Stiffness/ Implicit Euler

A scalar equation. We first consider a simple scalar equation

$$x' = -ax, \quad x(0) = 1$$

where $a > 0$ is a constant, possibly very large. The exact solution is

$$x(t) = e^{-at}.$$

This is an exponential decay. We see that

$$x \to 0 \text{ as } t \to +\infty.$$

Furthermore, the larger the value a, the faster the decay.

We now solve it by forward Euler's method:

$$x_0 = 1, \quad x_{n+1} = x_n - ahx_n = (1 - ah)x_n, \quad n \geq 1.$$

Simple induction argument shows that

$$x_n = (1 - ah)^n x_0 = (1 - ah)^n.$$

We expect that the numerical solution should preserve the important property (1), i.e.,

$$x_n \to 0, \text{ as } n \to +\infty.$$

We must require

$$|1 - ah| < 1, \quad \Rightarrow \quad h < \frac{2}{a}.$$

This gives a restriction to the time step size h, i.e., h must be sufficiently small. The larger the value of a, the smaller h must be, even though the solution is almost 0 after a very short time!

To improve the stability, we now use the implicit Euler step:

$$x_0 = 1, \quad x_{n+1} = x_n - ahx_{n+1}, \quad n \geq 1.$$

This implies

$$x_{n+1} = \frac{1}{1 + ah} x_n.$$

Simple induction argument shows that for all $n \geq 0$, we have

$$x_n = \left(\frac{1}{1 + ah}\right)^n.$$

Since $ah > 0$, we have

$$0 < \frac{1}{1 + ah} < 1$$

leading to

$$\lim_{n \to +\infty} x_n = 0$$

for any values of h. This is called unconditionally stable.

Remark. Stability condition occurs also for nonlinear equations. But if one applies an implicit method, it becomes unconditionally stable, but at a price. Consider the general equation

$$x' = f(t, x), \quad x(t_0) = x_0.$$

where $f(t, x)$ is nonlinear in x. The implicit Euler step becomes

$$x_{n+1} = x_n + h \cdot f(t_{n+1}, x_{n+1}).$$

We see that this becomes a non-linear equation for x_{n+1}, which may or may not have solutions, or multiple solutions. An approximate solution could be obtained using a possible Newton iteration or some varieties of it. It can be very time consuming.

In [1]:
```
h = 0.0001  # step size
t_max = 1.0  # maximum time
x_0 = 1  # initial condition
n_steps = int(t_max / h)

t_values = [h * i for i in range(n_steps + 1)]
x_values = [x_0]

for i in range(n_steps):
    x_new = x_values[-1] * (1 - 1000 * h)
    x_values.append(x_new)

x_values[-10:]
```

Out[1]:
```
[2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323,
 2.5e-323]
```

Two-point BVP

We now consider a second order ODE in the form

$$y''(x) = f\left(x, y(x), y'(x)\right), \quad y(a) = \alpha, \quad y(b) = \beta.$$

Here $y(x)$ is the unknown function defined on the interval $a \leq x \leq b$. The values of y at the boundary points $x = a, x = b$ are given.

Such differential equations arise in many physical models. For example, the model for an elastic string:

$$y'' = ky + mx(x - L), \quad y(0) = 0, \quad y(L) = 0.$$

Note that this equation is linear.

We study two numerical methods for this two-point boundary value problem:

- Shooting method: based on ODE solvers;
- Finite Difference Method (FDM).

Shooting method (Linear)

Given some two-point boundary value problem on $a \leq x \leq b$. Main algorithm:

- Solve two-point boundary value problem as an initial value problem, with initial data given at $x = a$ (a guess).
- Compute the solution and the value in the solution at $x = b$.
- Compare this with the given boundary condition at $x = b$. Then adjust your guess at $x = a$ and iterate if needed.

It makes a difference if the differential equation is linear and nonlinear. The linear case is simpler.

Let's consider the linear problem in the general form:

$$y''(x) = u(x) + v(x)y(x) + w(x)y'(x), \quad y(a) = \alpha, \quad y(b) = \beta.$$

Let $\bar{y}$ solve the same equation, but with initial conditions:

$$\bar{y}''(x) = u(x) + v(x)\bar{y}(x) + w(x)\bar{y}'(x), \quad \bar{y}(a) = \alpha, \quad \bar{y}'(a) = 0.$$

Note that $\bar{y}'(a) = 0$ is the "guess" we make.

