CS 770 Numerical Analysis

Instructor: Hans De Sterck
Typesetting: Egor Larionov

University of Waterloo

December 12, 2013

Contents

Abstract

Introduction to basic algorithms and techniques for numerical computing. Error analysis, interpolation (including splines), numerical differentiation and integration, numerical linear algebra (including methods for linear systems, eigenvalue problems, and the singular value decomposition), root finding for nonlinear equations and systems, numerical ordinary differential equations, and approximation methods (including least squares, orthogonal polynomials, and Fourier transforms).

Lecture 1

1 Computer arithmetic and error analysis

1.1 Floating point number system [1, 2.4]

Example 1.1. Floating point number system

$$A = (\beta = 10, t = 3, L = -99, U = 99)$$

$x = 7.304 \cdot 10^4$ is in this number system. In general:

$$x = m\beta^e$$

where

- $\beta = 10$ is the base (decimal base),
- $m = 7.304$ is the mantissa,
- $f = 304$ is the fraction,
- $t = 3$ digits in the fraction,
- $e = 4$ is the exponent,
- $L \le e \le U$ are the bounds.

Example 1.2. $B = (\beta = 2, t = 2, L = -10, U = 10)$

$$\begin{aligned} x = 1.01 \cdot 2^2 &= 101 \text{ (binary)} \\ &= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 5 \text{ in decimal} \end{aligned}$$

1.1.1 Rules for the "normalized" mantissa

$m = \pm d_0.d_1d_2\ldots d_t = (-1)^s d_0.f$ where s is the sign bit (0 or 1) with

$$\begin{aligned} 1 \le |m| < \beta \\ 1 \le d_0 \le \beta - 1 \\ \text{and } 0 \le d_i \le \beta - 1 \qquad (i = 1, \ldots, t) \end{aligned}$$

for example in system A,

$$x = 1.000 \cdot 10^6$$

is valid, but

$$x = 0.123 \cdot 10^3$$

is not and must be normalized to

$$x = 1.230 \cdot 10^2.$$

In computers, the floating point (f.p.) systems normally used are:

1. single precision: (4 bytes, 32 bits)

   $$(\beta = 2, t = 23, L = -126, U = 127)$$

   where $x = (-1)^s d_0.f \cdot 2^e$ is valid.

s	f	E
1 bit	23 bits	8 bits

   Note that $E = e+127$. Also $d_0 = 1$ always for normalized numbers (in binary) $\implies$ no need to store it.

   Smallest positive (non-zero) number:

   $$1.0\ldots0 \cdot 2^{-126} \approx 1.2 \cdot 10^{-38}.$$

   Largest positive number:

   $$1.1\ldots1 \cdot 2^{127} = (2 - 2^{-23})2^{127} \approx 2^{128} \approx 3.4 \cdot 10^{38}.$$

2. double precision: (8 bytes, 64 bits)

   $$(\beta = 2, t = 52, L = -1022, U = 1023)$$

   Smallest positive (non-zero) number:

   $$2^{-1022} \approx 2.2 \cdot 10^{-308}$$

   Largest positive number:

   $$(2 - 2^{-52})2^{1023} \approx 2^{1024} \approx 1.8 \cdot 10^{308}$$

Note the greater range and greater relative accuracy. Note: [1, 2.8]"IEEE Standard" for floating point numbers has some additional subtle points. Single precision:

$$\begin{aligned} 1 \le E \le 254 &: x = (-1)^s(1.f)2^{E-127} \text{ (normalized numbers)} \\ E = 255 &: f \ne 0 \implies x = \text{NaN (not a number, e.g. } 0/0) \\ &\quad f = 0 \implies x = (-1)^s\text{Inf (infinity, e.g. } 1/0) \\ E = 0 &: f = 0 \implies x = 0 \\ &\quad f \ne 0 \implies \text{(denormalized numbers)} \\ &\qquad \text{e.g. } x = 0.0001760\ldots0 \cdot 2^{-126} \end{aligned}$$

Lecture 2

1.2 Rounding in f.p. systems [1, 2.5]

$$A = (\beta = 10, t = 4, L = -10, U = 10)$$

Consider exact $X = 11.3486$. Represent X in system A by "rounding to nearest" rounded, normalized representation of X: $x = 1.1349 \cdot 10^1$.

$$\begin{aligned} \text{ex. } x = 11.3484 &\rightarrow x = 1.1348 \cdot 10^1 \\ \text{ex. } x = 11.3485 &\rightarrow x = 1.1348 \cdot 10^1 \\ \text{ex. } x = 11.3475 &\rightarrow x = 1.1348 \cdot 10^1 \end{aligned}$$

here we followed a rule: "round to nearest, tie to even" (default in IEEE).

Note: after rounding in f.p. system A, we say that x has $t+1$ "correctly rounded digits". Consider f.p. system

$$\begin{aligned} &(\beta, t, L, U) \\ \text{exact: } &X := M\beta^e \qquad (1 \le |M| < \beta) \\ \text{rounded: } &x := m\beta^e \qquad (t+1 \text{ (correct) digits in } m) \end{aligned}$$

Definition 1.3. Define *absolute error*:

$$\Delta X := x - X,$$

where x is the approximation and X is exact.

Define *relative error*:

$$\delta X := \frac{x - X}{X} = \frac{\Delta X}{X}$$

where $X \neq 0$.

Example 1.4. $A = (\beta = 10, t = 3, L, U)$

$$\begin{aligned} X &= 4.732896 \cdot 10^6 \\ x &= 4.733 \cdot 10^6 \end{aligned}$$

$|m - M| \le 0.5 \cdot 10^{-3}$ after rounding.

In general, after rounding we have

$$|m - M| \le \frac{1}{2}\beta^{-t},$$

which implies

$$\begin{aligned} |x - X| &\le \frac{1}{2}\beta^{-t}\beta^e \\ \text{and so } \left|\frac{x - X}{X}\right| &\le \frac{\frac{1}{2}\beta^{-t}\beta^e}{|M|\beta^e} \le \frac{1}{2}\beta^{-t} \end{aligned}$$

Definition 1.5. Define *unit roundoff* to be

$$\mu := \frac{1}{2}\beta^{-t}$$

So for rounding we have that

$$|\delta X| = \left|\frac{x - X}{X}\right| \le \frac{1}{2}\beta^{-t} \implies |\delta X| \le \mu.$$

Note: single precision: $t = 23$,

$$\mu = 2^{-24} \approx 6.0 \cdot 10^{-8} \approx 0.5 \cdot 10^{-7},$$

which is roughly equivalent to

$$(\beta = 10, t = 7) \rightarrow \mu = 0.5 \cdot 10^{-7}$$

"approximately $t + 1 = 8$ decimal digits can be represented correctly in single precision binary". double precision: $t = 52$

$$\mu = 2^{-53} \approx 1.1 \cdot 10^{-16},$$

which is roughly equivalent to $(\beta = 10, t = 16)$. Approximately $t + 1 = 17$ decimal digits can be represented correctly in double precision.

Note: in MATLAB, we have:

- `format long` with 16 or 17 digits
- `format short` with 7 or 8 digits

Note: there is also a *fixed point* number system.

Example 1.6.

$$x = \pm d_{-1}d_0.d_1 \text{ (no exponent)},$$

where $0 \le d_i \le 9$ for $i \in \{-1, 0, 1\}$ (decimal base), and $t = 3$ total digits, with range -99.9 to 99.9. Now consider the accuracy after rounding:

$$|x - X| \le \frac{1}{2}\beta^{-1} = \frac{1}{2}10^{-1} = 0.05,$$

so absolute error is bound by unit round off.

Note: that the *fixed point* system has a fixed distance between representable numbers while in the *floating point* system, there are more representable numbers closer to zero, and less farther from zero. Recall that in the floating point system we have

$$\begin{aligned} |\delta X| &= \left|\frac{x - X}{X}\right| \le \mu \\ \text{or } &|x - X| \le |X|\mu \text{ where } |X|\mu \text{ is small if } X \text{ is small.} \end{aligned}$$

1.2.1 Representation of rounding effect

Consider X exact, x rounded in the floating point system. We know

$$\left|\frac{x - X}{X}\right| \le \mu.$$

We also write:

$$x = fl(X) = X(1 + \varepsilon) \text{ with } |\varepsilon| \le \mu$$

1.3 Rounding in basic arithmetic operations [1, 2.6]

It is possible to implement $x+y$ (x and y in computer representation) on a computer CPU s.t.

$$\left|\frac{fl(x+y)-(x+y)}{x+y}\right| \le \mu$$

$$\text{or } \; fl(x+y) = (x+y)(1+\varepsilon) \text{ with } |\varepsilon| \le \mu.$$

the same holds for xy, $x-y$ and x/y

1.4 Loss of information / accuracy [1, 2.3]

In some f.p. operations, information / accuracy is lost, sometimes this is unavoidable, but sometimes, algorithms can be reformulated to avoid these 'catastrophic' steps.

Example 1.7. Loss of information:
Consider a, b, and c in

$$(\beta = 10, t = 3, L, U), \qquad \mu = \frac{1}{2}\beta^{-t} = \frac{1}{2}10^{-3}$$

$$\begin{aligned} a &= 9.876 \cdot 10^4 = 98760 \\ b &= -9.880 \cdot 10^4 = -98800 \\ c &= 3.456 \cdot 10^1 = 34.56 \end{aligned}$$

Consider $a+c$:

$$fl(a+c) = fl(9879\underbrace{4.56}_{\text{lost}}) = 9.879 \cdot 10^4$$

information is lost, which in this case is unavoidable, but not a big deal. Relative error:

$$\left|\frac{a+c-fl(a+c)}{a+c}\right| \approx \frac{1}{2}10^{-4} \le \mu$$

Now consider $a+b+c$:

(1) $(a+b)+c$:

$$\begin{aligned} fl(fl(a+b)+c) &= fl(-4 \cdot 10^1 + 3.456 \cdot 10^1) \\ &= -0.544 \cdot 10^1 \\ &= a+b+c \text{ (exact)} \end{aligned}$$

(2) $(a+c)+b$:

$$\begin{aligned} fl(fl(a+c)+b) &= fl(9.879 \cdot 10^4 - 9.880 \cdot 10^4) \\ &= -1.0 \cdot 10^1 \text{ (only 1 correct digit)} \end{aligned}$$

relative error:

$$\left|\frac{a+b+c-fl(fl(a+c)+b)}{a+b+c}\right| \approx 0.84$$

84% relative error due to loss of information and cancellation.

So the order of operations may matter, and subtracting two almost equal numbers can result in large error.

Example 1.8. Loss of information via catastrophic cancellation:

We introduce new notation for the general context of error propagation: rounding error, measurement error, discretization error, and accumulation of rounding error.

$$\begin{aligned} \text{exact: } & x \\ \text{approximate: } & \bar{x} \\ \text{absolute error: } & \Delta x = \bar{x} - x \end{aligned}$$

Notation 1.9. We say $\bar{x}$ approximates x with

$$|\bar{x} - x| = |\Delta x| \le \nu \iff x = \bar{x} \pm \nu.$$

1.4.1 Catastrophic cancellation

If you subtract two almost equal numbers, the relative error in the result may be very large.

$$\begin{aligned} y &= x_1 - x_2 \\ \bar{y} &= \bar{x}_1 - \bar{x}_2 \\ \Delta y = \bar{y} - y &= (\bar{x}_1 - \bar{x}_2) - (x_1 - x_2) = \Delta x_1 - \Delta x_2 \end{aligned}$$

Note that Δx_1 is not known, we only know that $|\Delta x_1| \le \nu$.

$$\begin{aligned} |\Delta y| &\le |\Delta x_1| + |\Delta x_2| \\ |\delta y| &\le \frac{|\Delta x_1| + |\Delta x_2|}{|x_1 - x_2|} \end{aligned}$$

Note that $|x_1 - x_2|$ is small if $x_1 \approx x_2$ and is smaller than $|x_1|$ or $|x_2|$. For example:

$$\begin{aligned} x_1 &= 10.123456 \pm 0.5 \cdot 10^{-6} \\ x_2 &= 10.123788 \pm 0.5 \cdot 10^{-6} \\ |\delta x_2| &= \frac{|\Delta x_2|}{|x_2|} \le 0.5 \cdot 10^{-7} \\ y &= x_1 - x_2 = -0.000332 \pm 10^{-6} \text{ (only 3 digits left)} \\ |\delta y| &= \frac{|\Delta y|}{|y|} \le \frac{0.5 \cdot 10^{-6} + 0.5 \cdot 10^{-6}}{0.000332} = 10^{-3} \gg 0.5 \cdot 10^{-6} \end{aligned}$$

Example 1.10. Sometimes catastrophic cancellation can be avoided. Consider the equation $x^2 - 18x + 1 = 0$. With roots $x_1 = 17.944271\ldots$ and $x_2 = 0.0557280\ldots$.

We use $(\beta = 10, t = 3, L, U)$.

Consider algorithm (1):

$$\begin{aligned} x_1 &= 9 + \sqrt{80} \\ \bar{x}_1 &= fl(9 + \sqrt{80}) = fl(9 + 8.944) = 17.94 \\ |\delta x_1| &\approx 0.24 \cdot 10^{-3} \end{aligned}$$

And similarly

$$\begin{aligned} x_2 &= 9 - \sqrt{80} \\ \bar{x}_2 &= fl(9 - \sqrt{80}) = fl(9 - 8.944) = 0.056 \\ |\delta x_2| &\approx 0.5 \cdot 10^{-2} \end{aligned}$$

As we can see $\bar{x}_2$ has only 2 correct digits.

Algorithm (2):

$$x_2 = \frac{1}{x_1}$$
$$\bar{x}_2 = fl(1/x_1) = 0.05574$$
$$|\delta x_2| \approx 0.21 \cdot 10^{-3} \leq \mu$$

Where we have 4 correct digits.

Lecture 3

1.5 Error propagation [1, 2.3]

(1) Take $y = f(x)$ and let $x = \bar{x} \pm \varepsilon$, with $|\bar{x} - x| = |\Delta x| \leq \varepsilon$.

For example, $\bar{x}$ is obtained from x by rounding in floating point systems $\implies \varepsilon = \mu|x|$.

How does the error in x propagate to an error in y?

What can we say about

$$|\Delta y| = |y(\bar{x}) - y(x)| = |\Delta f| = |f(\bar{x}) - f(x)|?$$

Assume $f(x)$ is differentiable. We use the Mean Value Theorem (MVT):

$$\exists \xi \in (x, \bar{x}) : f'(\xi) = \frac{f(\bar{x}) - f(x)}{\bar{x} - x}$$

or equivalently, $\exists \xi \in (x, \bar{x}) : \Delta f = f'(\xi)\Delta x$, so

$$|\Delta f| = |f'(\xi)||\Delta x|$$

or, approximately

$$|\Delta f| \approx |f'(x)||\Delta x|$$
$$\text{or } |\Delta f| \approx |f'(\bar{x})||\Delta x|$$

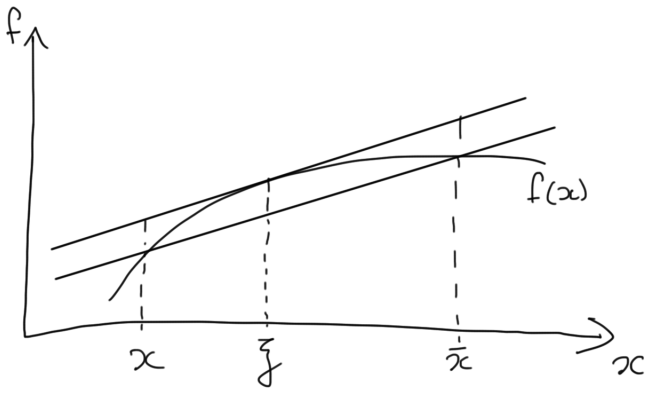


Figure 1.1: Mean Value Theorem.

(2) Take $y = f(x_1, x_2)$, we consider two approaches:

(a) We use definition and inequalities, For example:

$$y = x_1 + x_2 \qquad y = \bar{y} - \Delta y$$
$$x_1 = \bar{x}_1 - \Delta x_1$$
$$x_2 = \bar{x}_2 - \Delta x_2$$

$$\Delta y = \bar{y} - y = \bar{x}_1 + \bar{x}_2 - (x_1 + x_2) = \Delta x_1 + \Delta x_2$$
$$\implies \Delta y = \Delta x_1 + \Delta x_2$$
$$\implies |\Delta y| \leq |\Delta x_1| + |\Delta x_2|$$

Another example:

$$y = x_1 x_2$$
$$\Delta y = \bar{x}_1\bar{x}_2 - (x_1 x_2) = (x_1 + \Delta x_1)(x_2 + \Delta x_2) - x_1 x_2$$
$$= x_1\Delta x_2 + x_2\Delta x_1 + \Delta x_1 \Delta x_2$$
$$\frac{\Delta y}{y} = \frac{\Delta x_1}{x_1} + \frac{\Delta x_2}{x_2} + \frac{\Delta x_1 \Delta x_2}{x_1 x_2}$$

assume $\left|\frac{\Delta x_1}{x_1}\right| \ll 1$ and $\left|\frac{\Delta x_2}{x_2}\right| \ll 1$, then

$$\frac{\Delta y}{y} \approx \frac{\Delta x_1}{x_1} + \frac{\Delta x_2}{x_2}$$

also:

$$\left|\frac{\Delta y}{y}\right| \lesssim \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right|.$$

(b) Use multivariate MVT:

Theorem 1.11. *Let $f(\vec{x})$ be differentiable, where $\vec{x} \in \mathbb{R}^n$. Then $\exists \theta \in (0,1)$ such that*

$$f(\vec{\bar{x}}) - f(\vec{x}) = \nabla f(\vec{x} + \theta\Delta\vec{x}) \cdot \Delta\vec{x}$$
$$= \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(\vec{x} + \theta\Delta\vec{x})\Delta x_k$$

Note that $\Delta\vec{x} = \vec{\bar{x}} - \vec{x}$. We denote $\vec{x}^* := \vec{x} + \theta\Delta\vec{x}$.

Proof. Note: $f(\vec{x}+t\Delta\vec{x})$ is a single-variable function in t. Let $F(t) = f(\vec{x} + t\Delta\vec{x})$. Then $\exists\theta \in (0,1)$ s.t.

$$\frac{F(1) - F(0)}{1 - 0} = \left.\frac{dF(t)}{dt}\right|_{t=\theta} \text{ (by MVT).}$$

$$\text{or } f(\vec{\bar{x}}) - f(\vec{x}) = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(\vec{x} + \theta\Delta\vec{x})\Delta x_k$$

□

Therefore:

$$\Delta f \approx \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(\vec{\bar{x}})\Delta x_k \quad \text{or} \quad \Delta f \approx \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(\vec{x})\Delta x_k$$

and so

$$|\Delta f| \lesssim \sum_{k=1}^{n} \left|\frac{\partial f}{\partial x_k}(\vec{x})\right| |\Delta x_k|$$
$$\text{or } |\Delta f| \lesssim \sum_{k=1}^{n} \left|\frac{\partial f}{\partial x_k}(\vec{\bar{x}})\right| |\Delta x_k|$$

For example: $y = f(x_1, x_2) = x_1 x_2$.

$$\frac{\partial f}{\partial x_1} = x_2 \qquad \Delta f \approx x_2\Delta x_1 + x_1\Delta x_2$$
$$\frac{\partial f}{\partial x_2} = x_1 \qquad \text{and } \frac{\Delta f}{f} \approx \frac{\Delta x_1}{x_1} + \frac{\Delta x_1}{x_2}$$

2 Root finding methods

2.1 Problem description [1, 4.1]

Problem: find the root x^*, such that $f(x^*) = 0$, where f is a non-linear function.

For example: $f(x) = x - \cos(x) = 0 \implies x^* \approx 0.74$. Often no closed-form solution exists so iteration will be necessary.

Definition 2.1. "x^* is a root of $f(x)$ with multiplicity q" means $f(x) = (x-x^*)^q g(x)$ where $g(x^*) \neq 0$ and $|g(x^*)| < \infty$.

Example 2.2.

$$f(x) = (x-3)^2 \underbrace{\sin(x)}_{g} \qquad x = 3 \text{ is a double root.}$$

$$= (x-3)^3 \underbrace{\frac{\sin(x)}{x-3}}_{g} \qquad \text{not a triple root.}$$

$$= (x-3) \underbrace{\sin(x)(x-3)}_{g} \qquad \text{not a single root.}$$

Note: $f'(x) = q(x-x^*)^{q-1}g(x) + (x-x^*)^q g'(x)$

$q > 1 \implies f'(x^*) = 0$ assiming $|g'(x^*)| < \infty$

e.g. $q = 2$: double root $f(x^*) = 0,\ f'(x^*) = 0$

$q > 2 \implies f'(x^*) = 0$ and $f''(x^*) = 0$

e.g. $q = 3$: triple root $f(x^*) = 0,\ f'(x^*) = 0,\ f''(x^*) = 0$

2.2 Four root finding methods [1, 4.2, 4.3]

2.2.1 Bisection

Assume $f(x)$ is continuous and let $f(a)f(b) < 0$.
Then $\exists c \in (a,b)$ s.t. $f(c) = 0$ (IVT).

```
d = (a+b)/2  # bisection
if f(d) == 0:
  success
else
  if f(a)f(d) < 0:
    b = d
  else
    a = d
  end
end
repeat until |b-a| < tol  # some input tolerance
```

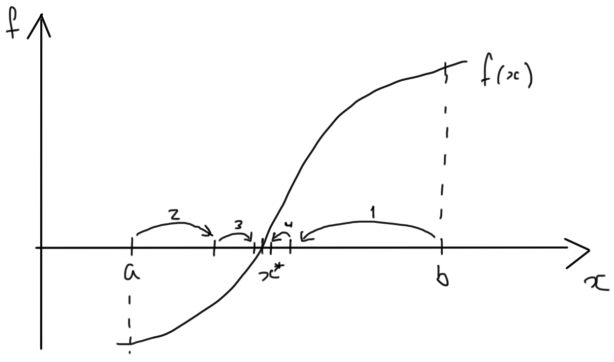


Figure 2.1: Bisection method.

Note: this method is guaranteed to find a root (also if there are multiple roots between a and b).

2.2.2 Newton-Raphson method

Have $f(x^*) = 0$, and assume $f(x)$ is differentiable. Then $0 = f(x^*) \approx f(x_0) + f'(x_0)(x^* - x_0)$ (truncated Taylor series). Define x_1 s.t. $f(x_0) + f'(x_0)(x_1 - x_0) = 0$. Then in general

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0$$

$$\text{or} \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (\text{assume } f'(x_k) \neq 0)$$

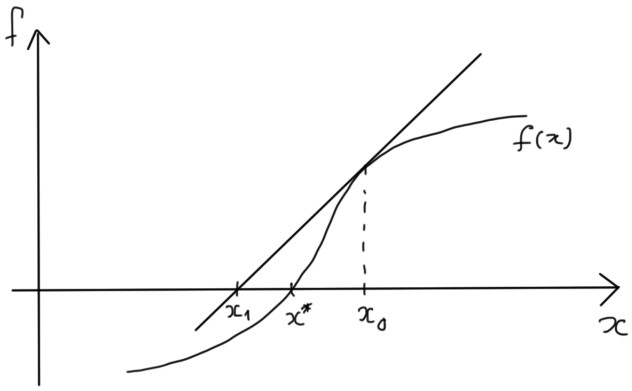


Figure 2.2: Newton-Raphson method.

Note: NR is an example of a fixed-point method:

$$x_{k+1} = \varphi(x_k) \text{ with } \varphi(x) = x - \frac{f(x)}{f'(x)} \text{ (NR)}$$

(fixed point: $x^* = \varphi(x^*)$)

2.2.3 General fixed-point methods

Rewrite $f(x) = 0$ as $x = \varphi(x)$. Then use iteration:

$$x_{k+1} = \varphi(x_k).$$

Example 2.3. $f(x) = x - \cos(x) = 0$.

