CS205b/CME306

Lecture 4

1 Time Integration

We now consider several popular approaches for integrating an ODE.

1.1 Forward Euler

Forward Euler evolution takes on the form

$$\begin{pmatrix} x \\ v \end{pmatrix}^{n+1} = \begin{pmatrix} x \\ v \end{pmatrix}^{n} + \Delta t \begin{pmatrix} v \\ a \end{pmatrix}^{n}.$$

Because forward Euler is unstable when the system contains pure imaginary eigenvalues, it is unstable without damping. In particular, $a_v < 0$. Forward Euler is stable when $|\Delta t\lambda + 1| < 1$. A time step restriction of $\Delta t \approx |\lambda|^{-1}$ is required. As problems get stiffer, $\max|\lambda|$ becomes larger, and consequently the time step that can be taken becomes smaller. The stability region for forward Euler is shown in Figure 1(a). If the system has eigenvalues near the imaginary axis, we would be much better off using using a different method such as one of the higher order Runge-Kutta methods.

1.2 Runge-Kutta Methods

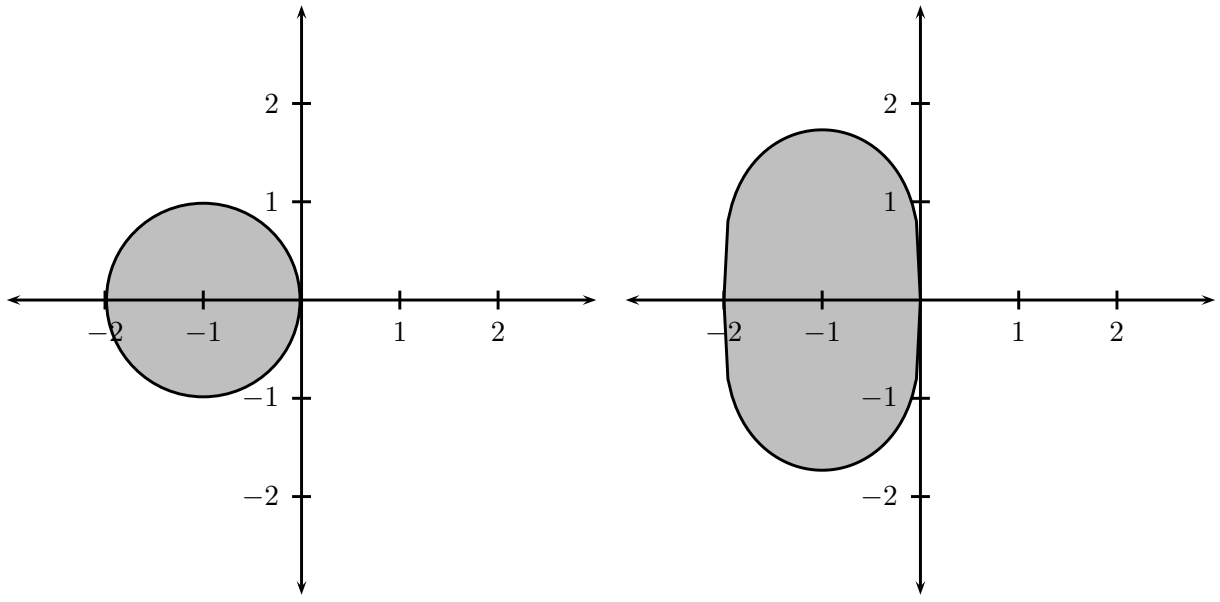
There are a number of Runge-Kutta methods in common use. The Runge-Kutta methods were discussed in more detail in CS205A, so they are only highlighted here. The second order Runge-Kutta method is technically linearly unstable without damping since its stability region does not include the imaginary axis. Roundoff with $a_v < 0$ may be sufficient, and nonlinear effects can sometimes rectify the problem. The third order Runge-Kutta method is stable without damping, since its stability region includes part of the imaginary axis. It is suitable for system with $a_v = 0$. Stability regions for various Runge-Kutta methods are shown in Figure 1.

