JDEP 384H: Numerical Methods in Business

Instructor: Thomas Shores
Department of Mathematics

Lecture 17, February 20, 2007
110 Kaufmann Center

Outline

1. BT 3.1: Basics of Numerical Analysis
    - Finite Precision Representation
    - Error Analysis

Outline

1. BT 3.1: Basics of Numerical Analysis
   - Finite Precision Representation
   - Error Analysis

Outline

Error Terminology

Error Definitions:

Suppose the exact quantity that we want to estimate is x_T and we end up calculating the quantity x_A.

- **Absolute error** of our calculation is

$$\text{Error}(x_A) = |x_A - x_T|$$

- **Relative error** is

$$\text{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|},$$

provided $x_T \neq 0$.

Error Terminology

Error Definitions:

Suppose the exact quantity that we want to estimate is x_T and we end up calculating the quantity x_A.

- **Absolute error** of our calculation is

$$\text{Error}(x_A) = |x_A - x_T|$$

- **Relative error** is

$$\text{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|},$$

provided $x_T \neq 0$.

Error Terminology

Error Definitions:

Suppose the exact quantity that we want to estimate is x_T and we end up calculating the quantity x_A.

- **Absolute error** of our calculation is

$$\mathrm{Error}(x_A) = |x_A - x_T|$$

- **Relative error** is

$$\mathrm{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|},$$

provided $x_T \neq 0$.

Error Terminology

An Application:

Significant digits.

- We say x_A has m significant digits with respect to x_T if m is the largest nonnegative integer for which

$$\text{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|} \leq 5 \times 10^{-m}.$$

- Use the definition to answer this question: $x_A = 25.489$ has how many significant digits with respect to $x_T = 25.482$.
- Subtraction of nearly equal quantities can cause **catastrophic loss of significance digits**. Addition of them does not cause such a loss.

An Application:

Significant digits.

- We say x_A has m significant digits with respect to x_T if m is the largest nonnegative integer for which

$$\text{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|} \leq 5 \times 10^{-m}.$$

- Use the definition to answer this question: $x_A = 25.489$ has how many significant digits with respect to $x_T = 25.482$.
- Subtraction of nearly equal quantities can cause **catastrophic loss of significance digits**. Addition of them does not cause such a loss.

An Application:

Significant digits.

- We say x_A has m significant digits with respect to x_T if m is the largest nonnegative integer for which

$$\mathrm{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|} \leq 5 \times 10^{-m}.$$

- Use the definition to answer this question: $x_A = 25.489$ has how many significant digits with respect to $x_T = 25.482$.
- Subtraction of nearly equal quantities can cause **catastrophic loss of significance digits**. Addition of them does not cause such a loss.

An Application:

Significant digits.

- We say x_A has m significant digits with respect to x_T if m is the largest nonnegative integer for which

$$\text{Rel}(x_A) = \frac{|x_A - x_T|}{|x_T|} \leq 5 \times 10^{-m}.$$

- Use the definition to answer this question: $x_A = 25.489$ has how many significant digits with respect to $x_T = 25.482$.
- Subtraction of nearly equal quantities can cause **catastrophic loss of significance digits**. Addition of them does not cause such a loss.

Vector Error?

Vector Error Definitions:

Suppose the exact quantity that we want to estimate is $\mathbf{x}_T$ and we end up calculating the quantity $\mathbf{x}_A$. Choose a vector norm $\|\cdot\|$

- **Absolute error** of our calculation is

$$\text{Error}(\mathbf{x}_A) = \|\mathbf{x}_A - \mathbf{x}_T\|$$

- **Relative error** is

$$\text{Rel}(\mathbf{x}_A) = \frac{\|\mathbf{x}_A - \mathbf{x}_T\|}{\|\mathbf{x}_T\|},$$

provided $\|\mathbf{x}_T\| \neq 0$.

Vector Error?

Vector Error Definitions:

Suppose the exact quantity that we want to estimate is $\mathbf{x}_T$ and we end up calculating the quantity $\mathbf{x}_A$. Choose a vector norm $\|\cdot\|$

- **Absolute error** of our calculation is

$$\text{Error}(\mathbf{x}_A) = \|\mathbf{x}_A - \mathbf{x}_T\|$$

- **Relative error** is

$$\text{Rel}(\mathbf{x}_A) = \frac{\|\mathbf{x}_A - \mathbf{x}_T\|}{\|\mathbf{x}_T\|},$$

provided $\|\mathbf{x}_T\| \neq 0$.

