Computer Arithmetic

CS/SE 4X03

Ned Nedialkov

McMaster University

September 5, 2023

The Patriot disaster

During the Gulf War in 1992, a Patriot missile missed an Iraqi Skud, which killed 28 Americans. What happened?

- Patriot's internal clock counted tenths of a second and stored the result as an integer.
- To convert to a floating-point number, the time was multiplied by 0.1 stored in 24 bits.
- 0.1 in binary is $0.001\ 1001\ 1001\ ...$, which was chopped to 24 bits. Roundoff error $\approx 9.5 \times 10^{-8}$.
- After 100 hours the measured time had an error of

$$100 \times 60 \times 60 \times 10 \times 9.5 \times 10^{-8} \approx 0.34 \text{ seconds.}$$

- A Skud flies at $\approx 1,676$ meters per second. 0.34 seconds error results in

$$0.34 \times 1,676 \approx 569 \text{ meters}$$

Vancouver Stock Exchange

- In 1982, the Vancouver Stock Exchange started an electronic stock index set initially to 1,000 points.
- The index was updated after each transaction.
- In 22 months the index fell to 520.
- It was not supposed to fall in a bull market.
- Investigation showed each intermediate result was rounded to 2 decimals by chopping, e.g. 568.958 rounds to 568.95.
- When this was fixed, the index was 1098.892.

Ariane 5

- Launched on June 4, 1996.
- 36 seconds before self-destruction.
- A 64-bit floating-point number was converted to a 12-bit integer.

What is the output of this Matlab code?

```
a(1) =  (1/cos(100*pi+pi/4))^2; % (1/cos(100π + π/4))^2 = 2
a(2) = 3*tan(atan(1e7))/1e7;     % 3 tan(arctan(10^7))/10^7 = 3
x = 4;
for i=1:100 x = sqrt(x); end
for i=1:100 x = x*x; end
a(3) = x;                        % = 4
a(4) = 5*(1+exp(-100)-1)/(1+exp(-100)-1); % 5 (1+e^-100 -1)/(1+e^-100 -1) = 5
a(5) = log(exp(6e+3))/1e+3;               % ln(e^6000)/1000 = 6
for i = 1:5
    fprintf('%d: %.16f\n', i+1, a(i));
end
```

Useful links

- IEEE 754 double precision visualization
- C. Moler. Floating Point Numbers
- IEEE 754
- N. Higham. Half Precision Arithmetic: fp16 Versus bfloat16
- GNU Multiple Precision Arithmetic Library
- Quadruple-precision floating-point format

Outline

Floating-point number system

A floating-point (FP) system is characterized by four integers (β, t, L, U), where

- β is base of the system or radix
- t is number of digits or precision
- $[L, U]$ is exponent range

A common way of expressing a FP number x is

$$x = \pm\, d_0.d_1 \cdots d_{t-1} \times \beta^e$$

where

- $0 \leq d_i \leq \beta - 1$, $i = 0, \ldots, t-1$
- $e \in [L, U]$

$$x = \pm\, d_0.d_1 \cdots d_{t-1} \times \beta^e$$

- The string of base β digits $d_0 d_1 \cdots d_{t-1}$ is called mantissa or significand
- $d_1 d_2 \cdots d_{t-1}$ is called fraction
- A FP number is normalized if d_0 is nonzero
  denormalized otherwise

Floating-point number system cont.

Example 1. Consider the FP $(10, 3, -2, 2)$.

- The normalized numbers are of the form

$$\pm d_0.d_1d_2 \times 10^e, \quad d_0 \neq 0,\ e \in [-2, 2]$$

- largest positive number is 9.99×10^2
- smallest positive normalized number is 1.00×10^{-2}
- smallest positive denormalized number 0.01×10^{-2}
- denormalized numbers are e.g. 0.23×10^{-2}, 0.11×10^{-2}
- 0 is represented as 0.00×10^0

Rounding

How to store a real number

$$x = \pm\, d_0.d_1 \cdots d_{t-1} d_t d_{t+1} \cdots \times \beta^e$$

in t digits?

Denote by $\mathsf{fl}(x)$ the FP representation of x

- Rounding by chopping (also called rounding towards zero)
- Rounding to nearest. $\mathsf{fl}(x)$ is the nearest FP to x
  If a tie, round to the even FP
- Rounding towards $+\infty$. $\mathsf{fl}(x)$ is the smallest FP $\geq x$
- Rounding towards $-\infty$. $\mathsf{fl}(x)$ is the largest FP $\leq x$

Rounding cont.

Example 2. Consider the FP $(10, 3, -2, 2)$.
Let $x = 1.2789 \times 10^1$

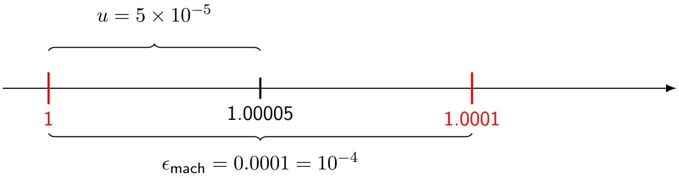
- chopping: $\mathrm{fl}(x) = 1.27 \times 10^1$
- nearest: $\mathrm{fl}(x) = 1.28 \times 10^1$
- $+\infty$: $\mathrm{fl}(x) = 1.28 \times 10^1$
- $-\infty$: $\mathrm{fl}(x) = 1.27 \times 10^1$

Let $x = 1.275000$. It is in the middle between 1.27 and 1.28.
When a tie, round to the even, the number with even last digit

- nearest: $\mathrm{fl}(x) = 1.28$

Machine epsilon

- Machine epsilon: the distance from 1 to the next larger FP number
  E.g. in $t = 5$ decimal digits, $\epsilon_{\text{mach}} = 1.0001 - 1.0000 = 10^{-4}$



  Note: 1.00005 is not representable in this FP system, just denotes the middle

- Unit roundoff: $u = \epsilon_{\text{mach}}/2$

Machine epsilon cont.

When rounding to the nearest

$$\mathrm{fl}(x) = x(1+\epsilon), \quad \text{where } |\epsilon| \le u$$

i.e.

$$\frac{\mathrm{fl}(x) - x}{x} = \epsilon$$

$$\left| \frac{\mathrm{fl}(x) - x}{x} \right| = |\epsilon| \le u$$

ϵ is the relative error in $\mathrm{fl}(x)$.

Machine epsilon cont.

Example 3. Consider the FP $(10, 3, -2, 2)$.

- The machine epsilon is $\epsilon_{\text{mach}} = 1.01 - 1.00 = 0.01$.
- Unit roundoff is $\epsilon_{\text{mach}}/2 = 0.01 = 0.005 = 5 \times 10^{-3}$.

Let $x = 1.2789 \times 10^1$. With rounding to nearest,

$$\text{fl}(x) = 1.28 \times 10^1.$$

Then

$$\left|\frac{\text{fl}(x) - x}{x}\right| = \frac{|1.28 \times 10^1 - 1.2789 \times 10^1|}{1.2789 \times 10^1} = \frac{|1.28 - 1.2789|}{1.2789}$$
$$\approx 8.6011 \times 10^{-4} < 5 \times 10^{-3}$$

Machine epsilon cont.

Example 4. Consider the FP $(10, 3, -2, 2)$. Let $x = 3.4950001 \times 10^2$. With rounding to nearest,

$$\text{fl}(x) = 3.50 \times 10^2.$$

The absolute error in $\text{fl}(x)$ is

$$\text{fl}(x) - x = 3.50 \times 10^2 - 3.4950001 \times 10^2 \approx 0.5$$

which is large.
But the relative error is within $u = 5 \times 10^{-3}$:

$$\left|\frac{\text{fl}(x) - x}{x}\right| = \frac{|3.50 \times 10^2 - 3.4950001 \times 10^2|}{3.4950001 \times 10^2} = \frac{|3.50 - 3.4950001|}{3.4950001}$$
$$\approx 1.4306 \times 10^{-3} < 5 \times 10^{-3}$$

IEEE 754

- IEEE 754 standard for FP arithmetic (1985)
- IEEE 754-2008, IEEE 754-2019
- Most common (binary) single and double precision since 2008 half precision

	bits	t	L	U	ϵ_{mach}
single	32	24	-126	127	$\approx 1.2 \times 10^{-7}$
double	64	53	-1022	1023	$\approx 2.2 \times 10^{-16}$

	range	smallest normalized	smallest denormalized
single	$\pm 3.4 \times 10^{38}$	$\pm 1.2 \times 10^{-38}$	$\pm 1.4 \times 10^{-45}$
double	$\pm 1.8 \times 10^{308}$	$\pm 2.2 \times 10^{-308}$	$\pm 4.9 \times 10^{-324}$

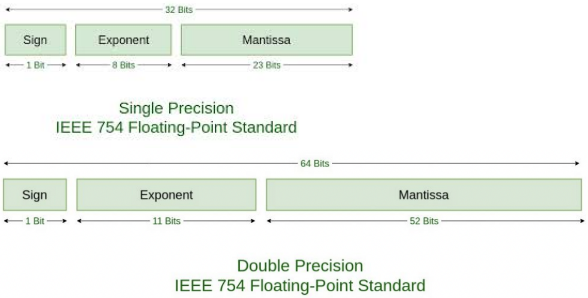
(These are $\approx$ values)

IEEE 754 cont.

Exceptional values

- `Inf`, `-Inf` when the result overflows, e.g. `1/0.0`
- `NaN` "Not a Number" results from undefined operations e.g. `0/0`, `0*Inf`, `Inf/Inf`
  `NaNs` propagate through computations

IEEE 754 cont.



- sign 0 positive, 1 negative
- exponent is biased
- first bit of mantissa is not stored, sticky bit, assumed 1

(Figures are from IEEE Standard 754 Floating Point Numbers

IEEE 754 cont.

Single precision

- FP numbers
  - biased exponent: from 1 to 254, bias: 127
  - actual exponent: $1 - 127 = -126$ to $254 - 127 = 127$
- `Inf`
  - sign: 0 for +`Inf`, 1 for -`Inf`
  - biased exponent: all 1's, 255
  - fraction: all 0's
- `NaN`
  - sign: 0 or 1
  - biased exponent: all 1's, 255
  - fraction: at least one 1
- 0
  - sign: 0 for $+0$, 1 for -0
  - biased exponent: all 0's
  - mantissa: all 0's

IEEE 754 cont.

