

1 Linear Multistep Methods

1.1 Introduction

Consider an initial value problem (IVP) for a time-dependent ordinary differential equation (ODE).

$$\begin{cases} y'(t) = f(y(t), t) & t_0 < t < t_1 \\ y(t_0) = y_0 \end{cases}$$

- Solve for $y(t)$ for $t_0 < t \leq t_1$
- Let $t_n = nh$ for $n = 0, 1, 2, \dots$

1.2 Multi step methods

An r -step LMM has the form (need previous values from r steps)

$$\sum_{j=0}^r \alpha_j \phi_{n-j} = h \sum_{j=0}^r \beta_j f(\phi_{n-j}, t_{n-j}) = h \sum_{j=0}^r \beta_j f_{n-j}$$

- Compute the update ϕ_n based on previous values $\phi_{n-r+1}, \phi_{n-r+2}, \dots, \phi_n$.

- $f(t_{n-j}, \phi_{n-j})$ can be stored and reused.
- If $\beta_0 = 0$, the method is explicit.
- Assume $\alpha_0 = 1$ (normalize)

The methods are so named because they involve values from the r previous steps, and are linear in f (unlike Runge-Kutta methods).

1.2.1 Adams methods (explicit)

We start again with (assume we have r previous values, and want to compute the next value at t_n for simplicity of notation) The explicit Adams methods have the form

$$\phi_n = \phi_{n-1} + h \sum_{j=1}^r \beta_j f(t_{n-j}, \phi_{n-j})$$

What are β 's?

[Adams Bashforth methods](#): explicit

- Methods are derived from numerical quadrature,

$$y(t_n) = y(t_{n-1}) + \int_{t_{n-1}}^{t_n} f(t, y(t)) dt = y(t_{n-1}) + \int_{t_{n-1}}^{t_n} y'(t) dt.$$

$$\int_{t_{n-1}}^{t_n} y'(t) dt \approx h \sum_{j=0}^r \beta_j f(t_{n-j}, y(t_{n-j})).$$

- Construct interpolating polynomial of degree $r-1$ at points $t_{n-1}, t_{n-2}, \dots, t_{n-r}$,

and then integrate to compute β_j .

$$p(t) = \sum_{j=1}^r f(y(t_{n-j}), t_{n-j}) p_j(t)$$

- **Lagrange** interpolating polynomials

$$p_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^r \frac{(t - t_{n-k})}{(t_{n-j} - t_{n-k})}$$

Example: Derive the Adams-Bashforth formulas for $r = 1$ and $r = 2$.

Q: what happens when $r = 1$?

Q: what happens when $r = 2$? Two methods

We set up $P_1(t) = At + B$ to interpolate the points (t_{n-2}, y_{n-2}) and (t_{n-1}, y_{n-1}) .

We have

$$P_1(t_{n-1}) = y'_{n-1} = f(t_{n-1}, y_{n-1}) = At_{n-1} + B,$$

$$P_1(t_{n-2}) = f(t_{n-2}, y_{n-2}) = y'_{n-2} = At_{n-2} + B.$$

Let $f_{n-1} = f(t_{n-1}, y_{n-1})$ and $f_{n-2} = f(t_{n-2}, y_{n-2})$. Solving the above equations, we get

$$A = \frac{f_{n-1} - f_{n-2}}{t_{n-1} - t_{n-2}}, \quad B = \frac{f_{n-2}t_{n-1} - f_{n-1}t_{n-2}}{t_{n-1} - t_{n-2}}.$$

We should note given $P_1(t) = At + B$, the integral

$$\int_{t_{n-1}}^{t_n} y'(t)dt \approx \int_{t_{n-1}}^{t_n} P_1(t)dt = \frac{A}{2}(t_n^2 - t_{n-1}^2) + B(t_n - t_{n-1})$$

If even spaced time points are used, $t_n - t_{n-1} = t_{n+1} - t_n = h$, and the integral above is reduced to

$$\int_{t_{n-1}}^{t_n} P_1(t)dt = h \left(\frac{3}{2}f_{n-1} - \frac{1}{2}f_{n-2} \right).$$

This leads to the two-step Adams-Bashforth method, which is

$$\phi_n = \phi_{n-1} + \frac{h}{2}(3f(t_{n-1}, \phi_{n-1}) - f(t_{n-2}, \phi_{n-2})).$$

We note that the Adams-Bashforth method is explicit in time, and it is a two-step method (uses two previous time points). The method is 2nd order accurate in terms of global error.

