Numerical Methods for Computational Science and Engineering

Autumn Semester 2018

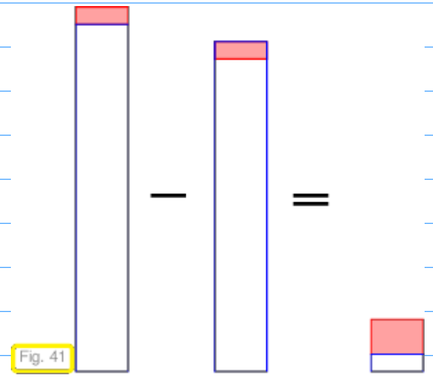
Prof. Rima Alaifari, SAM, ETH Zurich

Week 2

1.4.2. Costs of basic operations

Operation	Description	Number of $\cdot$, / operations	Number of $+, -$ operations	Asymp. Compl.
Dot product	$(\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^n) \mapsto \mathbf{x}^{\mathrm{H}}\mathbf{y}$	n	$n-1$	$\mathcal{O}(n)$
Tensor product	$(\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n) \mapsto \mathbf{x}\mathbf{y}^{\mathrm{H}}$	nm	0	$\mathcal{O}(nm)$
Matrix product †	$(\mathbf{A} \in \mathbb{R}^{m,n}, \mathbf{B} \in \mathbb{R}^{n,k}) \mapsto \mathbf{AB}$	mnk	$mk(n-1)$	$\mathcal{O}(mnk)$

1.5. Cancellation



Subtraction of two close-by numbers: Can result in the amplification of roundoff error (in the sense of relative errors)

Example 1: Roots of quadratic polynomial

$$p(\xi) = \xi^2 + \alpha\xi + \beta$$

$$\boxed{p(\xi_{1/2}) = 0} \quad ?$$

$$\xi_{1/2} = -\frac{\alpha}{2} \pm \frac{1}{2}\sqrt{\alpha^2 - 4\beta} \qquad (*)$$

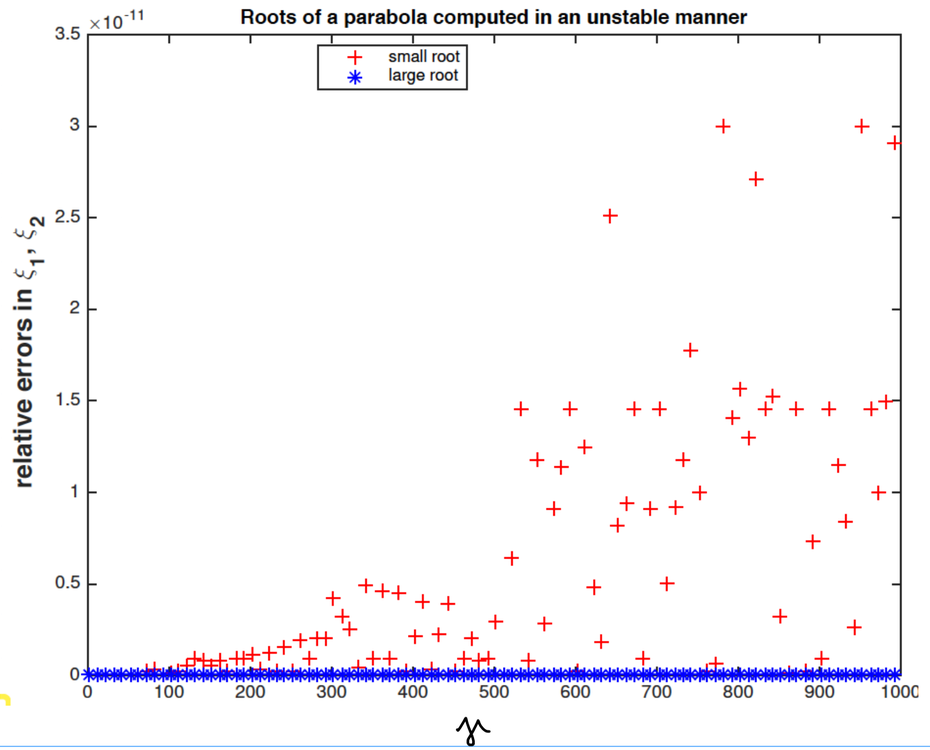
When can cancellation happen?