1.3 Backward Euler

The need for a time step restriction is true of all explicit methods. By contract, implicit methods often have a much better time step restriction that makes them more suitable even though they require a linear system solve and are thus much more expensive per step.

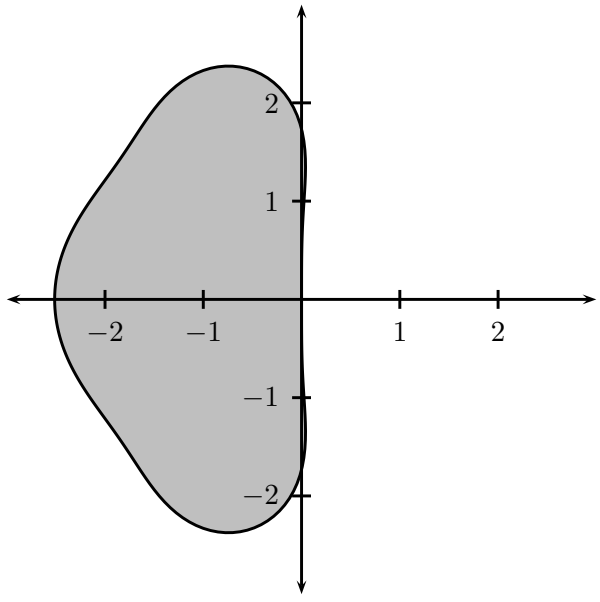
Backward Euler is the simplest of the implicit methods. Its stability region (Figure 2(a)) looks very different from those of the explicit integrators. It takes the form

$$\begin{pmatrix} x \\ v \end{pmatrix}^{n+1} = \begin{pmatrix} x \\ v \end{pmatrix}^{n} + \Delta t \begin{pmatrix} v \\ a \end{pmatrix}^{n+1}.$$

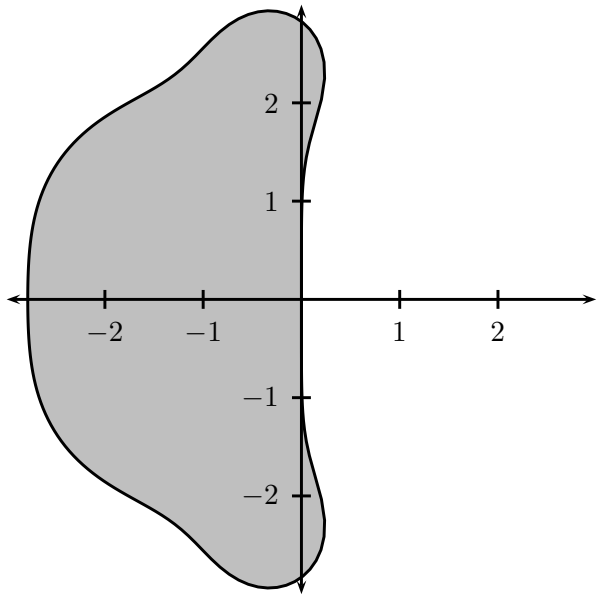


(a) Forward Euler (First Order Runge-Kutta)