Double precision

- bias 1023
- biased exponent: from 1 to 2046
- actual exponent: from -1022 to 1023
- rest similar to single

Try IEEE 754 double precision visualization

IEEE 754 cont.

Why biased exponent?

What if the exponent is stored as a signed number in 2's complement representation?

Example 5.

- Consider single precision, and assume the exponent is stored as a signed integer.
- Assume we have two positive numbers $x > y$ with exponents 5 and -5, respectively.

Example 5. cont.

- 5 in 8 bits is 00000101
- -5 in 2's complement is 11111011
- Then x and y are of the form

$$x = \underbrace{0}_{+}\underbrace{00000101}_{5}\underbrace{\cdots}_{\text{23 bits}}$$
$$y = \underbrace{0}_{+}\underbrace{11111011}_{-5}\underbrace{\cdots}_{\text{23 bits}}$$

  If we compare them bit by bit, $x < y$, which is not the case.
- By having exponents as unsigned integers, it is easy to compare FP numbers.

IEEE 754 cont.

FP arithmetic

For a real x and rounding to nearest

$$\mathsf{fl}(x) = x(1+\epsilon), \quad |\epsilon| \le u$$

u is the unit roundoff of the precision

The arithmetic operations are correctly rounded, i.e. for x and y IEEE numbers and rounding to the nearest

$$\mathsf{fl}(x \circ y) = (x \circ y)(1+\epsilon), \quad \circ \in \{+,-,*,/\}, \quad |\epsilon| \le u$$

Also correctly rounded are

- conversions between formats and to and from strings
- square root
- fused multiply and add, FMA
  Computes $a * x + b$ with single rounding

IEEE 754 cont.

Example 6. Consider a decimal floating-point system with $t = 5$ and rounding to nearest

- The machine epsilon is $1.0001 - 1.0000 = 0.0001 = 10^{-4}$
- Unit roundoff is $u = 10^{-4}/2 = 5 \times 10^{-5}$
- Let $x = \underline{1.1626}11735194631$
  With rounding to nearest, $\mathrm{fl}(x) = 1.1626$

$$\mathrm{fl}(x) = x(1+\epsilon)$$

$$\epsilon = \frac{\mathrm{fl}(x) - x}{x} = \frac{1.1626 - 1.162611735194631}{1.162611735194631} \approx -1.0094 \times 10^{-5}$$

$$|\epsilon| \approx 1.0094 \times 10^{-5} < \underbrace{5 \times 10^{-5}}_{u}$$

IEEE 754 cont.

Example 7. Assume $t = 5$. Suppose x is close to the middle of two FP numbers, e.g. $x = \underline{1.0000}50000000000001 \times 10^4$. Then

$$\epsilon = \frac{\mathsf{fl}(x) - x}{x} = \frac{1.0001 \times 10^4 - 1.000050000000000001 \times 10^4}{1.000050000000000001 \times 10^4}$$
$$\approx 4.9998 \times 10^{-5} < 5 \times 10^{-5}$$

That is, the relative error is close to the unit roundoff of 5×10^{-5}

IEEE 754 cont.

Example 8. Assume x, y, z are FP numbers. Find the error in $\mathsf{fl}(z(x+y))$.

Since they are FP numbers, $\mathsf{fl}(x) = x$, $\mathsf{fl}(y) = y$, $\mathsf{fl}(z) = z$. Then

$$
\begin{aligned}
\mathsf{fl}(z(x+y)) &= \mathsf{fl}(z)\,\mathsf{fl}(x+y)\,(1+\delta_1) && \delta_1 \text{ roundoff in } \mathsf{fl}(z)\,\mathsf{fl}(x+y) \\
&= z(\mathsf{fl}(x)+\mathsf{fl}(y))(1+\delta_2)(1+\delta_1) && \delta_2 \text{ roundoff in } x+y \\
&= z(x+y)(1+\delta_1)(1+\delta_2) \\
&= z(x+y)(1+\delta_1+\delta_2+\delta_1\delta_2) && \text{drop } \delta_1\delta_2 \\
&\approx z(x+y)(1+\delta_1+\delta_2),
\end{aligned}
$$

where $|\delta_{1,2}| \leq u$. $|\delta_1\delta_2|$ is very small compared to $|\delta_1|$ and $|\delta_2|$, so we neglect it

Denoting $\delta = \delta_1 + \delta_2$, $|\delta| = |\delta_1 + \delta_2| \leq |\delta_1| + |\delta_2| \leq 2u$ and

$$\mathsf{fl}(z(x+y)) = z(x+y)(1+\delta), \quad \text{where} |\delta| \leq 2u$$

IEEE 754 cont.

Example 9. Assume x, y real. What is the error in $\mathrm{fl}(xy)$?

We have $\mathrm{fl}(x) = x(1+\delta_1)$, $\mathrm{fl}(y) = y(1+\delta_2)$, where $|\delta_{1,2}| \le u$.

$$\begin{aligned}
\mathrm{fl}(xy) &= \mathrm{fl}(x)\,\mathrm{fl}(y)\,(1+\delta_3) \qquad \delta_3 \text{ is the roundoff in } \mathrm{fl}(x)\,\mathrm{fl}(y)\\
&= x(1+\delta_1)y(1+\delta_2)(1+\delta_3)\\
&= xy(1+\delta_1+\delta_2+\delta_3\\
&\qquad\quad \underbrace{+\delta_1\delta_2+\delta_1\delta_3+\delta_2\delta_3+\delta_1\delta_2\delta_3}_{\text{very small}})\\
&\approx xy(1+\delta_1+\delta_2+\delta_3).
\end{aligned}$$

Denoting $\delta = \delta_1 + \delta_2 + \delta_3$,

$$|\delta| \le |\delta_1| + |\delta_2| + |\delta_3| \le 3u$$

and

$$\mathrm{fl}(xy) = xy(1+\delta), \quad \text{where } |\delta| \le 3u$$

Example 10 (Computing $\sqrt{x^2+y^2}$).

- One can do `sqrt(x*x+y*y)`
- Assume double precision and suppose `x=1e200` and `y=1e100`
- `x*x` will overflow and the result is `Inf`
- `sqrt(Inf+1e200)` gives `Inf`
- Let $M = \max\{|x|, |y|\}$ and assume $M = |x|$. Then

$$\sqrt{x^2+y^2} = M\sqrt{1+(y/M)^2}$$

- Setting `M=1e200`, `y1=y/M`, compute `M*sqrt(1+y1*y1)`, which gives `1e200`

IEEE 754 cont.

Note

expression	evaluates to
`y1=y/M`	`1e100/1e200 = 1e-100`
`y1*y1`	`1e-200`
`1+y1*y1`	`1`
`sqrt(1+y1*y1)`	`1`

Cancellations

Cancellations occur when subtracting nearby numbers that contain roundoff

Example 11. Assume a decimal FP system with $t = 5$ digits and rounding to nearest. Let $x = \underline{1.23456}7$ and $y = \underline{1.23451}2$ and compute $x - y$ in this FP system

$$
\begin{aligned}
\mathsf{fl}(x) &= \mathsf{fl}(\underline{1.23456}7) = 1.2346 && \text{roundoff error} \\
\mathsf{fl}(y) &= \mathsf{fl}(\underline{1.23451}2) = 1.2345 && \text{roundoff error} \\
\mathsf{fl}(x) - \mathsf{fl}(y) &= 0.0001 && \text{NO roundoff error} \\
&= 1.0000 \times 10^{-4}
\end{aligned}
$$

- 1 is the result of subtracting 6 and 5, both containing roundoff
- $\mathsf{fl}(x) - \mathsf{fl}(y) = 1.0000 \times 10^{-4}$ has no correct diggits: catastrophic cancellation

Cancellations cont.

Example 11. cont.

- True result is
  $x - y = 1.234567 - 1.234512 = 0.000055 = 5.5 \times 10^{-5}$
- The absolute error in $\mathrm{fl}(x) - \mathrm{fl}(y)$ is small:

$$\begin{aligned}[\mathrm{fl}(x) - \mathrm{fl}(y)] - (x - y) &= 1 \times 10^{-4} - 5.5 \times 10^{-5} \\ &= 10 \times 10^{-5} - 5.5 \times 10^{-5} \\ &= 4.5 \times 10^{-5}\end{aligned}$$

- The relative error in $\mathrm{fl}(x) - \mathrm{fl}(y)$ is

$$\frac{[\mathrm{fl}(x) - \mathrm{fl}(y)] - (x - y)}{x - y} = \frac{4.5 \times 10^{-5}}{5.5 \times 10^{-5}} = \frac{4.5}{5.5} \approx 0.82$$

or $\approx 82\%$.

Cancellations cont.

Example 12.
Let now $x = \underline{5.3845}76$ and $y = \underline{4.8940}80$

$$\begin{aligned} \mathsf{fl}(x) &= \mathsf{fl}(\underline{5.3845}76) = 5.3846 && \text{roundoff error} \\ \mathsf{fl}(y) &= \mathsf{fl}(\underline{4.8940}80) = 4.8941 && \text{roundoff error} \\ \mathsf{fl}(x) - \mathsf{fl}(y) &= 0.4905 && \text{NO roundoff error} \\ &= 4.9050 \times 10^{-1} \end{aligned}$$

- 5 is the result of subtracting 1 from 6, both containing roundoff errors
- The digits 4.90 are correct

Cancellations cont.

Example 12. cont.

- True result is $x - y = 5.384576 - 4.894080 = 0.490496$
- The absolute error in $\mathrm{fl}(x) - \mathrm{fl}(y)$ is

$$[\mathrm{fl}(x) - \mathrm{fl}(y)] - (x - y) \approx 4.0000 \times 10^{-6}$$

- The relative error in $\mathrm{fl}(x) - \mathrm{fl}(y)$ is

$$\frac{[\mathrm{fl}(x) - \mathrm{fl}(y)] - (x - y)}{x - y} \approx \frac{4.0000 \times 10^{-6}}{0.490496} \approx 8.16 \times 10^{-6}$$

Cancellations cont.