Answer: ξ_1 large, ξ_2 small

For example: $p(\xi) = (\xi - \mu)\left(\xi - \frac{1}{\mu}\right)$

$$= \xi^2 - \underbrace{\left(\mu + \frac{1}{\mu}\right)}_{\alpha} \xi + \underbrace{1}_{\beta}$$

Use (*) to compute ξ_1, ξ_2:



Large rel. errors in ξ_2 compared to EPS.

Cancellation happens because

$$\xi_2 = -\frac{\alpha}{2} + \frac{1}{2}\underbrace{\sqrt{\alpha^2 - 4}}_{\approx \alpha}$$

Can we avoid this? Design a stable algorithm?

Idea: Use Vieta's formula $\beta = \xi_1 \cdot \xi_2$

Solution:

1. Compute the root with large magnitude:

$$\xi_1 = -\frac{\alpha}{2} - \frac{1}{2}\sqrt{\alpha^2 - 4\beta} \quad \text{(no cancellation)}$$

2. $\xi_2 = \beta / \xi_1$ (stable).

Example: Difference quotient

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

approximation

Analysis: if f is suff. smooth: e.g. $f \in C^2$

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Approximation error: tends to 0 as $h \to 0$.

BUT: if h is small:

cancellation in $f(x+h) - f(x)$

+ amplification by $\frac{1}{h}$ (large)

Example: $f(x) = e^x$

Code Snippet 1. 20: *Difference quotient approximation of the derivative of* exp

```
//!  Difference quotient approximation
//!  of the derivative of exp
void diffq(){
  double h = 0.1, x = 0.0;
  for(int i = 1; i <= 16; ++i){
    double df = (exp(x+h)-exp(x))/h;
    cout << setprecision(14) << fixed;
    cout  << setw(5) <<  -i
    << setw(20) << df-1 << endl;
    h /= 10;
  }
}
/*
```

$$\frac{df - f'(0)}{f'(0)}$$

$\log_{10}(h)$	relative error
-1	0.05170918075648
-2	0.00501670841679
-3	0.00050016670838
-4	0.00005000166714
-5	0.00000500000696
-6	0.00000049996218
-7	0.00000004943368
-8	-0.00000000607747
-9	0.00000008274037
-10	0.00000008274037
-11	0.00000008274037
-12	0.00008890058234
-13	-0.00079927783736
-14	-0.00079927783736
-15	0.11022302462516
-16	-1.00000000000000

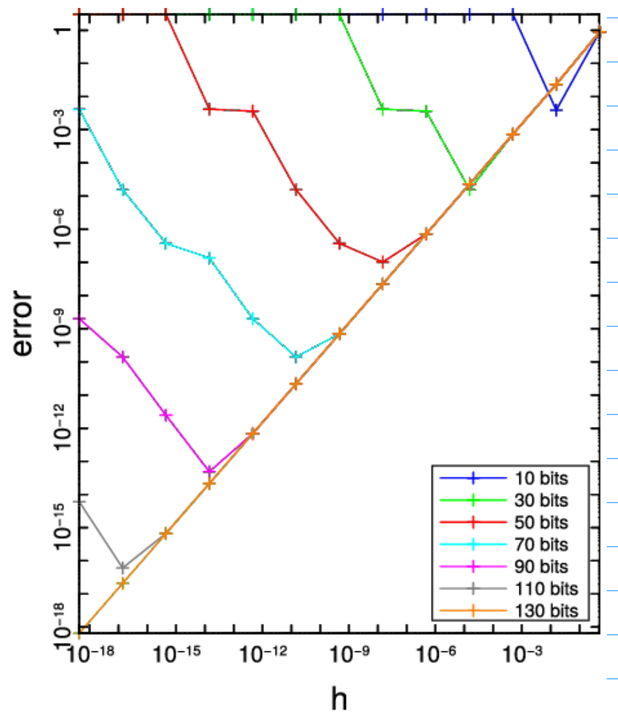


Fig. 42

There seems to be an optimal choice of h depending on the machine precision.

