Numerical Analysis II

Xiaojing Ye, Math & Stat, Georgia State University

Section 1

Initial Value Problems for ODEs

IVP of ODE

We study numerical solution for initial value problem (IVP) of ordinary differential equations (ODE).

- A basic IVP:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \quad \text{for } a \leq t \leq b$$

with initial value $y(a) = \alpha$.

Remark

- *f is given and called the defining function of IVP.*
- *α is given and called the initial value.*
- *$y(t)$ is called the solution of the IVP if*
  - $y(a) = \alpha$;
  - $y'(t) = f(t, y(t))$ *for all* $t \in [a, b]$.

IVP of ODE

Example

The following is a basic IVP:

$$y' = y - t^2 + 1, \quad t \in [0, 2], \text{ and } y(0) = 0.5$$

- The defining function is $f(t, y) = y - t^2 + 1$.
- Initial value is $y(0) = 0.5$.
- The solution is $y(t) = (t+1)^2 - \frac{e^t}{2}$ because:
  - $y(0) = (0+1)^2 - \frac{e^0}{2} = 1 - \frac{1}{2} = \frac{1}{2}$;
  - We can check that $y'(t) = f(t, y(t))$:

$$y'(t) = 2(t+1) - \frac{e^t}{2}$$

$$f(t, y(t)) = y(t) - t^2 + 1 = (t+1)^2 - \frac{e^t}{2} - t^2 + 1 = 2(t+1) - \frac{e^t}{2}$$

IVP of ODE (cont.)

More general or complex cases:

- IVP of ODE system:

$$\begin{cases} \dfrac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_n) \\ \dfrac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_n) \\ \quad \vdots \\ \dfrac{dy_n}{dt} = f_n(t, y_1, y_2, \ldots, y_n) \end{cases} \quad \text{for } a \le t \le b$$

  with initial value $y_1(a) = \alpha_1, \ldots, y_n(a) = \alpha_n$.

- High-order ODE:

$$y^{(n)} = f(t, y, y', \ldots, y^{(n-1)}) \quad \text{for } a \le t \le b$$

with initial value $y(a) = \alpha_1, y'(a) = \alpha_2, \ldots, y^{(n-1)}(a) = \alpha_n$.

Why numerical solutions for IVP?

- ODEs have extensive applications in real-world: science, engineering, economics, finance, public health, etc.
- Analytic solution? Not with almost all ODEs.
- Fast improvement of computers.

Some basics about IVP

Definition (Lipschitz functions)

A function $f(t, y)$ defined on $D = \{(t, y) : t \in \mathbb{R}_+, y \in \mathbb{R}\}$ is called **Lipschitz with respect to** *y if there exists a constant $L > 0$*

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

for all $t \in \mathbb{R}_+$, and $y_1, y_2 \in \mathbb{R}$.

Remark

We also call f is Lipschitz with respect to y with constant L, or simply f is L-Lipschitz with respect to y.

Some basics about IVP

Example

Function $f(t, y) = t|y|$ *is Lipschitz with respect to* y *on the set* $D := \{(t, y)|t \in [1, 2], y \in [-3, 4]\}$.

Solution: *For any* $t \in [1, 2]$ *and* $y_1, y_2 \in [-3, 4]$, *we have*

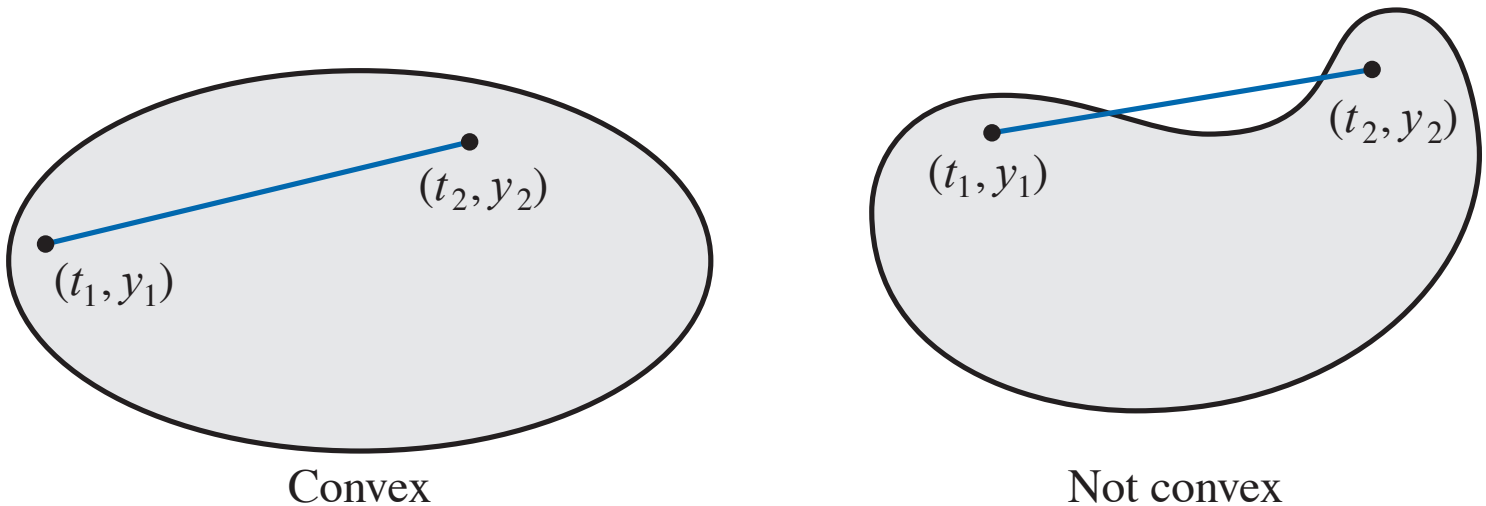
$$|f(t, y_1) - f(t, y_2)| = \big|t|y_1| - t|y_2|\big| \leq t|y_1 - y_2| \leq 2|y_1 - y_2|.$$

So $f(t, y) = t|y|$ *is Lipschitz with respect to* y *with constant* $L = 2$.

Some basics about IVP

Definition (Convex sets)

A set $D \in \mathbb{R}^2$ *is* **convex** *if whenever* $(t_1, y_1), (t_2, y_2) \in D$ *there is* $(1-\lambda)(t_1, y_1) + \lambda(t_2, y_2) \in D$ *for all* $\lambda \in [0, 1]$.



Some basics about IVP

Theorem

If $D \subset \mathbb{R}^2$ is convex, and $|\frac{\partial f}{\partial y}(t, y)| \leq L$ for all $(t, y) \in D$, then f is Lipschitz with respect to y with constant L.

Remark

This is a sufficient (but not necessary) condition for f to be Lipschitz with respect to y.

Some basics about IVP

Proof.

For any $(t, y_1), (t, y_2) \in D$, define function g by

$$g(\lambda) = f(t, (1-\lambda)y_1 + \lambda y_2)$$

for $\lambda \in [0, 1]$ (need convexity of D!). Then we have

$$g'(\lambda) = \partial_y f(t, (1-\lambda)y_1 + \lambda y_2) \cdot (y_2 - y_1)$$

So $|g'(\lambda)| \leq L|y_2 - y_1|$. Then we have

$$|g(1) - g(0)| = \left| \int_0^1 g'(\lambda)\, \mathrm{d}\lambda \right| \leq L|y_2 - y_1| \left| \int_0^1 \mathrm{d}\lambda \right| = L|y_2 - y_1|$$

Note that $g(0) = f(t, y_1)$ and $g(1) = f(t, y_2)$. This completes the proof. □

Some basics about IVP

Theorem

Suppose $D = [a, b] \times \mathbb{R}$, a function f is continuous on D and Lipschitz with respect to y, then the initial value problem $y' = f(t, y)$ for $t \in [a, b]$ with initial value $y(a) = \alpha$ has a unique solution $y(t)$ for $t \in [a, b]$.

Remark

This theorem says that there must be one and only one solution of the IVP, provided that the defining f of the IVP is continuous and Lipschitz with respect to y on D.

Some basics about IVP

Example

Show that $y' = 1 + t\sin(ty)$ *for* $t \in [0, 2]$ *with* $y(0) = 0$ *has a unique solution.*

Solution: First, we know $f(t, y) = 1 + t\sin(ty)$ is continuous on $[0, 2] \times \mathbb{R}$. Second, we can see

$$\left|\frac{\partial f}{\partial y}\right| = \left|t^2\cos(ty)\right| \leq |t^2| \leq 4$$

So $f(t, y)$ is Lipschitz with respect to y (with constant 4). From theorem above, we know the IVP has a unique solution $y(t)$ on $[0, 2]$.

Some basics about IVP

Definition (Well-posedness)

An IVP $y' = f(t, y)$ for $t \in [a, b]$ with $y(a) = \alpha$ is called **well-posed** if

- It has a unique solution $y(t)$;
- There exist $\epsilon_0 > 0$ and $k > 0$, such that $\forall \epsilon \in (0, \epsilon_0)$ and function $\delta(t)$, which is continuous and satisfies $|\delta(t)| < \epsilon$ for all $t \in [a, b]$, the perturbed problem $z' = f(t, z) + \delta(t)$ with initial value $z(a) = \alpha + \delta_0$ (where $|\delta_0| \leq \epsilon$) satisfies

$$|z(t) - y(t)| < k\epsilon, \quad \forall t \in [a, b].$$

Remark

This theorem says that a small perturbation on defining function f by $\delta(t)$ and initial value $y(a)$ by δ_0 will only cause small change to original solution $y(t)$.

Some basics about IVP

Theorem

Let $D = [a, b] \times \mathbb{R}$. If f is continuous on D and Lipschitz with respect to y, then the IVP is well-posed.

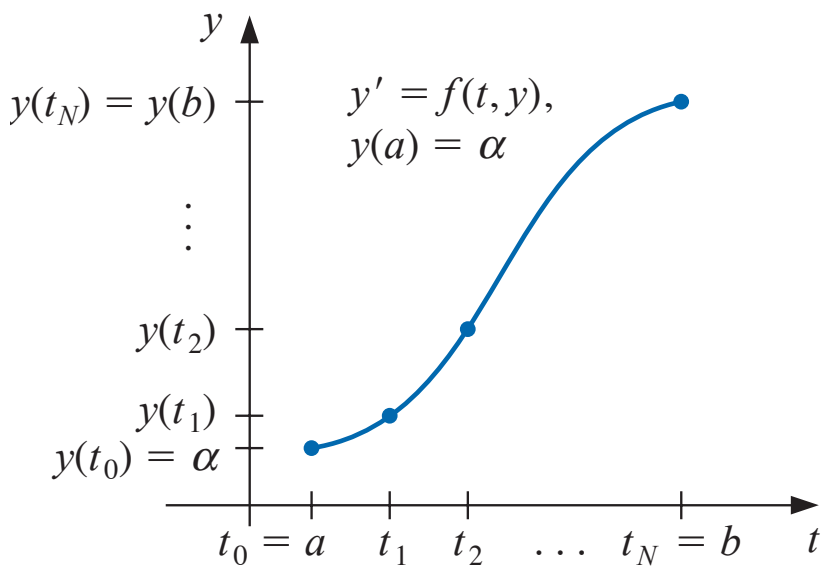
Remark

Again, a sufficient but not necessary condition for well-posedness of IVP.

Euler's method

Given an IVP $y' = f(t, y)$ for $t \in [a, b]$ and $y(a) = \alpha$, we want to compute $y(t)$ on **mesh points** $\{t_0, t_1, \ldots, t_N\}$ on $[a, b]$.

To this end, we partition $[a, b]$ into N equal segments: set $h = \frac{b-a}{N}$, and define $t_i = a + ih$ for $i = 0, 1, \ldots, N$. Here h is called the **step size**.



Euler's method

From Taylor's theorem, we have

$$y(t_{i+1}) = y(t_i) + y'(t_i)(t_{i+1} - t_i) + \frac{1}{2}y''(\xi_i)(t_{i+1} - t_i)^2$$

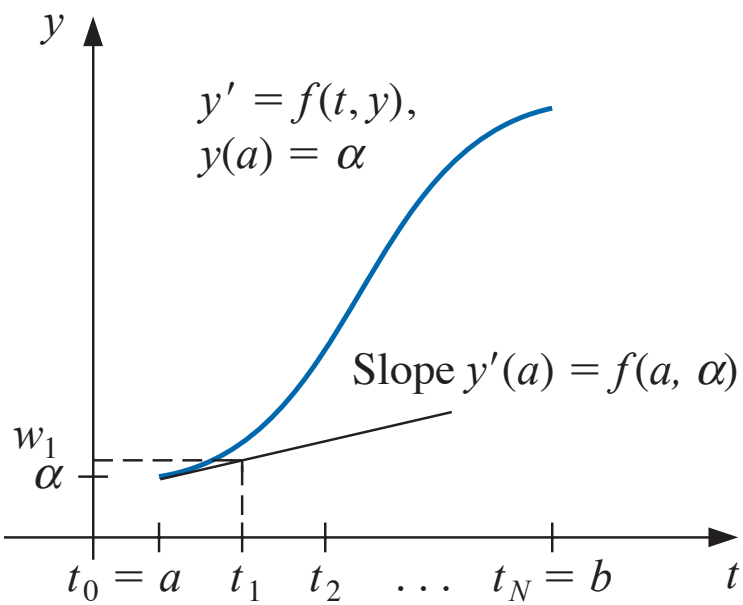
for some $\xi_i \in (t_i, t_{i+1})$. Note that $t_{i+1} - t_i = h$ and $y'(t_i) = f(t_i, y(t_i))$, we get

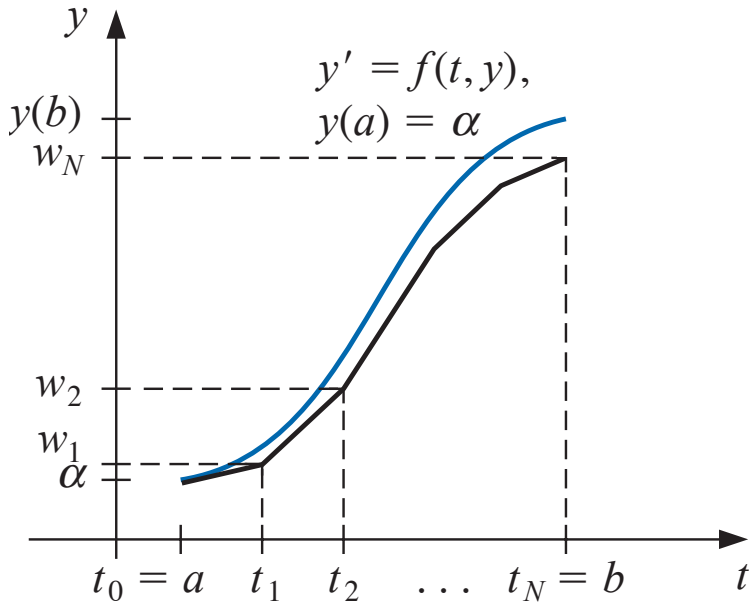
$$y(t_{i+1}) \approx y(t_i) + hf(t, y(t_i))$$

Denote $w_i = y(t_i)$ for all $i = 0, 1, \ldots, N$, we get the **Euler's method**:

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + hf(t_i, w_i), \quad i = 0, 1, \ldots, N-1 \end{cases}$$

Euler's method





Euler's method

Example

Use Euler's method with $h = 0.5$ *for IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *with initial value* $y(0) = 0.5$.

Solution: We follow Euler's method step-by-step:

$$
\begin{aligned}
t_0 = 0: &\quad w_0 = y(0) = 0.5 \\
t_1 = 0.5: &\quad w_1 = w_0 + hf(t_0, w_0) = 0.5 + 0.5 \times (0.5 - 0^2 + 1) = 1.25 \\
t_2 = 1.0: &\quad w_2 = w_1 + hf(t_1, w_1) = 1.25 + 0.5 \times (1.25 - 0.5^2 + 1) = 2.25 \\
t_3 = 1.5: &\quad w_3 = w_2 + hf(t_2, w_2) = 2.25 + 0.5 \times (2.25 - 1^2 + 1) = 3.375 \\
t_4 = 2.0: &\quad w_4 = w_3 + hf(t_3, w_3) = 3.375 + 0.5 \times (3.375 - 1.5^2 + 1) = 4.4375
\end{aligned}
$$

Error bound of Euler's method

Theorem

Suppose $f(t, y)$ in an IVP is continuous on $D = [a, b] \times \mathbb{R}$ and Lipschitz with respect to y with constant L. If $\exists M > 0$ such that $|y''(t)| \leq M$ ($y(t)$ is the unique solution of the IVP), then for all $i = 0, 1, \ldots, N$ there is

$$\left|y(t_i) - w_i\right| \leq \frac{hM}{2L}\left(e^{L(t_i - a)} - 1\right)$$

Remark

- *Numerical error depends on h (also called $O(h)$ error).*
- *Also depends on M, L of f.*
- *Error increases for larger t_i.*

Error bound of Euler's method

Proof. Taking the difference of

$$y(t_{i+1}) = y(t_i) + hf(t_i, y_i) + \frac{1}{2}y''(\xi_i)(t_{i+1} - t_i)^2$$
$$w_{i+1} = w_i + hf(t_i, w_i)$$

we get

$$\begin{aligned} |y(t_{i+1}) - w_{i+1}| &\leq |y(t_i) - w_i| + h|f(t_i, y_i) - f(t_i, w_i)| + \frac{Mh^2}{2} \\ &\leq |y(t_i) - w_i| + hL|y_i - w_i| + \frac{Mh^2}{2} \\ &= (1 + hL)|y_i - w_i| + \frac{Mh^2}{2} \end{aligned}$$

Error bound of Euler's method

Proof (cont).
Denote $d_i = |y(t_i) - w_i|$, then we have

$$d_{i+1} \le (1+hL)d_i + \frac{Mh^2}{2} = (1+hL)\left(d_i + \frac{hM}{2L}\right) - \frac{hM}{2L}$$

for all $i = 0, 1, \ldots, N-1$. So we obtain

$$\begin{aligned} d_{i+1} + \frac{hM}{2L} &\le (1+hL)\left(d_i + \frac{hM}{2L}\right) \\ &\le (1+hL)^2\left(d_{i-1} + \frac{hM}{2L}\right) \\ &\le \cdots \\ &\le (1+hL)^{i+1}\left(d_0 + \frac{hM}{2L}\right) \end{aligned}$$

and hence $d_i \le (1+hL)^i \cdot \frac{hM}{2L} - \frac{hM}{2L}$ (since $d_0 = 0$).

Error bound of Euler's method

Proof (cont).
Note that $1 + x \leq e^x$ for all $x > -1$, and hence $(1 + x)^a \leq e^{ax}$ if $a > 0$.
Based on this, we know $(1 + hL)^i \leq e^{ihL} = e^{L(t_i - a)}$ since $ih = t_i - a$. Therefore we get

$$d_i \leq e^{L(t_i - a)} \cdot \frac{hM}{2L} - \frac{hM}{2L} = \frac{hM}{2L}(e^{L(t_i - a)} - 1)$$

This completes the proof.

Error bound of Euler's method

Example

Estimate the error of Euler's method with $h = 0.2$ *for IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *with initial value* $y(0) = 0.5$.

Solution: We first note that $\frac{\partial f}{\partial y} = 1$, so f is Lipschitz with respect to y with constant $L = 1$. The IVP has solution $y(t) = (t-1)^2 - \frac{e^t}{2}$ so $|y''(t)| = |\frac{e^t}{2} - 2| \leq \frac{e^2}{2} - 2 =: M$. By theorem above, the error of Euler's method is

$$\left|y(t_i) - w_i\right| \leq \frac{hM}{2L}\left(e^{L(t_i - a)} - 1\right) = \frac{0.2(0.5e^2 - 2)}{2}\left(e^{t_i} - 1\right)$$

Error bound of Euler's method

Example

Estimate the error of Euler's method with $h = 0.2$ *for IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *with initial value* $y(0) = 0.5$.

Solution: (cont)

t_i	w_i	$y_i = y(t_i)$	$\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0.0000000
0.2	0.8000000	0.8292986	0.0292986
0.4	1.1520000	1.2140877	0.0620877
0.6	1.5504000	1.6489406	0.0985406
0.8	1.9884800	2.1272295	0.1387495
1.0	2.4581760	2.6408591	0.1826831
1.2	2.9498112	3.1799415	0.2301303
1.4	3.4517734	3.7324000	0.2806266
1.6	3.9501281	4.2834838	0.3333557
1.8	4.4281538	4.8151763	0.3870225
2.0	4.8657845	5.3054720	0.4396874

Round-off error of Euler's method

Due to round-off errors in computer, we instead obtain

$$\begin{cases} u_0 = \alpha + \delta_0 \\ u_{i+1} = u_i + hf(t_i, u_i) + \delta_i, \quad i = 0, 1, \ldots, N-1 \end{cases}$$

Suppose $\exists \delta > 0$ such that $|\delta_i| \leq \delta$ for all i, then we can show

$$|y(t_i) - u_i| \leq \frac{1}{L}\left(\frac{hM}{2} + \frac{\delta}{h}\right)\left(e^{L(t_i - a)} - 1\right) + \delta e^{L(t_i - a)}.$$

Note that $\frac{hM}{2} + \frac{\delta}{h}$ does not approach 0 as $h \to 0$. $\frac{hM}{2} + \frac{\delta}{h}$ reaches minimum at $h = \sqrt{\frac{2\delta}{M}}$ (often much smaller than what we choose in practice).

Higher-order Taylor's method

Definition (Local truncation error)

We call the difference method

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + h\phi(t_i, w_i), \quad i = 0, 1, \ldots, N-1 \end{cases}$$

to have **local truncation error**

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h}$$

where $y_i := y(t_i)$.

Example

Euler's method has local truncation error

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + hf(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i)$$

Higher-order Taylor's method

Note that Euler's method has local truncation error $\tau_{i+1}(h) = \frac{y_{i+1}-y_i}{h} - f(t_i, y_i) = \frac{hy''(\xi_i)}{2}$ for some $\xi_i \in (t_i, t_{i+1})$. If $|y''| \leq M$ we know $|\tau_{i+1}(h)| \leq \frac{hM}{2} = O(h)$.

Question: What if we use higher-order Taylor's approximation?

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i) + R$$

where $R = \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_i)$ for some $\xi_i \in (t_i, t_{i+1})$.

Higher-order Taylor's method

First note that we can always write $y^{(n)}$ using $f(t, y(t))$:

$$
\begin{aligned}
y'(t) &= f \\
y''(t) &= f' = \partial_t f + (\partial_y f) f \\
y'''(t) &= f'' = \partial_t^2 f + (\partial_t \partial_y f + (\partial_y^2 f) f) f + \partial_y f (\partial_t f + (\partial_y f) f) \\
&\dots \\
y^{(n)}(t) &= f^{(n-1)} = \dots
\end{aligned}
$$

albeit it's quickly getting very complicated.

Higher-order Taylor's method

Now substitute them back to high-order Taylor's approximation (ignore residual R)

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i) \\ &= y(t_i) + hf + \frac{h^2}{2}f' + \cdots + \frac{h^n}{n!}f^{(n-1)} \end{aligned}$$

We can get the n-**th order Taylor's method**:

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + hT^{(n)}(t_i, w_i), \quad i = 0, 1, \ldots, N-1 \end{cases}$$

where

$$T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}f'(t_i, w_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, w_i)$$

Higher-order Taylor's method

- Euler's method is the first order Taylor's method.
- High-order Taylor's method is more accurate than Euler's method, but at much higher computational cost.
- Together with Hermite interpolating polynomials, it can be used to interpolate values not on mesh points more accurately.

Higher-order Taylor's method

Theorem

If $y(t) \in C^{n+1}[a, b]$, then the n-th order Taylor method has local truncation error $O(h^n)$.

Runge-Kutta (RK) method

Runge-Kutta (RK) method attains high-order local truncation error **without** expensive evaluations of derivatives of f.

Runge-Kutta (RK) method

To derive RK method, first recall Taylor's formula for two variables (t, y):

$$f(t, y) = P_n(t, y) + R_n(t, y)$$

where $\partial_t^{n-k}\partial_y^k f = \frac{\partial^n f(t_0,y_0)}{\partial t^{n-k}\partial y^k}$ and

$$\begin{aligned} P_n(t, y) = & f(t_0, y_0) + (\partial_t f \cdot (t - t_0) + \partial_y f \cdot (y - y_0)) \\ & + \frac{1}{2}\left(\partial_t^2 f \cdot (t - t_0)^2 + 2\partial_y\partial_t f \cdot (t - t_0)(y - y_0) + \partial_y^2 f \cdot (y - y_0)^2\right) \\ & + \cdots + \frac{1}{n!}\sum_{k=0}^{n}\binom{n}{k}\partial_t^{n-k}\partial_y^k f \cdot (t - t_0)^{n-k}(y - y_0)^k \end{aligned}$$

$$R_n(t, y) = \frac{1}{(n+1)!}\sum_{k=0}^{n+1}\binom{n+1}{k}\partial_t^{n+1-k}\partial_y^k f(\xi, \mu) \cdot (t - t_0)^{n+1-k}(y - y_0)^k$$

Runge-Kutta (RK) method

The second order Taylor's method uses

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2}f'(t,y) = f(t,y) + \frac{h}{2}(\partial_t f + \partial_y f \cdot f)$$

to get $O(h^2)$ error. Suppose we use $af(t+\alpha, y+\beta)$ (with some

a, α, β to be determined) to reach the same order of error. To that end, we first have

$$af(t+\alpha, y+\beta) = a\left(f + \partial_t f \cdot \alpha + \partial_y f \cdot \beta + R\right)$$

where $R = \frac{1}{2}(\partial_t^2 f(\xi,\mu)\cdot\alpha^2 + 2\partial_y\partial_t f(\xi,\mu)\cdot\alpha\beta + \partial_y^2 f(\xi,\mu)\cdot\beta^2)$.

