Sources of error

- round-off error, especially if the steps get too small.
- local error: the error assuming that y_n is the true value.
- global error: how far we have strayed from our original solution curve.

Local error for Euler's method

Taylor series tells us that

$$y(t_{n+1}) - y_{n+1} = \frac{h^2}{2} y''(\xi)$$

Global error

We know

$$y(t_{n+1}) = y(t_n) + h_n f(t_n, y(t_n)) + \frac{h^2}{2} y''(\xi_n), \quad n = 0, 1, 2, \ldots,$$

and

$$y_{n+1} = y_n + h_n f(t_n, y_n)\,.$$

Therefore,

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

- The plum-colored terms are global errors.
- The black term is the local error.
- Using the MVT, the blue term can be written as

$$h_n(f(t_n, y(t_n)) - f(t_n, y_n)) = h_n f_y(\eta)(y(t_n) - y_n)\,.$$

So we have the expression

$$\text{new global error} = (1 + h_n f_y(\eta))\ \text{(old global error)} + \text{local error}_n \quad .$$

Therefore, the errors will be magnified if

$$|1 + h_n f_y(\eta)| > 1$$

and we say that the Euler method is unstable in this case.

Note: We use the word stable in two ways: stability of an ODE and stability region for a method for solving ODEs.

So the stability interval for Euler's method is

$$-2 < h f_y < 0$$

for a single equation. For a system of equations, the stability region is the region where the eigenvalues of $\mathbf{I} + h\mathbf{f}_y$ are in the unit circle.

BACKWARD EULER METHOD

Taylor series derivation

$$y(t) = y(t+h) - hy'(t+h) + \frac{h^2}{2}y''(\xi)$$

where $\xi \in [t, t+h]$, so

$$y_{n+1} = y_n + hf_{n+1}$$

How to use it

Given y_0 and $t_0, t_1, \ldots t_N$,

For $n = 0, \ldots, N-1$,

Solve the nonlinear equation

$$y_{n+1} = y_n + (t_{n+1} - t_n) f(t_{n+1}, y_{n+1})$$

Example: $y' = -y$

$$y_{n+1} = y_n - h_n y_{n+1}$$

so

$$(1 + h_n) y_{n+1} = y_n$$

and

$$y_{n+1} = y_n / (1 + h_n) \, .$$

In general, it is not this easy to solve the nonlinear equation, and we use a nonlinear equation solver from Chapter 24, or we use functional iteration:

P (predict): Guess y_{n+1} (perhaps using Euler's method).

E (evaluate): Evaluate $f_{n+1} = f(t_{n+1}, y_{n+1})$.

C (correct): Plug the current guess in, to get a new guess:

$$y_{n+1} = y_n + h_n f_{n+1} \, .$$

E: Evaluate $f_{n+1} = f(t_{n+1}, y_{n+1})$.

Repeat the CE steps if necessary.

We call this a PECE (or PE(CE)^k) scheme.

Note: If we fail to solve the nonlinear equation exactly, this adds an additional source of error.

Jargon

If y_{n+1} is on the right-hand side of the equation, we say that the ODE solver is implicit. Example: Backward Euler.

If y_{n+1} is not on the right-hand side of the equation, we say that the ODE solver is explicit. Example: Euler.

Question: Implicit methods certainly cause more work. Are they worth it?

Answer: Their stability properties allow us to solve problems that are not easy with explicit methods.

Local error

Taylor series says that the local error is

$$\frac{h^2}{2} y''(\xi)$$

This is first order, just like Euler's method.

Global error

We know

$$y(t_{n+1}) = y(t_n) + h_n f(t_{n+1} y(t_{n+1})) + \frac{h^2}{2} y''(\xi_n), \quad n = 0, 1, 2, \ldots,$$

and

$$y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1}) \,.$$

Therefore,

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

- The plum-colored terms are global errors.
- The black term is the local error.
- Using the MVT, the blue term can be written as

$$h_n(f(t_{n+1}, y(t_{n+1})) - f(t_{n+1}, y_{n+1})) = h_n f_y(\eta)(y(t_{n+1}) - y_{n+1}) \,.$$

So we have the expression

$$(1 - h_n f_y(\eta))\ \text{(new global error)} = \text{(old global error)} + \text{local error}_n \quad .$$

so

$$\text{new global error} = (1 - h_n f_y(\eta))^{-1}[\ \text{(old global error)} + \text{local error}_n] \quad .$$

Therefore, the errors will be magnified if

$$|1 - h_n f_y(\eta)| < 1$$

and we say that the backward Euler method is unstable in this case.