Method 2 We know the goal is to construct

$$\frac{y(t_n) - y(t_{n-1})}{h} = \int_{t_{n-1}}^{t_n} y'(t)dt \approx h \sum_{j=1}^2 \beta_j f(t_{n-j}, y(t_{n-j})).$$

And we can see it is

$$\frac{y_n - y_{n-1}}{h} = \beta_1 y'_{n-1} + \beta_2 y'_{n-2} + \mathbf{error}$$

We expect to have the error to be $O(h^2)$.

First, expand the left hand side in a Taylor series expansion about t_n :

$$\text{LHS} = \frac{y_n - y_{n-1}}{h} = y'(t_n) - \frac{h}{2} y''(t_n) + O(h^2)$$

Next, expand the right hand side in a Taylor series expansion about t_n :

$$\text{RHS} = \beta_1(y'_n - hy''_n) + \beta_2(y'_n - 2hy''_n) + O(h^2)$$

Now we match the coefficients of the Taylor series expansions to get the coefficients β_1 and β_2 .

$$\begin{aligned}\beta_1 + \beta_2 &= 1, \\ \beta_1 + \beta_2 b &= -\frac{1}{2}\end{aligned}$$

So we know $\beta_1 = 3/2$ and $\beta_2 = -1/2$. Therefore, the two-step Adams-Bashforth method is

$$\phi_n = \phi_{n-1} + \frac{h}{2}(3f(t_{n-1}, \phi_{n-1}) - f(t_{n-2}, \phi_{n-2})).$$

Remarks:

- AB2 is a 2-step method and is 2nd order accurate. It is also efficient, as we can use the previous value to compute the next value.
- Starting value: A multistep method has the disadvantage that it requires more than one starting value, even though the ODE only requires one. This means that the first $r - 1$ steps (for $\phi_1, \dots, \phi_{r-1}$) must be computed by some other method. In those cases, usually a one step method is used.

Adams-Moulton methods (implicit) In general linear multi-step method

$$\sum_{j=0}^r \alpha_j \phi_{n-j} = h \sum_{j=0}^r \beta_j f(\phi_{n-j}, t_{n-j}) = h \sum_{j=0}^r \beta_j f_{n-j}$$

- Let $\beta_r \neq 0$.
- Polynomial has degree r
- Lagrange interpolating polynomials of degree r , $\phi_j(t)$, $n - r \leq j \leq n$.

$$p(t) = \sum_{j=0}^r f(t_{n-j}, \phi(t_{n-j})) p_j(t)$$

- Yields an implicit method of order $r + 1$

Example: Derive the Adams-Moulton formulas for $r = 1$ using numerical quadrature and Lagrange interpolating polynomials.

Remark: Adams-Moulton Methods The Adams-Moulton methods are implicit in time, and they are constructed by using the polynomial $P_1(t)$ to interpolate the points (t_{n+1}, y_{n+1}) and (t_n, y_n) . The second order Adams-Moulton method is

$$y_n = y_{n-1} + \frac{h}{2}(f(t_n, y_n) + f(t_{n-1}, y_{n-1})).$$

The method is 2nd order accurate in terms of global error, and it is implicit in time and has zero-stability. This is also called the trapezoidal method. We note in general the implicit Moulton method has $r + 1$ order of accuracy, with r being the steps involved.

1.3 Predictor Corrector Methods

A strategy of combining the explicit and implicit methods is to use a predictor-corrector pair. For example, we can use the explicit Adams-Bashforth method to predict the value of at t_{n+1} , and denote it by ϕ_{n+1}^* , and then use the implicit Adams-Moulton method to correct the value of y_{n+1} , and denote it by ϕ_{n+1} . The predictor-corrector pair is

$$\begin{aligned}\phi_{n+1}^* &= \phi_n + \frac{h}{2}(3f(t_n, \phi_n) - f(t_{n-1}, \phi_{n-1})), \\ \phi_{n+1} &= \phi_n + \frac{h}{2}(f(t_{n+1}, \phi_{n+1}^*) + f(t_n, \phi_n)).\end{aligned}$$

This predictor-corrector pair is 2nd order accurate in terms of global error, and it is *explicit* in time!