(b) Second Order Runge-Kutta

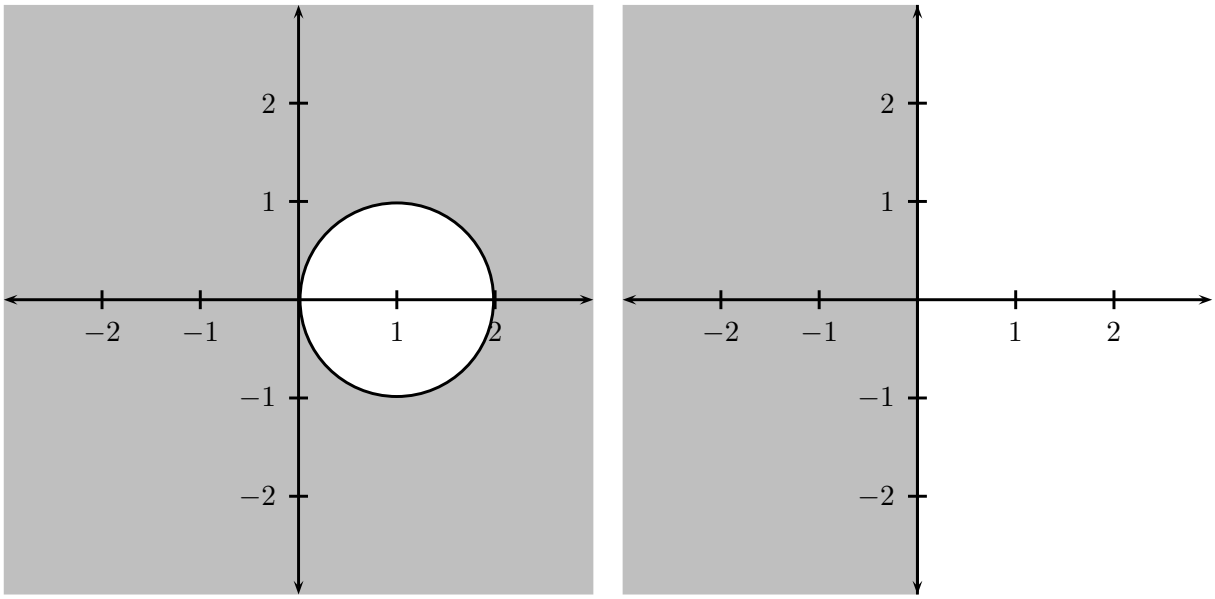


(c) Third Order Runge-Kutta



(d) Fourth Order Runge-Kutta

Figure 1: Stability regions for Runge-Kutta. The scheme is stable with time step size Δt if $\Delta t\lambda$, a complex number in general, is located within the shaded region of the complex plane. Note that the third and fourth order schemes have stability regions that encompass the imaginary axis, making them stable on undamped problems. The first and second order schemes do not have this property.



(a) Backward Euler (b) Trapezoid Rule

Figure 2: Stability regions for two implicit time integration schemes. The scheme is stable with time step size Δt if $\Delta t\lambda$, a complex number in general, is located within the shaded region of the complex plane.

Note that the new values being obtained occur on both sides of the equation, so a solve is typically necessary at each time step. Backward Euler has no time step restriction for stability, but large time steps do result in poor accuracy.

The solve required for backward Euler can be relatively easy in the case of linear ODEs. For example, if the only force is drag, $F = -kv$ and $a = -kv/m$. Then, $v^{n+1} = v^n - \Delta t k v^{n+1}/m$ so that $v^{n+1} = (1 + \Delta t k/m)^{-1} v^n$ and $x^{n+1} = x^n + \Delta t v^{n+1} = x^n + \Delta t (1 + \Delta t k/m)^{-1} v^n$ are relatively easily obtained. Note that this damps velocity each time step, and it damps more with larger time steps. This is in contrast to forward Euler, where $v^{n+1} = (1 - \Delta t k/m) v^n$. Damping here occurs if $\Delta t < 2m/k = (\max |\lambda|)^{-1}$ but amplification occurs for larger time steps.

More generally, backward Euler requires solving a linear system. If the ODE is nonlinear, it will typically be more difficult and expensive to solve and requires that the nonlinear system first be linearized.

