Chapter 8: Factorizations and Sensitivity

1. Apply Gaussian elimination to obtain triangular system:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix} \qquad \text{ca. } \tfrac{1}{3}n^3 \text{ LOps}$$

2. Back substitution

$$\begin{aligned} x_n &= b_n / a_{nn} \\ x_{n-1} &= (b_{n-1} - a_{n-1,n} \cdot x_n) / a_{n-1,n-1} \qquad \text{ca. } \tfrac{1}{2}n^2 \text{ LOps} \\ &\vdots \end{aligned}$$

LU factorization

Another formulation of Gaussian elimination with pivoting:

$$\boldsymbol{P}\,\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U},$$

where

- $\boldsymbol{P}$ is a permutation matrix,
- $\boldsymbol{U}$ is an upper triangular *after* Gaussian eliminationen, and
- $\boldsymbol{L}$ is a unit lower triangular including elimination coefficients:

$$\ell_{ik} = f_{ik}, \quad k < i \qquad \text{and} \qquad \ell_{kk} = 1.$$

When we have LU factorization of $\boldsymbol{A}$, the solution of the linear system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ equals $\boldsymbol{x} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{P}\,\boldsymbol{b}$, and it only costs n^2 LOps to obtain $\boldsymbol{x}$.

LDL^T factorization (page 367)

If the matrix $\boldsymbol{A}$ is symmetric, then we have

$$\boldsymbol{L}\,\boldsymbol{U} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{L}\,\boldsymbol{U})^T = \boldsymbol{U}^T\,\boldsymbol{L}^T.$$

LDL^T factorization (page 367)

If the matrix $\boldsymbol{A}$ is symmetric, then we have

$$\boldsymbol{L}\,\boldsymbol{U} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{L}\,\boldsymbol{U})^T = \boldsymbol{U}^T\,\boldsymbol{L}^T.$$

Since $\boldsymbol{L}$ is unit lower triangular, it's invertible, then

$$\boldsymbol{U} = \boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{L}^T$$

LDL^T factorization (page 367)

If the matrix $\boldsymbol{A}$ is symmetric, then we have

$$\boldsymbol{L}\,\boldsymbol{U} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{L}\,\boldsymbol{U})^T = \boldsymbol{U}^T\,\boldsymbol{L}^T.$$

Since $\boldsymbol{L}$ is unit lower triangular, it's invertible, then

$$\boldsymbol{U} = \boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{L}^T \quad\Longrightarrow\quad \boldsymbol{U}\,(\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-1}\boldsymbol{U}^T.$$

LDL^T factorization (page 367)

If the matrix $\boldsymbol{A}$ is symmetric, then we have

$$\boldsymbol{L}\,\boldsymbol{U} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{L}\,\boldsymbol{U})^T = \boldsymbol{U}^T\,\boldsymbol{L}^T.$$

Since $\boldsymbol{L}$ is unit lower triangular, it's invertible, then

$$\boldsymbol{U} = \boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{L}^T \quad \Longrightarrow \quad \boldsymbol{U}\,(\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-1}\boldsymbol{U}^T.$$

Since the left-hand side is upper triangular and the right-hand side is lower triangular, both sides have to be diagonal, say, D. Then, we have

$$\boldsymbol{U}\,(\boldsymbol{L}^T)^{-1} = \mathrm{D}$$

LDL^T factorization (page 367)

If the matrix $\boldsymbol{A}$ is symmetric, then we have

$$\boldsymbol{L}\,\boldsymbol{U} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{L}\,\boldsymbol{U})^T = \boldsymbol{U}^T\,\boldsymbol{L}^T.$$

Since $\boldsymbol{L}$ is unit lower triangular, it's invertible, then

$$\boldsymbol{U} = \boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{L}^T \quad \Longrightarrow \quad \boldsymbol{U}\,(\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-1}\boldsymbol{U}^T.$$

