How do errors propagate?

Two main procedures:
1. **Forward error analysis**
   - Superficially the easiest; often very ineffective
2. **Backward error analysis**
   - Extremely effective for many tasks in linear algebra, such as Gaussian elimination

**Forward error analysis**

**Example:** What accuracy do we get for \( y = x^2 \sin x \) if \( x = 1.211713 \) (correctly rounded) and functions are evaluated with errors \( \leq 0.7 \cdot 10^{-6} \).

**Idea:** Consider each error source independently:
- Use partial derivatives to see how each error source alone affects the end result.
- Superpose.

In current example:

\[
|\Delta y| \leq |2x| |\Delta x| + \left| e^{\sin x} \cdot 0.05x \right| |\Delta x| + \left| x \right| \frac{\Delta (\sin x)}{\sin x} + |\Delta x^2| \leq \frac{5 \cdot 10^{-6}}{\sin x} + \frac{5 \cdot 10^{-6}}{x^2} + \frac{5 \cdot 10^{-6}}{\sin x} + \frac{5 \cdot 10^{-6}}{x^2} \]

We have here used \(|\Delta x|, |\Delta x^2| = 0.5 \cdot 10^{-6}\), \(|\Delta (\sin x)|, |\Delta (\sin x)| = 0.7 \cdot 10^{-6}\).

For a lengthy algorithm (such as G.Eh.), this becomes a complete mess ----
Backward error analysis

Idea: Each time an error is committed, see how much the original problem needs to be changed in order to leave the calculated result as an exact result.

Very schematic example:
Given \(x_1, x_2, \ldots, x_n\), compute product \(x_1 \times x_2 \times \ldots \times x_n\).

Forward:

\[
\ldots ((x_1 \times x_2) \times x_3) \times x_4) \times \ldots \times x_n \rightarrow \text{Answer } \tilde{x}_1 \times \tilde{x}_2 \times \ldots \times \tilde{x}_n
\]

\[\rightarrow \text{Error} \rightarrow \text{Error} \rightarrow \text{Error} \rightarrow \text{etc.}\]

Backward:

\[
\left\{ \begin{array}{l}
\text{How much to change } x_2 \text{ so that } \text{product } x_1 \times x_2 \text{ is exact.}
\end{array} \right.
\]

\[
\text{How much to change } x_3 \text{ so that product } \ldots \times x_3 \text{ is exact.}
\]

Then:
How much does the product \(x_1 \times \ldots \times x_n\) change if each input number is changed as above, but all arithmetic is exact.

Power of backwards analysis will not become apparent until we apply it to a major problem, such as G.EL:

1. 
2. Use condition number for linear system to assess how much its solution has changed.
Major advantages

1. Turns out, all \( L \) are easy to do.
2. Often, effect of rounding errors escalate so that the \( LU \) decomposition we get is far from the exact one, but nevertheless, the system solution remains essentially intact. The analysis captures this.
3. The analysis separates:
   i. Sensitivity of system itself with regard to perturbations in its entries (described by 'condition number')
   ii. Algorithm-dependent features.

Any forward error analysis would mix together these two separate issues.

Will turn out:

Independently of the \( A \)-matrix, all the accumulated errors will not be much worse than the errors committed when the matrix entries initially were initially rounded to fit into the floating point format.
BACKWARD ERROR ANALYSIS
FOR LU DECOMPOSITION

Given \( A_0 \)

\[
\begin{bmatrix}
1 \\
3 \\
3
\end{bmatrix}
\]

Total matrix
RHS included
Assume pivoting has been done
in advance.

All operations that we do are multiplications from
the left with matrices of the form

\[
N_c = \begin{bmatrix}
1 & \cdots & 0 \\
0 & \cdots & 1 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
N_{n-1} & \cdots & N_2 & N_1 \\
N_{n-1} & \cdots & N_2 & N_1
\end{bmatrix}
\]

If no errors, systems \( A_0 \) and \( A_{n-1} \) have the same
solution. We do however all sorts of rounding
errors along the way. Thus, we do not get \( A_{n-1} \) but instead
some \( \widetilde{A}_{n-1} \).