Use the axiom of roundoff analysis to compute an optimal choice for h:

diff. quotient $dq = \dfrac{e^{x+h}(1+\delta_1) - e^x(1+\delta_2)}{h}$

$|\delta_1|, |\delta_2| \leq EPS$

Task: Choose h s.t. $\left| \dfrac{dq - e^x}{e^x} \right|$ is minimized.

$$dq = e^x \left(\frac{e^h - 1}{h} + \frac{\delta_1 e^h - \delta_2}{h} \right) \qquad / : e^x$$

$$\underbrace{\frac{dq - e^x}{e^x}}_{\frac{dq}{e^x} - 1} = \underbrace{\frac{e^h - 1}{h} - 1}_{(*)} + \underbrace{\frac{\delta_1 e^h - \delta_2}{h}}_{|\cdot| \leq \frac{2|EPS|}{h}} \qquad (x=0)$$

$\approx \frac{h}{2}$

(*) Use: $$\underbrace{\frac{e^{x+h}-e^{x}}{h}}_{\text{diff. quot.}} - \underbrace{e^{x}}_{f'(x)} = \frac{1}{2} h e^{\xi}$$ for some $\xi \in [x, x+h]$

with $x=0$: $$\frac{e^{h}-1}{h} - 1 = \underbrace{\frac{1}{2} h e^{\xi}}_{\approx \frac{h}{2}} \qquad \xi \in [0,h]$$

$$\left| \frac{dq - e^{x}}{e^{x}} \right| \approx \frac{h}{2} + \frac{2EPS}{h} \quad \leftarrow \text{minimize}$$

$$\frac{1}{2} - \frac{2EPS}{h^2} \approx 0 \quad \Rightarrow \quad h \approx 2\sqrt{EPS}$$

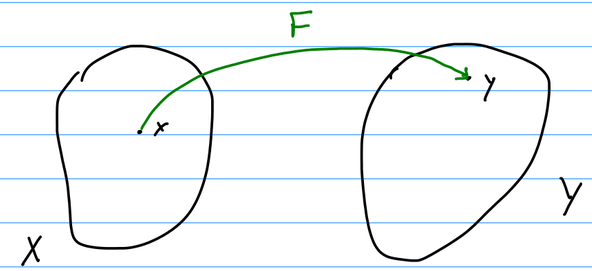
1.6. Numerical stability

Mathematical problem:

(input) data space X with $\|\cdot\|_X$

(output) solution space Y with $\|\cdot\|_Y$

mapping $F: X \to Y$



$F(x) = y$

Problem: Given $x \in X$, find $y \in Y$.

F is well-defined: for any $x \in X$ there exists a solution $y \in Y$.

Problem

$F : X \to Y \subseteq \mathbb{R}^m$
injective

Algorithm

$\tilde{F} : X \to \tilde{Y} \subset \mathbb{M}^m$

When is $\tilde{F}$ numerically stable?

Definition: An algorithm $\tilde{F}$ is numerically stable if for any input $x \in X$ the backward error is small, i.e.

$$\frac{\|x - \tilde{x}\|}{\|x\|} = \Theta(\text{cost}(\tilde{F}) \cdot \text{EPS})$$

where $\underbrace{\tilde{F}(x)}_{\text{output}} = F(\tilde{x})$.