Since the left-hand side is upper triangular and the right-hand side is lower triangular, both sides have to be diagonal, say, D. Then, we have

$$\boldsymbol{U}\,(\boldsymbol{L}^T)^{-1} = \mathrm{D} \quad \Longrightarrow \quad \boldsymbol{U} = \mathrm{D}\,\boldsymbol{L}^T \quad \Longrightarrow \quad \boldsymbol{A} = \boldsymbol{L}\,\mathrm{D}\,\boldsymbol{L}^T.$$

Cholesky factorization (page 369–370)

The Matrix $\boldsymbol{A}$ is *symmetric* and *positive definite* (SPD) when

$$\boldsymbol{A}^T = \boldsymbol{A} \qquad \text{and} \qquad \boldsymbol{x}^T \boldsymbol{A}\,\boldsymbol{x} > 0 \quad \text{for all } \boldsymbol{x} \neq 0.$$

An important example: normal equation in data-fitting.

Due to symmetry, we have $\boldsymbol{A} = \boldsymbol{L}\,\mathrm{D}\,\boldsymbol{L}^T$. Further, $\boldsymbol{A}$ is SPD, so D has a positive diagonal. Then, we have

$$\boldsymbol{A} = \boldsymbol{L}\sqrt{\mathrm{D}}\sqrt{\mathrm{D}}^T\boldsymbol{L}^T = \boldsymbol{L}\,\boldsymbol{L}^T\ , \qquad \boldsymbol{L} = \boldsymbol{L}\sqrt{\mathrm{D}}$$

This is called as Cholesky factorization.

NB: In Cholesky factorization $\boldsymbol{L}$ is lower triangular with a positive diagonal, which is different as $\boldsymbol{L}$ in LU factorization!

We can derive the elements in $\boldsymbol{L}$ by comparing the elements in $\boldsymbol{L}\,\boldsymbol{L}^T$ and $\boldsymbol{A}$. For example, in a 2×2 matrix:

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{21} \\ 0 & \ell_{22} \end{bmatrix} = \begin{bmatrix} \ell_{11}^2 & \ell_{11}\ell_{21} \\ \ell_{11}\ell_{21} & \ell_{21}^2 + \ell_{22}^2 \end{bmatrix} \quad \Rightarrow$$

$$\ell_{11} = \sqrt{a_{11}}, \qquad \ell_{21} = a_{21}/\ell_{11}, \qquad \ell_{22} = \sqrt{a_{22} - {\ell_{21}}^2}\ .$$

Python-code for Cholesky factorization

This code is a bit different from the one in page 370:

```
for k in range(n):
    A[k, k] = np.sqrt(A[k, k])
    A[k+1:n, k] = A[k+1:n, k] / A[k, k]
    for j in range(k+1, n):
        A[j:n, j] = A[j:n, j] - A[j, k]*A[j:n, k]
L = np.tril(A) # lower triangle of A
```

Pivoting is not necessary in Cholesky factorization!

The algorithm overwrites the current element in $\boldsymbol{A}$, which is saved in $\boldsymbol{L}$ in the end.

Only the lower triangular part of $\boldsymbol{A}$ is needed – so in the end we only need save a part of the matrix.

$\boxed{\text{Operation count} \approx 1/6\, n^3\text{, i.e. } \textit{only half} \text{ as in LU factorization.}}$

Cholesky factorization in Python with `np.linalg.cholesky`

```
>>> A = np.array([[6, 15, 55],
                  [15, 55, 225],
                  [55, 225, 979]], dtype=float)
>>> L = np.linalg.cholesky(A)
>>> print(L)

L =
[[ 2.44948974 0. 0. ]
[ 6.12372436 4.18330013 0. ]
[22.45365598 20.91650066 6.11010093]]
```

Solve linear systems with Cholesky factorization

$$
\begin{aligned}
\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad &\Rightarrow \quad \boldsymbol{L}\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{b} \\
&\Rightarrow \quad \boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{L}^{-1}\boldsymbol{b} \\
&\Rightarrow \quad \boldsymbol{x} = (\boldsymbol{L}^T)^{-1}\left(\boldsymbol{L}^{-1}\boldsymbol{b}\right)
\end{aligned}
$$

