Useful properties of matrix & vector norms

We previously saw that

$$\|Ax\| \leq \|A\| \cdot \|x\|$$  \hspace{1cm} (1)

for any matrix $A$, and any vector $x$ (of dimensions $m \times m$ and $m \times 1$, respectively).

Note that, when writing an expression such as (1), the matrix norm $\|A\|$ is understood to be the inferred norm from the vector norm used in $\|Ax\|$ and $\|x\|$. Thus

$$\|Ax\|_1 \leq \|A\|_1 \cdot \|x\|_1$$

and

$$\|Ax\|_{\infty} \leq \|A\|_{\infty} \cdot \|x\|_{\infty}$$

are both valid, but we cannot mix and match, e.g.:

$$\|Ax\|_{\infty} \leq \|A\|_2 \cdot \|x\|_1 \quad \Rightarrow \quad \text{NOT CORRECT}$$
When solving a linear system $A\bar{x}=b$, computer algorithms are only providing an approximation ($\bar{x}_{\text{app}}$) to the exact solution ($\bar{x}_{\text{ex}}$). This is due to factors such as finite precision, roundoff errors or even imperfect solution algorithms. In either case, we have an error (error vector, in fact) defined as

$$e = \bar{x}_{\text{app}} - \bar{x}_{\text{ex}}$$

Naturally, we would like to have an understanding of the magnitude of this error (e.g. some appropriate norm $\|e\|_1$). The problem is that we do not know the exact, pristine solution $\bar{x}_{\text{ex}}$!

One remedy is offered via the residual vector defined as:

$$r = b - A\bar{x}_{\text{app}}$$

The vector $r$ is something we can compute practically since it involves only known quantities ($b, A, \bar{x}_{\text{app}}$).
Furthermore, we have:

\[ \Gamma = b - A\hat{x}_{app} \]
\[ = A\hat{x}_{exact} - A\hat{x}_{app} \]
\[ = A(\hat{x}_{app} - \hat{x}_{exact}) \]
\[ = -Ae \quad \Rightarrow \quad \Gamma = -Ae \]
\[ e = -A^{-1}\Gamma \]

The last equation links the error with the residual.

Furthermore, we can write

\[ \| e \| = \| A^{-1}\Gamma \| \leq \| A^{-1} \| \| r \| \]

This equation provides a bound for the error, as a function of \( \| A^{-1} \| \) and the norm of the computable vector \( \Gamma \). Note that:

- We can obtain this estimate without knowing the exact solution, but
- We need \( \| A^{-1} \| \) and generally, computing \( A^{-1} \) is just as difficult (if not more) than finding \( \hat{x}_{exact} \). However, there are special cases where an estimate of \( \| A^{-1} \| \) can be obtained.
A different source of error:

Sometimes, the right-hand-side \( b \) of \( Ax = b \) has errors that make it deviate from its intended value. For example, in the Vandermonde matrix method for polynomial interpolation, \( b \) contains the samples \( (y_1 = f(x_1), y_2, \ldots, y_n) \) where \( y_i = f(x_i) \). An error in a measuring device supposed to sample \( f(x) \) could lead to erroneous readings \( y_i^* \) instead of \( y_i \). In general, measuring inaccuracies can lead to the right-hand-side vector \( b \) being mis-represented as \( b^* (\neq b) \).

In this case, instead of the intended solution \( x = A^{-1}b \) we in fact compute \( x^* = A^{-1}b^* \).

How important is the error \( e = x^* - x \) that is caused by this misrepresentation of \( b \)?
Let us introduce some notation:

Let \( \delta b := b^* - b \)
\( \delta x := x^* - x \)

\[
A x = b
\]
\[
A x^* = b^*
\]

\[
A (x^* - x) = b^* - b
\]

\[
A \delta x = \delta b
\]

\[
\delta x = A^{-1} \delta b
\]

Taking norms:

\[
\| \delta x \| = \| A^{-1} \delta b \| \leq \| A^{-1} \| \| \delta b \| \quad (a)
\]

Thus the error in the computed solution \( \delta x \) is proportional to the error in \( b \).