(A) Try $x = \cos(x)$ or $x = \varphi(x)$ with $\varphi(x) = \cos(x)$. More generally, given $f(x) = 0$, we may consider

$$\varphi(x) = x + f(x) \implies x + f(x) = x$$
$$\text{or } \varphi(x) = x - f(x) \implies x - f(x) = x$$
$$\text{or } \varphi(x) = x + f(x)^2 \implies x + f(x)^2 = x$$

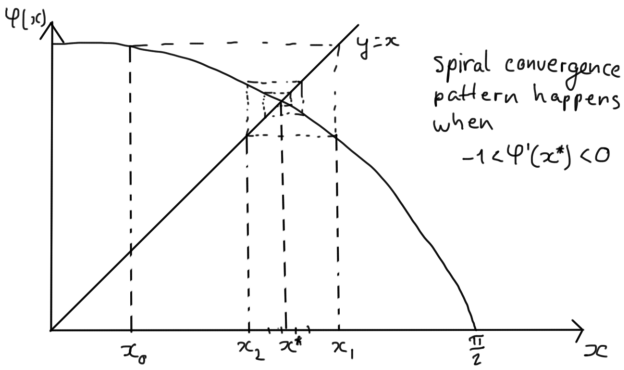


Figure 2.3: Spiral *convergence* happens when $-1 < \varphi'(x^*) < 0$

(B) Try $\arccos(x) = x$. Then $x_{k+1} = \varphi(x_k)$ with $\varphi(x) = \arccos(x)$.

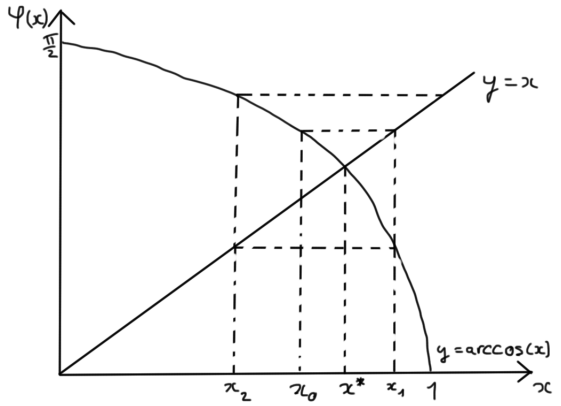


Figure 2.4: Spiral *divergence* happens when $\varphi'(x^*) < -1$

Lecture 4

In the general fixed point method, there are 4 cases:

(A) Spiral convergence: $-1 < \varphi'(x^*) < 0$.

(B) Spiral divergence: $\varphi'(x^*) < -1$.

(C) Staircase convergence: $0 < \varphi'(x^*) < 1$.

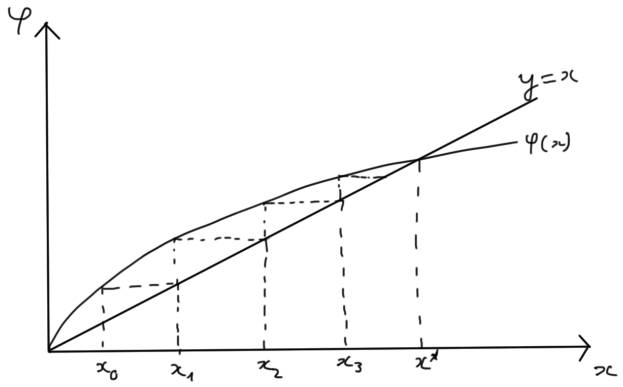


Figure 2.5: Staircase convergence.

(D) Staircase divergence: $1 < \varphi'(x^*)$.

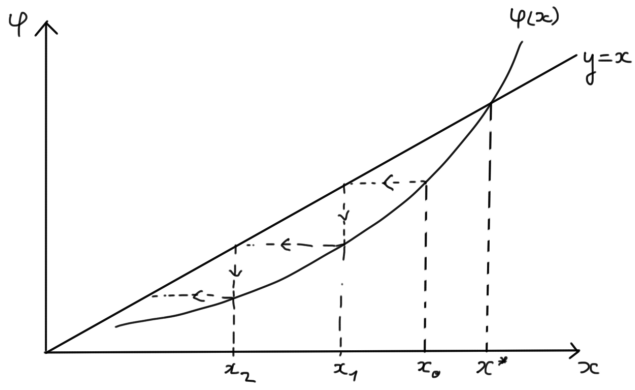


Figure 2.6: Staircase divergence.

Note: convergence also depends on the initial guess.

2.2.4 Secant method

Recall that in NR we have $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$. Approximate $f'(x_k)$ by a finite difference:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}},$$

which gives

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})},$$

where $f(x_k) \neq f(x_{k-1})$.

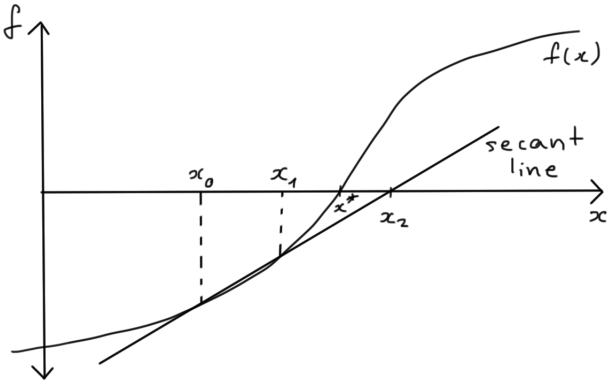


Figure 2.7: Secant method.

Comparison to NR:

1) Secant doesn't need the derivative.

2) Secant converges more slowly than NR.

3) Both may diverge, depending on initial guess and the function $f(x)$.

4) Secant needs two initial guesses as opposed to only one.

2.3 Convergence (of fixed point methods) [1, 4.4]

Have $x_{k+1} = \varphi(x_k)$, and assume that φ is differentiable.

Theorem 2.4 (Convergence for fixed-point methods)**.** *Let $x^* = \varphi(x^*)$. Assume $\exists \delta > 0$, $\exists m \in [0, 1)$ s.t.*

$$|\varphi'(x)| \leq m, \quad \forall x \in I,$$

where $I = \{x : |x - x^*| \leq \delta\}$ (closed interval), (φ is a contraction mapping). Assume $x_0 \in I$, then

1. *$x_k \in I$, $\forall k \in \{1, 2, 3, \dots\}$.*
2. *$\lim_{k\to\infty} x_k = x^*$ (convergence to a solution).*
3. *x^* is the only solution in I (of $x = \varphi(x)$) (uniqueness of the solution)*

Proof. We prove each part separately:

1. Proof by induction. Assume $x_{k-1} \in I$. Then

$$x_k - x^* = \varphi(x_{k-1}) - \varphi(x^*) = \varphi'(\xi_k)(x_{k-1} - x^*).$$

By MVT, $\exists \xi_k \in (x_{k-1}, x^*)$, and thus $\xi_k \in I$, and $|\varphi'(\xi_k)| \le m < 1$. Therefore

$$|x_k - x^*| \le m|x_{k-1} - x^*| \le m\delta < \delta,$$

since $x_{k-1} \in I$. Now since $x_0 \in I$, we have that $x_k \in I$, $\forall k \in \{1, 2, 3, \dots\}$.

2. Note that $|x_k - x^*| \le m^k |x_0 - x^*|$, so we have

$$\lim_{k\to\infty} |x_k - x^*| = 0, \quad \text{since } m < 1.$$

3. Assume there is a second root $\hat{x} \in I$, s.t. $\hat{x} \ne x^*$. Then

$$|\hat{x} - x^*| = |\varphi(\hat{x}) - \varphi(x^*)| = |\varphi'(\xi)||\hat{x} - x^*|$$

with $\xi \in (\hat{x}, x^*)$, so $\xi \in I$, or $1 = |\varphi'(\xi)|$, which is a contradiction. Thus x^* is the unique solution.

□

Note that intuitively, a contraction mapping "makes intervals smaller".

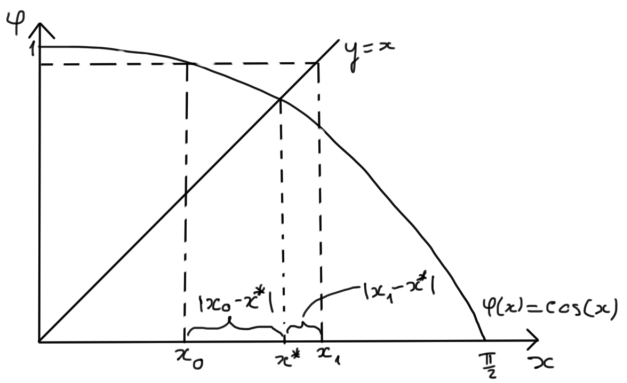


Figure 2.8: $|x_1 - x^*| = |\varphi(x_0) - \varphi(x^*)| < |x_0 - x^*|$

Note:
$\varphi(x) = \cos(x) \implies |\varphi'(0.74\dots)| < 1$
$\varphi(x) = \arccos(x) \implies |\varphi'(0.74\dots)| > 1$
Note: If there are two solutions in I, then $|\varphi'(x)| = 1$ for some $x \in I$.

2.3.1 Convergence speed

Definition 2.5 (Convergence of a sequence with order p)**.** Let $\{x_k\}_{k=0}^{\infty}$ be a sequence that converges to x^*. Then the order of convergence, p, is the largest number $p \ge 1$ s.t.

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} = c, \text{ with } 0 < c < \infty.$$

c is called the asymptotic error constant.

1) If $p = 1$, and $0 < c < 1$, then we have *linear convergence.*

$$|x_{k+1} - x^*| \approx c|x_k - x^*|$$

2) If $p = 2$ and $0 < c < \infty$, then we have *quadratic convergence.*

$$|x_{k+1} - x^*| \approx c|x_k - x^*|^2$$

Convergence speed of the fixed-point method. Let's first assume that we have a converging fixed-point method and $\varphi'(x^*) \ne 0$. We know that

$$x_{k+1} - x^* = \varphi(x_k) - \varphi(x^*) = \varphi'(\xi_k)(x_k - x^*)$$

by (MVT) with $\xi_k \in (x_k, x^*)$. Then

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} = |\varphi'(\xi_k)| \implies \lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = c = |\varphi'(x^*)|.$$

Conclude that $c = |\varphi'(x^*)| \in (0, 1) \implies$ linear convergence. Note, if $\varphi'(x^*) = 0$, then convergence is faster, e.g. NR.

Convergence and convergence speed of NR.
We have that

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad \text{and} \quad \varphi(x) = x - \frac{f(x)}{f'(x)}.$$

We will need $\varphi'(x)$, and $\varphi''(x)$:

$$\varphi'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}$$
$$\varphi''(x) = f(x)\left(\frac{f''(x)}{f'(x)^2}\right)' + \frac{f''(x)}{f'(x)}$$

1. Convergence of NR. We know the sufficient conditions for convergence:

$$\exists \delta, m: \quad |\varphi'(x)| \le m < 1, \quad \forall x \in I = \{x : |x - x^*| \le \delta\},$$

so for NR:

$$\left|\frac{f(x)f''(x)}{(f'(x))^2}\right| \le m < 1$$

for x sufficiently close to x^*. Suppose there is a closed interval J, containing x^*, s.t.

- $f(x)$ is continuous in J,
- $f'(x)$ is continuous in J. $f'(x^*) \ne 0$,
- $f''(x)$ is continuous in J (bounded).

Then there exists an interval $I \subset J$ containing x^*, s.t. NR converges to x^* for any initial guess in I.

Proof. (Sketch). We can find such an interval I where

$$\left|\frac{f(x)f''(x)}{(f'(x))^2}\right| \le m < 1$$

by continuity of f, f', f'', and using $f'(x^*) \ne 0$. Then use the convergence theorem for fixed-point methods. Note: $f'(x^*) \ne 0$ means that x^* is a simple root. □

2. Convergence order of NR. How does $|x_{k+1} - x^*|$ relate to $|x_k - x^*|$? We know $x_{k+1} - x^* = \varphi(x_k) - \varphi(x^*)$. Use Taylor series (with remainder theorem):

$$\varphi(x_k) = \varphi(x^*) + \varphi'(x^*)(x_k - x^*) + \tfrac{1}{2}\varphi''(\xi_k)(x_k - x^*)^2,$$

with $\xi_k \in (x_k, x^*)$. Assume a simple root ($f'(x^*) \neq 0$), then $\varphi'(x^*) = 0$, so

$$\varphi(x_k) - \varphi(x^*) = \tfrac{1}{2}\varphi''(\xi_k)(x_k - x^*)^2$$

or

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2} = \tfrac{1}{2}|\varphi''(x^*)|$$

Convergence with order $p = 2$

$$c = \tfrac{1}{2}|\varphi''(x^*)| = \tfrac{1}{2}\left|\frac{f''(x^*)}{f'(x^*)}\right|$$

Note: double root $\implies$ linear convergence.

Bisection method. Convergence speed is equivalent to linear, $c = 0.5$, $(p = 1)$

Secant method. $p = \frac{1+\sqrt{5}}{2} \approx 1.618$ (simple root).

Lecture 5

2.4 Error estimation and stopping criteria [1, 4.5]

2.4.1 Conditioning of the root finding problem

In numerical analysis:

Definition 2.6. A problem is *well-conditioned* if the solution is not highly sensitive to small perturbations in the problem formulation. Similarly a problem is *ill-conditioned* if the result is highly sensitive to small perturbations.

For the root finding problem, find x^* s.t. $f(x^*) = 0$.
Consider perturbed problem: $f(x)+\delta = 0$, where δ is small.

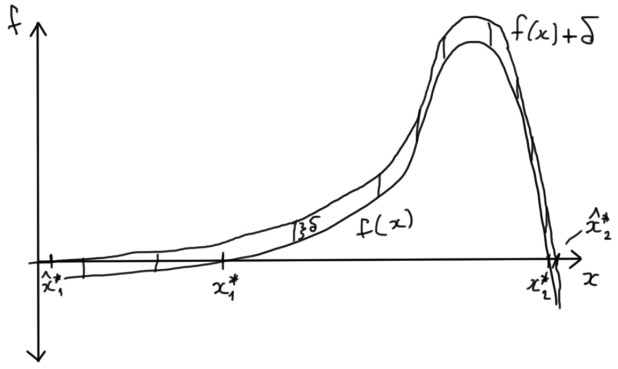


Figure 2.9: Diagram of $f(x) + \delta$.

If $|f'(x^*)|$ is small (≈ 0), then $|\hat{x}^* - x^*|$ is large for small δ, so root is ill-conditioned. (e.g. $|f'(x_1)|$ is small, $|\hat{x}_1^* - x_1^*|$ is large: ill-conditioned).

If $|f'(x^*)|$ is large ($\gg 1$), then $|\hat{x}^* - x^*|$ is small for small δ, so root is well-conditioned.

Note: the solution to an ill-conditioned problem is hard (or impossible) to compute accurately on a computer due to rounding errors etc.

2.4.2 Error estimation and attainable accuracy

Problem: find x^* s.t. $f(x^*) = 0$.
single root: $(f'(x^*) \neq 0)$.
Assume $f(x)$ and $f'(x)$ are continuous. Consider a root finding method (iterative) on a computer. Iteration is stopped at $x_k = \bar{x}$ (in most cases, $\bar{x} = x^*$). Let $\tilde{f}(\bar{x})$ be the computational approximation of $f(\bar{x})$.
(Note: due to rounding and other approximation errors, $\tilde{f}(\bar{x}) \neq f(\bar{x})$.)
Assume $|\tilde{f}(\bar{x}) - f(\bar{x})| \leq \delta$ (e.g. unit roundoff).
Question: can we bind $|\bar{x} - x^*|$?
Yes:

$$f(\bar{x}) = f(\bar{x}) - f(x^*) = f'(\xi)(\bar{x} - x^*) \quad (\text{MVT},\ \xi \in (\bar{x}, x^*)).$$

Hence

$$|\bar{x} - x^*| = \left|\frac{f(\bar{x})}{f'(\xi)}\right|.$$

Assume $|f'(x)| \geq M > 0$ in a neighbourhood of x^* that includes $\bar{x}$. Then $|\bar{x} - x^*| \leq \frac{|f(\bar{x})|}{M}$. Then use

$$|\tilde{f}(\bar{x}) - f(\bar{x})| \leq \delta \implies |f(\bar{x})| \leq |\tilde{f}(\bar{x})| + \delta$$

So

$$|\bar{x} - x^*| \leq \frac{|\tilde{f}(\bar{x})| + \delta}{M} \quad \text{“method-independent error estimate”.}$$

Note: the error bound can be made smaller by reducing $\tilde{f}(\bar{x})$ (more iterations). If $\tilde{f}(\bar{x}) = 0$, then

$$|\bar{x} - x^*| \leq \frac{\delta}{M} \quad \text{“attainable accuracy”.}$$

Note: $M \sim f'(x^*)$: link with condition of root. So small $f'(x^*) \iff$ ill-conditioned root $\iff$ large error bound.
double root: $f(x^*) = 0$, $f'(x^*) = 0$ $(f''(x^*) \neq 0)$.

$$f(\bar{x}) = \underbrace{f(x^*)}_{=0} + \underbrace{f'(x^*)}_{=0}(\bar{x} - x^*) + \frac{f''(\xi)}{2}(\bar{x} - x^*)^2$$

$$\implies |\bar{x} - x^*|^2 = 2\frac{|f(\bar{x})|}{|f''(\xi)|} \leq \frac{2|f(\bar{x})|}{M_2}$$

with $|f''(x)| \geq M_2 > 0$. So we have

$$|\bar{x} - x^*| \leq \sqrt{2\left(\frac{|\tilde{f}(\bar{x})| + \delta}{M_2}\right)} \quad \text{“method-ind. error estimate”}$$

and

$$|\bar{x} - x^*| \leq \sqrt{\frac{2\delta}{M_2}} \quad \text{“attainable accuracy”}$$

Note: double precision, rounding: $\delta \approx 10^{-16}$, $\sqrt{\delta} \approx 10^{-8}$, which is much worse for double root than for single root (if $M \sim M_2$).

2.4.3 Stopping criteria

Stop if

(1) $|x_{k+1} - x_k| \leq \tau_1$ (a tolerance)

(2) $f(x_k) \leq \tau_2$ (a tolerance)

(3) $k = k_{max}$

or a combination of these.

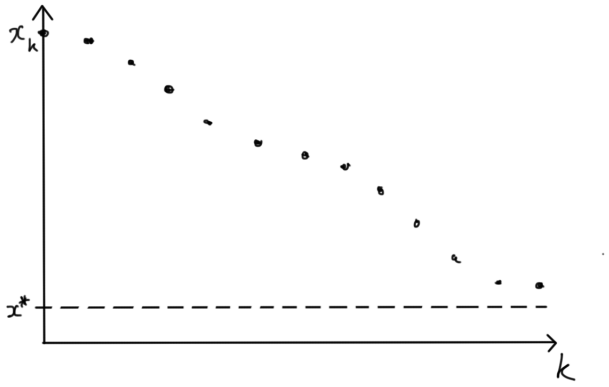


Figure 2.10: x_k converges to x^*.

Notes: it is difficult to find good values for τ_1, τ_2, and k_{max}.

(1) potential problem: may stop too soon

(2) hard to choose τ_2

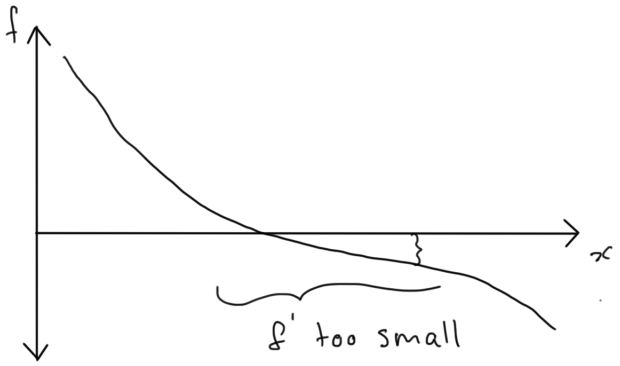


Figure 2.11: f' may be too small near the root.

Conclusion: trial-and-error.

2.5 Roots of a polynomial.

Consider $p(x) = a_1x^3 + a_2x^2 + a_3x + a_4$ (degree 3) (in general, degree n). One possibility: find roots by using NR efficiently

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}$$

How to compute $p(x)$ efficiently:

1) *naive approach*: compute $x^2, x^3, \ldots, x^n$: $(n-1)M$, where M is one multiplication.

   Compute $p(x)$: additionally, nM and nA (A is one addition). Then the total work done is:

$$\begin{aligned} W &= (2n-1)M + nA \\ &= 3n - 1 \text{ flops (floating point operations)} \end{aligned}$$

   (assume that, on the computer, additions and multiplications take about the same time).

2) *Horner's rule*: $p(x) = ((a_1x + a_2)x + a_3)x + a_4$.

$$W = nM + nA = 2n \text{ flops}$$

$p'(x)$ can also be computed efficiently using Horner, see [1].

2.5.1 Compute all roots of the polynomial.

1) *deflation*: find x_1 via NR. Then apply NR to $\frac{p(x)}{x-x_1}$.

   Problem:

   - Complex roots cannot be found.
   - NR may not converge unless the initial guess is chosen very close to x^*.

2) *eigenvalue method*: rescale $\tilde{p}(x) = x^3 + c_2x^2 + c_3x + c_4$ with $c_i = \frac{a_i}{a_1}$ ($a_1 \neq 0$). Consider the companion matrix of $\tilde{p}(x)$:

$$C = \begin{bmatrix} -c_2 & -c_3 & -c_4 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

   Characteristic polynomial of C:

$$C\vec{x} = \lambda\vec{x} \implies \begin{cases} -c_2x_1 - c_3x_2 - c_4x_3 = \lambda x_1 \\ x_1 = \lambda x_2 \\ x_2 = \lambda x_3 \end{cases}$$

   So $-c_2\lambda^2x_3 - c_3\lambda x_3 - c_4x_3 = \lambda^3x_3$, which gives

$$\lambda^3 + c_2\lambda^2 + c_3\lambda + c_4 = 0.$$

   Finding the roots of $\tilde{p}(x)$ is the same as finding the eigenvalues of C. We use iterative methods for eigenvalues from numerical linear algebra to find the eigenvalues (also the complex eigenvalues).

Lecture 6

2.6 Nonlinear systems [1, 4.8]

$$\begin{aligned} \text{ex. } f_1(x_1, x_2) &= 4x_1^2 + 9x_2^2 - 36 = 0 \\ f_2(x_1, x_2) &= 16x_1^2 - 9x_2^2 - 36 = 0. \end{aligned}$$

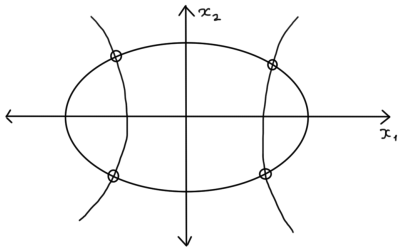


Figure 2.12: Example of a non-linear system.

2 equations, 2 unknowns, 4 solutions (nonlinear eq.)