So the stability interval for Backward Euler's method is

$$hf_y < 0 \ \text{ or } \ hf_y > 2$$

for a single equation. For a system of equations, the stability region is the region where all eigenvalues of $\mathbf{I} - h\mathbf{f}_y$ are outside the unit circle.

Example: Backward Euler is stable on the equation $y' = -y$ for all positive h.

Backward Euler is unstable for $y' = y$ when h is small and positive and inaccurate when h is large.

Boundary value problems for ODEs

Reference: Section 20.5

- Some basics
- Shooting methods
- finite difference methods

Some basics

In an initial value problem, all of the data values are given at a single point t_0.

In a boundary value problem, more than one point is involved.

Example:

$$\begin{aligned} u'' &= 6u' - tu + u^2 \\ u(0) &= 5 \\ u(1) &= 2 \end{aligned}$$

Find $u(t)$ for $t \in (0, 1)$.

If we convert this to a system of first order equations, we let $y_1 = u$, $y_2 = u'$, and

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= 6y_2 - ty_1 + y_1^2 \\ y_1(0) &= 5 \\ y_1(1) &= 2 \end{aligned}$$

So we have values of y_1 at 0 and 1. If we had values of y_1 and y_2 at 0, we could use our old (IVP) methods. But now we have a boundary value problem.

What to do?

There are two alternatives:

- adapt our old methods to this problem: shooting.
- develop new methods: finite differences.

Shooting methods

When in doubt, guess. The idea behind shooting methods is to guess at the missing initial values, integrate the equation using our favorite method, and then use the results to improve our guess.

In fact, we recognize this as a nonlinear system of equations: to solve our example problem,

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= 6y_2 - ty_1 + y_1^2 \\ y_1(0) &= 5 \\ y_1(1) &= 2 \end{aligned}$$

we want to solve the nonlinear equation

$$F(z) = 0$$

where z is the value we give to $y_2(0)$ and $F(z)$ is the difference between 2 and the value that our (IVP) ODE solver returns for $y_1(1)$.

So a shooting method involves using our favorite nonlinear equation solver, with function evaluation through our favorite IVP-ODE solver. Once we find the initial value z, then the IVP-ODE solver can give us values $\mathbf{y}(t)$ for any t.

Challenge: Write the code to solve this IVP using `ODE45` and `FZERO`.

Warnings

- If the IVP is difficult to solve (for example, stiff), then it will be difficult to get an accurate value of z.
- Our function evaluation for FZERO is noisy: it includes all of the round-off error and the global discretization error introduced by the IVP-ODE solver. The resulting wiggles in the values of F can cause the nonlinear equation solver to have trouble finding an accurate solution, and can also introduce multiple solutions where there is really only one.
- If the interval of integration is long, these difficulties can be overwhelming and we need to go to more complicated methods; for example, multiple shooting. See Ascher and Petzold for further discussion.

Finite difference methods

Unquiz: Suppose y has enough continuous derivatives. Prove that

$$\begin{aligned} y'(t) &= \frac{y(t+h) - y(t-h)}{2h} + O(h^2), \\ y''(t) &= \frac{y(t-h) - 2y(t) + y(t+h)}{h^2} + O(h^2) \end{aligned}$$

for small values of h. []

Now consider our example problem in its original form:

$$\begin{aligned} u'' &= 6u' - tu + u^2 \\ u(0) &= 5 \\ u(1) &= 2 \end{aligned}$$

Given a large number n (for example, $n = 100$), let $h = 1/n$ and define

$$u_j \approx u(jh), \quad j = 0, \ldots, n.$$

Then we can approximate our original equation

$$u'' = 6u' - tu + u^2$$

at $t = t_j$ $(0 < j < n)$ by

$$\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} = 6\frac{u_{j+1} - u_{j-1}}{2h} - t_j u_j + u_j^2 .$$

Since we already know that

$$\begin{aligned} u_0 \approx u(0) &= 5 \\ u_n \approx u(1) &= 2 \end{aligned}$$

we have a system of $n-1$ nonlinear equations in $n-1$ unknowns and we can solve it using our favorite method.