Example 13. Consider the equivalent expressions $x^2 - y^2$ and $(x-y)(x+y)$. Suppose $|x| \approx |y|$. Which one is better to evaluate? Assume $x, y > 0$; the case $x, y < 0$ is similar

- $x - y$ may have cancellations; $x + y$ does not
- x^2 and y^2 would have (in general) roundoff errors from the multiplications
- due to them, cancellations in $x^2 - y^2$ can be worse than in $(x - y)$

Try

```
x = 10000 * rand; y = x * (1 + 1e-10);
eval1 = (x - y) * (x + y); eval2 = x * x - y * y;
%compute more accurate result using vpa
xv = vpa(x); yv = vpa(y); acc = (xv - yv) * (xv + yv);
fprintf('rel. error in (x-y)*(x+y) = % e\n', (acc - eval1)/acc);
fprintf('rel. error in x*x - y*y   = % e\n', (acc - eval2)/acc);
```


Computer Arithmetic—Cancellations

CS/SE 4X03

Ned Nedialkov

McMaster University

September 14, 2023

Consider $x - y$, $x \neq y$.

Assume no roundoff in the subtraction, i.e. $\mathsf{fl}(x - y) = \mathsf{fl}(x) - \mathsf{fl}(y)$.
From $\mathsf{fl}(x) = x(1 + \epsilon_1)$, $\mathsf{fl}(y) = y(1 + \epsilon_2)$,

$$\begin{aligned}
\mathsf{fl}(x - y) &= \mathsf{fl}(x) - \mathsf{fl}(y) \\
&= x(1 + \epsilon_1) - y(1 + \epsilon_2) \\
&= (x - y) + x\epsilon_1 - y\epsilon_2 \\
&= (x - y)\left(1 + \frac{x\epsilon_1 - y\epsilon_2}{x - y}\right)
\end{aligned}$$

The error

$$\delta = \frac{x\epsilon_1 - y\epsilon_2}{x - y}$$

can be arbitrary large when $x \approx y$.

Example 1. Consider a decimal FP system with $t = 5$ digits. Let $x = 9.23450001$ and $y = 9.23455001$.

Assuming rounding to the nearest, what is the relative error in
(a) $\mathrm{fl}(x + y)$, (b) $\mathrm{fl}(x - y)$?

x and y are represented as $\mathrm{fl}(x) = 9.2345$ and $\mathrm{fl}(y) = 9.2346$

Unit roundoff is 5×10^{-5}

(a)

$$\begin{aligned}\mathrm{fl}(x + y) &= \mathrm{fl}\big[\mathrm{fl}(x) + \mathrm{fl}(y)\big] = \mathrm{fl}(9.2345 + 9.2346) = \mathrm{fl}(1.84691 \times 10)\\ &= 1.8469 \times 10\end{aligned}$$

$$\left|\frac{\mathrm{fl}(x + y) - (x + y)}{x + y}\right| = \left|\frac{1.8469 \times 10 - 1.846905002 \times 10}{1.846905002 \times 10}\right|$$
$$\approx 2.7 \times 10^{-6} < 5 \times 10^{-5}$$

Example 1. cont.

(b)

$$\mathsf{fl}(x-y) = \mathsf{fl}\big[\mathsf{fl}(x) - \mathsf{fl}(y)\big] = \mathsf{fl}(9.2345 - 9.2346) = \mathsf{fl}\big(-1.0000 \times 10^{-4}\big)$$
$$= -1.0000 \times 10^{-4}$$

$$\left|\frac{\mathsf{fl}(x-y) - (x-y)}{x-y}\right| = \left|\frac{-1.0000 \times 10^{-4} - (-5.0000 \times 10^{-5})}{-5.0000 \times 10^{-5}}\right|$$
$$= \left|\frac{-5 \times 10^{-5}}{-5 \times 10^{-5}}\right|$$
$$= 1 \gg 5 \times 10^{-5}$$

Example 2. How to evaluate $\sqrt{x+1} - \sqrt{x}$ to avoid cancellations?

For large x, $\sqrt{x+1} \approx \sqrt{x}$.

$$(\sqrt{x+1} - \sqrt{x})\frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

Evaluate

$$\frac{1}{\sqrt{x+1} + \sqrt{x}}$$

Let $x = 100000$. In a 5-digit decima arithmetic,
$x + 1 = 1.0000 \times 10^5 + 1 = 100001$ rounds to 1.0000×10^5.

Then $\sqrt{x+1} - \sqrt{x}$ gives 0, but

$$\frac{1}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{1.0000 \times 10^5} + \sqrt{1.0000 \times 10^5}} = 1.5811 \times 10^{-3}$$

Example 3. Consider approximating e^{-x} for $x > 0$ by

$$e^{-x} \approx 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots (-1)^k \frac{x^k}{k!}$$

for some k

From $e^{-x} = 1/e^x$, it is better to approximate

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^k}{k!}$$

and then compute $1/e^x$

Solving $ax^2 + bx + c$

Compute the roots of $ax^2 + bx + c = 0$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If $b^2 \gg 4ac > 0$, there may be cancellations

Example 4. Consider 4-digit decimal arithmetic and take $a = 1.01$, $b = 98.73$, $c = 4.03$.

	$=$	rounds to
b^2	9747.6129	9748
$4ac$	16.2812	16.28
$b^2 - 4ac$	$9748 - 16.28$	9732
$d = \sqrt{b^2 - 4ac}$	$\sqrt{9732}$	98.65
$-b + d$	$-98.73 + 98.65$	-0.08
$-b - d$	$-98.73 - 98.71$	-197.4
$x_1 = (-b + d)/(2a)$	$-0.08/(2.02)$	-3.960×10^{-2}
$x_2 = (-b - d)/(2a)$	$-197.4/(2.02)$	-97.72

Exact roots rounded to 4 digits -4.084×10^{-2}, -97.71

Solving $ax^2 + bx + c$ cont.

$d = \sqrt{b^2 - 4ac}$, avoid cancellations in $-b \pm d$

Use $x_1 x_2 = c/a$

Compute using

$d = \sqrt{b^2 - 4ac}$
if $b \geq 0$
  $x_1 = -(b + d)/(2a)$
  $x_2 = c/(ax_1)$
else
  $x_1 = (-b + d)/(2a)$
  $x_2 = c/(ax_1)$

This algorithm gives $x_1 = -97.71$, $x_2 = -4.084 \times 10^{-2}$

Exact roots rounded to 4 digits: -97.71, -4.084×10^{-2}

Background

CS/SE 4X03

Ned Nedialkov

McMaster University

September 18, 2023

Outline

Taylor series

Taylor series of an infinitely differentiable (real or complex) f at c

$$
\begin{aligned}
f(x) &= f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \cdots \\
&= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k
\end{aligned}
$$

Maclaurin series $c = 0$

$$
\begin{aligned}
f(x) &= f(0) + f'(c)x + \frac{f''(0)}{2!}x^2 + \cdots \\
&= \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k
\end{aligned}
$$

Taylor series cont.

Assume f has $n+1$ continuous derivative in $[a,b]$, denoted $f \in C^{n+1}[a,b]$
Then for any c and x in $[a,b]$

$$f(x) = \sum_{k}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1},$$

where

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1} \quad \text{and } \xi = \xi(c,x) \text{ is between } c \text{ and } x$$

Replacing x by $x+h$ and c by x, we obtain

$$f(x+h) = \sum_{k}^{n} \frac{f^{(k)}(x)}{k!}h^k + E_{n+1},$$

where $E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$ and ξ is between x and $x+h$

Taylor series cont.

We say the error term E_{n+1} is of order $n+1$ and write as

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1} = O(h^{n+1})$$

That is,

$$|E_{n+1}| \leq c h^{n+1}, \quad \text{for some } c > 0$$

Taylor series cont.

Example 1. How to approximate e^x for given x?

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Suppose we approximate using $e^x \approx 1 + x + \frac{x^2}{2!}$
Then

$$e^x = 1 + x + \frac{x^2}{2!} + E_3, \quad \text{where } E_3 \qquad = \frac{e^\xi}{3!}x^3, \quad \xi \text{ between } 0 \text{ and } x$$

Let $x = 0.1$. Then $e^{0.1} \approx 1.1052$. The error is

$$E_3 = \frac{e^\xi}{3!}x^3 \lessapprox \frac{1.1052}{3!}0.1^3 \approx 1.8420 \times 10^{-4}$$

Taylor series cont.

How to check our calculation?

Example 2. We can compute a more accurate value using MATLAB's `exp` function
The error in our approximation is

$$\texttt{exp(x)-(1+x+x\^{}2/2)} \approx 1.7092 \times 10^{-4}$$

This is within the bound 1.8420×10^{-4}:

$$1.7092 \times 10^{-4} < 1.8420 \times 10^{-4}$$

Taylor series cont.

Example 3. If we approximate using three terms

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

the error is

$$E_4 = \frac{e^\xi}{4!}x^4 \lessapprox \frac{1.1052}{4!}0.1^4 \approx 4.6050 \times 10^{-6}$$

Using `exp(0.1)`, the error is

$$\texttt{exp(x)-(1+x+x\^{}2/2+x\^{}3/6)} \approx 4.2514 \times 10^{-6}$$

Mean-value theorem

If $f \in C^1[a, b]$, $a < b$, then

$$f(b) = f(a) + (b - a)f'(\xi), \quad \text{for some } \xi \in (a, b)$$

From which

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

Errors in computing

Roundoff errors

Example 4.

- Consider computing `exp(0.1)`
- `0.1` binary's representation is infinite:

$$0.1_{10} = (0.0\ 0011\ 0011 \cdots)_2$$

- In floating-point arithmetic, this binary representation is rounded: roundoff error
- The input to the `exp` function is not exactly 0.1 but $0.1 + \epsilon$, for some ϵ
- The `exp` function has its own error
- Then the output of `exp(0.1)` is rounded when converting from binary to decimal

Errors in computing cont.

Truncation errors

Consider

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{k=4}^{\infty} \frac{x^k}{k!}$$

Suppose we approximate

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

That is we truncate the series. The resulting error is a truncation error

Errors in computing cont.

Approximating first derivative

$f(x)$ scalar with continuous second derivative

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)}{2}h^2, \quad \xi \text{ between } x \text{ and } x+h$$

$$f'(x)h = f(x+h) - f(x) - \frac{f''(\xi)}{2}h^2$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(\xi)}{2}h$$

If we approximate

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad \text{the truncation error is } -\frac{f''(\xi)}{2}h$$

Computational error

Computational error = (truncation error) + (rounding error)

Truncation error: difference between the true result and the result that would be produced by an algorithm using exact arithmetic

Due to e.g. truncating an infinite series or replacing a derivative by finite differences

Example 5. Replace $f'(x)$ by $(f(x+h) - f(x))/h$ From

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2} f''(\xi) h$$

the truncation error is $-\frac{1}{2} f''(\xi) h$

Computational error cont.