Vector Error?

Vector Error Definitions:

Suppose the exact quantity that we want to estimate is $\mathbf{x}_T$ and we end up calculating the quantity $\mathbf{x}_A$. Choose a vector norm $\|\cdot\|$

- **Absolute error** of our calculation is

$$\text{Error}(\mathbf{x}_A) = \|\mathbf{x}_A - \mathbf{x}_T\|$$

- **Relative error** is

$$\text{Rel}(\mathbf{x}_A) = \frac{\|\mathbf{x}_A - \mathbf{x}_T\|}{\|\mathbf{x}_T\|},$$

  provided $\|\mathbf{x}_T\| \neq 0$.

Big Oh Notation

Definition:

Function $f(x)$ is **big oh** of $g(x)$ as $x \to a$ if there exists a positive number M such that for x sufficiently near to a, $|f(x)| \leq M|g(x)|$.

- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h), \; h \to 0,$$
- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2), \; h \to 0.$$
- If Gaussian elimination is used to solve $A\mathbf{x} = \mathbf{b}$, A an $n \times n$ matrix, then the number of flops needed is a measure of the **complexity** of this **polynomial time** algorithm:
$$\frac{2}{3}n^3 + an^2 + bn + d = \frac{2}{3}n^3 + \text{l.o.t.} = \mathcal{O}(n^3), \; n \to \infty.$$

Big Oh Notation

Definition:

Function $f(x)$ is **big oh** of $g(x)$ as $x \to a$ if there exists a positive number M such that for x sufficiently near to a, $|f(x)| \leq M|g(x)|$.

- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h), \ h \to 0,$$
- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2), \ h \to 0.$$
- If Gaussian elimination is used to solve $A\mathbf{x} = \mathbf{b}$, A an $n \times n$ matrix, then the number of flops needed is a measure of the **complexity** of this **polynomial time** algorithm:
$$\frac{2}{3}n^3 + an^2 + bn + d = \frac{2}{3}n^3 + \text{l.o.t.} = \mathcal{O}(n^3), \ n \to \infty.$$

Big Oh Notation

Definition:

Function $f(x)$ is **big oh** of $g(x)$ as $x \to a$ if there exists a positive number M such that for x sufficiently near to a, $|f(x)| \leq M|g(x)|$.

- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h), \; h \to 0,$$
- For approximating derivatives:
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2), \; h \to 0.$$
- If Gaussian elimination is used to solve $A\mathbf{x} = \mathbf{b}$, A an $n \times n$ matrix, then the number of flops needed is a measure of the **complexity** of this **polynomial time** algorithm:
$$\frac{2}{3}n^3 + an^2 + bn + d = \frac{2}{3}n^3 + \mathrm{l.o.t.} = \mathcal{O}(n^3), \; n \to \infty.$$

Big Oh Notation

Definition:

Function $f(x)$ is **big oh** of $g(x)$ as $x \to a$ if there exists a positive number M such that for x sufficiently near to a, $|f(x)| \leq M\,|g(x)|$.

- For approximating derivatives:
  $$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h), \ \ h \to 0,$$
- For approximating derivatives:
  $$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}\left(h^2\right), \ \ h \to 0.$$
- If Gaussian elimination is used to solve $A\mathbf{x} = \mathbf{b}$, A an $n \times n$ matrix, then the number of flops needed is a measure of the **complexity** of this **polynomial time** algorithm:
  $$\frac{2}{3}n^3 + an^2 + bn + d = \frac{2}{3}n^3 + \mathrm{l.o.t.} = \mathcal{O}\left(n^3\right), \ n \to \infty.$$

Sources of Error

A Catalogue:

- **Inaccurate model:** Implied volatility observations suggest that Black-Scholes might not be entirely accurate. Hence, no matter how refined we make our calculations, we can expect some error when we compare to reality.
- **Inaccurate data:** solve a Black-Scholes equation with $r = 0.065$ instead of the correct $r = 0.06$. Nothing we do short of getting exact data will save us from error. In computer science, the principle is GIGO.
- **Blunders:** both hardware and software. Hardware problems are relatively rare nowadays, but software errors flourish.
- **Machine error:** rounding error and the error of finite precision floating point computation as in our first few examples.