Notation 2.7.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = (x_1, x_2)^T$$
$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = (f_1(x), f_2(x))^T$$

Then $f(x) = 0$ (also for $n = 3, 4, \ldots$, where n is the number of unknowns)

2.6.1 Newton-Raphson for systems

$$x^{[k]} = \begin{bmatrix} x_1^{[k]} \\ x_2^{[k]} \end{bmatrix}$$

Truncated Taylor: ($x^{[0]}$ is the initial guess)

$$\begin{aligned} 0 = f_1(x^*) \approx f_1(x^{[0]}) &+ \frac{\partial f_1}{\partial x_1}(x^{[0]})(x_1^* - x_1^{[0]}) \\ &+ \frac{\partial f_1}{\partial x_2}(x^{[0]})(x_2^* - x_2^{[0]}) \\ 0 = f_2(x^*) \approx f_2(x^{[0]}) &+ \frac{\partial f_2}{\partial x_1}(x^{[0]})(x_1^* - x_1^{[0]}) \\ &+ \frac{\partial f_2}{\partial x_2}(x^{[0]})(x_2^* - x_2^{[0]}) \end{aligned}$$

Definition 2.8. The *Jacobian* matrix of $f(x)$ is:

$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) \end{bmatrix}.$$

In NR, we want to find $x^{[1]} = (x_1^{[1]}, x_2^{[1]})^T$, such that

$$0 = \begin{bmatrix} f_1(x^{[0]}) \\ f_2(x^{[0]}) \end{bmatrix} + J(x^{[0]}) \begin{bmatrix} x_1^{[1]} - x_1^{[0]} \\ x_2^{[1]} - x_2^{[0]} \end{bmatrix}$$
$$\text{or} \quad 0 = f(x^{[0]}) + J(x^{[0]})(x^{[1]} - x^{[0]}),$$

or in general

$$x^{[k+1]} = x^{[k]} - J(x^{[k]})^{-1} f(x^{[k]}).$$

$$\left(\text{compare to 1D version: } x_{k+1} = x_k - \tfrac{f(x_k)}{f'(x_k)}\right).$$

In practice:

$$J(x^{[k]})h^{[k]} = -f(x^{[k]}) \text{ (solve linear system)}$$
$$x^{[k+1]} = x^{[k]} + h^{[k]}$$

Since inverting a matrix is about 3 times more expensive than solving a linear system (see later).

Theorem 2.9 (NR Convergence Theorem)**.** *If we have that*

- *$f(x)$ is thrice continuously differentiable in a neighbourhood of x^*,*
- *$J(x^*)$ is nonsingular, and*
- *$x^{[0]}$ is chosen sufficiently close to x^*,*

then NR converges quadratically:

$$\lim_{k\to\infty} \frac{\|x^{[k+1]} - x^*\|_2}{\|x^{[k]} - x^*\|_2^2} = c$$

Proof. No proof. □

2.6.2 Fixed-point method

As before we have $f(x^*) = 0 \iff x^* = \varphi(x^*)$.

Theorem 2.10 (Convergence Theorem for fixed-point)**.** *Let $I = \{x : \|x - x^*\|_2 \le \delta\}$. Let $D(x) = [d_{ij}(x)]$ (Jacobian of $\varphi(x)$), where $d_{ij}(x) = \frac{\partial \varphi_i}{\partial x_j}(x)$. Choose $x^{[0]} \in I$. If*

$$\exists 0 \le m < 1 : \|D(x)\|_2 \le m < 1 \ \forall x \in I,$$

then $x^{[k+1]} = \varphi(x^{[k]})$ converges linearly:

$$\|x^{[k+1]} - x^*\|_2 \le m\|x^{[k]} - x^*\|_2$$

Proof. No proof □

3 Numerical methods for ODEs [1, 10]

3.1 Introduction [1, 10.1]

Definition 3.1. The *initial value problem* (IVP) for a first-order scalar ODE is to find $y(x)$ s.t.

$$\begin{cases} y'(x) = f(x, y(x)) & \text{(ODE)} \\ x \in [a, b] & \text{(domain)} \\ y(a) = \alpha & \text{(initial condition)} \end{cases}$$

Example 3.2. Consider the following IVP:

$$\begin{cases} y'(x) = y(x) \ \longrightarrow \ y(x) = ce^x \\ x \in [0, 10] \\ y(0) = 1 \end{cases}$$

$y(0) = 1 = c \cdot 1 = c.$

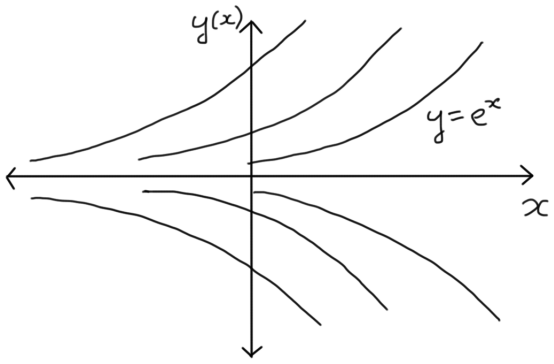


Figure 3.1: Given IVP, with a unique solution: $y = e^x$.

Note: we only consider first-order systems of ODEs, because high-order ODEs can be converted into first-order systems.

Example 3.3.

$$y'' + 3y' + 4y = \cos(x)$$

$$\begin{aligned} y_2'(x) + 3y_2(x) + 4y_1 &= \cos(x) \\ y_1'(x) - y_2(x) &= 0 \end{aligned}$$

Introduce new unknown functions:

$$\begin{aligned} y_1(x) &= y(x) \\ y_2(x) &= y'(x) \ (= y_1'(x)) \\ \to y_2'(x) &= y''(x) \end{aligned}$$

This gives us a linear first-order ODE system:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -4 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \cos(x) \end{bmatrix},$$

or rather $y' = Ay + b$, where

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -1 \end{bmatrix}, \quad y = \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \cos(x) \end{bmatrix}$$

More generally we have a nonlinear first-order ODE system:

$$y' = f(x, y(x)),$$

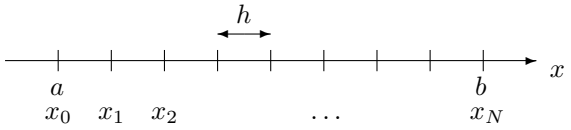
where y and f are vector functions, x is a scalar.

3.2 Euler's method [1, 10.2]

Consider an IVP:

$$\begin{cases} y'(x) = f(x, y(x)) \text{ (scalar)} \\ x \in [a, b] \\ y(a) = \alpha \end{cases}$$

Divide $[a, b]$ into N equidistant subintervals:



where $h = \frac{b-a}{N}$, and $x_n = a + nh$, for $n \in \{0, 1, \ldots, N\}$, are the "grid points".

Notation 3.4.
$y(x_n)$ is the exact solution.
y_n is the approximate, numerical solution.

3.2.1 Deriving the method

$$y'(x_n) = f(x_n, y(x_n))$$

approximate $y'(x_n)$ by a finite difference:

$$f(x_n, y(x_n)) = y'(x_n) \approx \frac{y(x_{n+1}) - y(x_n)}{h},$$

from the definition of the derivative.

Assume $y_n \approx y(x_n)$ is known. Then define y_{n+1} by:

$$f(x_n, y_n) = \frac{y_{n+1} - y_n}{h}$$

or

$$y_{n+1} = y_n + hf(x_n, y_n) \text{ (Euler's method)}$$

3.2.2 Local and global truncation errors

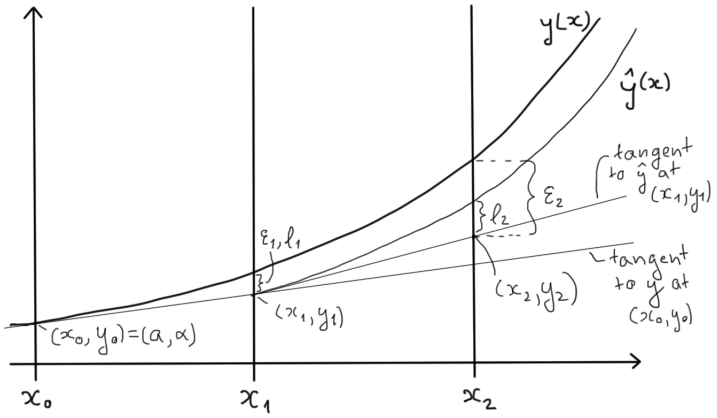


Figure 3.2: Euler's method.

Note: $\hat{y}(x)$ is an exact solution of $y' = f(x, y)$, with initial condition $y(x_1) = y_1$.

Definition 3.5.

$$\begin{aligned} \textit{Global truncation error:} \quad & \varepsilon_{n+1} := y(x_{n+1}) - y_{n+1} \\ \textit{Local truncation error:} \quad & \ell_{n+1} := \hat{y}(x_{n+1}) - y_{n+1} \end{aligned}$$

where $\hat{y}(x)$ is the exact solution of $y' = f(x, y)$ that goes through (x_n, y_n). Note: $\hat{y}(x_n) = y_n$.

Local truncation error for Euler's method.

$$\begin{aligned} \ell_{n+1} &= \hat{y}(x_{n+1}) - y_{n+1} \\ &= \hat{y}(x_n + h) - (y_n + hf(x_n, y_n)) \\ &= \hat{y}(x_n) + \hat{y}'(x_n)h + \tfrac{1}{2}\hat{y}''(\xi)h^2 - y_n - hf(x_n, y_n) \\ &\quad \text{(where } \xi \in (x_n, x_n + h)) \\ &= y_n + f(x_n, \underbrace{\hat{y}(x_n)}_{y_n})h + \tfrac{1}{2}\hat{y}''(\xi)h^2 - y_n - hf(x_n, y_n) \\ &= \tfrac{1}{2}\hat{y}''(\xi)h^2 \in \mathcal{O}(h^2) \text{ (second order)} \end{aligned}$$

Lecture 7

Recall: IVP

$$\begin{cases} y'(x) = f(x, y(x)) \\ x \in [a, b] \\ y(a) = \alpha \end{cases}$$

Euler's method:

$$\begin{cases} y_0 = \alpha \\ y_{n+1} = y_n + hf(x_n, y_n) \end{cases}$$

Note that Euler is an *explicit method* since there is an explicit formula to compute y_{n+1} from the previous value, y_n. Also recall the truncation errors:

$$\text{global:} \quad \varepsilon_{n+1} = \underbrace{y(x_{n+1})}_{\text{exact}} - \underbrace{y_{n+1}}_{approx.}$$

$$\text{and local:} \quad \ell_{n+1} = \hat{y}(x_{n+1}) - y_{n+1} = \tfrac{1}{2}\hat{y}''(\xi)h^2 \in \mathcal{O}(h^2).$$

With $\xi \in (x_n, x_n + h)$ and

$$\begin{cases} \hat{y}'(x) = f(x, \hat{y}(x)) \\ \hat{y}(x_n) = y_n \end{cases}$$

3.2.3 Convergence proof for Euler's method

Definition 3.6. We say $f(x, y)$ is *Lipschitz continuous* in y on $[a, b] \times (-\infty, \infty)$, if $\exists L \geq 0$ such that $\forall x \in [a, b]$, and $\forall y, \hat{y} \in (-\infty, \infty)$, we have

$$|f(x, y) - f(x, \hat{y})| \leq L|y - \hat{y}|$$

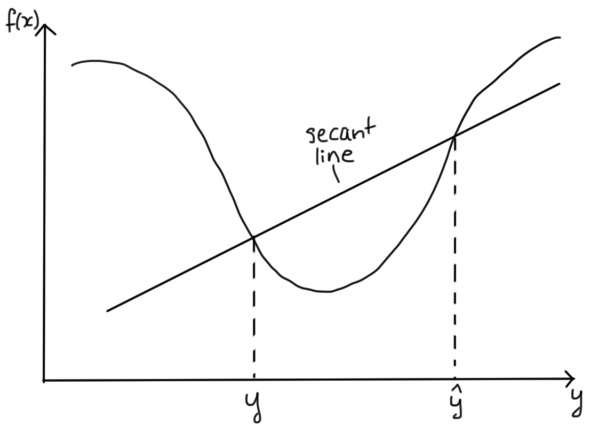


Figure 3.3: The slope of any secant line is bounded by L.

Note: Differentiable $\implies$ Lipschitz cont. $\implies$ continuous.

Theorem 3.7 (Global convergence of Euler's method with global order $\mathcal{O}(h)$)**.** *Consider IVP:*

$$\begin{cases} y'(x) = f(x, y(x)) \\ x \in [a, b] \\ y(a) = \alpha \end{cases}$$

Let $f(x, y)$ be continuous on $[a, b] \times (-\infty, \infty)$ and Lipschitz continuous in y on $[a, b] \times (-\infty, \infty)$ with Lipschitz constant L. Assume[1] *$\exists M < \infty : |y''(x)| \leq M$, $\forall x \in [a, b]$. Then Euler's method converges for any fixed $x_n = c$ (c fixed, $n \to \infty$) with order $\mathcal{O}(h)$.*

Proof.

$$\begin{aligned} \varepsilon_{n+1} &= y(x_{n+1}) - y_{n+1} \\ &= y(x_n) + hy'(x_n) + \tfrac{1}{2}h^2 y''(\xi_n) - (y_n + hf(x_n, y_n)) \\ &\qquad (\text{where } \xi_n \in (x_n, x_{n+1})) \\ \varepsilon_{n+1} &= \underbrace{\varepsilon_n + h(f(x_n, y(x_n)) - f(x_n, y_n))}_{\text{(propagation of) previous error}} + \underbrace{\tfrac{1}{2}h^2 y''(\xi_n)}_{\substack{\text{local truncation} \\ \text{error, new in this} \\ \text{step}}} \\ |\varepsilon_{n+1}| &\leq |\varepsilon_n| + h|f(x_n, y(x_n)) - f(x_n, y_n)| + \tfrac{1}{2}h^2|y''(\xi_n)| \\ &\leq |\varepsilon_n| + hL|y(x_n) - y_n| + \tfrac{1}{2}h^2 M \\ |\varepsilon_{n+1}| &\leq (1 + hL)|\varepsilon_n| + \tfrac{1}{2}h^2 M \\ |\varepsilon_n| &\leq (1 + hL)^n \underbrace{|\varepsilon_0|}_{=0} + \tfrac{1}{2}h^2 M \left(\sum_{k=0}^{n-1} (1 + hL)^k \right) \end{aligned}$$

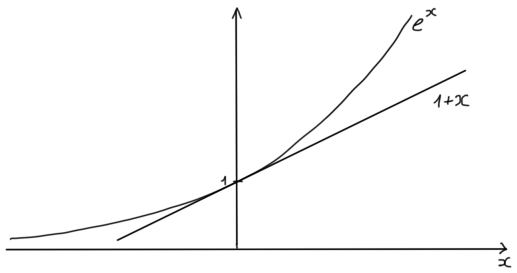
[1] This assumption also guarantees the existence of a unique exact solution to the IVP

Now use geom. series: $\sum_{k=0}^{p} r^k = s = \frac{r^{p+1}-1}{r-1}$, where $r \neq 1$. Then we have

$$|\varepsilon_n| \leq \tfrac{1}{2}h^2 M \left(\frac{(1 + hL)^n - 1}{hL} \right)$$

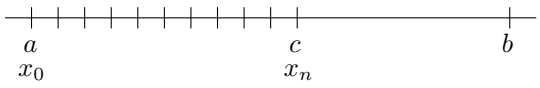
We then use that $e^x = 1 + x + \frac{1}{2}e^{\eta_x} x^2$, where $\eta_x \in (0, x)$. Hence

$$e^x \geq 1 + x \implies (e^x)^n \geq (1 + x)^n \quad (x \geq -1)$$



$$\implies |\varepsilon_n| \leq \tfrac{1}{2} \frac{hM}{L} (e^{hLn} - 1)$$

Note: $x_n = x_0 + hn$. Now fix c in $x_n = c$, and let $n \to \infty$ as $h \to 0$ (s.t. hn is fixed).



Then

$$\begin{aligned} |\varepsilon_n| &= |y(x_n) - y_n| = |y(c) - y_n| \\ &\leq \tfrac{1}{2} \frac{hM}{L} (e^{L(c - x_0)} - 1) \in \mathcal{O}(h) \end{aligned}$$

Note:
local truncation error: $\mathcal{O}(h^{p+1})$
global truncation error: $\mathcal{O}(h^p)$
due to accumulation of errors (reducing h requires more intervals). □

Take into account rounding errors, i.e. consider $\{\bar{y}_n\}_{n=0}^N$, instead of $\{y_n\}_{n=0}^N$. Have

$$\bar{y}_{n+1} = \bar{y}_n + hf(x_n, \bar{y}_n) + \mu\rho_n$$

where $\mu\rho_n$ is due to rounding errors in $\bar{y}_n + hf(x_n, \bar{y}_n)$. Then global error is given by

$$\delta_n = y(x_n) - \bar{y}_n.$$

So we get

$$|\delta_{n+1}| \leq (1 + hL)|\delta_n| + \tfrac{1}{2}h^2 M + \mu R$$

with $|\rho_n| \leq R$, $\forall n$. Assume that $fl(\alpha) = \alpha$, so $\delta_0 = 0$. $x_n = c$ (c fixed), then

$$|y(c) - \bar{y}_n| \leq \left| \left(\frac{h^2 M}{2} + \mu R \right) / (hL) \right| [\exp(L(c - x_0)) - 1]$$

$$\text{or} \quad |y(c) - \bar{y}_n| \le \left| \frac{hM}{2L} + \frac{\mu R}{hL} \right| [\exp(L(c - x_0)) - 1]$$

Note: $\mu \sim 10^{-16}$.

For 'large' h ($h \gg \mu$), we have $\mathcal{O}(h)$ convergence.

For 'small' h ($h \approx \mu$), rounding error may dominate.

Note: Euler for systems. Consider IVP as before, but with

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} f_1(x, y_1, y_2) \\ f_2(x, y_1, y_2) \end{bmatrix}$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad y' = f(x, y), \quad f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$

Then Euler's method for systems can be written as:

$$y^{[n+1]} = y^{[n]} + hf(x^{[n]}, y^{[n]}).$$

3.3 Runge-Kutta Methods [1, 10.4]

Try to get a better accuracy than Euler's $\mathcal{O}(h)$.

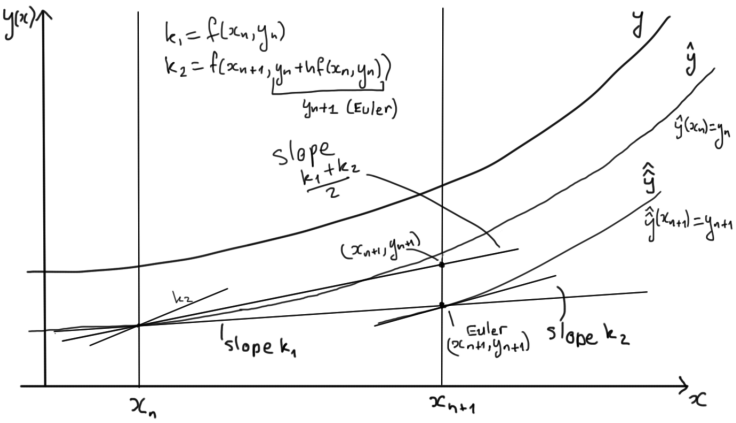


Figure 3.4: Runge-Kutta. Use slope $k = \frac{k_1+k_2}{2}$ to determine y_{n+1}.

Example 3.8 (Heun's Method)**.**

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_{n+1}, y_n + hk_1) \\ y_{n+1} &= y_n + h\left(\frac{k_1 + k_2}{2}\right) \end{aligned}$$

Note that this is a *non-linear* method.

Truncation errors:

$$\begin{aligned} &\ell_{n+1} = \hat{y}(x_{n+1}) - y_{n+1} \in \mathcal{O}(h^3) \\ &\varepsilon_{n+1} \in \mathcal{O}(h^2) \quad \text{(1 order better than Euler)} \end{aligned}$$

Note: "classical" 4-stage RK method: [1, p. 321] global order $\mathcal{O}(h^4)$. 4 stages, 4 evaluations of $f(x, y)$ per step.

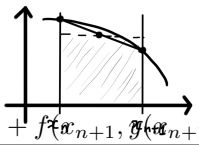
Lecture 8

3.3.1 An implicit method [1, 10.5]

Trapezoidal method. Consider $y' = f(x, y(x))$ as usual. Then integrate (assume $y(x)$ is known):

$$\int_{x_n}^{x_{n+1}} y'(x)\,dx = \int_{x_n}^{x_{n+1}} f(x, y(x))\,dx$$

Now use the trapezoid rule for numerical integration to approximate the integral:



$$\int_{x_n}^{x_{n+1}} f(x, y(x))\,dx \approx h\frac{f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))}{2}$$

$$\implies y(x_{n+1}) - y(x_n) \approx h\frac{f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))}{2}$$

Now find y_n and y_{n+1} such that

$$y_{n+1} - y_n = h\frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1})}{2}$$

Note: implicit: if f is a non-linear function, we need to solve a non-linear equation to find y_{n+1}.

Truncation errors:

local truncation error $\mathcal{O}(h^3)$,

global truncation error $\mathcal{O}(h^2)$ (better than Euler).

Note: how to solve a linear system?

1) NR

2) fixed-point

$$\begin{aligned} y_{n+1}^{[k+1]} &= \varphi(y_{n+1}^{[k]}) \\ y_{n+1}^{[k+1]} &= y_n + h\frac{f(x_n, y_n)}{2} + h\frac{f(x_{n+1}, y_{n+1}^{[k]})}{2}. \end{aligned} \tag{3.1}$$

Initial guess:

predictor: 1 step of Euler: $y_{n+1}^{[0]} = y_n + hf(x_n, y_n)$.

corrector: iterate on equation (3.1).

This is called the *predictor-corrector* method.

Sufficient condition for convergence of corrector:

$$\begin{aligned} &\left|\frac{d\varphi(y_{n+1})}{dy_{n+1}}\right| < 1 \\ \implies &\frac{d\varphi}{dy_{n+1}} = \left|\frac{h}{2}\frac{\partial f}{\partial y}\right| < 1 \\ \implies &h < \frac{2}{|\partial f/\partial y|} \end{aligned}$$

in a neighbourhood of the point (x_{n+1}, y_{n+1}). Note that the disadvantage of predictor-corrector procedure (fixed-point) is the limitation on the size of h. h should be not too small.

Note: many implicit methods are more numerically stable than explicit methods (larger h can be used without "blowup"). Many explicit methods become unstable[2] numerically when h is too large.

[2] Exponential growth of the error for $n \to \infty$, when h is fixed.

3.3.2 Numerical Stability [1, 10.6]

Previously we studied convergence of Euler methods. Fix $c = x_n = x_0 + nh$, then let $h \to 0$ as $n \to \infty$ such that hn is constant. Then we asked "does y_n converge to $y(x_n) = y(c)$?"

Now we study numerical stability: fix h, and let $n \to \infty$. Stability question: "does y_n stay close to $y(x_n)$ as $n \to \infty$ (fixed h) and $x_n \to \infty$?"

Note: some convergence methods may be unstable for any h.

First: Consider scalar ODEs. We study the "test equation":

$$\begin{cases} y' = \lambda y \quad (\lambda < 0) \implies & y(x) = \exp(\lambda x) \\ y(0) = 1 & \text{note } \lim_{x\to\infty} y(x) = 0 \end{cases}$$

Numerical stability: we require that $\lim_{n\to\infty} y_n = 0$.
This implies that $\lim_{n\to\infty} \varepsilon_n = 0$ (global truncation error).

