Numerical Methods
in Engineering and Applied Science

Lecture 8. Multistep methods.

The *model problem* that we consider here is the following **initial value problem**: *For a function* $\boldsymbol{f} : [0,T]\times\mathbb{R}^N \to \mathbb{R}^N$ *and* $\boldsymbol{u}_0 \in \mathbb{R}^N$ *find a differentiable function* $\boldsymbol{u}(t)$ *such that*

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{u}(t)), \\ \boldsymbol{u}(0) = \boldsymbol{u}_0. \end{cases} \tag{1}$$

Most commonly it describes the **time evolution** of some quantity. The numerical methods for it are called **time stepping** or **time marching** methods. These methods evaluate the numerical solution $\boldsymbol{u}^{n+1}$ at a time instant t_{n+1} using the information from previous time instants t_n, t_{n-1} etc.
We consider three groups of such methods:

- **Taylor series methods**;
- **Multistage (Runge-Kutta) methods**;
- **Multistep methods**.

Multistep methods.

A linear s-step method is a formula for calculating the value at a new time step u_{n+1} using previous time step values u_{n-s+1}, ..., u_n and $f(t_{n-s+1}, u_{n-s+1})$, ..., $f(t_n, u_n)$, where $s \geq 1$.

The main idea consists in minimizing the number of evaluations of the right-hand side $f(t, u)$ at each step, because calculation of $f(t, u)$ may be computationally expensive.

Some of the multi-stage methods can be considered as one-step methods, for example

- the Euler method (order 1),

$$u_{n+1} = u_n + hf(t_n, u_n). \tag{2}$$

- the trapezoidal rule (order 2)

$$u_{n+1} = u_n + h\frac{f(t_n, u_n) + f(t_{n+1}, u_{n+1})}{2}. \tag{3}$$

  This scheme is *implicit*, therefore, iteration or a first-order prediction is required to estimate u_{n+1} in the right-hand side.

- yet another one-step method is the *backward Euler* method (order 1),

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}). \tag{4}$$

For simplicity, let us write f_j instead of $f(t_j, u_j)$.
Below are some examples of multistep methods of order 2 or higher.

- leap-frog scheme (explicit, order 2)

$$u_{n+1} = u_{n-1} + 2hf_n \tag{5}$$

- fourth-order Adams–Bashforth (AB4) scheme. AB schemes are explicit.

$$u_{n+1} = u_n + \frac{h}{24}\left(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}\right) \tag{6}$$

- fourth-order Adams–Moulton (AM4) scheme. AM schemes are implicit.

$$u_{n+1} = u_n + \frac{h}{24}\left(9 f_{n+1} + 19 f_n - 5 f_{n-1} + f_{n-2}\right) \tag{7}$$

- third order backward differentiation formula (BDF3). BDF schemes are implicit.

$$u_{n+1} = \frac{18}{11} u_n - \frac{9}{11} u_{n-1} + \frac{2}{11} u_{n-2} + \frac{6}{11} h f_{n+1} \tag{8}$$

There is a subtlety about performing the first iterations using multistep methods. For example, suppose we want to use the leapfrog formula. We know the value of u_0, but it is not sufficient for evaluating u_1. Where can we find u_{-1} required for u_1? Otherwise, if we start from u_2, how to calculate u_1? There are several practical solutions to this problem.

- Start up using the explicit Euler method with a sufficiently small h.
- Start up a multistep scheme of order p using a Runge–Kutta method of order $p-1$ to calculate $u_1, u_2, ..., u_s$.
- Use analytical formulate for the solution in the neighborhood of the initial condition.