A more involved predictor-corrector pair is the Adams-Bashforth-Moulton method, which is a combination of the Adams-Bashforth and Adams-Moulton methods. We illustrate the two-step as follows. First we use the Adams-Bashforth method to predict the value at t_{n+1} :

$$\phi_{n+1}^* = \phi_n + \frac{h}{24}(55f(t_n, \phi_n) - 59f(t_{n-1}, \phi_{n-1}) + 37f(t_{n-2}, \phi_{n-2}) - 9f(t_{n-3}, \phi_{n-3})),$$

and then we use the Adams-Moulton method to correct the value at t_{n+1} :

$$\phi_{n+1} = \phi_n + \frac{h}{24}(9f(t_{n+1}, \phi_{n+1}) + 19f(t_n, \phi_n) - 5f(t_{n-1}, \phi_{n-1}) + f(t_{n-2}, \phi_{n-2})).$$

This predictor-corrector pair is 4th order accurate in terms of global error, and it is *implicit* in time! The corrector step solved by a fixed number of fixed point iterations, using the solution ϕ_{n+1}^* by the explicit method as starting values for the iterations.

The standard procedure of using the Adams-Bashforth-Moulton method with given order r is to use several steps of RK methods to get the initial values, and then do the predictor and corrector update. The explicit predictor use the Adams-Bashforth method, and the implicit corrector use the Adams-Moulton method.

1.4 Other multi-step methods

Backward Differentiation Formulas (BDF): The BDF methods use not only previous f values, but also previous y values. The key idea is to approximate

$$hy'(t) \approx c_0y(t_n) + c_1y(t_{n-1}) + \cdots + c_r y(t_{n-r}).$$

We note the first order BDF is the backward Euler method, and the second order BDF is the trapezoidal method.

$$\text{BDF1 : } \phi_n = \phi_{n-1} + hf(t_n, \phi_{n-1}) \rightarrow \frac{\phi_n - \phi_{n-1}}{h} = f(t_n, \phi_n)$$

$$\text{BDF2 : } \frac{3\phi_n - 4\phi_{n-1} + \phi_{n-2}}{2h} = f(t_n, \phi_n).$$

The BDF methods are implicit in time, and they are $r + 1$ order accurate. The BDF methods are zero-stable, and they are used for stiff problems. The BDF methods are used in Matlab's `ode15s` function, which is a stiff solver.

2 Systems of ODEs

2.1 General ideas

For systems of ODEs, we can use the same methods as for scalar ODEs. For example, for the system

$$\begin{aligned}y_1' &= f_1(t, y_1, y_2), \\y_2' &= f_2(t, y_1, y_2),\end{aligned}$$

we can use the RK4 method to advance the solution from t_n to t_{n+1} as follows:

$$\begin{aligned}k_{11} &= f_1(t_n, y_{1n}, y_{2n}), \quad k_{21} = f_2(t_n, y_{1n}, y_{2n}), \\k_{12} &= f_1(t_n + \frac{h}{2}, y_{1n} + \frac{h}{2}k_{11}, y_{2n} + \frac{h}{2}k_{21}), \quad k_{22} = f_2(t_n + \frac{h}{2}, y_{1n} + \frac{h}{2}k_{11}, y_{2n} + \frac{h}{2}k_{21}), \\k_{13} &= f_1(t_n + \frac{h}{2}, y_{1n} + \frac{h}{2}k_{12}, y_{2n} + \frac{h}{2}k_{22}), \quad k_{23} = f_2(t_n + \frac{h}{2}, y_{1n} + \frac{h}{2}k_{12}, y_{2n} + \frac{h}{2}k_{22}), \\k_{14} &= f_1(t_n + h, y_{1n} + hk_{13}, y_{2n} + hk_{23}), \quad k_{24} = f_2(t_n + h, y_{1n} + hk_{13}, y_{2n} + hk_{23}), \\y_{1n+1} &= y_{1n} + \frac{h}{6}(k_{11} + 2k_{12} + 2k_{13} + k_{14}), \quad y_{2n+1} = y_{2n} + \frac{h}{6}(k_{21} + 2k_{22} + 2k_{23} + k_{24}).\end{aligned}$$

We note that the RK4 method is 4th order accurate in terms of global error for systems of ODEs. And the method is explicit in time, which makes the implementation much easier. For *harder* problems (stiff problems, whose solutions change rapidly), we can use the implicit methods, which are more stable than the explicit methods.

2.2 High order ODEs to system of 1st order eqns

For higher order ODEs, we can convert them to systems of first order ODEs.