Runge-Kutta (RK) method

Suppose we try to match the terms of these two formulas (ignore R):

$$T^{(2)}(t,y) = f + \frac{h}{2}\partial_t f + \frac{hf}{2}\partial_y f$$
$$af(t+\alpha, y+\beta) = af + a\alpha\partial_t f + a\beta\partial_y f$$

then we have

$$a = 1, \quad \alpha = \frac{h}{2}, \quad \beta = \frac{h}{2}f(t,y)$$

So instead of $T^{(2)}(t,y)$, we use

$$af(t+\alpha, y+\beta) = f\left(t + \frac{h}{2}, y + \frac{h}{2}f(t,y)\right)$$

Runge-Kutta (RK) method

Note that R we ignored is

$$R = \frac{1}{2}\left(\partial_t^2 f(\xi,\mu)\cdot\left(\frac{h}{2}\right)^2 + 2\partial_y\partial_t f(\xi,\mu)\cdot\left(\frac{h}{2}\right)^2 f + \partial_y^2 f(\xi,\mu)\cdot\left(\frac{h}{2}\right)^2 f^2\right)$$

which means $R = O(h^2)$.

Also note that

$$R = T^{(2)}(t,y) - f\left(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)\right) = O(h^2)$$

and the error of $T^{(2)}(t,y)$ is of $O(h^2)$, we know
$f\left(t+\frac{h}{2}, y+\frac{h}{2}f(t,y)\right)$ has error of $O(h^2)$.

Runge-Kutta (RK) method

This is the **RK2 method (Midpoint method)**:

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + h\, f\left(t_i + \dfrac{h}{2}, w_i + \dfrac{h}{2} f(t_i, w_i)\right), \quad i = 0, 1, \ldots, N-1. \end{cases}$$

Remark

If we have (t_i, w_i), we only need to evaluate f twice (i.e., compute $k_1 = f(t_i, w_i)$ and $k_2 = f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_1)$) to get w_{i+1}.

Runge-Kutta (RK) method

We can also consider higher-order RK method by fitting

$$T^{(3)}(t,y) = f(t,y) + \frac{h}{2}f'(t,y) + \frac{h}{6}f''(t,y)$$

with $af(t,y) + bf(t+\alpha, y+\beta)$ (has 4 parameters a, b, α, β).

Unfortunately we can't make match to the $\frac{hf''}{6}$ term of $T^{(3)}$, which contains $\frac{h^2}{6}f \cdot (\partial_y f)^2$, by this way. But it leaves us open choices if we're OK with $O(h^2)$ error: let $a = b = 1$, $\alpha = h$, $\beta = hf(t,y)$, then we get the **modified Euler's method**:

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + \frac{h}{2}\left(f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))\right), \ i = 0, 1, \ldots, N-1. \end{cases}$$

Also need evaluation of f twice in each step.

Runge-Kutta (RK) method

Example

Use Midpoint method (RK2) and Modified Euler's method with $h = 0.2$ *to solve IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *and* $y(0) = 0.5$.

Solution: Apply the main steps in the two methods:

$$\textbf{Midpoint}: \quad w_{i+1} = w_i + h\, f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right)$$

$$\textbf{Modified Euler's}: \quad w_{i+1} = w_i + \frac{h}{2}\left(f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))\right)$$

Runge-Kutta (RK) method

Example

Use Midpoint method (RK2) and Modified Euler's method with $h = 0.2$ *to solve IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *and* $y(0) = 0.5$.

Solution: (cont)

t_i	$y(t_i)$	Midpoint Method	Error	Modified Euler Method	Error
0.0	0.5000000	0.5000000	0	0.5000000	0
0.2	0.8292986	0.8280000	0.0012986	0.8260000	0.0032986
0.4	1.2140877	1.2113600	0.0027277	1.2069200	0.0071677
0.6	1.6489406	1.6446592	0.0042814	1.6372424	0.0116982
0.8	2.1272295	2.1212842	0.0059453	2.1102357	0.0169938
1.0	2.6408591	2.6331668	0.0076923	2.6176876	0.0231715
1.2	3.1799415	3.1704634	0.0094781	3.1495789	0.0303627
1.4	3.7324000	3.7211654	0.0112346	3.6936862	0.0387138
1.6	4.2834838	4.2706218	0.0128620	4.2350972	0.0483866
1.8	4.8151763	4.8009586	0.0142177	4.7556185	0.0595577
2.0	5.3054720	5.2903695	0.0151025	5.2330546	0.0724173

Midpoint (RK2) method is better than modified Euler's method.

Runge-Kutta (RK) method

We can also consider higher-order RK method by fitting

$$T^{(3)}(t,y) = f(t,y) + \frac{h}{2}f'(t,y) + \frac{h^2}{6}f''(t,y)$$

with $af(t,y) + bf(t+\alpha_1, y+\delta_1(f(t+\alpha_2, y+\delta_2 f(t,y)))$ (has 6 parameters $a, b, \alpha_1, \alpha_2, \delta_1, \delta_2$) to reach $O(h^3)$ error.

For example, Heun's choice is $a = \frac{1}{4}$, $b = \frac{3}{4}$, $\alpha_1 = \frac{2h}{3}$, $\alpha_2 = \frac{h}{3}$, $\delta_1 = \frac{2h}{3}f$, $\delta_2 = \frac{h}{3}f$.

Nevertheless, methods of order $O(h^3)$ are rarely used in practice.

4-th Order Runge-Kutta (RK4) method

Most commonly used is the **4-th order Runge-Kutta method (RK4)**: start with $w_0 = \alpha$, and iteratively do

$$\begin{cases} k_1 = f(t_i, w_i) \\ k_2 = f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_1) \\ k_3 = f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_2) \\ k_4 = f(t_{i+1}, w_i + hk_3) \\ w_{i+1} = w_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

Need to evaluate f for 4 times in each step. Reach error $O(h^4)$.

4-th Order Runge-Kutta (RK4) method

Example

Use RK4 (with $h = 0.2$) to solve IVP $y' = y - t^2 + 1$ for $t \in [0, 2]$ and $y(0) = 0.5$.

Solution: With $h = 0.2$, we have $N = 10$ and $t_i = 0.2i$ for $i = 0, 1, \ldots, 10$. First set $w_0 = 0.5$, then the first iteration is

$$k_1 = f(t_0, w_0) = f(0, 0.5) = 0.5 - 0^2 + 1 = 1.5$$

$$k_2 = f(t_0 + \frac{h}{2}, w_0 + \frac{h}{2}k_1) = f(0.1, 0.5 + 0.1 \times 1.5) = 1.64$$

$$k_3 = f(t_0 + \frac{h}{2}, w_0 + \frac{h}{2}k_2) = f(0.1, 0.5 + 0.1 \times 1.64) = 1.654$$

$$k_4 = f(t_1, w_0 + hk_3) = f(0.2, 0.5 + 0.2 \times 1.654) = 1.7908$$

$$w_1 = w_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.8292933$$

So w_1 is our RK4 approximation of $y(t_1) = y(0.2)$.

4-th Order Runge-Kutta (RK4) method

Example

Use RK4 (with $h = 0.2$) to solve IVP $y' = y - t^2 + 1$ for $t \in [0, 2]$ and $y(0) = 0.5$.

Solution: (cont) Continue with $i = 1, 2, \cdots, 9$:

t_i	Exact $y_i = y(t_i)$	Runge-Kutta Order Four w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272027	0.0000269
1.0	2.6408591	2.6408227	0.0000364
1.2	3.1799415	3.1798942	0.0000474
1.4	3.7324000	3.7323401	0.0000599
1.6	4.2834838	4.2834095	0.0000743
1.8	4.8151763	4.8150857	0.0000906
2.0	5.3054720	5.3053630	0.0001089

High-order Runge-Kutta method

Can we use even higher-order method to improve accuracy?

$\#f$ eval	2	3	4	$5 \leq n \leq 7$	$8 \leq n \leq 9$	$n \geq 10$
Best error	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^{n-1})$	$O(h^{n-2})$	$O(h^{n-3})$

So

RK4 is the sweet spot.

Remark

Note that RK4 requires 4 evaluations of f each step. So it would make sense only if it's accuracy with step size $4h$ is higher than Midpoint with $2h$ or Euler's with h!

High-order Runge-Kutta method

Example

Use RK4 (with $h = 0.1$), Midpoint (with $h = 0.05$), and Euler's method (with $h = 0.025$) to solve IVP $y' = y - t^2 + 1$ for $t \in [0, 0.5]$ and $y(0) = 0.5$.

Solution:

t_i	Exact	Euler $h = 0.025$	Modified Euler $h = 0.05$	Runge-Kutta Order Four $h = 0.1$
0.0	0.5000000	0.5000000	0.5000000	0.5000000
0.1	0.6574145	0.6554982	0.6573085	0.6574144
0.2	0.8292986	0.8253385	0.8290778	0.8292983
0.3	1.0150706	1.0089334	1.0147254	1.0150701
0.4	1.2140877	1.2056345	1.2136079	1.2140869
0.5	1.4256394	1.4147264	1.4250141	1.4256384

RK4 is better with same computation cost!

Error control

Can we control the error of Runge-Kutta method by using variable step sizes?

Let's compare two difference methods with errors $O(h^n)$ and $O(h^{n+1})$ (say, RK4 and RK5) for fixed step size h, which have schemes below:

$$w_{i+1} = w_i + h\phi(t_i, w_i, h) \qquad O(h^n)$$
$$\tilde{w}_{i+1} = \tilde{w}_i + h\tilde{\phi}(t_i, \tilde{w}_i, h) \qquad O(h^{n+1})$$

Suppose $w_i \approx \tilde{w}_i \approx y(t_i) =: y_i$. Then for any given $\epsilon > 0$, we want to see how small h should be for the $O(h^n)$ method so that its error $|\tau_{i+1}(h)| \leq \epsilon$?

Error control

We recall that the local truncation errors of these two methods are:

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i, h) \approx O(h^n)$$
$$\tilde{\tau}_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - \tilde{\phi}(t_i, y_i, h) \approx O(h^{n+1})$$

Given that $w_i \approx \tilde{w}_i \approx y_i$ and $O(h^{n+1}) \ll O(h^n)$ for small h, we see

$$\begin{aligned}\tau_{i+1}(h) &\approx \tau_{i+1}(h) - \tilde{\tau}_{i+1}(h) = \tilde{\phi}(t_i, y_i, h) - \phi(t_i, y_i, h) \\ &\approx \tilde{\phi}(t_i, \tilde{w}_i, h) - \phi(t_i, w_i, h) = \frac{\tilde{w}_{i+1} - \tilde{w}_i}{h} - \frac{w_{i+1} - w_i}{h} \\ &\approx \frac{\tilde{w}_{i+1} - w_{i+1}}{h} \approx Kh^n\end{aligned}$$

for some $K > 0$ independent of h, since $\tau_{i+1}(h) \approx O(h^n)$.

Error control

Suppose that we can scale h by $q > 0$, such that

$$|\tau_{i+1}(qh)| \approx K(qh)^n = q^n K h^n \approx q^n \frac{|\tilde{w}_{i+1} - w_{i+1}|}{h} \leq \epsilon$$

So we need q to satisfy

$$q \leq \left(\frac{\epsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/n}$$

- $q < 1$: reject the initial h and recalculate using qh.
- $q \geq 1$: accept computed value and use qh for next step.

Runge-Kutta-Fehlberg method

The **Runge-Kutta-Fehlberg (RKF) method** uses specific 4th-order and 5th-order RK schemes, which share some computed values and together only need 6 evaluation of f, to estimate

$$q = \left(\frac{\epsilon h}{2|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/4} = 0.84\left(\frac{\epsilon h}{|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/4}$$

This q is used to tune step size so that error is always bounded by the prescribed ϵ.

Multistep method

Definition

*Let $m > 1$ be an integer, then an m-***step multistep method** *is given by the form of*

$$w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \cdots + a_0 w_{i-m+1} + h\left[b_m f(t_{i+1}, w_{i+1}) + b_{m-1} f(t_i, w_i) + \cdots + b_0 f(t_{i-m+1}, w_{i-m+1})\right]$$

for $i = m-1, m, \ldots, N-1$.

Here $a_0, \ldots, a_{m-1}$, $b_0, \ldots, b_m$ are constants. Also $w_0 = \alpha, w_1 = \alpha_1, \ldots, w_{m-1} = \alpha_{m-1}$ need to be given.

- $b_m = 0$*: Explicit m-step method.*
- $b_m \neq 0$*: Implicit m-step method.*

Multistep method

Definition

The **local truncation error** *of the m-step multistep method above is defined by*

$$\tau_{i+1}(h) = \frac{y_{i+1} - (a_{m-1}y_i + \cdots + a_0 y_{i-m+1})}{h} - \left[b_m f(t_{i+1}, y_{i+1}) + b_{m-1} f(t_i, y_i) + \cdots + b_0 f(t_{i-m+1}, y_{i-m+1})\right]$$

where $y_i := y(t_i)$.

Adams-Bashforth Explicit method

Adams-Bashforth Two-Step Explicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \\ w_{i+1} = w_i + \dfrac{h}{2}\left[3f(t_i, w_i) - f(t_{i-1}, w_{i-1})\right] \end{cases}$$

for $i = 1, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = \frac{5}{12} y'''(\mu_i) h^2$$

for some $\mu_i \in (t_{i-1}, t_{i+1})$.

Adams-Bashforth Explicit method: local truncation error

We denote $y_i^{(k)} := y^{(k)}(t_i)$ for short. If $w_j = y_j$ for $j \leq i$, then

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2}y_i'' + \frac{h^3}{6}y'''(\xi_i), \quad \xi_i \in (t_i, t_{i+1})$$

$$w_{i+1} = y_i + hy_i' + \frac{h}{2}(y_i' - y_{i-1}'),$$

$$y_{i-1}' = y_i' - hy_i'' + \frac{h^2}{2}y'''(\eta_i), \quad \eta_i \in (t_{i-1}, t_i)$$

Plugging the equations above into the formula of local truncation error:

$$\tau_{i+1}(h) = \frac{y_{i+1} - w_{i+1}}{h} = \left(\frac{1}{6}y'''(\xi_i) + \frac{1}{4}y'''(\eta_i)\right)h^2 = \frac{5}{12}y'''(\mu_i)h^2$$

for some $\mu_i \in (t_{i-1}, t_{i+1})$, where in the last equality we used the intermediate value theorem and $y \in C^3$ (so y''' is continuous) to obtain $y'''(\mu_i) = \frac{\frac{1}{6}y'''(\xi_i) + \frac{1}{4}y'''(\eta_i)}{\frac{1}{6}+\frac{1}{4}}$ which is between $y'''(\xi_i)$ and $y'''(\eta_i)$.

Adams-Bashforth Explicit method

Adams-Bashforth Three-Step Explicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \\ w_{i+1} = w_i + \dfrac{h}{12}\left[23f(t_i, w_i) - 16f(t_{i-1}, w_{i-1}) + 5f(t_{i-2}, w_{i-2})\right] \end{cases}$$

for $i = 2, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = \frac{3}{8}y^{(4)}(\mu_i)h^3$$

for some $\mu_i \in (t_{i-2}, t_{i+1})$.

Adams-Bashforth Explicit method

Adams-Bashforth Four-Step Explicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3 \\ w_{i+1} = w_i + \frac{h}{24}\left[55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3})\right] \end{cases}$$

for $i = 3, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = \frac{251}{720} y^{(5)}(\mu_i) h^4$$

for some $\mu_i \in (t_{i-3}, t_{i+1})$.

Adams-Bashforth Explicit method

Adams-Bashforth Five-Step Explicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3, \quad w_4 = \alpha_4 \\ w_{i+1} = w_i + \frac{h}{720}[1901f(t_i, w_i) - 2774f(t_{i-1}, w_{i-1}) + 2616f(t_{i-2}, w_{i-2}) \\ \qquad\qquad - 1274f(t_{i-3}, w_{i-3}) + 251f(t_{i-4}, w_{i-4})] \end{cases}$$

for $i = 4, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = \frac{95}{288} y^{(6)}(\mu_i) h^5$$

for some $\mu_i \in (t_{i-4}, t_{i+1})$.

Adams-Moulton Implicit method

Adams-Moulton Two-Step Implicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \\ w_{i+1} = w_i + \dfrac{h}{12}[5f(t_{i+1}, w_{i+1}) + 8f(t_i, w_i) - f(t_{i-1}, w_{i-1})] \end{cases}$$

for $i = 1, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = -\frac{1}{24} y^{(4)}(\mu_i) h^3$$

for some $\mu_i \in (t_{i-1}, t_{i+1})$.

Adams-Moulton Implicit method

Adams-Moulton Three-Step Implicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2 \\ w_{i+1} = w_i + \dfrac{h}{24}[9f(t_{i+1}, w_{i+1}) + 19f(t_i, w_i) - 5f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})] \end{cases}$$

for $i = 2, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = -\frac{19}{720} y^{(5)}(\mu_i) h^4$$

for some $\mu_i \in (t_{i-2}, t_{i+1})$.

Adams-Moulton Implicit method

Adams-Moulton Four-Step Implicit method:

$$\begin{cases} w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3 \\ w_{i+1} = w_i + \frac{h}{720}[251f(t_{i+1}, w_{i+1}) + 646f(t_i, w_i) - 264f(t_{i-1}, w_{i-1}) \\ \qquad\qquad + 106f(t_{i-2}, w_{i-2}) - 19f(t_{i-3}, w_{i-3})] \end{cases}$$

for $i = 3, \ldots, N-1$.

The local truncation error is

$$\tau_{i+1}(h) = -\frac{3}{160}y^{(6)}(\mu_i)h^5$$

for some $\mu_i \in (t_{i-3}, t_{i+1})$.

Steps to develop multistep methods

- Construct interpolating polynomial $P(t)$ (e.g., Newton's backward difference method) using previously computed $(t_{i-m+1}, w_{i-m+1}), \ldots, (t_i, w_i)$.
- Approximate $y(t_{i+1})$ based on

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} y'(t)\,\mathrm{d}t = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\,\mathrm{d}t$$
$$\approx y(t_i) + \int_{t_i}^{t_{i+1}} f(t, P(t))\,\mathrm{d}t$$

and construct difference method:

$$w_{i+1} = w_i + h\phi(t_i, \ldots, t_{i-m+1}, w_i, \ldots, w_{i-m+1})$$

Explicit vs. Implicit

- Implicit methods are generally more accurate than the explicit ones (e.g., Adams-Moulton three-step implicit method is even more accurate than Adams-Bashforth four-step explicit method).
- Implicit methods require solving for w_{i+1} from

$$w_{i+1} = \cdots + \frac{h}{xxx} f(t_{i+1}, w_{i+1}) + \cdots$$

  which can be difficult or even impossible.
- There could be multiple solutions of w_{i+1} when solving the equation above in implicit methods.

Predictor-Corrector method

Due to the aforementioned issues, implicit methods are often cast in "predictor-corrector" form in practice.

In each step i:

- **Prediction:** Compute w_{i+1} using an explicit method ϕ to get $w_{i+1,p}$ using

$$w_{i+1,p} = w_i + h\phi(t_i, w_i, \ldots, t_{i-m+1}, w_{i-m+1})$$

- **Correction:** Substitute w_{i+1} by $w_{i+1,p}$ in the implicit method $\tilde{\phi}$ and compute w_{i+1} using

$$w_{i+1} = w_i + h\tilde{\phi}(t_{i+1}, w_{i+1,p}, t_i, w_i, \ldots, t_{i-m+1}, w_{i-m+1})$$

Predictor-Corrector method

Example

Use the Adams-Bashforth 4-step explicit method and Adams-Moulton 3-step implicit method to form the **Adams 4th-order Predictor-Corrector** *method.*

With initial value $w_0 = \alpha$, suppose we first generate w_1, w_2, w_3 using RK4 method. Then for $i = 3, 4, \ldots, N-1$:

- Use Adams-Bashforth 4-step explicit method to get a predictor $w_{i+1,p}$:

$$w_{i+1,p} = w_i + \frac{h}{24}\left[55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3})\right]$$

- Use Adams-Moulton 3-step implicit method to get a corrector w_{i+1}:

$$w_{i+1} = w_i + \frac{h}{24}[9f(t_{i+1}, w_{i+1,p}) + 19f(t_i, w_i) - 5f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})]$$

Predictor-Corrector method

Example

Use Adams Predictor-Corrector Method with $h = 0.2$ *to solve IVP* $y' = y - t^2 + 1$ *for* $t \in [0, 2]$ *and* $y(0) = 0.5$.

t_i	$y_i = y(t_i)$	w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272056	0.0000239
1.0	2.6408591	2.6408286	0.0000305
1.2	3.1799415	3.1799026	0.0000389
1.4	3.7324000	3.7323505	0.0000495
1.6	4.2834838	4.2834208	0.0000630
1.8	4.8151763	4.8150964	0.0000799
2.0	5.3054720	5.3053707	0.0001013

Other Predictor-Corrector method

We can also use Milne's 3-step explicit method and Simpson's 2-step implicit method below:

$$w_{i+1,p} = w_{i-3} + \frac{4h}{3}\left[2f(t_i, w_i) - f(t_{i-1}, w_{i-1}) + 2f(t_{i-2}, w_{i-2})\right]$$

$$w_{i+1} = w_{i-1} + \frac{h}{3}[f(t_{i+1}, w_{i+1,p}) + 4f(t_i, w_i) + f(t_{i-1}, w_{i-1})]$$

This method is $O(h^4)$ and generally has better accuracy than Adams PC method. However it is more likely to be vulnerable to round-off error.

Predictor-Corrector method

- PC methods have comparable accuracy as RK4, but often require only 2 evaluations of f in each step.
- Need to store values of f for several previous steps.
- Sometimes are more restrictive on step size h, e.g., in the stiff differential equation case later.

Variable step-size multistep method

Now let's take a closer look at the errors of the multistep methods. Denote $y_i := y(t_i)$.

The Adams-Bashforth 4-step explicit method has error

$$\tau_{i+1}(h) = \frac{251}{720} y^{(5)}(\mu_i) h^4$$

The Adams-Moulton 3-step implicit method has error

$$\tilde{\tau}_{i+1}(h) = -\frac{19}{720} y^{(5)}(\tilde{\mu}_i) h^4$$

where $\mu_i \in (t_{i-3}, t_{i+1})$ and $\tilde{\mu}_i \in (t_{i-2}, t_{i+1})$.

Question: Can we find a way to scale step size h so the error is under control?

Variable step-size multistep method

Consider the their local truncation errors:

$$y_{i+1} - w_{i+1,p} = \frac{251}{720} y^{(5)}(\mu_i) h^5$$
$$y_{i+1} - w_{i+1} = -\frac{19}{720} y^{(5)}(\tilde{\mu}_i) h^5$$

Assume $y^{(5)}(\mu_i) \approx y^{(5)}(\tilde{\mu}_i)$, we take their difference to get

$$w_{i+1} - w_{i+1,p} = \frac{1}{720}(19 + 251) y^{(5)}(\mu_i) h^5 \approx \frac{3}{8} y^{(5)}(\mu_i) h^5$$

So the error of Adams-Moulton (corrector step) is

$$\tilde{\tau}_{i+1}(h) = \frac{|y_{i+1} - w_{i+1}|}{h} \approx \frac{19|w_{i+1} - w_{i+1,p}|}{270h} = Kh^4$$

where K is independent of h since $\tilde{\tau}_{i+1}(h) = O(h^4)$.

Variable step-size multistep method

If we want to keep error under a prescribed ϵ, then we need to find $q > 0$ such that with step size qh, there is

$$\tilde{\tau}_{i+1}(qh) = \frac{|y(t_i + qh) - w_{i+1}|}{qh} \approx \frac{19q^4|w_{i+1} - w_{i+1,p}|}{270h} < \epsilon$$

This implies that

$$q < \left(\frac{270h\epsilon}{19|w_{i+1} - w_{i+1,p}|}\right)^{1/4} \approx 2\left(\frac{h\epsilon}{|w_{i+1} - w_{i+1,p}|}\right)^{1/4}$$

To be conservative, we may replace 2 by 1.5 above.

In practice, we tune q (as less as possible) such that the estimated error is between $(\epsilon/10, \epsilon)$

System of differential equations

The IVP for a system of ODE has form

$$\begin{cases} \dfrac{\mathrm{d}u_1}{\mathrm{d}t} = f_1(t, u_1, u_2, \ldots, u_m) \\ \dfrac{\mathrm{d}u_2}{\mathrm{d}t} = f_2(t, u_1, u_2, \ldots, u_m) \\ \quad \vdots \\ \dfrac{\mathrm{d}u_m}{\mathrm{d}t} = f_m(t, u_1, u_2, \ldots, u_m) \end{cases} \quad \text{for } a \le t \le b$$

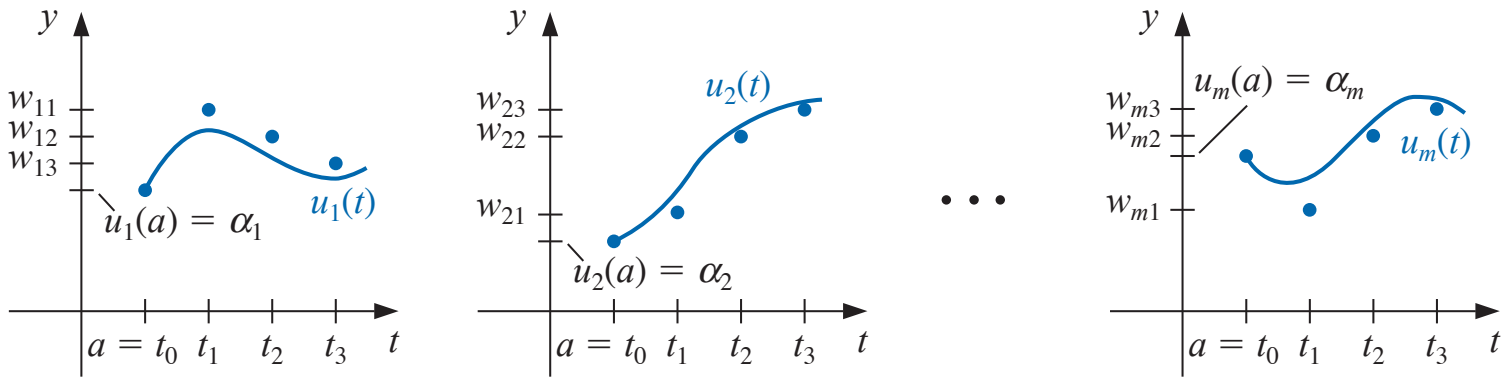
with initial value $u_1(a) = \alpha_1, \ldots, u_m(a) = \alpha_m$.

