Lecture 14
Stability

NLA Reading Group Spring ’13

by Işık Barış Fidaner

setup

a **mathematical problem** is a function:

$$f : X \rightarrow Y$$

an **algorithm** is a function:

$$\tilde{f} : X \rightarrow Y$$

where

X is the vector space of **data**

Y is the vector space of **solutions**

floating point approximation

fl error smaller than epsilon relative to x:

$$\text{For all } x \in \mathbb{R}, \text{ there exists } \epsilon \text{ with } |\epsilon| \leq \epsilon_{\text{machine}} \text{ such that } \mathrm{fl}(x) = x(1+\epsilon). \tag{13.5}$$

$\circledast$ is any floating point operation, $+$, $-$, $\times$, or $\div$

$$x \circledast y = \mathrm{fl}(x * y). \tag{13.6}$$

Fundamental Axiom of Floating Point Arithmetic

For all $x, y \in \mathbf{F}$, there exists ϵ with $|\epsilon| \leq \epsilon_{\text{machine}}$ such that

$$x \circledast y = (x * y)(1+\epsilon). \tag{13.7}$$

accuracy

absolute error: $\|\tilde{f}(x) - f(x)\|$

relative error: $\dfrac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|}$

the algorithm is **accurate**, if for each $x \in X$

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O(\epsilon_{\text{machine}})$$

"rel. error is on the order of machine epsilon"

stability

the algorithm is **stable**, if for each $x \in X$

$$\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = O(\epsilon_{\text{machine}})$$

for some $\tilde{x}$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{machine}}).$$

A stable algorithm gives nearly the right answer
to nearly the right question.

backward stability

the algorithm is **backward stable**, if for $x \in X$

$$\tilde{f}(x) = f(\tilde{x}) \quad \text{for some } \tilde{x} \text{ with } \quad \frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{machine}}).$$

simpler and stronger than stability.

A backward stable algorithm gives exactly the right answer to nearly the right question.

"on the order of machine epsilon"

mathematical notation:

$$\varphi(t) = O(\psi(t))$$

it means: when $t \to 0 \text{ or } t \to \infty$,

$$|\varphi(t)| \le C\psi(t).$$

if phi has an additional parameter *s*,

$$\varphi(s,t) = O(\psi(t)) \quad \text{uniformly in } s,$$

it means: there's a single *C* that holds for all *s*

in our case, *s*=*x* is the data vector, and $\epsilon_{\text{machine}} \to 0$

$$\|\text{computed quantity}\| = O(\epsilon_{\text{machine}}).$$

independence of norm

Theorem 14.1. *For problems f and algorithms $\tilde{f}$ defined on finite-dimensional spaces X and Y, the properties of accuracy, stability, and backward stability all hold or fail to hold independently of the choice of norms in X and Y.*

proof: for any $\|\cdot\|$ and $\|\cdot\|'$ on the same space,
there exists positive C_1,C_2 that for all x,

$$C_1\|x\| \leq \|x\|' \leq C_2\|x\|$$

norm changes the constant, not the order.

exercises

14.1. True or False?

(a) $\sin x = O(1)$ as $x \to \infty$.

(b) $\sin x = O(1)$ as $x \to 0$.

(c) $\log x = O(x^{1/100})$ as $x \to \infty$.

(d) $n! = O((n/e)^n)$ as $n \to \infty$.

(e) $A = O(V^{2/3})$ as $V \to \infty$, where A and V are the surface area and volume of a sphere measured in square microns and cubic miles, respectively.

(f) $\mathrm{fl}(\pi) - \pi = O(\epsilon_{\text{machine}})$. (We do not mention that the limit is $\epsilon_{\text{machine}} \to 0$, since that is implicit for all expressions $O(\epsilon_{\text{machine}})$ in this book.)

(g) $\mathrm{fl}(n\pi) - n\pi = O(\epsilon_{\text{machine}})$, uniformly for all integers n. (Here $n\pi$ represents the exact mathematical quantity, not the result of a floating point calculation.)

exercises

(a,b) true, sine is already less than constant

c) $x=y^{100}$ => $|100\log y| < Cy$ true since $\log y<y$

e) $A=4\ pi\ (mr)^2$ $V=4\ pi\ (r)^3\ /3$ true.