Example 3.9. Apply

$$y_{n+1} = y_n + hf(x_n, y_n) \quad \text{(Euler's method)}$$

to the test equation:

$$\begin{aligned} y_{n+1} &= y_n + h\lambda y_n \\ &= (1 + h\lambda) y_n \\ &= (1 + h\lambda)^n y_0 \end{aligned}$$

we require $\lim_{n\to\infty} y_{n+1} = 0$

$$\begin{aligned} &\implies |1 + h\lambda| < 1, \\ &\text{or } -1 < 1 + h\lambda < 1 \\ &\text{or } -2 < h\lambda < 0 \end{aligned}$$

So we get the stability condition for Euler method:

$$h < \frac{-2}{\lambda}.$$

In general, $y' = f(x, y)$ is stable iff

$$h < \frac{2}{|\partial f/\partial y|} \quad \text{(approximately)}. \tag{3.2}$$

Reason: use Taylor to get

$$y' = f(x, y) \approx f(x, y_0) + \frac{\partial f}{\partial y}(x, y_0)(y - y_0),$$

then apply the previous stability analysis to the linearized equation to get the result (3.2). Note that (3.2) is a guideline, because it may not actually be stable depending on the function. Similarily:

$$\left.\begin{aligned} \text{Heun RK:} \quad & h < \tfrac{2}{|\lambda|} \\ \text{RK4:} \quad & h \lesssim \tfrac{2.785}{|\lambda|} \end{aligned}\right\} \text{ explicit methods}$$

Question: why not consider $\lambda > 0$ in test equation?
Suppose $\lambda > 0$, then $\lim_{x\to\infty} y(x) = \infty$, so $\lim_{n\to\infty} y_n = \infty$ is okay, and $\lim_{n\to\infty} |y(x_n) - y_n| = \infty$ is not a problem.

Observe: the concept of *absolute stability*[3] is not relevant in this case. It is possible to define a more relevant concept of *relative stability*.

[3] also known as numerical stability

Example 3.10 (Trapezoidal method)**.** Test equation:

$$y' = \lambda y \quad (\lambda < 0)$$

Trapezoid:

$$y_{n+1} = y_n + h\frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1})}{2}$$

apply to test equation

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}(\lambda y_n + \lambda y_{n+1}) \\ y_{n+1} &= \left(\frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda}\right) y_n \end{aligned}$$

we require

$$\left|\frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda}\right| < 1 \implies \textit{conditionally stable}$$

(holds for any $h > 0$ since $\lambda < 0$)

Example 3.11 (Midpoint Rule)**.** Have $y' = f(x, y)$. Approximate $y'(x_n)$:

$$y'(x_n) \approx \frac{y(x_{n+1}) - y(x_{n-1})}{2h} \quad (\textit{central finite difference})$$

Instead of

$$y'(x_n) \approx \frac{y(x_{n+1}) - y(x_n)}{h} \quad \text{(Euler)}.$$

Then we have that

$$\frac{y(x_{n+1}) - y(x_{n-1})}{2h} \approx f(x_n, y(x_n))$$

$$\begin{aligned} &\implies \frac{y_{n+1} - y_{n-1}}{2h} = f(x_n, y_n) && \textit{midpoint rule} \\ &\implies y_{n+1} = y_{n-1} + 2hf(x_n, y_n) && \text{explicit, 2-step} \end{aligned}$$

Local truncation error $\mathcal{O}(h^3)$,
global truncation error $\mathcal{O}(h^2)$.
This method is convergent on a fixed interval (as $h \to 0$).

Absolute stability. Apply to test equation:

$$y_{n+1} = y_{n-1} + 2h\lambda y_n \quad (\textit{difference equation}) \tag{3.3}$$

Assume: $y_n = cr^n$ (ODE: $y(x) = c\exp(rx)$). Plug into (3.3):

$$\begin{aligned} cr^{n+1} &= cr^{n-1} + 2h\lambda cr^n \\ \implies r^2 - 2hr\lambda - 1 &= 0 \quad (\textit{characteristic polynomial}) \end{aligned}$$

Two roots:

$$r_{1,2} = \frac{2h\lambda \pm \sqrt{4h^2\lambda^2 + 4}}{2} = h\lambda \pm \sqrt{h^2\lambda^2 + 1}$$

General solution of (3.3):

$$y_n = c_1 r_1^n + c_2 r_2^n.$$

We require

$$\begin{cases} \lim_{n\to\infty} y_n = 0 \quad (\lambda < 0,\ h\lambda < 0) \\ |r_1| < 1 \\ |r_2| < 1 \end{cases}$$

We have

$$\begin{aligned} r_1 &= h\lambda + \sqrt{h^2\lambda^2 + 1} \\ r_2 &= h\lambda - \sqrt{h^2\lambda^2 + 1} \\ \implies r_2 &< -1 \end{aligned}$$

Since $h\lambda + 1 < \sqrt{h^2\lambda^2 + 1}$:

a) $h\lambda + 1 > 0 \implies h^2\lambda^2 + 2h\lambda + 1 < h^2\lambda^2 + 1$ which is ok since $2h\lambda < 0$.

b) $h\lambda + 1 < 0$ clearly ok.

Therefore midpoint rule is unstable for any h.

Midpoint rule is not used since it is not stable.

Lecture 9

3.4 Numerical ODEs

Recall, numerical stability [1, 10.6]. We have the test equation

$$y' = \lambda y \ (\lambda < 0)$$

$$y(t) = c\exp(\lambda t)$$

$$\lim_{t\to\infty} y(t) = 0$$

Fix h; apply numerical ODE method to test equation. Require

$$\lim_{n\to\infty} y_n = 0$$

ex: Euler:

$$\begin{aligned} y_{n+1} &= y_n + hf(x_n, y_n) \\ y_{n+1} &= y_n + h\lambda y_n \\ h &< \frac{2}{|\lambda|} \text{ is required for stability} \end{aligned}$$

3.4.1 Numerical stability for systems

Consider linear systems: (we can linearize nonlinear ODE systems) IVP:

$$\begin{cases} y' = Ay \\ x \in [a, b] \\ y(a) = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{cases}$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \in \mathbb{R}^2$$

assume A has two linearly independent eigenvectors (A is diagonalizable)

$$\begin{aligned} Av^{[1]} &= \lambda_1 v^{[1]} \\ Av^{[2]} &= \lambda_2 v^{[2]} \end{aligned}$$

$$\begin{aligned} A[v^{[1]}|v^{[2]}] &= [v^{[1]}|v^{[2]}] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \\ \text{or } AR &= R\Lambda \quad (R = [v^{[1]}|v^{[2]}]) \\ A = R\Lambda R^{-1} &= R\Lambda L \quad \left(\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\right) \end{aligned}$$

or $LAR = \Lambda$ (A is diagonalizable). We call LAR the similarity transformation. Note: general solution to $y' = Ay$:

$$y(x) = c_1 v^{[1]} \exp(\lambda_1 x) + c_2 v^{[2]} \exp(\lambda_2 x)$$

So we have

$$\begin{aligned} y' &= R\Lambda L y \\ (Ly)' &= \Lambda(Ly) \\ \text{let } z &= Ly \quad \text{(change of variables)} \\ \implies z' &= \Lambda z \\ \text{or } z_1' &= \lambda_1 z_1 \\ z_2' &= \lambda_2 z_2 \quad \text{(the ODE system has been decoupled)} \end{aligned}$$

TFAE:

(a) numerical stability for $y' = Ay$.

(b) numerical stability for $z' = \Lambda z$ (follows from (a) by linearity)

(c) numerical stability for $z_i' = \lambda_i z_i$ $(i = 1, 2, \ldots, n)$.

Note: For a real A, there may be complex conjugate λ_i.

Note: This is really our good old test equation with complex λ_i.

Test eq. $z' = \lambda z$ where z is complex and $\mathrm{Re}(\lambda) < 0$. Note: why $\mathrm{Re}(\lambda) < 0$: let $\lambda = s + it$, with $\mathrm{Re}(\lambda) = s < 0$. Then

$$\begin{aligned} z(x) &= c\exp(\lambda x) \\ &= c\exp(sx)\exp(itx) \\ &= c\exp(sx)(\cos(tx) + i\sin(tx)) \\ \lim_{x\to\infty} z(x) &= 0 \quad \text{if} \quad s = \mathrm{Re}(\lambda) < 0 \end{aligned}$$

Example 3.12 (Euler (explicit))**.**

$$y_{n+1} = y_n + hf(x_n, y_n)$$

ODE: $y' = Ay$.

A: eigenvalues λ_i $(i = 1, \ldots, n)$, assume $\mathrm{Re}(\lambda_i) < 0$. consider test equations

$$z' = \lambda_i z \quad \text{for } (i = 1, \ldots, n) \tag{3.4}$$

numerical stability: choose h s.t. Euler is stable ($\lim_{n\to\infty} z_n = 0$) for $z' = \lambda_i z$ for all $i = 1, \ldots, n$. Note that

$z_{n+1} = (1 + h\lambda_i)^n z_0$ by applying Euler repeatedly to (3.4). We require: $|1 + h\lambda_i| < 1$ (because then $\lim_{n\to\infty} z_n = 0$)

If λ_i is real: $h < \frac{2}{|\lambda_i|}$ or $h < \frac{-2}{\lambda_i}$ or $h\lambda_i > -2$.

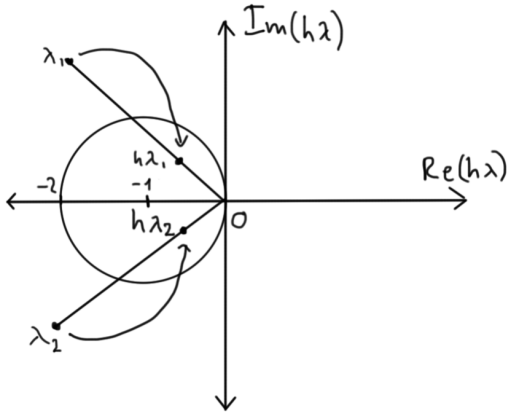


Figure 3.5: Region of absolute stability for Euler is a circle with radius 1 centred at (-1,0). We choose h such that $h\lambda_i \in$ stability region.

TFAE:

- $|1 + h\lambda| < 1$
- $|(1, 0) + (\mathrm{Re}(h\lambda), \mathrm{Im}(h\lambda))| < 1$
- $\sqrt{(1 + \mathrm{Re}(h\lambda))^2 + (\mathrm{Im}(h\lambda))^2} < 1$ and $\sqrt{(x - x_0)^2 + (y - y_0)^2} = r$ is the equation for a circle with radius r centred at (x_0, y_0).

Example 3.13 (Heun (2-stage RK) (explicit))**.**

$$|1 + h\lambda + \frac{1}{2}h^2\lambda^2| < 1$$

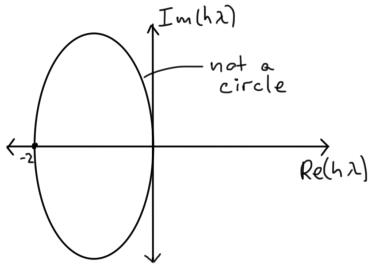


Figure 3.6: Stability region for Heun.

Example 3.14 (Trapezoid (implicit))**.**

$$\left|\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}\right| < 1$$

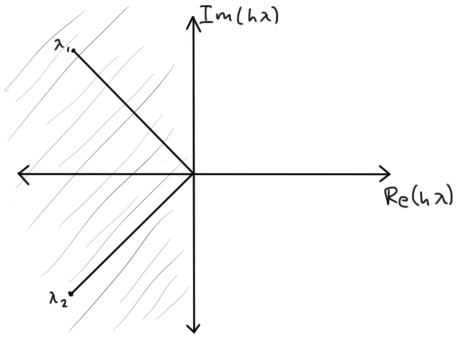


Figure 3.7: The left half-plane is the region of absolute stability in the trapezoid method.

Let $\frac{h\lambda}{2} = -a + ib$ where $(a > 0)$, then

$$\sqrt{(1 - a)^2 + b^2} < \sqrt{(1 + a)^2 + b^2}$$

is always true for $a > 0$. Hence the left sub-plane is the region of absolute stability. Therefore this method is stable for any h, i.e. unconditionally stable.

In general implicit methods tend to be more stable than explicit methods (but not always unconditionally stable).

3.5 Stiff ODE IVPs

"Stiffness" of ODEs is hard to define mathematically, so instead we will give an example:

Consider the IVP:

$$\begin{cases} u'' + 101u' + 100u = 0 \\ u(0) = 1.1 \\ u'(0) = -11 \\ x \in [0, 1000] \end{cases}$$

(1) Exact solution:

$$u(x) = c\exp(\lambda x)$$
$$0 = c\lambda^2 \exp(\lambda x) + c101\lambda \exp(\lambda x) + c100\lambda \exp(\lambda x)$$

characteristic polynomial: $\lambda^2 + 101\lambda + 100 = 0$, with

$$\lambda_{1,2} = \frac{-101 \pm \sqrt{101^2 - 400}}{2} = \begin{cases} -1 \\ -100 \end{cases}$$

general solution:

$$u(x) = c_1 \exp(-x) + c_2 \exp(-100x)$$

$$c_1 = 1 \quad (\text{since } u(0) = 1.1 \text{ and } u'(0) = -11)$$
$$c_2 = 0.1$$

(2) Write as a system to investigate numerical stability

$$u = y_1$$
$$u' = y_2 = y_1'$$

$$y_2' = -101y_2 - 100y_1$$
$$y_1' = y_2$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -100 & -101 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

where $y_1(0) = 1.1$ and $y_2(0) = -11$. Eigenvalues of A as given by the characteristic polynomial:

$$\begin{vmatrix} -\lambda & 1 \\ -100 & -101 - \lambda \end{vmatrix} = \lambda^2 + 101\lambda + 100 \qquad (3.5)$$

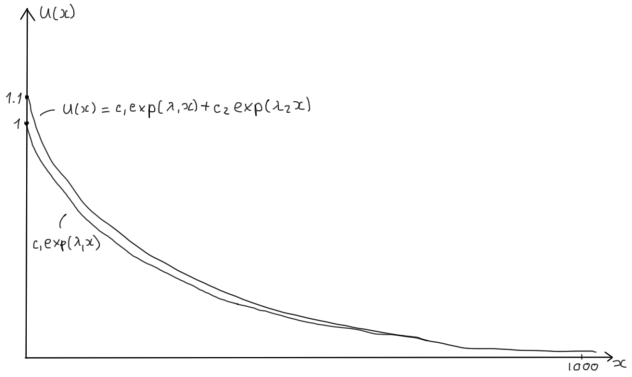


Figure 3.8: The difference between the two functions diminishes with increasing x.

Note: $c_2 \exp(\lambda_2 x)$ only matters[4] for small x.

Note: Assume x is time, then

$c_2 \exp(\lambda_2 x)$ changes on a short (fast) timescale and $c_1 \exp(\lambda_1 x)$ changes on a long (slow) timescale

Definition 3.15. IVP is called *stiff* when

1) there are multiple disparate timescales in the problem and

2) the fast timescale is not important on the timescale of the IVP.

(suppose we are interested in modelling a physical problem, and we are particularly interested in the slow, long-term changes)

In case of a stiff problem:

- a small h (timestep) is not required for accuracy (because changes are slow, large h is sufficient)
- a short timescale in the general solution ($\lambda_2 = -100$) forces us to use a very small timestep for Euler, Heun (inefficient).

Unconditionally stable (implicit) methods are useful for stiff IVPs (choose h large based on accuracy, h not limited by stability)

Lecture 10

[4]Significantly contributes to the solution

3.6 Adaptive step length control [1, 10.7]

Choose h adaptively in order to limit the local truncation error in every step.

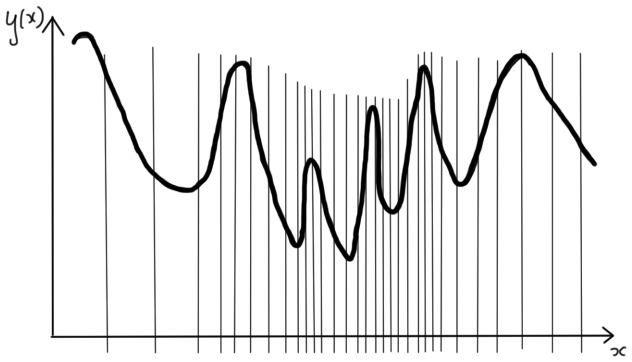


Figure 3.9: Adaptive steps length.

- We need to estimate the local truncation error and
- Take smaller steps when the error is estimated to be too large.

3.6.1 Estimate the local truncation error

Given

$$x_0, x_1, \ldots, x_n$$
$$y_0, y_1, \ldots, y_n$$

with current h. Take the next step: x_{n+1}, y_{n+1}. Estimate the error in step $n+1$. Consider two methods of different order.

Method A: RK4[5] (4 stages, local order 5)

local truncation error:

$$\ell_{n+1}^{(A)} = y_{n+1}^{(A)} - \hat{y}(x_{n+1}) = \mathcal{O}(h^5) = ch^5 + \mathcal{O}(h^6) \approx ch^5$$

where $\hat{y}(x)$ is an exact solution of $y' = f(x, y(x))$ going through (x_n, y_n).

Method B: RK5 (5 stages, local order 6, global order 5)

$$\ell_{n+1}^{(B)} = y_{n+1}^{(B)} - \hat{y}(x_{n+1}) = \mathcal{O}(h^6)$$

$$y_{n+1}^{(A)} - y_{n+1}^{(B)} = ch^5 + \mathcal{O}(h^6) = \ell_{n+1}^{(A)} \approx ch^5$$

use $y_{n+1}^{(A)} - y_{n+1}^{(B)}$ to estimate $\ell_{n+1}^{(A)}$ but we also know how $\ell_{n+1}^{(A)}$ depends on h. (approximately)

3.6.2 Take smaller steps when error is estimated too large

Once we know $\ell_{n+1}^{(A)}$, how can we adapt h to limit $\ell_{n+1}^{(A)}$ to a fixed tolerance.

Given h, used to compute $y_{n+1}^{(A)}$ and $y_{n+1}^{(B)}$. We want to find a new h s.t.

$$|\ell_{n+1}^{(A)}| < \delta$$

[5]the 4 indicates the global order

determine the estimate for the optimal timestep, h_{opt}:

$$
\begin{aligned}
|\ell_{n+1}^{(A)}| &\approx \delta \\
|ch_{opt}^5| &\approx \delta \\
h_{opt} &\approx \left(\frac{\delta}{c}\right)^{1/5} \\
h_{opt} &\approx \left(\frac{\delta h^5}{|y_{n+1}^{(A)} - y_{n+1}^{(B)}|}\right)^{1/5} \\
\text{or } h_{opt} &= h\left(\frac{\delta}{|y_{n+1}^{(A)} - y_{n+1}^{(B)}|}\right)^{1/5} \\
\text{or } h_{opt} &= h\gamma \\
\text{where } \gamma &= \left(\frac{\delta}{|y_{n+1}^{(A)} - y_{n+1}^{(B)}|}\right)^{1/5}
\end{aligned}
$$

3.6.3 Algorithm (similar to `ode45`)

Consider interval $[x_n, x_{n+1}]$.

$$h = x_{n+1} - x_n$$

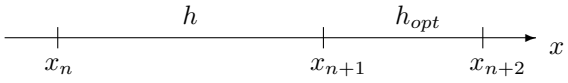
Compute

$$\left.\begin{array}{l} y_{n+1}^{(A)} \text{ (with } \mathcal{O}(h^5) \text{ local order)} \\ y_{n+1}^{(B)} \text{ (with } \mathcal{O}(h^6) \text{ local order)} \end{array}\right\} \text{ using timestep } h$$

Estimate $\ell_{n+1}^{(A)} \approx y_{n+1}^{(A)} - y_{n+1}^{(B)}$.

Compute $h_{opt} = h\left(\frac{\delta}{|y_{n+1}^{(A)} - y_{n+1}^{(B)}|}\right)^{1/5} = h\gamma$.

Case 1: $|y_{n+1}^{(A)} - y_{n+1}^{(B)}| < \delta$: accuracy using h is sufficient.



Accept the current computation for y_{n+1}.
Use $(x_{n+1}, y_{n+1}^{(B)})$ as new approximation.
Use h_{opt} as initial step length in step $[x_{n+1}, x_{n+2}]$.

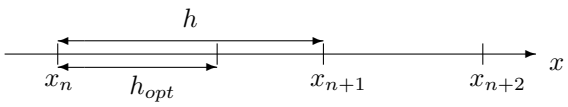
Case 2: $|y_{n+1}^{(A)} - y_{n+1}^{(B)}| \geq \delta$: not sufficiently accurate using h. Recompute step $[x_n, x_{n+1}]$ using $h \leftarrow h_{opt}$.

Notes:

1. Safety factors are needed.

$$\gamma = \min\left\{0.8\left(\frac{\delta}{|y_{n+1}^{(A)} - y_{n+1}^{(B)}|}\right)^{1/5}, 5\right\}$$

Where 0.8 is for efficiency[6] and 5 for stability.



[6] Conservative estimate, to avoid redoing a step if the estimate was just a bit too small

2. Also works for systems: $\|y_{n+1}^{(A)} - y_{n+1}^{(B)}\|_2$

3. For efficiency: construct special RK4 and RK5 pairs that use common function evaluation points ("nested").

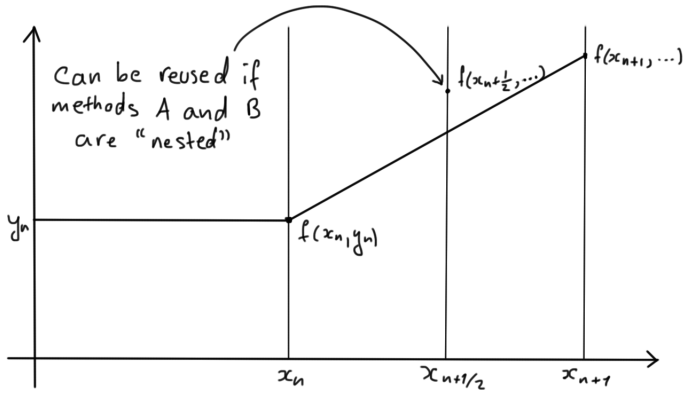


Figure 3.10: RK45 - Fehlberg pair [1, 10.7].

4 LU Decomposition of a square matrix - solving linear systems $Ax = b$ [1, 8]

4.1 Introduction

Solve $Ax = b$, where $A \in \mathbb{R}^{n\times n}$, and $x, b \in \mathbb{R}^n$.
Recall: Gaussian elimination two phases:

1) Reduce A to upper triangular form by row operations.

2) Solve the reduced system by back substitution.

Example 4.1.

$$
\begin{aligned}
A &= \begin{bmatrix} \mathbf{1} & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \\
\to A' &= \begin{bmatrix} 1 & 2 & 3 \\ 0 & \mathbf{-3} & -6 \\ 0 & -6 & -20 \end{bmatrix} \quad \begin{array}{ll} (2') = (2) - m_{21}(1) & m_{21} = \frac{4}{1} \\ (3') = (3) - m_{31}(1) & m_{31} = \frac{7}{1} \end{array} \\
\to A'' &= \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & -28 \end{bmatrix} \quad \begin{array}{ll} (3'') = (3') - m_{32}(2') & m_{32} = \frac{-6}{-3} \end{array}
\end{aligned}
$$

Where A'' is upper triangular. Matrix elements in **bold** are called *pivot elements*.

4.2 LU decomposition

Definition 4.2. Gauss transformation matrix:

$$L_k = \begin{array}{c} \\ k \end{array} \begin{bmatrix} & & k & & \\ 1 & & \vdots & & \\ & \ddots & \vdots & & \\ \cdots & \cdots & 1 & & \\ & & m_{k+1,k} & \ddots & \\ & & \vdots & & \ddots \\ & & m_{n,k} & & 1 \end{bmatrix}$$

Example 4.3.