Let $\tilde{y}$ solve the same equation, but with different initial conditions:

$$\tilde{y}''(x) = u(x) + v(x)\tilde{y}(x) + w(x)\tilde{y}'(x), \quad \tilde{y}(a) = \alpha, \quad \tilde{y}'(a) = 1.$$

Note that $\bar{y}'(a) = 1$ is the other "guess" we make.

As we will see later, it doesn't matter with guesses we make here. Any numbers will work, as long as they are different for $\bar{y}$ and $\tilde{y}$.

Now let

$$y(x) = \lambda \cdot \bar{y}(x) + (1 - \lambda) \cdot \tilde{y}(x)$$

where λ is a constant to be determined, such that $y(x)$ becomes the solution.

We now check which equation y solves the DE. We have

$$\begin{aligned}
y'' &= \lambda \cdot \bar{y}''(x) + (1 - \lambda) \cdot \tilde{y}''(x) \\
&= \lambda \left(u + v\bar{y} + w\bar{y}'\right) + (1 - \lambda) \left(u + v\tilde{y} + w\tilde{y}'\right) \\
&= u + v(\lambda\bar{y} + (1 - \lambda)\tilde{y}) + w \left(\lambda\bar{y}' + (1 - \lambda)\tilde{y}'\right) \\
&= u + vy + wy'.
\end{aligned}$$

We now check the boundary conditions. At $x = a$, we have

$$y(a) = \lambda\bar{y}(a) + (1 - \lambda)\tilde{y}(a) = \lambda\alpha + (1 - \lambda)\alpha = \alpha.$$

The boundary condition is satisfied for any choices of λ.

At $x = b$, we have

$$y(b) = \lambda \bar{y}(b) + (1-\lambda)\tilde{y}(b).$$

Since we must require $y(b) = \beta$, this gives us a equation to find λ,

$$\lambda \bar{y}(b) + (1-\lambda)\tilde{y}(b) = \beta, \quad \Rightarrow \quad \lambda = \frac{\beta - \tilde{y}(b)}{\bar{y}(b) - \tilde{y}(b)}.$$

Conclusion. The $y(x)$ given as

$$y(x) = \lambda \cdot \bar{y}(x) + (1-\lambda) \cdot \tilde{y}(x)$$

with λ given as

$$\lambda = \frac{\beta - \tilde{y}(b)}{\bar{y}(b) - \tilde{y}(b)}$$

is the solution of the BVP.

Extensions

Case 1. We now consider the effect of different boundary conditions.

$$y''(x) = u(x) + v(x)y(x) + w(x)y'(x), \quad y(a) = \alpha, \quad y'(b) = \beta.$$

A same shooting method can be designed, with minimum adjustment for the boundary condition at $x = b$. The function y in the general algorithm will satisfy the differential equation as well as the boundary condition at $x = a$. For the boundary condition at $x = b$, we must require

$$y'(b) = \lambda \bar{y}'(b) + (1-\lambda)\tilde{y}'(b) = \beta, \quad \Rightarrow \quad \lambda = \frac{\beta - \tilde{y}'(b)}{\bar{y}'(b) - \tilde{y}'(b)}.$$

Case 2. Consider a higher order linear equation

$$y''' = f\left(x, y, y', y''\right), \quad y(a) = \alpha, \quad y'(a) = \gamma, \quad y(b) = \beta.$$

Here $f\left(x, y, y', y''\right)$ is an affine function in y, y', y''. A shooting method can be designed as follows. Let $\bar{y}$ and $\tilde{y}$ solve the same equation (1), but with initial conditions:

$$\begin{aligned} \bar{y}(a) = \alpha, \quad \bar{y}'(a) = \gamma, \quad \bar{y}''(a) = 0. \\ \tilde{y}(a) = \alpha, \quad \tilde{y}'(a) = \gamma, \quad \tilde{y}''(a) = 1. \end{aligned}$$