The general form of s-step methods is

$$\sum_{j=0}^{s} \alpha_j u_{n+1+j-s} = h \sum_{j=0}^{s} \beta_j f_{n+1+j-s}, \tag{9}$$

where the coefficients α_j et β_j are constant, $\alpha_s = 1$ (the coefficient in front of u_{n+1}), and $\alpha_0 \neq 0$ and/or $\beta_0 \neq 0$ (coefficients in front of u_{n+1-s} and f_{n+1-s}, respectively.). If $\beta_s = 0$, the method is *explicit*. If $\beta_s \neq 0$, the method is *implicit* (β_s is the coefficient in front of f_{n+1}).
Let us define the *linear difference operator* $\mathcal{L}_h$ of an s-step scheme as

$$\begin{aligned} \mathcal{L}_h z(t) = z(t+sh) + \alpha_{s-1} z(t+(s-1)h) + ... + \alpha_0 z(t) \\ -h\left(\beta_s z'(t+sh) + \beta_{s-1} z'(t+(s-1)h) + ... + \beta_0 z'(t)\right), \end{aligned} \tag{10}$$

where $z(t)$ is an arbitrary differentiable function ($u(t)$ is a special case of $z(t)$). $\mathcal{L}_h$ is a linear operator because

$$\mathcal{L}_h(az(t) + bw(t)) = a\mathcal{L}_h z(t) + b\mathcal{L}_h w(t).$$

Definition. The operator $\mathcal{L}_h$ and the corresponding linear multistep method are *consistent of order* p if

$$\mathcal{L}_h z(t) = \mathcal{O}(h^{p+1}) \tag{11}$$

with $p > 0$ for any sufficiently smooth function z.
Example. The linear difference operator of the Euler method is

$$\mathcal{L}_h z(t) = z(t+h) - z(t) - hz'(t),$$

which gives, by Taylor series expansion,

$$\mathcal{L}_h z(t) = \frac{1}{2}h^2 z''(t) + \mathcal{O}(h^3),$$

hence $\mathcal{L}_h z(t) = \mathcal{O}(h^2)$ and the method is consistent of order 1.

Example. The linear difference operator of the BDF3 scheme:

$$\mathcal{L}_h z(t) = z(t+3h) - \frac{18}{11}z(t+2h) + \frac{9}{11}z(t+h) - \frac{2}{11}z(t) - \frac{6}{11}hz'(t+3h).$$

Its Taylor series expansion:

$$\begin{aligned}
z(t+3h) &= z(t) + 3hz'(t) + \tfrac{9}{2}h^2z''(t) + \tfrac{9}{2}h^3z'''(t) + \tfrac{27}{8}h^4z''''(t) + \mathcal{O}(h^5),\\
z(t+2h) &= z(t) + 2hz'(t) + 2h^2z''(t) + \tfrac{4}{3}h^3z'''(t) + \tfrac{2}{3}h^4z''''(t) + \mathcal{O}(h^5),\\
z(t+h) &= z(t) + hz'(t) + \tfrac{1}{2}h^2z''(t) + \tfrac{1}{6}h^3z'''(t) + \tfrac{1}{24}h^4z''''(t) + \mathcal{O}(h^5),\\
z'(t+3h) &= \qquad\qquad z'(t) + 3hz''(t) + \tfrac{9}{2}h^2z'''(t) + \tfrac{9}{2}h^3z''''(t) + \mathcal{O}(h^4).
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_h z(t) &= \left(1 - \tfrac{18}{11} + \tfrac{9}{11} - \tfrac{2}{11}\right) z(t) + h\left(3 - 2\cdot\tfrac{18}{11} + \tfrac{9}{11} - \tfrac{6}{11}\right) z'(t)\\
&\quad + h^2\left(\tfrac{9}{2} - 2\cdot\tfrac{18}{11} + \tfrac{1}{2}\cdot\tfrac{9}{11} - 3\cdot\tfrac{6}{11}\right) z''(t)\\
&\quad + h^3\left(\tfrac{9}{2} - \tfrac{4}{3}\cdot\tfrac{18}{11} + \tfrac{1}{6}\cdot\tfrac{9}{11} - \tfrac{9}{2}\cdot\tfrac{6}{11}\right) z'''(t)\\
&\quad + h^4\left(\tfrac{27}{8} - \tfrac{2}{3}\cdot\tfrac{18}{11} + \tfrac{1}{24}\cdot\tfrac{9}{11} - \tfrac{9}{2}\cdot\tfrac{6}{11}\right) z''''(t) + \mathcal{O}(h^5)\\
&= -\tfrac{3}{22}h^4z''''(t) + \mathcal{O}(h^5) = \mathcal{O}(h^4).
\end{aligned}$$

Conclusion: the method is consistent of order 3.