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 0 & 1 \end{bmatrix}$$

Properties of L_k:

(1)

$$L_k^{-1} = \begin{array}{c} \\ k \end{array} \begin{bmatrix} & & k & & \\ 1 & & \vdots & & \\ & \ddots & \vdots & & \\ \cdots & \cdots & 1 & & \\ & & -m_{k+1,k} & \ddots & \\ & & \vdots & & \ddots \\ & & -m_{n,k} & & 1 \end{bmatrix}$$

Proof.

$$y = L_k x$$

$$\Longrightarrow \begin{cases} y_i = x_i, & \text{for } i = 1, \ldots, k, \text{ and} \\ y_i = x_i + m_{ik} x_k, & \text{for } i = k+1, \ldots, n \end{cases}$$

$$\Longrightarrow \begin{cases} x_i = y_i, & \text{for } i = 1, \ldots, k \text{ and} \\ x_i = y_i - m_{ik} \underbrace{y_k}_{x_k} & \text{for } i = k+1, \ldots, n \end{cases}$$

□

(2) Assume $k < \ell$:

$$L_k L_\ell = \begin{array}{c} \\ k \\ \\ \ell \end{array} \begin{bmatrix} & & k & & \ell & \\ 1 & & \vdots & & \vdots & \\ & \ddots & \vdots & & \vdots & \\ \cdots & \cdots & 1 & & \vdots & \\ & & m_{k+1,k} & \ddots & \vdots & \\ \cdots & \cdots & \vdots & \cdots & 1 & \\ & & \vdots & & m_{\ell+1,\ell} & \ddots \\ & & m_{n,k} & & m_{n,\ell} & & \ddots \\ & & & & & & 1 \end{bmatrix}$$

ex. $\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & c & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix}$

Note:

$$A[b_1|b_2|b_3] = [Ab_1|Ab_2|Ab_3]$$

back to Example 4.3:

$$\begin{aligned} & A \\ \to & A' = L_1^{-1} A \\ \to & \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -20 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -7 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix} \end{aligned}$$

Note:

$$\begin{bmatrix} \frac{a_1}{a_2} \\ \hline a_3 \end{bmatrix} B = \begin{bmatrix} \frac{a_1 B}{a_2 B} \\ \hline a_3 B \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}$$

$$\to A'' = L_2^{-1} A'$$

This gives

$$U = A'' = L_2^{-1} L_1^{-1} A$$

$$\Longrightarrow L_1 L_2 U = A$$

let $L = L_1 L_2$, then

$$LU = A$$

this is the LU decomposition of A

$$A = LU$$

with U: upper triangular,
L: unit lower triangular.
Gaussian elimination is the same as LU decomposition.

Lecture 11

4.2.1 Solving a linear system, $Ax = b$

1) $A = LU$

2) $L \underbrace{Ux}_{y} = b$:
Solve $Ly = b$ for y by forward substitution.
Solve $Ux = y$ for x by backward substitution.

Matlab code and computational cost.

(a) LU decomposition:

```
1  function [L,U] = lufac(A)
2    n = size(A,1);
3    L = eye(n);
4    U = A;
5    for k = 1:n-1 % pivot rows
6      for l = k+1:n % rows below pivot
7        m = U(l,k)/U(k,k);
8        U(l,k) = 0;
9        for c = k+1:n
10         U(l,c) = U(l,c) - m*U(k,c);
11       end
12       L(l,k) = m;
13     end
14   end
```

Note on optimizations:

1. Store U in A (save memory)

2. Store L in A (save memory)

3. vectorize in MATLAB, get 1 `for` loop [1, p. 212]

Computational work: $A \in \mathbb{R}^{n\times n}$
Computational complexity:
work: $W \in \mathcal{O}(n^3)$.
Recall:

$$\sum_{k=1}^{n-1} 1 = n-1$$
$$\sum_{k=1}^{n-1} k = \frac{1}{2}n(n-1)$$
$$\sum_{k=1}^{n-1} k^2 = \frac{1}{6}n(n-1)(2n-1) \quad \text{(proof in [1, p. 195])}$$

Pivot rows: `k=1:n-1` $\implies n-1$ pivot rows.
for each k: $n-k$ rows below the pivot:
row l: 1 flop[7] for computing $m_{lk} = \frac{a_{lk}}{a_{kk}}$.

$$\begin{aligned} W &= \sum_{k=1}^{n-1}(2\underbrace{(n-k)}_{\text{cols}}\underbrace{(n-k)}_{\text{rows}}+(n-k)) \\ &= \sum_{k=1}^{n-1}(2(n-k)^2+(n-k)) \\ &= 2\sum_{k=1}^{n-1}(n-k)^2 + \mathcal{O}(n^2) \\ &= 2\sum_{r=1}^{n-1} r^2 + \mathcal{O}(n^2) \\ &= \frac{2}{6}n(n-1)(2n-1) + \mathcal{O}(n^2) \\ &= \frac{2}{3}n^3 + \mathcal{O}(n^2) \in \mathcal{O}(n^3) \qquad \text{(for large } n\text{)} \end{aligned}$$

(b) $Ly = b$: forward substitution.

```
1  function y = forward(L,b)
2    n = size(L,1);
3    y = b;
4    for k = 2:n
5      for c = 1:k-1
6        y(k) = y(k) - L(k,c)*y(c);
7      end
8    end
```

(c) $Ux = y$: backward substitution.
$W = n^2 + \mathcal{O}(n)$ flops.

4.3 Pivoting [1, 8.4]

Consider the equation $Ax = b$.

Example 4.4.

$$\begin{bmatrix} \textcircled{0} & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad Ax = b$$
$$\begin{bmatrix} \textcircled{1} & 2 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad Cx = d$$

where the circled position has to be non-zero to be used as a pivot. Say

$$C = LU = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

Note: A and C are related by a permutation matrix.

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad PA = C \quad \to \quad PA = LU$$

$$PA = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} A = \begin{bmatrix} p_1 A \\ p_2 A \end{bmatrix} = \begin{bmatrix} [0\ 1]A \\ [1\ 0]A \end{bmatrix}$$

[7] A floating point operation, which could be one of: multiplication, addition, division or subtraction.

Definition 4.5. P is a *permutation matrix* iff P is all zero, except for a single 1 in each row and each column. Such P can be obtained from the identity matrix by interchanging rows (or columns).

Note: multiplication by P on the left switches rows.
right switches columns.

Proposition 4.6 (A property of a permutation matrix)**.**

$$\det(P) = \pm 1.$$

Proof. Interchanging two rows of a matrix changes the sign of the determinant (or determinant formula).
Note: permutation matrices are non-singular. □

Definition 4.7. P is an *elementary permutation matrix* iff P can be obtained from I (the identity) by interchanging 2 rows.
Note: we write P_{rs} to indicate that rows r and s are switched.

Proposition 4.8. *The following are properties of elementary permutation matrices:*

1) $P_{rs} = P_{rs}^T$

2) $P_{rs}^{-1} = P_{rs}$

Proposition 4.9. *Any permutation matrix P can be written as a product of elementary permutation matrices.*

Proof. See [1, p. 204] □

Definition 4.10. $A \in \mathbb{R}^{n\times n}$ is an *orthogonal matrix* iff

$$AA^T = I = A^T A$$

where I is the identity. Note: $A^T = A^{-1}$.

Proposition 4.11. *Any permutation matrix P is orthogonal.*

Proof. Assume $P = P_{ab}P_{cd}$, by Proposition 4.9. Then

$$P^T P = P_{cd}^T \underbrace{P_{ab}^T P_{ab}}_{I} P_{cd} = \underbrace{P_{cd}^T P_{cd}}_{} I = I.$$

Similar for more elementary matrices. □

Proposition 4.12. *A product of two permutation matrices is a permutation matrix.*

Proposition 4.13. *The following are properties of the determinant:*

1. $\det(AB) = \det(A)\det(B)$, *for* $A, B \in \mathbb{R}^{n\times n}$.
2. *Suppose U is upper triangular, that is*

$$U = \begin{bmatrix} u_{11} & & X \\ & \ddots & \\ 0 & & u_{nn} \end{bmatrix}$$

then

$$\det(U) = \prod_{i=1}^{n} u_{ii}.$$

The same it true for lower triangular matrices.
Note: $\det(L_k) = 1 = \det(L_k^{-1})$.

Block matrices.

Example 4.14.

$$AB = \overset{3}{3}\left\{\left[\begin{array}{c|c} \overbrace{A_{11}}^{2} & \overbrace{A_{12}}^{1} \\ \hline A_{21} & A_{22} \end{array}\right]\begin{matrix}2\\1\end{matrix}\right. \quad \overset{4}{3}\left\{\left[\begin{array}{c|c} \overbrace{B_{11}}^{2} & \overbrace{B_{12}}^{2} \\ \hline B_{21} & B_{22} \end{array}\right]\begin{matrix}2\\1\end{matrix}\right.$$

$$= \left[\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{array}\right]$$

Lecture 12

Theorem 4.15 (LU decomposition)**.** *(Theorem 8.6.1 in [1]) Every non-singular $A \in \mathbb{R}^{n\times n}$ can be factored as $PA = LU$, where P is a permutation matrix, L is unit lower triangular, and U is non-singular upper triangular.*

Proof. We use induction on n.

Base Case $n = 1$: trivial
$(1\cdot\alpha = 1\cdot\alpha, \quad \alpha \neq 0$, where $P = L = 1$, and $A = U = \alpha)$.

Induction Hypothesis Assume the statement is true for n, let $N = n + 1$.

Induction Step Consider $A \in \mathbb{R}^{N\times N}$. Find a non-zero pivot element in the first column of A:
$\exists a_{i1} \neq 0$ since A is non-singular, we use a_{i1} as a pivot. Then perform one step of Gaussian elimination:

$$L_1^{-1} P_1 A = A_1$$

with $L_1 = \left[\begin{array}{c|c} 1 & 0 \\ \hline m & I \end{array}\right]$, where $m = [m_{21}, \ldots, m_{N1}]^T$, P_1 is a permutation matrix that switches rows 1 and i, and

$$A_1 = \left[\begin{array}{c|c} u_{11} & u_1^T \\ \hline 0 & \tilde{A}_2 \end{array}\right] \qquad \text{with} \quad \begin{array}{ll} u_{11} & = a_{i1} \neq 0 \\ u_1^T & = [u_{12}, \ldots, u_{1N}] \end{array}$$

then

$$\begin{aligned} \det(A_1) &= \det(L_1^{-1})\det(P_1)\det(A) \\ &= 1 \cdot (-1) \cdot \det(A) \\ &= -\det(A) \neq 0 \quad \text{since } A \text{ is non-singular} \end{aligned}$$

Hence A_1 is non-singular, so

$$0 \neq \det(A_1) = \underbrace{u_{11}}_{\neq 0} \det(\tilde{A}_2) \implies \det(\tilde{A}_2) \neq 0,$$

meaning that $\tilde{A}_2$ is non-singular.
We have $P_1 A = L_1 A_1$. Now use the induction hypothesis on $\tilde{A}_2 \in \mathbb{R}^{n\times n}$, since $\tilde{A}_2$ is non-singular. Then

$$\tilde{P}_2 \tilde{A}_2 = \tilde{L}_2 \tilde{U}_2.$$

Decompose A_1: Let

$$P_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{P}_2 \end{array}\right], \quad \text{a permutation matrix,}$$

$$L_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{L}_2 \end{array}\right], \quad \text{a unit lower triangular matrix, and}$$

$$U = \left[\begin{array}{c|c} u_{11} & u_1^T \\ \hline 0 & \tilde{U}_2 \end{array}\right], \quad \text{an upper triangular matrix.}$$

Now putting everything together, we get:

$$\begin{aligned} P_1 A &= L_1 A_1 \\ P_1 A &= L_1 \underbrace{P_2^T P_2}_{I} A_1 \\ P_2 P_1 A &= P_2 L_1 P_2^T P_2 A_1 \\ \underbrace{P_2 P_1}_{P} A &= \underbrace{P_2 L_1 P_2^T L_2}_{L} U \\ \implies PA &= LU \end{aligned}$$

where

P is a permutation matrix,

L is unit lower triangular, and

U is a non-singular upper triangular matrix.

because

1) $P_2 P_1$ is a permutation matrix by Proposition 4.12
2) $L = P_2 L_1 P_2^T L_2$ is unit lower triangular:

$$\begin{aligned} P_2 L_1 P_2^T &= \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{P}_2 \end{array}\right] \left[\begin{array}{c|c} 1 & 0 \\ \hline m & I \end{array}\right] \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{P}_2^T \end{array}\right] \\ &= \left[\begin{array}{c|c} 1 & 0 \\ \hline \tilde{P}_2 m & \tilde{P}_2 \end{array}\right] \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{P}_2^T \end{array}\right] \\ &= \left[\begin{array}{c|c} 1 & 0 \\ \hline \tilde{P}_2 m & \tilde{P}_2 \tilde{P}_2^T \end{array}\right] \\ &= \left[\begin{array}{c|c} 1 & 0 \\ \hline \tilde{P}_2 m & I \end{array}\right] \end{aligned}$$

and so

$$\begin{aligned} L &= P_2 L_1 P_2^T L_2 \\ &= \left[\begin{array}{c|c} 1 & 0 \\ \hline \tilde{P}_2 m & I \end{array}\right] \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{L}_2 \end{array}\right] \\ &= \left[\begin{array}{c|c} 1 & 0 \\ \hline \tilde{P}_2 m & \tilde{L}_2 \end{array}\right], \end{aligned}$$

which is unit lower triangular.

3)

$$U = \left[\begin{array}{c|c} u_{11} & u_1^T \\ \hline 0 & \tilde{U}_2 \end{array}\right]$$

is upper triangular since $\tilde{U}_2$ is upper triangular, and it's non-singular since

$$0 \neq \det(U) = \underbrace{u_{11}}_{\neq 0} \det(\tilde{U}_2).$$

□

Note: we can use LU decomposition to compute $\det(A)$:

$$\begin{aligned} PA = LU \quad &\implies \quad \det(PA) = \det(LU) \\ &\implies \quad \det(P)\det(A) = \det(L)\det(U) \\ &\implies \quad (\pm 1) \cdot \det(A) = 1 \cdot \prod_{i=1}^{n} u_{ii} \end{aligned}$$

4.4 Vector and matrix norms [1, 8.10]

We would like to analyse sensitivity of x to perturbations in A and b, in $Ax = b$.

Definition 4.16 (Vector norm)**.** Let $\mathcal{V}$ be a finite dimensional vector space over $\mathbb{R}$. A *vector norm* $\|\cdot\|$ on $\mathcal{V}$, is a mapping $\mathcal{V} \mapsto \mathbb{R}$ that satisfies the following conditions for all $x, y \in \mathcal{V}$, and $\alpha \in \mathbb{R}$:

$$\begin{aligned} \|x\| &\geq 0 \\ \|x\| = 0 &\iff x = 0, \\ \|\alpha x\| &= |\alpha| \|x\|, \\ \|x + y\| &\leq \|x\| + \|y\|. \qquad \text{(triangle inequality)} \end{aligned}$$

For $\mathcal{V} = \mathbb{R}^n$, we study the following vector norms:

$$\begin{aligned} \|x\|_1 &:= \sum_{i=1}^{n} |x_i| && \text{to be the 1-norm,} \\ \|x\|_2 &:= \sqrt{\sum_{i=1}^{n} |x_i|^2} && \text{to be the 2-norm, and} \\ \|x\|_\infty &:= \max_{1 \leq i \leq n} \{|x_i|\} && \text{to be the infinity norm} \end{aligned}$$

where $x \in \mathbb{R}^n$.

Definition 4.17 (Induced matrix norms[8])**.** Let $\|\cdot\|$ be a vector norm. Then for $A \in \mathbb{R}^{n \times n}$, the *induced matrix norm* is given by

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

Note: we may write

$$\|A\| = \sup_{x \neq 0} \left\| A \frac{x}{\|x\|} \right\| = \sup_{\|z\|=1} \|Az\|$$

In particular we shall study the following induced matrix norms:

$$\|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \qquad (\text{for } p = 1, 2, \infty)$$

Note: induced matrix norm satisfies all conditions in Definition 4.16:

1. $\|A\|_p \geq 0$,
2. $\|A\|_p = 0 \iff A = 0$,

[8] Induced by the vector norms

3. $\|\alpha A\|_p = |\alpha| \|A\|_p$, and
4. $\|A + B\|_p \leq \|A\|_p + \|B\|_p$.

Other properties include:

5. $\|Ax\|_p \leq \|A\|_p \|x\|_p$ since $\frac{\|Ay\|_p}{\|y\|_p} \leq \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$, $\forall y \neq 0$ with equality when $y = 0$.
6. $\|AB\|_p \leq \|A\|_p \|B\|_p$.

Definition 4.18. $A \in \mathbb{R}^{n \times n}$ is *symmetric positive definite* (SPD) iff $A = A^T$ and $x^T A x > 0$, $\forall x \neq 0$.

Definition 4.19. $A \in \mathbb{R}^{n \times n}$ is *symmetric positive semi-definite* (SPSD) iff $A = A^T$ and $x^T A x \geq 0$, $\forall x \neq 0$.

Proposition 4.20 (Properties)**.** *Take $A \in \mathbb{R}^{n \times n}$, then*

1. *$A = A^T \implies$ A has real eigenvalues and a complete set of n orthogonal eigenvectors.*
2. *A is SPD $\implies \lambda_i > 0$, $\forall i \in \{1, \ldots, n\}$.*
3. *A is SPSD $\implies \lambda_i \geq 0$, $\forall i \in \{1, \ldots, n\}$.*
4. *A is SPD $\implies a_{ii} > 0$, $\forall i \in \{1, \ldots, n\}$, which means $\max_{ij} |a_{ij}| = \max_k a_{kk}$ (See Lemma 8.7.1, [1, p. 214]).*

Lemma 4.21 (Lemma 8.10.5 in [1])**.**

$$\|A\|_\infty = \max_{1 \leq i \leq n} \left(\sum_{j=1}^{n} |a_{ij}| \right) \quad \text{"maximum absolute row sum"}$$

Proof. Recall:

$$\|A\|_\infty = \sup_{\|x\|_\infty = 1} \|Ax\|_\infty.$$

$$\|x\|_\infty = 1 \iff \max_{1 \leq i \leq n} |x_i| = 1.$$

Let

$$r = \max_{1 \leq i \leq n} \left(\sum_{j=1}^{n} |a_{ij}| \right) \quad \text{(largest absolute row sum)}.$$

Let $\|x\|_\infty = 1$, then $\|Ax\|_\infty \leq r$ since

$$|(Ax)_i| = \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \sum_{j=1}^{n} |a_{ij}||x_j| \leq \sum_{j=1}^{n} |a_{ij}| \leq r.$$

Now it's sufficient to find $\hat{x}$ s.t. $\|A\hat{x}\|_\infty = r$ and $\|\hat{x}\|_\infty = 1$. Let ν be the index of the row in A with the maximum absolute row sum, meaning that

$$\sum_{j=1}^{n} |a_{\nu j}| = r.$$

Define $\hat{x}$ as follows:

$$\hat{x}_j := \operatorname{sgn}(a_{\nu j}) = \begin{cases} 1 & \text{if } a_{\nu j} > 0 \\ 0 & \text{if } a_{\nu j} = 0 \\ -1 & \text{if } a_{\nu j} < 0 \end{cases}$$

Note that

$$|(A\hat{x})_\nu| = \left| \sum_{j=1}^{n} a_{\nu j} \hat{x}_j \right| = \sum_{j=1}^{n} |a_{\nu j}| = r.$$

Therefore $\|A\hat{x}\|_\infty = r$ and so $\|A\|_\infty = r$. □

Lemma 4.22.

$$\|A\|_1 = \max_{1 \leq j \leq n} \left(\sum_{i=1}^{n} |a_{ij}| \right) \quad \text{"max absolute column sum"}$$

Proof. Assignment 4 □

Lecture 13

Proposition 4.23. *If $A \in \mathbb{R}^{n \times n}$, then $A^T A$ is SPSD.*

Proof. Let $\lambda_i(A^T A)$ denote the i^{th} eigenvalue of $A^T A$.

$$(A^T A)^T = A^T A \implies \lambda_i(A^T A) \text{ are real}$$

and

$$x^T A^T A x = (Ax)^T (Ax) = \|Ax\|_2^2 \geq 0 \text{ for any } x,$$

so $\lambda_i(A^T A) \geq 0$, $\forall i$. □

Definition 4.24. Let $A \in \mathbb{R}^{n \times n}$, where $\lambda_i(A)$ can be complex. Then

$$\rho(A) := \max_{1 \leq i \leq n} |\lambda_i(A)|$$

is called the *spectral radius of A*.

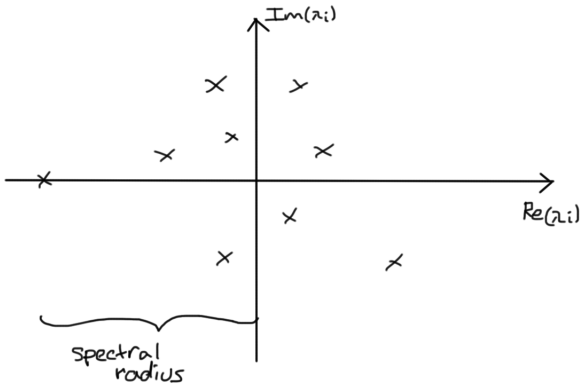


Figure 4.1: Spectral radius.

2-norm.

$$\begin{aligned} \|A\|_2 &= \max_{1 \leq i \leq n} \sqrt{\lambda_i(A^T A)} \\ &= \max_{1 \leq i \leq n} \sqrt{\lambda_i(A A^T)} \\ &= \max_{1 \leq i \leq n} \sigma_i \text{ where } \sigma_i \text{ are the singular values of } A \end{aligned}$$

Special case: if A is symmetric (i.e. $A = A^T$, so real eigenvalues), then

$$\begin{aligned}\|A\|_2 &= \max_{1\le i\le n} \sqrt{\lambda_i(A^2)} \\ &= \max_{1\le i\le n} \sqrt{(\lambda_i(A))^2} \\ &= \max_{1\le i\le n} |\lambda_i(A)| = \rho(A)\end{aligned}$$

Example 4.25.

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$$

$$A = \begin{bmatrix}3 & 1\\ 1 & 3\end{bmatrix}, \qquad \underbrace{\lambda_{1,2} = 2, 4}_{\text{eigenvalues}} \qquad \underbrace{x_1 = \begin{bmatrix}1\\-1\end{bmatrix}, \quad x_2 = \begin{bmatrix}1\\1\end{bmatrix}}_{\text{eigenvectors}}$$

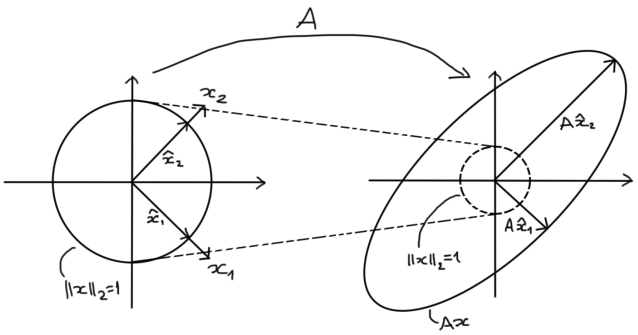


Figure 4.2: A stretches and rotates. $\|A\|_2 = 4$.

4.5 *Sensitivity* and *conditioning* of the problem $Ax = b$ [1, 8.11]

Definition 4.26. $Ax = b$ is a *well-conditioned* problem iff small perturbations[9] in A or b always give small perturbations in x. Otherwise we say $Ax = b$ is an *ill-conditioned* problem, which means small perturbations in A or b may give large perturbations in x.