Solve linear systems with Cholesky factorization

$$
\begin{aligned}
\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad &\Rightarrow \quad \boldsymbol{L}\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{b} \\
&\Rightarrow \quad \boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{L}^{-1}\boldsymbol{b} \\
&\Rightarrow \quad \boldsymbol{x} = (\boldsymbol{L}^T)^{-1}(\boldsymbol{L}^{-1}\boldsymbol{b})
\end{aligned}
$$

In Python, there are two ways to implement this procedure:

```
>>> tmp = scipy.linalg.solve_triangular(L,b,lower=True)
>>> x = scipy.linalg.solve_triangular(L.T, tmp, lower=False)
```

OR

```
>>> c, low = scipy.linalg.cho_factor(A)
>>> x = scipy.linalg.cho_solve((c, low), b)
```

Sensitivity analysis for $f(x) = 0$ (Motivation)

$\boxed{\text{How much does the root change, when the function } f \text{ is perturbed?}}$

Relevant to: when f is described by noisy data – in this case, is the root affected by the data error?

Sensitivity analysis/error assessment:
How sensitive is the root to the perturbation of the function f?

Error assessment of a simple root

When ξ is close to x^* we can always assume that $0 < M \leq |f'(\xi)|$.

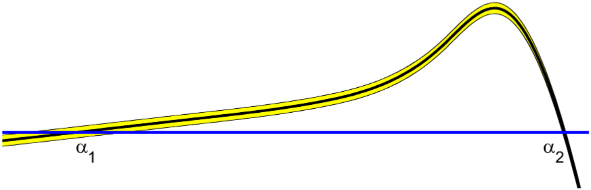
We want to calculate the root x^* of f, but we only have a *perturbed* function $\tilde{f}$ and can calculate its root $\tilde{x}^*$. Assume that when ξ is close to x^* we have

$$|\widetilde{f}(\xi) - f(\xi)| \leq \delta \ ,$$

where δ is an upper bound for the absolute error of the function values. So we can obtain the upper bound for the absolute error:

$$|\tilde{x}^* - x^*| \leq \delta / M.$$

Smaller M (lower bound of the slope) is, more sensitive the root is to the error in $\tilde{f}$:



Sensitivity analysis for $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

The rest of the lecture is about, how the solution $\boldsymbol{x}$ of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is affected by the perturbation in the data, i.e. the error in the right-hand side.

How much does $\boldsymbol{x}$ change, when the right-hand side $\boldsymbol{b}$ is perturbed?

Example: the right-hand side $\tilde{\boldsymbol{b}}$ includes measurement error

$$\tilde{\boldsymbol{b}} = \boldsymbol{b} + \text{error} = \begin{bmatrix} 1.3 \\ 2.4 \\ 3.5 \end{bmatrix} = \begin{bmatrix} 1.2 \\ 2.5 \\ 3.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \\ 0.2 \end{bmatrix}.$$

How can we measure the difference between $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ and $\tilde{\boldsymbol{x}} = \boldsymbol{A}^{-1}\tilde{\boldsymbol{b}}$.

Content:

1. Vector and matrix norm.
2. Sensitivity analysis and condition number.

Review: absolute and relative error (page 5–6)

Absolute error

When we say the error in a is ± 0.01, we mean

$$-0.01 \leq a - \bar{a} \leq 0.01, \qquad \bar{a} = \text{exact value.}$$

Relative error

If we say a has a *relative* error 0.01 (or 1%), we mean

$$\frac{|a - \bar{a}|}{|\bar{a}|} \leq 0.01 \ .$$

But what can we say when we talk about the error in the vectors $\boldsymbol{x}$ and $\boldsymbol{b}$?

Vector norm (page 405)

For a vector, we define vector norm:

$$\|\boldsymbol{x}\|_2 = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{1/2}.$$

It is in fact an Euclidean distance in 2 and 3 dimensions.