$V^{(2/3)} = 16\ pi^{(2/3)}\ r^2\ /9$ both $= C\ r^2$

exercises

14.2. (a) Show that $(1+O(\epsilon_{\text{machine}}))(1+O(\epsilon_{\text{machine}})) = 1+O(\epsilon_{\text{machine}})$. The precise meaning of this statement is that if f is a function satisfying $f(\epsilon_{\text{machine}}) = (1+O(\epsilon_{\text{machine}}))(1+O(\epsilon_{\text{machine}}))$ as $\epsilon_{\text{machine}} \to 0$, then f also satisfies $f(\epsilon_{\text{machine}}) = 1+O(\epsilon_{\text{machine}})$ as $\epsilon_{\text{machine}} \to 0$.

(b) Show that $(1+O(\epsilon_{\text{machine}}))^{-1} = 1+O(\epsilon_{\text{machine}})$.

a) $|x(e)|<Ce$, $|y(e)|<Ce$ $(1+x)(1+y)=1+xy+x+y$
$|xy+x+y| < 2Ce+e^2 < 3Ce$

b) $|x|<Ce \Rightarrow 1/(1+x) = 1 - x/(1+x)$
$|-x/(1+x)|<Ce \Rightarrow 1/(1+x) = 1 + O(e)$

Lecture 18
Conditioning of Least Squares Problems

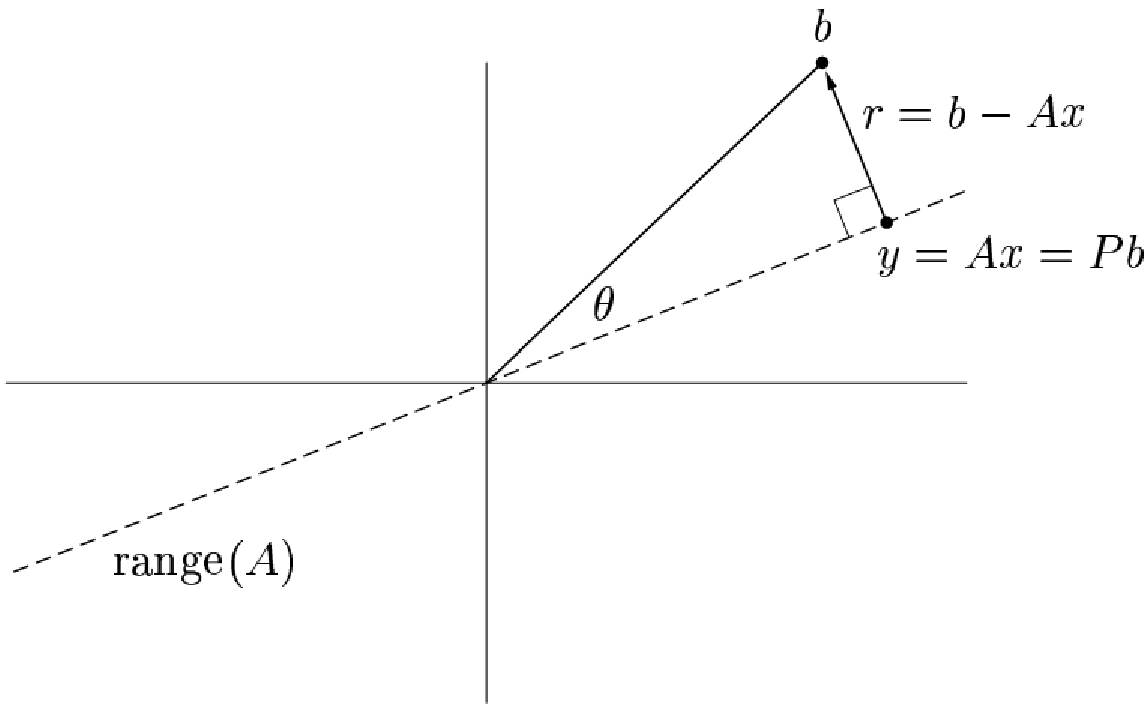
NLA Reading Group Spring ’13

by Işık Barış Fidaner

least squares problem

problem: Given $A \in \mathbb{C}^{m \times n}$ of full rank, $m \geq n$, $b \in \mathbb{C}^m$, find $x \in \mathbb{C}^n$ such that $\|b - Ax\|$ is minimized.