So we have two cases:

1) Perturbation in b only: $A(x + \delta x) = b + \delta b$, where x is the exact solution of the unperturbed problem $Ax = b$. What is the relative error $\|\delta x\|/\|x\|$ compared to $\|\delta b\|/\|b\|$? We have:

$$\begin{aligned}&A\delta x = \delta b \\ \implies &\delta x = A^{-1}\delta b \\ \implies &\|\delta x\| \le \|A^{-1}\|\|\delta b\|\end{aligned}$$

Use $\|b\| = \|Ax\| \le \|A\|\|x\|$ or $\|x\|^{-1} \le \|A\|\|b\|^{-1}$ to get

$$\frac{\|\delta x\|}{\|x\|} \le \|A^{-1}\|\|\delta b\|\frac{\|A\|}{\|b\|}$$

$$\text{or } \frac{\|\delta x\|}{\|x\|} \le \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

[9] For example rounding errors.

Definition 4.27. The *condition number* of A is given by

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Note that $\kappa(A) = \|A\|\|A^{-1}\| \ge 1$, because:

$$1 = \|I\| = \|AA^{-1}\| \le \|A\|\|A^{-1}\| = \kappa(A).$$

If $\kappa(A) \gg 1$ then problem $Ax = b$ is *ill-conditioned*.
If $\kappa(A) \approx 1$ then problem $Ax = b$ is *well-conditioned*.
Note:

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} = \frac{|\lambda|_{max}}{|\lambda|_{min}}$$

where the last equality holds if $A = A^T$ since

$$Ax = \lambda x \implies \frac{1}{\lambda}x = A^{-1}x \quad \text{if } \lambda \ne 0.$$

Note that conditioning is a property of the problem $Ax = b$, and is independent of the algorithm.

2) Perturbations in both A and b:

Lemma 4.28 (Lemma 8.10.6 in [1])**.** *If there is a p such that $\|F\|_p < 1$, then $I + F$ is non-singular.*

Proof. Assume $I + F$ is singular. Then $(I + F)x = 0$ for some $x \ne 0$. So

$$\begin{aligned}&\|x\|_p = \| - Fx\|_p \le \|F\|_p\|x\|_p \\ \implies \quad &\|x\|_p < \|x\|_p \implies \text{a contradiction!}\end{aligned}$$

□

Theorem 4.29 (Theorem 8.11.2 in [1])**.** *Let $Ax = b$, $A \in \mathbb{R}^{n\times n}$ non-singular. Consider*

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

Suppose $\tau = \kappa(A)\frac{\|\delta A\|}{\|A\|} < 1$ (Note: $\tau = \|A^{-1}\|\|\delta A\|$). then $A + \delta A$ is non-singular and

$$\frac{\|\delta x\|}{\|x\|} \le \frac{\kappa(A)}{1-\tau}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right)$$

Proof. First, we show that $A + \delta A$ is non-singular. Rewrite $A + \delta A = A(I + F)$ with $F = A^{-1}\delta A$. Then

$$\|F\| = \|A^{-1}\delta A\| \le \|A^{-1}\|\|\delta A\| = \tau < 1.$$

Which implies that $I + F$ is non-singular by Lemma 4.28. Therefore $A + \delta A$ is non-singular since

$$\det(A + \delta A) = \underbrace{\det(A)}_{\ne 0}\underbrace{\det(I + F)}_{\ne 0}$$

Now we derive the bound:

$$\begin{aligned}
(A+\delta A)(x+\delta x) &= b+\delta b \\
A\delta x &= \delta b - \delta A(x+\delta x) \\
\delta x &= A^{-1}\delta b - A^{-1}\delta A(x+\delta x) \\
\|\delta x\| &\le \|A^{-1}\|\|\delta b\| + \|A^{-1}\|\|\delta A\|\|x+\delta x\| \\
\|\delta x\| &\le \|A^{-1}\|\|\delta b\| + \tau(\|x\| + \|\delta x\|) \\
(1-\tau)\|\delta x\| &\le \|A^{-1}\|\|\delta b\| + \tau\|x\| \\
\frac{\|\delta x\|}{\|x\|} &\le \frac{1}{1-\tau}\left(\|A^{-1}\|\frac{\|\delta b\|}{\|x\|} + \tau\right) \\
\frac{\|\delta x\|}{\|x\|} &\le \frac{1}{1-\tau}\left(\|A^{-1}\|\|A\|\frac{\|\delta b\|}{\|b\|} + \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}\right) \\
\frac{\|\delta x\|}{\|x\|} &\le \frac{\kappa(A)}{1-\tau}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right)
\end{aligned}$$

□

Note: these are the upper bounds ("worst case"). There exist certain matrices with very large condition numbers ($\kappa(A)$) for which there are algorithms that can compute the solution very accurately[10]. However, most matrix problems where $\kappa(A)$ is very large cannot be solved accurately with any algorithm.

Lecture 14

4.6 Stability of Gaussian Elimination and LU Decomposition for $Ax = b$ [1, 8.4, 8.12]

Example 4.30. Consider the problem $Ax = b$, with

$$A = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad 0 < \varepsilon \ll 1.$$

$$A^{-1} = \frac{1}{\varepsilon - 1}\begin{bmatrix} 1 & -1 \\ -1 & \varepsilon \end{bmatrix}$$

$$x = A^{-1}b = \frac{1}{\varepsilon-1}\begin{bmatrix} -1 \\ -1+2\varepsilon \end{bmatrix} = \begin{bmatrix} \frac{1}{\varepsilon-1} \\ \frac{1-2\varepsilon}{1-\varepsilon} \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty = 2\cdot\frac{2}{1-\varepsilon} \approx 4 \implies$ well-conditioned

4.6.1 Gaussian Elimination

Consider the following:

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \to \begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix} \qquad m_{21} = \frac{1}{\varepsilon}.$$

Then

$$A = LU, \qquad L = \begin{bmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix}$$

$$L^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{\varepsilon} & 1 \end{bmatrix}$$

[10] For example sparse discretizations of Poisson differential equations

Example 4.31. Take $\varepsilon = 10^{-5}$, and $m_{21} = 10^5$. Consider the floating point system:

$$(\beta = 10, t = 3, L = -10, U = 10)$$

Then we have

$$\left[\begin{array}{cc|c} 10^{-5} & 1 & 1 \\ 0 & 1 - 10^5 & 2 - 10^5 \end{array}\right]$$

$$\begin{aligned} fl(1 - 10^5) &= fl(-99999) \\ &= -1 \times 10^5 \end{aligned} \qquad \begin{aligned} fl(2 - 10^5) &= fl(-99998) \\ &= -1 \times 10^5 \end{aligned}$$

Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, then

$$\left[\begin{array}{cc|c} 10^{-5} & 1 & 1 \\ 0 & -10^5 & -10^5 \end{array}\right] \to x = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{array}{l} \leftarrow \text{the first component} \\ \quad \text{should be close to 1} \end{array}$$

This problem is well-conditioned, but the algorithm finds a highly inaccurate result, so the algorithm is unstable.

Why is it unstable? Due to a poor choice of pivot element, some steps become ill-conditioned problems, which cause the algorithm to be unstable. Note:

$$\|L\|_\infty \approx \frac{1}{\varepsilon} = 10^5, \qquad \|L^{-1}\|_\infty \approx \frac{1}{\varepsilon} = 10^5$$

$$\kappa(L) \approx 10^{10} \qquad \text{(ill-conditioned)}$$

Similarly $\kappa(U) \approx 10^{10} \implies$ ill-conditioned steps in the algorithm.

Stability is a problem for the algorithm, not the problem itself.
Conditioning, on the other hand, refers to the problem. Note that the choice of pivot element creates very large numbers in the transformed A (i.e. U).

4.6.2 Gaussian Elimination with Partial Pivoting

We modify the original Gaussian Elimination algorithm. In every step we will switch rows so that the largest element (in absolute value) in the column is chosen as the pivot.

Example 4.32.

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \to \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1-\varepsilon & 1-2\varepsilon \end{array}\right], \qquad \begin{aligned} fl(1-\varepsilon) &= 1 \\ fl(1-2\varepsilon) &= 1 \end{aligned}$$

We have $x_2 = 2$, and $x_1 = 1$.

$$L = \begin{bmatrix} 1 & 0 \\ \varepsilon & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{bmatrix}$$

$$\kappa_\infty(L) \approx 1, \kappa_\infty(U) \approx 4, \quad \text{(better conditioning)}$$

Gaussian Elimination with partial pivoting is a stable algorithm.
For full pivoting, select largest pivot element from remaining rows and columns (more work, but not much more stable).

Theorem 4.33 (Theorem 8.12.1 in [1]). *Consider the problem $Ax = b$. Let $\hat{x}$ be the solution given by Gaussian Elimination with partial pivoting. Let $\tau = \mu(n^3 + 3n^2)g_n\kappa(A)$, where*

$$g_n = \frac{\max_{i,j,k} |\hat{a}_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|} \qquad \text{"growth factor"}$$

where $\hat{a}_{ij}^{(k)}$ are the elements of the transformed A at kth step of Gaussian Elimination, and a_{ij} are the elements of the original A. Then

$$\frac{\|\hat{x} - x\|_\infty}{\|x\|_\infty} \leq \frac{\tau}{1 - \tau}$$

Proof. No proof. □

Notes:

- $\tau \ll 1$ is good ("stable" result)
- τ can be large if any one of the following holds:
  - $\kappa(A)$ is large
  - n is large
  - g_n is large
- There are more stable algorithms for $Ax = b$ than LU decomposition. For example QR decomposition (no growth in matrix elements).

5 QR decomposition

5.1 QR decomposition of a square matrix $A \in \mathbb{R}^{n\times n}$

Definition 5.1. $Q \in \mathbb{R}^{n\times n}$ is orthogonal iff $Q^TQ = I = QQ^T$.

Proposition 5.2. *Orthogonal matrices preserve Euclidean length (2-norm).*

Proof. Let $Q^TQ = O$. Suppose $y = Qx$. Then

$$\begin{aligned} \|y\|_2 &= \|Qx\|_2 \\ &= \sqrt{(Qx)^TQx} \\ &= \sqrt{x^TQ^TQx} \\ &= \sqrt{x^Tx} \\ &= \|x\|_2 \end{aligned}$$

□

(From now on we shall use only 2-norms).

Proposition 5.3. *Product of orthogonal matrices is orthogonal.*

Proof. Let $Q = Q_1Q_2$, where $Q_1,\ Q_2$ are orthogonal. Then

$$Q^TQ = (Q_1Q_2)^TQ_1Q_2 = Q_2^TQ_1^TQ_1Q_2 = I$$

Note $Q^TQ = I$ is sufficient because

$$\det(Q^T)\det(Q) = \det(I) = 1$$
$$\det(Q) = \pm 1 \ \text{ thus } \ Q^T = Q^{-1}$$

and so $QQ^T = QQ^{-1} = I$. □

Consider solving $Ax = b$. If we use LU decomposition (with partial pivoting) then matrix elements may grow:

$$A = \begin{bmatrix} & \\ & \end{bmatrix} \to \begin{bmatrix} X & X \\ 0 & X \end{bmatrix} = L_1A$$

One idea is to use an orthogonal transformation matrix to create the zeros:

$$A = \begin{bmatrix} & \\ & \end{bmatrix} \to A_2 = Q_1A = \begin{bmatrix} X & X \\ 0 & X \end{bmatrix}$$

where Q_1 is an orthogonal matrix. An advantage is that the 2-norms of the columns of A_2 are the same as for A (no uncontrolled growth in matrix elements). This is somewhat more stable than LU, however it requires more work than LU. Main advantage is in least-squares problems.

Theorem 5.4 (QR factorization of $A \in \mathbb{R}^{n\times n}$). *Let $A \in \mathbb{R}^{n\times n}$, then $\exists Q \in \mathbb{R}^{n\times n}$, orthogonal and $R \in \mathbb{R}^{n\times n}$ upper triangular such that $A = QR$.*
Note: to solve $Ax = b$, we solve $QRx = b \implies Rx = Q^Tb$ by back substitution.

How to build Q and R:

1. Gram-Shmidt orthogonalization of the columns of A.

Example 5.5.

$$A = \left[\ \vec{a}_1 \mid \vec{a}_2 \mid \vec{a}_3\ \right]$$

$$\vec{t}_1 = \vec{a}_1 \qquad \to \quad \vec{q}_1 = \frac{\vec{t}_1}{\|\vec{t}_1\|}$$

$$\vec{t}_2 = \vec{a}_2 - \frac{(\vec{a}_2, \vec{t}_1)}{(\vec{t}_1, \vec{t}_1)}\vec{t}_1 \qquad \to \quad \vec{q}_2 = \frac{\vec{t}_2}{\|\vec{t}_2\|}$$

$$\vec{t}_3 = \vec{a}_3 - \frac{(\vec{a}_3, \vec{t}_1)}{(\vec{t}_1, \vec{t}_1)}\vec{t}_1 - \frac{(\vec{a}_3, \vec{t}_2)}{(\vec{t}_2, \vec{t}_2)}\vec{t}_2 \qquad \to \quad \vec{q}_3 = \frac{\vec{t}_3}{\|\vec{t}_3\|}$$

So we get

$$\begin{aligned} \vec{a}_1 &= r_{11}\vec{q}_1 \\ \vec{a}_2 &= r_{12}\vec{q}_1 + r_{22}\vec{q}_2 \\ \vec{a}_3 &= r_{13}\vec{q}_1 + r_{23}\vec{q}_2 + r_{33}\vec{a}_3 \end{aligned}$$

$$\underbrace{\left[\ \vec{a}_1 \mid \vec{a}_2 \mid \vec{a}_3\ \right]}_{A} = \underbrace{\left[\ \vec{q}_1 \mid \vec{q}_2 \mid \vec{q}_3\ \right]}_{Q} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}}_{R}$$

$$A = QR, \qquad (Q^TQ = I)$$

For large n, Gram-Schmidt itself is numerically unstable (due to rounding, cancellation → lose orthogonality). We need a better algorithm for QR: *Householder reflections.*

Lecture 15

2. Householder reflections:

$$A = [\vec{a}_1 | \dots | \vec{a}_n] \implies Q_1 A = \left[\begin{array}{c|c} \pm\|\vec{a}_1\| & r_1^T \\ \hline 0 & \tilde{A}_2 \end{array}\right]$$

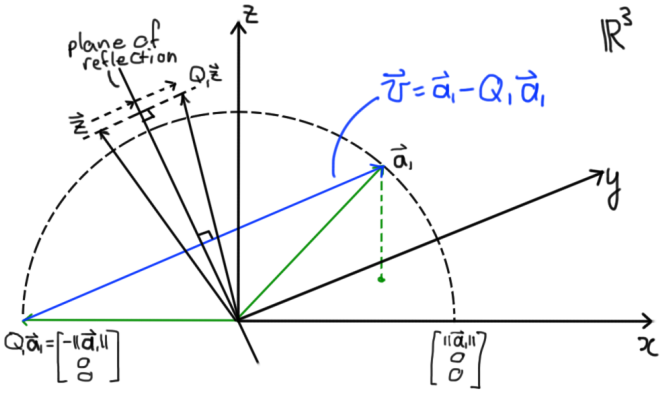


Figure 5.1: Householder reflection.

Let $\vec{u} = \frac{\vec{v}}{\|\vec{v}\|}$, it's called the Householder vector for this reflection. $\vec{u}$ is a unit vector. For any vector $\vec{z} \in \mathbb{R}^n$:

$$Q_1\vec{z} = \vec{z} - 2(\vec{u}^T\vec{z})\vec{u}$$
$$\text{or } Q_1 = I - 2\vec{u}\vec{u}^T$$

Note: $Q_1^T = Q_1$. Also Q_1 is orthogonal.

$$\begin{aligned} Q_1^T Q_1 &= (I - 2\vec{u}\vec{u}^T)(I - 2\vec{u}\vec{u}^T) \\ &= (I - 4\vec{u}\vec{u}^T + 4\vec{u}\vec{u}^T\vec{u}\vec{u}^T) \\ &= I \end{aligned}$$

The algorithm:

Step 1: $A_2 = Q_1 A = \left[\begin{array}{c|c} r_{11} & r_1^T \\ \hline 0 & \tilde{A}_2 \end{array}\right]$.

Step 2: $\tilde{A}_3 = \tilde{Q}_2\tilde{A}_2 = \left[\begin{array}{c|c} r_{22} & r_2^T \\ \hline 0 & \tilde{A}_3 \end{array}\right]$,

$$\text{or } Q_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{Q}_2 \end{array}\right].$$

$$A_3 = Q_2 A_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{Q}_2 \end{array}\right]\left[\begin{array}{c|c} r_1 & r_1^T \\ \hline 0 & \tilde{A}_2 \end{array}\right] = \left[\begin{array}{c|c} r_{11} & r_1^T \\ \hline 0 & \tilde{Q}_2\tilde{A}_2 \end{array}\right]$$

$$\text{or } A_3 = \left[\begin{array}{c|cc} r_{11} & r_1^T & \\ \hline 0 & r_{22} & r_2^T \\ & 0 & \tilde{A}_3 \end{array}\right] = Q_2 Q_1 A$$

do n steps like this

$$R = Q^T A \text{ with } Q^T = Q_n Q_{n-1} \dots Q_2 Q_1$$
$$\text{or } QR = A \text{ with } Q = Q_1^T Q_2^T \dots Q_{n-1}^T Q_n^T$$
$$\implies Q = Q_1 Q_2 \dots Q_{n-1} Q_n$$

Note: for numerical stability, choose $\pm\|\vec{a}_1\|$ s.t. its sign is opposite to the sign of $(\vec{a}_1)_1$ to avoid catastrophic cancellation in

$$\vec{v} = \vec{a}_1 - Q_1\vec{a}_1$$

Note: $W = \frac{4}{3}n^3 + \mathcal{O}(n^2)$ (double of LU). QR is more work than LU but is somewhat more stable (no growth in matrix elements).
Note: How to compute Q in practice two possibilities:

1) Multiply Q_i's iteratively:

$$Q^T = Q_n Q_{n-1} \dots \underbrace{Q_2 \underbrace{Q_1 I}}$$

Problem: $\tilde{Q}_i$ has to be applied to all columns. (expensive)

2) Store $\vec{u}$ vectors.

$$Q = Q_1 Q_2 \dots \underbrace{Q_{n-1} \underbrace{Q_n I}}$$

advantage: there are still leading columns with zeros in each step. (less work)

3) (detail) Store the $\vec{u}$ vectors.

5.2 QR decomposition of a rectangular matrix $A \in \mathbb{R}^{m\times n}$

Assume $m \geq n$. Then $A = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$

$$m\left[\begin{array}{c} \\ \\ \\ \end{array}\right]^{n} = m\left[\begin{array}{c|c} \hat{Q} & \bar{Q} \\ n & m-n \end{array}\right]\begin{bmatrix} R \\ 0 \end{bmatrix}^{\begin{matrix} n \\ m-n \end{matrix}}$$

$\hat{Q}$: orthogonalization of the n columns of A (in the general case).
$\bar{Q}$: more orthogonal columns, to complete the orthogonal basis of $\mathbb{R}^m$. Note: Householder algorithm also works for this case (n steps) Note "thin form" QR decomposition of A:

$$A = \hat{Q}R \qquad \hat{Q} = m\left[\begin{array}{c} \\ \\ \\ \end{array}\right]^{n}, \qquad R = n\left[\begin{array}{cc} & \\ 0 & \end{array}\right]^{X}$$

Amount of work for "thin form" QR using Householder, $A \in \mathbb{R}^{m\times n}$. Only count work to form R: n orthogonal transformations of form $(k = 1, \dots, n)$.

$$\tilde{Q}_i\tilde{A}_i = (I - 2\vec{u}_i\vec{u}_i^T)\tilde{A}_i$$

is the operation which dominates the cost of computation. Note that $\vec{u}_i\vec{u}_i^T \in \mathbb{R}^{m-k+1}$ and $\tilde{A}_i \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$.
first compute $\vec{u}_i^T\tilde{A}_i$: $\tilde{A}_i$ has $n-k+1$ columns:

$$(n-k+1)(m-k+1) \text{ multiplications}$$
$$(n-k+1)(m-k) \text{ additions}$$

Then compute $\vec{u}_i(\vec{u}_i^T \tilde{A}_i) \to (m-k+1)(n-k+1)$ multiplications.
Then $\tilde{A}_i - 2\vec{u}_i(\vec{u}_i^T \tilde{A}_i)$: $(m-k+1)(n-k+1)$ additions[11].

$$W = 2mn^2 - \frac{2}{3}n^3 + \text{lower order terms}$$

Lecture 16

5.3 Least-squares solution using QR [1, 8.14, 8.15]

Consider $Ax = b$, $A \in \mathbb{R}^{m\times n}$, where $m \geq n$. This is called an overdetermined system. For example:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad \begin{matrix} 3 \text{ euqations} \\ 2 \text{ unknowns} \end{matrix}$$

$\implies$ usually no exact solution (overdetermined).

Definition 5.6 (Least squares problem (LS))**.** Find $\hat{x} \in \mathbb{R}^n$ such that $\|b - A\hat{x}\|_2$ is minimal.

5.3.1 Geometric interpretation

Definition 5.7. The *residual* is $r := b - Ax$ (for some x)

Definition 5.8. Define the *column space* of A by

$$\mathcal{C}(A) := \{A\vec{x} : \text{ for } x \in \mathbb{R}^n\}$$

These are all linear combinations of the columns of A.

Least-Squares (LS): find $A\hat{x}$ in the column space of A s.t. the residual r has minimal 2-norm. This is achieved by $r \perp \mathcal{C}(A)$. (residual orthogonal to the column space). Or $A\hat{x}$ is the orthogonal projection of b onto $\mathcal{C}(A)$. How to find $\hat{x}$?

$$\begin{aligned} r \perp Ax \ \forall x &\iff (r, Ax) = 0 \ \forall x \\ &\iff (b - A\hat{x}, Ax) = 0 \ \forall x \\ &\iff (b - A\hat{x})^T Ax = 0 \ \forall x \\ &\iff (b - A\hat{x})^T A = 0 \\ &\iff A^T(b - A\hat{x}) = 0 \\ &\iff A^T A\hat{x} = A^T b \qquad (\diamond) \end{aligned}$$

where $A^T A \in \mathbb{R}^{n\times n}$. Note that equation ($\diamond$) gives the *normal equations*, the first way to compute the LS solution. One problem in this approach is that $A^T A$ can be ill-conditioned, even more so than A.

Theorem 5.9. *Let $A \in \mathbb{R}^{m\times n}$, $m \geq n$.*

1) x is a global minimizer of $\|Ax - b\|_2 \iff$ x satisfies the normal equations $A^T Ax = A^T b$.

2) If the columns of A are linearly independent, then there exists a unique minimizer.

Proof. 1) x is a global minimum

$$\begin{aligned} &\iff \|A(x+y) - b\|_2^2 \geq \|Ax - b\|_2^2 \ \forall y \\ &\iff (Ax - b, Ax - b) + 2(Ax - b, Ay) + (Ay, Ay) \\ &\qquad \geq (Ax - b, Ax - b) \ \forall y \\ &\iff 2(A^T Ax - A^T b, y) + (Ay, Ay) \geq 0 \ \forall y \qquad (\spadesuit) \\ &\iff A^T Ax = A^T b \end{aligned}$$

We prove the last implication (" $\implies$ ") rigorously: Assume $g = A^T Ax - A^T b \neq 0$. Choose $y = -\varepsilon g$, $\varepsilon > 0$. Then we need ($\spadesuit$) to hold $\forall \varepsilon > 0$. Observe that ($\spadesuit$) becomes:

$$-\varepsilon(g, g) + \varepsilon^2(Ag, Ag) \geq 0 \qquad \forall \varepsilon$$

Suppose $Ag \neq 0$, then

$$\varepsilon \geq \frac{2(g, g)}{(Ag, Ag)} \quad \forall \varepsilon$$

which is a contradiction since ($\spadesuit$) must hold for all $\varepsilon > 0$. Furthermore, if $Ag = 0$, then ($\spadesuit$) becomes $-\varepsilon(g, g) \geq 0$, which is also a contradiction, since $\varepsilon > 0$ and $(g, g) \geq 0$.