Definition

A set of functions $u_1(t), \ldots, u_m(t)$ is a **solution** *of the IVP above if they satisfy both the system of ODEs and the initial values.*

System of differential equations

In this case, we will solve for $u_1(t), \ldots, u_m(t)$ which are interdependent according to the ODE system.



System of differential equations

Definition

A function f is called **Lipschitz** *with respect to $u_1, \ldots, u_m$ on $D := [a, b] \times \mathbb{R}^m$ if there exists $L > 0$ s.t.*

$$|f(t, u_1, \ldots, u_m) - f(t, z_1, \ldots, z_m)| \leq L \sum_{j=1}^{m} |u_j - z_j|$$

for all $(t, u_1, \ldots, u_m), (t, z_1, \ldots, z_m) \in D$.

System of differential equations

Theorem

If $f \in C^1(D)$ and $|\frac{\partial f}{\partial u_j}| \leq L$ for all j, then f is Lipschitz with respect to $u = (u_1, \ldots, u_m)$ on D.

Proof.

Note that D is convex. For any $(t, u_1, \ldots, u_m), (t, z_1, \ldots, z_m) \in D$, define

$$g(\lambda) = f(t, (1-\lambda)u_1 + \lambda z_1, \ldots, (1-\lambda)u_m + \lambda z_m)$$

for all $\lambda \in [0, 1]$. Then from $|g(1) - g(0)| \leq \int_0^1 |g'(\lambda)| \, \mathrm{d}\lambda$ and the definition of g, the conclusion follows. □

System of differential equations

Theorem

If $f \in C^1(D)$ and is Lipschitz with respect to $u = (u_1, \ldots, u_m)$, then the IVP with f as defining function has a unique solution.

System of differential equations

Now let's use vector notations below

$$\begin{aligned}\mathbf{a} &= (\alpha_1, \ldots, \alpha_m)\\ \mathbf{y} &= (y_1, \ldots, y_m)\\ \mathbf{w} &= (w_1, \ldots, w_m)\\ \mathbf{f}(t, \mathbf{w}) &= (f_1(t, \mathbf{w}), \ldots, f_m(t, \mathbf{w}))\end{aligned}$$

Then the IVP of ODE system can be written as

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad t \in [a, b]$$

with initial value $\mathbf{y}(a) = \mathbf{a}$. So the difference methods developed above, such as RK4, still apply.

System of differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

$$\begin{cases} I_1'(t) = f_1(t, I_1, I_2) = -4I_1 + 3I_2 + 6 \\ I_2'(t) = f_2(t, I_1, I_2) = -2.4I_1 + 1.6I_2 + 3.6 \end{cases}$$

with initial value $I_1(0) = I_2(0) = 0$.

Solution: The exact solution is

$$\begin{cases} I_1(t) = -3.375e^{-2t} + 1.875e^{-0.4t} + 1.5 \\ I_2(t) = -2.25e^{-2t} + 2.25e^{-0.4t} \end{cases}$$

for all $t \geq 0$.

System of differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

$$\begin{cases} I_1'(t) = f_1(t, I_1, I_2) = -4I_1 + 3I_2 + 6 \\ I_2'(t) = f_2(t, I_1, I_2) = -2.4I_1 + 1.6I_2 + 3.6 \end{cases}$$

with initial value $I_1(0) = I_2(0) = 0$.

Solution: (cont) The result by RK4 is

t_j	$w_{1,j}$	$w_{2,j}$	$\lvert I_1(t_j) - w_{1,j}\rvert$	$\lvert I_2(t_j) - w_{2,j}\rvert$
0.0	0	0	0	0
0.1	0.5382550	0.3196263	0.8285×10^{-5}	0.5803×10^{-5}
0.2	0.9684983	0.5687817	0.1514×10^{-4}	0.9596×10^{-5}
0.3	1.310717	0.7607328	0.1907×10^{-4}	0.1216×10^{-4}
0.4	1.581263	0.9063208	0.2098×10^{-4}	0.1311×10^{-4}
0.5	1.793505	1.014402	0.2193×10^{-4}	0.1240×10^{-4}

High-order ordinary differential equations

A general **IVP for mth-order ODE** is

$$y^{(m)} = f(t, y, y', \ldots, y^{(m-1)}), \quad t \in [a, b]$$

with initial value $y(a) = \alpha_1, y'(a) = \alpha_2, \ldots, y^{(m-1)}(a) = \alpha_m$.

Definition

A function $y(t)$ is a **solution of IVP for the mth-order ODE** *above if $y(t)$ satisfies the differential equation for $t \in [a, b]$ and all initial value conditions at $t = a$.*

High-order ordinary differential equations

We can define a set of functions $u_1, \ldots, u_m$ s.t.

$$u_1(t) = y(t), \quad u_2(t) = y'(t), \quad \ldots, \quad u_m(t) = y^{(m-1)}(t)$$

Then we can convert the mth-order ODE to a system of first-order ODEs (total of m coupled ODEs):

$$\begin{cases} u_1' = u_2 \\ u_2' = u_3 \\ \quad \vdots \\ u_m' = f(t, u_1, u_2, \ldots, u_m) \end{cases} \qquad \text{for } a \le t \le b$$

with initial values $u_1(a) = \alpha_1, \ldots, u_m(a) = \alpha_m$.

High-order ordinary differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

$$y'' - 2y' + 2y = e^{2t} \sin t, \quad t \in [0, 1]$$

with initial value $y(0) = -0.4, y'(0) = -0.6$.

Solution: The exact solution is
$y(t) = u_1(t) = 0.2e^{2t}(\sin t - 2\cos t)$. Also $u_2(t) = y'(t) = u_1'(t)$ but we don't need it.

High-order ordinary differential equations

Example

Use RK4 (with $h = 0.1$) to solve IVP for ODE system

$$y'' - 2y' + 2y = e^{2t} \sin t, \quad t \in [0, 1]$$

with initial value $y(0) = -0.4, y'(0) = -0.6$.

Solution: (cont) The result by RK4 is

t_j	$y(t_j) = u_1(t_j)$	$w_{1,j}$	$y'(t_j) = u_2(t_j)$	$w_{2,j}$	$\lvert y(t_j) - w_{1,j}\rvert$	$\lvert y'(t_j) - w_{2,j}\rvert$
0.0	−0.40000000	−0.40000000	−0.6000000	−0.60000000	0	0
0.1	−0.46173297	−0.46173334	−0.6316304	−0.63163124	3.7×10^{-7}	7.75×10^{-7}
0.2	−0.52555905	−0.52555988	−0.6401478	−0.64014895	8.3×10^{-7}	1.01×10^{-6}
0.3	−0.58860005	−0.58860144	−0.6136630	−0.61366381	1.39×10^{-6}	8.34×10^{-7}
0.4	−0.64661028	−0.64661231	−0.5365821	−0.53658203	2.03×10^{-6}	1.79×10^{-7}
0.5	−0.69356395	−0.69356666	−0.3887395	−0.38873810	2.71×10^{-6}	5.96×10^{-7}
0.6	−0.72114849	−0.72115190	−0.1443834	−0.14438087	3.41×10^{-6}	7.75×10^{-7}
0.7	−0.71814890	−0.71815295	0.2289917	0.22899702	4.05×10^{-6}	2.03×10^{-6}
0.8	−0.66970677	−0.66971133	0.7719815	0.77199180	4.56×10^{-6}	5.30×10^{-6}
0.9	−0.55643814	−0.55644290	1.534764	1.5347815	4.76×10^{-6}	9.54×10^{-6}
1.0	−0.35339436	−0.35339886	2.578741	2.5787663	4.50×10^{-6}	1.34×10^{-5}

A brief summary

The difference methods we developed above, e.g., Euler's, midpoints, RK4, multistep explicit/implicit, predictor-corrector methods, are

- based on step-by-step derivation and easy to understand;
- widely used in many practical problems;
- fundamental to more advanced and complex techniques.

Stability of difference methods

Definition (Consistency)

A difference method is called **consistent** *if*

$$\lim_{h \to 0} \left(\max_{1 \le i \le N} \tau_i(h) \right) = 0$$

where $\tau_i(h)$ *is the local truncation error of the method.*

Remark

Since local truncation error $\tau_i(h)$ *is defined assuming previous* $w_i = y_i$*, it does not take error accumulation into account. So the consistency definition above only considers how good* $\phi(t, w_i, h)$ *in the difference method is.*

Stability of difference methods

For any step size $h > 0$, the difference method $w_{i+1} = w_i + h\phi(t_i, w_i, h)$ can generate a sequence of w_i which depend on h. We call them $\{w_i(h)\}_i$. Note that w_i gradually accumulate errors as $i = 1, 2, \ldots, N$.

Definition (Convergent)

A difference method is called **convergent** *if*

$$\lim_{h \to 0} \left(\max_{1 \le i \le N} |y_i - w_i(h)| \right) = 0$$

Stability of difference methods

Example

Show that Euler's method is convergent.

Solution: We have showed before that for fixed $h > 0$ there is

$$\left|y(t_i) - w_i\right| \leq \frac{hM}{2L}\left(e^{L(t_i-a)} - 1\right) \leq \frac{hM}{2L}\left(e^{L(b-a)} - 1\right)$$

for all $i = 0, \ldots, N$. Therefore we have

$$\max_{1\leq i\leq N}\left|y(t_i) - w_i\right| \leq \frac{hM}{2L}\left(e^{L(b-a)} - 1\right) \to 0$$

as $h \to 0$. Therefore $\lim_{h\to 0}(\max_{1\leq i\leq N}\left|y(t_i) - w_i\right|) = 0$.

Stability of difference method

Definition

A numerical method is called **stable** *if its results depend on the initial data continuously.*

Stability of difference methods

Theorem

For a given IVP $y' = f(t, y)$, $t \in [a, b]$ with $y(a) = \alpha$, consider a difference method $w_{i+1} = w_i + h\phi(t_i, w_i, h)$ with $w_0 = \alpha$. If there exists $h_0 > 0$ such that ϕ is continuous on $[a, b] \times \mathbb{R} \times [0, h_0]$, and ϕ is L-Lipschitz with respect to w, then

- *The difference method is stable.*
- *The difference method is convergent if and only if it is consistent (i.e., $\phi(t, y, 0) = f(t, y)$).*
- *If there exists bound $\tau(h)$ such that $|\tau_i(h)| \leq \tau(h)$ for all $i = 1, \ldots, N$, then $|y(t_i) - w_i| \leq \tau(h)e^{L(t_i - a)}/L$.*

Stability of difference methods

Proof.
Let h be fixed, then $w_i(\alpha)$ generated by the difference method are functions of α. For any two values $\alpha, \hat{\alpha}$, there is

$$\begin{aligned}
|w_{i+1}(\alpha) - w_{i+1}(\hat{\alpha})| &= |(w_i(\alpha) - h\phi(t_i, w_i(\alpha), h)) - (w_i(\hat{\alpha}) - h\phi(t_i, w_i(\hat{\alpha}), h))| \\
&\leq |w_i(\alpha) - w_i(\hat{\alpha})| + h|\phi(t_i, w_i(\alpha), h) - \phi(t_i, w_i(\hat{\alpha}), h)| \\
&\leq |w_i(\alpha) - w_i(\hat{\alpha})| + hL|w_i(\alpha) - w_i(\hat{\alpha})| \\
&= (1 + hL)|w_i(\alpha) - w_i(\hat{\alpha})| \\
&\leq \cdots \\
&\leq (1 + hL)^{i+1}|w_0(\alpha) - w_0(\hat{\alpha})| \\
&= (1 + hL)^{i+1}|\alpha - \hat{\alpha}| \\
&\leq (1 + hL)^{N}|\alpha - \hat{\alpha}|
\end{aligned}$$

Therefore $w_i(\alpha)$ is Lipschitz with respect to α (with constant at most $(1 + hL)^N$), and hence is continuous with respect to α. We omit the proofs for the other two assertions here. □

Stability of difference method

Example

Use the result of Theorem above to show that the Modified Euler's method is stable.

Solution: Recall the Modified Euler's method is given by

$$w_{i+1} = w_i + \frac{h}{2}\Big(f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))\Big)$$

So we have $\phi(t, w, h) = \frac{1}{2}(f(t, w) + f(t + h, w + hf(t, w)))$.
Now we want to show ϕ is continuous in (t, w, h), and Lipschitz with respect to w.

Stability of difference method

Solution: (cont) It is obvious that ϕ is continuous in (t, w, h) since $f(t, w)$ is continuous. Fix t and h. For any $w, \bar{w} \in \mathbb{R}$, there is

$$\begin{aligned}
|\phi(t, w, h) - \phi(t, \bar{w}, h)| &\leq \frac{1}{2}|f(t, w) - f(t, \bar{w})| \\
&\quad + \frac{1}{2}|f(t + h, w + hf(t, w)) - f(t + h, \bar{w} + hf(t, \bar{w}))| \\
&\leq \frac{L}{2}|w - \bar{w}| + \frac{L}{2}|(w + hf(t, w)) - (\bar{w} + hf(t, \bar{w}))| \\
&\leq L|w - \bar{w}| + \frac{Lh}{2}|f(t, w) - f(t, \bar{w})| \\
&\leq L|w - \bar{w}| + \frac{L^2 h}{2}|w - \bar{w}| \\
&= (L + \frac{L^2 h}{2})|w - \bar{w}|
\end{aligned}$$

So ϕ is Lipschitz with respect to w. By first part of Theorem above, the Modified Euler's method is stable.

Stability of multistep difference method

Definition

Suppose a multistep difference method given by

$$w_{i+1} = a_{m-1}w_i + a_{m-2}w_{i-1} + \cdots + a_0 w_{i-m+1} + hF(t_i, h, w_{i+1}, \ldots, w_{i-m+1})$$

Then we call the following the **characteristic polynomial** *of the method:*

$$\lambda^m - (a_{m-1}\lambda^{m-1} + \cdots + a_1\lambda + a_0)$$

Definition

A difference method is said to satisfy the **root condition** *if all the m roots $\lambda_1, \ldots, \lambda_m$ of its characteristic polynomial have magnitudes ≤ 1, and all of those which have magnitude $=1$ are single roots.*

Stability of multistep difference method

Definition

- *A difference method that satisfies root condition is called* **strongly stable** *if the only root with magnitude* 1 *is* $\lambda = 1$.
- *A difference method that satisfies root condition is called* **weakly stable** *if there are multiple roots with magnitude 1.*
- *A difference method that does not satisfy root condition is called* **unstable**.

Stability of multistep difference method

Theorem

- *A difference method is stable if and only if it satisfies the root condition.*
- *If a difference method is consistent, then it is stable if and only if it is convergent.*

Stability of multistep difference method

Example

Show that the Adams-Bashforth 4-step explicit method is strongly stable.

Solution: Recall that the method is given by

$$w_{i+1} = w_i + \frac{h}{24}\left[55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3})\right]$$

So the characteristic polynomial is simply $\lambda^4 - \lambda^3 = \lambda^3(\lambda - 1)$, which only has one root $\lambda = 1$ with magnitude 1. So the method is strongly stable.

Stability of multistep difference method

Example

Show that the Milne's 4-step explicit method is weakly stable but not strongly stable.

Solution: Recall that the method is given by

$$w_{i+1} = w_{i-3} + \frac{4h}{3}\left[2f(t_i, w_i) - f(t_{i-1}, w_{i-1}) + 2f(t_{i-2}, w_{i-2})\right]$$

So the characteristic polynomial is simply $\lambda^4 - 1$, which have roots $\lambda = \pm 1, \pm \mathrm{i}$. So the method is weakly stable but not strongly stable.

Remark

This is the reason we chose Adams-Bashforth-Moulton PC rather than Milne-Simpsons PC since the former is strongly stable and likely to be more robust.

Stiff differential equations

Stiff differential equations have e^{-ct} terms ($c > 0$ large) in their solutions. These terms $\to 0$ quickly, but their derivatives (of form $c^n e^{-ct}$) do not, especially at small t.

Recall that difference methods have errors proportional to the derivatives, and hence they may be inaccurate for stiff ODEs.

Stiff differential equations

Example

Use RK4 to solve the IVP for a system of two ODEs:

$$\begin{cases} u_1' = 9u_1 + 24u_2 + 5\cos t - \dfrac{1}{3}\sin t \\ u_2' = -24u_1 - 51u_2 - 9\cos t + \dfrac{1}{3}\sin t \end{cases}$$

with initial values $u_1(0) = 4/3$ *and* $u_2(0) = 2/3$.

Solution: The exact solution is

$$\begin{cases} u_1(t) = 2e^{-3t} - e^{-39t} + \dfrac{1}{3}\cos t \\ u_2(t) = -e^{-3t} + 2e^{-39t} - \dfrac{1}{3}\cos t \end{cases}$$

for all $t \geq 0$.

Stiff differential equations

Solution: (cont) When we apply RK4 to this stiff ODE, we obtain

t	$u_1(t)$	$w_1(t)$ $h = 0.05$	$w_1(t)$ $h = 0.1$	$u_2(t)$	$w_2(t)$ $h = 0.05$	$w_2(t)$ $h = 0.1$
0.1	1.793061	1.712219	−2.645169	−1.032001	−0.8703152	7.844527
0.2	1.423901	1.414070	−18.45158	−0.8746809	−0.8550148	38.87631
0.3	1.131575	1.130523	−87.47221	−0.7249984	−0.7228910	176.4828
0.4	0.9094086	0.9092763	−934.0722	−0.6082141	−0.6079475	789.3540
0.5	0.7387877	9.7387506	−1760.016	−0.5156575	−0.5155810	3520.00
0.6	0.6057094	0.6056833	−7848.550	−0.4404108	−0.4403558	15697.84
0.7	0.4998603	0.4998361	−34989.63	−0.3774038	−0.3773540	69979.87
0.8	0.4136714	0.4136490	−155979.4	−0.3229535	−0.3229078	311959.5
0.9	0.3416143	0.3415939	−695332.0	−0.2744088	−0.2743673	1390664.
1.0	0.2796748	0.2796568	−3099671.	−0.2298877	−0.2298511	6199352.

which can blow up for larger step size h.

Stiff differential equations

Now let's use a simple example to see why this happens: consider an IVP $y' = \lambda y$, $t \geq 0$, and $y(0) = \alpha$. Here $\lambda < 0$. We know the problem has solution $y(t) = \alpha e^{\lambda t}$.

Suppose we apply Euler's method, which is

$$\begin{aligned} w_{i+1} &= w_i + hf(t_i, w_i) = w_i + h\lambda w_i = (1 + \lambda h)w_i \\ &= \cdots = (1 + \lambda h)^{i+1} w_0 = (1 + \lambda h)^{i+1}\alpha \end{aligned}$$

Therefore we simply have $w_i = (1 + \lambda h)^i \alpha$. So the error is

$$|y(t_i) - w_i| = |\alpha e^{\lambda i h} - (1 + \lambda h)^i \alpha| = |e^{\lambda i h} - (1 + \lambda h)^i||\alpha|$$

In order for the error not to blow up, we need at least $|1 + \lambda h| < 1$, which yields $h < \frac{2}{|\lambda|}$. So h needs to be sufficiently small for large λ.

Stiff differential equations

Similar issue occurs for other one-step methods, which for this IVP can be written as $w_{i+1} = Q(\lambda h)w_i = \cdots = (Q(\lambda h))^{i+1}\alpha$. For the solution not to blow up, we need $|Q(\lambda h)| < 1$.

For example, in nth-order Taylor's method, we need

$$|Q(\lambda h)| = \left|1 + \lambda h + \frac{\lambda^2 h^2}{2} + \cdots + \frac{\lambda^n h^n}{n!}\right| < 1$$

which requires h to be very small.

The same issue occurs for multistep methods too.

Stiff differential equations

A remedy of stiff ODE is using implicit method, e.g., the implicit Trapezoid method:

$$w_{i+1} = w_i + \frac{h}{2}(f(t_{i+1}, w_{i+1}) + f(t_i, w_i))$$

In each step, we need to solve for w_{i+1} from the equation above. Namely, we need to solve for the root of $F(w)$:

$$F(w) := w - w_i - \frac{h}{2}(f(t_{i+1}, w) + f(t_i, w_i)) = 0$$

We can use fixed point iteration or Newton's method to solve $F(x) = 0$.

Section 2

Direct Methods for Linear Systems

Linear system of equations

In many real-world applications, we need to solve linear system of n equations with n variables $x_1, \ldots, x_n$:

$$\begin{aligned}
E_1: &\quad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
E_2: &\quad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \vdots \\
E_n: &\quad a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n
\end{aligned}$$

We're given $a_{ij},\ 1 \le i, j \le n$ and $b_i,\ (1 \le i \le n)$, and want to find $x_1, \ldots, x_n$ that satisfy the n equations $E_1, \ldots, E_n$.

Linear system of equations

General approach: Gauss elimination.

We use three operations to simplify the linear system:

- Equation E_i can be multiplied by λE_i for any $\lambda \neq 0$: $\lambda E_i \to E_i$
- E_j is multiplied by λ and added to E_i: $\lambda E_j + E_i \to E_i$
- Switch E_i and E_j: $E_i \leftrightarrow E_j$

The goal is to simply the linear system into a triangular form, and apply backward substitution to get $x_1, \ldots, x_n$.

Linear system of equations

Generally, we form the augmented matrix

$$\tilde{A} = [A\ \mathbf{b}],\ \text{where } A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

and apply Gaussian elimination to get a triangular form of $\tilde{A}$ then apply backward substitution. Total cost is $O(n^3)$.

Pivoting strategies

Standard Gauss elimination may not work properly in numerical computations.

Example

Apply Gauss elimination to the system

$$\begin{aligned} E_1: \quad & 0.003000x_1 + 59.14x_2 = 59.17 \\ E_2: \quad & 5.291x_1 - 6.130x_2 = 46.78 \end{aligned}$$

with four digits for arithmetic rounding. Compare the result to exact solution $x_1 = 10.00$ *and* $x_2 = 1.000$.

Pivoting strategies

Solution: We need to multiply E_1 by $\frac{5.291}{0.003000} = 1763.66\bar{6} \approx 1764$, then subtract it from E_2 and get:

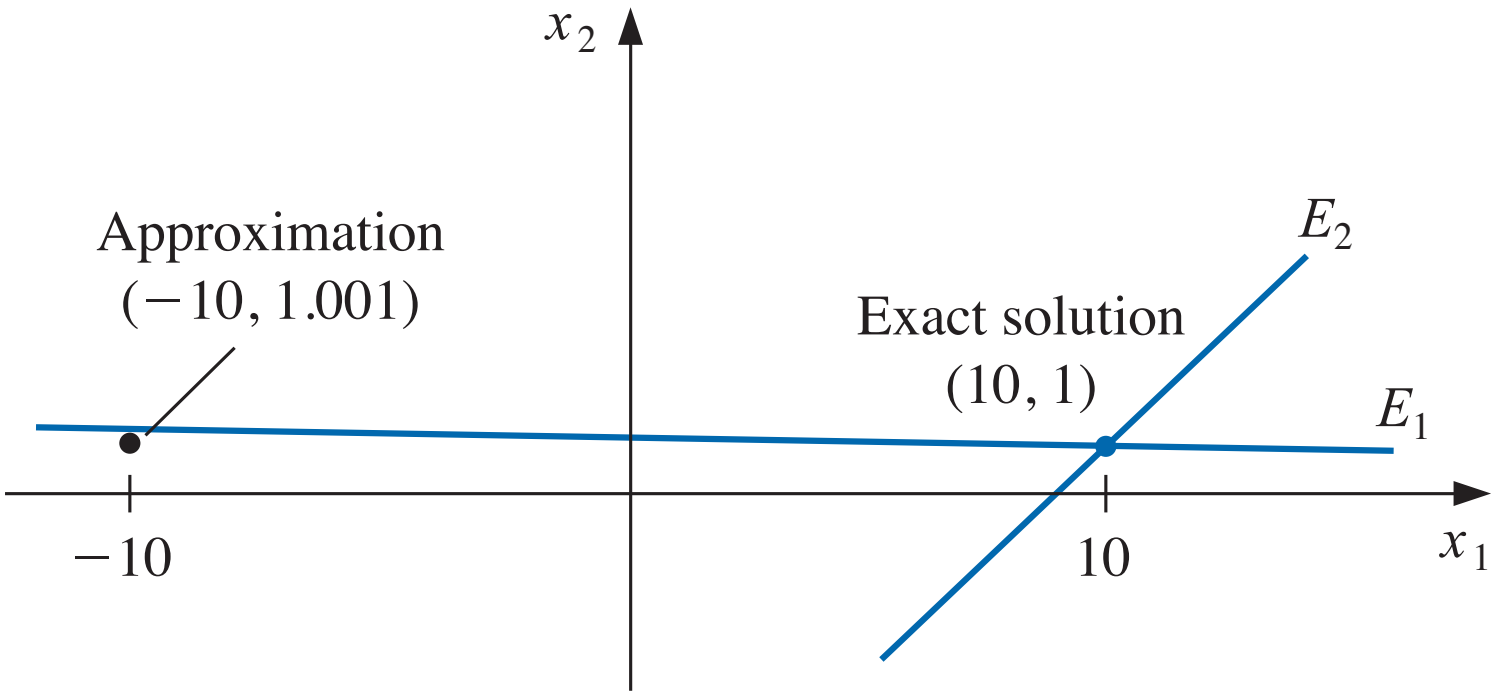
$$0.003000x_1 + 59.14x_2 \approx 59.17$$
$$-104300x_2 \approx -104400$$

On the other hand, the exact system without rounding error:

$$0.003000x_1 + 59.14x_2 \approx 59.17$$
$$-104309.37\bar{6}x_2 \approx -104309.37\bar{6}$$

Solving the former yields $x_2 = 1.001$ (still close to exact solution 1.000), but $x_1 = \frac{59.17-59.14x_2}{0.003000} = -10.00$ (far from exact solution 10.00).