Assume now we solved both equations (2) and (3), and the values $\bar{y}(b)$ and $\tilde{y}(b)$ are computed. Let

$$y(x) = \lambda \cdot \bar{y}(x) + (1-\lambda) \cdot \tilde{y}(x)$$

where λ is a constant to be determined, such that $y(x)$ in (4) becomes the solution for (1). It is easy to check that y solves the equation in (1), and satisfies the boundary conditions $y(a) = \alpha, y'(a) = \gamma$, due to the linear properties. It remains to check the last boundary condition at $x = b$. At $x = b$, we have

$$y(b)=\lambda \bar{y}(b)+(1-\lambda) \tilde{y}(b)=\beta .$$

which give the same formula to compute λ, i.e,

$$\lambda=\frac{\beta-\tilde{y}(b)}{\bar{y}(b)-\tilde{y}(b)} .$$

Nonlinear shooting method

We now consider the general nonlinear equation

$$y^{\prime \prime}=f\left(x, y, y^{\prime}\right), \quad y(a)=\alpha, \quad y(b)=\beta$$

Let $\tilde{y}$ solve the IVP

$$\tilde{y}^{\prime \prime}=f\left(x, \tilde{y}, \tilde{y}^{\prime}\right), \quad \tilde{y}(a)=\alpha, \quad \tilde{y}^{\prime}(a)=z .$$

Note that the condition $\tilde{y}^{\prime}(a)=z$ is our guess.

The solution of (1) depends on z. Denote

$$\tilde{y}(b) \doteq \phi(z),$$

where ϕ is a non-linear function denoting the relation on how the value $\tilde{y}(b)$ depend on z. We need to find the value z such that

$$\phi(z)=\beta, \quad \Rightarrow \phi(z)-\beta=0 .$$

Since $\phi(z)$ is a non-linear function, we need to find a root for the above nonlinear equation. One can use secant method.

The algorithm goes as follows.

1. Choose some initial guess z_1, z_2, and compute the values
$$\phi_1=\phi\left(z_1\right), \quad \phi_2=\phi\left(z_2\right)$$
2. Then, the next value z_3 could be computed by a secant step:
$$z_3=z_2+\left(\beta-\phi_2\right) \cdot \frac{z_2-z_1}{\phi_2-\phi_1} .$$
3. One can then iterate and get values $z_4, z_5, \cdots$ until converges.

Finite Difference method for two-point boundary value problem

We consider the linear problem, in the general form, with Dirichlet boundary condition

$$y^{\prime \prime}(x)=u(x)+v(x) y(x)+w(x) y^{\prime}(x), \quad y(a)=\alpha, \quad y(b)=\beta .$$

Discretize the domain: Choose n, make a uniform grid:

$$h=\frac{b-a}{n}, \quad x_i=a+ih, \quad i=0,1,2,\cdots,n, \quad x_0=a, \quad x_n=b$$

Goal: Find approximations $y_i \approx y\left(x_i\right)$.

Tool: finite difference approximation to the derivatives:

$$\begin{aligned}
y'\left(x_i\right) &\approx \frac{y\left(x_{i+1}\right)-y\left(x_{i-1}\right)}{x_{i+1}-x_{i-1}}=\frac{y_{i+1}-y_{i-1}}{2h}, \\
y''\left(x_i\right) &\approx \frac{y_{i+1}-2y_i+y_{i-1}}{h^2}.
\end{aligned}$$

Plug these into the ODE $y''(x)=u(x)+v(x)y(x)+w(x)y'(x)$, we get

$$\frac{1}{h^2}\left(y_{i+1}-2y_i+y_{i-1}\right)=u_i+v_iy_i+\frac{w_i}{2h}\left(y_{i+1}-y_{i-1}\right),$$

for $i=1,2,\cdots n-1$, where we used the notation

$$u_i=u\left(x_i\right), \quad v_i=v\left(x_i\right), \quad w_i=w\left(x_i\right)$$

We can clean up a bit, and get

$$-\left(1+\frac{h}{2}w_i\right)y_{i-1}+\left(2+h^2v_i\right)y_i-\left(1-\frac{h}{2}w_i\right)y_{i+1}=-h^2u_i$$

Calling

$$a_i=-\left(1+\frac{h}{2}w_i\right), \quad d_i=\left(2+h^2v_i\right), \quad c_i=-\left(1-\frac{h}{2}w_i\right), \quad b_i=-h^2u_i$$

discrete equations can be written in a simpler way

$$a_iy_{i-1}+d_iy_i+c_iy_{i+1}=b_i, \quad i=1,2,\cdots,n-1.$$

By the boundary conditions $y_0=\alpha, y_n=\beta$, the first and last equation become

$$d_1y_1+c_1y_2=b_1-a_1\alpha$$

The discrete equations

$$a_iy_{i-1}+d_iy_i+c_iy_{i+1}=b_i, \quad i=1,2,\cdots,n-1,$$

lead to a tri-diagonal system of linear equations.

