0).$$

Implicit Runge-Kutta methods: First examples IV

Hammer-Hollingsworth method:

$$\begin{aligned} k_1 &= f(t_0, x_0), \\ k_2 &= f\left(t_0 + \frac{2}{3}h, x_0 + \frac{h}{3}(k_1 + k_2)\right), \\ x_1 &= x_0 + \frac{h}{4}(k_1 + 3k_2). \end{aligned}$$

Definition of Implicit Runge-Kutta method and applicability I

Definition

Let b_i, a_{ij} $(i,j = 1,\ldots,s)$ *be real numbers and let* c_i *such that* $c_i = \sum_{j=1}^{s} a_{ij}$. *The method*

$$\begin{aligned} k_i &= f\left(t_0 + c_i h, x_0 + h\sum_{j=1}^{s} a_{ij} k_j\right) \quad i = 1,\ldots,s \\ x_1 &= x_0 + h\sum_{i=1}^{s} b_i k_i \end{aligned} \tag{1}$$

is called an s-**stage Runge-Kutta method**. *When* $a_{ij} = 0$ *for* $i \leq j$ *we have an explicit (ERK) method. If* $a_{ij} = 0$ *for* $i < j$ *and at least one* $a_{ii} \neq 0$, *we have a* **diagonally implicit Runge-Kutta method** *(DIRK). If in addition all diagonal elements are identical, we speak of a* **singly diagonal** *implicit (SDIRK) method. In other cases we speak of an* **implicit** *Runge-Kutta method (IRK).*

Definition of Implicit Runge-Kutta method and applicability II

Example

As before, we use the Butcher tableau. For example, the implicit Euler method has reperesentation

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

the implicit midpoint rule

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

the Hammer-Hollingsworth method

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

Definition of Implicit Runge-Kutta method and applicability III

Theorem (Existence of a Numerical Solution)

Let $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be continuous and satisfy a Lipschitz condition with constant L (with respect to x). If

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

there exists a unique solution of (1), which can be obtain by iteration. If $f(t, x)$ is of class C^p, the function k_i (as a functions of h) are also in C^p.

Definition of Implicit Runge-Kutta method and applicability IV

Proof.

We define $K \in \mathbb{R}^{sn}$ as $K = (k_1, \ldots, k_s)^T$ and use the norm $\|K\| = \max_i(\|k_i\|)$. Then (1) can be written as $K = F(K)$ where

$$F_i(K) = f\left(t_0 + c_i h, x_0 + h\sum_{j=1}^{s} a_{ij} k_j\right), \qquad i = 1, \ldots, s.$$

Then

$$\|F(K_1) - F(K_2)\| \leq hL \max_{i=1,\ldots,s} \sum_{j=1}^{s} |a_{ij}| \cdot \ \|K_1 - K_2\|,$$

By the contraction mapping theorem $\exists!$ solution and the iterates $K^{(i+1)} = F(K^{(i)})$ converge.
Differentiability by the IFT: Define $\Phi(h, K) = K - F(K) = 0$. Then $\Phi(0, f(t_0, x_0), \ldots, f(t_0, x_0)) = 0$ and $D_K\Phi(0, K) = I$. □

Comment

In order to use these methods, we will use a Newton method to obtain the approximations.

Convergence of one-step methods I

We want to solve the Cauchy problem

$$x' = f(t, x), \qquad x(t_0) = x_0,$$

Assumptions

- All the partial derivatives exist on the strip $S = \{(t, x) | \, a \leq t \leq b, \, x \in \mathbb{R}\}$
- The derivatives are continouos and bounded.
- We have a one-step method:

$$\left.\begin{array}{rcl} t_{n+1} & = & t_n + h_n, \\ \tilde{x}_{n+1} & = & \tilde{\varphi}(h_n; t_n, \tilde{x}_n) \end{array}\right\}.$$

We recall

Convergence of one-step methods II

- Local truncation error:

$$\tilde{x}_n - \Phi(t_n; t_{n-1}, \tilde{x}_{n-1}),$$

- Global truncation error

$$\tilde{x}_n - \Phi(t_n; t_0, x_0).$$

Goal

To prove that if the local error of the method is $O(h^{p+1})$ then the global error is $O(h^p)$.