An even more relevant question is: How does the relative error \( \frac{\| \delta x \|}{\| x \|} = \frac{\| x^* - x \|}{\| x \|} \) compare to the relative error in \( b \) \( \frac{\| \delta b \|}{\| b \|} \)? This may be more useful to know, since \( \| \delta b \| \) may be impossible to compute (if we don’t know the real \( b \)!).
For this, we write

\[ A\mathbf{x} = b \Rightarrow \|b\| = \|Ax\| \leq \|A\| \cdot \|x\| \]

\[ \Rightarrow \frac{1}{\|x\|} \leq \|A\| \cdot \frac{1}{\|b\|} \quad (2) \]

Multiplying (1) & (2) we get:

\[ \frac{\|\delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta\|}{\|b\|} \]

Thus the relative error in \( x \) is bounded by a multiple of the relative error in \( b \)! The multiplicative constant \( \kappa(A) = \|A\| \cdot \|A^{-1}\| \) is called the condition number of \( A \), and is an important measure of the sensitivity of a linear system \( Ax = b \) to being solved on a computer, in the presence of inaccurate values. 

E.g. If the relative error \( \frac{\|\delta b\|}{\|b\|} \) is 0.0001 %, but \( \kappa(A) = 100.000 \) (could happen!), then we could have up to a 10% error in the computed \( x \)!
Why is this always relevant?

Simply, almost any \( b \) will have some, small relative error, due to the fact it is represented on a computer up to machine precision! The relative error will be at least as much as the machine epsilon due to roundoff:

\[
\frac{\| \delta b \|_\infty}{\| b \|} \geq \varepsilon \approx 10^{-7} \quad \text{(for floats)}.
\]

But, how bad can the condition number get?

Very bad, at times. For example:

Hilbert matrices \( H_n \in \mathbb{R}^{n \times n} \)

\[
(H_n)_{ij} = \frac{1}{i + j - 1}
\]

\[
H_5 = \begin{bmatrix}
1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\
\frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\
\frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\
\frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9}
\end{bmatrix}
\]

\[K_\infty(H_5) = \| H_5 \|_\infty \cdot \| H_5^{-1} \|_\infty \approx 10^6 !
\]

Thus, any attempt at solving \( H_5 x = b \) would be subject to a relative error up to 10\% just due to roundoff errors in \( b \)!
Another case: near-singular matrices

\[ A = \begin{bmatrix} 1 & 2 \\ 3 & 6+\varepsilon \end{bmatrix} \]

As \( \varepsilon \to 0 \), \( A \) becomes singular (non-invertible).

In this case, \( \kappa(A) \to \infty \).

What is the best case for \( \kappa(A) \)?

**Lemma** For any vector-induced matrix norm, we have \( \|I\| = 1 \).

**Proof** From definition:

\[ \|I\| = \max_{x \neq 0} \frac{\|Ix\|}{\|x\|} = \max_{x \neq 0} \frac{\|x\|}{\|x\|} = 1 \]

Using property (iv) of matrix norms, we get:

\[ I = A \cdot A^{-1} \Rightarrow \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\| \]

Thus \( \kappa(A) \geq 1 \).

The "best" conditioned matrices are of the form \( A = c \cdot I \) and have \( \kappa(A) = 1 \).
General case: Backslash operator \ 

If \( x, y \in \mathbb{R}^n \), \( A \in \mathbb{R}^{n \times n} \) and \( Ax = y \) then \( x = A^{-1}y \) is implemented in MATLAB as \( x = A \backslash y \).

The actual algorithm used is selected "automatically" given the nature of \( A \). Worst case, though, will require a run-time proportional to \( n^3 \) ! (We write: \( \text{Cost} = \mathcal{O}(n^3) \)).

Alternative LU factorization.

Any non-singular matrix can be written as

\[
A = LU \quad \text{where} \quad L = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{n1} & \cdots & \cdots & \cdots & l_{nn} \end{bmatrix} \quad \text{lower triangular}
\]

\[
U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \quad \text{upper triangular}
\]
\[ [L, U] = lu(A); \quad \Rightarrow \text{cost } O(n^3) \]

We then reduce \( Ax = b \) to \( 2 \) triangular systems

\[
\begin{align*}
Ax &= b \\
LUx &= b \\
Lz &= b \quad (1) \\
Uz &= 2 \quad (2)
\end{align*}
\]

followed by \( Ux = z \) (2)

In matlab:

\[
\begin{align*}
z &= L \backslash b \quad \text{Cost } O(n^2) \! \\
x &= U \backslash z \quad \text{Cost } O(n^2) \!
\end{align*}
\]

Beneficial, if we need to solve several systems

\[ Ax_1 = b_1, \quad Ax_2 = b_2, \quad \ldots, \quad Ax_k = b_k \]

In which case the cost of factorization is paid only once!