Example:

$$\boldsymbol{x} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}, \qquad \|\boldsymbol{x}\|_2 \approx 3.74 \qquad \text{Python: } \texttt{np.linalg.norm(x)}.$$

We can show that: $\max_i |x_i| \le \|\boldsymbol{x}\|_2 \le \sqrt{n}\, \max_i |x_i|$.

- If $\|\boldsymbol{x}\|_2$ is large, $\boldsymbol{x}$ has at least one large element (because $\max_i |x_i| \ge \|\boldsymbol{x}\|_2/\sqrt{n}$).
- If $\|\boldsymbol{x}\|_2$ is small, $\boldsymbol{x}$ only has small elements (because $\max_i |x_i| \le \|\boldsymbol{x}\|_2$).

An application of norm: error in vectors

Assume that $\tilde{\boldsymbol{x}}$ is an approximation of $\boldsymbol{x}$. With the difference

$$\delta\boldsymbol{x} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$$

it is not practical to be used for measuring the error, since we need observe all elements in the vector $\delta\boldsymbol{x}$.

By using the vector norm, we can define the **absolute error**

$$\|\delta\boldsymbol{x}\|_2 = \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2$$

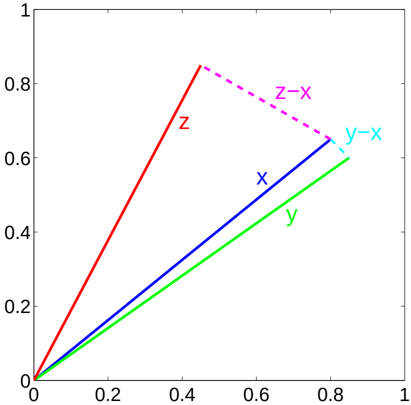
and the **relative error**

$$\frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}.$$

If the norm $\|\delta\boldsymbol{x}\|_2$ is small, then all elements in $\delta\boldsymbol{x} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$ are small, then we can say that $\tilde{\boldsymbol{x}}$ is close to $\boldsymbol{x}$. $\rightarrow$ In the end of the lecture, we will show how to *estimate* these errors without known $\boldsymbol{x}$.

Example to use vector norm

Which vector is close to $\boldsymbol{x}$? $\boldsymbol{y}$ or $\boldsymbol{z}$?



Relative error: $\|\boldsymbol{y}-\boldsymbol{x}\|_2/\|\boldsymbol{x}\|_2 = 0.07$ and $\|\boldsymbol{z}-\boldsymbol{x}\|_2/\|\boldsymbol{x}\|_2 = 0.20$.

Matrix norm (page 406)

We can also define the 2-norm for a matrix to measure its "magnitude":

$$\|\boldsymbol{A}\|_2 = \sup\big\{\|\boldsymbol{Ax}\|_2 : \boldsymbol{x} \in \mathbb{R}^n,\ \|\boldsymbol{x}\|_2 = 1\big\} \tag{1}$$

$$= \big(\text{the largest eigenvalue of } \boldsymbol{A}^T\boldsymbol{A}\big)^{1/2} = \texttt{np.linalg.norm(A)} \tag{2}$$

Example:

$$\boldsymbol{A} = \begin{bmatrix} 1 & -0.01 & 0 \\ 0.02 & -5 & 3 \\ -4 & 1 & -0.01 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 0.01 & -0.02 & 0 \\ 0.03 & -0.01 & 0.05 \\ -0.04 & 0 & 0.03 \end{bmatrix}$$

$$\|\boldsymbol{A}\|_2 = 5.95\ , \qquad \|\boldsymbol{B}\|_2 = 0.060\ .$$

If the norm of $\|\boldsymbol{B}\|_2$ is small, then we can conclude that there are only small elements in $\boldsymbol{B}$.