solution: $x = A^+ b, \qquad y = Pb,$



three measures

•condition number of A:

$$\kappa(A) \;=\; \|A\|\,\|A^+\| \;=\; \frac{\sigma_1}{\sigma_n}$$

•angle, closeness of the fit:

$$\theta = \cos^{-1} \frac{\|y\|}{\|b\|}$$

•how much y falls short of its maximum value:

$$\eta \;=\; \frac{\|A\|\,\|x\|}{\|y\|} \;=\; \frac{\|A\|\,\|x\|}{\|Ax\|}$$

•their ranges:

$$1 \leq \kappa(A) < \infty, \qquad 0 \leq \theta \leq \pi/2, \qquad 1 \leq \eta \leq \kappa(A)$$

sensitivities of x,y

Theorem 18.1. *Let $b \in \mathbb{C}^m$ and $A \in \mathbb{C}^{m\times n}$ of full rank be fixed. The least squares problem* (18.1) *has the following 2-norm relative condition numbers* (12.5) *describing the sensitivities of y and x to perturbations in b and A:*

	y	x
b	$\dfrac{1}{\cos\theta}$	$\dfrac{\kappa(A)}{\eta\cos\theta}$
A	$\dfrac{\kappa(A)}{\cos\theta}$	$\kappa(A) + \dfrac{\kappa(A)^2\tan\theta}{\eta}$

The results in the first row are exact, being attained for certain perturbations δb, and the results in the second row are upper bounds.

proof, step 1

$$A = U\Sigma V^*$$

unitary change of basis does not affect the perturbations in 2-norm

assume $A = \Sigma$ and write:

$$A = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \end{bmatrix}.$$

proof, step 0

$$A = U\Sigma V^*$$

unitary change of basis does not affect the perturbations in 2-norm

assume $A = \Sigma$ and write:

$$A = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \end{bmatrix}.$$

as a result, orthogonal projector and pseudoinverse become

$$P = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \qquad A^+ = \begin{bmatrix} A_1^{-1} & 0 \end{bmatrix}.$$

proof, step 1: sensitivity of **y** to perturbations in **b**

$$y = Pb.$$

apply condition number formula:

$$\kappa_{b \mapsto y} = \frac{\|P\|}{\|y\|/\|b\|} = \frac{1}{\cos\theta}.$$

proof, step 2: sensitivity of **x** to perturbations in **b**

$$x = A^+ b$$

apply condition number formula:

$$\kappa_{b \mapsto x} = \frac{\|A^+\|}{\|x\|/\|b\|} = \|A^+\| \frac{\|b\|}{\|y\|} \frac{\|y\|}{\|x\|} = \|A^+\| \frac{1}{\cos\theta} \frac{\|A\|}{\eta} = \frac{\kappa(A)}{\eta\cos\theta}.$$

proof, step 2.5: tilting the range of A

when A is perturbed

(1) either range(A) is tilted by $\delta\alpha$

(2) or mapping onto range(A) is changed

1) what is maximum $\delta\alpha$?

let v be a point on unit sphere $\|v\| = 1$. $p = Av$ is on range(A)

to tilt range(A) maximally, we move p orthogonal to range(A).

$$\delta A = (\delta p) v^* \quad \Rightarrow \quad \|\delta A\| = \|\delta p\| \qquad ((\delta A) v = \delta p)$$

take the smallest eigenvalue $p = \sigma_n u_n$ => tilt angle: $\tan(\delta\alpha) = \|\delta p\| / \sigma_n$

$$\delta\alpha \le \tan(\delta\alpha) \quad \Longrightarrow \quad \delta\alpha \le \frac{\|\delta A\|}{\sigma_n} = \frac{\|\delta A\|}{\|A\|}\kappa(A),$$

equality only attained by infinitesimal angles.

proof, step 3: sensitivity of **y** to perturbations in **A**

y is a projection. it is determined only by b and range(A)

fix b and tilt range(A) by $\delta\alpha$. 0-y and 0-b are orthogonal: y on sphere