Rounding error: difference between the result produced using finite-precision arithmetic and exact arithmetic

Example 6. Consider evaluating

$$\frac{f(x+h)-f(x)}{h}$$

In finite-precision arithmetic, we do not compute $f(x+h)$ exactly. Denote the computed value by f_1. Then

$$f_1 = f(x+h) + \delta_1$$

for some δ_1. Similarly, we compute f_2 and for some δ_2,

$$f_2 = f(x) + \delta_2$$

Note $f(x+h)$ and $f(x)$ are the mathematically correct results, what we would compute in infinite arithmetic

f_1 and f_2 are what is computed in floating-point arithmetic

Example 6. cont.

Then we approximate $f'(x)$ by

$$\frac{f_1 - f_2}{h} = \frac{f(x+h) - f(x)}{h} + \frac{\delta_1 - \delta_2}{h}$$

Ignoring the error in the subtraction and division in $(f_1 - f_2)/h$, the total computational error is

$$\begin{aligned} f'(x) - \frac{f_1 - f_2}{h} &= \frac{f(x+h) - f(x)}{h} - \frac{1}{2}f''(\xi)h - \frac{f(x+h) - f(x)}{h} - \frac{\delta_1 - \delta_2}{h} \\ &= -\frac{1}{2}f''(\xi)h - \frac{\delta_1 - \delta_2}{h} \end{aligned}$$

$f'(x)$ is the mathematically correct value, as if computed in infinite arithmetic

Denote by M the maximum of $|f''(x)|$ for x between x and $x+h$

Assume $|\delta_1|, |\delta_1| \leq \epsilon_{\text{mach}}$

Example 6. cont.

Then

$$\begin{aligned}\left|f'(x) - \frac{f_1 - f_2}{h}\right| &= \left|-\frac{1}{2}f''(\xi)h - \frac{\delta_1 - \delta_2}{h}\right| \\ &\le \left|\frac{1}{2}f''(\xi)h\right| + \left|\frac{\delta_1 - \delta_2}{h}\right| \\ &\le \frac{1}{2}Mh + \frac{2\epsilon_{\mathsf{mach}}}{h}\end{aligned}$$

Let $g(h) = \frac{1}{2}Mh + 2\epsilon_{\mathsf{mach}}/h$. Then

$$g'(h) = \frac{1}{2}M - \frac{2\epsilon_{\mathsf{mach}}}{h^2} = 0 \quad \text{when}$$

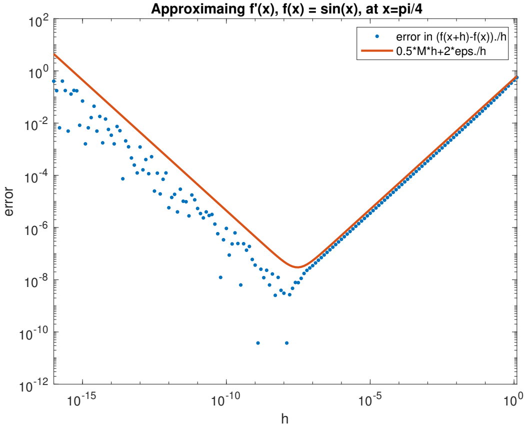
$$h^2 = \frac{4\epsilon_{\mathsf{mach}}}{M}, \qquad h = 2\sqrt{\frac{\epsilon_{\mathsf{mach}}}{M}}$$

$g(h)$ is smallest when

$$h = \frac{2}{\sqrt{M}}\sqrt{\epsilon_{\mathsf{mach}}}$$

Try

```
clear all; close all;
x = pi/4;
h = 10.^(-16:.1:.1);
f = @(x) sin(x);
fpaccurate = cos(x);
fp = (f(x+h)-f(x))./h;
error = abs(fpaccurate - fp);
M = 1;
loglog(h, error,'.', 'MarkerSize', 10);
hold on;
loglog(h, 0.5*M*h+2*eps./h, 'LineWidth',2);
xlabel('h'); ylabel('error');
title("Approximaing f'(x), f(x) = sin(x), at x=pi/4");
xlim([h(1) h(end)]);
legend('error in (f(x+h)-f(x))./h', '0.5*M*h+2*eps./h')
set(gca, 'FontSize', 12);
print("-depsc2", "deriverr.eps")
```



The error is smallest at $h \approx \sqrt{\epsilon_{\text{mach}}} \approx 10^{-8}$

Examples

Example 7. Compute `(3*(4/3-1)-1)*2^52` in your favourite language

exact value	0
double precision	-1
single precision	536870912

Example 8. This code

```
#include <stdio.h>
int main() {
  int    i = 0, j = 0;
  float  f;
  double d;
  for (f = 0.5; f < 1.0; f += 0.1)
    i++;
  for (d = 0.5; d < 1.0; d += 0.1)
    j++;
  printf("float loop %d  double loop %d \n", i, j);
}
```

outputs `float loop 5 double loop 6`

Examples cont.

Example 9. Let $a_i = i \cdot a_{i-1} - 1$, where $a_0 = e - 1$. Find a_{25}

```
#include <stdio.h>
#include <math.h>
int main(){
  int i;
  a = exp(1)-1;
  for (i = 1; i <= 25; i++)
    a = i * a - 1;
  printf("%e\n", a);
  return 0;
}
```

Matlab

```
a = exp(1)-1;
for i = 1:25
    a = i * a - 1;
end
fprintf('%e\n', a);
```

true value	$\approx$ 3.993873e-02	
C	-2.242373e+09	clang v11.0.3, MacOS X
Matlab	4.645988e+09	R2020b
Octave	-2.242373e+09	

Examples cont.

In Matlab, do `doc vpa`

- `vpa(x)`
  - uses variable-precision floating-point arithmetic (VPA)
  - evaluates `x` to $\geq$ `d` significant digits
  - `d` is the value of the `digits` function
    default default value for the number of digits is 32
- `vpa(x,d)` uses at least $\geq$ `d` significant digits

Example 9. cont.

```
clear all;
a = exp(vpa(1))-1;
for i = 1:25
    a(i+1) = i * a(i) - 1;
end
fprintf('%e \n', a(end));
```

outputs 3.993873e-02

Absolute and relative errors

Suppose y is exact result and $\widetilde{y}$ is an approximation for y

- Absolute error $|y - \widetilde{y}|$
- Relative error $|y - \widetilde{y}|/|y|$

Example 10. Suppose $y = 8.1472 \times 10^{-1}$ (accurate value), $\widetilde{y} = 8.1483 \times 10^{-1}$ (approximation). Then

$$|y - \widetilde{y}| = 1.1000 \times 10^{-4}, \qquad \frac{|y - \widetilde{y}|}{|y|} = 1.3502 \times 10^{-4}$$

Suppose $y = 1.012 \times 10^{18}$ (accurate value), $\widetilde{y} = 1.011 \times 10^{18}$ (approximation). Then

$$|y - \widetilde{y}| = 10^{15}, \qquad \frac{|y - \widetilde{y}|}{|y|} \approx 9.8814 \times 10^{-4} \approx 10^{-3}$$

Solving Linear Systems
Gauss Elimination
CS/SE 4X03

Ned Nedialkov

McMaster University

September 24, 2023

Outline

Linear systems

- Given an $n \times n$ nonsingular matrix A and an n-vector b solve

$$Ax = b$$

  The following are equivalent
  - A is nonsingular
  - The determinant of A is nonzero, $\det(A) \neq 0$
  - Columns (rows) are linearly independent
  - There exists A^{-1} such that $A^{-1}A = AA^{-1} = I$, where I is the $n \times n$ identity matrix

Linear systems cont.

- Dense system: A may have a small number of nonzeros
- Sparse system: most of the elements are zeros
  See Florida Sparse Matrix Collection
- Direct methods: based on Gauss elimination
- Iterative methods: for large A

Example

$$Ax = \begin{bmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \\ 3 \end{bmatrix} = b$$

Multiply first row by 1 and subtract from second row, multiply first row by 3 and subtract from third row

$$A|b = \left[\begin{array}{rrr|r} 1 & -1 & 3 & 11 \\ 1 & 1 & 0 & 3 \\ 3 & -2 & 1 & 3 \end{array}\right] \begin{array}{cc} \times 1 & \times 3 \\ \downarrow & \\ & \downarrow \end{array}$$

$$A|b \leftarrow \left[\begin{array}{rrr|r} 1 & -1 & 3 & 11 \\ 0 & 2 & -3 & -8 \\ 0 & 1 & -8 & -30 \end{array}\right]$$

Example cont.

Multiply second row by $\frac{1}{2}$ and subtract from third row

$$A|b \leftarrow \left[\begin{array}{rrr|r} 1 & -1 & 3 & 11 \\ 0 & 2 & -3 & -8 \\ 0 & 1 & -8 & -30 \end{array}\right] \quad \begin{array}{c} \\ \times\frac{1}{2} \\ \downarrow \end{array}$$

$$A|b \leftarrow \left[\begin{array}{rrr|r} 1 & -1 & 3 & 11 \\ 0 & 2 & -3 & -8 \\ 0 & 0 & -6.5 & -26 \end{array}\right]$$

This is Gauss elimination, also called forward elimination

Example cont.

$$\begin{bmatrix}1 & -1 & 3\\ 0 & 2 & -3\\ 0 & 0 & -6.5\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & a_{13}\\ 0 & a_{22} & a_{23}\\ 0 & 0 & a_{33}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} = \begin{bmatrix}b_1\\ b_2\\ b_3\end{bmatrix}\begin{bmatrix}11\\ -8\\ -26\end{bmatrix}$$

$$\begin{aligned}
x_3 &= b_3/a_{33} && = -26/(-6.5) && = 4\\
x_2 &= (b_2 - a_{23}x_3)/a_{22} && = (-8 - (-3)\times 4)/2 && = 2\\
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} && = (11 - (-1)\times 2 - 3\times 4)/1 && = 1
\end{aligned}$$

This is called backward substitution

Gauss elimination

Algorithm

Algorithm 3.1 (Gauss elimination).

for $k = 1 : n-1$ % *for each row*

  for $i = k+1 : n$ % *for each row below kth*

    $m_{ik} = a_{ik}/a_{kk}$ % *multiplier*

    % *update row*

    for $j = k+1 : n$

      $a_{ij} = a_{ij} - m_{ik}a_{kj}$

    $b_i = b_i - m_{ik}b_k$ % *update b_i*

Gauss elimination cont.