Continue: vector and matrix norm (page 405–406)

Some important properties of vector and matrix norm:

$$\|\boldsymbol{x}\|_2 = 0 \quad \Leftrightarrow \quad \boldsymbol{x} = 0\ , \qquad \|\boldsymbol{A}\|_2 = 0 \quad \Leftrightarrow \quad \boldsymbol{A} = 0$$

$$\|\alpha\,\boldsymbol{x}\|_2 = |\alpha|\,\|\boldsymbol{x}\|_2\ , \qquad \|\alpha\,\boldsymbol{A}\|_2 = |\alpha|\,\|\boldsymbol{A}\|_2$$

$$\|\boldsymbol{x}+\boldsymbol{y}\|_2 \le \|\boldsymbol{x}\|_2 + \|\boldsymbol{y}\|_2\ , \qquad \|\boldsymbol{A}+\boldsymbol{B}\|_2 \le \|\boldsymbol{A}\|_2 + \|\boldsymbol{B}\|_2\ .$$

Very important properties:

$$\|\boldsymbol{A}\,\boldsymbol{x}\|_2 \le \|\boldsymbol{A}\|_2\,\|\boldsymbol{x}\|_2\ , \qquad \|\boldsymbol{A}\,\boldsymbol{B}\|_2 \le \|\boldsymbol{A}\|_2\,\|\boldsymbol{B}\|_2\ .$$

Any properties with "$-$"?

$$\|\boldsymbol{y} + (\boldsymbol{x}-\boldsymbol{y})\|_2 \le \|\boldsymbol{y}\|_2 + \|\boldsymbol{x}-\boldsymbol{y}\|_2 \Longrightarrow \|\boldsymbol{x}\|_2 - \|\boldsymbol{y}\|_2 \le \|\boldsymbol{x}-\boldsymbol{y}\|_2$$

Python example with perturbation in the right-hand side

```
True solution x, perturbation solution xt, and x-xt:
   5.5090e+00   5.4010e+00   1.0810e-01
   3.2564e+00   3.3586e+00   1.0214e-01
   1.6696e+00   1.6794e+00   9.8571e-03

||b-bt||_2:
   1.5684e+00

||x-xt||_2:
   1.4905e-01
```

Is there any relation between $\|\boldsymbol{b} - \tilde{\boldsymbol{b}}\|_2$ and $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2$?

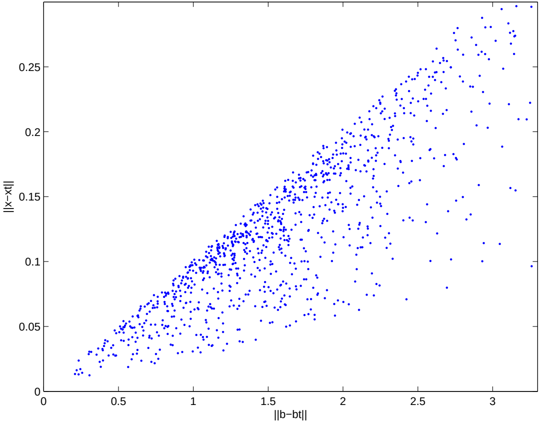
Let's test on 1000 different perturbation on $\boldsymbol{b}$ $\rightarrow$ next slide.

Python example with perturbation in the right-hand side

```
import numpy as np
import matplotlib.pyplot as plt
E = np.empty(1000)
D = np.empty(1000)
for i in range(1000):
    e = np.random.randn(3) # vector with 3 samples from N(0,1)
    E[i] = np.linalg.norm(e)
    d = np.linalg.solve(A, b) - np.linalg.solve(A, b+e)
    D[i] = norm(d)
end
```

Python example with perturbation in the right-hand side

```
plt.figure()
plt.plot(E, D, linewidth=0, marker='.')
plt.show()
```



Analysis of a perturbation in the right-hand side (page 407)

Let $\boldsymbol{x}$ denote the solution of $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$.