Same approach to determine the coefficients of any method.

$$\mathcal{L}_h z(t) = \sum_{j=0}^{s} \alpha_j z(t+jh) - h \sum_{j=0}^{s} \beta_j z'(t+jh) \tag{12}$$

Taylor series:

$$\begin{aligned} z(t+jh) &= z(t) + jhz'(t) + \tfrac{1}{2}(jh)^2 z''(t) + ..., \\ z'(t+jh) &= z'(t) + jhz''(t) + \tfrac{1}{2}(jh)^2 z'''(t) + ... \end{aligned} \tag{13}$$

Substitute (13) in (12) to obtain

$$\begin{aligned} &\mathcal{L}_h z(t) = C_0 z(t) + C_1 h z'(t) + C_2 h^2 z''(t) + ..., \quad \text{où} \\ &C_0 = \alpha_0 + ... + \alpha_s, \\ &C_1 = (\alpha_1 + 2\alpha_2 + ... + s\alpha_s) - (\beta_0 + ... + \beta_s), \\ &C_2 = \tfrac{1}{2}(\alpha_1 + 4\alpha_2 + ... + s^2\alpha_s) - (\beta_1 + 2\beta_2 + ... + s\beta_s), \\ &\vdots \\ &C_m = \textstyle\sum_{j=0}^{s} \frac{j^m}{m!}\alpha_j - \sum_{j=0}^{s} \frac{j^{m-1}}{(m-1)!}\beta_j \end{aligned} \tag{14}$$

We conclude that a method is consistent of order p if $C_0 = C_1 = ... = C_p = 0$ et $C_{p+1} \neq 0$. This condition defines a system of $p + 1$ linear equations with $2s + 1$ unknowns at most: α_0,...,α_{s-1} and β_0,...,β_s (remember that α_s=1). One may think it could be possible to construct an s-step method of order $2s$. In reality, it turns out that such a method is not convergent and, therefore, useless. We will see that a convergent s-step method can reach the convergence order $s + 2$ at the maximum (Dahlquist barrier). For this reason, in the most used schemes, a large part of the coefficients α_j and β_j are equal to zero. For example, Adams–Bashforth and Adams–Moulton s-step schemes have order s. Their general form is

$$u_{n+1} = u_n + h \sum_{j=0}^{s} \beta_j f_{n+1+j-s}. \tag{15}$$

The coefficients α are $\alpha_s = 1$, $\alpha_{s-1} = -1$ and $\alpha_{s-2} = ... = \alpha_0 = 0$. The coefficients $\beta_0 ... \beta_{s-1}$ are non-zero. $\beta_s = 0$ in the Adams–Bashforth schemes and $\beta_s \neq 0$ in the Adams–Moulton schemes.
There exist an alternative approach to determine the coefficients β_j of these schemes. Function values f_{n+1-s}, ..., f_{n+1} are treated as $F(t) = f(t, u(t))$ ($u(t)$ being the exact solution) evaluated at t_{n+1-s}, ..., t_{n+1}. Then we rewrite the differential equation as

$$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} F(t)\mathrm{d}t. \tag{16}$$

Let $q(t)$ be a polynomial of degree $s-1$ (AB) or s (AM) that interpolates the data points $\{f_j\}$. Then we can write the AB and AM schemes as

$$u_{n+1} - u_n = \int_{t_n}^{t_{n+1}} q(t)\mathrm{d}t. \tag{17}$$

As $q(t)$ is a linear combination of the $\{f_j\}$, we can take them out of the integral and calculate the weights – coefficients of the numerical scheme.