Cost

- We do not count the operations for updating b
- The third nested **for** loop executes $n-k$ times
  - $n-k$ multiplications
  - $n-k$ additions
- The work per one iteration of the second nested **for** loop is $2(n-k)+1$, the 1 comes from the division
- This loop executes $n-k$ times
- The total work for the second nested **for** loop is $2(n-k)^2+(n-k)$
- The work for the outermost **for** loop is

$$\sum_{k=1}^{n-1}\left[2(n-k)^2+(n-k)\right] = 2\sum_{k=1}^{n-1}k^2+\sum_{k=1}^{n-1}k$$

Gauss elimination cont.

Cost

Since $1^2 + 2^2 + 3^2 + \cdots + n^2 = n(n+1)(2n+1)/6$

$$
\begin{aligned}
\sum_{k=1}^{n-1} k^2 &= (n-1)(n-1+1)(2(n-1)+1)/6 \\
&= (n-1)n(2n-1)/6 = (n^2-n)(2n-1)/6 \\
&= (2n^3 - n^2 - 2n^2 + n)/6 = \\
&= \tfrac{1}{3}n^3 - \tfrac{1}{2}n^2 + \tfrac{1}{6}n
\end{aligned}
$$

Using the above and $\sum_{k=1}^{n-1} k = \frac{(n-1)n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$,

$$
\begin{aligned}
2\sum_{k=1}^{n-1} k^2 + \sum_{k=1}^{n-1} k &= 2\left(\tfrac{1}{3}n^3 - \tfrac{1}{2}n^2 + \tfrac{1}{6}n\right) + \tfrac{1}{2}n^2 - \tfrac{1}{2}n \\
&= \tfrac{2}{3}n^3 - n^2 + \tfrac{1}{3}n + \tfrac{1}{2}n^2 - \tfrac{1}{2}n \\
&= \tfrac{2}{3}n^3 - \tfrac{1}{2}n^2 - \tfrac{1}{6}n = \tfrac{2}{3}n^3 + O(n^2)
\end{aligned}
$$

Total work for Gauss elimination is $\frac{2}{3}n^3 + O(n^2)$

Backward substitution

- After GE, we have

$$
\begin{bmatrix}
a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\
 & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\
 & & a_{3,3} & \cdots & a_{3,n} \\
 & & & & \vdots \\
 & & & a_{n-1,n-1} & a_{n-1,n} \\
 & & & & a_{n,n}
\end{bmatrix}
\begin{bmatrix}
x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n
\end{bmatrix}
$$

- $x_n = b_n / a_{n,n}$
- $a_{n-1,n-1} x_{n-1} + a_{n-1,n} x_n = b_{n-1}$
  $x_{n-1} = (b_{n-1} - a_{n-1,n} x_n) / a_{n-1,n-1}$
- $x_k = \left(b_k - \sum_{j=k+1}^{n} a_{k,j} x_j \right) / a_{k,k}$

Backward substitution

Algorithm

Algorithm 4.1 (Backward substitution).

for $k = n : -1 : 1$

$\quad x_k = \left(b_k - \sum_{j=k+1}^{n} a_{k,j} x_j\right) / a_{k,k}$

Backward substitution

Cost

- The work per iteration is
  - $n - k$ multiplications
  - $(n - k - 1) + 1$ additions
  - 1 division
  - total $2(n - k) + 1$ operations
- Total work is

$$
\begin{aligned}
\sum_{k=1}^{n} (2(n-k)+1) &= 2\sum_{k=1}^{n}(n-k) + \sum_{k=1}^{n} 1 \\
&= 2\sum_{k=1}^{n-1} k + n = 2\frac{n(n-1)}{2} + n \\
&= n^2 - n + n = n^2
\end{aligned}
$$

Total cost

- GE: $\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$
- Backward substitution: n^2
- Total cost is

$$\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{1}{6}n = \frac{2}{3}n^3 + O(n^2) = O(n^3)$$

Gauss Elimination with Partial Pivoting (GEPP)

CS/SE 4X03

Ned Nedialkov

McMaster University

September 26, 2023

Outline

Example 1

GEPP

Example 2

Example 1. Consider

$$10^{-5}x_1 + x_2 = 1$$
$$2x_1 + x_2 = 2$$

The solution is

$$x_1^* \approx 5.000025000125 \cdot 10^{-1} \approx 0.5$$
$$x_2^* \approx 9.999949999750 \cdot 10^{-1} \approx 1$$

Solve by Gauss elimination in $t = 5$ digit decimal floating-point arithmetic

Example 1. cont.

- Eliminate with the first row, also called pivot row
- 10^{-5} is the pivot
- Multiply the first row by $2/10^{-5} = 2 \cdot 10^5$:

$$2x_1 + 2 \cdot 10^5 x_2 = 2 \cdot 10^5$$

and subtract from the second row:

$$(1 - 2 \cdot 10^5)x_2 = 2 - 2 \cdot 10^5$$

- $1 - 2 \cdot 10^5$ and $2 - 2 \cdot 10^5$ round to $-2.0000 \cdot 10^5$
- The second equation becomes

$$-2.0000 \cdot 10^5 x_2 = -2.0000 \cdot 10^5$$

from which we find $\widetilde{x}_2 = 1.0000$

Example 1. cont.

- Using $10^{-5}x_1 + x_2 = 1$, compute

$$\widetilde{x}_1 = \frac{1-\widetilde{x}_2}{10^{-5}} = \frac{0}{10^{-5}} = 0,$$

  which is quite inaccurate

- The error in $\widetilde{x}_2$ is

$$\widetilde{x}_2 - x_2^* \approx 1 - 9.99994999975 \cdot 10^{-1} \approx 5 \cdot 10^{-6}$$

- Hence

$$\widetilde{x}_2 \approx x_2^* + 5 \cdot 10^{-6}$$

Example 1. cont.

- Consider $\widetilde{x}_1$. We have

$$\widetilde{x}_1 = \frac{1-\widetilde{x}_2}{10^{-5}} \approx \frac{1-(x_2^* + 5\cdot 10^{-6})}{10^{-5}}$$
$$\approx \underbrace{\frac{1-x_2^*}{10^{-5}}}_{x_1^*} - \underbrace{5\cdot 10^{-6}}_{\text{error in } \widetilde{x}_2} \cdot \underbrace{\frac{1}{10^{-5}}}_{1/\text{pivot}}$$

$$= x_1^* \underbrace{-(\text{error in } \widetilde{x}_2)\cdot \frac{1}{\text{pivot}}}_{\text{error in } \widetilde{x}_1} = x_1^* - 0.5$$

- The error in $\widetilde{x}_2$ is multiplied by $1/\text{pivot} = 10^5$
  The error in $\widetilde{x}_1$ is -0.5

Example 1. cont.

- Avoid small pivots. Swap the equations

$$\begin{aligned} 2x_1 + x_2 &= 2 \\ 10^{-5}x_1 + x_2 &= 1 \end{aligned}$$

- Multiply the first row by $10^{-5}/2$:

$$10^{-5}x_1 + \frac{10^{-5}}{2}\,x_2 = 10^{-5}$$

  and subtract from the second row

$$\left(1 - \frac{10^{-5}}{2}\right)x_2 = 1 - 10^{-5}$$

- $1 - 10^{-5}/2$ and $1 - 10^{-5}$ round to 1

Example 1. cont.

- The second equation is $x_2 = 1$, find $\widetilde{x}_2 = 1$
- Using $2x_1 + x_2 = 2$, $\widetilde{x}_1 = \frac{2-\widetilde{x}_2}{2} = 0.5$
- Using $\widetilde{x}_2 \approx x_2^* + 5 \cdot 10^{-6}$

$$
\begin{aligned}
\widetilde{x}_1 &= \frac{2-\widetilde{x}_2}{2} \approx \frac{2-(x_2^* + 5 \cdot 10^{-6})}{2} \\
&= \underbrace{\frac{2-x_2^*}{2}}_{x_1^*} - \underbrace{5 \cdot 10^{-6}}_{\text{error in } \widetilde{x}_2} \cdot \underbrace{\frac{1}{2}}_{1/\text{pivot}} \\
&= x_1^* \underbrace{-(\text{error in } \widetilde{x}_2) \cdot \frac{1}{\text{pivot}}}_{\text{error in } \widetilde{x}_1} \\
&= x_1^* - 2.5 \cdot 10^{-6}
\end{aligned}
$$

GEPP

GEPP

- Eliminate with the row with the largest (in magnitude) entry

Example 2. Solve

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_1 + 1.0001x_2 + 2x_3 &= 2 \\ x_1 + 2x_2 + 2x_3 &= 3 \end{aligned}$$

with partial pivoting and $t = 5$ decimal arithmetic

Can chose any row to eliminate x_1. Use first row:

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 0.0001x_2 + x_3 &= 1 \\ x_2 + x_3 &= 2 \end{aligned}$$

Swap rows 2 and 3 and eliminate with second row

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_2 + x_3 &= 2 \\ 0.0001x_2 + x_3 &= 1 \end{aligned} \quad \rightarrow \quad \begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_2 + x_3 &= 2 \\ (1 - 0.0001)x_3 &= 1 - 0.0002 \end{aligned}$$

Example 2. cont. Using MATLAB's backslash operator, `A\b` where

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

we obtain

$$[-1, 1.000100010001, 9.99899989999 \cdot 10^{-1}]$$

In 5-digit arithmetic,

$$\begin{aligned} 0.9999x_3 &= 0.9998 & & \\ x_3 &= 9.9990 \cdot 10^{-1} & \text{error } &\approx 10^{-8} \\ x_2 &= 2 - x_3 = 1.0001 & \text{error } &\approx -10^{-8} \\ x_1 &= 1 - x_2 - x_3 = -1 & \text{error } &\approx 0 \end{aligned}$$

The errors in x_1, x_2, x_3 are (in absolute value) $\approx 0, 10^{-8}, 10^{-8}$, respectively.

Example 2. cont.