Consider for $t_0 \in [a, b]$, $x_0 \in \mathbb{R}$, the initial-value problem

$$x' = f(t, x), \qquad x(t_0) = x_0,$$

having the exact solution $x(t)$.

Convergence of one-step methods III

Theorem

Let the function $\tilde{\varphi}$ be continuous on

$$G := \{(t, x, h) | a \leq t \leq b,\ |x - x(t)| \leq \gamma,\ 0 \leq |h| \leq h_0\}, \quad h_0 > 0, \gamma > 0,$$

and let there exist positive constants M and N such that

$$|\tilde{\varphi}(h; t, x_1) - \tilde{\varphi}(h; t, x_2)| \leq (1 + M|h|)\,|x_1 - x_2|, \quad \forall\,(h, t, x_i) \in G, \quad i = 1, 2,$$

$$|\tilde{\varphi}(h; t, x(t)) - \varphi(h; t, x(t))| \leq N|h|^{p+1},\ (p > 0), \quad \forall\, t \in [a, b],\ |h| \leq h_0.$$

Then there exists an $\bar{h}$, $0 < \bar{h} \leq h_0$, such that

$$|\tilde{x}_n - x(t)| \leq |h_n|^p N \frac{e^{M|t - t_0|} - 1}{M},$$

for all $t \in [a, b]$ and all $h_n = (t - t_0)/n$, $n = 1, 2, \ldots$, with $|h_n| \leq \bar{h}$. If $\gamma = \infty$ then $\bar{h} = h_0$.

Proof of the Theorem I

First we need the following lemma:

Lemma

If the numbers ξ_i satisfy estimates of the form

$$|\xi_{i+1}| \leq (1+\delta)|\xi_i| + B, \qquad \delta > 0, \quad B \geq 0, \quad i = 0, 1, 2, \ldots,$$

then

$$|\xi_n| \leq e^{n\delta}|\xi_0| + \frac{e^{n\delta} - 1}{\delta} B.$$

Proof of the Theorem II

Proof.

From the assumptions we get

$$|\xi_1| \leq (1+\delta)|\xi_0| + B,$$

$$|\xi_2| \leq (1+\delta)^2|\xi_0| + B(1+\delta) + B,$$

$$\vdots$$

$$|\xi_n| \leq (1+\delta)^n|\xi_0| + B[1 + (1+\delta) + (1+\delta)^2 + \cdots (1+\delta)^{n-1}] \leq$$

$$\leq (1+\delta)^n|\xi_0| + B\frac{(1+\delta)^n - 1}{\delta} \leq$$

$$\leq e^{n\delta}|\xi_0| + B\frac{e^{n\delta} - 1}{\delta},$$

since $0 < 1 + \delta \leq e^{\delta}$ for $\delta > 0$. □

Proof of the Theorem III

Continuing with the proof of the theorem, the function

$$\bar{\varphi}(h;t,x) = \begin{cases} \tilde{\varphi}(h;t,x) & \text{if} \quad (h,t,x) \in G \\ \tilde{\varphi}(h;t,x(t)+\gamma) & \text{if} \quad t \in [a,b],\ |h| \leq h_0,\ x \geq x(t)+\gamma \\ \tilde{\varphi}(h;t,x(t)-\gamma) & \text{if} \quad t \in [a,b],\ |h| \leq h_0,\ x \leq x(t)-\gamma \end{cases}$$

is evidently continuous on

$$\tilde{G} := \{(h,t,x) |\ t \in [a,b], x \in \mathbb{R},\ |h| \leq h_0\}$$

and satisfies the condition

$$|\bar{\varphi}(h;t,x_1) - \bar{\varphi}(h;t,x_2)| \leq (1 + M|h|)|x_1 - x_2|,$$

for all $(h,t,x) \in \tilde{G}$, $i = 1,2$, and, as $\bar{\varphi}(h;t,x(t)) = \tilde{\varphi}(h;t,x(t))$, also the condition

$$|\bar{\varphi}(h;t,x(t)) - \varphi(h;t,x(t))| \leq N|h|^{p+1}\ (p > 0) \quad \forall\, t \in [a,b],\ |h| \leq h_0.$$