The equations are sparse (matrix has band structure)

Final Words

- Initial value problems for ordinary differential equations that arise in practice can be very difficult to solve.
  - Beware of stiff equations.
  - If there is a conservation law or Hamiltonian, make sure to incorporate it into the formulation. (Otherwise, your customer will be very unhappy with the numerical results.) But be aware that if you don't do this in a smart way, it may cause the ODE solver to take very small steps.
- We have just touched on the existence, uniqueness, and stability theory for ODEs and DAEs. If you need to solve an important problem, be ready to study these issues further before you go to the computer.
- Numerical solution of DAEs is still an evolving science, so watch the literature if you are working in this field.
- For an alternate set of methods for solving ODEs, see J. C. Butcher, "General Linear Methods," *Acta Numerica* (15) 2006, 157-255.

Hamiltonian systems

Reference: Section 20.3

In some ODE systems, there is an associated conservation principle, and if possible, we formulate the problem so that conservation is observed.

Definition: A Hamiltonian system is one for which there exists a scalar Hamiltonian function $H(\mathbf{y})$ so that

$$\mathbf{y}' = \mathbf{D}\nabla_{\mathbf{y}} H(\mathbf{y}) \,,$$

where $\mathbf{D}$ is a block-diagonal matrix with blocks equal to

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} .$$

Example: Linear harmonic oscillator. Let $q(t)$ and $p(t)$ be unknown functions satisfying

$$\begin{aligned} q' &= \omega p \\ p' &= -\omega q \end{aligned}$$

where $\omega > 0$ is a fixed parameter.

The Hamiltonian of the system is defined to be

$$H = \frac{\omega}{2}(p^2 + q^2)\,.$$

To verify this, note that if $\mathbf{y} = [q, p]^T$, then

$$\nabla_{\mathbf{y}} H(\mathbf{y}) = \begin{bmatrix} \omega q \\ \omega p \end{bmatrix}$$

so that

$$\mathbf{y}' = \begin{bmatrix} \omega p \\ -\omega q \end{bmatrix} = \mathbf{D}\nabla_{\mathbf{y}} H(\mathbf{y}) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \omega q \\ \omega p \end{bmatrix}\,.$$

(See http://scienceworld.wolfram.com/physics/HamiltonsEquations.html for more information on Hamiltonian systems.)

Note that

$$\begin{aligned} H' &= \frac{\omega}{2}(2pp' + 2qq') \\ &= \frac{\omega}{2}(2\frac{q'}{\omega}p' + 2\frac{-p'}{\omega}q') \\ &= 0\,, \end{aligned}$$

so $H(t)$ must be constant; in other words, the quantity H is conserved or invariant.

We can verify this a different way by writing the general solution to the problem:

$$\begin{bmatrix} q(t) \\ p(t) \end{bmatrix} = \begin{bmatrix} \cos \omega t & \sin \omega t \\ -\sin \omega t & \cos \omega t \end{bmatrix} \begin{bmatrix} q(0) \\ p(0) \end{bmatrix}$$

and computing $p(t)^2 + q(t)^2$.

The eigenvalues of the matrix defining the solution are imaginary numbers, so a small perturbation of the matrix can cause the quantity H to either grow or shrink, and this will not produce a useful solution.

[]

Therefore, in solving systems involving Hamiltonians (conserved quantities), it is important to build conservation into the numerical method whenever possible!

Example: If the ODE has the form

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}) \\ h(\mathbf{y}) &= 0 \end{aligned}$$

(as in the previous example), then we can rewrite it as

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}) - \mathbf{g}(\mathbf{y})z \\ h(\mathbf{y}) &= 0 \end{aligned}$$

where $z(t)$ is a scalar function (added so that the system is not overdetermined) and $\mathbf{g}$ is any bounded function whose Jacobian matrix with respect to the variables $\mathbf{y}$ has an inverse that is bounded away from singularity for all t.

If we solve this system exactly, then we will get $z(t) = 0$ and we recover our original solution. But if we solve it numerically, the second equation forces z to be nonzero in order to keep the solution satisfying the conservation law $h(\mathbf{y}) = 0$.

For example, we can rewrite our harmonic oscillator example as

$$\begin{aligned} q' &= \omega p - \omega q z \\ p' &= -\omega q - \omega p z \\ 5 &= \frac{\omega}{2}(p^2 + q^2) \end{aligned}$$

(5 used as an example) []

Adding an invariant, or conservation law, generally changes the ODE system to a system that includes nonlinear equations not involving derivatives – a system of differential-algebraic equations (DAEs).