Pivoting strategies



Partial pivoting

- The issue above comes up because the pivot a_{kk} is smaller than the remaining a_{ij} $(i, j > k)$.
- One remedy, called **partial pivoting**, is interchanging rows k and p (where $|a_{ik}| = \max\{|a_{ik}| : i = k, \ldots, n\}$).
- Sometimes interchange columns can also be performed.

For example, when we are about to do pivoting for the k-th time (i.e., $a_{kk}x_k$ term), we switch row p and current row k so that

$$p = \underset{k \le i \le n}{\operatorname{argmax}} |a_{ik}|$$

Redo the example above, we will get exact solution.

Scaled partial pivoting

Consider the following example:

$$\begin{aligned} E_1: &\quad 30.00x_1 + 591400x_2 = 591700 \\ E_2: &\quad 5.291x_1 - 6.130x_2 = 46.78 \end{aligned}$$

This is equivalent to example above, except that E_1 is multiplied by 10^4.

If we apply partial pivoting above, we will not exchange E_1 and E_2 since $30.00 > 5.291$, and will end up with the same *incorrect* answer $x_2 = 1.001$ and $x_1 = -10.00$.

To overcome this issue, we can scale the coefficients of each row i by $1/s_i$ where $s_i = \max_{1 \le j \le n} |a_{ij}|$. Then apply partial pivoting based on the scaled values.

Scaled partial pivoting

Applying scaled partial pivoting to the example above, we first have

$s_1 = \max\{30.00, 519400\} = 519400, s_2 = \max\{5.291, 6.130\} = 6.130$

Hence we get $\frac{a_{11}}{s_1} = \frac{30.00}{519400} \approx 0.5073 \times 10^{-4}$, and $\frac{a_{21}}{s_2} = \frac{5.291}{6.130} = 0.8631$, the others are ± 1. By comparing $\frac{a_{11}}{s_1}$ and $\frac{a_{21}}{s_2}$, we will exchange E_1 and E_2, and hence obtain

$$\begin{aligned} E_1: &\quad 5.291x_1 - 6.130x_2 = 46.78 \\ E_2: &\quad 30.00x_1 + 591400x_2 = 591700 \end{aligned}$$

and apply Gauss elimination to obtain correct answer $x_2 = 1.000$ and $x_1 = 10.00$.

Complete pivoting

For each of the n steps, find the largest magnitude among all coefficients a_{ij} for $k \leq i, j \leq n$. Then switch rows and/or columns so that the one with largest magnitude is in the pivot position.

This requires $O(n^3)$ comparisons. Only worth it if the accuracy improvement justifies the cost.

Linear algebra: quick review

We call A an $m \times n$ (m-by-n) **matrix** if A is an array of mn numbers with m rows and n columns

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

We may simply denote it by $A = [a_{ij}]$, when its size is clear in the context.

Linear algebra: quick review

- We call two matrices equal, i.e., $A = B$, if $a_{ij} = b_{ij}$ for all i, j.
- The sum of two matrices of same size is: $A \pm B = [a_{ij} \pm b_{ij}]$.
- Scalar multiplication of A by $\lambda \in \mathbb{R}$ is $\lambda A = [\lambda a_{ij}]$.
- We denote the matrix of all zeros by 0, and $-A = [-a_{ij}]$.

Linear algebra: quick review

The set of all $m \times n$ matrices forms a **vector space**:

- $A + B = B + A$
- $(A + B) + C = A + (B + C)$
- $A + 0 = 0 + A$
- $A + (-A) = 0$
- $\lambda(A + B) = \lambda A + \lambda B$
- $(\lambda + \mu)A = \lambda A + \mu A$
- $\lambda(\mu A) = (\lambda\mu)A$
- $1A = A$

Linear algebra: quick review

For matrix A of size $m \times n$ and (column) vector b of dimension n, we define the matrix-vector multiplication (product) by

$$Ab = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j} b_j \\ \sum_{j=1}^{n} a_{2j} b_j \\ \vdots \\ \sum_{j=1}^{n} a_{mj} b_j \end{bmatrix}$$

Linear algebra: quick review

For matrix A of size $m \times n$ and matrix B of size $n \times k$, we define the matrix-matrix multiplication (product) by

$$AB = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nk} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^n a_{1j} b_{j1} & \sum_{j=1}^n a_{1j} b_{j1} & \cdots & \sum_{j=1}^n a_{1j} b_{jk} \\ \sum_{j=1}^n a_{2j} b_{j1} & \sum_{j=1}^n a_{2j} b_{j1} & \cdots & \sum_{j=1}^n a_{2j} b_{jk} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^n a_{mj} b_{j1} & \sum_{j=1}^n a_{mj} b_{j1} & \cdots & \sum_{j=1}^n a_{mj} b_{jk} \end{bmatrix} \in \mathbb{R}^{m \times k}$$

That is, if $C = AB$, then $[c_{ij}] = [\sum_r a_{ir} b_{rj}]$ for all i, j.

Linear algebra: quick review

Some properties of matrix product

- $A(BC) = (AB)C$
- $A(B + D) = AB + AD$
- $\lambda(AB) = (\lambda A)B = A(\lambda B)$

Remark

Note that $AB \neq BA$ in general, even if both exists.

Linear algebra: quick review

Some special matrices

- Square matrix: A is of size $n \times n$
- Diagonal matrix: $a_{ij} = 0$ if $i \neq j$.
- Identity matrix of order n: $I = [\delta_{ij}]$ where $\delta_{ij} = 1$ if $i = j$ and $= 0$ otherwise.
- Upper triangle matrix: $a_{ij} = 0$ if $i > j$.
- Lower triangle matrix: $a_{ij} = 0$ if $i < j$.

Linear algebra: quick review

Definition (Inverse of matrix)

An $n \times n$ matrix A is said to be **nonsingular** *(or* **invertible***) if there exists an $n \times n$ matrix, denoted by A^{-1}, such that $A(A^{-1}) = (A^{-1})A = I$. Here A^{-1} is called the* **inverse** *of matrix A.*

Definition

An $n \times n$ matrix A without an inverse is called **singular** *(or* **noninvertible***)*

Linear algebra: quick review

Several properties of inverse matrix:

- A^{-1} is unique.
- $(A^{-1})^{-1} = A$.
- If B is also nonsingular, then $(AB)^{-1} = B^{-1}A^{-1}$.

Linear algebra: quick review

Definition (Transpose)

The **transpose** *of an* $m \times n$ *matrix* $A = [a_{ij}]$ *is the* $n \times m$ *matrix* $A^\top = [a_{ji}]$.

Sometimes $A^\top$ is also written as A^t, A', A^T.

- $(A^\top)^\top = A$
- $(AB)^\top = B^\top A^\top$
- $(A + B)^\top = A^\top + B^\top$
- If A is nonsingular, then $(A^{-1})^\top = (A^\top)^{-1}$.

Linear algebra: quick review

Definition (Determinant)

- *If $A = [a]$ is a 1×1 matrix, then $\det(A) = a$.*
- *If A is $n \times n$ where $n > 1$, then the* **minor** *M_{ij} is the determinant of the $(n-1) \times (n-1)$ submatrix of A by deleting its ith row and jth column.*
- *The* **cofactor** *A_{ij} associated with the minor M_{ij} is defined by $A_{ij} = (-1)^{i+j} M_{ij}$.*
- *The* **determinant** *of the $n \times n$ matrix A, denoted by $\det(A)$ (or $|A|$), is given by either of the followings:*

$$\det(A) = \sum_{j=1}^{n} a_{ij} A_{ij} = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} M_{ij}, \quad \text{for any } i = 1, \ldots, n$$

$$\det(A) = \sum_{i=1}^{n} a_{ij} A_{ij} = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} M_{ij}, \quad \text{for any } j = 1, \ldots, n$$

Linear algebra: quick review

Some properties of determinant

- If A has any zero row or column, then $\det(A) = 0$.
- If two rows (or columns) of A are the same, or one is a multiple of the other, then $\det(A) = 0$.
- Switching two rows (or columns) of A results in a matrix with determinant $-\det(A)$.
- Multiplying a row (or column) of A by λ results in a matrix with determinant $\lambda \det(A)$.
- $(E_i + \lambda E_j) \to E_i$ results in a matrix of the same determinant.

Linear algebra: quick review

Some properties of determinant

- $\det(AB) = \det(A)\det(B)$ if A and B are square matrices of same size.
- $\det(A^\top) = \det(A)$
- A is singular if any only if $\det(A) = 0$.
- If A is nonsingular, then $\det(A) \neq 0$ and $\det(A^{-1}) = \det(A)^{-1}$.
- If A is an upper or lower triangular matrix, then $\det(A) = \prod_{i=1}^{n} a_{ii}$.

Linear algebra: quick review

The following statements are equivalent:

- $Ax = 0$ has unique solution $x = 0$.
- $Ax = b$ has a unique solution for every b.
- A is nonsingular, i.e., A^{-1} exists.
- $\det(A) \neq 0$.

Matrix factorization

Gauss elimination can be used to compute **LU factorization** of a square matrix A:

$$A = LU$$

where L is a lower triangular matrix, and U is an upper triangular matrix.

Matrix factorization

If we have **LU factorization** of A, then

$$Ax = LUx = L(Ux) = b$$

so we solve x easily:

1. Solve y from $Ly = b$ by forward substitution;
2. Solve x from $Ux = y$ by backward substitution.

Total cost is $O(2n^2)$.

Matrix factorization

The cost reduction from $O(n^3/3)$ to $O(2n^2)$ is huge, especially for large n:

n	$n^3/3$	$2n^2$	% Reduction
10	$3.\bar{3} \times 10^2$	2×10^2	40
100	$3.\bar{3} \times 10^5$	2×10^4	94
1000	$3.\bar{3} \times 10^8$	2×10^6	99.4

Unfortunately, LU factorization itself requires $O(n^3)$ in general.

LU factorization

Now let's see how to obtain LU factorization by Gauss elimination.

Suppose we can perform Gauss elimination without any row exchange. In first round, we use a_{11} as the pivot and cancel each of $a_{21}, \ldots, a_{n1}$ by

$$(E_j - m_{j1}E_1) \to E_j \quad \text{where } m_{j1} = \frac{a_{j1}}{a_{11}}, \quad j = 2, \ldots, 4$$

This is equivalent to multiplying $M^{(1)}$ to A and get $A^{(2)} := M^{(1)}A$ where

$$M^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & \cdots & 1 \end{bmatrix} \text{ and } A^{(2)} = \begin{bmatrix} a_{11} & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix}$$

LU factorization

In second round, we use current a_{22} as the pivot and cancel each of $a_{32}, \ldots, a_{n2}$ by

$$(E_j - m_{j2}E_2) \to E_j \quad \text{where } m_{j2} = \frac{a_{j2}}{a_{22}}, \quad j = 3, \ldots, 4$$

This is equivalent to multiplying $M^{(2)}$ to $A^{(2)}$ and get $A^{(3)} := M^{(2)}A^{(2)}$ where

$$M^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -m_{n2} & 0 & \cdots & 1 \end{bmatrix} \text{ and } A^{(3)} = \begin{bmatrix} a_{11} & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & \cdots & * \end{bmatrix}$$

LU factorization

When Gauss elimination finishes (total $n-1$ rounds), we will get an upper triangular matrix U:

$$U := M^{(n-1)} M^{(n-2)} \dots M^{(1)} A$$

Define matrix L

$$L = (M^{(n-1)} M^{(n-2)} \dots M^{(1)})^{-1} = (M^{(1)})^{-1} \dots (M^{(n-2)})^{-1} (M^{(n-1)})^{-1}$$

Note that L is lower triangular (because each M is lower triangular, and inverse and product of lower triangular matrices are still lower triangular). So we get the LU factorization of A:

$$LU = (M^{(1)})^{-1} \dots (M^{(n-2)})^{-1} (M^{(n-1)})^{-1} M^{(n-1)} M^{(n-2)} \dots M^{(1)} A = A$$

LU factorization

It is easy to check that:

$$M^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & \cdots & 1 \end{bmatrix} \text{ and } (M^{(1)})^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & 0 & \cdots & 1 \end{bmatrix}$$

$$M^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -m_{n2} & 0 & \cdots & 1 \end{bmatrix} \text{ and } (M^{(2)})^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & m_{n2} & 0 & \cdots & 1 \end{bmatrix}$$

LU factorization

and finally there is

$$L = (M^{(1)})^{-1} \cdots (M^{(n-2)})^{-1}(M^{(n-1)})^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{bmatrix}$$

To summarize, the LU factorization of A gives L as above, and U as the result of Gauss elimination of A.

Gauss elimination row exchange

If Gauss elimination is done with row exchanges, then we will get LU factorization of PA where P is some row permutation matrix.

For example, to switch rows 2 and 4 of a 4×4 matrix A, the permutation matrix P is

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Some properties of permutation matrices:

- If P_1, P_2 are permutations, then $P_2 P_1$ is still permutation.
- $P^{-1} = P^\top$.

Diagonally dominate matrices

Now we consider two types of matrices for which Gauss elimination can be used effectively without row interchanges.

Definition (Diagonally dominate matrices)

An $n \times n$ matrix A is called **diagonally dominate** *if*

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \quad \forall i = 1, 2, \ldots, n$$

An $n \times n$ matrix A is called **strictly diagonally dominate** *if*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \forall i = 1, 2, \ldots, n$$

Diagonally dominate matrices

Example

Consider the following matrices:

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 7 & 2 & 0 \\ 3 & 5 & -1 \\ 0 & 5 & -6 \end{bmatrix} \quad C = \begin{bmatrix} 6 & 4 & -3 \\ 4 & -2 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

A (and $A^\top$) is diagonally dominate, B is strictly diagonally dominate, $B^\top, C, C^\top$ are not diagonally dominate.

Diagonally dominate matrices

Theorem

If A is strictly diagonally dominant, then A is nonsingular. Moreover, Gauss elimination can be performed without row interchange to obtain the unique solution of $Ax = b$.

Diagonally dominate matrices

Proof.

If A is singular, then $Ax = 0$ has nonzero solution x. Suppose x_k is the component of x with largest magnitude:

$$|x_k| > 0 \quad \text{and} \quad |x_k| \geq |x_j|, \ \forall j \neq k$$

Then the product of x and the k-th row of A gives

$$a_{kk}x_k + \sum_{j \neq k} a_{kj}x_j = 0$$

From this we obtain

$$|a_{kk}| = \left| -\sum_{j \neq k} \frac{a_{kj}x_j}{x_k} \right| \leq \sum_{j \neq k} \frac{|x_j|}{|x_k|} |a_{kj}| \leq \sum_{j \neq k} |a_{kj}|$$

Contradiction. So A is nonsingular. □

Diagonally dominate matrices

Proof (cont.) Now let's see how Gauss elimination works when A is strictly diagonally dominant. Consider 1st and ith ($i \geq 2$) rows of A:

$$|a_{11}| > \sum_{j \neq 1} |a_{1j}|, \quad |a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

If we perform $E_i - \frac{a_{i1}}{a_{11}} E_1 \to E_i$, the new values in row i are $a_{i1}^{(2)} = 0$ and $a_{ij}^{(2)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}$ for $j \geq 2$. Therefore

$$\sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}^{(2)}| \leq \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}| + \sum_{\substack{j=2 \\ j \neq i}}^{n} \left| \frac{a_{1j}}{a_{11}} \right| |a_{i1}| < |a_{ii}| - |a_{i1}| + \frac{|a_{11}| - |a_{1i}|}{|a_{11}|} |a_{i1}|$$

$$= |a_{ii}| - \frac{|a_{1i}|}{|a_{11}|} |a_{i1}| \leq \left| |a_{ii}| - \frac{|a_{1i}|}{|a_{11}|} |a_{i1}| \right| = |a_{ii}^{(2)}|$$

As i is arbitrary, we know A remains strictly diagonally dominant after first round. By induction we know A stays as strictly diagonally dominant and Gauss elimination can be performed without row interexchange.

Positive definite matrices

Definition (Positive definite matrix)

A matrix A is called **positive definite** *(PD) if it is symmetric and $x^\top Ax > 0$ for any $x \neq 0$*

Remark

In some texts, A is called positive definite as long as $x^\top Ax > 0$ for any $x \neq 0$, so A is not necessarily symmetric. In these texts, the matrix in our definition above is called **symmetric positive definite** *(SPD).*

Positive definite matrices

We first have the following formula: if $x = (x_1, \ldots, x_n)^\top$ and $A = [a_{ij}]$, then

$$x^\top A x = \sum_{i,j} a_{ij} x_i x_j$$

Positive definite matrices

Example

Show that the matrix A below is PD:

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Solution: First A is symmetric. For any $x \in \mathbb{R}^3$, we have

$$\begin{aligned} x^\top Ax &= 2x_1^2 - 2x_1x_2 + 2x_2^2 - 2x_2x_3 + 2x_3^2 \\ &= x_1^2 + (x_1^2 - 2x_1x_2 + x_2^2) + (x_2^2 - 2x_2x_3 + x_3^2) + x_3^2 \\ &= x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_3^2 \end{aligned}$$

Therefore $x^\top Ax = 0$ if and only if $x_1 = x_2 = x_3 = 0$. So A is PD.

Positive definite matrices

Theorem

If A is an $n \times n$ positive definite matrix, then

- *A is nonsingular;*
- *$a_{ii} > 0$ for all i;*
- $\max_{i \neq j} |a_{ij}| \leq \max_i |a_{ii}|$;
- $(a_{ij})^2 < a_{ii} a_{jj}$ *for any* $i \neq j$.

Positive definite matrices

Proof.

- If $Ax = 0$, then $x^\top Ax = 0$ and hence $x = 0$ since A is PD. So A is nonsingular.
- Set $x = e_i$, where $e_i \in \mathbb{R}^n$ has 1 as the i-th component and zeros elsewhere. Then $x^\top Ax = e_i^\top Ae_i = a_{ii} > 0$.
- For any k, j, define $x, z \in \mathbb{R}^n$ such that $x_j = z_k = z_j = 1$ and $x_k = -1$, and $x_i = z_i = 0$ if $i \neq k, j$. Then we can show

$$0 < x^\top Ax = a_{jj} + a_{kk} - a_{kj} - a_{jk}$$
$$0 < z^\top Az = a_{jj} + a_{kk} + a_{kj} + a_{jk}$$

  Note that $a_{kj} = a_{jk}$, so we get $|a_{kj}| < \frac{a_{jj}+a_{kk}}{2} \leq \max_i a_{ii}$.
- For any $i \neq j$, set $x \in \mathbb{R}^n$ such that $x_i = \alpha$ and $x_j = 1$, and 0 elsewhere. Therefore $0 < x^\top Ax = a_{ii}\alpha^2 + 2a_{ij}\alpha + a_{jj}^2$ for any α. This implies that $4a_{ij}^2 - 4a_{ii}a_{jj} < 0$.

Positive definite matrices

Definition (Leading principal submatrix)

A **leading principal submatrix** *of A is the $k \times k$ upper left submatrix*

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix}$$

Positive definite matrices

Theorem

A symmetric matrix A is PD if and only if every leading principal submatrix has a positive determinant.

Example

Use the Theorem above to check A is PD:

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Positive definite matrices

Theorem

A matrix A is PD if and only either of the followings is true:

- *There exist a lower triangular matrix L with all 1 on its diagonal and a diagonal matrix D with all diagonal entries positive, such that* $A = LDL^\top$.
- *There exists a lower triangular matrix L with all diagonal entries positive such that* $A = LL^\top$ *(Cholesky factorization).*
- *Gauss elimination of A without row interchanges can be performed and all pivot elements are positive.*

Band matrices

Definition (Band matrix)

An $n \times n$ matrix A is called **band matrix** *if there exist p, q such that a_{ij} can be nonzero only if $i - q \leq j \leq i + p$. The band width is defined by $w = p + q + 1$.*

Definition (Tridiagonal matrix)

A band matrix with $p = q = 1$ is called **tridiagonal** *matrix.*

Crout factorization

The **Crout factorization** of a tridiagonal matrix is $A = LU$ where L is lower triangle, U is upper triangle, and both L, U are tridiagonal:

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 & 0 \\ l_{21} & l_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & l_{n-1,n-1} & 0 \\ 0 & 0 & \cdots & l_{n,n-1} & l_{nn} \end{bmatrix} \quad U = \begin{bmatrix} 1 & u_{12} & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u_{n-1,n} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Note that a tridiagonal matrix A has $3n-2$ unknowns, and the L and U together also have $3n-2$ unknowns.

Crout factorization

Theorem

A tridiagonal matrix A has a Crout factorization if either of the following statements is true:

- *A is positive definite;*
- *A is strictly diagonally dominant;*
- *A is diagonally dominant, $|a_{11}| > |a_{12}|$, $|a_{nn}| > |a_{n,n-1}|$, and $a_{i,i-1}, a_{i,i+1} \neq 0$ for all $i = 2, \ldots, n-1$.*

Crout factorization

With the special form of A, L and U, we can obtain the Crout factorization $A = LU$ by solving l_{ij} $(i = 1, \ldots, n$ and $j = i - 1, i)$ and $u_{i,i+1}$ $(i = 1, \ldots, n - 1)$ from

$$\begin{aligned}
a_{11} &= l_{11} \\
a_{i,i-1} &= l_{i,i-1}, && \text{for } i = 2, \ldots, n \\
a_{i,i} &= l_{i,i-1} u_{i-1,i} + l_{ii}, && \text{for } i = 2, \ldots, n \\
a_{i,i+1} &= l_{ii} u_{i,i+1}, && \text{for } i = 1, \ldots, n - 1
\end{aligned}$$

When we use Crout factorization to solve $Ax = b$, the cost is only $5n - 4$ multiplications/divisions and $3n - 3$ additions/subtractions.

Section 3

Iterative Methods in Matrix Algebra

Vector norm

Definition

A **vector norm** *on* $\mathbb{R}^n$*, denoted by* $\|\cdot\|$*, is a mapping from* $\mathbb{R}^n$ *to* $\mathbb{R}$ *such that*

- $\|x\| \geq 0$ *for all* $x \in \mathbb{R}^n$,
- $\|x\| = 0$ *if and only if* $x = 0$,
- $\|\alpha x\| = |\alpha|\|x\|$ *for all* $\alpha \in \mathbb{R}$ *and* $x \in \mathbb{R}^n$,
- $\|x + y\| \leq \|x\| + \|y\|$ *for all* $x, y \in \mathbb{R}$.

Vector norm

Definition (l_p norms)

The l_p (sometimes L_p or ℓ_p) norm of a vector is defined by

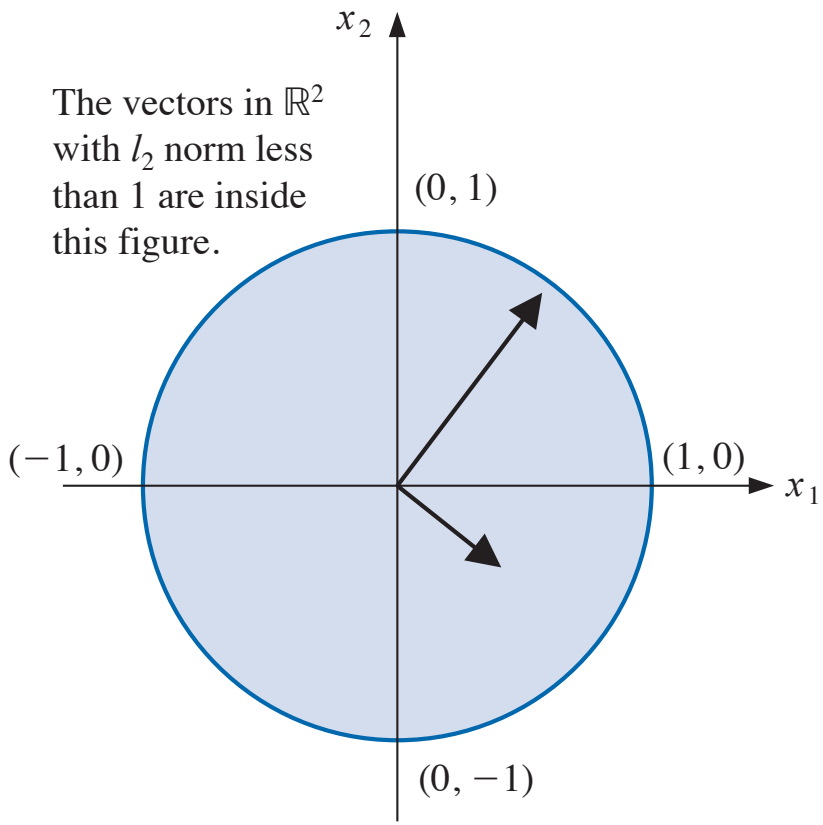
$$1 \le p < \infty: \qquad \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

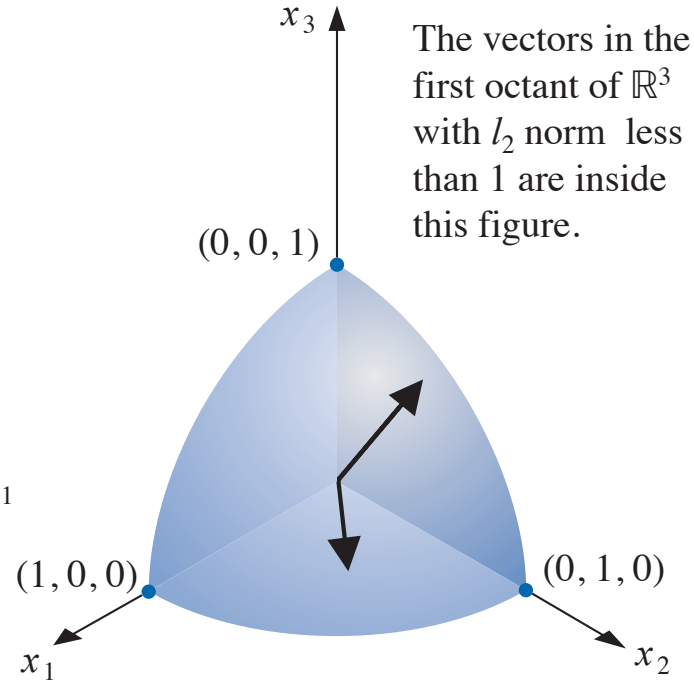
$$p = \infty: \qquad \|x\|_\infty = \max_{1 \le i \le n} |x_i|$$

In particular, the l_2 norm is also called the **Euclidean norm**.