Example: solving a linear system of equations with system matrix $A \in \mathbb{R}^{n,n}$.

$$Ay = b$$

$$F : \mathbb{R}^n \to \mathbb{R}^n$$

$$F(b) = y$$

An algorithm $\tilde{F}$ for finding y for given b is numerically stable if:

For all $b \in \mathbb{M}^n$: $\dfrac{\|b - b_{app}\|}{\|b\|} = \Theta(\text{cost}(\tilde{F}) \text{EPS})$

where $b_{app} = A y_{app}$

$y_{app} = \tilde{F}(b)$ (y_{app} is the output of $\tilde{F}$ for b)

$= F(b_{app})$ (y_{app} is <u>exact</u> solution to b_{app})

<u>Roughly</u>: Numerical stability (in this backward sense) guarantees: no amplification of roundoff by the algorithm

Recall previous section:

How close is $\tilde{F}(x)$ to $F(x)$?

$$\tilde{F}(x) = F(\tilde{x})$$

Ask: Sensitivity of F to input data:

When is $F(\tilde{x})$ close to $F(x)$

Condition number of F:

$$c_F(x) = \sup_{\Delta x \in B_r(0)} \left(\frac{\|F(x+\Delta x) - F(x)\|}{\|F(x)\|} \Big/ \frac{\|\Delta x\|}{\|x\|} \right)$$

In other words:

How does my output react to small changes in the input? How sensitive is my problem to small errors in input?

Recall: Condition of a matrix

$$c_A(x) = \sup_{\Delta x \in B_r(0)} \left(\frac{\|A\Delta x\|}{\|Ax\|} \Big/ \frac{\|\Delta x\|}{\|x\|} \right)$$

$$= \underbrace{\left(\sup_{\Delta x} \frac{\|A\Delta x\|}{\|\Delta x\|} \right)}_{\sigma_{max}} \Big/ \underbrace{\frac{\|Ax\|}{\|x\|}}_{\inf(\ ) = \sigma_{min}}$$

$$\text{cond}(A) := \frac{\sigma_{max}}{\sigma_{min}}$$

small: well-conditioned

large: ill-conditioned

We get $\dfrac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|} \leq \mathcal{O}(c_F(x) \cdot EPS)$

Thus:
- numerically stable algorithms solve accurately if c_F is small
- no guarantee on its accuracy w.r.t. input errors for large c_F.

impact of roundoff errors within $\tilde{F}$	is not worse than	impact of rounding the input for F

2. Direct Methods for Solving LSEs

Sensitivity for $\underline{Ax = b}$:

$Ax = b$ $\qquad$ $\Delta x = x - x_{app}$

$A x_{app} = b_{app}$ $\qquad$ $\Delta b = b - b_{app}$

$A \Delta x = \Delta b$ $\qquad$ rel. err. $\dfrac{\|\Delta x\|}{\|x\|}$

$\Delta x = A^{-1} \Delta b$

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \, \|\Delta b\|}{\|x\|}$$

$$\leq \|A^{-1}\| \cdot \|A\| \cdot \|\Delta b\| \cdot \underbrace{\frac{1}{\|A\| \cdot \|x\|}}_{\leq \frac{1}{\|b\|}}$$

$Ax = b$

$\Rightarrow \|b\| \leq \|A\| \, \|x\|$

$$\Rightarrow \quad \boxed{\underbrace{\frac{\|\Delta x\|}{\|x\|}}_{\text{output error}} \leq \underbrace{\|A^{-1}\| \cdot \|A\|}_{= \operatorname{cond}(A)} \cdot \underbrace{\frac{\|\Delta b\|}{\|b\|}}_{\text{input error}}}$$

LSE $\quad Ax = b \qquad A \in \mathbb{K}^{n,n}, \; b \in \mathbb{K}^n$

solve for x

if A is regular: x exists & is unique

A regular / invertible $\Leftrightarrow \mathcal{N}(A) = \{x : Ax = 0\} = \{0\}$

$\Leftrightarrow \operatorname{rank}(A) = n \Leftrightarrow \det A \neq 0$

In general: we do not compute A^{-1}

2.2. Gauss elimination

Idea:

$$\left[\begin{array}{c|c} A & b \end{array}\right] \in \mathbb{K}^{n, n+1}$$

apply row trafos & permutations to obtain

$$\left(\begin{array}{ccccc|c} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{array}\right) \qquad \text{forward substitution}$$

then: back substitution

Q: What is the complexity of forward substitution?