1.4 Trapezoid Rule

Forward Euler evolution takes on the form

$$\begin{pmatrix} x \\ v \end{pmatrix}^{n+1} = \begin{pmatrix} x \\ v \end{pmatrix}^{n} + \frac{1}{2}\Delta t \begin{pmatrix} v \\ a \end{pmatrix}^{n} + \frac{1}{2}\Delta t \begin{pmatrix} v \\ a \end{pmatrix}^{n+1}.$$

Because forward Euler is unstable when the system contains pure imaginary eigenvalues, it is unstable without damping. In particular, $a_v < 0$. Forward Euler is stable when $|\Delta t\lambda + 1| < 1$. A time step restriction of $\Delta t \approx |\lambda|^{-1}$ is required. As problems get stiffer, $\max |\lambda|$ becomes larger, and consequently the time step that can be taken becomes smaller. The stability region for forward

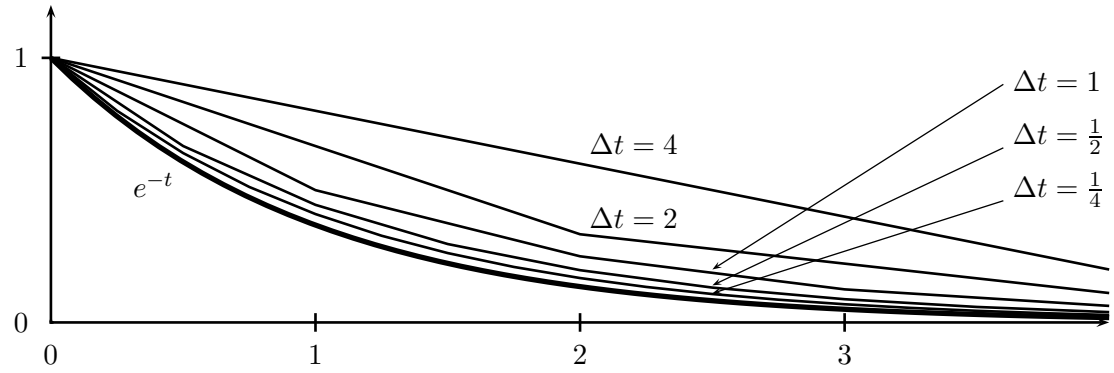


Figure 3: Effects of large time step sizes on numerical simulation of $y' = -y$ with backward Euler.

Euler is shown in Figure 2(b). If the system has eigenvalues near the imaginary axis, we would be much better off using using a different method such as one of the higher order Runge-Kutta methods.

1.5 Newmark Method

Newmark methods are the most famous of the multivalue methods in computation solids. Multivalue methods are those that take advantage of the fact that higher order derivatives can be computed at lower accuracy to be more efficient. The Newmark method is a two-parameter family of methods:

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2}((1-2\beta)a^n + 2\beta a^{n+1})$$
$$v^{n+1} = v^n + \Delta t((1-\gamma)a^n + \gamma a^{n+1}).$$

The specific method depends on the choice of β and γ chosen.

Constant acceleration equations, $\beta = \gamma = 0$

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2}a^n \qquad v^{n+1} = v^n + \Delta t a^n$$

Implicit acceleration, $\beta = 1/2$ **and** $\gamma = 1$

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2}a^{n+1} \qquad v^{n+1} = v^n + \Delta t \gamma a^{n+1}$$

The velocity update is the first order backward Euler update. The position update can be written as

$$x^{n+1} = x^n + \Delta t \frac{v^n + v^{n+1}}{2},$$

which is just the second order accurate trapezoid rule. This choice is still only first order accurate overall.

There is a theorem that states that second order accuracy is obtained if and only if $\gamma = 1/2$. The following two methods are second order Newmark methods.

Trapezoid rule, $\beta = 1/4$ and $\gamma = 1/2$

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2}\frac{a^n + a^{n+1}}{2} \qquad v^{n+1} = v^n + \Delta t \frac{a^n + a^{n+1}}{2}$$

This method gets its name since the position equation can also be rewritten as a trapezoid rule update as

$$x^{n+1} = x^n + \Delta t \frac{v^n + v^{n+1}}{2}.$$

Central differencing, $\beta = 0$ and $\gamma = 1/2$

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2} a^n \qquad v^{n+1} = v^n + \Delta t \frac{a^n + a^{n+1}}{2}.$$

The name of this method comes from the ability of its updates to be expressed as the central differences

$$\frac{x^{n+1} - x^{n-1}}{2\Delta t} = v^n \qquad \frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2} = a^n.$$