Sources of Error

A Catalogue:

- **Inaccurate model:** Implied volatility observations suggest that Black-Scholes might not be entirely accurate. Hence, no matter how refined we make our calculations, we can expect some error when we compare to reality.
- **Inaccurate data:** solve a Black-Scholes equation with $r = 0.065$ instead of the correct $r = 0.06$. Nothing we do short of getting exact data will save us from error. In computer science, the principle is GIGO.
- **Blunders:** both hardware and software. Hardware problems are relatively rare nowadays, but software errors flourish.
- **Machine error:** rounding error and the error of finite precision floating point computation as in our first few examples.

Sources of Error

A Catalogue:

- **Inaccurate model:** Implied volatility observations suggest that Black-Scholes might not be entirely accurate. Hence, no matter how refined we make our calculations, we can expect some error when we compare to reality.
- **Inaccurate data:** solve a Black-Scholes equation with $r = 0.065$ instead of the correct $r = 0.06$. Nothing we do short of getting exact data will save us from error. In computer science, the principle is GIGO.
- **Blunders:** both hardware and software. Hardware problems are relatively rare nowadays, but software errors flourish.
- **Machine error:** rounding error and the error of finite precision floating point computation as in our first few examples.

Sources of Error

A Catalogue:

- **Inaccurate model:** Implied volatility observations suggest that Black-Scholes might not be entirely accurate. Hence, no matter how refined we make our calculations, we can expect some error when we compare to reality.
- **Inaccurate data:** solve a Black-Scholes equation with $r = 0.065$ instead of the correct $r = 0.06$. Nothing we do short of getting exact data will save us from error. In computer science, the principle is GIGO.
- **Blunders:** both hardware and software. Hardware problems are relatively rare nowadays, but software errors flourish.
- **Machine error:** rounding error and the error of finite precision floating point computation as in our first few examples.

Sources of Error

A Catalogue:

- **Inaccurate model:** Implied volatility observations suggest that Black-Scholes might not be entirely accurate. Hence, no matter how refined we make our calculations, we can expect some error when we compare to reality.
- **Inaccurate data:** solve a Black-Scholes equation with $r = 0.065$ instead of the correct $r = 0.06$. Nothing we do short of getting exact data will save us from error. In computer science, the principle is GIGO.
- **Blunders:** both hardware and software. Hardware problems are relatively rare nowadays, but software errors flourish.
- **Machine error:** rounding error and the error of finite precision floating point computation as in our first few examples.

Catalogue (continued):

- **Mathematical truncation:** consider the formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \text{ for } h > 0.$$

  No matter how small we make h, we will not get the exact answer because mathematically the formula is not an exact equality. This is a bit like "mathematical roundoff."

- **Algorithmic instability:** we'll see an example of this in Example 7, where we compute the sequence $1/3^n$ by a stable algorithm and an unstable algorithm. The problem is not in the sequence itself, but how we try to compute it. This is also an example of **error propagation**.

- **Mathematical instability:** this is more subtle. In Example 5 we see the problem is not with algorithms for solving linear systems. It's deeper than that, because the Hilbert matrix itself is extremely sensitive to change

Sources of Error

Catalogue (continued):

- **Mathematical truncation:** consider the formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \text{ for } h > 0.$$

  No matter how small we make h, we will not get the exact answer because mathematically the formula is not an exact equality. This is a bit like "mathematical roundoff."

- **Algorithmic instability:** we'll see an example of this in Example 7, where we compute the sequence $1/3^n$ by a stable algorithm and an unstable algorithm. The problem is not in the sequence itself, but how we try to compute it. This is also an example of **error propagation**.

- **Mathematical instability:** this is more subtle. In Example 5 we see the problem is not with algorithms for solving linear systems. It's deeper than that, because the Hilbert matrix itself is extremely sensitive to change

Sources of Error

Catalogue (continued):

- **Mathematical truncation:** consider the formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \text{ for } h > 0.$$

  No matter how small we make h, we will not get the exact answer because mathematically the formula is not an exact equality. This is a bit like "mathematical roundoff."

- **Algorithmic instability:** we'll see an example of this in Example 7, where we compute the sequence $1/3^n$ by a stable algorithm and an unstable algorithm. The problem is not in the sequence itself, but how we try to compute it. This is also an example of **error propagation**.

