

Lecture 14: IVP Euler, Runge-Kutta and Adams methods

Initial Value Problems (IVP) consider general systems of differential equations with a corresponding initial value

$$\frac{dy}{dt} = f(t, y)$$

$$y(0) = y_0$$

where y represents a vector, and $t \in [0, T]$.

Some cases can be solved analytically, but not generally so we rely on numerical methods.

The simplest algorithm is derived from the definition of the derivative

$$\frac{dy}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\Delta y}{\Delta t}$$

taking a numerical approximation of this by taking a time span $\Delta t = t_{n+1} - t_n$

$$\frac{dy}{dt} = f(t, y) \Rightarrow \frac{y_{n+1} - y_n}{\Delta t} \approx f(t_n, y_n)$$

Euler's method (Forward Euler) being

$$y_{n+1} = y_n + \Delta t f(t_n, y_n)$$

This idea is generalized as the following scheme

$$y_{n+1} = y_n + \Delta t \phi$$

where ϕ could include a derivative at the midpoint (between t_{n+1} and t_n and the derivative at the right end point t_{n+1} to improve accuracy).

Modified Euler-Cauchy (otherwise known as second order Runge-Kutta) method is one derived from a more general scheme which includes information from intermediate points in ϕ :

$$y(t + \Delta t) =$$

$$y(t) + \Delta t \cdot f \left(t + \frac{\Delta t}{2}, y(t) + \frac{\Delta t}{2} \cdot f(t, y(t)) \right)$$

Generally methods for iterating forward in time given a single initial point are known as Runge-Kutta methods, and they can be developed to have arbitrary accuracy.

A very popular Runge-Kutta scheme is the "fourth order" Runge-Kutta (the global error is of $O(\Delta^4)$).

$$y_{n+1} = y_n + \frac{\Delta t}{6}[f_1 + 2f_2 + 2f_3 + f_4]$$

$$f_1 = f(t_n, y_n)$$

$$f_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}f_1\right)$$

$$f_3 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}f_2\right)$$

$$f_4 = f(t_n + \Delta t, y_n + \Delta t f_3)$$

Alternatively to develop methods which solve the IVP, we can start with the fundamental theorem of Calculus

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) \Rightarrow \mathbf{y}(t + \Delta t) - \mathbf{y}(t) = \int_t^{t+\Delta t} f(t, \mathbf{y}) dt$$

Approximating the function $f(t, \mathbf{y})$ by a polynomial $f(t, \mathbf{y}) \approx p(t, \mathbf{y})$ and rewriting this for iteration

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p(t, \mathbf{y}) dt \quad (1)$$

we can use this to develop methods of the Adams-Bashforth type (methods which use present points and past points thus requiring more than one initial point).

For the case of a polynomial of 0^{th} order (a constant) polynomial $p_1 = f(t_n, \mathbf{y}_n)$ the resulting method is the Euler method, alternately called the **forward Euler** method.

If instead we take a polynomial of order 2 and insert it into equation (1)

$$p_2(t) = f_{n-1} + \frac{f_n - f_{n-1}}{\Delta t}(t - t_n)$$

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f_{n-1} + \frac{f_n - f_{n-1}}{\Delta t}(t - t_n) dt$$

We get

$$y_{n+1} = y_n + \frac{\Delta t}{2}[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})]$$

Generally speaking for Adams-Bashforth methods as the accuracy is increased so is the number of initial points needed to make one time step Δt .

⇒ This means that you need to find a way to get these initial conditions if they are not provided. One way is to use a one step method like Runge-Kutta to get the initial points required before applying the multi-step methods.

A further addition to our suite of ODE solvers is the Adams-Moulton methods. These include information from the present points, past points and the future points.

The simplest Adams-Moulton method is obtained by taking a polynomial of 0^{th} order $p_1 = f(t_{n+1}, y_{n+1})$ in equation (1) to obtain the **Backward Euler** method

$$y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1})$$

{Note that this is an implicit method.}

A second order Adams-Moulton method derived by taking a linear polynomial approximation is

$$y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_{n+1}, y_{n+1}) + f(t_n, y_n)]$$

Predictor-Corrector methods These combine both the Adams-Bashforth and Adams-Moulton first using the Adams-Bashforth to predict the updated value and then inserting the predicted value into the Adams-Moulton method to get a corrected value, such as:

$$y_{n+1}^P = y_n + \frac{\Delta t}{2}[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})]$$

$$y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_{n+1}, y_{n+1}^P) + f(t_n, y_n)]$$

The methods for solving ODEs available in MATLAB are

- ode23 is a second order Runge-Kutta routine
- ode45 is a fourth-order Runge-Kutta routine
- ode13 is a variable order predictor-corrector routine
- ode15s is a variable order Gear method for stiff problems

help ode45 help ode23

Turning higher ODEs into first order ODEs

is something you can do easily and necessary for applying the methods we've learned here:

A practical example: given a mass spring system (the following is a reduced model from a model of a wave energy device) where:

- The float (first mass) displacement, velocity and acceleration (measured from its initial values) are defined as the variables z_1 , \dot{z}_1 and \ddot{z}_1 .
- The piston (second mass) displacement, velocity and acceleration (measured from its initial values) are defined as the variables z_2 , \dot{z}_2 and \ddot{z}_2 .

- The (very reduced) linear second order ordinary differential equations of motion for the vertical motion of the two bodies can be expressed as follows:

$$\begin{aligned}(M_1)\ddot{z}_1 + b\dot{z}_1 + c(z_1 - z_2) &= F(t) \\ M_2\ddot{z}_2 &= c(z_1 - z_2)\end{aligned}\tag{2}$$

where c, b are constants ("Power Take Off" spring stiffness c and Hydro dynamic damping of float b) and $F(t)$ is the forcing function of time t .

Rearrangement of equations in (2) In order to solve the system (2) using the Matlab solver packages we need to convert the second order differential equations into a system of first order differential equations.

The following 'trick' is a standard technique covered in general ordinary differential equations courses in university. We introduce the variables $v_1 = \dot{z}_1$ and $v_2 = \dot{z}_2$.

The second order system of ordinary differential equations (2) can then be rearranged into the following system of first order ordinary differential equations as:

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{v}_1 &= \frac{F_w(t) - bv_1 - c(z_1 - z_2)}{M_1} \\ \dot{v}_2 &= \frac{c(z_1 - z_2)}{M_2} \end{aligned} \quad (3)$$

Let $\mathbf{y} = (z_1, z_2, v_1, v_2)^T$ then $\dot{\mathbf{y}} = (\dot{z}_1, \dot{z}_2, \dot{v}_1, \dot{v}_2)^T$ so that the above system is written in matrix form as

$$M\dot{\mathbf{y}} = f(\mathbf{y}) \quad (4)$$

where M (in this case) is the identity matrix,

$$f(\mathbf{y}) = \begin{bmatrix} v_1 \\ v_2 \\ \frac{F(t) - bv_1 - c(z_1 - z_2)}{M_1} \\ \frac{c(z_1 - z_2)}{M_2} \end{bmatrix} \quad (5)$$

and the initial conditions at time $t = 0$ are

$$\begin{aligned} z_1(0) &= 0, \\ z_2(0) &= 0, \\ v_1(0) &= 0, \\ v_2(0) &= 0. \end{aligned} \quad (6)$$

Now solve in MATLAB: using `aquabouy.m` and `solve_aquabouy.m`