These equations can be obtained by

$$\begin{aligned}
x^{n+1} + x^n &= \left(x^n + \Delta t v^n + \frac{\Delta t^2}{2} a^n\right) + \left(x^{n-1} + \Delta t v^{n-1} + \frac{\Delta t^2}{2} a^{n-1}\right) \\
x^{n+1} - x^{n-1} &= \Delta t v^n + \frac{\Delta t^2}{2} a^n + \Delta t v^{n-1} + \frac{\Delta t^2}{2} a^{n-1} \\
\frac{x^{n+1} - x^{n-1}}{\Delta t} &= v^n + v^{n-1} + \Delta t \frac{a^n + a^{n-1}}{2} \\
\frac{x^{n+1} - x^{n-1}}{2\Delta t} &= v^n \\
x^{n+1} - x^n &= \left(x^n + \Delta t v^n + \frac{\Delta t^2}{2} a^n\right) - \left(x^{n-1} + \Delta t v^{n-1} + \frac{\Delta t^2}{2} a^{n-1}\right) \\
x^{n+1} - 2x^n + x^{n-1} &= \Delta t v^n + \frac{\Delta t^2}{2} a^n - \Delta t v^{n-1} - \frac{\Delta t^2}{2} a^{n-1} \\
\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t} &= v^n - v^{n-1} + \frac{\Delta t}{2}(a^n - a^{n-1}) \\
\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t} &= v^{n-1} + \Delta t \frac{a^{n-1} + a^n}{2} - v^{n-1} + \frac{\Delta t}{2}(a^n - a^{n-1}) \\
\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2} &= a^n
\end{aligned}$$

1.6 Staggering the Position and Velocity

Define velocity halfway between time steps so that the update

$$v^{n+1/2} = \frac{x^{n+1} - x^n}{\Delta t}$$

is second order accurate. Note that x^n is still defined at the normal times. We can then define v^n by averaging nearby velocities as

$$v^n = \frac{v^{n-1/2} + v^{n+1/2}}{2} = \frac{x^{n+1} - x^{n-1}}{2\Delta t},$$

which is also central differencing for velocity. The update formula for positions is then just

$$x^{n+1} = x^n + \Delta t v^{n+1/2}.$$

Acceleration is also at grid points

$$a^n = a(x^n, v^n) = a\left(x^n, \frac{v^{n-1/2} + v^{n+1/2}}{2}\right)$$

so the update

$$v^{n+1/2} = v^{n-1/2} + \Delta t a^n$$

is also second order accurate. We also have the second order accurate central difference formula

$$\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2} = a^n.$$

We can combine these formulas to obtain the explicit formula for a half time step

$$\begin{aligned} v^n &= \frac{v^{n-1/2} + v^{n+1/2}}{2} \\ v^n &= \frac{(v^{n+1/2} - \Delta t a^n) + v^{n+1/2}}{2} \\ v^{n+1/2} &= v^n + \frac{\Delta t}{2} a^n \end{aligned}$$

which is then used to update positions. Note that these two steps are equivalent to

$$x^{n+1} = x^n + \Delta t v^n + \frac{\Delta t^2}{2} a^n.$$

Finally, velocities are updated another half step using the implicit update

$$\begin{aligned} v^{n+1} &= \frac{v^{n+1/2} + v^{n+3/2}}{2} \\ v^{n+1} &= \frac{v^{n+1/2} + (v^{n+1/2} + \Delta t a^{n+1})}{2} \\ v^{n+1} &= v^{n+1/2} + \frac{\Delta t}{2} a^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(x^{n+1}, v^{n+1}) \end{aligned}$$

The overall velocity update is

$$v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(x^{n+1}, v^{n+1}) = v^n + \frac{\Delta t}{2} a(x^n, v^n) + \frac{\Delta t}{2} a(x^{n+1}, v^{n+1}).$$

which is just the trapezoid rule.

The dependence of acceleration on velocity is often symmetric (e.g., damping forces), so that a fast solver like preconditioned conjugate gradient can be employed.