Proof of the Theorem IV

Now we consider the one-step method generated by $\bar{\varphi}$:

$$\left.\begin{array}{rcl} t_{i+1} & = & t_i + h, \\ \bar{x}_{i+1} & = & \bar{\varphi}(h; t_i, \bar{x}_i) \end{array}\right\}.$$

In view of

$$x_{i+1} = \varphi(h; t_i, x_i),$$

on obtains for the error $\bar{e}_i = \bar{x}_i - x_i$,

$$\bar{e}_{i+1} = \bar{\varphi}(h; t_i, \bar{x}_i) - \bar{\varphi}(h; t_i, x_i) + \bar{\varphi}(h; t_i, x_i) - \varphi(h; t_i, x_i).$$

Now, from the conditions it follows that

$$|\bar{\varphi}(h; t_i, \bar{x}_i) - \bar{\varphi}(h; t_i, x_i)| \leq (1 + |h|M)|\bar{e}_i|,$$

$$|\bar{\varphi}(h; t_i, x_i) - \varphi(h; t_i, x_i)| \leq N|h|^{p+1},$$

Proof of the Theorem V

and hence we have the recursive estimate

$$|\bar{e}_{i+1}| \leq (1 + |h|M)|\bar{e}_i| + N|h|^{p+1}.$$

The previous lemma, since $\bar{e}_0 = x_0 - x_0 = 0$, gives

$$|\bar{e}_k| \leq N|h|^p \frac{e^{k|h|M} - 1}{M}.$$

If $t \in [a, b]$, $t \neq t_0$, and $h := h_n = (t - t_0)/n$, then $t_n = t_0 + nh = t$, and

$$|\bar{e}_n| \leq N|h_n|^p \frac{e^{n|h_n|M} - 1}{M} = N|h_n|^p \frac{e^{(t-t_0)M} - 1}{M} \leq N|h_n|^p \frac{e^{(b-a)M} - 1}{M},$$

for $|h_n| \leq h_0$.

Proof of the Theorem VI

Then there exists an $\bar{h}$, $0 < \bar{h} \leq h_0$, such that $|\bar{e}_n| \leq \gamma$ for all $t \in [a, b]$, $|h_n| \leq \bar{h}$, i.e., for the one-step method generated by $\tilde{\varphi}$ we have for $|h| \leq \bar{h}$, according to the definition of $\bar{\varphi}$,

$$\bar{x}_i = \tilde{x}_i, \qquad \bar{e}_i = \tilde{e}_i, \qquad \text{and} \qquad \bar{\varphi}(h; t_i, \bar{x}_i) = \tilde{\varphi}(h; t_i, \tilde{x}_i).$$

The assertion of the theorem,

$$|\tilde{e}_n| \leq |h_n|^p N \frac{e^{M|t-t_0|} - 1}{M},$$

thus follows for all $t \in [a, b]$ and all $h_n = (t - t_0)/n$, $n = 1, 2, \ldots$, with $|h_n| \leq \bar{h}$.

Comment

Note that if $\tilde{\varphi}(h; t, x) = x + h\Gamma(h; t, x)$ *and* $|\Gamma(h; t, x_1 - \Gamma(h; t, x_2)| \leq M|x_1 - x_2|$ *then* $|\tilde{\varphi}(h; t, x_1) - \tilde{\varphi}(h; t, x_2)| \leq (1 + |h|M)|x_1 - x_2|$. *In particular, all the Runge-Kutta methods are of this type.*

The influence of Rounding Errors in one-step methods I

Suppose

- $x_i = x(t_i)$ values of the exact solution of the initial value problem,
- $\tilde{x}_i$ the discrete solutions produced by the one-step method in exact arithmetic, and
- $\bar{x}_i$ the approximate values of $\tilde{x}_i$ obtained in floating-point arithmetic.