Example. Calculation of the second-order Adams–Bashforth (AB2) coefficients. We have $s = 2$, we need to determine the coefficients β_0 and β_1. We use the values of f_{n-1} and f_n to construct a linear polynomial,

$$q(t) = f_n - \frac{f_n - f_{n-1}}{h}(t_n - t). \tag{18}$$

Using (17), we obtain

$$\begin{aligned} u_{n+1} - u_n &= \int_{t_n}^{t_{n+1}} \left[f_n - \tfrac{f_n - f_{n-1}}{h}(t_n - t) \right] \mathrm{d}t \\ &= hf_n - \tfrac{f_n - f_{n-1}}{h} \int_{t_n}^{t_{n+1}} (t_n - t)\mathrm{d}t \\ &= hf_n - \tfrac{f_n - f_{n-1}}{h} \left(-\tfrac{h^2}{2} \right) \\ &= \tfrac{3}{2}hf_n - \tfrac{1}{2}hf_{n-1}. \end{aligned} \tag{19}$$

We derived the AB2 scheme:

$$u_{n+1} = u_n + h \left(\tfrac{3}{2}f_n - \tfrac{1}{2}f_{n-1} \right). \tag{20}$$

One can use the Newtonian interpolation to calculate the coefficients of the AB and AM schemes of any order. We introduce the backward difference operator,

$$\nabla z(t) = z(t) - z(t-h). \tag{21}$$

Applied to discrete data, (21) yields

$$\nabla z_n = z_n - z_{n-1}. \tag{22}$$

We will use the powers of this operator, ∇^j. For example,

$$\begin{aligned} \nabla^2 z_n &= \nabla(\nabla z_n) = \nabla(z_n - z_{n-1}) \\ &= (z_n - z_{n-1}) - (z_{n-1} - z_{n-2}) = z_n - 2z_{n-1} + z_{n-2}. \end{aligned} \tag{23}$$

- For any $s \geq 1$, the s-step Adams–Bashforth scheme is consistent of order s. It can be written in the following form:

$$u_{n+1} = u_n + h \sum_{j=0}^{s-1} \gamma_j \nabla^j f_n. \tag{24}$$

The coefficients γ_j, $j \geq 0$, are calculated using the recurrent relation

$$\gamma_j + \frac{1}{2}\gamma_{j-1} + \frac{1}{3}\gamma_{j-2} + ... + \frac{1}{j+1}\gamma_0 = 1. \tag{25}$$

- For all $s \geq 0$, the Adams–Moulton scheme with s steps (with 1 step if $s = 0$) is consistent of order $s + 1$.

$$u_{n+1} = u_n + h \sum_{j=0}^{s} \gamma_j^* \nabla^j f_{n+1}. \tag{26}$$

The coefficients γ_j, $j \geq 1$, are calculated using the relation

$$\gamma_j^* + \frac{1}{2}\gamma_{j-1}^* + \frac{1}{3}\gamma_{j-2}^* + ... + \frac{1}{j+1}\gamma_0^* = 0 \tag{27}$$

with $\gamma_0^* = 1$. The case $s = 0$ corresponds to the backward Euler scheme.

- For any $s \geq 1$, the s steps backward differentiation formula (BDF) is consistent of order s.

$$\sum_{j=1}^{s} \frac{1}{j} \nabla^j v_{n+1} = h f_{n+1}. \tag{28}$$

We must divide (28) by the coefficient in front of $u_n + 1$ to write this scheme in its classical form.

Using the same procedure, we can construct adaptive schemes. For example, an AB2 method with variable time step:

$$u_{n+1} = u_n + \beta_{1a} f_n + \beta_{0a} f_{n-1} \tag{29}$$

The coefficients need to be recalculated on each time step:

$$\beta_{1a} = \frac{1}{2}\frac{h_{n+1}}{h_n}(h_{n+1} + 2h_n), \qquad \beta_{0a} = -\frac{1}{2}\frac{h_{n+1}^2}{h_n}, \tag{30}$$

where $h_n = t_n - t_{n-1}$, $h_{n+1} = t_{n+1} - t_n$.