Now we *perturb* the right-hand side with a vector $\delta\boldsymbol{b}$ and obtain $\tilde{\boldsymbol{b}} = \boldsymbol{b} + \delta\boldsymbol{b}$, and compute

$$\boldsymbol{A}\,\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}} \quad \Rightarrow \quad \tilde{\boldsymbol{x}} = \boldsymbol{A}^{-1}\tilde{\boldsymbol{b}}.$$

We want to measure the error $\delta\boldsymbol{x} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$ in the solution, i.e. find the relation between $\|\delta\boldsymbol{x}\|_2$ and the perturbation $\|\delta\boldsymbol{b}\|_2$. We can easily get

$$\boldsymbol{A}\,\delta\boldsymbol{x} = \boldsymbol{A}\,(\tilde{\boldsymbol{x}} - \boldsymbol{x}) = \boldsymbol{A}\,\tilde{\boldsymbol{x}} - \boldsymbol{A}\,\boldsymbol{x} = \tilde{\boldsymbol{b}} - \boldsymbol{b} = \delta\boldsymbol{b} \quad \Leftrightarrow \quad \delta\boldsymbol{x} = \boldsymbol{A}^{-1}\delta\boldsymbol{b}.$$

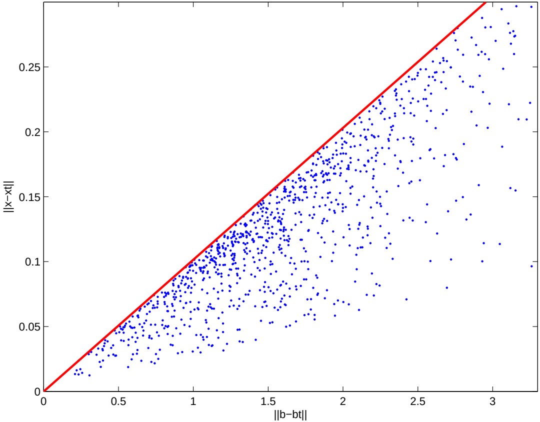
Now based on the properties of vector and matrix norm, we obtain

$$\|\delta\boldsymbol{x}\|_2 = \|\boldsymbol{A}^{-1}\delta\boldsymbol{b}\|_2 \leq \|\boldsymbol{A}^{-1}\|_2\,\|\delta\boldsymbol{b}\|_2.$$

This is an upper bound (a "worst-case") for the absolute error.

```
t = np.array([0, np.max(E)])
upp_bound = np.linalg.norm(np.linalg.inv(A))*t
plt.plot(t, upp_bound, color='red', linewidth=2)
```

$\|\boldsymbol{A}^{-1}\|_2 = 0.1054.$



Relative error – condition number (page 406–407)

Now we want to measure the *relative* error $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2 = \|\delta \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$.

From the original linear system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, we have

$$\|\boldsymbol{b}\|_2 = \|\boldsymbol{A}\,\boldsymbol{x}\|_2 \leq \|\boldsymbol{A}\|_2 \, \|\boldsymbol{x}\|_2 \qquad \Leftrightarrow \qquad \frac{1}{\|\boldsymbol{x}\|_2} \leq \frac{\|\boldsymbol{A}\|_2}{\|\boldsymbol{b}\|_2} \, .$$

Then combining with the former result on absolute error, we get

$$\frac{\|\delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \frac{1}{\|\boldsymbol{x}\|_2} \, \|\delta \boldsymbol{x}\|_2 \leq \frac{\|\boldsymbol{A}\|_2}{\|\boldsymbol{b}\|_2} \, \|\boldsymbol{A}^{-1}\|_2 \, \|\delta \boldsymbol{b}\|_2 = \kappa(\boldsymbol{A}) \, \frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}.$$

We define the **condition number**:

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \, \|\boldsymbol{A}^{-1}\|_2 = \texttt{np.linalg.cond(A)} \, .$$

Larger the condition number is, larger the relative error can be.