Moreover,

- $\bar{x}_0 = x_0$, and

$$t_{i+1} = t_i + h, \qquad \bar{x}_{i+1} = \tilde{\varphi}(h; t_i, \bar{x}_i) + \epsilon_{i+1},$$

  where $|\epsilon_{i+1}| \leq \epsilon$ for all $i \geq 0$.
- $|\tilde{\varphi}(h; t, x_1) - \tilde{\varphi}(h; t; x_2)| \leq (1 + |h|M)|x_1 - x_2|$.

The influence of Rounding Errors in one-step methods II

Then, for the error $r_i = \bar{x}_i - \tilde{x}_i$, there follows by substraction of the previous formula from the present one

$$r_{i+1} = \tilde{\varphi}(h, t_i, \bar{x}_i) - \tilde{\varphi}(h; t_i, \tilde{x}_i) + \epsilon_{i+1},$$

and thus

$$|r_{i+1}| \leq (1 + |h|M)|r_i| + \epsilon.$$

Since $r_0 = 0$, the previous lemma gives

$$|r_n| \leq \frac{\epsilon}{|h|} \frac{e^{M|x-x_0|} - 1}{M}$$

for all $t \in [a, b]$ and $h = h_n = (x - x_0)/n$, $n = 1, 2, \ldots$. Then

The influence of Rounding Errors in one-step methods III

For a method of order p the total error

$$v_i := \bar{x}_i - x_i = r_i + e_i,$$

under the assumptions of the previous theorem, obeys the estimate

$$|v_n| \leq \left[N|h|^p + \frac{\epsilon}{|h|} \right] \frac{e^M |x - x_0| - 1}{M}$$

for all $t \in [a, b]$ and for all sufficiently small $h := h_n = (x - x_0)/n$.

Comment

This formula reveals that, on account of the influence of rounding errors, the total error begins to increase again, once h is reduced beyond a certain critical value.

Stability of explicit Runge-Kutta Methods I

We apply an s-stage explicit Runge-Kutta method to the Cauchy problem $x' = \lambda x$, $x(0) = x_0$:

$$\begin{aligned}
k_1 &= \lambda x_0, \\
k_2 &= \lambda x_0 + h a_{21} \lambda^2 x_0 = \lambda P_1(\lambda h) x_0, \\
k_3 &= \lambda(x_0 + h(a_{31}\lambda x_0 + a_{32}(\lambda x_0 + h a_{21}\lambda^2 x_0))) = \lambda P_2(\lambda h) x_0, \\
&\dots \\
k_s &= \lambda P_{s-1}(\lambda h), \\
x_1 &= x_0 + h\lambda P_{s-1}(\lambda h) x_0 = P_s(\lambda h) x_0
\end{aligned}$$

where $P_i(z)$ is a polynomial of degree $\leq i$.

Stability of explicit Runge-Kutta Methods II

Definition

The function $R(z) = P_s(z)$ *is called the* **stability function** *of the method. It can be interpreted as the numerical solution after one step for*

$$x' = \lambda x, \qquad x_0 = 1, \qquad z = \lambda h.$$

The set

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

is called the **stability domain** *of the method.*

Comment

This definition is extended to a general Runge-Kutta method.

Stability of explicit Runge-Kutta Methods III

Theorem

If the explicit Runge-Kutta method is of order p, then

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

Proof:

The exact solution of $x' = \lambda x$, $x_0 = 1$ is e^z and therefore the numerical solution must satisfy