Implicit schemes, like AM or BDF, result in a nonlinear equation (or system of equations). We can use a fixed point method to solve this equation at each time step. For example, for the trapezoidal rule, at all t_n, $n = 1, ..., N$, we calculate, for $k = 0, 1, ...$ until convergence,

$$u_{n+1}^{(k+1)} = u_n + \frac{h}{2}(f_n + f(t_n, u_{n+1}^{(k)})).$$

Predictor-corrector methods avoid iteration. Suppose we want to compute u_{n+1} using an implicit method of order p. To estimate the value of $f(t_n, u_{n+1})$, we can use the 'prediction' $\tilde{u}_{n+1}$ calculated by a explicit method of order $\geq p-1$. Example. Improved Euler method (Heun) where $\tilde{u}$ is calculated using the first-order explicit Euler method,

$$\begin{aligned} \tilde{u}_{n+1} &= u_n + hf(t_n, u_n) \\ u_{n+1} &= u_n + \tfrac{h}{2}(f(t_n, u_n) + f(t_{n+1}, \tilde{u}_{n+1})) \end{aligned} \tag{31}$$

Comparison between two different fourth-order methods.

We consider again the example of a mathematical pendulum. A program that calculates the right side of the equation:

```
function f = rhs_p(u)
global nop;
f = [u(2) -sin(u(1))];
nop = nop + 1;
return
```

We are going to compare the accuracy of Gill's RK4 scheme and of the AB4 scheme, for a given computational cost. We use RK4 as a startup method for AB4.

Gill RK4 method:

```
function [uend,h] = func_pendulum_rk4gill(theta0,omega0,tmax,nt)
sqrt2 = sqrt(2);
sqrt05 = sqrt(0.5);
h = tmax/nt;
u = [theta0 omega0];
% t = 0;
for i = 1:nt
k = h * rhs_p(u);
u = u + 0.5*k;
q = k;
k = h * rhs_p(u);
u = u + (1-sqrt05)*(k-q);
q = (2-sqrt2)*k + (-2+3*sqrt05)*q;
k = h * rhs_p(u);
u = u + (1+sqrt05)*(k-q);
q = (2+sqrt2)*k + (-2-3*sqrt05)*q;
k = h * rhs_p(u);
u = u + k/6 - q/3;
% t = t + h;
end;
uend = u;
return
```

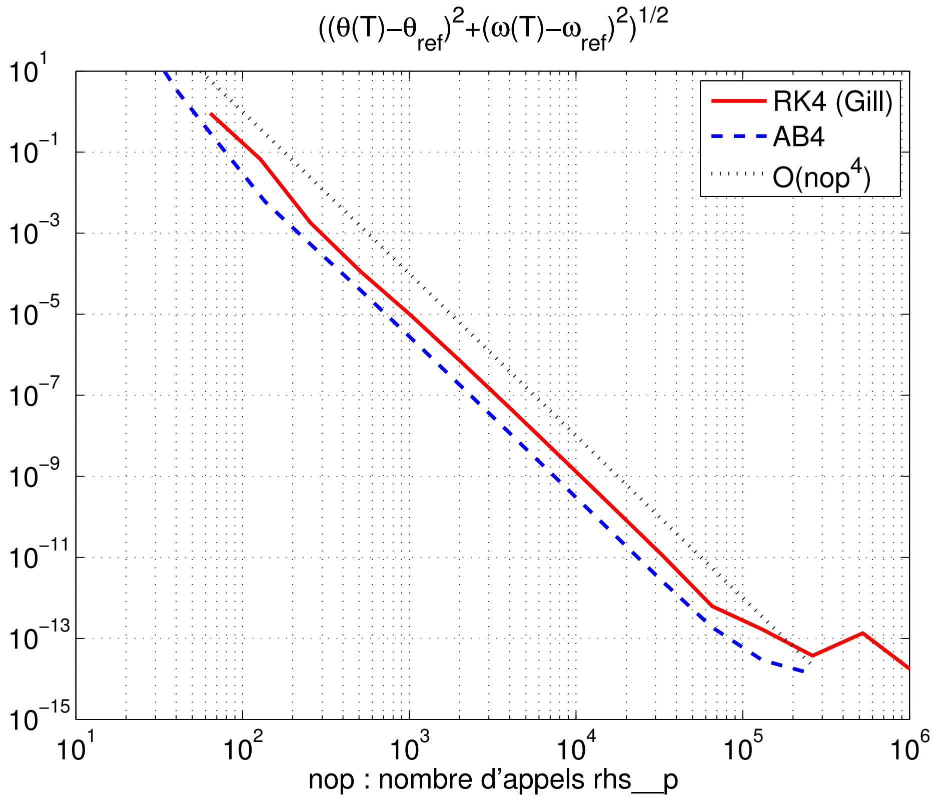
AB4 method:

```
function [uend,h] = func_pendulum_ab4(theta0,omega0,tmax,nt)
sqrt2 = sqrt(2); sqrt05 = sqrt(0.5);
h = tmax/nt;
u = [theta0 omega0];
f3 = [0 0]; f2 = [0 0]; f1 = [0 0];
% t = 0;
for i = 1:nt
f = rhs_p(u);
if (i < 4)
k = h * f; u = u + 0.5*k; q = k;
k = h * rhs_p(u); u = u + (1-sqrt05)*(k-q);
q = (2-sqrt2)*k + (-2+3*sqrt05)*q;
k = h * rhs_p(u); u = u + (1+sqrt05)*(k-q);
q = (2+sqrt2)*k + (-2-3*sqrt05)*q;
k = h * rhs_p(u); u = u + k/6 - q/3;
else
u = u + h/24 * (55*f - 59*f1 + 37*f2 - 9*f3);
end;
% t = t + h;
f3 = f2; f2 = f1; f1 = f;
end;
uend = u;
return
```