If we eliminate with the second row, we multiply it by 10^4

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 0.0001x_2 + x_3 &= 1 \\ x_2 + x_3 &= 2 \end{aligned} \quad \rightarrow \quad \begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 0.0001x_2 + x_3 &= 1 \\ -9.9990 \cdot 10^3 x_3 &= -9.9980 \cdot 10^3 \end{aligned}$$

Then

$$\begin{aligned} x_3 &= 9.9990 \cdot 10^{-1} & \text{error in } x_3\text{: } \approx 10^{-8} \\ x_2 &= \frac{1 - x_3}{0.0001} = (1 - x_3) \cdot 10^4 = 1.0000 & -(\text{error in } x_3) \cdot 10^4 \approx -10^{-4} \\ x_1 &= 1 - x_2 - x_3 = -9.9990 \cdot 10^{-1} & \text{error} \approx 10^{-4} - 10^{-8} \approx 10^{-4} \end{aligned}$$

The errors now are (in absolute value) $\approx 10^{-4}, 10^{-4}, 10^{-8}$

LU Decomposition

CS/SE 4X03

Ned Nedialkov

McMaster University

October 2, 2023

Outline

LU decomposition

- Decompose A as $A = LU$, where
  - L is unit lower-triangular
    1's on the main diagonal, 0's above it
  - U is upper-triangular
    0's below the main diagonal
- Consider solving $Ax = b$. From

$$\begin{aligned} Ax = LUx &= b \\ L\underbrace{(Ux)}_{y} &= b \end{aligned}$$

we can solve first $Ly = b$ for y and then $Ux = y$ for x

LU decomposition cont.

A is $n \times n$

- Gauss elimination takes $O(n^3)$ arithmetic operations
- LU decomposition takes $O(n^3)$ arithmetic operations
- Solving each of $Ly = b$ and $Ux = y$ takes $O(n^2)$ arithmetic operations
- Suppose we need to solve m systems $Ax = b^{(i)}$, $i = 1, \ldots, m$
  A is the same, the right-hand side changes
- If we solve them with GE $\quad O(mn^3)$
- Do LU decomposition first $\quad O(n^3)$
- Solve $Ly = b^{(i)},\ Ux = y$, for $i = 1 : m$ $\quad O(mn^2)$
  Total LU+triangular solves $\quad O(n^3 + mn^2)$

Example of LU decomposition

$$A = \begin{array}{c} \begin{bmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} \end{array} \begin{array}{cc} \times 1 & \times 3 \\ \downarrow & \\ & \downarrow \end{array}$$

- multipliers $l_{2,1} = 1$, $l_{3,1} = 3$

$$M_1 A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 1 & -8 \end{bmatrix} = A^{(1)}$$

- multiplier $l_{3,2} = \frac{1}{2}$

$$M_2A^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 1 & -8 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 0 & -6.5 \end{bmatrix} = A^{(2)} = U$$

We have

$$M_2A^{(1)} = (M_2M_1)A = U$$
$$A = \underbrace{(M_1^{-1}M_2^{-1})}_{L}U$$

To compute M_1^{-1}, M_2^{-1} flip the signs of nonzero entries below the main diagonal

Then

$$L = M_1^{-1} M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & \frac{1}{2} & 1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & \frac{1}{2} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 0 & -6.5 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix}}_{A}$$

Small pivots

- The matrix

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

  is nonsingular, but does not have LU factorization
  Gauss elimination breaks down on this matrix since the multiplier is $1/0$

-

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

  is singular and has the LU factorization

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = LU$$

Consider

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}$$

- Multiply the first row by $1/\epsilon$ and subtract from the second

$$L = \begin{bmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{bmatrix}$$

- When ϵ small, in floating-point arithmetic,

$$U \approx \begin{bmatrix} \epsilon & 1 \\ 0 & -\frac{1}{\epsilon} \end{bmatrix}$$

as $1 - \frac{1}{\epsilon} \approx -\frac{1}{\epsilon}$. Take e.g. $\epsilon = 10^{-16}$ in double precision

$$LU \approx \begin{bmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -\frac{1}{\epsilon} \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} = A$$

- Loss of accuracy

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}$$

- Permute the rows

$$\overline{A} = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix}$$

- Multiple first row by ϵ and subtract from second row

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{bmatrix}$$

$$\overline{L} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix}, \qquad \overline{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{bmatrix}$$

- Permuting the rows of A is PA, where P is permutation matrix

$$PA = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix}$$

Partial pivoting

- If a pivot is small, then $1/(\text{pivot})$ is large
- Roundoff errors are multiplied

Partial pivoting

- at step $k = 1 : n - 1$ chose the row q for which $|a_{qk}|$ is the largest
- eliminate with row q
  now we divide by the largest element in column k

MATLAB's lu

`[L,U,P] = lu(A)` returns `L` unit lower triangular, `U` upper triangular, and `P` a permutation matrix such that `A = P'*L*U`.

That is $A = P^T LU$, $PA = LU$

`[L,U] = lu(A)` returns permuted lower triangular `L` and upper triangular `U` such that `A = L*U`.

Example 1.

Find the LU decomposition of

$$\begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 8 & 2 & 3 \end{bmatrix}$$

To eliminate with the first row, the multipliers are $1/4$ and 2. We have

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 0.75 & 1.5 \\ 0 & -8 & -9 \end{bmatrix}$$

To eliminate with the second row, the multiplier is $-8/0.75$. We have

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 0.75 & 1.5 \\ 0 & 0 & 7 \end{bmatrix}$$

Example 1. cont.

Then

$$\begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 8 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 2 & -8/0.75 & 1 \end{bmatrix} \begin{bmatrix} 4 & 5 & 6 \\ 0 & 0.75 & 1.5 \\ 0 & 0 & 7 \end{bmatrix}$$

Example 2.

Using partial pivoting, find the LU decomposition of

$$\begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 8 & 2 & 3 \end{bmatrix}$$

We pivot with the third row. To swap the first and third rows,

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{P_1} \begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 8 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 2 & 3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

To eliminate with the first row, the multipliers are $1/8$ and $1/2$. We have

$$\begin{bmatrix} 8 & 2 & 3 \\ 0 & 1.75 & 21/8 \\ 0 & 4 & 4.5 \end{bmatrix}$$

Example 2. cont.

Now we need to swap rows 2 and 3. This is the same as multiplying by a permutation matrix

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{P_2} \begin{bmatrix} 8 & 2 & 3 \\ 0 & 1.75 & 21/8 \\ 0 & 4 & 4.5 \end{bmatrix} = \begin{bmatrix} 8 & 2 & 3 \\ 0 & 4 & 4.5 \\ 0 & 1.75 & 21/8 \end{bmatrix}$$

Now the multiplier is $1.75/4$ and we have

$$\begin{bmatrix} 8 & 2 & 3 \\ 0 & 4 & 4.5 \\ 0 & 0 & 0.6562 \end{bmatrix}$$

Example 2. cont.

The total permutation is

$$P = P_2 P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Then

$$PA = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 8 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/8 & 1 & 0 \\ 1/2 & 1.75/4 & 1 \end{bmatrix} \begin{bmatrix} 8 & 2 & 3 \\ 0 & 4 & 4.5 \\ 0 & 0 & 0.6562 \end{bmatrix} = LU$$

Check this result with Matlab's `lu`.

Errors in Linear Systems Solving

CS/SE 4X03

Ned Nedialkov

McMaster University

October 2, 2023

Outline

Norms

Residual

Relative solution error

Norms

Vector norms

Norm is a function $\|\cdot\|$ that satisfies for any $x \in \mathbb{R}^n$

1. $\|x\| \geq 0$, and $\|x\| = 0$ iff $x = 0$, the zero vector
2. $\|\alpha x\| = |\alpha|\|x\|$, $\alpha \in \mathbb{R}$
3. $\|x + y\| \leq \|x\| + \|y\|$ for any $x, y \in \mathbb{R}^n$

lp norms

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \qquad 1 \leq p \leq \infty$$

Norms cont.

- $p = 1$, one norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

- $p = \infty$, infinity or max norm

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$$

- $p = 2$, two or Euclidean norm

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Norms cont.

Matrix norms

- $A \in \mathbb{R}^{m\times n}$, $\|\cdot\|$ is a vector norm
- Matrix norm induced by this vector norm

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

- Properties
  1. $\|A\| \geq 0$, and $\|A\| = 0$ iff $A = 0$, the zero matrix
  2. $\|\alpha A\| = |\alpha|\|A\|$, $\alpha \in \mathbb{R}$
  3. $\|A + B\| = \|A\| + \|B\|$, for any $A, B \in \mathbb{R}^{m\times n}$
  4. $\|AB\| \leq \|A\| \cdot \|B\|$, for any $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$

- Infinity norm, max row sum

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$$

- One norm, max column sum

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$$

- Two norm

$$\|A\|_2 = \max_i \sqrt{\lambda_i(A^TA)},$$

where $\lambda_i(A^TA)$ is the ith eigenvalue of A^TA

Residual

Consider $Ax = b$

- Let $\widetilde{x}$ be the computed solution, and let x be the exact solution
- Relative error in the solution is

$$\frac{\|x - \widetilde{x}\|}{\|x\|}$$

- Residual is

$$r = b - A\widetilde{x}$$

$$r = 0 \iff b - A\widetilde{x} = 0 \iff \widetilde{x} = x$$

- In practice $r \neq 0$

- $Ax = b$ and $\alpha Ax = \alpha b$ have the same solution
  α is a scalar
- $r_\alpha = \alpha b - \alpha A\widetilde{x} = \alpha(b - A\widetilde{x})$ can be arbitrarily large
- residual can be arbitrarily large

Residual cont.