$$A\vec{y}=\vec{b}$$

with

$$\begin{pmatrix}
d_1 & c_1 & & & \\
a_2 & d_2 & c_2 & & \\
 & \ddots & \ddots & \ddots & \\
 & & a_{n-2} & d_{n-2} & c_{n-2} \\
 & & & a_{n-1} & d_{n-1}
\end{pmatrix}\cdot\begin{pmatrix}
y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1}
\end{pmatrix}=\begin{pmatrix}
b_1-a_1\alpha \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1}-c_{n-1}\beta
\end{pmatrix}$$

Example

Example Set up the FDM for the problem

$$y'' = -4(y-x), \quad y(0)=0, \quad y(1)=2.$$

Note that the exact solution is $y(x)=(1/\sin 2)\sin 2x + x$. Answer. Fix an n, we make a uniform grid:

$$h=\frac{1}{n}, \quad x_i = ih, \quad i=0,1,2,\cdots n.$$

Central Finite Difference for the second derivative $y''(x_i)$ gives us

$$y''(x_i) \approx \frac{1}{h^2}(y_{i-1}-2y_i+y_{i+1}) = -4y_i + 4x_i.$$

After some cleaning up, we get

$$y_{i-1} - \left(2-4h^2\right) y_i + y_{i+1} = 4h^2 x_i, \quad i=1,2,\cdots,n-1,$$

with boundary conditions

$$y_0 = 0, \quad y_n = 2.$$

We end up with the tri-diagonal system $A\vec{y}=\vec{b}$:

$$A = \begin{pmatrix} -2+4h^2 & 1 & & & \\ 1 & -2+4h^2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2+4h^2 & 1 \\ & & & 1 & -2+4h^2 \end{pmatrix}$$

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} 4h^2 x_1 - a \\ 4h^2 x_2 \\ \vdots \\ 4h^2 x_{n-2} \\ 4h^2 x_{n-1} - b \end{pmatrix}.$$

Neumann and Robin Boundary Condition

Neumann Boundary condition is when the derivative of the unknown is given at the boundary. For example, we consider the Poisson equation in 1D:

$$u''(x) = f(x), \quad u'(0)=a, \quad u(1)=b.$$

Note the condition at $x=0$ is given as the derivative of the unknown $u(x)$.

Uniform grid: Fix an N, let $h=1/N$ and $x_i=ih$ for $i=0,1,2,\cdots,N$, and let $u_i \approx u(x_i)$ be the approximation.

We now have N unknowns, namely $u_0, u_1, \cdots, u_{N-1}$. We also have $u_N = b$ which is the Dirichlet boundary condition.

We set up the finite difference scheme

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i), \quad \Rightarrow \quad u_{i-1} - 2u_i + u_{i+1} = h^2 f(x_i),$$

which holds for $i = 1, 2 \cdots, N-1$.

Since the central finite difference approximation to $u''(x)$ is second order, we want also to approximate the boundary condition $u'(0) = a$ with a second order finite difference.

The central finite difference for $u'(0)$ is second, but it requires information at $x = -h$.

To handle this, we add an additional grid point outside the domain, $x_{-1} = x_0 - h = -h$.

This point is called ghost boundary.

Writing $u_{-1} \approx u(x_{-1})$, we now write out the central finite different for the boundary condition:

$$\frac{u_1 - u_{-1}}{2h} = a, \quad \Rightarrow \quad u_{-1} = u_1 - 2ha.$$

We also write out the central difference at $i = 0$:

$$u_{-1} - 2u_0 + u_1 = h^2 f(x_0)$$

we get the discrete equation for $i = 0$

$$u_1 - 2ha - 2u_0 + u_1 = h^2 f(x_0), \quad \Rightarrow \quad -2u_0 + 2u_1 = h^2 f(x_0) + 2ha.$$

The equation $i = N-1$ is slightly different due to the boundary condition $u_N = b$:

$$u_{N-2} - 2u_{N-1} = h^2 f(x_{N-1}) - b.$$

Collecting all the equation with $i = 0, 1, 2, \cdots, N-1$, we obtain the following tri-diagonal system of linear equations:

$$\begin{pmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} h^2 f(x_0) + 2ha \\ h^2 f(x_1) \\ \vdots \\ h^2 f(x_{N-2}) \\ h^2 f(x_{N-1}) - b \end{pmatrix}.$$

Robin boundary conditions can be handled in a similar way.