$$e^z - R(z) = O(h^{p+1}) = O(z^{p+1}).$$

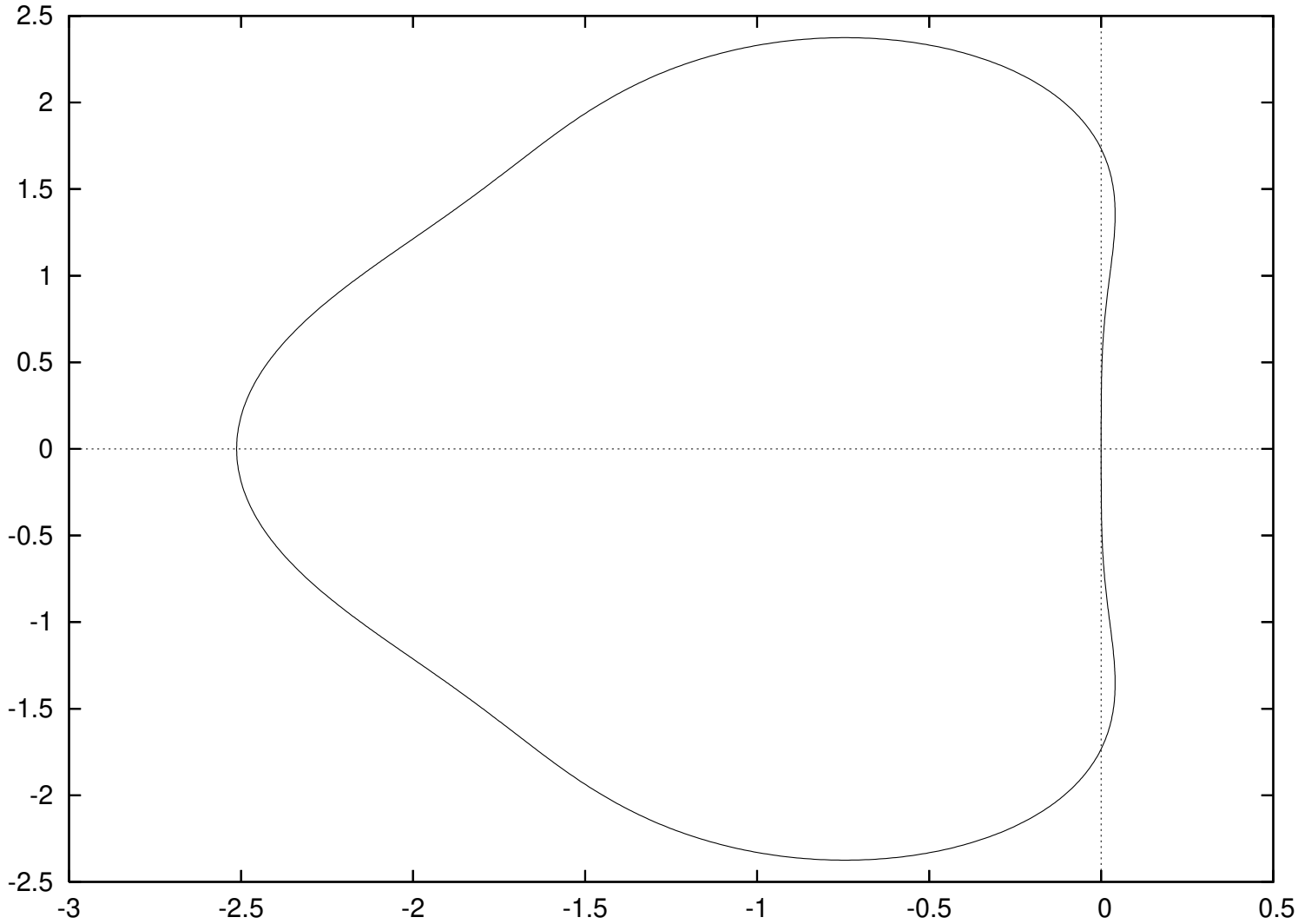
□

As a consequence, all explicit Runge-Kutta methods with $p = s$ posses the stability function

$$R(z) = 1 + z + \frac{z^2}{2!} + \cdots + \frac{z^s}{s!}.$$

Example

The stability domain of a Runge-Kutta method of 3 stages and order 3:



Step size control: Embedded Runge-Kutta Formulas I

- Question: How to estimate the error?
- Possible answer: Construct Runge-Kutta formulas which themselves contains, besides the numerical approximation $\tilde{x}_1$, a second (better) aproximation $\hat{x}_1$.
- Possible problem: Expensive?
- Solution: Consider two Runge-Kutta methods (one for $\tilde{x}_1$ and one for $\hat{x}_1$) such that both use the *same* function values.