Example with a small perturbation on the right-hand side

$$\boldsymbol{b} = \begin{bmatrix} 0.582 \\ 0.294 \end{bmatrix}, \qquad \tilde{\boldsymbol{b}} = \begin{bmatrix} 0.583 \\ 0.293 \end{bmatrix}, \qquad \frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} = \frac{\|\tilde{\boldsymbol{b}} - \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} = 0.00218$$

$$\boldsymbol{A} = \begin{bmatrix} 0.333 & 0.250 \\ 0.200 & 0.100 \end{bmatrix}, \qquad \kappa(\boldsymbol{A}) = 13.3$$

$$\boldsymbol{x} = \begin{bmatrix} 0.916 \\ 1.108 \end{bmatrix}, \qquad \tilde{\boldsymbol{x}} = \begin{bmatrix} 0.895 \\ 1.140 \end{bmatrix}, \qquad \frac{\|\delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = 0.0266.$$

The upper bound of the relative error:

$$\kappa(\boldsymbol{A}) \frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} = 13.3 \cdot 0.00218 = 0.0289.$$

The upper bound ("worst-case") of the relative error usually is only a little bit larger.

Summary

- **Simple LU factorization.**
  $\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U}$ where $\boldsymbol{L}$ is a unit lower triangular (with 1 on the diagonal) and $\boldsymbol{U}$ is an upper triangular. It is direct from Gaussian elimination.
- **LU factorization with pivoting.**
  $\boldsymbol{P}\,\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{U}$ where $\boldsymbol{P}$ is a permutation matrix, which is from pivoting in Gaussian elimination.
- **Use LU factorization.**
  ```
  P, L, U = scipy.linalg.lu(A)
  x = np.linalg.solve(U, np.linalg.solve(L, P@b))
  ```
- **Cholesky factorization.**
  For a symmetric positive definite matrix, we have $\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{L}^T$ and pivoting is not necessary.
- **Properties of vector and matrix norm.**
  $\|\boldsymbol{A}\,\boldsymbol{x}\|_2 \le \|\boldsymbol{A}\|_2\,\|\boldsymbol{x}\|_2\ , \quad \|\boldsymbol{A}\,\boldsymbol{B}\|_2 \le \|\boldsymbol{A}\|_2\,\|\boldsymbol{B}\|_2$
- **Upper bound of the relative error for the solution.**

$$\frac{\|\delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \le \kappa(\boldsymbol{A})\,\frac{\|\delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}, \qquad \kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|_2\,\|\boldsymbol{A}^{-1}\|_2 = \texttt{np.linalg.cond(A)}$$

How about the residual? (not in the book)

We have an *approximated* solution $\tilde{\boldsymbol{x}}$ and calculate the *residual:*

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\,\tilde{\boldsymbol{x}}.$$

But can we use it to check if $\tilde{\boldsymbol{x}}$ is a good approximation of the true solution?

Theorem. The relative error of $\tilde{\boldsymbol{x}}$ is upper bounded by

$$\frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leq \kappa(\boldsymbol{A})\,\frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{b}\|_2}.$$

If the condition number $\kappa(\boldsymbol{A})$ is large, we can still have a large error on $\tilde{\boldsymbol{x}}$ even with small residual $\boldsymbol{r}$!

Reminders

Final evaluation is open until Nov. 29.

Next week will be 4-hour exercises in databars.

Homework assignment for Block 4 should be handed in before Dec. 5 10pm.

Dec. 6: mock exam from 8am to 11am, then the evaluation will start at 11:15am at B208 A054.

Office hour: Dec. 13 from 10am to 12pm.

Exam is on Dec. 17.

Reminders

Final evaluation is open until Nov. 29.

Next week will be 4-hour exercises in databars.

Homework assignment for Block 4 should be handed in before Dec. 5 10pm.

Dec. 6: mock exam from 8am to 11am, then the evaluation will start at 11:15am at B208 A054.

Office hour: Dec. 13 from 10am to 12pm.

Exam is on Dec. 17.

Thank you for attending the course!