Example 1. Consider

$$A = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}, \qquad b = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix}$$

and the approximate solution $\widetilde{x} = [0.9911, -0.487]^T$

- The residual is small:

$$r = b - A\widetilde{x} \approx [10^{-8}, -10^{-8}]^T, \qquad \|r\|_\infty \approx 10^{-8}$$

- The exact solution is $x = [2,\ -2]^T$. The error in $\widetilde{x}$ is large:

$$x - \widetilde{x} = [1.513, -1.0089], \qquad \|x - \widetilde{x}\|_\infty = 1.513$$

- Small residual does not imply small solution error

Relative solution error

Given $\widetilde{x}$, how large is

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \tag{1}$$

Using $r = b - A\widetilde{x} = Ax - A\widetilde{x} = A(x - \widetilde{x})$,

$$\begin{aligned} x - \widetilde{x} &= A^{-1}r \\ \|x - \widetilde{x}\| &= \|A^{-1}r\| \leq \|A^{-1}\|\|r\| \end{aligned} \tag{2}$$

Using $b = Ax$, $\|b\| = \|Ax\| \leq \|A\|\|x\|$, and

$$\|x\| \geq \frac{\|b\|}{\|A\|} \tag{3}$$

The condition number of A is

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

Using (2–3) in (1),

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \frac{\|A^{-1}\|\|r\|}{\frac{\|b\|}{\|A\|}} = \|A^{-1}\|\|A\|\frac{\|r\|}{\|b\|} = \text{cond}(A)\frac{\|r\|}{\|b\|}$$

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \text{cond}(A)\frac{\|r\|}{\|b\|}$$

- If $\text{cond}(A)$ is not large and $\|r\|/\|b\|$ is small then small relative error
- As a rule of thumb, if $\text{cond}(A) \approx 10^k$, then about k decimal digits are lost when solving $Ax = b$.

- In our example

$$A^{-1} = 10^8 \begin{bmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2869 \end{bmatrix}$$

- In the two norm, $\text{cond}(A) \approx 2.4973 \cdot 10^8$

$$\begin{aligned} \text{cond}(A)\frac{\|r\|}{\|b\|} &\approx 4.0311 \\ \frac{\|x - \widetilde{x}\|}{\|x\|} &\approx 0.6429 \end{aligned}$$

Polynomial Interpolation

CS/SE 4X03

Ned Nedialkov

McMaster University

October 3, 2023

Outline

- The problem
- Representation
- Basis functions
- Monomial interpolation
- Uniqueness of the interpolating polynomial
- Lagrange interpolation

The problem

Given data points $\left\{(x_i, y_i)\right\}_{i=0}^{n}$ find a function $v(x)$ that fits the data such that

$$v(x_i) = y_i, \qquad i = 0, \ldots, n$$

Some applications

- Approximating functions. For a complicated function $f(x)$ find a simpler $v(x)$ that approximates $f(x)$. Usually it is less expensive to work with $v(x)$ than with $f(x)$
- We can use $v(x)$ to approximate $f(x)$ at some $x^* \neq x_0, x_1, \ldots x_n$
- We may need derivatives or an integral of f, and we can differentiate/integrate v

Representation

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x)$$

- The c_j are unknown coefficients
- The ϕ_j are given basis functions
  They must be linearly independent
  If $v(x) = 0$ for all x then $c_j = 0$ for all j

Representation cont.

From

$$v(x_i) = c_0\phi_0(x_i) + c_1\phi_1(x_i) + \cdots + c_n\phi_n(x_i) = y_i, \quad i = 0, \ldots, n$$

we have the linear system of $(n+1)$ equations for the c_i

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Basis functions

- Monomial basis

$$\phi_j(x) = x^j, \quad j = 0, 1, \ldots, n$$
$$v(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

- Trigonometric functions, e.g.

$$\phi_j(x) = \cos(jx), \quad j = 0, 1, \ldots, n$$

  Useful in signal processing, for wave and other periodic behavior

- Piecewise interpolation: linear, quadratic, cubic, splines

Monomial interpolation

The polynomial is of the form $p_n(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$

Example 1. Interpolate

x_i	1	2	4
y_i	1	3	3

using a polynomial of degree 2. We seek the coefficients of $p_2(x) = c_0 + c_1x + c_2x^2$

From

$$\begin{aligned} p_2(1) &= c_0 + c_1 + 1c_2 = 1 \\ p_2(2) &= c_0 + 2c_1 + 4c_2 = 3 \\ p_2(4) &= c_0 + 4c_1 + 16c_2 = 3 \end{aligned}$$

Solve this linear system to obtain

$$p_2(x) = -\tfrac{7}{3} + 4x - \tfrac{2}{3}x^2$$

Uniqueness of the interpolating polynomial

From

$$p_n(x_i) = c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n = y_i$$

we have the linear system

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

- The coefficient matrix is a Vandermonde matrix
  Denote it by X
- $\det(X) = \prod_{i=0}^{n-1} \left[\prod_{j=i+1}^{n} (x_j - x_i) \right]$

Uniqueness of the interpolating polynomial cont.

If all x_i are distinct then

- $\det(X) \neq 0$
- X is nonsingular
- this system has a unique solution
- there is a unique polynomial of degree $\leq n$ that interpolates the data

However,

- this system can be poorly conditioned
- work is $O(n^3)$
- difficult to add new points

Lagrange interpolation

- Lagrange basis functions

$$L_j(x_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

- Lagrange polynomial $p_n(x) = \sum_{j=0}^{n} y_j L_j(x)$

Then

$$\begin{aligned} p_n(x_i) &= \sum_{j=0}^{n} y_j L_j(x_i) \\ &= \sum_{j=0}^{i-1} y_j \underbrace{L_j(x_i)}_{=0} + y_i \underbrace{L_i(x_i)}_{=1} + \sum_{j=i+1}^{n} y_j \underbrace{L_j(x_i)}_{=0} \\ &= y_i \end{aligned}$$

Lagrange interpolation cont.

$$
\begin{aligned}
L_j(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\
&= \prod_{i=0, i\neq j}^{n} \frac{x-x_i}{x_j-x_i}
\end{aligned}
$$

Example: write the Lagrange polynomial for $(1,1)$, $(2,3)$, $(4,3)$

Polynomial Interpolation
Newton's Form
CS/SE 4X03

Ned Nedialkov

McMaster University

October 15, 2023

Outline

Basis

- Basis functions are

$$\phi_j(x) = \prod_{i=0}^{j-1}(x - x_i) = (x - x_0)(x - x_1)\cdots(x - x_{j-1}), \quad j = 0 : n$$

- Example: for a cubic interpolant, we have

$$\begin{aligned}\phi_0(x) &= 1\\ \phi_1(x) &= x - x_0\\ \phi_2(x) &= (x - x_0)(x - x_1)\\ \phi_3(x) &= (x - x_0)(x - x_1)(x - x_2)\end{aligned}$$

Computing coefficients

Let $y_i = f(x_i)$. From

$$\begin{aligned} p_n(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots \\ &\qquad + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \\ p_n(x_i) &= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots \\ &\qquad + c_n(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{n-1}) = f(x_i) \end{aligned}$$

at $x = x_0$, we have

$$\begin{aligned} p_n(x_0) &= c_0 + c_1(x_0 - x_0) + c_2(x_0 - x_0)(x_0 - x_1) + \cdots \\ &\qquad + c_n(x_0 - x_0)(x_0 - x_1)\cdots(x_0 - x_{n-1}) = f(x_0) \\ c_0 &= f(x_0) \end{aligned}$$

Computing coefficients

At x_1,

$$p_n(x_1) = c_0 + c_1(x_1 - x_0) + c_2(x_1 - x_0)(x_1 - x_1) + \cdots$$
$$+ c_n(x_1 - x_0)(x_1 - x_1)\cdots(x_1 - x_{n-1}) = f(x_1)$$

$$\begin{aligned} c_0 + c_1(x_1 - x_0) &= f(x_1) \\ c_1 &= \tfrac{f(x_1) - c_0}{x_1 - x_0} \\ &= \tfrac{f(x_1) - f(x_0)}{x_1 - x_0} \end{aligned}$$

Computing coefficients

At x_2,

$$\begin{aligned} p_n(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) \\ + c_3(x_2 - x_0)(x_2 - x_1)(x_2 - x_2) + \cdots \\ + c_n(x_1 - x_0)(x_1 - x_1) \cdots (x_1 - x_{n-1}) = f(x_1) \end{aligned}$$

Then

$$c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = f(x_2)$$

$$c_2 = \frac{f(x_2) - c_0 - c_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

Exercise: verify the last equality

Divided differences

Given $x_0, x_1, \ldots, x_n$, where $0 \le i < j \le n$, define

$$f[x_i] = f(x_i)$$

$$f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}$$

$f[x_i, \ldots, x_j]$ are divided differences over $x_i, \ldots, x_j$

Divided differences

$$
\begin{aligned}
c_0 &= f(x_0) = f[x_0] \\
c_1 &= \tfrac{f(x_1)-f(x_0)}{x_1-x_0} = f[x_0, x_1] \\
c_2 &= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0} = \tfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} = f[x_0, x_1, x_2] \\
&\vdots \\
c_n &= \tfrac{f[x_1,\ldots,x_n]-f[x_0,\ldots,x_{n-1}]}{x_n-x_0} = f[x_0, x_1, \ldots, x_n]
\end{aligned}
$$

$$
\begin{aligned}
p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
+ f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1})
\end{aligned}
$$

Example

i	x_i	$f[x_i]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$
0	1	1		
1	2	3	2	
2	4	3	0	$-\frac{2}{3}$

$$
\begin{aligned}
p_2(x) &= f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) \\
&= 1 + 2(x-1) - \tfrac{2}{3}(x-1)(x-2)
\end{aligned}
$$

Example

Suppose we add a new point $(3, 5)$

Then

i	x_i	$f[x_i]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
0	1	1			
1	2	3	2		
2	4	3	0	$-\frac{2}{3}$	
3	3	5	-2	-2	$-\frac{2}{3}$

$$
\begin{aligned}
p_3(x) = 1 + 2(x-1) - \tfrac{2}{3}(x-1)(x-2) \\
- \tfrac{2}{3}(x-1)(x-2)(x-4)
\end{aligned}
$$

Errors in Polynomial Interpolation

CS/SE 4X03

Ned Nedialkov

McMaster University

October 16, 2023

Outline

Polynomial interpolation error

Chebyshev nodes

Polynomial interpolation error

- Assume
  - Polynomial p_n of degree $\leq n$ interpolates f at $n+1$ distinct points $x_0, x_1, \ldots, x_n$, where $x_i \in [a, b]$
  - $f^{(n+1)}$ is continuous on $[a, b]$
- Then, for each $x \in [a, b]$, there is a $\xi = \xi(x) \in (a, b)$ such that

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

Polynomial interpolation error cont.

- Let $M = \max_{a \le t \le b} |f^{(n+1)}(t)|$
  Then
  $$|f(x) - p_n(x)| \le \frac{M}{(n+1)!} \prod_{i=0}^{n} |x - x_i|$$
- Let $h = (b-a)/n$ and let $x_i = a + ih$ for $i = 0, 1, \ldots, n$
  Then
  $$|f(x) - p_n(x)| \le \frac{M}{4(n+1)} h^{n+1}$$

Polynomial interpolation error cont.