Consider the scheme of coefficients

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s \\
\hline
 & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s
\end{array}
$$

Step size control: Embedded Runge-Kutta Formulas II

such that

$$\tilde{x}_{i+1} = \tilde{x}_i + h(b_1 k_1 + \cdots b_s k_s)$$

is of order p and

$$\hat{x}_{i+1} = \hat{x}_i + h(\hat{b}_1 k_1 + \cdots \hat{b}_s k_s)$$

is of order $\hat{p}$ (usually $\hat{p} = p - 1$ or $\hat{p} = p + 1$). The approximation $\tilde{x}_i$ is used to continue the integration.
One of the most famous methods is the **Runge-Kutta-Fehlberg (RKF) family of methods**.

A method "name $p(\tilde{p})$" means that the order of $\tilde{x}_i$ is p and the order of $\hat{x}_i$ is $\tilde{p}$. For example, we have RKF 4(5) and RKF 7(8).

The Runge-Kutta-Fehlberg 4(5) formula I

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	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
$\tilde{x}_1$	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
$\hat{x}_1$	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

The Runge-Kutta-Fehlberg 4(5) formula II

$$
\begin{aligned}
k_1 &= f(t_j, x_j), \\
k_2 &= f(t_j + \tfrac{1}{4}h, x_j + h(\tfrac{1}{4}k_1)), \\
k_3 &= f(t_j + \tfrac{3}{8}h, x_j + h(\tfrac{3}{32}k_1 + \tfrac{9}{32}k_2)), \\
k_4 &= f(t_j + \tfrac{12}{13}h, x_j + h(\tfrac{1932}{2197}k_1 - \tfrac{7200}{2197}k_2 + \tfrac{7296}{2197}k_3)), \\
k_5 &= f(t_j + h, x_j + h(\tfrac{439}{216}k_1 - 8k_2 + \tfrac{3680}{513}k_3 - \tfrac{845}{4104}k_4)), \\
k_6 &= f(t_j + \tfrac{1}{2}h, x_j + h(-\tfrac{8}{27}k_1 + 2k_2 - \tfrac{3544}{2565}k_3 + \tfrac{1859}{4104}k_4 - \tfrac{11}{40}k_5)).
\end{aligned}
$$

Using these constants, we approximate the solution by using a RK4 formula:

$$\tilde{x}_{j+1} = \tilde{x}_j + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right),$$

and determine another approximation by using a RK5:

$$\hat{x}_{j+1} = \tilde{x}_j + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right).$$

Error estimation

Substracting one formula from the other:

$$\|\hat{x}_{n+1} - \tilde{x}_{n+1}\|_2 = |h| \left\| \frac{1}{360} k_1 - \frac{128}{4275} k_3 - \frac{2197}{75240} k_4 + \frac{1}{50} k_5 + \frac{2}{55} k_6 \right\|_2 .$$

Moreover, if $\|\hat{x}_n - \tilde{x}_n\|_2 \leq \epsilon$, and

$$\tilde{x}_n = \tilde{\varphi}(h_{n-1}; t_{n-1}, \tilde{x}_{n-1}),$$
$$\hat{x}_n = \hat{\varphi}(h_{n-1}; t_{n-1}, \tilde{x}_{n-1})$$

then

$$\tilde{\varphi}(h_{n-1}; t_{n-1}, \tilde{x}_{n-1}) - \varphi(h_{n-1}; t_{n-1}, \tilde{x}_{n-1}) = N_4(t_{n-1}) h_{n-1}^5 + O(h_{n-1}^6)$$
$$\hat{\varphi}(h_{n-1}; t_{n-1}, \tilde{x}_{n-1}) - \varphi(h_{n-1}; t_{n-1}, \tilde{x}_{n-1}) = N_5(t_{n-1}) h_{n-1}^6 + O(h_{n-1}^7),$$

that is,

$$\hat{x}_n - \tilde{x}_n = N_4(t_{n-1}) h_{n-1}^5 + O(h_{n-1}^6).$$

Finally, if $|h_{n-1}|$ is small,

$$\|N_4(t_{n-1})\| \approx \frac{\|\hat{x}_n - \tilde{x}_n\|}{h_{n-1}^5}. \tag{2}$$

Automatic step size control I

We have:

- $\hat{x}_n$ and $\tilde{x}_n$, s.t. $\|\hat{x}_n - \tilde{x}_n\|_2 \leq \epsilon_n$.
- A predicted step h_n.