2) Columns of A are linearly independent.
$\implies$ $Ax \neq 0$ if $x \neq 0$
$\implies$ $\|Ax\|_2^2 = (Ax)^T Ax = x^T A^T Ax > 0$ if $x \neq 0$.
$\implies$ $A^T A$ is SPD.
$\implies$ $A^T A$ is non-singular.
$\implies$ $A^T Ax = A^T b$ has a unique solution.

□

LS using QR. $Ax = b$, $A \in \mathbb{R}^{m\times n}$, $m \geq n$. $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$. Minimize $\|r\|_2 = \|b - Ax\|_2$. Let $A = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$, with $Q = \begin{bmatrix} \hat{Q} | \bar{Q} \end{bmatrix} \in \mathbb{R}^{m\times m}$ and $\hat{Q} \in \mathbb{R}^{m\times n}$. Then

$$\begin{aligned} \|r\|_2^2 &= \|Q^T r\|_2^2 \\ &= \left\| Q^T \left(b - Q \begin{bmatrix} R \\ 0 \end{bmatrix} x \right) \right\|_2^2 = \left\| \begin{bmatrix} \hat{Q}^T b \\ \bar{Q}^T b \end{bmatrix} - \begin{bmatrix} Rx \\ 0 \end{bmatrix} \right\|_2^2 \\ &= \|\hat{Q}^T b - Rx\|_2^2 + \underbrace{\|\bar{Q}^T b\|_2^2}_{\text{indep. of } x}. \end{aligned}$$

Thus $\|r\|_2^2$ is minimal when $\hat{Q}^T b - Rx = 0$ or $Rx = \hat{Q}^T b$. We solve the system by backward substitution to find $\hat{x}$. Determine $\hat{Q}$ and R using Householder algorithm. This is numerically more stable than solving normal equations.

6 Basic iterative methods for $Ax = b$ [2], [3]

Context problem: $Ax = b$ with $A \in \mathbb{R}^{n\times n}$. So far we have two direct methods: LU and QR, each having $W \in \mathcal{O}(n^3) \to n$ steps, $\mathcal{O}(n^2)$ work per step.
Alternatively we may use iterative methods. These have $\ll n$

[11] Note that we don't count multiplication by 2 because we can just increase the exponent in the floating point representation by 1

iterations with $\ll n^2$ work per iteration. These give us approximate solutions, and we stop after a small number of iterations, as soon as desired accuracy is reached. Iterative methods are useful for large, sparse matrices[12].

Lecture 17

6.1 Diagonal dominance

Definition 6.1. $A \in \mathbb{R}^{n\times n}$ is (strictly) row diagonally dominant iff

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}| \quad \forall i$$

Theorem 6.2 (Gershgorin circles)**.** *Let $A \in \mathbb{R}^{n\times n}$. Let $\Sigma(A)$ be the spectrum of A. Let D_i be the closed discs in the complex plane with centres at a_{ii} and radii $r_i = \sum_{j\neq i} |a_{ij}|$ for $i = 1, \ldots, n$. Then*

$$\Sigma(A) \subset \bigcup_{i=1}^{n} D_i$$

Proof. Let λ_i, x be an eigenvalue-eigenvector pair of A:

$$Ax = \lambda x \quad (x \neq 0).$$

Let i be an index where $|x_i|$ is maximal: $|x_i| = \|x\|_\infty$. So we know $x_i \neq 0$ since it is the maximal (in absolute value) element of an eigenvector (so $x \neq 0$). We know that $\sum_{j=1}^{n} a_{ij}x_j = \lambda x_i$, so it also follows that

$$\sum_{j\neq i} a_{ij}x_j = (\lambda - a_{ii})x_i$$

and thus we have

$$|\lambda - a_{ii}| \leq \left|\sum_{j\neq i} a_{ij}\frac{x_j}{x_i}\right| \leq \sum_{j\neq i} |a_{ij}| \left|\frac{x_j}{x_i}\right| \leq \sum_{j\neq i} |a_{ij}|,$$

which yields $\lambda \in D_i$. Applying the same reasoning to all eigenvalue-eigenvector pairs gives

$$\Sigma(A) \subset \bigcup_{i=1}^{n} D_i.$$

□

Theorem 6.3. *If $A \in \mathbb{R}^{n\times n}$ is strictly diagonally dominant, then A is non-singular.*

Proof. We know $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$ for all i. Therefore, non of the Gershgorin discs contain the origin. Thus $0 \notin \Sigma(A)$, so A is non-singular. □

[12]For example in the discretization of partial differential equations.

6.2 Jacobi and Gauss-Seidel iterative methods

6.2.1 Jacobi iterative method

Given a problem $Ax = b$ with $x, b \in \mathbb{R}^3$, we compute

$$\begin{aligned} a_{11}x_1^{new} + a_{12}x_2^{old} + a_{13}x_3^{old} &= b_1 \\ a_{21}x_1^{old} + a_{22}x_2^{new} + a_{23}x_3^{old} &= b_2 \\ a_{31}x_1^{old} + a_{32}x_2^{old} + a_{33}x_3^{new} &= b_3 \end{aligned}$$

In general we have

$$x_i^{new} = \frac{1}{a_{ii}}(b_i - \sum_{j\neq i} a_{ij}x_j^{old})$$

Notes:

1. $a_{ii} \neq 0$ is required for all i
2. all x_i^{new} can be computed independently from each other (good for parallel computing)
3. convergence is not obvious, it depends on properties of the matrix

6.2.2 Gauss-Seidel iterative method

Given a problem $Ax = b$ with $x, b \in \mathbb{R}^3$, we compute

$$\begin{aligned} a_{11}x_1^{new} + a_{12}x_2^{old} + a_{13}x_3^{old} &= b_1 \\ a_{21}x_1^{new} + a_{22}x_2^{new} + a_{23}x_3^{old} &= b_2 \\ a_{31}x_1^{new} + a_{32}x_2^{new} + a_{33}x_3^{new} &= b_3 \end{aligned}$$

In general we have

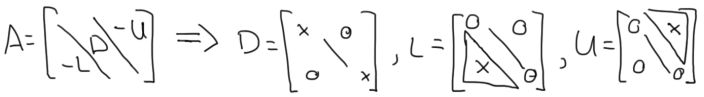
$$x_i^{new} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{new} - \sum_{j=i+1}^{n} a_{ij}x_j^{old}\right)$$

Notes:

1. If it converges, GS will often converge faster than Jacobi
2. Sequential (not good for parallel computing)
3. Other orders are possible too

6.2.3 In matrix form

Let $A = D - L - U$:



Jacobi: $x^{(k+1)} = D^{-1}(b + (L + U)x^{(k)})$ (k is the iteration number). So we get

$$\begin{aligned} Ax &= b \\ (D - L - U)x &= b \\ Dx &= b + (L + U)x \\ \implies Dx^{(k+1)} &= b + (L + U)x^{(k)} \end{aligned}$$

Gauss-Seidel: $x^{(k+1)} = (D-L)^{-1}(b+Ux^{(k)})$. So we get

$$\begin{aligned} Ax &= b \\ (D-L-U)x &= b \\ (D-L)x &= b+Ux \\ \implies x^{(k+1)} &= (D-L)^{-1}(b+Ux^{(k)}) \end{aligned}$$

6.2.4 Convergence theorems

For $Ax = b$, with $\det(A) \neq 0$.

Theorem 6.4. *If A is SPD, then GS converges to the unique solution of $Ax = b$, for any initial guess $x^{(0)}$.*

Theorem 6.5. *If A is strictly diagonally dominant, then GS and Jacobi converge to the unique solution of $Ax = b$, for any initial guess $x^{(0)}$.*

6.3 General form of *stationary linear iterative methods* for $Ax = b$, and the error equation

6.3.1 The error equation

Consider $Au = f$ ($\det(A) \neq 0$) where u is the exact solution, and v is some approximate solution. Define the error $e := u - v = exact - approximate$, and residual $r := f - Av$ as before. Derive the error equation:

$$\begin{aligned} Ae &= A(u-v) \\ &= Au - Av \\ &= f - Av \\ &= r \end{aligned}$$

6.3.2 General form of stationary linear iterative methods

We have $u = v + (u - v)$, so $u = v + e$.
Note: knowing v, the residual, r, is easy to compute (one matrix-vector product). Use the error equation:

$$u = v + A^{-1}r$$

where computing the inverse of A is the most expensive operation. An idea is to replace A^{-1} with a "cheap" approximation: $B \approx A^{-1}$. So we get

$$v^{(i+1)} = v^{(i)} + Br^{(i)} \quad \text{with } r^{(i)} = f - Av^{(i)},$$

the *stationary iterative method*. Error:

$$\begin{aligned} u - v^{(i+1)} &= u - \left(v^{(i)} + Br^{(i)}\right) \\ u - v^{(i+1)} &= u - v^{(i)} - B(f - Av^{(i)}) \\ u - v^{(i+1)} &= u - v^{(i)} - B(Au - Av^{(i)}) \\ e^{(i+1)} &= (I - BA)e^{(i)}, \end{aligned}$$

or $e^{(i+1)} = Re^{(i)}$ with $R = I - BA$ error iteration matrix. Note: if $B = A^{-1}$ then $R = 0$, so $e^{(1)} = 0$ in one step.

Jacobi: Specific example of a stationary iterative method:

$$v^{(i+1)} = D^{-1}(f + (L+U)v^{(i)})$$

Using $A = D - L - U$, we get

$$\begin{aligned} L + U &= D - A \\ D^{-1}(L+U) &= I - D^{-1}A. \end{aligned}$$

So $v^{(i+1)} = D^{-1}f + (I - D^{-1}A)v^{(i)} = v^{(i)} + D^{-1}(f - v^{(i)})$, or

$$v^{(i+1)} = v^{(i)} + D^{-1}r^{(i)}$$

So for the Jacobi method, we have

$$B_J = D^{-1} \approx A^{-1}, \quad \text{and} \quad R_J = I - D^{-1}A.$$

Gauss-Seidel: Specific example of a stationary iterative method:

$$v^{(i+1)} = (D-L)^{-1}(f + Uv^{(i)})$$

Using $A = D - L - U$, we get

$$\begin{aligned} v^{(i+1)} &= (D-L)^{-1}f + (D-L)^{-1}(D-L-A)v^{(i)} \\ &= (D-L)^{-1}f + (I - (D-L)^{-1}A)v^{(i)} \\ &= v^{(i)} + (D-L)^{-1}(f - Av^{(i)}) \\ &= v^{(i)} + (D-L)^{-1}r^{(i)} \end{aligned}$$

So for the Gauss-Seidel method, we have

$$B_{GS} = (D-L)^{-1} \approx A^{-1}, \quad \text{and} \quad R_{GS} = I - (D-L)^{-1}A.$$

Theorem 6.6 (Convergence theorem)**.** *Consider stationary iterative method:*

$$v^{(i+1)} = v^{(i)} + B(f - Av^{(i)}) \tag{6.1}$$

for linear system $Au = f$, with $\det(A) \neq 0$. If there exists a p-norm such that $\|I - BA\|_p < 1$, then iteration (6.1) converges to the unique solution of $Au = f$ for any initial guess $v^{(0)}$.

Proof. (6.1) holds iff $u - v^{(i+1)} = u - v^{(i)} - B(f - Av^{(i)})$ iff $e^{(i+1)} = (I - BA)e^{(i)}$. Then

$$\begin{aligned} & \|e^{(i+1)}\|_p \leq \|I - BA\|_p \|e^{(i)}\|_p \\ \implies & \|e^{(i+1)}\|_p \leq \|I - BA\|_p^{i+1} \|e^{(0)}\|_p \\ \implies & \lim_{i\to\infty} \|e^{(i)}\|_p = 0 \qquad (\because \|I - BA\|_p < 1) \\ \implies & \lim_{i\to\infty} e^{(i)} = 0 \end{aligned}$$

□

Lecture 18

Theorem 6.7. *Let A be strictly row diagonally dominant. Then Jacobi converges to the exact solution of $Au = f$ for any initial guess.*

Proof. We know that $A = D - L - U$, and $B_J = D^{-1}$. Note that A^{-1} exists. Then

$$\begin{aligned}\|R_J\|_p &= \|I - D^{-1}A\|_p \\ &= \|I - D^{-1}(D - L - U)\|_p \\ &= \|D^{-1}(L + U)\|_p.\end{aligned}$$

We know $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$ $\forall i$ (row sums). Consider $p = \infty$. Then the maximal absolute row sum $= \|A\|_p$. So

$$\|D^{-1}(L+U)\|_\infty = \max_{1\leq i\leq n} \frac{\sum_{j\neq i} |a_{ij}|}{|a_{ii}|} < 1$$

by the previous convergence theorem, Jacobi converges. □

6.4 Spectral radius convergence theory:

Definition 6.8. Spectral radius of $A \in \mathbb{R}^{n\times n}$ is defined to be

$$\rho(A) = \max_{1\leq i\leq n} |\lambda_i|$$

Recall $\|A\|_2 = \max_i \sqrt{\lambda_i(A^T A)}$.
So if $A = A^T$, then $\|A\|_2 = \rho(A)$.

Theorem 6.9.

$$\rho(A) \leq \|A\|_p \text{ for all } p$$

Proof. Consider any eigenvalue, eigenvector pair λ, x; normalized such that $\|x\|_p = 1$. From $Ax = \lambda x$, we got

$$\begin{aligned} & \|\lambda x\|_p \leq \|A\|_p \|x\|_p \\ \implies \quad & |\lambda| \|x\|_p \leq \|A\|_p \|x\|_p \\ \implies \quad & |\lambda| \leq \|A\|_p, \end{aligned}$$

which implies that

$$\rho(A) \leq \|A\|_p.$$

□

Definition 6.10. $A \in \mathbb{R}^{n\times n}$ is convergent iff $\lim_{n\to\infty} A^n = 0$.

Theorem 6.11. *The following statements are equivalent*

1) A *is convergent*

2) $\lim_{n\to\infty} A^n = 0$

3) $\lim_{n\to\infty} \|A^n\|_p = 0$ *for some* p

4) $\lim_{n\to\infty} \|A^n\|_p = 0$ *for all* p

5) $\rho(A) < 1$

6) $\lim_{n\to\infty}(A^n x) = 0$ *for all* x

Recall that $e^{(i+1)} = Re^{(i)}$. Note:

$$\begin{aligned} & \text{stationary iterative method converges} \\ \iff \quad & \lim_{i\to\infty} e^{(i)} = 0 \\ \iff \quad & \lim_{i\to\infty} R^i e^{(0)} = 0 \\ \iff \quad & R \text{ is convergent} \\ \iff \quad & \rho(R) < 1 \end{aligned}$$

This is consistent with Theorems 6.6, 6.9, and 6.11.
Theorem 6.6: convergence of $\|R\|_p < 1$ for some p implies (by Theorem 6.9) that $\rho(R) < 1$ (consistent with Theorem 6.11)
Note: it is possible that $\rho(A) < 1$, but $\|A\|_p > 1$ for some p.

7 Multigrid methods for $Au = f$

Use slides: "A multigrid tutorial" by Brandt, McCormick and Henson. This method is useful for solving $Au = f$, when A is sparse.
Recall: LU: $W \in \mathcal{O}(n^3)$ Multigrid: $W \in \mathcal{O}(n)$.

7.1 Stationary iterative method

$$v^{(i+1)} = v^{(i)} + Br^{(i)}$$

Rewrite:

$$\begin{aligned} v^{(i+1)} &= v^{(i)} + B(f - Av^{(i)}) \\ &= (I - BA)v^{(i)} + Bf \\ v^{(i+1)} &= Rv^{(i)} + Bf \end{aligned}$$

Slide 16:

$$e^{(new)} = Re^{(old)}$$

we would like $e^{(new)}$ to be small.
case 1: oscillatory error ("high-frequency changes")
→ is quickly reduced ("relaxed") (just a few iterations)
case 2: "smooth error" ("low-frequency")
→ many iterations are required to reduce the smooth error

Lecture 19

Slide 26:
Suppose $R = R^T$, and $\rho(R) = 0.1 = 10^{-1}$, choose $e^{(0)}$.
Note: $\|R\|_2 = \rho(R)$.
Assume we require reduction in $\|e\|_2$ by 10^5 $(d = 5)$.
How many iterations are required?
5 steps are sufficient to reduce the initial error by 10^5, because each step reduces the error by at least 10.

In general, we require

$$\begin{aligned} & \frac{\|e^{(n)}\|}{\|e^{(0)}\|} \leq \|R\|^n = \rho(R)^n \sim 10^{-d} \\ \implies \quad & n \log_{10} \rho(R) \sim -d \\ \implies \quad & n \sim \frac{d}{-\log_{10} \rho(R)} \left(= \frac{5}{-\log_{10}(0.1)} = 5 \right) \end{aligned}$$

$\rho(R)$ is called the convergence factor of the method. If $R \neq R^T$, then $\rho(R)$ is still called the convergence factor of the method, but its interpretation is valid only asymptotically.
Recall: $\rho(R) \leq \|R\|_p$, but note: it is possible that $\rho(R) < 1$ and $\|R\|_p > 1$. Recall: $\|e^{(new)}\|_p \leq \|R\|_p \|e^{(old)}\|_p$, but:

$$\rho(R) = \lim_{n\to\infty} \left(\|R^n\|_p^{1/n} \right)$$

Slide 31:
High-frequency error: smoothing, relaxation is efficient in reducing the error.
Low-frequency error: relaxation is not efficient in reducing the error.
Slide 37:
Nested Iteration:

Smooth error on the fine grid (what remains after relaxation) is relatively more oscillatory on the course grid.

Slide 42:

From coarse (Ω^{2h}) to fine (Ω^h): interpolation

$$v^h = I_{2h}^h v^{2h}$$

v^h: vector of find grid.
v^{2h}: vector coarse grid.
I_{2h}^h: interpolation matrix from coarse to fine.

Lecture 20

Slide 51: Coarse error equation: Two possibilities:

1. $P^T A P e^{2h} = P^T r$
2. Model problem discretized on $\Omega^h : A^h$, $\Omega^{2h} : A^{2h}$.
   On coarse level: use $A^{2h} e^{2h} = I_h^{2h} r^h$, where I_h^{2h} is the restriction matrix.

We use the second version.

Slide 63:

initial guess $v_0^h \to r_0^h = f - A v_0^h, \quad \|r_0^h\|_2$
first V-cycle $v_1^h \to r_1^h = f - A v_1^h, \quad \|r_1^h\|_2$
second V-cycle $v_2^h \to r_2^h = f - A v_2^h, \quad \|r_2^h\|_2$
...
every V-cycle reduces the current residual (approximately) by a constant factor, which is (approximately) independent of h

$$\frac{\|r_{i+1}^h\|_2}{\|r_i^h\|_2} \approx \rho \quad \text{convergence factor (e.g. } \rho \approx 0.1)$$

Lecture 21

Slide 67:

1 Work unit (WU) = cost of 1 relaxation iteration on the finest grid.
For example: in a sparse 2D model problem, let $n = N^2$ (total number of unknowns on the finest grid), $A \in \mathbb{R}^{n \times n}$.
Consider RBGS: work for 1 WU = $\mathcal{O}(n)$.
Consider Multigrid V(1,1)-cycle: work $< \frac{8}{3}$ WUs = $\mathcal{O}(n)$.
($\mathcal{O}(n^2)$ if not sparse for both)
Note: multigrid is a "divide-and-conquer" algorithm.

Slide 69:

1D model problem[13] $\to$ exact solution $u(x)$. Define:

$u^{(h)}$: exact solution vector $u(x)$ sampled on the discrete grid.

u^h: exact solution of linear system $A^h u^h = f^h$.

v^h: approximate solution of $A^h u^h = f^h$.

Discretization error: $E_i = u(x_i) - u_i^h \implies \|E\|_h \le K h^p$ (in our case, $p = 2$).

Algebraic error: $e_i^h = u_i^h - v_i^h$

Total error: $e_i = u(x_i) - v_i^h$

[13] ODE, continuous problem

Note: $e = u^{(h)} - v^h = u^{(h)} - u^h + u^h - v^h$.

Slide 72:

MG cost to reduce algebraic error to the level of discretization

$$W = \mathcal{O}(N^d \log N) = \mathcal{O}(n \log N) \quad \text{where } n = N^d$$
$\mathcal{O}(\log N)$ V-cycles
$\mathcal{O}(N^d)$ work per V-cycle

***Grid norm* of a vector: ($\|\cdot\|_h$)**

$$\|r^h\|_h = h\|r^h\|_2$$
$$\|e^h\|_h = h\|e^h\|_2$$

Why the h?
Consider functions $u(x,y)$, $v(x,y)$ on Ω. define:

$$\begin{aligned}\|u(x,y) - v(x,y)\|_2 &= \sqrt{\iint_\Omega (u(x,y) - v(x,y))^2 dx dy} \\ &\approx \sqrt{\sum_i \sum_j (u(x_i,y_j) - v(x_i,y_j))^2 \Delta x \Delta y} \\ &= h\sqrt{\sum_i \sum_j (u_{ij}^h - v_{ij}^h)^2} \quad (\Delta x = \Delta y = h) \\ &= h\|u^h - v^h\|_2\end{aligned}$$

7.2 Red-Black Gauss-Seidel

Traditional GS ("lexicographic"):

$$v_i^{new} = \frac{1}{a_{ii}} \left(f_i - \sum_{j<1} a_{ij} v_j^{new} - \sum_{j>i} a_{ij} v_j^{old} \right)$$

All GS: compute v_i^{new} from equation i, using any new values that were computed previously. In lexicographic GS, we consider equations in the order of the rows of A.

2D model problem: Suppose $N = 7$ (# of interior points), $A \in \mathbb{R}^{7^2 \times 7^2}$, and $u \in \mathbb{R}^{7^2} = \mathbb{R}^{49}$.

Lexicographic: GS is sequential (no parallelism).

Red-Black GS: First update all the red points $\to$ can be done in parallel: because red points only depend on black points. (make sure all new red values are visible to the black points). Then update all the black points, using the new red values $\to$ can be done in parallel.

Definition 7.1. The multigrid convergence factor is defined by

$$\rho_{MG} := \frac{\|r_{i+1}\|_2}{\|r_i\|_2} \quad \text{(in the "asymptotic" region)}$$

where r_i are the residuals after multigrid V-cycle i.

Note: ρ_{MG} is small for the 2D model problem for all problem sizes.

Proposition 7.2. *ρ_{MG} is small and bounded uniformly in h:*

$$\exists c < 1 : \rho_{MG} < c < 1 \ \forall h$$

Lecture 22

7.3 Full Multigrid Method (FMG)

Find a better initial guess on grid h by finding an approximate solution on grid $2h$ and interpolating it up to grid h. (Do so recursively using smaller V-cycles).

7.4 Summary

To reduce $\|r\|_h$ by a mixed factor, the V-cycle multigrid method requires:
Total work $= \mathcal{O}(n)$ (linear scaling), since work per V-cycle is $\mathcal{O}(n)$, and the number of V-cycles required is constant in n.

To achieve convergence up to discretization error, the V-cycle multigrid method requires:
Total work $= \mathcal{O}(N^2 \log N)$ (not linear in n), since work per V-cycle is $\mathcal{O}(n)$, but the number of V-cycles grows with $\mathcal{O}(\log N)$.

To achieve convergence up to discretization error, the full multigrid method requires:
Total work $= \mathcal{O}(N^d) = \mathcal{O}(n)$ (linear in n), since only one full V-cycle is required.