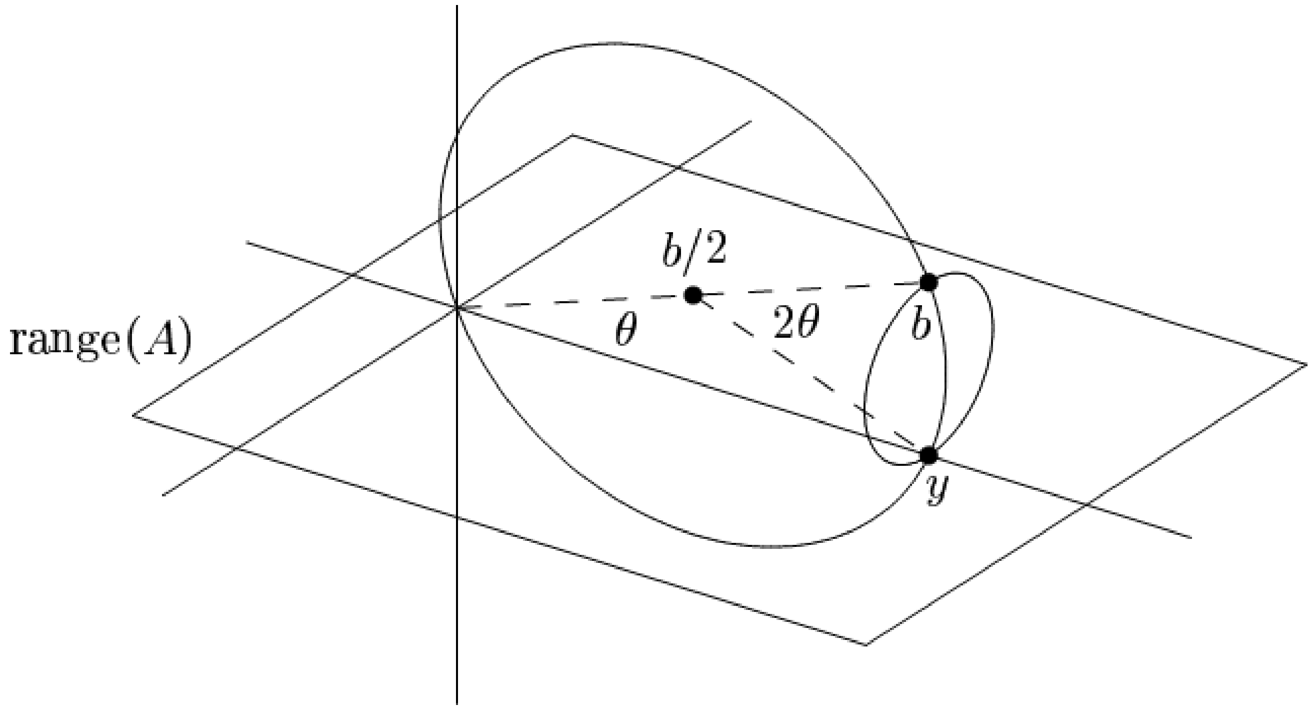


Figure 18.2. *Two circles on the sphere along which y moves as* range(A) *varies. The large circle, of radius $\|b\|/2$, corresponds to tilting* range(A) *in the plane 0–b–y, and the small circle, of radius $(\|b\|/2)\sin\theta$, corresponds to tilting it in an orthogonal direction. However* range(A) *is tilted, y remains on the sphere of radius $\|b\|/2$ centered at $b/2$.*

proof, step 3: sensitivity of **y** to perturbations in **A**

large circle implies

$$\|\delta y\| \ \le \ \|b\| \sin(\delta\alpha) \ \le \ \|b\|\delta\alpha$$

definition of angle θ and upper bound of $\delta\alpha$ gives:

$$\|\delta y\| \le \|\delta A\| \kappa(A) \|y\| / \|A\| \cos\theta$$

thus, the sensitivity of y to A is:

$$\frac{\|\delta y\|}{\|y\|} \bigg/ \frac{\|\delta A\|}{\|A\|} \ \le \ \frac{\kappa(A)}{\cos\theta}$$

proof, step 4: sensitivity of **x** to perturbations in **A**

split the perturbation of A into 1 and 2

$$\delta A = \begin{bmatrix} \delta A_1 \\ \delta A_2 \end{bmatrix} = \begin{bmatrix} \delta A_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \delta A_2 \end{bmatrix}$$

perturbation 1 does not change range(A), only the mapping onto it.

$$\frac{\|\delta x\|}{\|x\|} \Big/ \frac{\|\delta A_1\|}{\|A\|} \leq \kappa(A_1) = \kappa(A)$$

perturbation 2 tilts range(A) without changing the mapping onto it.