We want:

- $\hat{x}_{n+1}$ and $\tilde{x}_{n+1}$, s.t. $\|\hat{x}_{n+1} - \tilde{x}_{n+1}\|_2 \leq \epsilon_{n+1}$.
- A new predicted step h_{n+1}.

Algorithm

1. Compute $\hat{x}_{n+1}$ and $\tilde{x}_{n+1}$ with step h_n and initial condition $\tilde{x}_n$.
2. Compute
$$h_N = 0.9 h_n \sqrt[5]{\frac{\epsilon_{n+1}}{\|\hat{x}_{n+1} - \tilde{x}_{n+1}\|_2}}$$
3. If $\|\hat{x}_{n+1} - \tilde{x}_{n+1}\|_2 \leq \epsilon_{n+1}$, then take $h_{n+1} = h_N$.
4. Else, redefine $h_n = h_N$ and go to step 1).

Automatic step size control II

Comment (The formula for h_N)

We have

$$\left.\begin{array}{rcl} \|\hat{x}_{n+1} - \tilde{x}_{n+1}\| & = & N_4(t_n)h_N^5 + O(h_N^6), \\ N_4(t_n) & = & N_4(t_{n-1} + h_N) = N_4(t_{n-1}) + O(h_N) \end{array}\right\}$$

Hence,

$$\left.\begin{array}{rcl} \|\hat{x}_{n+1} - \tilde{x}_{n+1}\| & = & N_4(t_{n-1})h_N^5 + O(h_N^6), \\ N_4(t_{n-1}) & \approx & \frac{\|\hat{x}_n - \tilde{x}_n\|}{h_{n-1}^5} \end{array}\right\}$$

Negleting the terms of order $O(h_N^6)$ (assuming that h_N and h_{n-1} are comparable), to have $\|\hat{x}_{n+1} - \tilde{x}_{n+1}\| \lesssim \epsilon_{n+1}$ it is enough to impose

$$h_N^5 \lesssim \frac{\epsilon_{n+1}}{N_4(t_{n-1})} \approx \frac{h_{n-1}^5 \epsilon_{n+1}}{\|\hat{x}_n - \tilde{x}_n\|}.$$

Then, we take $h_N = 0.9 \cdot h_{n-1} \sqrt[5]{\frac{\epsilon_{n+1}}{\|\hat{x}_n - \tilde{x}_n\|}}$.

Implementation of RKF4(5) I

```
int rkf45(double *at, double *x, int n, double *ah, int sc,
          double tol, double *atf, double *aer,
          void (*ode)(double,double*,int,double*))
```

where

- `at`: Time. input: time corresponding to the present initial condition. output: new value corresponding to the new initial condition.
- `x`: Position. input: present initial condition. output: new position at time `*at`.
- `n`: dimension of the system of ode's.
- `ah`: time step (it can be modified by the routine according to the given threshold). If negative, the integration goes backwards. On exit, it will contain the time step for the next rkf78 call
- `sc`: stepsize control.
  - 0: no stepsize control, the step `*ah` is used
  - $\neq 0$: stepsize control according to the threshold tol
- `tol`: threshold to control the integration error
- `atf`: final integration time. if NULL, it is ignored. Otherwise, if the stepsize is too large (`*at+*ah>*atf`), it is reduced so that the new time at is exactly atf (in that case, the function returns 1)

Implementation of RKF4(5) II

```
int rkf45(double *at, double *x, int n, double *ah, int sc,
          double tol, double *atf, double *aer,
          void (*ode)(double,double*,int,double*))
```

`aer`: if NULL, the routine stops if the estimated error is larger than tol. if not NULL, the integration returns the estimated absolute error of the performed step. This allows, for instance, to integrate with a constant stepsize (`sc`$= 0$) and to know an estimate of the error.

`ode`: pointer to the the vectorfield. The parameters of the function are: `(t,x,n,f)`, where `t` is the time, `x` the position vector, `n` the dimension and `f` the value of the vectorfield at `(t,x)`

`return value`: 0: ok. 1: ok and `at=tf`.