For every row: $k = 1, \dots, n-1$

modify all rows below:

for $i = k+1, \dots, n$

$l_{ik} = a_{ik} / a_{kk}$

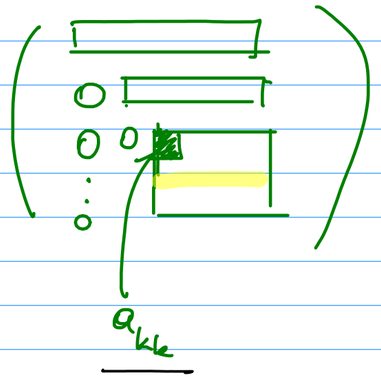
for $j = k, \dots, n$

$a_{ij} = a_{ij} - l_{ik} a_{kj}$

end

end

end



forward substitution: $\Theta(n^3)$

backward substitution: $\Theta(n^2)$

overall: cost (GE) $= \Theta(n^3)$

Q: Is Gauss elimination numerically stable?

NO.

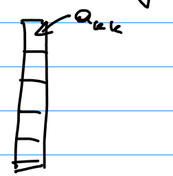
- Division by small numbers is possible
- Cancellation can also occur

Good remedy in practice: Partial pivoting

looks through current column k

& permutes rows of the matrix

so that the largest abs. value lies on diagonal.



Example:

$$\begin{pmatrix} 1 & 10 & -10 \\ 0 & \varepsilon & 9 \\ 0 & 4 & 6.2 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 10 & -10 \\ 0 & 4 & 6.2 \\ 0 & \varepsilon & 9 \end{pmatrix}$$

Partial pivoting works well in practice

When to expect numerical stability?

- s.p.d. matrices
- diagonally dominant matrices

$$\forall\, i \in \{1,\dots,n\} : |(A)_{ii}| \geq \sum_{j \neq i} |(A)_{ij}|$$

<u>Often</u>: We often want to solve for the same matrix A, but with multiple RHS $b_1, \dots b_N$

Then: We should "remember" how we modified A to an upper triangular matrix (independent of A)

LU factorization

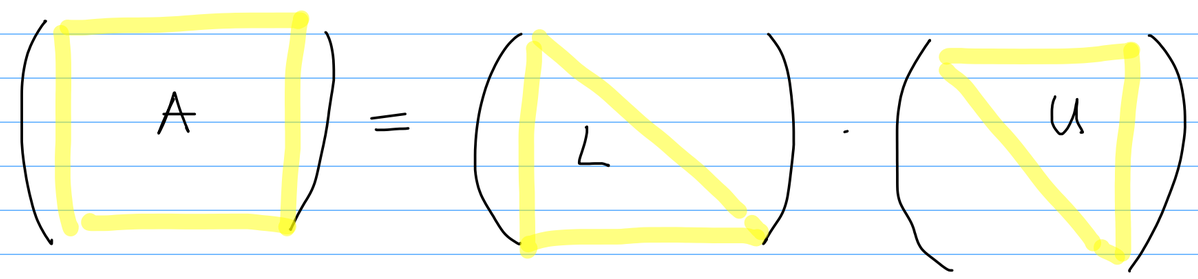
Gauss elimination: $\underbrace{(M_k \cdots M_1 A)}_{=:U \text{ upper triangular}} x = M_k \cdots M_1 b$

$$A = \underbrace{(M_1^{-1} \cdots M_k^{-1})}_{=:L} U$$

It turns out: L is lower triangular

$$L \cdot U = P \cdot A$$

P: permutation matrix (pivoting)