- **Mathematical instability:** this is more subtle. In Example 5 we see the problem is not with algorithms for solving linear systems. It's deeper than that, because the Hilbert matrix itself is extremely sensitive to change

Sources of Error

Catalogue (continued):

- **Mathematical truncation:** consider the formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \text{ for } h > 0.$$

  No matter how small we make h, we will not get the exact answer because mathematically the formula is not an exact equality. This is a bit like "mathematical roundoff."
- **Algorithmic instability:** we'll see an example of this in Example 7, where we compute the sequence $1/3^n$ by a stable algorithm and an unstable algorithm. The problem is not in the sequence itself, but how we try to compute it. This is also an example of **error propagation**.
- **Mathematical instability:** this is more subtle. In Example 5 we see the problem is not with algorithms for solving linear systems. It's deeper than that, because the Hilbert matrix itself is extremely sensitive to change

Convergence Concepts

Definitions

We say that the sequence of numbers $\{x_n\}_{n=1}^{\infty}$ **converges** to x^* if $\lim_{n\to\infty} |x_n - x^*| = 0$. We say the sequence of vectors $\{\mathbf{x}_n\}_{n=1}^{\infty}$ **converges** to the vector $\mathbf{x}^*$ if

$$\lim_{n\to\infty} \|\mathbf{x}_n - \mathbf{x}^*\| = 0$$

where $\|\cdot\|$ is some vector norm. If a sequence of iterates $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}, \ldots$ produced by some algorithm converges to the desired point $\mathbf{x}^*$, we say that the sequence **converges with order** q (an integer greater than or equal to 1 called the **order of convergence**) if

$$\left\|\mathbf{x}^{(n+1)} - \mathbf{x}^*\right\| = \mathcal{O}\left(\left\|\mathbf{x}^{(n)} - \mathbf{x}^*\right\|^q\right),\ n \to \infty.$$

Examples:

- $\left\{ \frac{1}{2^n} \right\}_{n=0}^{\infty}$ converges linearly to zero.
- $\left\{ \frac{1}{2^{2^n}} \right\}_{n=0}^{\infty}$ converges quadratically to zero.

Examples:

- $\left\{\frac{1}{2^n}\right\}_{n=0}^{\infty}$ converges linearly to zero.
- $\left\{\frac{1}{2^{2^n}}\right\}_{n=0}^{\infty}$ converges quadratically to zero.

Examples:

- $\left\{\frac{1}{2^n}\right\}_{n=0}^{\infty}$ converges linearly to zero.
- $\left\{\frac{1}{2^{2^n}}\right\}_{n=0}^{\infty}$ converges quadratically to zero.

Examples

Example

(Variant on Example 5 of NumericalAnalysisNotes) We find the least integer n such that at least one entry of a certain system $H_n\mathbf{x} = \mathbf{b}$ has zero significant digits relative to the answer. Here H_n is the n-th Hilbert matrix and $\mathbf{x}$ is a vector whose i-th coordinate is i.

```
> n = 4
> H = hilb(n)
> xtrue = (1:n)'
> b = H*xtrue
> xapprox = inv(H)*b
> % now repeat for larger n
> % also try H\b...any improvement?
```

Example

(Example 7 of NumericalAnalysisNotes) Let $p_n = 1/3^n$, $n = 0, 1, 2, \ldots$. This sequence obeys the rule $p_{n+1} = p_{n-1} - \frac{8}{3} p_n$ with $p_0 = 1$ and $p_1 = 1/3$. Similarly, we see that $p_{n+1} = \frac{1}{3} p_n$ with $p_0 = 1$. Use Matlab to plot the sequence $\{p_n\}_{n=0}^{50}$ directly, and then using the above recursion algorithms with p_0 and p_1 given and overlay the plot of those results. Repeat the plot with the last 11 of the points.

```
>N=50
>p1 = (1/3).^(0:N);
>p2 = p1; p3 = p1;
>for n = 1:N,p2(n+1) = (1/3)*p2(n);end
>for n = 2:N,p3(n+1) = p3(n-1)-8/3*p3(n);end
>plot([p1',p2',p3'])
>plot([p1(N-11:N)',p2(N-11:N)',p3(N-11:N)'])
```