For example, for the second order ODE

$$u''(t) = f(t, u(t), u'(t)), \quad u(a) = u_0, \quad u'(a) = u_1,$$

we can introduce a new variable $y_1(t) = u(t)$, $y_2(t) = u'(t)$, and then we have the system

$$\begin{aligned} y_1'(t) &= y_2(t), \\ y_2'(t) &= f(t, y_1(t), y_2(t)). \end{aligned}$$

This is a system of first order ODEs, and we can use the RK4 method to solve it. We can also use the Adams-Bashforth-Moulton method to solve the system.

Example 1, for the second order ODE (Van der Pol's equation)

$$u'' + \mu(u^2 - 1)u' + u = 0$$

we can convert it to the system

$$\begin{aligned} y_1'(t) &= y_2(t), \\ y_2'(t) &= -\mu(y_1^2 - 1)y_2 - y_1. \end{aligned}$$

Example 2, for the given system of equations

$$y_1' = -2y_1 + y_2, y_2' = 998y_1 - 999y_2, \quad y_1(0) = 1, \quad y_2(0) = 1,$$

We can try to use Euler's method and other methods to solve the system, and compare the results (check $h = 0.00210$ and $h = 0.00200$) We note the eigenvalues of the matrix $A = \begin{bmatrix} -2 & 1 \\ 998 & -999 \end{bmatrix}$ are $\lambda_1 = -1000$ and $\lambda_2 = -1$, and the system is **stiff**.

For a nonlinear system of ODEs, we need to check the Jacobian matrix of the system for analyzing the stability of the system.

2.3 Use Matlab's built-in function for system of ODEs

Matlab has a built-in function `ode45` for solving systems of ODEs. The function `ode45` uses the Runge-Kutta method to solve the system of ODEs. The function is called as

$$[t, y] = \text{ode45}(@f, [a, b], y_0),$$

where `@f` is the function that defines the system of ODEs, $[a, b]$ is the interval of the solution, and y_0 is the initial value of the system. The function `@f` should be defined as

$$\text{function dydt} = f(t, y),$$

where `dydt` is the derivative of the system of ODEs. For example, for the system

$$\begin{aligned}y_1' &= -2y_1 + y_2, \\y_2' &= 998y_1 - 999y_2, \\y_1(0) &= 1, \quad y_2(0) = 1,\end{aligned}$$

the function `@f` should be defined as

$$\text{function dydt} = f(t, y), \text{dydt} = [-2 * y(1) + y(2); 998 * y(1) - 999 * y(2)];$$

The function `ode45` returns the solution y at the time points t . The solution y is a matrix with the first column being the solution of the first ODE, and

the second column being the solution of the second ODE. The time points t are stored in the vector t . We can plot the solution using the command `plot(t,y)`.

For stiff problems, we can use the function `ode15s`, which is a stiff solver. The function `ode15s` uses the implicit methods to solve the system of ODEs. The function `ode15s` is called as

$$[t, y] = \text{ode15s}(@f, [a, b], y_0).$$

The function `@f` is defined as before.

We note that the function `ode45` is a general purpose solver. The function implements a Runge-Kutta method, and it is explicit in time. The function use adaptive time stepping to solve the system of ODEs. The method is called embedded method, as it uses two Runge-Kutta methods of order 4 and 5 to solve the system of ODEs. The method is 4th order accurate in terms of global error.

2.4 Model with ODEs

The Robertson's chemical reaction model is a system of three ODEs that models the reaction of hydrogen, nitrogen, and ammonia. The system of

ODEs is given by

$$\begin{aligned}y_1' &= -0.04y_1 + 10^4y_2y_3, \\y_2' &= 0.04y_1 - 10^4y_2y_3 - 3 \cdot 10^7y_2^2, \\y_3' &= 3 \cdot 10^7y_2^2,\end{aligned}$$

with initial conditions $y_1(0) = 1$, $y_2(0) = 0$, and $y_3(0) = 0$. The system is stiff, and we can use the function `ode15s` to solve the system. The function `@f` should be defined as

```
function dydt = f(t,y)
dydt = [-0.04 * y(1) + 10^4 * y(2) * y(3);
0.04 * y(1) - 10^4 * y(2) * y(3) - 3 * 10^7 * y(2)^2;
3 * 10^7 * y(2)^2];
```

Use both `ode45` and `ode15s` to solve the system, and compare the results.

<https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html>