LU decomposition of A: $\Theta(n^3)$

forward substitution: Solve $Lz = b$: $\Theta(n^2)$

backward substitution: Solve $Ux = z$: $\Theta(n^2)$

Solve for $Ax = b_i$ $\quad i = 1, \dots, N$

Gauss elimination: $\Theta(N n^3)$

LU decomp.: $\Theta(n^3 + N n^2)$

Note:
- Use existing LSE solvers that are built-in
- If A s.p.d.
  $\rightarrow$ A admits factorization $A = LL^H$
  Cholesky factorization

2.3 Exploiting structure when solving LSEs

simple example: LSE with triangular matrix: $\Theta(n^2)$

2.3.1 Block elimination

Block matrix multiplication

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$$

Suppose we want to solve

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

A_{11} has a simple structure

$$\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 = \mathbf{b}_1 ,$$
$$\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 = \mathbf{b}_2 .$$

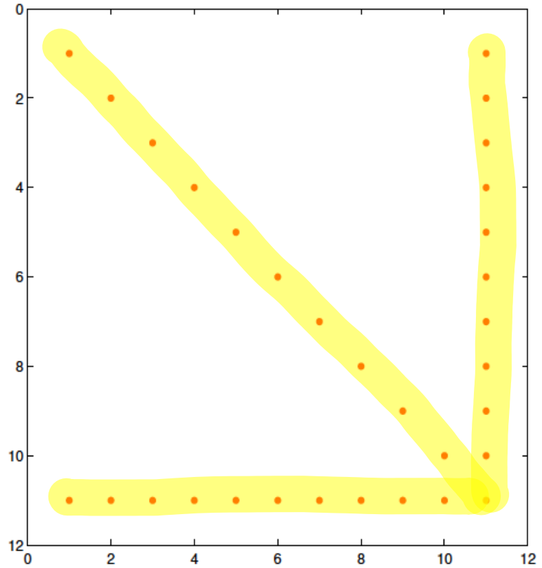
$$\mathbf{x}_1 = \mathbf{A}_{11}^{-1}(\mathbf{b}_1 - \mathbf{A}_{12}\mathbf{x}_2) ,$$
$$\underbrace{(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})}_{\textit{Schur complement}}\,\mathbf{x}_2 = \mathbf{b}_2 - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{b}_1 .$$

A_{11}^{-1} can be easily computed

this can be easy to solve

Example : arrow matrix

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{D} & \mathbf{c} \\ \hline \mathbf{b}^\top & \alpha \end{array}\right] \tag{2.9}$$



diagonal $\in \mathbb{R}^{n,n}$

$$\begin{bmatrix} D & c \\ b^T & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ \xi \end{bmatrix} = \begin{bmatrix} b_1 \\ \beta \end{bmatrix}$$

scalar

Solving for LU : $\mathcal{O}(n^3)$

$$\mathcal{D}x_1 + c\xi = b_1 \quad \in \mathbb{R}^n$$
$$b^T x_1 + \alpha\xi = \beta \quad \in \mathbb{R}$$

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{A}_{11}^{-1}(\mathbf{b}_1 - \mathbf{A}_{12}\mathbf{x}_2)\,, \\ \underbrace{(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})}_{\textit{Schur complement}}\,\mathbf{x}_2 &= \mathbf{b}_2 - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{b}_1\,. \end{aligned}$$

$$\begin{aligned} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 &= \mathbf{b}_1\,, \\ \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 &= \mathbf{b}_2\,. \end{aligned}$$