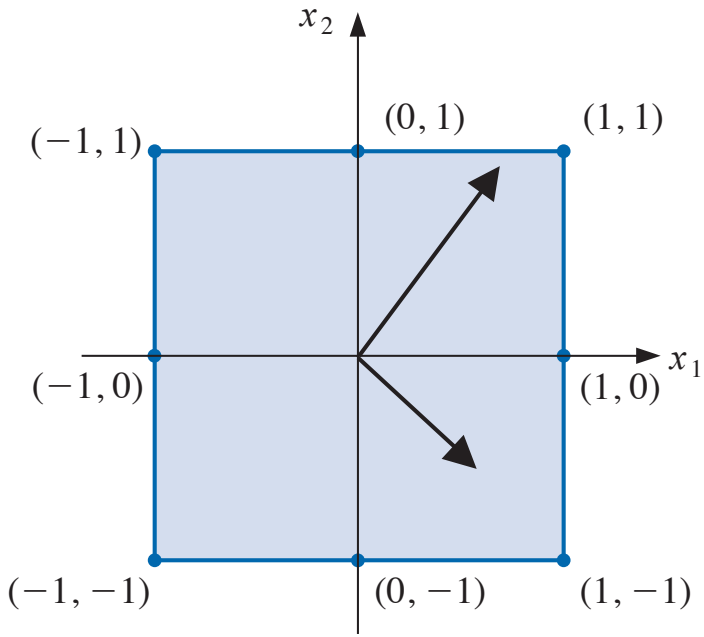
Note that when $0 \le p < 1$, $\|\cdot\|_p$ is not norm, strictly speaking, but have some usages in specific applications.

l_2 norm

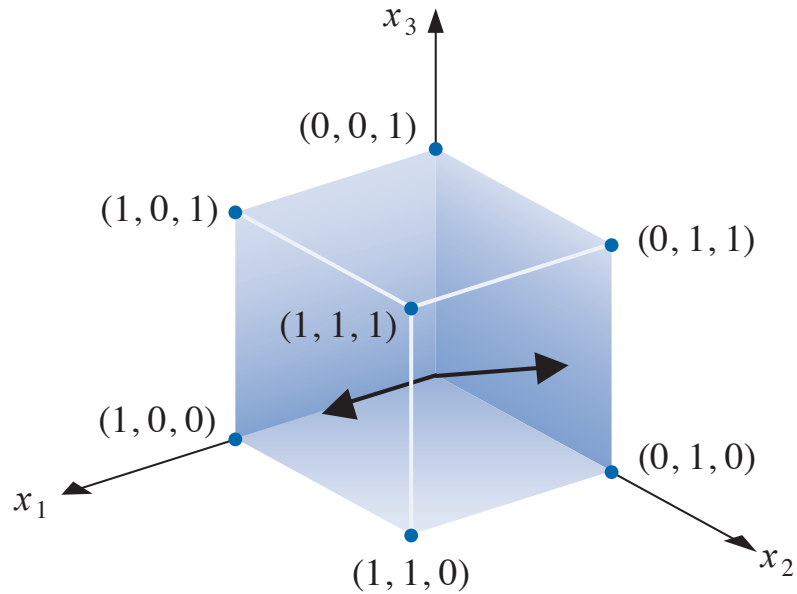

x_2
The vectors in $\mathbb{R}^2$ with l_2 norm less than 1 are inside this figure.
$(0, 1)$
$(-1, 0)$
$(1, 0)$
x_1
$(0, -1)$


x_3
The vectors in the first octant of $\mathbb{R}^3$ with l_2 norm less than 1 are inside this figure.
$(0, 0, 1)$
$(1, 0, 0)$
$(0, 1, 0)$
x_1
x_2

l_∞ norm



The vectors in $\mathbb{R}^2$ with l_∞ norm less than 1 are inside this figure.



The vectors in the first octant of $\mathbb{R}^3$ with l_∞ norm less than 1 are inside this figure.

Vector norms

Example

Compute the l_2 and l_∞ norms of vector $x = (1, -1, 2) \in \mathbb{R}^3$.

Solution:

$$\|x\|_2 = \sqrt{|1|^2 + |-1|^2 + |2|^2} = \sqrt{6}$$

$$\|x\|_\infty = \max_{1 \le i \le 3} |x_i| = \max\{|1|, |-1|, |2|\} = 2$$

Theorem (Cauchy-Schwarz inequality)

For any vectors $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ *and* $y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$, *there is*

$$|x^\top y| = \left|\sum_{i=1}^n x_i y_i\right| \le \left(\sum_{i=1}^n |x_i|^2\right)^{1/2} \left(\sum_{i=1}^n |y_i|^2\right)^{1/2} = \|x\|_2 \|y\|_2$$

Proof.

It is obviously true for $x = 0$ or $y = 0$. If $x, y \neq 0$, then for any $\lambda \in \mathbb{R}$, there is

$$0 \le \|x - \lambda y\|_2^2 = \|x\|_2^2 - 2\lambda x^\top y + \lambda^2 \|y\|_2^2$$

and the equality holds when $\lambda = \|x\|_2 / \|y\|_2$. □

Distance between vectors

Definition (Distance between two vectors)

The l_p **distance** *($1 \leq p \leq \infty$) between two vectors $x, y \in \mathbb{R}^n$ is defined by $\|x - y\|_p$.*

Definition (Convergence of a sequence of vectors)

A sequence $\{x^{(k)}\}$ is said to **converge with respect to the l_p norm** *if for any given $\epsilon > 0$, there exists an integer $N(\epsilon)$ such that*

$$\|x^{(k)} - x\| < \epsilon, \quad \text{for all } k \geq N(\epsilon)$$

Convergence of a sequence of vectors

Theorem

A sequence of vectors $\{x^{(k)}\}$ *converges to* x *if and only if* $x_i^{(k)} \to x_i$ *for every* $i = 1, 2, \ldots, n$.

Theorem

For any vector $x \in \mathbb{R}^n$, there is

$$\|x\|_\infty \le \|x\|_2 \le \sqrt{n}\|x\|_\infty$$

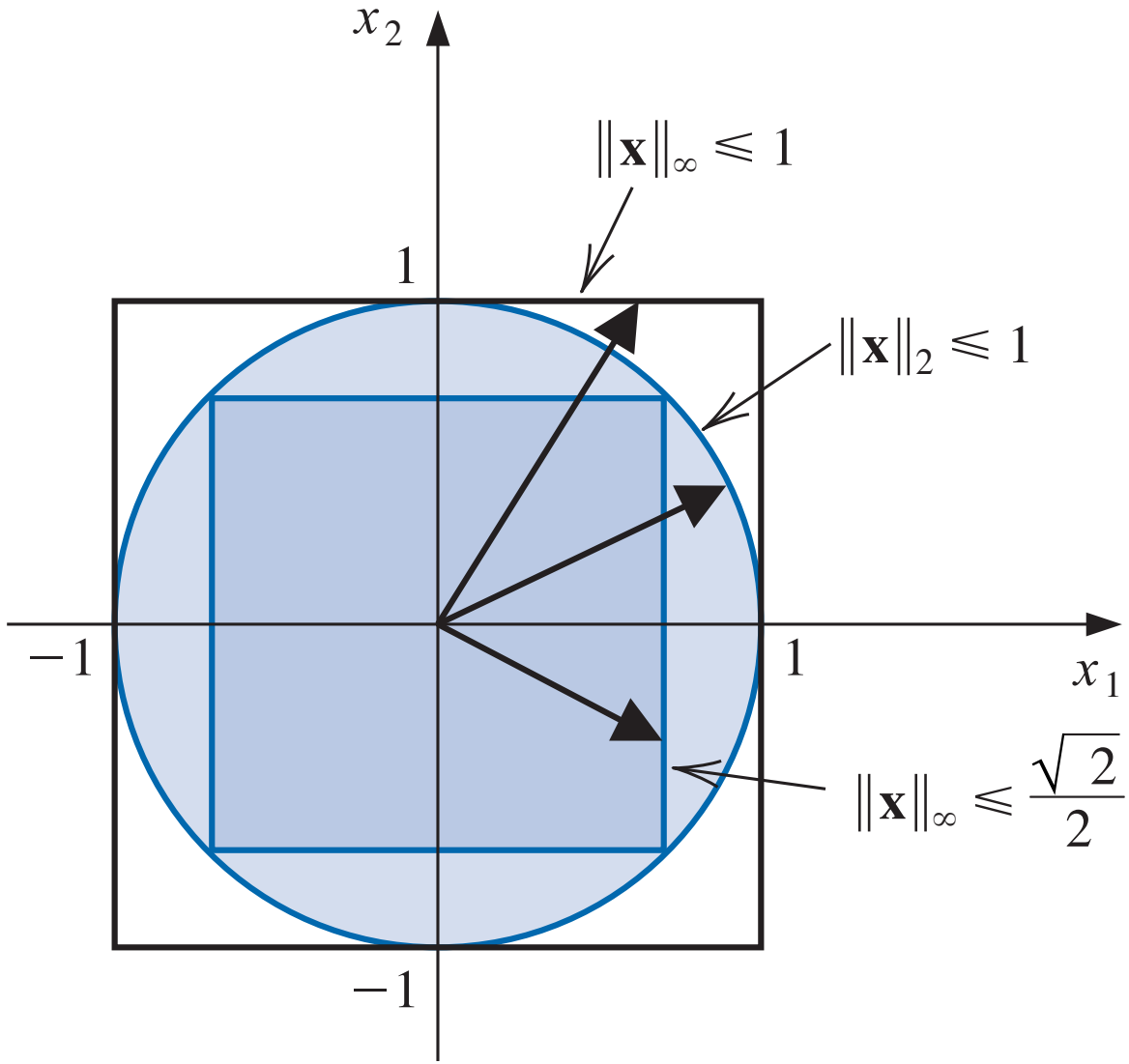
Proof.

$$\|x\|_\infty = \max_i |x_i| = \sqrt{\max_i |x_i|^2} \le \sqrt{|x_1|^2 + \cdots + |x_n|^2} = \|x\|_2$$

$$\begin{aligned}\|x\|_2 &= \sqrt{|x_1|^2 + \cdots + |x_n|^2} \le \sqrt{n \max_i |x_i|^2} \\ &= \sqrt{n}\sqrt{\max_i |x_i|^2} = \sqrt{n}\max_i |x_i| = \sqrt{n}\|x\|_\infty\end{aligned}$$

□

Compare l_2 and l_∞ norms in $\mathbb{R}^2$



Matrix norm

Definition

A **matrix norm** *on the set of $n \times n$ matrices is a real-valued function, denoted by $\|\cdot\|$, that satisfies the follows for all $A, B \in \mathbb{R}^{n\times n}$ and $\alpha \in \mathbb{R}$:*

- $\|A\| \geq 0$
- $\|A\| = 0$ *if and only if* $A = 0$ *the zero matrix,*
- $\|\alpha A\| = |\alpha|\|A\|$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|AB\| \leq \|A\|\|B\|$

Distance between matrices

Definition

Suppose $\|\cdot\|$ *is a norm defined on* $\mathbb{R}^{n\times n}$. *Then the* **distance between two** $n \times n$ **matrices** A **and** B *with respect to* $\|\cdot\|$ *is* $\|A - B\|$ *(check that it's a distance)*

Matrix norm can be induced by vector norms, and hence there are many choices. Here we focus on those induced by l_2 and l_∞ vector norms.

Matrix norm

Definition

If $\|\cdot\|$ is a vector norm on $\mathbb{R}^n$, then the norm defined below

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

is called the **matrix norm induced by vector norm** $\|\cdot\|$.

Matrix norm

Remark

- *Induced norms are also called natural norms of matrices.*
- *Unless otherwise specified, by matrix norms most books/papers refer to induced norms.*
- *The induced norm can be written equivalently as*

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

- *It can be easily extended to case* $A \in \mathbb{R}^{m \times n}$.

Matrix norm

Corollary

For any vector $x \in \mathbb{R}^n$, there is $\|Ax\| \leq \|A\|\|x\|$.

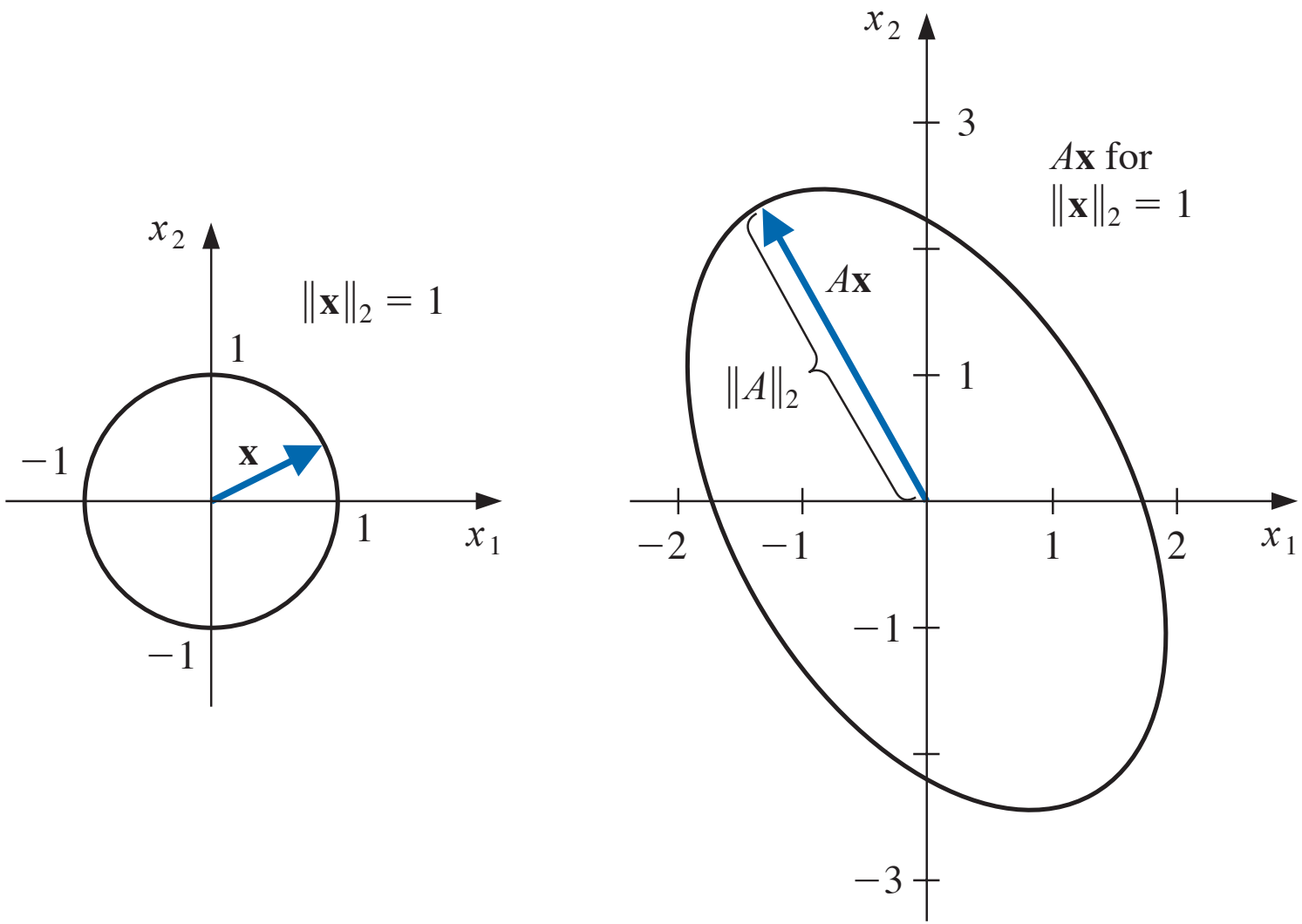
Proof.

It is obvious for $x = 0$. If $x \neq 0$, then

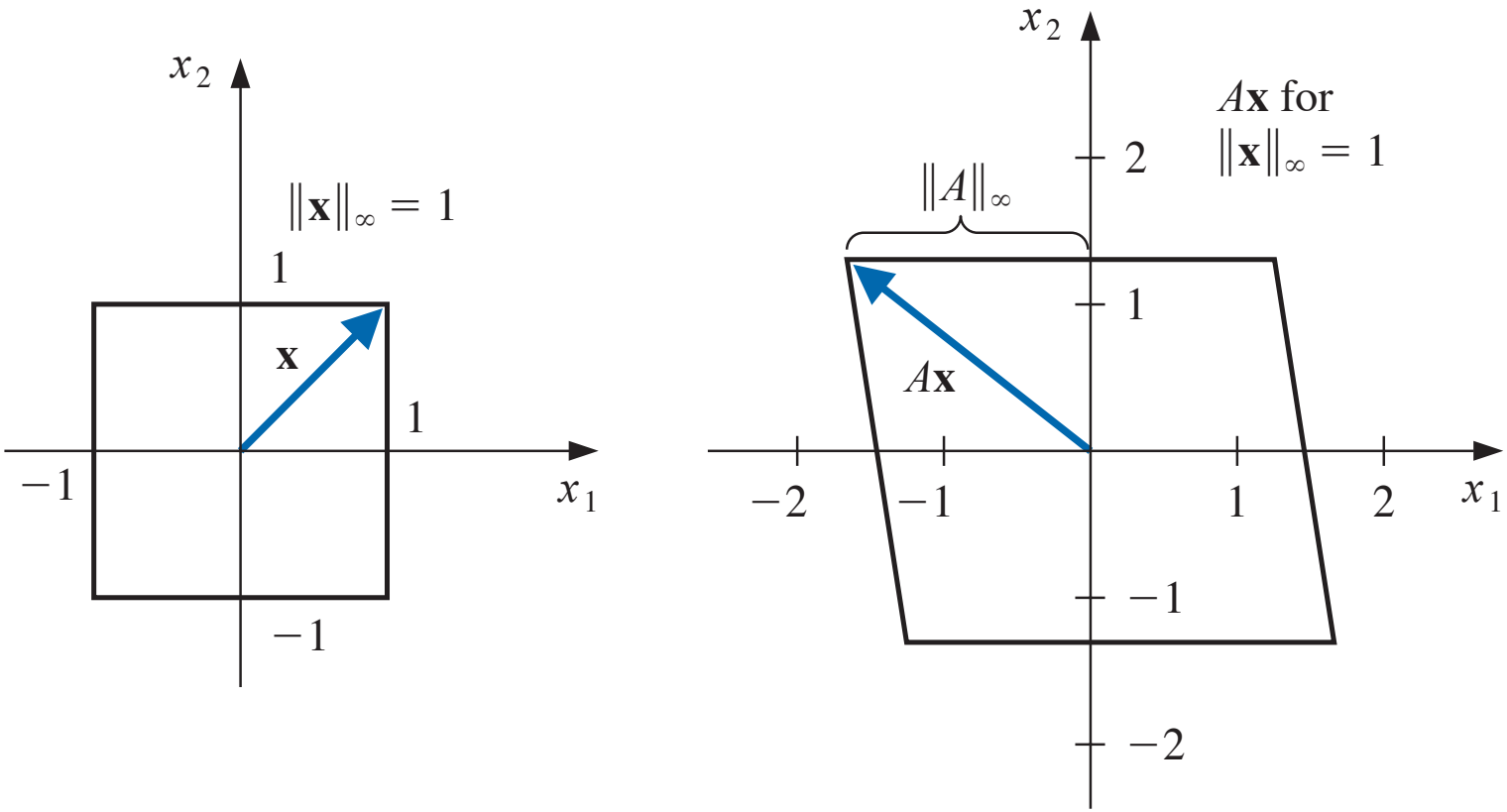
$$\frac{\|Ax\|}{\|x\|} \leq \max_{x' \neq 0} \frac{\|Ax'\|}{\|x'\|} = \|A\|$$

□

Induced l_2 matrix norm



Induced l_∞ matrix norm



Matrix norm

Theorem

Suppose $A = [a_{ij}] \in \mathbb{R}^{n\times n}$, *then* $\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$.

Matrix norm

Proof.

For any x with $\|x\|_\infty = 1$, i.e., $\max_i |x_i| = 1$, there is

$$\begin{aligned}
\|Ax\|_\infty &= \max\left\{\left|\sum_j a_{1j}x_j\right|, \ldots, \left|\sum_j a_{nj}x_j\right|\right\} \\
&\le \max\left\{\sum_j |a_{1j}||x_j|, \ldots, \sum_j |a_{nj}||x_j|\right\} \\
&\le \max\left\{\sum_j |a_{1j}|, \ldots, \sum_j |a_{nj}|\right\} = \max_{1\le i\le n}\sum_{j=1}^n |a_{ij}|
\end{aligned}$$

Suppose i' is such that $\sum_{j=1}^n |a_{i'j}| = \max_{1\le i\le n}\sum_{j=1}^n |a_{ij}|$, then by choosing $\hat{x}$ such that $\hat{x}_j = 1$ if $a_{i'j} \ge 0$ and -1 otherwise, we have $\sum_{j=1}^n a_{i'j}\hat{x}_j = \sum_{j=1}^n |a_{i'j}|$. So $\|A\hat{x}\|_\infty = \max_{1\le i\le n}\sum_{j=1}^n |a_{ij}|$. Note that $\|\hat{x}\|_\infty = 1$. Therefore $\|A\|_\infty = \max_{1\le i\le n}\sum_{j=1}^n |a_{ij}|$. □

Eigenvalues and eigenvectors of square matrices

Definition

The **characteristic polynomial** *of a square matrix* $A \in \mathbb{R}^{n\times n}$ *is defined by*

$$p(\lambda) = \det(A - \lambda I)$$

We call λ *an* **eigenvalue** *of* A *if* λ *is a root of* p*, i.e.,* $\det(A - \lambda I) = 0$*. Moreover, any nonzero solution* $x \in \mathbb{R}^n$ *of* $(A - \lambda I)x = 0$ *is called an* **eigenvector** *of* A *corresponding to the eigenvalue* λ*.*

Eigenvalues and eigenvectors of square matrices

Remark

- *$p(\lambda)$ is a polynomial of degree n, and hence has n roots.*
- *x is an eigenvector of A corresponding to eigenvalue λ iff $(A - \lambda I)x = 0$, i.e., $Ax = \lambda x$. This also means A applied to x is stretching x by λ.*
- *If x is an eigenvector of A corresponding to λ, so is αx for any $\alpha \neq 0$:*

$$A(\alpha x) = \alpha Ax = \alpha \lambda x = \lambda(\alpha x)$$

Eigenvalues and eigenvectors of square matrices

Definition

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A \in \mathbb{R}^{n \times n}$, then the **spectral radius** *$\rho(A)$ is defined by $\rho(A) = \max_i |\lambda_i|$ where $|\cdot|$ is the absolute value (aka magnitude) of complex numbers.*

Eigenvalues and eigenvectors of square matrices

Some properties

Theorem

For a matrix $A \in \mathbb{R}^{n\times n}$, there are

- $\|A\|_2 = \sqrt{\rho(A^\top A)}$
- $\rho(A) \le \|A\|$ *for any norm* $\|\cdot\|$ *of* A

Proof.