8 Conjugate Gradient (CG) Method for $Ax = b$, [4, Ch. 38]

8.1 Algorithm

Let $A \in \mathbb{R}^{m \times m}$ be SPD. Let x_* be the exact solution of $Ax = b$. Let $e_n = x_* - x_n$ be the error ($n = 0, 1, 2, \dots$). Let x_0 be the initial guess, and $r_0 = b - Ax_0$, the initial residual.

Definition 8.1. Define Krylov space to be:

$$K_n = \text{span}\{r_0, Ar_o, A^2 r_0, \dots, A^{n-1} r_0\}.$$

CG is an iterative method that proceeds as follows:
In step n ($n = 0, 1, 2, \dots$), find $q_n \in K_n$ s.t.
$x_n = x_{n-1} + q_n$ minimizes

$$\|e_n\|_A = \|x_* - x_n\|_A = \|x_* - x_{n-1} - q_n\|_A$$

Note: the minimizer q_n can be computed very efficiently. Often the error is reduced very fast (only a small number of iterations is needed).

Algorithm 1 Conjugate Gradient (CG)

Choose x_0
$r_0 \leftarrow b - Ax_0$
$p_0 \leftarrow r_0$
$n \leftarrow 0$
repeat
  $n \leftarrow n + 1$
  $\alpha_n \leftarrow (r_{n-1}^T r_{n-1})/(p_{n-1}^T A p_{n-1})$
  $x_n \leftarrow x_{n-1} + \alpha_n p_{n-1}$
  $r_n \leftarrow r_{n-1} - \alpha_n A p_{n-1}$
  $\beta_n \leftarrow (r_n^T r_n)/(r_{n-1}^T r_{n-1})$
  $p_n \leftarrow r_n + \beta_n p_{n-1}$
until convergence criterion is satisfied

Notes:

- Efficient implementation: reuse certain intermediate results.
- p_{n-1} is called a "search direction" (α_n is computed such that the *optimal* solution in the direction p_{n-1} is found).
- We will show that every CG step minimizes $\|e_n\|_A$ over all
$$q_n = \alpha_n p_{n-1} \in K_n$$
- Even though we optimize over the whole of K_n, we only need to store one previous r_n and p_n.
- Work per step (assume sparse A).
  1. 1 matrix-vector product:
  $$(Ap_{n-1}) : W = \mathcal{O}(m)$$
  2. 2 scalar products:
  $$r_n^T r_n \text{ and } p_{n-1}^T A p_{n-1} : W = \mathcal{O}(m) \text{ each}$$
  3. 3 vector "multiply-add":
  $$\left.\begin{array}{l} x_{n-1} + \alpha_n p_{n-1} \\ r_{n-1} + \alpha_n (Ap_{n-1}) \\ r_n + \beta_n p_{n-1} \end{array}\right\} W = \mathcal{O}(m) \text{ each}$$

  total work per CG step: $\mathcal{O}(m)$.

8.2 CG as an optimization problem

Note: $Ax = b$ with A SPD has a unique solution x_*.
Let $e = x_* - x$. Observe:

$$\begin{aligned} \|x_* - x\|_A^2 = \|e\|_A^2 &= e^T A e = (x_* - x)^T A (x_* - x) \\ &= x^T A x - 2x^T A x_* + x_*^T A x_* \quad (\text{use } A = A^T) \\ &= x^T A x - 2x^T b + x_*^T b \\ &= 2\phi(x) + x_*^T b = 2\phi(x) - 2\phi(x_*) \end{aligned}$$

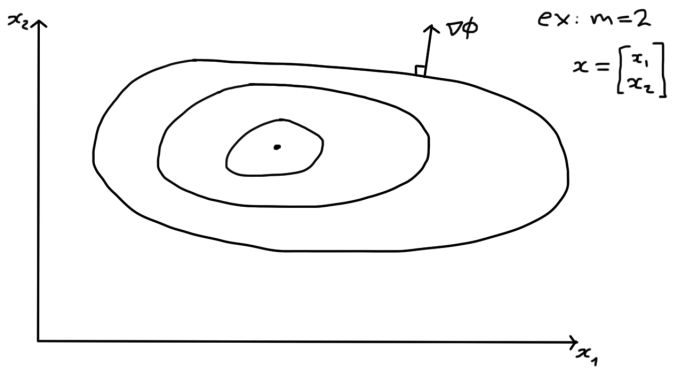
where

$$\phi(x) = \frac{1}{2} x^T A x - x^T b$$

Proposition 8.2. *x_* is the unique minimizer of $\phi(x)$ over $\mathbb{R}^m$. Alternatively we can say that solving $Ax = b$ is equivalent to minimizing $\phi(x)$ over $\mathbb{R}^m$.*

8.3 Steepest descent algorithm

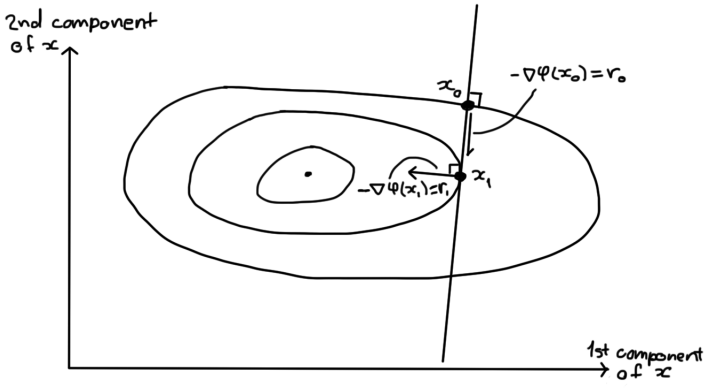
Property: $\nabla\phi(x)$ point in the direction of steepest ascent.



$-\nabla\phi(x)$ points in the direction of steepest descent.

$$\begin{aligned} -\nabla\phi(x) &= -\nabla(\frac{1}{2}x^TAx - b^Tx) \\ &= -(Ax - b) \qquad (\text{since } A = A^T) \\ &= b - Ax = r(x) \end{aligned}$$

The general idea:
Have $x_0, x_1, x_2, \ldots, x_n, \ldots$.
Start from x_0:
Compute $-\nabla\phi(x_0) = r_0 = b - Ax_0$
Let $x_1 = x_0 + \alpha_1 r_0$ (r_0 is the search direction)
Find the optimal approximation (smallest $\phi(x_1)$) in the direction of r_0 (steepest descent direction).
Determine α_1 s.t. $\phi(x_1)$ is minimal.



We require $\frac{d}{d\alpha_1}(\phi(x_1(\alpha_1))) = 0$:

$$\begin{aligned} \nabla\phi(x_1)^T\frac{dx_1}{d\alpha_1} &= 0 \\ -r_1^Tr_0 &= 0 \\ -(b - Ax_1)^Tr_0 &= 0 \\ (A(x_0 + \alpha_1 r_0) - b)^Tr_0 &= 0 \\ (-r_0 + \alpha_1 Ar_0)^Tr_0 &= 0 \\ \implies \alpha_1 &= \frac{r_0^Tr_0}{r^TAr_0} \end{aligned}$$

Note:

$$\begin{aligned} r_1 &= b - Ax_1 \\ &= b - A(x_0 + \alpha_1 r_0) \\ &= r_0 - \alpha_1 Ar_0 \end{aligned}$$

Algorithm 2 STEEPEST DESCENT (SD)

1: Choose x_0
2: $r_0 \leftarrow b - Ax_0$
3: $n \leftarrow 0$
4: **repeat**
5: $\quad n \leftarrow n + 1$
6: $\quad \alpha_n \leftarrow (r_{n-1}^Tr_{n-1})/(r_{n-1}^TAr_{n-1})$
7: $\quad x_n \leftarrow x_{n-1} + \alpha_n r_{n-1}$
8: $\quad r_n \leftarrow r_{n-1} - \alpha_n Ar_{n-1}$
9: **until** convergence criterion is satisfied

Compare with CG:

- SD also has $W = \mathcal{O}(m)$ per step (1 matrix-vector product, 2 scalar products, 2 scalar multiply-adds).
- Similar to CG, but $p_n = r_n$ in SD.
- CG also does 1D minimization in step 7 (Algorithm 3), but p_n is chosen s.t. the minimization is also automatically over the whole K_n.
- SD can converge very slowly if $\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}} \gg 1$.
  CG often converges much faster (in particular, convergence in at most m steps).

8.4 Examples and Convergence of CG and SD methods

Example 8.3. Solve $Ax = b$:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix},$$
$$\implies x_* = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

Eigenvectors of A:

$$w_1 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \qquad \lambda_1 = 2 \qquad (A \text{ SPD})$$
$$w_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \lambda_2 = 7$$

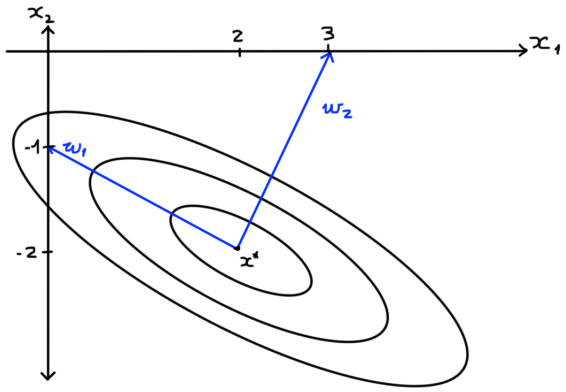


Figure 8.1: Level curves of $\phi(x)$.

Note:

$$\|x_* - x\|_A^2 = 2\phi(x) - \phi(x_*)$$
$$\implies \phi(x_* + w_1) = \phi(x_*) + \frac{1}{2}\|w_1\|_A^2$$
$$= \phi(x_*) + \frac{1}{2}\lambda_1\|w_1\|_2^2$$
$$\phi(x_* + w_2) = \phi(x_*) + \frac{1}{2}\lambda_2\|w_2\|_2^2$$

Recall: $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\lambda_{max}}{\lambda_{min}}$ (A SPD).
Note: $\kappa_2(A)$ large $\iff$ strongly elongated ellipsoids $\kappa_2(A) = 1 \iff$ circle (sphere)

Recall: Steepest Descent Method (Algorithm 2)
Note that computing α_n gives us the optimal point in search direction (1D minimization). Also r_{n-1} is the search direction:

$$r_{n-1} = -\nabla\phi(x_{n-1}) = \text{direction of steepest descent}$$

Problem: SD may "zig-zag" (slow convergence) if $\kappa(A)$ is large.

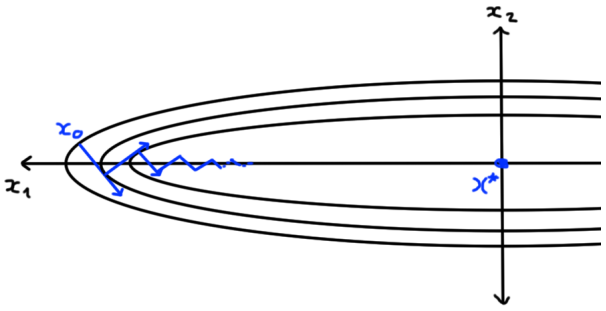


Figure 8.2: Example of how for large $\kappa(A)$, SD may have slow convergence.

In contrast, CG most often converges much faster (no zig-zag effect) (at most m steps)
Looking ahead: ($\kappa = \kappa_2(A)$)

Theorem 8.4. *For the CG method, we have:*

$$\frac{\|e_n\|_A}{\|e_0\|_A} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n$$

where $(\sqrt{\kappa}-1)(\sqrt{\kappa}+1)$ *is the error reduction per step.*

For example,

$$\kappa = 4 \implies \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{2-1}{2+1} = \frac{1}{3} \quad \leftarrow \text{nice reduction}$$

$$\kappa = 100 \implies \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{10-1}{10+1} = \frac{9}{11} \quad \leftarrow \text{not so nice}$$

Theorem 8.5. *For the SD method, we have:*

$$\frac{\|e_n\|_A}{\|e_0\|_A} \le \left(\frac{\kappa-1}{\kappa+1}\right)^n$$

where $(\kappa-1)(\kappa+1)$ *is the error reduction per step.*

For example,

$$\kappa = 4 \implies \frac{\kappa-1}{\kappa+1} = \frac{4-1}{4+1} = \frac{3}{5} \quad \leftarrow \text{ok reduction}$$

$$\kappa = 100 \implies \frac{\kappa-1}{\kappa+1} = \frac{100-1}{100+1} = \frac{99}{101} \quad \leftarrow \text{terrible}$$

How many iterations of CG are required to reduce the error by a given factor, e.g. 10^8?
Require:

$$\frac{\|e_n\|_A}{\|e_0\|_A} \le \varepsilon \qquad (\text{e.g. } \varepsilon = 10^{-8})$$

Require:

$$2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \approx \varepsilon$$

where n is the # of steps required. Alternatively we need

$$2\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^n \approx \varepsilon$$

or

$$n\log\left(1 - \frac{2}{\sqrt{\kappa}+1}\right) \approx \log\left(\frac{\varepsilon}{2}\right)$$

Assume $\kappa \gg 1$, and $|t| \ll 1$. Then

$$\begin{aligned} \sqrt{\kappa}+1 &\approx \sqrt{\kappa} \\ \log(1+t) &\approx t \\ \implies n\left(\frac{-2}{\sqrt{\kappa}}\right) &\approx \log\left(\frac{\varepsilon}{2}\right) \\ \text{or } n &\approx \frac{\sqrt{\kappa}}{2}\log\left(\frac{2}{\varepsilon}\right) \end{aligned}$$

Conclusion:
CG: $n \in \mathcal{O}(\sqrt{\kappa})$ iterations required to reach convergence tolerance. Similarly: SD: $n \in \mathcal{O}(\kappa)$ iterations required.

Example 8.6. 2D model problem.

$$\kappa(A^h) \approx \frac{4}{\pi^2}m \qquad (m = \text{total \# of unknowns})$$

CG, SD: $\mathcal{O}(m)$ work per iteration.
Conclusion:

$$\begin{aligned} &\text{CG: } \mathcal{O}(\sqrt{\kappa}) = \mathcal{O}(\sqrt{m}) \quad \text{iterations} \\ &\qquad \mathcal{O}(m) \quad \text{work per iteration} \\ &\implies \text{total work } = \mathcal{O}(m^{3/2}) \\ &\text{SD: } \mathcal{O}(\kappa) = \mathcal{O}(m) \quad \text{iterations (like GS)} \\ &\qquad \mathcal{O}(m) \quad \text{work per iteration} \\ &\implies \text{total work } = \mathcal{O}(m^2) \\ &\text{MG: total work } = \mathcal{O}(m) \end{aligned}$$

Lecture 24

8.5 Properties of the CG algorithm

Recall the conjugate gradient algorithm:

Algorithm 3 Conjugate Gradient (CG)

1: Choose x_0
2: $r_0 \leftarrow b - Ax_0$
3: $p_0 \leftarrow r_0$
4: $n \leftarrow 0$
5: **repeat**
6: $\quad n \leftarrow n+1$
7: $\quad \alpha_n \leftarrow (r_{n-1}^T r_{n-1})/(p_{n-1}^T A p_{n-1})$
8: $\quad x_n \leftarrow x_{n-1} + \alpha_n p_{n-1}$
9: $\quad r_n \leftarrow r_{n-1} - \alpha_n A p_{n-1}$
10: $\quad \beta_n \leftarrow (r_n^T r_n)/(r_{n-1}^T r_{n-1})$
11: $\quad p_n \leftarrow r_n + \beta_n p_{n-1}$
12: **until** convergence criterion is satisfied

Recall the Krylov space:

$$K_n = \text{span}\underbrace{\{r_0, Ar_0, \ldots, A^{n-1}r_0\}}_{n \text{ vectors}}$$

For simplicity, assume $x_0 = 0 \implies r_0 = b - Ax_0 = b$.

$$\begin{aligned} \implies K_n &= \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\} = V_{r_0} \\ &= \text{span}\{b, Ab, \ldots, A^{n-1}b\} = V_b \end{aligned}$$

Theorem 8.7. *Assume $r_{n-1} \neq 0$ (not converged yet). Then*

$$\left.\begin{aligned} K_n &= \text{span}\{r_0, r_1, r_2, \ldots, r_{n-1}\} = V_r \\ &= \text{span}\{p_0, p_1, p_2, \ldots, p_{n-1}\} = V_p \\ &= \text{span}\{x_1, x_2, \ldots, x_n\} = V_x \end{aligned}\right\} \text{(a)}$$

Moreover,

$$\left.\begin{aligned} r_n^T r_j &= 0 \quad (j < n) \\ p_n^T A p_j &= 0 \quad (j < n) \end{aligned}\right\} \text{(b)}$$

Proof.

(a) $\implies V_x = V_p$ follows from line 8 in the CG algorithm: $x_n = x_{n-1} + \alpha_n p_{n-1}$. Two steps:

(a) $V_p \subset V_x$: every element of V_p is in V_x. This is true because p_{n-1} is a linear combination of x_n and x_{n-1}.

(b) $V_x \subset V_p$: because

$$\begin{aligned} x_1 &= \alpha_1 p_0 \qquad (x_0 = 0) \\ x_2 &= x_1 + \alpha_a p_1 \\ &\ldots \end{aligned}$$

We use induction: assume x_{i-1} is a linear combination of $\{p_0, \ldots, p_{i-2}\}$; then, from line 8, x_i is a linear combination of $\{p_0, \ldots, p_{i-1}\}$.

Therefore $V_x = V_p$.
$\implies V_r = V_p$ follows from line 11: $p_n = r_n + \beta_n p_{n-1}$.
Observe: $p_0 = r_0$. Proof is the same as for $V_x = V_p$.
$\implies V_r = V_b$ follows from line 9 and $V_r = V_p$.

$$r_n = r_{n-1} - \alpha_n A p_{n-1}.$$

Observe: $p_0 = r_0 = b$ (base case), which implies that $\text{span}\{r_0\} = \text{span}\{b\}$.

Induction: Assume

$$\begin{aligned} \text{span}\{r_0, \ldots, r_i\} &= \text{span}\{b, Ab, \ldots, A^i b\} \\ &(= \text{span}\{p_0, \ldots, p_i\} \quad \because V_r = V_p). \end{aligned}$$

Then from line 9 we get:

$$r_{i+1} = r_i - \alpha_{i+1} A p_i,$$

$$\begin{aligned} \text{then} \quad & r_{i+1} \in \text{span}\{b, Ab, \ldots, A^{i+1}b\} \\ \text{since} \quad & p_i \in \text{span}\{b, \ldots, A^i b\} \\ \text{so} \quad & Ap_i \in \text{span}\{Ab, \ldots, A^{i+1}b\} \\ \text{and} \quad & r_i \in \text{span}\{b, \ldots, A^i b\} \\ \text{and thus} \quad & A^{i+1}b \in \text{span}\{r_0, \ldots, r_i\} \\ \text{since} \quad & Ap_i \in \text{span}\{\underbrace{Ab, \ldots, A^i b}_{\in \text{span}\{r_0, \ldots, r_i\}}, A^{i+1}b\} \end{aligned}$$

Therefore $\text{span}\{r_0, \ldots, r_{i+1}\} = \text{span}\{b, Ab, \ldots, A^{i+1}b\}$.

(b) Proof by induction: Assume

$$r_{n-1}^T r_j = 0 \quad (j < n-1)$$
$$p_{n-1}^T A p_j = 0 \quad (j < n-1)$$

Base case:

$$\left.\begin{aligned} r_1^T r_0 &= 0 \\ p_1^T A p_0 &= 0 \end{aligned}\right\} \text{can be shown}$$

we will show

$$r_n^T r_j = 0 \quad (j < n)$$
$$p_n^T A p_j = 0 \quad (j < n)$$

To show that $r_n^T r_j = 0$ $(j < n)$, use line 9:

$$r_n = r_{n-1} - \alpha_n A p_{n-1}$$
$$\implies r_n^T r_j = r_{n-1}^T r_j - \alpha_n p_{n-1}^T A r_j$$

Case $j < n-1$:

$$\left.\begin{aligned} &r_{n-1}^T r_j = 0 \text{ by induction} \\ &p_{n-1}^T A r_j = 0 \text{ by induction } \because\ V_r = V_p \end{aligned}\right\} \implies r_n^T r_j = 0$$

Case $j = n-1$: $r_n^T r_j = 0$ if

$$\alpha_n = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A r_{n-1}}.$$

Why? Compare with line 7 which gives the same α_n because

$$r_{n-1} = p_{n-1} - \beta_{n-1} p_{n-2} \text{ from line 11}$$

$$\begin{aligned} \implies \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A r_{n-1}} &= \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A (p_{n-1} - \beta n - 1 p_{n-2})} \\ &= \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}} \quad \begin{array}{l}\because\ p_{n-1}^T A p_{n-2} = 0 \\ \text{by induction}\end{array} \\ &= \alpha_n \quad \text{from the CG algorithm} \end{aligned}$$

To show that $p_n^T A p_j = 0$ $(j < n)$, use line 11:

$$p_n = r_n + \beta_n p_{n-1}$$

to get $p_n^T A p_j = r_n^T A p_j + \beta_n p_{n-1}^T A p_j$.

Case $j < n-1$: $r_n^T A p_j = 0$, since

$$\begin{aligned} p_j &\in \text{span}\{b, \ldots, A^{n-2} b\} && (\because\ V_p = B_b) \\ A p_j &\in \text{span}\{Ab, \ldots, A^{n-1} b\} \\ &\subset \text{span}\{r_0, \ldots, r_{n-1}\} && (\because\ V_r = V_b) \\ \implies r_n^T A p_j &= 0 \quad (j < n-1) \quad (\because\ r_n^T r_j = 0 \quad (j < n)) \end{aligned}$$

Case $j = n-1$: $p_n^T A p_j = 0$ if

$$\beta_n = \frac{-r_n^T A p_{n-1}}{p_{n-1}^T A p_{n-1}}$$

Show that this equals the β_n from the CG algorithm:

$$\begin{aligned} \frac{-r_n^T A p_{n-1}}{p_{n-1}^T A p_{n-1}} &= \frac{-r_n^T A p_{n-1} \alpha_n}{r_{n-1}^T r_{n-1}} \\ &= \frac{-r_n^T (r_{n-1} - r_n)}{r_{n-1}^T r_{n-1}} \\ &= \frac{r_n^T r_n}{r_{n-1}^T r_{n-1}} = \beta_n \quad \text{from the CG algorithm} \\ \implies p_n^T A p_j &= 0 \quad (j < n) \end{aligned}$$

□

Note: $r_n^T r_j = 0$ $(j < n)$: residual orthogonal.

Proposition 8.8. *CG needs at most m iterations to converge exactly (assuming no rounding errors).*

Proof. Consider $r_0, r_1, \ldots, r_{n-1}$, m vectors.

Case 1: $r_{m-1} = 0$: OK (we have converged within m steps)

Case 2: $r_{m-1} \neq 0$: $\implies r_m = 0$ because there are at most m non-zero orthogonal vectors in $\mathbb{R}^m$ $(r_0, \ldots, r_{m-1})$, and r_m has to be orthogonal to all of them.

□

Note: $p_n^T A p_j = 0$ $(j < n)$, "the search directions are A-conjugate" (A-orthogonal).
$\implies$ "conjugate gradient method"

Theorem 8.9. *CG selects $x_n \in K_n$ with*

$$K_n = \text{span}\{r_0, \ldots, r_{n-1}\}$$

such that $\|e_n\|_A$ is minimal. $\implies$ $\|e_n\|_A$ decreases monotonically.

References

[1] L. Elden, L. Wittmeyer-Koch, and H. B. Nielsen. *Introduction to Numerical Computation - analysis and Matlab illustrations.* Studentlitteratur, Lund (jul 2004).

[2] Y. Saad. *Iterative methods for sparse linear systems.* Siam (2003).

[3] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, vol. 37. Springer (2007).

[4] L. N. Trefethen and D. Bau III. *Numerical linear algebra.* 50. Siam (1997).