Warning: Sometimes, adding conservation makes the problem too expensive to solve; for example, if the solution is rapidly oscillating. In such cases, we may decide to allow the conservation law to be violated.

Next we'll consider a little of the theory and computation of DAEs.

Differential-Algebraic Equations

Reference: Section 20.4

- Some basics
- Some numerical methods

Some basics

The general DAE has the form

$$\mathbf{F}(t, \widehat{\mathbf{y}}(t), \widehat{\mathbf{y}}'(t)) = \mathbf{0},$$

Important special case:

$$\mathbf{M}(t)\mathbf{y}'(t) = \mathbf{A}(t)\mathbf{y}(t) + \mathbf{f}(t).$$

The $\mathbf{M}(t)$ is called the **mass matrix**.

- $\mathbf{M}(t)$ full rank $\longrightarrow$ system of ODEs.
- $\mathbf{M}(t) = \mathbf{0} \longrightarrow$ time-dependent system of linear equations.

DAEs are classified by several parameters:

- m_a is the number of algebraic conditions in the DAE.
- m_d is the number of differential conditions in the DAE, and $m_a + m_d = m$.
- ℓ is the **strangeness** of the DAE.

Often a fourth parameter is considered: the **differential-index** of a DAE is the number of differentiations needed to convert the problem to an (explicit) system of ODEs. A system of ODEs has differential-index 0, and a system of algebraic equations $\mathbf{F}(\mathbf{y}) = \mathbf{0}$ has differential-index 1.

Check existence and uniqueness using Pointer 20.6.
See Challenge 20.14 for an example.

A major difference between DAEs and ODEs

For ODEs, it is easy to count how many initial conditions we need to uniquely determine the solution.

For DAEs, it is not so simple. Some DAEs need no initial conditions.

And even if some initial conditions are necessary, it is difficult to determine whether the conditions given are consistent so that a solution exists.

Some numerical methods

The main idea: If we want to solve

$$\mathbf{F}(\mathbf{y}, \mathbf{y}', t) = \mathbf{0}\,,$$

then we can step from known values at $t = t_n, t_{n-1}, \ldots, t_{n-k}$ to unknown values at $t = t_{n+1}$ using our favorite approximation scheme to replace $\mathbf{y}'(t_{n+1})$ by

$$\mathbf{y}'(t_{n+1}) \approx \sum_{i=0}^{k} \alpha_i \mathbf{y}(t_{n-i})\,.$$

This gives us a nonlinear equation to solve for $y_{n+1} \approx \mathbf{y}(t_{n+1})$:

$$\mathbf{F}(\mathbf{y}_{n+1}, \sum_{i=0}^{k} \alpha_i \mathbf{y}(t_{n-i}), t_{n+1}) = \mathbf{0}\,.$$

We can solve this equation using our favorite method (Newton-like, homotopy, ...).

Complications

- Stability is an important consideration. The ODE method needs to be chosen carefully; usually a stiff method is needed.
- The nonlinear equation may fail to have a solution.
- Even if a solution exists, the method you choose for solving the nonlinear equation may fail to converge.
- Automatic control of order and stepsize is even more difficult than for ODEs.

Bottom line

Don't try to write your own solver for DAEs. Use high-quality software:

- The Matlab ODE solvers handle some DAEs. I believe they are well-written, but I don't have vast personal experience with them.
- See Pointer 20.7 for other software.

Final Words

- Initial value problems for ordinary differential equations that arise in practice can be very difficult to solve.
  - Beware of stiff equations.
  - If there is a conservation law or Hamiltonian, make sure to incorporate it into the formulation. (Otherwise, your customer will be very unhappy with the numerical results.) But be aware that if you don't do this in a smart way, it may cause the ODE solver to take very small steps.
- We have just touched on the existence, uniqueness, and stability theory for ODEs and DAEs. If you need to solve an important problem, be ready to study these issues further before you go to the computer.
- Numerical solution of DAEs is still an evolving science, so watch the literature if you are working in this field.
- For an alternate set of methods for solving ODEs, see J. C. Butcher, "General Linear Methods," *Acta Numerica* (15) 2006, 157-255.