- We later will show that both sides $= \sigma_1^2$, where σ_1 is the largest singular value of A.
- Let $\lambda := \rho(A)$ be the eigenvalue with largest magnitude. Then there exists eigenvector x such that

$$(\|A\| \ge) \frac{\|Ax\|}{\|x\|} = \frac{\|\lambda x\|}{\|x\|} = \frac{|\lambda|\|x\|}{\|x\|} = |\lambda|$$

Convergent matrix

Definition

A matrix $A \in \mathbb{R}^{n\times n}$ is said to be **convergent** *if*

$$\lim_{k\to\infty} A^k = 0$$

Theorem

The following statements are equivalent:

1. *A is convergent.*
2. *$\lim_{k\to\infty} \|A^k\| = 0$ for any norm $\|\cdot\|$.*
3. *$\rho(A) < 1$.*
4. *$\lim_{k\to\infty} A^k x = 0$ for any $x \in \mathbb{R}$.*

Jacobi iterative method

To solve x from $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, the **Jacobi iterative method** is

- Initialize $x^{(0)} \in \mathbb{R}^n$. Set $D = \mathrm{diag}(A)$, $R = A - D$.
- Repeat the following for $k = 0, 1, \ldots$ until convergence:

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)})$$

Remark

- *Needs nonzero diagonal entries, i.e., $a_{ii} \neq 0$ for all i.*
- *Usually faster convergence with larger $|a_{ii}|$.*
- *Stopping criterion can be $\frac{\|x^{(k)}-x^{(k-1)}\|}{\|x^{(k)}\|} \leq \epsilon$ for some prescribed $\epsilon > 0$.*

Gauss-Seidel iterative method

To solve x from $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, the **Gauss-Seidel iterative method** is

- Initialize $x^{(0)} \in \mathbb{R}^n$. Set L to the lower triangular part (including diagonal) of A and $U = A - L$.
- Repeat the following for $k = 0, 1, \ldots$ until convergence:

$$x^{(k+1)} = L^{-1}(b - Ux^{(k)})$$

Remark

- *Inverse of L requires forward substitution.*
- *Again needs nonzero diagonal entries, i.e., $a_{ii} \neq 0$ for all i.*
- *Stopping criterion can be $\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|} \leq \epsilon$ for some prescribed $\epsilon > 0$.*
- *Faster than Jacobi iterative method most of times.*

General iterative methods

Lemma ($\rho(T) < 1 \Rightarrow I - T$ invertible)

If $\rho(T) < 1$, then $(I - T)^{-1}$ exists and

$$(I - T)^{-1} = I + T + T^2 + \cdots = \sum_{j=0}^{\infty} T^j$$

General iterative methods

Proof.

We first show that $I - T$ is invertible, i.e., $(I - T)x = 0$ has unique solution $x = 0$. If not, then $\exists x \neq 0$ such that $(I - T)x = 0$, i.e., $Tx = x$, or x is an e.v. corresponding to e.w. 1, contradiction to $\rho(T) < 1$.

Define $S_m = I + T + \cdots + T^m$. Then $(I - T)S_m = I - T^{m+1}$. Note $\rho(T) < 1$ implies $\lim_{m\to\infty} T^m = 0$, and hence

$$(I - T) \lim_{m\to\infty} S_m = \lim_{m\to\infty} (I - T)S_m = \lim_{m\to\infty} (I - T^{m+1}) = I$$

That is, $\sum_{m=0}^{\infty} T^m = \lim_{m\to\infty} S_m = (I - T)^{-1}$. □

General iterative methods

General iterative method has form $x^{(k)} = Tx^{(k-1)} + c$ for $k = 1, 2, \ldots$.

Example (Jacobi and GS are iterative methods)

- Jacobi iterative method:

$$x^{(k)} = D^{-1}(b - Rx^{(k-1)}) = -(D^{-1}R)x^{(k-1)} + D^{-1}b$$

  So $T = -D^{-1}R$ and $c = D^{-1}b$.

- Gauss-Seidel iterative method:

$$x^{(k)} = L^{-1}(b - Ux^{(k-1)}) = -(L^{-1}U)x^{(k-1)} + L^{-1}b$$

  So $T = -L^{-1}U$ and $c = L^{-1}b$.

General iterative methods

Theorem (Sufficient and necessary condition of convergence)

For any initial $x^{(0)}$, the sequence $\{x^{(k)}\}_k$ defined by

$$x^{(k)} = Tx^{(k-1)} + c$$

converges to the unique solution of $x = Tx + c$ iff $\rho(T) < 1$.

Proof.

($\Leftarrow$) Suppose $\rho(T) < 1$. Then

$$\begin{aligned} x^{(k)} &= Tx^{(k-1)} + c = T(Tx^{(k-2)} + c) + c = T^2x^{(k-2)} + (I + T)c \\ &= \cdots = T^kx^{(0)} + (I + T + \cdots + T^k)c \end{aligned}$$

Note $\rho(T) < 1 \Rightarrow T^k \to 0$ and $(I + T + \cdots + T^k) \to (I - T)^{-1}$, so $x^{(k)} \to (I - T)^{-1}c$, the unique solution of $x = Tx + c$. $\square$

General iterative methods

Proof.

($\Rightarrow$) Let x^* be the unique solution of $x = Tx + c$. Then for any $z \in \mathbb{R}^n$, we set initial $x^{(0)} = x^* - z$. Then

$$\begin{aligned} x^* - x^{(k)} &= (Tx^* + c) - (Tx^{(k-1)} + c) = T(x^* - x^{(k-1)}) \\ &= \cdots = T^k(x^* - x^{(0)}) = T^k z \to 0 \end{aligned}$$

This implies $\rho(T) < 1$. $\square$

General iterative methods

Corollary (Linear convergence rate)

If $\|T\| < 1$ *for any matrix norm* $\|\cdot\|$, *and* c *is given, then* $\{x^{(k)}$ *generated by* $x^{(k)} = Tx^{(k-1)} + c$ *converges to the unique solution* x^* *of* $x = Tx + c$. *Moreover*

1. $\|x^* - x^{(k)}\| \leq \|T\|^k \|x^* - x^{(0)}\|$.
2. $\|x^* - x^{(k)}\| \leq \frac{\|T\|^k}{1-\|T\|} \|x^{(1)} - x^{(0)}\|$.

Proof.

1. Note $\rho(T) \leq \|T\| < 1$. Follow $(\Rightarrow)$ part of the theorem above.
2. Note that $\|x^* - x^{(1)}\| \leq \|T\| \|x^* - x^{(0)}\|$ and hence $\|x^{(1)} - x^{(0)}\| \geq \|x^* - x^{(0)}\| - \|x^* - x^{(1)}\| \geq (1 - \|T\|)\|x^* - x^{(0)}\|$.

□

General iterative methods

Theorem (Jacobi and GS are convergent)

If A is strictly diagonally dominant, then from any initial $x^{(0)}$ both Jacobi and Gauss-Seidel iterative methods generate sequences that converge to the unique solution of $Ax = b$.

Proof.

For Jacobi, we can show $\rho(D^{-1}R) < 1$: if not, then exists ew λ such that $|\lambda| = \rho(D^{-1}R) \geq 1$, and ev $x \neq 0$ such that $D^{-1}Rx = \lambda x$, i.e., $(R + \lambda D)x = 0$ or $R + \lambda D$ invertible, contradiction to $A = D + R$ strictly diagonally dominant given $|\lambda| \geq 1$. Similar for GS. □

Relaxation techniques

The theory of general iterative methods suggest using a matrix T with smaller spectrum $\rho(T)$. To this end, we can use the relaxation technique to modify the iterative scheme.

- Original Gauss-Seidel iterative method:

$$x^{(k)} = -(L^{-1}U)x^{(k-1)} + L^{-1}b$$

- **Successive Over-Relaxation**[1] (SOR) for Gauss-Seidel iterative method ($\omega > 1$):

$$x^{(k)} = (D - \omega L)^{-1}[(1-\omega)D + \omega U]x^{(k-1)} + \omega(D - \omega L)^{-1}b$$

where $D, -L, -U$ are the diagonal, strict lower, and strict upper triangular parts of A, respectively.

[1] $Ax = b \Leftrightarrow \omega(-L + D - U)x = \omega b \Leftrightarrow (D - \omega L)x = ((1-\omega)D + \omega U)x + \omega b.$

Relaxation techniques

Example

Compare Gauss-Seidel and SOR with $\omega = 1.25$, both using $x^{(0)} = (1, 1, 1)^\top$ as initial, to solve the system:

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

Relaxation techniques

Solution: Compare with true solution $(3, 4, -5)^\top$, we get:

Gauss-Seidel:

k	0	1	2	3	4	5	6	7
$x_1^{(2)}$	1	5.250000	3.1406250	3.0878906	3.0549316	3.0343323	3.0214577	3.0134110
$x_2^{(2)}$	1	3.812500	3.8828125	3.9667578	3.9542236	3.9713898	3.9821186	3.9888241
$x_3^{(2)}$	1	-5.046875	-5.0292969	-5.0183105	-5.0114441	-5.0071526	-5.0044703	-5.0027940

Successive Over-Relaxation:

k	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1	6.312500	2.6223145	3.1333027	2.9570512	3.0037211	2.9963276	3.0000498
$x_2^{(k)}$	1	3.5195313	3.9585266	4.0102646	4.0074838	4.0029250	4.0009262	4.0002586
$x_3^{(k)}$	1	-6.6501465	-4.6004238	-5.0966863	-4.9734897	-5.0057135	-4.9982822	-5.0003486

The 5th iteration of SOR is better than 7th of GS.

Relaxation techniques

Theorem (Kahan's theorem)

If all diagonal entries of A are nonzero, then $\rho(T_\omega) \geq |\omega - 1|$, where $T_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega U]$.

Proof.

Let $\lambda_1, \ldots, \lambda_n$ be the ew of T_ω, then

$$\prod_{i=1}^{n} \lambda_i = \det(T_\omega) = \det(D)^{-1} \det((1-\omega)D) = (1-\omega)^n$$

since $D - \omega L$ and $(1-\omega)D + \omega U$ are lower/upper triangular matrices. Hence $\rho(T_\omega)^n \geq \prod_{i=1}^{n} |\lambda_i| = |1-\omega|^n$. □

This result says that SOR can converge only if $|\omega - 1| < 1$.

Relaxation techniques

Theorem (Ostrowski-Reich theorem)

If A is positive definite and $|\omega - 1| < 1$, then the SOR converges starting from any initial $x^{(0)}$.

Theorem

If A is positive definite and tridiagonal, then $\rho(T_g) = [\rho(T_j)]^2 < 1$, where T_g and T_j are the T matrices of GS and Jacobi methods respectively, and the optimal ω for SOR is

$$\omega = \frac{2}{1 + \sqrt{1 - (\rho(T_j))^2}}$$

With this choice of ω, the spectrum $\rho(T_\omega) = \omega - 1$.

Iterative refinement

Definition (Residual)

Let $\tilde{x}$ be an approximation to the solution x of linear system $Ax = b$. Then $r = b - A\tilde{x}$ is called the **residual** *of approximation $\tilde{x}$.*

Remark

It seems intuitive that a small residual r implies a close approximation $\tilde{x}$ to x. However, it is not always true.

Iterative refinement

Example (small residual $\not\Rightarrow$ small approximation error)

The linear system $Ax = b$ is given by

$$\begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3.0001 \end{bmatrix}$$

has a unique solution $x = (1, 1)^\top$. Determine the residual vector r of a poor approximation $\tilde{x} = (3, -0.0001)^\top$.

Solution: The residual is

$$r = b - A\tilde{x} = \begin{bmatrix} 3 \\ 3.0001 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -0.0001 \end{bmatrix} = \begin{bmatrix} 0.0002 \\ 0 \end{bmatrix}$$

So $\|r\|_\infty = 0.0002$ is small but $\|\tilde{x} - x\|_\infty = 2$ is large.

Iterative refinement

Theorem (Relation between residual and error)

Suppose A is nonsingular, and $\tilde{x}$ is an approximation to the solution x of $Ax = b$, and $r = b - A\tilde{x}$ is the residual vector of $\tilde{x}$, then for any norm, there is

$$\|x - \tilde{x}\| \leq \|r\| \cdot \|A^{-1}\|$$

Moreover, if $x \neq 0$ and $b \neq 0$, then there is

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r\|}{\|b\|}$$

If $\|A\|\|A^{-1}\|$ is large, then small $\|r\|$ does not guarantee small $\|x - \tilde{x}\|$.

Iterative refinement

Proof.

Since x is a solution, we have $Ax = b$, we have $r = b - A\tilde{x} = Ax - A\tilde{x} = A(x - \tilde{x})$. Since A is nonsingular, we have $x - \tilde{x} = A^{-1}r$, and hence

$$\|x - \tilde{x}\| = \|A^{-1}r\| \leq \|r\| \cdot \|A^{-1}\|$$

If $x \neq 0$ and $b \neq 0$, from $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$ we have $1/\|x\| \leq \|A\|/\|b\|$. Multiplying this to the inequality above, we get

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r\|}{\|b\|}$$

□

Iterative refinement

The number $\|A\| \cdot \|A^{-1}\|$ provide an indication between the error of approximation $\|x - \tilde{x}\|$ and size of residual r. So the larger $\|A\| \cdot \|A^{-1}\|$ is, the less power we have to control error using residual.

Definition (Condition number)

The **condition number** *of a nonsingular matrix A relative to a norm $\|\cdot\|_p$ is*

$$K_p(A) = \|A\|_p \cdot \|A^{-1}\|_p$$

The subscript p is often omitted if it's clear from context or it's not important.

Condition number

Remark

- *The condition number* $K(A) \geq 1$*:*

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = K(A)$$

- *A matrix* A *is called* **well-conditioned** *if* $K(A)$ *is close to 1.*
- *A matrix* A *is called* **ill-conditioned** *if* $K(A) \gg 1$*.*

Condition number

Example (Condition number)

Determine the condition number of matrix

$$A = \begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix}$$

Condition number

Solution: Let's use l_∞ norm. Then

$$\|A\|_\infty = \max\{|1| + |2|, |1.0001| + |2|\} = 3.0001$$

Furthermore, there is

$$A^{-1} = \begin{bmatrix} -10000 & 10000 \\ 5000.5 & -5000 \end{bmatrix}$$

and hence $\|A^{-1}\|_\infty = 20000$. Therefore

$$K(A) = \|A\| \cdot \|A^{-1}\| = 3.0001 \times 20000 = 60002$$

Iterative refinement

Suppose $\tilde{x}$ is our current approximation to x. Let $\tilde{y} = x - \tilde{x}$, then $A\tilde{y} = A(x - \tilde{x}) = Ax - A\tilde{x} = b - A\tilde{x} = r$. If we can solve for $\tilde{y}$ here, we would get a new approximation $\tilde{x} + \tilde{y}$, expectedly to approximate x better.

This procedure is called **iterative refinement**.

Iterative refinement

Given A and b, Iterative Refinement first applies Gauss eliminations to $Ax = b$ and obtains approximation x.

Then, for each iteration $k = 1, 2, \ldots, N$, do the following:

- Compute residual $r = b - Ax$;
- Solve y from $Ay = r$ using the same Gauss elimination steps.
- Set $x \leftarrow x + y$

The actual Iterative Refinement algorithm can also find approximation of condition number $K_\infty(A)$ (See textbook).

Perturbed linear system

In reality, A and b may be perturbed by noise or rounding errors δA and δb. Therefore, we are actually solving

$$(A + \delta A)x = b + \delta b$$

rather than $Ax = b$. This won't cause much issue if A is well-conditioned, but could be a problem otherwise.

Perturbed linear system

Theorem

Suppose A is nonsingular and $\|\delta A\| < \frac{1}{\|A^{-1}\|}$, then the solution $\tilde{x}$ of perturbed linear system $(A+\delta A)x = b + \delta b$ has an error estimate given by

$$\frac{\|x-\tilde{x}\|}{\|x\|} \leq \frac{K(A)\|A\|}{\|A\| - K(A)\|\delta A\|}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right)$$

where x is the solution of the original linear system $Ax = b$.

Note that $K(A)\|\delta A\| = \|A\|\|A^{-1}\|\|\delta A\| < \|A\|$ so the denominator is positive.

Conjugate gradient method

Conjugate gradient (CG) method is particularly efficient for solving linear systems with large, sparse, and positive definite matrix A.

Equipped with proper preconditioning, CG can often reach very good result in $\sqrt{n}$ iterations (n the size of system).

The per-iteration cost is also low when A is sparse.

An alternate perspective of linear system

Theorem

Let A be positive definite, then x^* is the solution of $Ax = b$ iff x^* is the minimizer of

$$g(x) = \frac{1}{2}x^\top Ax - b^\top x$$

Proof.

Note that $\nabla g(x) = Ax - b$ and $\nabla^2 g(x) = A \succ 0$, so $g(x^*) = Ax^* - b = 0$ iff x^* is a minimizer of $g(x)$. □

An alternate perspective of linear system

We have following observations:

- $r = b - Ax = -\nabla g(x)$ is the residual and also the steepest descent direction of $g(x)$ (recall that $\nabla g(x)$ is the steepest ascent direction).
- It seems intuitive to update $x \leftarrow x + t \cdot r = x - t\nabla g(x)$ with proper step size t.
- It turns out that we can find such t that makes the most progress.
- This method is called the "steepest descent method".
- However, it converges slowly and exhibits "zigzag" path for ill-conditioned A.

A-orthogonal

Conjugate gradient method amends this issue of steepest descent. To derive CG, we first present the following concept:

Definition

*Two vectors v and w are called A-***orthogonal** *if $\langle v, Aw\rangle = 0$.*

Theorem

If A is positive definite, then there exists a set of independent vectors $\{v^{(1)}, \cdots, v^{(n)}\}$ such that $\langle v^{(i)}, Av^{(j)}\rangle = 0$ for all $i \neq j$.

Key idea of CG

Given previous estimate $x^{(k-1)}$ and a "search direction" $v^{(k)}$, CG will find scalars t_k and s_k to update x and v:

$$x^{(k)} = x^{(k-1)} + t_k v^{(k)}$$
$$v^{(k+1)} = r^{(k)} + s_k v^{(k)}$$

(where $r^{(k)} = b - Ax^{(k)}$), such that:

$$\langle v^{(k+1)}, Av^{(j)} \rangle = 0, \quad \forall j \leq k$$
$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad \forall j \leq k$$

If this can be done, then $\{v^{(1)}, \cdots, v^{(n)}\}$ is A-orthogonal.

Derivation of t_k and s_k

The main tool is mathematical induction: given $x^{(0)}$, first set $v^{(0)} = 0$, $r^{(0)} = b - Ax^{(0)}$, $v^{(1)} = r^{(0)}$. So

$$\langle v^{(k+1)}, Av^{(j)} \rangle = 0, \quad \forall j \leq k$$
$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad \forall j \leq k$$

is true for $k = 0$. Assume they hold for $k - 1$, we need to find t_k and s_k such that they also hold for k.

Derivation of t_k and s_k

We first find t_k: note that

$$r^{(k)} = b - Ax^{(k)} = b - A(x^{(k-1)} + t_k v^{(k)}) = r^{(k-1)} - t_k A v^{(k)}$$

Therefore, by induction hypothesis, there is

$$\begin{aligned}\langle r^{(k)}, v^{(j)} \rangle &= \langle r^{(k-1)} - t_k A v^{(k)}, v^{(j)} \rangle \\ &= \begin{cases} 0 & \text{if } j \leq k-1, \\ \langle r^{(k-1)}, v^{(k)} \rangle - t_k \langle v^{(k)}, A v^{(k)} \rangle, & \text{if } j = k \end{cases}\end{aligned}$$

So we just need

$$t_k = \frac{\langle r^{(k-1)}, v^{(k)} \rangle}{\langle v^{(k)}, A v^{(k)} \rangle}$$

to make $\langle r^{(k)}, v^{(j)} \rangle = 0$.

Derivation of t_k and s_k

Then we find s_k: by the update of $v^{(k+1)}$, we have

$$\begin{aligned}\langle v^{(k+1)}, Av^{(j)}\rangle &= \langle r^{(k)} + s_k v^{(k)}, Av^{(j)}\rangle \\ &= \begin{cases}\langle r^{(k)}, Av^{(j)}\rangle, & \text{if } j \le k-1 \\ \langle r^{(k)}, Av^{(k)}\rangle + s_k\langle v^{(k)}, Av^{(k)}\rangle, & \text{if } j = k\end{cases}\end{aligned}$$

Note that $Av^{(j)} = \frac{Ax^{(j)} - Ax^{(j-1)}}{t_j} = \frac{r^{(j-1)} - r^{(j)}}{t_j}$, and $r^{(j-1)} - r^{(j)}$ is linear combination of $v^{(j-1)}, v^{(j)}, v^{(j+1)}$, so $\langle r^{(k)}, Av^{(j)}\rangle = 0$ for $j \le k-1$ due to induction hypothesis. Hence we just need

$$s_k = -\frac{\langle r^{(k)}, Av^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$$

to make $\langle r^{(k)}, Av^{(j)}\rangle = 0$ for all $j \le k$.

Derivation of t_k and s_k

We can further simply t_k and s_k:

Since that $v^{(k)} = r^{(k-1)} + s_{k-1}v^{(k-1)}$ and $\langle r^{(k-1)}, v^{(k-1)} \rangle = 0$, we have

$$t_k = \frac{\langle r^{(k-1)}, v^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle} = \frac{\langle r^{(k-1)}, r^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$$

Since $r^{(k-1)} = v^{(k)} - s_{k-1}v^{(k-1)}$, we have $\langle r^{(k)}, r^{(k-1)} \rangle = 0$. Since $Av^{(k)} = \frac{Ax^{(k)} - Ax^{(k-1)}}{t_k} = \frac{r^{(k-1)} - r^{(k)}}{t_k}$, we have $\langle r^{(k)}, Av^{(k)} \rangle = -\frac{\langle r^{(k)}, r^{(k)} \rangle}{t_k}$. Combining t_k expression above, we have

$$s_k = -\frac{\langle r^{(k)}, Av^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle} = -\frac{-\frac{\langle r^{(k)}, r^{(k)} \rangle}{t_k}}{\frac{\langle r^{(k-1)}, r^{(k-1)} \rangle}{t_k}} = \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k-1)}, r^{(k-1)} \rangle}$$

Conjugate gradient method

Since $\langle r^{(n)}, v^{(k)} \rangle = 0$ for all $k = 1, \ldots, n$ and the A-orthogonal set $\{v^{(1)}, \cdots, v^{(n)}\}$ is independent when A is positive definite, we know $r^{(n)} = b - Ax^{(n)} = 0$, i.e., $x^{(n)}$ is the solution.

This shows that CG converges in at most n steps, assuming all arithmetics are exact.

Conjugate gradient method

- Input: $x^{(0)}$, $r^{(0)} = b - Ax^{(0)}$, $v^{(1)} = r^{(0)}$.
- Repeat the following for $k = 1, \ldots, n$ until $r^{(k)} = 0$:

$$
\begin{aligned}
t_k &= \frac{\langle r^{(k-1)}, r^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle} \\
x^{(k)} &= x^{(k-1)} + t_k v^{(k)} \\
r^{(k)} &= r^{(k-1)} - t_k A v^{(k)} \\
s_k &= \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k-1)}, r^{(k-1)} \rangle} \\
v^{(k+1)} &= r^{(k)} + s_k v^{(k)}
\end{aligned}
$$

- Output: $x^{(k)}$.

Preconditioning

The convergence rate of CG can be greatly improved by **preconditioning**. Preconditioning reduces condition number of A first if A is ill-conditioned. With preconditioning, CG usually converges in $\sqrt{n}$ steps.

The preconditioning is done by using some nonsingular matrix C, we can get $\tilde{A} = C^{-1}A(C^{-1})^{\top}$ such that $K(\tilde{A}) \ll K(A)$.

Now by defining $\tilde{x} = C^{\top}x$ and $\tilde{b} = C^{-1}b$, we obtain a new linear system $\tilde{A}\tilde{x} = \tilde{b}$, which is equivalent to $Ax = b$. Then we can apply CG to the new system $\tilde{A}\tilde{x} = \tilde{b}$.

Preconditioner

There are various methods to choose the preconditioner C.

- Choose $C = \text{diag}(\sqrt{a_{11}}, \ldots, \sqrt{a_{nn}})$.
- Approximate Cholesky's factorization $LL^\top \approx A$ (by ignoring small values in A) and set $C = L$ (then $C^{-1}A(C^{-1})^\top \approx L^{-1}(LL^\top)L^{-T} = I$).
- Many others...

Preconditioned conjugate gradient method

- Input: Preconditioner C, $x^{(0)}$, $r^{(0)} = b - Ax^{(0)}$, $w^{(0)} = C^{-1}r^{(0)}$, $v^{(1)} = C^{-T}w^{(0)}$.
- Repeat the following for $k = 1, \ldots, n$ until $r^{(k)} = 0$:

$$\begin{aligned}
\tilde{t}_k &= \frac{\langle w^{(k-1)}, w^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle} \\
x^{(k)} &= x^{(k-1)} + \tilde{t}_k v^{(k)} \\
r^{(k)} &= r^{(k-1)} - \tilde{t}_k Av^{(k)} \\
w^{(k)} &= C^{-1}r^{(k)} \\
\tilde{s}_k &= \frac{\langle w^{(k)}, w^{(k)} \rangle}{\langle w^{(k-1)}, w^{(k-1)} \rangle} \\
v^{(k+1)} &= C^{-\top}w^{(k)} + \tilde{s}_k v^{(k)}
\end{aligned}$$

- Output: $x^{(k)}$.

A comparison

Example

Given A and b below, we use the methods above to solve $Ax = b$.

$$A = \begin{bmatrix} 0.2 & 0.1 & 1 & 1 & 0 \\ 0.1 & 4 & -1 & 1 & -1 \\ 1 & -1 & 60 & 0 & -2 \\ 1 & 1 & 0 & 8 & 4 \\ 0 & -1 & -2 & 4 & 700 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

True solution is

$$x^* = \begin{bmatrix} 7.859713071 \\ 0.4229264082 \\ -0.07359223906 \\ -0.5406430164 \\ 0.01062616286 \end{bmatrix}$$

A comparison

A comparison of Jacobi, Gauss-Seidel, SOR, CG, and PCG on the problem above.

Method	Number of Iterations	$\mathbf{x}^{(k)}$	$\|\mathbf{x}^* - \mathbf{x}^{(k)}\|_\infty$
Jacobi	49	$(7.86277141, 0.42320802, -0.07348669, -0.53975964, 0.01062847)^t$	0.00305834
Gauss-Seidel	15	$(7.83525748, 0.42257868, -0.07319124, -0.53753055, 0.01060903)^t$	0.02445559
SOR ($\omega = 1.25$)	7	$(7.85152706, 0.42277371, -0.07348303, -0.53978369, 0.01062286)^t$	0.00818607
Conjugate Gradient	5	$(7.85341523, 0.42298677, -0.07347963, -0.53987920, 0.008628916)^t$	0.00629785
Conjugate Gradient (Preconditioned)	4	$(7.85968827, 0.42288329, -0.07359878, -0.54063200, 0.01064344)^t$	0.00009312

Section 4

Boundary Value Problems for ODEs

BVP for ODE

We study numerical solution for boundary value problem (BVP).