Comparison:

```
clear all; set(0,'DefaultaxesFontSize',15); set(0,'DefaulttextFontsize',15);

global nop;
tmax = 2.697200567700154e+001; % tmax = 4 periodes
theta0 = pi/3;
omega0 = 0;

% METHODE RK4
vecnt = 2.^[4:18]; vecnop = zeros(length(vecnt),1);

for it = 1:length(vecnt)
nop = 0;
[X,h] = func_pendulum_rk4gill(theta0,omega0,tmax,vecnt(it));
vecnop(it) = nop;
err_theta = abs(X(1)-theta0);
err_omega = abs(X(2)-omega0);
vecerr(it) = sqrt((err_theta).^2+(err_omega).^2);
end;

figure(1); clf;
loglog(vecnop,vecerr,'r-','LineWidth',2); hold on;
```

```
% METHODE AB4
vecnt = 2.^[4:18]; vecnop = zeros(length(vecnt),1);

for it = 1:length(vecnt)
nop = 0;
[X,h] = func_pendulum_ab4(theta0,omega0,tmax,vecnt(it));
vecnop(it) = nop;
err_theta = abs(X(1)-theta0);
err_omega = abs(X(2)-omega0);
vecerr(it) = sqrt((err_theta).^2+(err_omega).^2);
end;

figure(1);
loglog(vecnop,vecerr,'b--','LineWidth',2); hold on;
loglog(vecnop,1e8*vecnop.^(-4),'k:','LineWidth',2);
hold on;

figure(1); axis([1e1 1e6 1e-15 1e1]);
legend('RK4 (Gill)','AB4','O(nop^4)','Location','NorthEast');
xlabel('nop : nombre d''appels rhs__p');
title('((\theta(T)-\theta_{ref})^2+(\omega(T)-\omega_{ref})^2)^{1/2}');
set(gca,'XTick',10.^[1:1:6]); set(gca,'YTick',10.^[-15:2:1]); grid on;
print('-depsc','fig_err_nop.eps');
```



((θ(T)−θref)2+(ω(T)−ωref)2)1/2
RK4 (Gill)
AB4
O(nop4)
10 1
10 −1
10 −3
10 −5
10 −7
10 −9
10 −11
10 −13
10 −15
10 1
10 2
10 3
10 4
10 5
10 6
nop : nombre d'appels rhs__p