in terms of b1:

$$\frac{\|\delta x\|}{\|x\|} \Big/ \frac{\|\delta b_1\|}{\|b_1\|} \leq \frac{\kappa(A_1)}{\eta(A_1; x)} = \frac{\kappa(A)}{\eta}$$

now we need to replace denominator db1/b1 with dA2/A

proof, step 4: sensitivity of **x** to perturbations in **A**

when range(A) is tilted through the larger circle,

angle between δy and range(A) = $\pi/2 - \theta$

=> $\|\delta b_1\| = \sin\theta \|\delta y\|$ => $\|\delta b_1\| \leq (\|b\|\delta\alpha)\sin\theta$

when range(A) is tilted through the smaller circle,

y is parallel to range(A), but it is a factor of $\sin\theta$ smaller

=> $\|\delta y\| \leq (\|b\|\delta\alpha)\sin\theta$ => $\|\delta b_1\| \leq (\|b\|\delta\alpha)\sin\theta$ ($\|\delta b_1\| \leq \|\delta y\|$)

rewrite:

($\|b_1\| = \|b\|\cos\theta$) $$\frac{\|\delta b_1\|}{\|b_1\|} \leq (\delta\alpha)\tan\theta$$

upper bound of $\delta\alpha$ and eqn. of perturbation 2 gives

$$\frac{\|\delta x\|}{\|x\|} \Big/ \frac{\|\delta A_2\|}{\|A\|} \leq \frac{\kappa(A)^2\tan\theta}{\eta}$$

add this to result from perturbation 1, done.

Lecture 19
Stability of Least Squares Algorithms

NLA Reading Group Spring ’13

by Işık Barış Fidaner

accuracy of a backward stable alg.

Theorem 15.1. *Suppose a backward stable algorithm is applied to solve a problem $f : X \to Y$ with condition number κ on a computer satisfying the axioms (13.5) and (13.7). Then the relative errors satisfy*

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O(\kappa(x)\,\epsilon_{\text{machine}}).$$

above: x is data, f(x) is solution.

below: A is data, x is solution ($\kappa = \kappa(A)$)

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O\left(\left(\kappa + \frac{\kappa^2 \tan\theta}{\eta}\right)\epsilon_{\text{machine}}\right)$$

condition number lies in range κ *to* κ^2

example algorithm

task:

least squares fitting of the function $\exp(\sin(4\tau))$ on the interval $[0, 1]$ by a polynomial of degree 14.

code:

`m = 100; n = 15;`	
`t = (0:m-1)'/(m-1);`	Set t to a discretization of $[0, 1]$.
`A = []; for i=1:n,`	Construct Vandermonde matrix.
`  A = [A t.^(i-1)]; end`	
`b = exp(sin(4*t));`	Right-hand side.
`b = b/2006.787453080206;`	Normalization (see text).

after normalization, x(15)=1 is ground truth.

example algorithm: householder

compute the three measures

```
x = A\b; y = A*x;
kappa = cond(A)
    kappa = 2.2718e+10
theta = asin(norm(b-y)/norm(b))
    theta = 3.7461e-06
eta = norm(A)*norm(x)/norm(y)
    eta = 2.1036e+05
```

Solve least squares problem.

$\kappa(A)$

θ

η

kappa: "ill-conditioned basis" theta: "close fit"

eta: "y is around half of the maximum kappa"

example algorithm: householder

compute sensitivities/condition nrs of x and y:

	y	x
b	1.0	1.1×10^5
A	2.3×10^{10}	3.2×10^{10}

IEEE double precision arithmetic

$\epsilon_{\text{machine}} \approx 10^{-16}$

standard algorithm for solving least squares:

```
[Q,R] = qr(A,0);          Householder triang. of A.
x = R\(Q'*b);             Solve for x.
x(15)
   ans = 1.00000031528723
```

relative error of about 3×10^{-7}

rounding errors amplified by 10^9

condition number of x with respect to perturbations in A is of order 10^{10}

Algorithm appears to be backward stable.