If the BVP involves first-order ODE, then

$$y'(x) = f(x, y(x)), \quad a \le x \le b, \quad y(a) = \alpha.$$

This reduces to an initial value problem we learned before.

So we start by considering second-order ODE:

$$\begin{cases} y''(x) = f(x, y(x), y'(x)), & a \le x \le b \\ y(a) = \alpha, \ y(b) = \beta \end{cases}$$

Existence of solutions

Consider the BVP with second-order ODE:

$$\begin{cases} y''(x) = f(x, y(x), y'(x)), \quad a \le x \le b \\ y(a) = \alpha, \ y(b) = \beta \end{cases}$$

Theorem (Existence and uniqueness of solution)

Let $D = [a, b] \times \mathbb{R} \times \mathbb{R}$. Suppose $f(x, y, y')$ satisfies:

1. *f is continuous on D,*
2. *$\frac{\partial f}{\partial y} > 0$ in D,*
3. *$\exists M > 0$ such that $|\frac{\partial f}{\partial y'}| \le M$ in D.*

Then the BVP has unique solution.

Existence of solutions

Example (Existence and uniqueness of solution)

Show that the BVP below has unique solution:

$$\begin{cases} y''(x) = -e^{-xy} + \sin(y'), \quad 1 \le x \le 2 \\ y(a) = 0, \ y(b) = 0 \end{cases}$$

Solution: We have $f(x, y, y') = -e^{-xy} - \sin(y')$. It is obvious that f is continuous. Moreover $\partial_y f = xe^{-xy} > 0$, and $|\partial_{y'} f| = |-\cos(y')| \le 1$. So the BVP has unique solution by the theorem above.

BVP with linear ODE

Now we first consider a linear second-order ODE:

$$\begin{cases} y'' = p(x)y' + q(x)y + r(x), \quad a \le x \le b \\ y(a) = \alpha, \ y(b) = \beta \end{cases}$$

where $p, q, r : [a, b] \to \mathbb{R}$ are given functions.

Corollary

If p, q, r are continuous on $[a, b]$, $q > 0$ for all x, then the BVP with linear ODE above has a unique solution.

Proof.

Set $f = py' + qy + r$. Note that p is bounded since it is continuous on $[a, b]$. Hence the theorem (check the 3 conditions) above applies. □

Linear shooting method

Now we consider how to solve BVP with linear ODE:

$$\begin{cases} y'' = py' + qy + r, \quad a \le x \le b \\ y(a) = \alpha,\ y(b) = \beta \end{cases}$$

We consider two associated initial value problems:

$$\begin{cases} y_1'' = py_1' + qy_1 + r, \quad a \le x \le b \\ y_1(a) = \alpha,\ y_1'(a) = 0 \end{cases}$$

$$\begin{cases} y_2'' = py_2' + qy_2, \quad a \le x \le b \\ y_2(a) = 0,\ y_2'(a) = 1 \end{cases}$$

Linear shooting method

Suppose the solution y to the BVP can be written as $y = y_1 + cy_2$ for some constant c (to be determined soon), where y_1, y_2 are the solutions to the two IVPs. Then y satisfies the ODE:

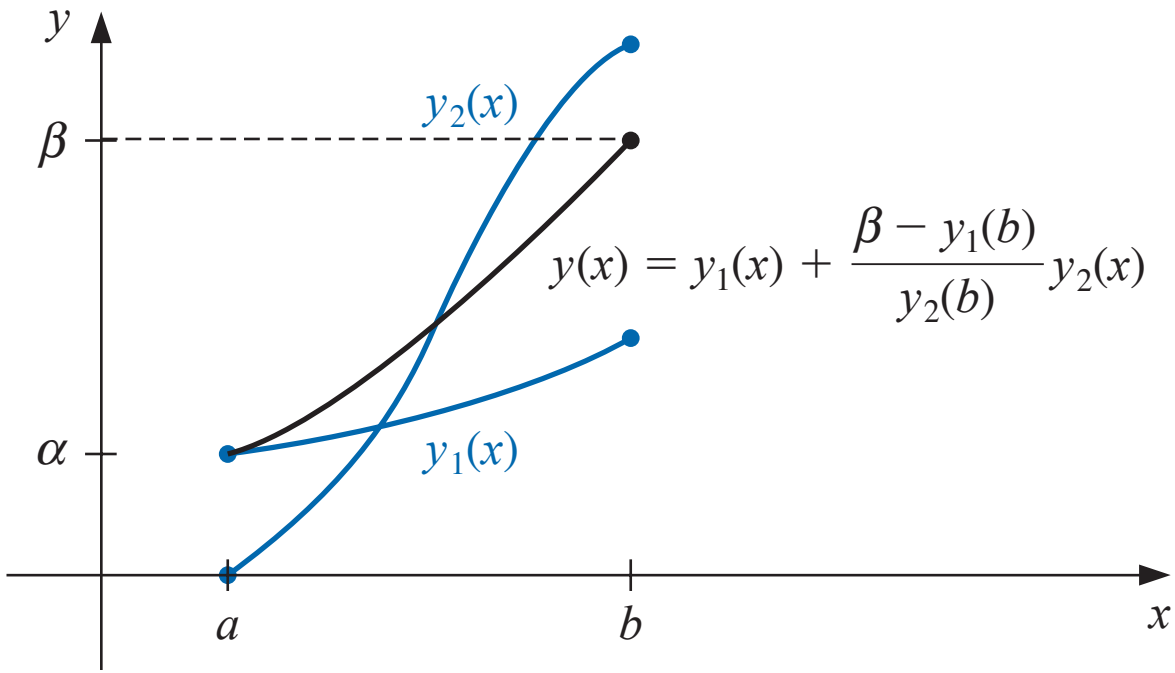
$$\begin{aligned} y'' &= (y_1 + cy_2)'' = y_1'' + cy_2'' \\ &= (py_1' + qy_1 + r) + c(py_2' + qy_2) \\ &= p(y_1 + cy_2)' + q(y_1 + cy_2) + r \\ &= py' + qy + r \end{aligned}$$

To make y satisfy the boundary conditions, we need c such that

$$\begin{aligned} y(a) &= y_1(a) + cy_2(a) = y_1(a) = \alpha \\ y(b) &= y_1(b) + cy_2(b) = \beta \end{aligned}$$

So we just need to set $c = \frac{\beta - y_1(b)}{y_2(b)}$.

Linear shooting method



Here y_1, y_2 are two shot trajectories based on their initial height and angle. Their linear combination $y_1 + \frac{\beta - y_1(b)}{y_2(b)} y_2$ is the solution y.

Linear shooting method

Steps of the linear shooting method:

1. Partition $[a, b]$ into N equal subintervals.
2. Solve y_1 and y_2 from their own IVPs (e.g., using RK4) $(u_1 = y_1, u_2 = y_1', v_1 = y_2, v_2 = y_2')$, and get $\{u_{1,i}, v_{1,i} : 0 \leq i \leq N\}$
3. Set $c = (\beta - u_{1,N})/v_{1,N}$, and set $w_{1,i} = u_{1,i} + cv_{1,i}$ for $0 \leq i \leq N$.

Linear shooting method

Example (Linear shooting method)

Solve the BVP with $N = 10$.

$$\begin{cases} y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2}, & 1 \le x \le 2 \\ y(1) = 1,\ y(2) = 2 \end{cases}$$

Solution: Partition $[1, 2]$ into $N = 10$ subintervals, and solve

$$\begin{cases} y_1'' = -\frac{2}{x}y_1' + \frac{2}{x^2}y_1 + \frac{\sin(\ln x)}{x^2}, & 1 \le x \le 2 \\ y_1(1) = 1,\ y_1'(1) = 0 \end{cases}$$

$$\begin{cases} y_2'' = -\frac{2}{x}y_2' + \frac{2}{x^2}y_2, & 1 \le x \le 2 \\ y_2(1) = 0,\ y_2'(1) = 1 \end{cases}$$

Then set $w_i = u_{1,i} + \frac{2-u_{1,N}}{v_{1,N}} v_{1,i}$ for $i = 0, \ldots, 10$.

Linear shooting method

Numerical result:

x_i	$u_{1,i} \approx y_1(x_i)$	$v_{1,i} \approx y_2(x_i)$	$w_i \approx y(x_i)$	$y(x_i)$	$\lvert y(x_i) - w_i \rvert$
1.0	1.00000000	0.00000000	1.00000000	1.00000000	
1.1	1.00896058	0.09117986	1.09262917	1.09262930	1.43×10^{-7}
1.2	1.03245472	0.16851175	1.18708471	1.18708484	1.34×10^{-7}
1.3	1.06674375	0.23608704	1.28338227	1.28338236	9.78×10^{-8}
1.4	1.10928795	0.29659067	1.38144589	1.38144595	6.02×10^{-8}
1.5	1.15830000	0.35184379	1.48115939	1.48115942	3.06×10^{-8}
1.6	1.21248372	0.40311695	1.58239245	1.58239246	1.08×10^{-8}
1.7	1.27087454	0.45131840	1.68501396	1.68501396	5.43×10^{-10}
1.8	1.33273851	0.49711137	1.78889854	1.78889853	5.05×10^{-9}
1.9	1.39750618	0.54098928	1.89392951	1.89392951	4.41×10^{-9}
2.0	1.46472815	0.58332538	2.00000000	2.00000000	

This accurate result is due to $O(h^4)$ of RK4 used for the two IVPs.

Round-off error in linear shooting method

If $y_1(x)$ grows too fast such that $y_1(b) \gg \beta$, then

$$\frac{\beta - y_1(b)}{y_2(b)} \approx -\frac{y_1(b)}{y_2(b)}$$

which is prone to round-off error.

In this case, we can solve the two IVPs backward in x:

$$\begin{cases} y_1'' = py_1' + qy_1 + r, \quad a \le x \le b \\ y_1(b) = \beta, \ y_1'(b) = 0 \end{cases}$$

$$\begin{cases} y_2'' = py_2' + qy_2, \quad a \le x \le b \\ y_2(b) = 0, \ y_2'(b) = 1 \end{cases}$$

and set $y(x) = y_1(x) + \frac{\alpha - y_1(a)}{y_2(a)} y_2(x)$ for $a \le x \le b$

Nonlinear shooting method

Consider the BVP with nonlinear ODE (f is a nonlinear function):

$$\begin{cases} y'' = f(x, y, y'), \quad a \le x \le b \\ y(a) = \alpha, \ y(b) = \beta \end{cases}$$

Suppose we try to solve the IVP with some given t:

$$\begin{cases} y'' = f(x, y, y'), \quad a \le x \le b \\ y(a) = \alpha, \ y'(a) = t \end{cases}$$

and obtain solution $y(x, t)$ (since the solution depends on t) for $a \le x \le b$.

Then we hope to find t such that $y(b, t) = \beta$.

Secant method for nonlinear shooting

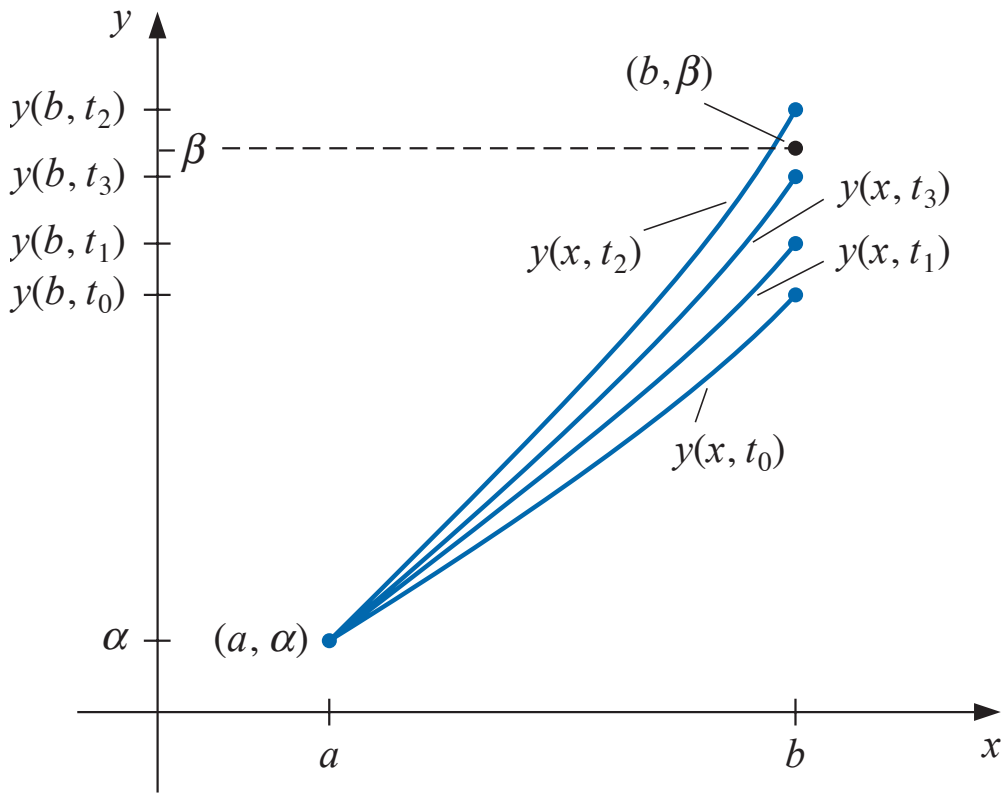
Suppose we have two initials t_0, t_1, then we use the secant method to solve $y(b,t) - \beta = 0$ by iterating

$$t_k = t_{k-1} - \frac{(y(b,t_{k-1}) - \beta)(t_{k-1} - t_{k-2})}{y(b,t_{k-1}) - y(b,t_{k-2})}$$

For each k, we need to compute $y(b,t_k)$ by solving the IVP:

$$\begin{cases} y'' = f(x,y,y'), \quad a \le x \le b \\ y(a) = \alpha, \ y'(a) = t_k \end{cases}$$

Nonlinear shooting method



Here $y(x, t_k)$ is "shooting" at an angle (with slope t_k) and try to "hit" β at $x = b$.

Newton's method for nonlinear shooting

We can also consider Newton's method to $y(b,t)-\beta=0$ for fewer iterations:

$$t_k = t_{k-1} - \frac{y(b,t_{k-1})-\beta}{\partial_t y(b,t_{k-1})}$$

However, we need to know $\partial_t y(b,t)$ ...

We denote the solution of IVP below by $y(x,t)$:

$$\begin{cases} y''(x,t) = f(x,y(x,t),y'(x,t)), \quad a \le x \le b \\ y(a,t) = \alpha,\ y'(a,t) = t \end{cases}$$

where $y' = \partial_x y$ and $y'' = \partial_x^2 y$ (i.e., the $'$ is on x).

Newton's method for nonlinear shooting

Taking partial derivatives with respect to t above yields:

$$\begin{cases} \partial_t y'' = \partial_y f \cdot \partial_t y + \partial_{y'} f \cdot \partial_t y', & a \le x \le b \\ \partial_t y(a,t) = 0, \ \partial_t y'(a,t) = 1 \end{cases}$$

Denote $z(x,t) = \partial_t y(x,t)$. Suppose ∂_x and ∂_t can exchange, then

$$\begin{cases} z''(x,t) = \partial_y f \cdot z(x,t) + \partial_{y'} f \cdot z'(x,t), & a \le x \le b \\ z(a,t) = 0, \ z'(a,t) = 1 \end{cases}$$

and set $\partial_t y(b,t) = z(b,t)$.

Newton's method for nonlinear shooting

Steps of Newton's method for nonlinear shooting:

1. Initialize t_0 (e.g. $t_0 = \frac{\beta-\alpha}{b-a}$). Set $k = 1$.
2. For $t = t_{k-1}$, solve $y(x,t)$ and $z(x,t)$ from

$$\begin{cases} y''(x,t) = f(x, y(x,t), y'(x,t)), & a \le x \le b \\ y(a,t) = \alpha, \ y'(a,t) = t \end{cases}$$

$$\begin{cases} z''(x,t) = \partial_y f \cdot z(x,t) + \partial_{y'} f \cdot z'(x,t), & a \le x \le b \\ z(a,t) = 0, \ z'(a,t) = 1 \end{cases}$$

   and set $t_k = t_{k-1} - \frac{y(b,t_{k-1})-\beta}{z(b,t_{k-1})}$.
3. Set $k \leftarrow k+1$ and go to Step 2.

Newton's method for nonlinear shooting

Example (Newton's method for nonlinear BVP)

Solve the BVP with nonlinear ODE using Newton's method with $N = 20$ for maximal of 10 iterations or $|w_N(t_k) - y(3)| \leq 10^{-5}$:

$$\begin{cases} y'' = \frac{1}{8}\left(32 + 2x^3 - yy'\right), & 1 \leq x \leq 3 \\ y(1) = 17,\ y(3) = \frac{43}{3} \end{cases}$$

Solution: Note that $\partial_y f = -\frac{1}{8}y'$ and $\partial_{y'} f = -\frac{1}{8}y$. For every t, the two IVPs are (note z depends on y but not vice versa):

$$\begin{cases} y'' = \frac{1}{8}\left(32 + 2x^3 - yy'\right), & 1 \leq x \leq 3 \\ y(1) = 17,\ y'(1) = t \end{cases}$$

$$\begin{cases} z'' = -\frac{1}{8}(y'z + yz'), & 1 \leq x \leq 3 \\ z(1) = 0,\ z'(1) = 1 \end{cases}$$

Nonlinear shooting using Newton's method

x_i	$w_{1,i}$	$y(x_i)$	$\lvert w_{1,i} - y(x_i)\rvert$
1.0	17.000000	17.000000	
1.1	15.755495	15.755455	4.06×10^{-5}
1.2	14.773389	14.773333	5.60×10^{-5}
1.3	13.997752	13.997692	5.94×10^{-5}
1.4	13.388629	13.388571	5.71×10^{-5}
1.5	12.916719	12.916667	5.23×10^{-5}
1.6	12.560046	12.560000	4.64×10^{-5}
1.7	12.301805	12.301765	4.02×10^{-5}
1.8	12.128923	12.128889	3.14×10^{-5}
1.9	12.031081	12.031053	2.84×10^{-5}
2.0	12.000023	12.000000	2.32×10^{-5}
2.1	12.029066	12.029048	1.84×10^{-5}
2.2	12.112741	12.112727	1.40×10^{-5}
2.3	12.246532	12.246522	1.01×10^{-5}
2.4	12.426673	12.426667	6.68×10^{-6}
2.5	12.650004	12.650000	3.61×10^{-6}
2.6	12.913847	12.913845	9.17×10^{-7}
2.7	13.215924	13.215926	1.43×10^{-6}
2.8	13.554282	13.554286	3.46×10^{-6}
2.9	13.927236	13.927241	5.21×10^{-6}
3.0	14.333327	14.333333	6.69×10^{-6}

Netwon's method requires solving two IVPs in each iteration, but converges much faster than secant method. Still sensitive to round-off errors if y or z increases rapidly.

Finite-difference method for linear problems

Idea: Partition $[a, b]$ into $N+1$ subintervals with nodes $a = x_0 < \cdots < x_{N+1} = b$ and step size $h = \frac{b-a}{N+1}$. Then approximate y', y'' by finite differences, and solve $w_i = y(x_i)$ for $0 \le i \le N+1$.

Recall the centered-difference approximation of $y'(x_i)$:

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(\eta_i^+)$$
$$y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{6}y'''(\eta_i^-)$$

where $\eta_i^{\pm}$ is between x_i and $x_{i\pm1}$. Then subtracting the two above:

$$y'(x_i) = \frac{y(x_{i+1}) - y(x_{i-1})}{2h} - \frac{h^2}{6}y'''(\eta_i)$$

for some $\eta_i \in (x_{i-1}, x_{i+1})$ due to IVT and $y \in C^3$.

Finite-difference method for linear problems

Similarly, we have the centered-difference approximation of $y''(x_i)$:

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^+)$$
$$y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^-)$$

where $\xi_i^{\pm}$ is between x_i and $x_{i\pm1}$. Then adding the two above:

$$y''(x_i) = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} - \frac{h^2}{12}y^{(4)}(\xi_i)$$

for some $\xi_i \in (x_{i-1}, x_{i+1})$ due to IVT and $y \in C^4$.

Finite-difference method for linear problems

Plugging the two identities about $y'(x_i)$ and $y''(x_i)$ above into $y'' = py' + qy + r$:

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} = p(x_i)\left[\frac{y(x_{i+1}) - y(x_{i-1})}{2h}\right] + q(x_i)\,y(x_i) + r(x_i) - \frac{h^2}{12}\left[2p(x_i)\,y'''(\eta_i) - y^{(4)}(\xi_i)\right]$$

which has truncation error $O(h^2)$.

Now we approximate $y(x_i)$ by w_i for $0 \le i \le N+1$. Note that $w_0 = y(a) = \alpha$ and $w_{N+1} = y(b) = \beta$, and for $i = 1, \ldots, N$:

$$\left(\frac{-w_{i+1} + 2w_i - w_{i-1}}{h^2}\right) + p(x_i)\left(\frac{w_{i+1} - w_{i-1}}{2h}\right) + q(x_i)\,w_i = -r(x_i)$$

Finite-difference method for linear problems

The equation above can be rearranged into

$$-\left(1+\frac{h}{2}p\left(x_i\right)\right)w_{i-1}+\left(2+h^2q\left(x_i\right)\right)w_i-\left(1-\frac{h}{2}p\left(x_i\right)\right)w_{i+1}=-h^2r\left(x_i\right)$$

This is a linear system $Aw = b$ where $w = (w_1, \ldots, w_N)^\top$, A is tridiagonal, and b is known.

Theorem

Suppose that p, q, r are continuous on $[a, b]$ and $q \geq 0$, then the tridiagonal linear system $Aw = b$ has a unique solution provided that $h < 2/L$ where $L = \max_{a \leq x \leq b} |p(x)|$.

Finite-difference method for linear problems

Example (Finite-difference method for linear problems)

Solve the BVP below using finite difference method with $N = 9$:

$$\begin{cases} y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2}, & 1 \le x \le 2 \\ y(1) = 1,\ y(2) = 2 \end{cases}$$

Solution: Note that $p(x) = -2/x$, $q(x) = 2/x^2$, and $r(x) = \sin(\ln x)/x^2$. Step size $h = (b - a)/(N + 1) = 0.1$.

Finite-difference method for linear problems

x_i	w_i	$y(x_i)$	$\lvert w_i - y(x_i)\rvert$
1.0	1.00000000	1.00000000	
1.1	1.09260052	1.09262930	2.88×10^{-5}
1.2	1.18704313	1.18708484	4.17×10^{-5}
1.3	1.28333687	1.28338236	4.55×10^{-5}
1.4	1.38140205	1.38144595	4.39×10^{-5}
1.5	1.48112026	1.48115942	3.92×10^{-5}
1.6	1.58235990	1.58239246	3.26×10^{-5}
1.7	1.68498902	1.68501396	2.49×10^{-5}
1.8	1.78888175	1.78889853	1.68×10^{-5}
1.9	1.89392110	1.89392951	8.41×10^{-6}
2.0	2.00000000	2.00000000	

The error is $O(h^2)$, which is worse than the linear shooting method.

Finite-difference method for linear problems

We can improve the error order by Richardson's extrapolation since the truncation errors are in even orders of h.

Consider the same example above, we use step sizes $h = 0.1$, 0.05, and 0.025 to compute $w(h = 0.1)$, $w(h = 0.05)$ and $w(h = 0.025)$, and compute

$$\text{Ext}_{1i} = \frac{4w_i(h = 0.05) - w_i(h = 0.1)}{3}$$

$$\text{Ext}_{2i} = \frac{4w_i(h = 0.025) - w_i(h = 0.05)}{3}$$

$$\text{Ext}_{3i} = \frac{16\text{Ext}_{2i} - \text{Ext}_{1i}}{15}$$

Finite-difference method for linear problems

x_i	$w_i(h = 0.05)$	$w_i(h = 0.025)$	Ext_{1i}	Ext_{2i}	Ext_{3i}
1.0	1.00000000	1.00000000	1.00000000	1.00000000	1.00000000
1.1	1.09262207	1.09262749	1.09262925	1.09262930	1.09262930
1.2	1.18707436	1.18708222	1.18708477	1.18708484	1.18708484
1.3	1.28337094	1.28337950	1.28338230	1.28338236	1.28338236
1.4	1.38143493	1.38144319	1.38144589	1.38144595	1.38144595
1.5	1.48114959	1.48115696	1.48115937	1.48115941	1.48115942
1.6	1.58238429	1.58239042	1.58239242	1.58239246	1.58239246
1.7	1.68500770	1.68501240	1.68501393	1.68501396	1.68501396
1.8	1.78889432	1.78889748	1.78889852	1.78889853	1.78889853
1.9	1.89392740	1.89392898	1.89392950	1.89392951	1.89392951
2.0	2.00000000	2.00000000	2.00000000	2.00000000	2.00000000

The error reduces to 6.3×10^{-11} which significantly improves the case with $h = 0.1$ (about 10^{-5}).

Finite-difference method for nonlinear problems

Consider the BVP with nonlinear ODE:

$$\begin{cases} y'' = f(x, y, y'), \quad a \le x \le b \\ y(a) = \alpha, \ y(b) = \beta \end{cases}$$

Theorem

Let $D = [a, b] \times \mathbb{R} \times \mathbb{R}$. If f satisfies the following conditions:

1. *f is continuous on D,*
2. *$\exists \delta > 0$ such that $\partial_y f(x, y, y') \ge \delta$ on D,*
3. *$\exists L > 0$ such that $|\partial_y f|, |\partial_{y'} f| \le L$ on D.*

Then the BVP has a unique solution.