$$x_1 = \mathcal{D}^{-1}(b_1 - c\xi)$$

$$(\alpha - b^T\mathcal{D}^{-1}c)\,\xi = \beta - b^T\mathcal{D}^{-1}b_1$$

computing $\mathcal{D}^{-1}c$: $\mathcal{O}(n)$

$\mathcal{D}^{-1}b_1$: $\mathcal{O}(n)$

$$\xi = \frac{\beta - b^T\mathcal{D}^{-1}b_1}{\alpha - b^T\mathcal{D}^{-1}c}$$

total complexity: $\mathcal{O}(n)$

$x_1 = \mathcal{D}^{-1}(b_1 - c\xi)$: $\mathcal{O}(n)$

overall : $\mathcal{O}(n)$ complexity

Low-rank modification of an LSE

Application: Broyden's method for Newton accelleration

Task: Solve $Ax = b$ $\quad A \in \mathbb{R}^{n,n}_{reg}$

Then: Solve $\tilde{A}x = \tilde{b}$ $\quad A - \tilde{A}$ is low-rank

Simplest example: $A, \tilde{A}$ differ by <u>one</u> entry

$$\tilde{a}_{ij} = \begin{cases} a_{ij} & (i,j) \neq (i^*, j^*) \\ z + a_{ij} & (i,j) = (i^*, j^*) \end{cases}$$

$$\tilde{A} = A + z \cdot e_{i^*} e_{j^*}^T$$

$$e_{i^*} e_{j^*}^T = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & & & \\ 0 & & & \\ 1 & & 0 & \\ 0 & & & \\ 0 & & & \end{pmatrix}$$

in General: rank-1 modification

$$\tilde{A} = A + \underbrace{uv^T}_{\text{rank-1 matrix}} \qquad u, v \in \mathbb{R}^n \setminus \{0\}$$

Trick: Use block elimination on the block partioned system:

$$\underbrace{\begin{bmatrix} A & u \\ v^T & -1 \end{bmatrix}}_{(n+1)\times(n+1) \text{ matrix}} \begin{bmatrix} x \\ \xi \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \qquad (*)$$

Verify: (I) $Ax + u\xi = b$

(II) $v^T x - \xi = 0 \Rightarrow \xi = v^T x$

(I): $Ax + uv^T x = b$

$$(A + uv^T)x = b$$

Block eliminating x:

Rewrite (I) $\quad x = A^{-1}(b - u\xi)$

Plug into (II):

$$v^T A^{-1}(b - u\xi) - \xi = 0$$

$$\xi = \frac{v^T A^{-1} b}{v^T A^{-1} u + 1}$$

$$\Rightarrow \quad x = A^{-1} b - A^{-1} u \cdot \frac{v^T A^{-1} b}{1 + v^T (A^{-1} u)}$$

Solve $\quad A y_1 = u$

$\quad\quad\quad A y_2 = b$

If LU decomposition of A is available:

Additional cost of solving $\underset{\substack{\uparrow \\ A + uv^T}}{\tilde{A}} x = \tilde{b}$

is $\Theta(n^2)$ or like solving LSE with A for one more RHS

More generally: For rank-k perturbations:

Lemma 2.4.1 (Sherman-Morrison-Woodbury formula)
For regular $\mathbf{A} \in \mathbb{K}^{n,n}$, *and* $\mathbf{U}, \mathbf{V} \in \mathbb{K}^{n,k}$, $n, k \in \mathbb{N}$, $k \leq n$, *holds*

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^{\mathrm{H}})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\underbrace{\mathbf{I} + \mathbf{V}^{\mathrm{H}}\mathbf{A}^{-1}\mathbf{U}}_{I \in \mathbb{R}^{k,k} \text{ (identity)}})^{-1}\mathbf{V}^{\mathrm{H}}\mathbf{A}^{-1},$$

if $\mathbf{I} + \mathbf{V}^{\mathrm{H}}\mathbf{A}^{-1}\mathbf{U}$ *is regular.*

Note: $I + V^H A^{-1} U$ is $k \times k$

$\rightarrow$ for small k: solve for a small system

Under some conditions:

$$\mathrm{cond}(I + V^H A^{-1} U) \leq \mathrm{cond}(A) \cdot \mathrm{cond}(A + UV^H)$$

roughly: if original & perturbed system are well-conditioned then also $I + V^H A^{-1} U$ is.

If $\tilde{A} - A$ has only a few nonzero columns (or rows):

can compute factorization UV^H in $\Theta(nk^2)$

write $\tilde{A} - A$ as $n \times k$ (or $k \times n$ resp.)

and perform Householder QR $\rightsquigarrow$ later in this course

2.4 Sparse Linear Systems