example algorithm: householder2

alternative algorithm that computes Q*b

```
[Q,R] = qr([A b],0);      Householder triang. of [A  b].
Qb = R(1:n,n+1);          Extract Q̂*b ...
R = R(1:n,1:n);             ... and R̂.
x = R\Qb;                 Solve for x.
x(15)
   ans = 1.00000031529465
```

gives similar error, therefore error from QR *swamps* the error from Q*b computation

example algorithm: householder3

matlab implementation of householder

```
x = A\b;                    Solve for x.
x(15)
   ans = 0.99999994311087
```

more accurate, uses column pivoting

all three methods are backward stable.

householders are backward stable

Theorem 19.1. *Let the full-rank least squares problem* (11.2) *be solved by Householder triangularization (Algorithm* 11.2*) on a computer satisfying* (13.5) *and* (13.7)*. This algorithm is backward stable in the sense that the computed solution $\tilde{x}$ has the property*

$$\|(A+\delta A)\tilde{x}-b\| = \min, \qquad \frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{machine}}) \tag{19.1}$$

for some $\delta A \in \mathbb{C}^{m\times n}$. This is true whether $\hat{Q}^*b$ is computed via explicit formation of $\hat{Q}$ or implicitly by Algorithm 10.2. *It also holds for Householder triangularization with arbitrary column pivoting.*

"solving for A in fact solves for A+dA"

example: gram-schmidt

```
[Q,R] = mgs(A);          Gram–Schmidt orthog. of A.
x = R\(Q'*b);            Solve for x.
x(15)
   ans = 1.02926594532672
```

result is very poor, because
GS produces Q with non-orthonormal columns

reformulate problem, becomes complicated.

example: gram-schmidt2

better method, similar to householder2:

`[Q,R] = mgs([A b]);`	Gram–Schmidt orthog. of $[A \;\; b]$.
`Qb = R(1:n,n+1);`	Extract $\hat{Q}^* b$ ...
`R = R(1:n,1:n);`	... and $\hat{R}$.
`x = R\Qb;`	Solve for x.
`x(15)`	
`ans = 1.00000005653399`	

Theorem 19.2. *The solution of the full-rank least squares problem* (11.2) *by Gram–Schmidt orthogonalization is also backward stable, satisfying* (19.1), *provided that $\hat{Q}^* b$ is formed implicitly as indicated in the code segment above.*

solving by normal equations

```
x = (A'*A)\(A'*b);
x(15)
   ans = 0.39339069870283
```

Form and solve normal equations.

clearly unstable.

the matrix A^*A has condition number κ^2, not κ.

best we can expect is:

$$\frac{\|\tilde{x}-x\|}{\|x\|} = O(\kappa^2 \epsilon_{\text{machine}})$$

Theorem 19.3. *The solution of the full-rank least squares problem* (11.2) *via the normal equations (Algorithm* 11.1*) is unstable. Stability can be achieved, however, by restriction to a class of problems in which* $\kappa(A)$ *is uniformly bounded above or* $(\tan\theta)/\eta$ *is uniformly bounded below.*

solving by SVD

```
[U,S,V] = svd(A,0);          Reduced SVD of A.
x = V*(S\(U'*b));            Solve for x.
x(15)
   ans = 0.99999998230471
```

best result. 3 digits better than householder3

Theorem 19.4. *The solution of the full-rank least squares problem* (11.2) *by the SVD (Algorithm* 11.3*) is backward stable, satisfying the estimate* (19.1).

general result:

- householder2 is the cheapest,
- SVD is the most accurate.

exercises

19.1. Given $A \in \mathbb{C}^{m\times n}$ of rank n and $b \in \mathbb{C}^m$, consider the block 2×2 system of equations

$$\begin{bmatrix} I & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{19.4}$$

where I is the $m \times m$ identity. Show that this system has a unique solution $(r, x)^T$, and that the vectors r and x are the residual and the solution of the least squares problem (18.1).

r+Ax=b   A*r=0

=> A*r+A*Ax=A*b => A*Ax=A*b => x: solution

=> Ax=b-r => A*Ax=A*(b-r) => r: residual

exercises

19.2. Here is a stripped-down version of one of MATLAB's built-in *m*-files.

```
[U,S,V] = svd(A);
S = diag(S);
tol = max(size(A))*S(1)*eps;
r = sum(S > tol);
S = diag(ones(r,1)./S(1:r));
X = V(:,1:r)*S*U(:,1:r)';
```

What does this program compute?

tol is a tolerance to disregard small singular values. r is the rank of A. first r singular values are chosen, and X is the approx inverse of A.