Finite-difference method for nonlinear problems

We apply the same partition of $[a, b]$ into $N + 1$ subintervals and centered-difference approximations for $y'(x_i)$ and $y''(x_i)$: $w_0 = \alpha$, $w_{N+1} = \beta$, and for $i = 1, \ldots, N$

$$-\frac{w_{i+1} - 2w_i + w_{i-1}}{h^2} + f\left(x_i, w_i, \frac{w_{i+1} - w_{i-1}}{2h}\right) = 0$$

This is a system of N nonlinear equations of $(w_1, \ldots, w_N)$:

$$\begin{aligned}
2w_1 - w_2 + h^2 f\left(x_1, w_1, \frac{w_2 - \alpha}{2h}\right) - \alpha &= 0 \\
-w_1 + 2w_2 - w_3 + h^2 f\left(x_2, w_2, \frac{w_3 - w_1}{2h}\right) &= 0 \\
&\vdots \\
-w_{N-2} + 2w_{N-1} - w_N + h^2 f\left(x_{N-1}, w_{N-1}, \frac{w_N - w_{N-2}}{2h}\right) &= 0 \\
-w_{N-1} + 2w_N + h^2 f\left(x_N, w_N, \frac{\beta - w_{N-1}}{2h}\right) - \beta &= 0
\end{aligned}$$

Finite-difference method for nonlinear problems

We can write the system as $F(w) = 0$ (note that $F : \mathbb{R}^N \to \mathbb{R}^N$).

To solve this system, we can apply the Newton's method:

$$w^{(k)} = w^{(k-1)} - J(w^{(k-1)})^{-1} F(w^{(k-1)})$$

starting from some initial value $w^{(0)}$. Here $J(w) \in \mathbb{R}^{N \times N}$ is the Jacobian of $F(w)$.

The key is to solve $v = J(w)^{-1} F(w)$ from $J(w)v = F(w)$ for given w.

Finite-difference method for nonlinear problems

Recall that $F(w) = (F_1(w), \ldots, F_N(w))^\top \in \mathbb{R}^N$ where

$$F_i(w) = -w_{i-1} + 2w_i - w_{i+1} + h^2 f\left(x_i, w_i, \frac{w_{i+1} - w_{i-1}}{2h}\right)$$

Jacobian $J(w) = [\frac{\partial F_i(w)}{\partial w_j}] \in \mathbb{R}^{N\times N}$ is tridiagonal:

$$J(w_1, \ldots, w_N)_{ij} = \frac{\partial F_i(w)}{\partial w_j}$$

$$= \begin{cases} -1 + \frac{h}{2} f_{y'}\left(x_i, w_i, \frac{w_{i+1}-w_{i-1}}{2h}\right), & \text{for } i = j-1 \text{ and } j = 2, \ldots, N \\ 2 + h^2 f_y\left(x_i, w_i, \frac{w_{i+1}-w_{i-1}}{2h}\right), & \text{for } i = j \text{ and } j = 1, \ldots, N \\ -1 - \frac{h}{2} f_{y'}\left(x_i, w_i, \frac{w_{i+1}-w_{i-1}}{2h}\right), & \text{for } i = j+1 \text{ and } j = 1, \ldots, N-1 \\ 0, & \text{for } |i-j| > 1 \end{cases}$$

Finite-difference method for nonlinear problems

Example (Finite-difference method for nonlinear BVP)

Solve the BVP with nonlinear ODE using finite difference method with $h = 0.1$:

$$\begin{cases} y'' = \frac{1}{8}\left(32 + 2x^3 - yy'\right), \quad 1 \le x \le 3 \\ y(1) = 17,\ y(3) = \frac{43}{3} \end{cases}$$

Finite-difference method for nonlinear problems

x_i	w_i	$y(x_i)$	$\lvert w_i - y(x_i) \rvert$
1.0	17.000000	17.000000	
1.1	15.754503	15.755455	9.520×10^{-4}
1.2	14.771740	14.773333	1.594×10^{-3}
1.3	13.995677	13.997692	2.015×10^{-3}
1.4	13.386297	13.388571	2.275×10^{-3}
1.5	12.914252	12.916667	2.414×10^{-3}
1.6	12.557538	12.560000	2.462×10^{-3}
1.7	12.299326	12.301765	2.438×10^{-3}
1.8	12.126529	12.128889	2.360×10^{-3}
1.9	12.028814	12.031053	2.239×10^{-3}
2.0	11.997915	12.000000	2.085×10^{-3}
2.1	12.027142	12.029048	1.905×10^{-3}
2.2	12.111020	12.112727	1.707×10^{-3}
2.3	12.245025	12.246522	1.497×10^{-3}
2.4	12.425388	12.426667	1.278×10^{-3}
2.5	12.648944	12.650000	1.056×10^{-3}
2.6	12.913013	12.913846	8.335×10^{-4}
2.7	13.215312	13.215926	6.142×10^{-4}
2.8	13.553885	13.554286	4.006×10^{-4}
2.9	13.927046	13.927241	1.953×10^{-4}
3.0	14.333333	14.333333	

Finite-difference method for nonlinear problems

The error order again can be improved by Richardson's extrapolation: solve the problem with $h = 0.1$, 0.05, and 0.025, and then use extrapolation as before. Accuracy can be improved from 10^{-3} to 10^{-10}.

Rayleigh-Ritz method

Idea: Convert the BVP to an integral minimization problem, and then find the minimizer from the function space spanned by a set of basis functions.

We consider a standard BVP with second-order ODE:

$$\begin{cases} -\frac{\mathrm{d}}{\mathrm{d}x}\left(p(x)\frac{\mathrm{d}y}{\mathrm{d}x}\right) + q(x)y = f, \quad 0 \le x \le 1 \\ y(0) = 0,\ y(1) = 0 \end{cases}$$

Problems with general interval $[a, b]$ and boundary conditions $y(a) = \alpha, y(b) = \beta$ can be converted into the standard one above.

For example, if $y(0) = \alpha, y(1) = \beta$, then set $z(x) = y(x) - ((1-x)\alpha + x\beta)$ and derive the ODE of z with boundary value $z(0) = z(1) = 0$.

Rayleigh-Ritz method

Theorem (Variational form of BVP)

Suppose $p \in C^1, q, f \in C$, $p \geq \delta$ *for some* $\delta > 0$ *and* $q \geq 0$ *on* $[0,1]$, *and* $y \in C_0^2$, *then* y *is the unique solution to*

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(p(x)\frac{\mathrm{d}y}{\mathrm{d}x}\right) + q(x)y = f, \quad 0 \leq x \leq 1 \quad \underline{ODE}$$

if and only if y *is the unique function that minimizes* $I[\cdot]$ *where*

$$I[u] = \int_0^1 \left(p(x)[u'(x)]^2 + q(x)[u(x)]^2 - 2f(x)u(x)\right)\mathrm{d}x \quad \underline{Energy}$$

Rayleigh-Ritz method

Proof.

1. A solution y to (ODE) satisfies:

$$\int_0^1 f(x)u(x)dx = \int_0^1 p(x)\frac{dy}{dx}(x)\frac{du}{dx}(x)+q(x)y(x)u(x)dx, \quad \forall u \in C_0^1[0,1] \quad \underline{Weak}$$

   This can be verified by multiplying u on both sides of ODE, taking integral, and integrating by part.

2. y minimizes Energy iff y satisfies Weak: For any $y, u \in C_0^1[0,1]$, define $g(\epsilon) = I[y+\epsilon u]$, then $g''(\epsilon) \geq 0$, so I is a convex functional. Therefore y minimizes Energy iff $g'(0) = 0$ for all u (i.e., y satisfies Weak).

3. Weak admits at most one solution: if y_1, y_2 both satisfies Weak, then $y = y_1 - y_2$ satisfies Weak with $f = 0$, i.e., y minimizes $J[u] = \int_0^1 (p(u')^2 + qu^2)dx$. Hence $y \equiv 0$ (since $J[u] \geq 0$ and $= 0$ only if $u \equiv 0$).

□

Rayleigh-Ritz method

Now we know BVP is equivalent to an energy minimization problem:

$$I[u] = \int_0^1 \Big(p(x)[u'(x)]^2 + q(x)[u(x)]^2 - 2f(x)u(x) \Big) \, \mathrm{d}x$$

Steps of Rayleigh-Ritz method:

1. Create a set of basis functions $\{\phi_i \mid 1 \le i \le n\}$, and set approximation $\phi = \sum_i c_i \phi_i$ to $y = \operatorname{argmin}_u I[u]$.
2. Find c by minimizing $I[\phi] = I[\sum_i c_i \phi_i]$, i.e., $\partial_{c_i} I[\sum_i c_i \phi_i] = 0$ for all i.

Rayleigh-Ritz method

Step 2 above yields a linear **normal equation** of c, denoted by $Ac = b$, where $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ with

$$a_{ij} = \int_0^1 \left[p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x) \right] dx$$

$$b_i = \int_0^1 f(x)\phi_i(x)dx$$

Once c is solved, the minimizer of I can be set to $\phi = \sum_i c_i\phi_i$.

Now the key is the design of basis functions in Step 1. If properly designed, A will be a band matrix (and even tridiagonal matrix).

Piecewise-linear basis

Steps to create a piecewise linear basis:

1. Partition $[0, 1]$ into $n + 1$ subintervals:

$$0 = x_0 < x_1 < \cdots < x_{n+1} = 1$$

   Step size $h_i = x_{i+1} - x_i$ for $i = 0, \ldots, n$.

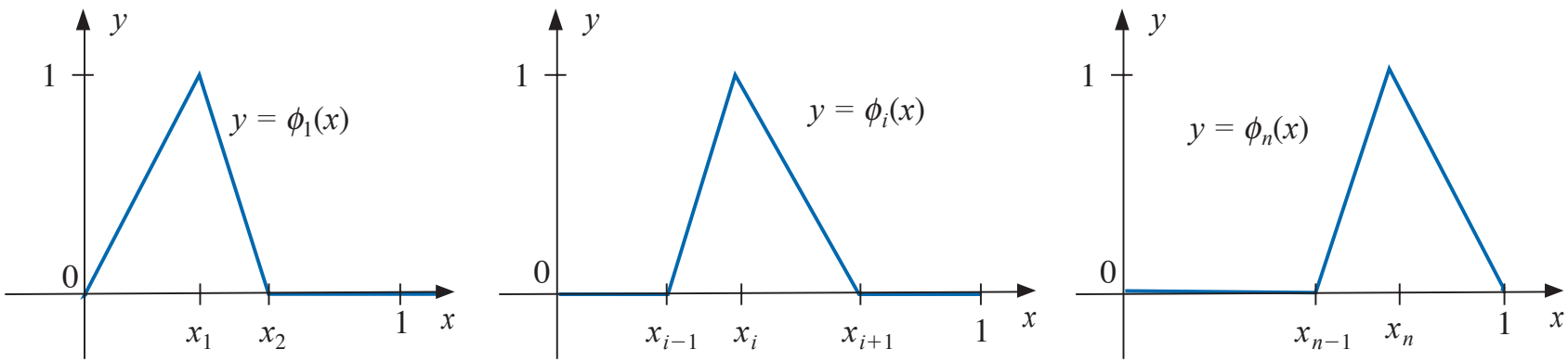
2. Set $\{\phi_i\}$ for $i = 1, \ldots, n$ by:

$$\phi_i(x) = \begin{cases} 0, & \text{if } 0 \le x \le x_{i-1} \\ \frac{1}{h_{i-1}}(x - x_{i-1}), & \text{if } x_{i-1} < x \le x_i \\ \frac{1}{h_i}(x_{i+1} - x), & \text{if } x_i < x \le x_{i+1} \\ 0, & \text{if } x_{i+1} < x \le 1 \end{cases}$$

Piecewise linear basis

Namely, $\phi_i(x)$ is 1 at $x = x_i$ and linearly decays to 0 at $x = x_{i\pm1}$, then stays as 0 outside of $[x_{i-1}, x_{i+1}]$.

Example of piecewise linear basis functions:



Piecewise linear basis

Several properties about piecewise linear basis:

1. ϕ_i is differentiable except at x_{i-1}, x_i, x_{i+1}:

$$\phi_i'(x) = \begin{cases} 0, & \text{if} \quad 0 < x < x_{i-1} \\ \frac{1}{h_{i-1}}, & \text{if} \quad x_{i-1} < x < x_i \\ -\frac{1}{h_i}, & \text{if} \quad x_i < x < x_{i+1} \\ 0, & \text{if} \quad x_{i+1} < x < 1 \end{cases}$$

2. ϕ_i and ϕ_j do not interfere if $|i-j| > 1$:

$$\phi_i(x)\phi_j(x) \equiv 0 \quad \text{and} \quad \phi_i'(x)\phi_j'(x) \equiv 0$$

Hence $A = [a_{ij}]$ in the normal equation $Ac = b$ is a tridiagonal matrix.

Piecewise linear basis

$$
\begin{aligned}
a_{ii} &= \int_0^1 \left\{ p(x) \left[\phi_i'(x)\right]^2 + q(x) \left[\phi_i(x)\right]^2 \right\} dx \\
&= \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)dx + \left(\frac{-1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} p(x)dx \\
&\quad + \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} \left(x - x_{i-1}\right)^2 q(x)dx + \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} \left(x_{i+1} - x\right)^2 q(x)dx \\
a_{i,i+1} &= \int_0^1 \left\{ p(x)\phi_i'(x)\phi_{i+1}'(x) + q(x)\phi_i(x)\phi_{i+1}(x) \right\} dx \\
&= -\left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} p(x)dx + \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} \left(x_{i+1} - x\right)\left(x - x_i\right) q(x)dx \\
a_{i,i-1} &= \int_0^1 \left\{ p(x)\phi_i'(x)\phi_{i-1}'(x) + q(x)\phi_i(x)\phi_{i-1}(x) \right\} dx \\
&= -\left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)dx + \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} \left(x_i - x\right)\left(x - x_{i-1}\right) q(x)dx \\
b_i &= \int_0^1 f(x)\phi_i(x)dx = \frac{1}{h_{i-1}} \int_{x_{i-1}}^{x_i} \left(x - x_{i-1}\right) f(x)dx + \frac{1}{h_i} \int_{x_i}^{x_{i+1}} \left(x_{i+1} - x\right) f(x)dx
\end{aligned}
$$

Piecewise linear bassis

There are $6n$ integrals to evaluate:

$$Q_{1,i} = \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)(x - x_i)\, q(x)dx, \quad \text{for each } i = 1, 2, \ldots, n-1$$

$$Q_{2,i} = \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} (x - x_{i-1})^2\, q(x)dx, \quad \text{for each } i = 1, 2, \ldots, n$$

$$Q_{3,i} = \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)^2\, q(x)dx, \quad \text{for each } i = 1, 2, \ldots, n$$

$$Q_{4,i} = \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)dx, \quad \text{for each } i = 1, 2, \ldots, n+1$$

$$Q_{5,i} = \frac{1}{h_{i-1}} \int_{x_{i-1}}^{x_i} (x - x_{i-1})\, f(x)dx, \quad \text{for each } i = 1, 2, \ldots, n$$

$$Q_{6,i} = \frac{1}{h_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - x)\, f(x)dx, \quad \text{for each } i = 1, 2, \ldots, n$$

Piecewise linear basis

Then A and b are computed as

$$
\begin{aligned}
a_{i,i} &= Q_{4,i} + Q_{4,i+1} + Q_{2,i} + Q_{3,i}, \quad \text{for each } i = 1, 2, \ldots, n \\
a_{i,i+1} &= -Q_{4,i+1} + Q_{1,i}, \quad \text{for each } i = 1, 2, \ldots, n-1 \\
a_{i,i-1} &= -Q_{4,i} + Q_{1,i-1}, \quad \text{for each } i = 2, 3, \ldots, n \\
b_i &= Q_{5,i} + Q_{6,i}, \quad \text{for each } i = 1, 2, \ldots, n
\end{aligned}
$$

We can show that A is positive definite.

Piecewise linear basis

Two ways to approximate the $6n$ integrals Q's:

1. Quadratures such as Simpson's rule.
2. Approximate p, q, r by piecewise linear functions and compute integrals. For example, $p(x) \approx \sum_i p(x_i)\phi_i(x)$ etc., then

$$
\begin{aligned}
Q_{1,i} &\approx \frac{h_i}{12}[q(x_i) + q(x_{i+1})] \\
Q_{2,i} &\approx \frac{h_{i-1}}{12}\left[3q\left(x_i\right) + q\left(x_{i-1}\right)\right], \\
Q_{3,i} &\approx \frac{h_i}{12}\left[3q\left(x_i\right) + q\left(x_{i+1}\right)\right] \\
Q_{4,i} &\approx \frac{h_{i-1}}{2}\left[p\left(x_i\right) + p\left(x_{i-1}\right)\right] \\
Q_{5,i} &\approx \frac{h_{i-1}}{6}\left[2f\left(x_i\right) + f\left(x_{i-1}\right)\right] \\
Q_{6,i} &\approx \frac{h_i}{6}\left[2f\left(x_i\right) + f\left(x_{i+1}\right)\right]
\end{aligned}
$$

Each approximation has error order $O(h_i^3)$.

Piecewise linear basis

Example (Rayleigh-Ritz method with piecewise linear basis)

Solve the BVP below using Rayleigh-Ritz method and piecewise linear basis with $h_i = h = 0.1$:

$$-y'' + \pi^2 y = 2\pi^2 \sin(\pi x), \quad 0 \le x \le 1, \quad y(0) = 0,\ y(1) = 0$$

Solution: We have $p(x) \equiv 1$, $q(x) \equiv \pi^2$, $f(x) = 2\pi^2 \sin(\pi x)$. Then apply the formula above to obtain $Q_{1,i}, \ldots, Q_{6,i}$ for $i = 0, \ldots, 9$, and then A and b. Then solve c from $Ac = b$, and obtain $\phi(x) = \sum_i c_i \phi_i(x)$ (note that $\phi(x)$ is piecewise linear function and $\phi(x_i) = c_i$).

Piecewise linear basis

i	x_i	$\phi(x_i)$	$y(x_i)$	$\lvert\phi(x_i) - y(x_i)\rvert$
1	0.1	0.3102866742	0.3090169943	0.00127
2	0.2	0.5902003271	0.5877852522	0.00241
3	0.3	0.8123410598	0.8090169943	0.00332
4	0.4	0.9549641896	0.9510565162	0.00390
5	0.5	1.0041087710	1.0000000000	0.00411
6	0.6	0.9549641893	0.9510565162	0.00390
7	0.7	0.8123410598	0.8090169943	0.00332
8	0.8	0.5902003271	0.5877852522	0.00241
9	0.9	0.3102866742	0.3090169943	0.00127

The error order is $O(h^2)$ due to the nature of linear (first-order) approximation of the integand.

B-spline basis

We can create C^2 basis functions using the idea of cubic splines. These are called the **B-splines** (basis splines).

We start from the cubic spline function S:

$$S(x) = \begin{cases} 0, & \text{if } x \leq -2 \\ \frac{1}{4}(2+x)^3, & \text{if } -2 \leq x \leq -1 \\ \frac{1}{4}\left[(2+x)^3 - 4(1+x)^3\right], & \text{if } -1 < x \leq 0 \\ \frac{1}{4}\left[(2-x)^3 - 4(1-x)^3\right], & \text{if } 0 < x \leq 1 \\ \frac{1}{4}(2-x)^3, & \text{if } 1 < x \leq 2 \\ 0, & \text{if } 2 < x \end{cases}$$

B-spline basis

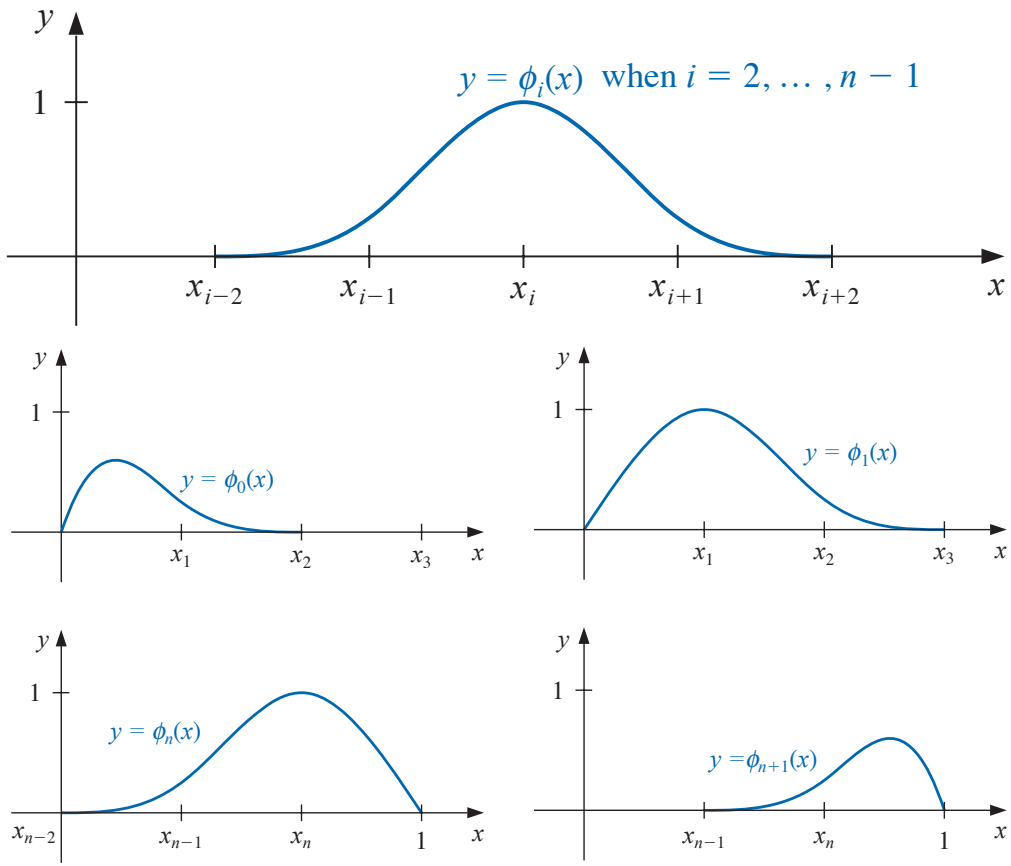
Then construct B-spline basis functions $\{\phi_i \mid 0 \le i \le n+1\}$:

$$\phi_i(x) = \begin{cases} S\left(\frac{x}{h}\right) - 4S\left(\frac{x+h}{h}\right), & \text{if } i = 0 \\ S\left(\frac{x-h}{h}\right) - S\left(\frac{x+h}{h}\right), & \text{if } i = 1 \\ S\left(\frac{x-ih}{h}\right), & \text{if } 2 \le i \le n-1 \\ S\left(\frac{x-nh}{h}\right) - S\left(\frac{x-(n+2)h}{h}\right), & \text{if } i = n \\ S\left(\frac{x-(n+1)h}{h}\right) - 4S\left(\frac{x-(n+2)h}{h}\right), & \text{if } i = n+1 \end{cases}$$

- $\phi_i \in C_0^2[0,1]$.
- $\{\phi_i\}$ are independent.

B-spline basis

$\phi_i(x)$ for $2 \leq i \leq n-1$ (top) and $\phi_0, \phi_1, \phi_n, \phi_{n+1}$ (bottom four).



B-spline basis

Let $\phi(x) = \sum_i c_i \phi_i(x)$. Then the normal equation $\partial_c I[\phi] = 0$ is $Ac = b$ where $A = [a_{ij}]$ is a positive definite band matrix with bandwidth ≤ 7, where

$$a_{ij} = \int_0^1 \left\{ p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x) \right\} dx$$

$$b_i = \int_0^1 f(x)\phi_i(x)dx$$

To compute these integrals, we can replace p, q, f by their cubic spline interpolations (so on each subinterval they are cubic polynomials), and integrals can be evaluated exactly (as the integrands are polynomials).

B-spline basis

Example (Rayleigh-Ritz with B-spline basis)

Solve the BVP below using Rayleigh-Ritz method and B-spline basis with $h_i = h = 0.1$:

$$-y'' + \pi^2 y = 2\pi^2 \sin(\pi x), \quad 0 \le x \le 1, \quad y(0) = 0,\ y(1) = 0$$

Solution: We have $p(x) \equiv 1$, $q(x) \equiv \pi^2$, $f(x) = 2\pi^2 \sin(\pi x)$. Then approximate $Q_{1,i}, \ldots, Q_{6,i}$ for $i = 0, \ldots, 9$, and then A and b. Then solve c from $Ac = b$, and obtain $\phi(x) = \sum_i c_i \phi_i(x)$.

B-spline basis

Numerical result:

i	c_i	x_i	$\phi(x_i)$	$y(x_i)$	$\lvert y(x_i) - \phi(x_i) \rvert$
0	$0.50964361 \times 10^{-5}$	0	0.00000000	0.00000000	0.00000000
1	0.20942608	0.1	0.30901644	0.30901699	0.00000055
2	0.39835678	0.2	0.58778549	0.58778525	0.00000024
3	0.54828946	0.3	0.80901687	0.80901699	0.00000012
4	0.64455358	0.4	0.95105667	0.95105652	0.00000015
5	0.67772340	0.5	1.00000002	1.00000000	0.00000020
6	0.64455370	0.6	0.95105130	0.95005520	0.00000061
7	0.54828951	0.7	0.80901773	0.80901699	0.00000074
8	0.39835730	0.8	0.58778690	0.58778525	0.00000165
9	0.20942593	0.9	0.30901810	0.30901699	0.00000111
10	$0.74931285 \times 10^{-5}$	1.0	0.00000000	0.00000000	0.00000000

This is much more accurate than the one with piecewise linear basis.