Lecture notes on ODEs based on Intro Numeric Comput (2nd Ed): Wen Shen: 9789811204418.

Review of Linear Algebra

Vector Space

If a set along with two operations (vector addition and scalar multiplication) satisfies all these axioms, then the set forms a vector space.

1. Closure under Addition: For any vectors $\mathbf{u}$ and $\mathbf{v}$ in the space, the sum $\mathbf{u}+\mathbf{v}$ is also in the space.
2. Closure under Scalar Multiplication: For any vector $\mathbf{u}$ in the space and any scalar c, the product $c\mathbf{u}$ is also in the space.
3. Additive Identity: There exists a vector $\mathbf{0}$ in the space such that for every vector $\mathbf{u}$ in the space, $\mathbf{u}+\mathbf{0}=\mathbf{u}$.
4. Additive Inverse: For every vector $\mathbf{u}$ in the space, there exists a vector $-\mathbf{u}$ in the space such that $\mathbf{u}+(-\mathbf{u})=\mathbf{0}$.
5. Associativity of Addition: For any vectors $\mathbf{u},\mathbf{v}$, and $\mathbf{w}$ in the space, $\mathbf{u}+(\mathbf{v}+\mathbf{w})=(\mathbf{u}+$ v) $+\mathbf{w}$.
6. Commutativity of Addition: For any vectors $\mathbf{u}$ and $\mathbf{v}$ in the space, $\mathbf{u}+\mathbf{v}=\mathbf{v}+\mathbf{u}$.
7. Distributivity of Scalar Multiplication with respect to Vector Addition: For any scalars a and b and any vector $\mathbf{u}$, $(a+b)\mathbf{u}=a\mathbf{u}+b\mathbf{u}$.
8. Distributivity of Scalar Multiplication with respect to Scalar Addition: For any scalar a and any vectors $\mathbf{u}$ and $\mathbf{v}$, $a(\mathbf{u}+\mathbf{v})=a\mathbf{u}+a\mathbf{v}$.
9. Compatibility of Scalar Multiplication with Field Multiplication: For any scalars a and b and any vector $\mathbf{u}$, $a(b\mathbf{u})=(ab)\mathbf{u}$.
10. Identity Element of Scalar Multiplication: For every vector $\mathbf{u}$, $1\mathbf{u}=\mathbf{u}$, where 1 is the multiplicative identity in the field of scalars.

Definition (Linear subspace)

A subset $W\subset V$ is a linear subspace of V if the W is again a linear space over the same field $\mathbb{F}$ of scalars.

Thus W is a linear subspace if $W\neq\emptyset$ and for all $u,v\in W$ and $a,b\in\mathbb{F}$ any linear combination of them is also in the subspace: $au+bv\in W$.

Finding the representation of a function or of data in a linear subspace is to project it onto only that subset of vectors. This may amount to finding an approximation, or to extracting (say) just the low-frequency structure of the data or signal.

Projecting onto a subspace is sometimes called dimensionality reduction.

Different transforms (we will talk about Fourier) can be regarded as "projections" into particular vector spaces.

Definition (Linear combinations and span)

If V is a linear space and $v_1,v_2,\ldots,v_n\in V$ are vectors in V then $u\in V$ is a linear combination of $v_1,v_2,\ldots,v_n$ if there exist scalars $a_1,a_2,\ldots,a_n\in\mathbb{F}$ such that

$$u=a_1v_1+a_2v_2+\cdots+a_nv_n.$$

We also define the span of a set of vectors as all such linear combinations:
$\operatorname{span}\{v_1,v_2,\ldots,v_n\}=\{u\in V: u \text{ is a linear combination of } v_1,v_2,\ldots,v_n\}$. Thus,

$W = \text{span}\{v_1, v_2, \ldots, v_n\}$ is a linear subspace of V.

The span of a set of vectors is "everything that can be represented" by linear combinations of them.