We want to show that \( \widetilde{A}_{n-1} \) has the same solution
as the system \( A_0 + \varepsilon B \), matrix of same norm as the \( A_i \)-matrices

If exact arithmetic all the way:

\[
A_r = N_r A_{r-1}, \quad r=1,2,\ldots,n-1.
\]

In reality:

\[
\widetilde{A}_r = N_r \widetilde{A}_{r-1} + H_r
\]

This is an exact formula.

\[ (*) \]

New roundoff errors,
of size \( \varepsilon \cdot B \)
machine rounding.

\( H_r \) size of \( A_{r-1} \),
	on desired form but
introduces new errors.
At this step \( r \)

\[
\begin{bmatrix}
1 & \cdots & c_r \\
\vdots & \ddots & \vdots \\
1 & \cdots & 1
\end{bmatrix}
\]

row \( r \)

col \( r \)

So: Rows 1 to \( r \) of the system matrix are left unchanged

\[ \Rightarrow H_r \text{ is zero in rows 1 to } r. \]

Write equation (\( x \))

\[
\tilde{A}_r = \tilde{N}_r \sum_{\ell=1}^{\infty} \tilde{A}_{r-\ell} + \tilde{N}_r^{-1} H_r \]

\[
\tilde{A}_{r-1} = \tilde{N}_{r-1} \sum_{\ell=1}^{\infty} \tilde{A}_{r-\ell-1} + \tilde{N}_{r-1}^{-1} H_{r-1}
\]

etc.

Combine all these relations together:

\[
\tilde{A}_{n-1} = \tilde{N}_{n-1} \tilde{N}_{n-2} \cdots \tilde{N}_1 \times
\begin{bmatrix}
A_0 + (\tilde{N}_1)^{-1} H_1 + (\tilde{N}_1)^{-1} (\tilde{N}_2)^{-1} H_2 + \cdots + (\tilde{N}_1)^{-1} \cdots (\tilde{N}_{n-1})^{-1} H_{n-1}
\end{bmatrix}
\]

We can now conclude:

1. The system \( \tilde{A}_{n-1} \) has exactly the same solution as \( \tilde{A}_n \).
2. It is totally irrelevant for the system solution if the product \( \tilde{N}_{n-1} \tilde{N}_{n-2} \cdots \tilde{N}_1 \) grows large.

To finish up, we need to estimate the size of the perturbation

\[
(\tilde{N}_1)^{-1} H_1 + (\tilde{N}_1)^{-1} (\tilde{N}_2)^{-1} H_2 + \cdots + (\tilde{N}_1)^{-1} \cdots (\tilde{N}_{n-1})^{-1} H_{n-1}
\]

Note:

1. \( \tilde{N}_r = \begin{bmatrix} 1 & \cdots & c_r \\ \vdots & \ddots & \vdots \\ \tilde{N}_{r-1} & \cdots & 1 \end{bmatrix} \Rightarrow \tilde{N}_r^{-1} = \begin{bmatrix} 1 & \cdots & -c_r \\ \vdots & \ddots & \vdots \\ \tilde{N}_{r-1}^{-1} & \cdots & 1 \end{bmatrix} \)

   - only sign change

2. \( H_r \) is zero in rows 1 to \( r \). Therefore

   \[
   (\tilde{N}_1)^{-1} (\tilde{N}_2)^{-1} \cdots (\tilde{N}_r)^{-1} H_r = H_r.
   \]
Therefore:
\[ A_{n-1} \] has exactly the same solution as
\[ A_0 + \sum \{ h_1 + h_2 + \ldots + h_{n-1} \} \]
All the \( h_i \) disappeared from the analysis.

Only remaining issue: can the \( h_i \) become large.
Recall: of size \( \frac{\varepsilon}{v} \).
Uncertainty \( \frac{\varepsilon}{v} \) all stay small because of pivoting.

CONCLUDING NOTES

1. Extremely simple form of \( (\star) \) - Exact result

2. \( A_{n-1} \) and \( A_{n-1} \) need not be close to each other - we cannot give any good bound for

3. With errors \( h_1, h_2, \ldots, h_{n-1} \) independent, would expect the sum to grow with \( n \) no worse than \( \varepsilon \sqrt{n} \).

4. Can show similarly; Back substitution is equally "harmless!"

REMAINS:
How much can the solution \( \mathbf{x} \) to \( \mathbf{A}\mathbf{x} = \mathbf{b} \) change due to changes in \( \mathbf{A} \) (and in \( \mathbf{b} \)).

Question entirely independent of Gaussian Elimination.
Addressed by "CONDITION NUMBERS."