Example 1. Consider $\cos(x)$ and assume values $f(x_i) = \cos(x_i)$ are given at 11 equally spaced points in $[a, b] = [-\pi, \pi]$. What is the error in the interpolating polynomial?

Here $n = 10$ and $h = (b - a)/n = 2\pi/10$.
$M = \max_{-\pi \le t \le \pi} |\cos^{(n+1)}(t)| = 1$.

Then

$$|f(x) - \cos(x)| \le \frac{M}{4(n+1)} h^{n+1} = \frac{1}{4(11)} (2\pi/10)^{11} \approx 1.3694 \times 10^{-4}$$

Chebyshev nodes

- Suppose $f(x_i)$ is given at $n+1$ distinct points $x_0, x_1, \ldots, x_n$ in $[a, b]$ and $p_n(x)$ of degree $\leq n$ interpolates f at these points
- We have for the error

$$\max_{x\in[a,b]} |f(x) - p_n(x)| \leq \frac{M}{(n+1)!} \max_{s\in[a,b]} \left| \prod_{i=0}^{n} (s - x_i) \right|$$

where $M = \max_{t\in[a,b]} |f^{(n+1)}(t)|$

- How to chose the x_i so

$$\max_{s\in[a,b]} \left| \prod_{i=0}^{n} (s - x_i) \right|$$

is minimized?

Chebyshev nodes cont.

- Chebyshev nodes on $[-1, 1]$:

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \quad i = 0, 1, \dots, n$$

- Min-max property: over all possible x_i they minimize $\max_{s\in[-1,1]} |(s - x_0)(s - x_1)\cdots(s - x_n)|$

$$\min_{x_0,x_1,\dots,x_n} \max_{s\in[-1,1]} |(s - x_0)(s - x_1)\cdots(s - x_n)| = 2^{-n}$$

- Error bound using Chebyshev nodes in $[-1, 1]$:

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \leq \frac{M}{2^n(n+1)!}$$

$M = \max_{t\in[-1,1]} |f^{(n+1)}(t)|$

Chebyshev nodes cont.

- For a general $[a, b]$,

$$x_i = 0.5(a+b) + 0.5(b-a)\cos\left(\frac{2i+1}{2n+2}\pi\right), \quad i = 0, 1, \ldots, n$$

Example 2. In the previous example, if we chose Chebyshev nodes,

$$|f(x) - \cos(x)| \leq \frac{M}{2^n(n+1)!} = \frac{1}{2^{10}(10+1)!} \approx 2.4465 \times 10^{-11}$$

Numerical Integration: Basic Rules

CS/SE 4X03

Ned Nedialkov

McMaster University

October 24, 2023

Outline

The problem

- Approximate numerically the integral

$$I_f = \int_a^b f(x)dx$$

- Closed form may not exist, e.g. $\int_a^b e^{-x^2}dx$, or may be difficult to compute
- The integrand $f(x)$ may be known only at certain points obtained via sampling (e.g. embedded applications)

Derivation

$$I_f = \int_a^b f(x)dx \approx \sum_{j=0}^{n} a_j f(x_j)$$

- The sum is called a *quadrature rule*
- The a_j are weights
- How to find them?

Derivation cont.

- Let $x_0, \ldots, x_n$ be distinct points in $[a, b]$
- Let $p_n(x)$ be the interpolating polynomial for $f(x)$ through these points
- $\int_a^b f(x)dx \approx \int_a^b p_n(x)dx$
- From the Lagrange form $p_n(x) = \sum_{j=0}^n f(x_j)L_j(x)$,

$$\int_a^b f(x)dx \approx \int_a^b p_n(x)dx = \int_a^b \sum_{j=0}^n f(x_j)L_j(x)dx$$

$$= \sum_{j=0}^n f(x_j) \underbrace{\int_a^b L_j(x)dx}_{a_j}$$

- $a_j = \int_a^b L_j(x)dx$

Trapezoidal rule

Let $n = 1$. Then $x_0 = a$ and $x_1 = b$ and

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{x - b}{a - b}, \quad L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - a}{b - a}$$

$$\begin{aligned} f(x) \approx p_1(x) &= f(x_0)L_0(x) + f(x_1)L_1(x) \\ &= f(a)L_0(x) + f(b)L_1(x) \end{aligned}$$

Integrating

$$\begin{aligned} I_f = \int_a^b f(x)dx &\approx f(a) \underbrace{\int_a^b L_0(x)dx}_{a_0} + f(b) \underbrace{\int_a^b L_1(x)dx}_{a_1} \\ &= f(a) \int_a^b \frac{x - b}{a - b} dx + f(b) \int_a^b \frac{x - a}{b - a} dx \\ &= \frac{b - a}{2} [f(a) + f(b)] \end{aligned}$$

Trapezoidal rule cont.

$$I_f \approx I_{\text{trap}} = \frac{b-a}{2}\left[f(a) + f(b)\right]$$

Example 1.

- Approximate $\int_0^1 e^x dx = e - 1 = 1.7182\ldots$ using the trapezoidal rule:
$$I_{\text{trap}} = \frac{1}{2}[f(0) + f(1)] = 0.5(1+e) = 1.8591\cdots$$

- Approximate $\int_0^{0.1} e^x dx = e^{0.1} - 1 = 0.10517\cdots$ using the trapezoidal rule:

$$I_{\text{trap}} = \frac{0.1}{2}[f(0) + f(0.1)] = 0.05\left(1 + e^{0.1}\right) = 0.10525\cdots$$

Errror of trapezoidal rule

In the trapezoidal rule, $f(x)$ is approximated by linear interpolation

$$p_1(x) = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a}$$

The error is

$$f(x) - p_1(x) = \tfrac{1}{2} f''(\xi(x))(x-a)(x-b)$$

Then

$$\begin{aligned}
\int_a^b (f(x) - p_1(x))dx &= \int_a^b f(x)dx - \frac{b-a}{2}[f(a) + f(b)] \\
&= \frac{1}{2}\int_a^b f''(\xi(x))(x-a)(x-b)dx
\end{aligned}$$

Errror of trapezoidal rule cont.

$(x-a)(x-b) \leq 0$ does not change sign on $[a,b]$

From the Mean-Value Theorem for integrals, there exists $\eta \in (a,b)$ such that

$$\int_a^b f''(\xi(x))(x-a)(x-b)dx = f''(\eta)\int_a^b (x-a)(x-b)dx$$

Using $\int_a^b (x-a)(x-b)dx = -(b-a)^3/6$, the error in the trapezoidal rule is

$$I_f - I_{\text{trap}} = -\frac{f''(\eta)}{12}(b-a)^3$$

Midpoint rule

$$I_f \approx I_{\mathsf{mid}} = (b-a) f\left(\frac{a+b}{2}\right)$$

Example 2.

- Approximate $\int_0^1 e^x dx = e - 1 \approx 1.7182\cdots$ using the midpoint rule:

$$I_{\mathsf{mid}} = (1-0) f(0.5) = e^{0.5} = 1.6487\cdots$$

- Approximate $\int_0^{0.1} e^x dx = e^{0.1} - 1 \approx 0.10517\cdots$ using the midpoint rule:

$$I_{\mathsf{mid}} = (0.1-0) f(0.05) = 0.1e^{0.05} = 0.10512\cdots$$

Error of midpoint rule

Let $m = (a+b)/2$. Expand f in Taylor series

$$f(x) = f(m) + f'(m)(x-m) + \tfrac{1}{2} f''(\xi(x))(x-m)^2$$

Then

$$I_f = \int_a^b f(x) = \underbrace{(b-a)f(m)}_{I_{\text{mid}}} + \frac{1}{2}\int_a^b f''(\xi(x))(x-m)^2 dx$$

Since $(x-m)^2$ does not change sign, there exists $\eta \in (a,b)$ such that

$$\frac{1}{2}\int_a^b f''(\xi(x))(x-m)^2 dx = \frac{1}{2} f''(\eta)\int_a^b (x-m)^2 dx = \frac{f''(\eta)}{24}(b-a)^3$$

Then

$$I_f - I_{\text{mid}} = \frac{f''(\eta)}{24}(b-a)^3$$

Simpson's rule

Let $n = 2$, and $x_0 = a$, $x_1 = (a + b)/2$, $x_2 = b$

Simpson's rule is obtained from integrating the second order polynomial

$$\begin{aligned} p_2(x) &= f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x) \\ &= f(a)L_0(x) + f((a + b)/2)L_1(x) + f(b)L_2(x) \end{aligned}$$

$$I_f \approx I_{\mathsf{Simpson}} = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

The error is

$$I_f - I_{\mathsf{Simpson}} = -\frac{f^{(4)}(\xi)}{90}\left(\frac{b-a}{2}\right)^5, \quad \xi \in (a, b)$$

Simpson's rule cont.

Example 3. Approximate $\int_0^1 e^x dx = e - 1 \approx 1.71828\cdots$ using Simpson's rule:

$$I_{\mathsf{Simpson}} = \frac{1}{6}\left[f(0) + 4f\left(0.5\right) + f(1)\right] = \frac{1}{6}(1 + 4e^{0.5} + e)$$
$$= 1.71886\cdots$$

Numerical Integration
Composite Rules
CS/SE 4X03

Ned Nedialkov

McMaster University

October 26, 2023

Outline

How to increase the accuracy of a rule

- We can increase the degree of the polynomial, but the error might be large
- Apply a basic rule over small subintervals
  - subdivide $[a, b]$ into r subintervals
  - $h = \frac{b-a}{r}$ length of each subinterval
  - $t_i = a + ih$, $i = 0, 1, \ldots, r$
    $t_0 = a$, $t_r = b$

$$\int_a^b f(x)dx = \sum_{i=1}^{r} \int_{t_{i-1}}^{t_i} f(x)dx$$

Composite trapezoidal rule

From the basic rule on $[t_{i-1}, t_i]$, $i = 1, \ldots, r$

$$\int_{t_{i-1}}^{t_i} f(x)dx \approx \frac{t_i - t_{i-1}}{2}\left[f(t_{i-1}) + f(t_i)\right] = \frac{h}{2}\left[f(t_{i-1}) + f(t_i)\right]$$

we derive

$$\begin{aligned}
\int_a^b f(x)dx &= \sum_{i=1}^{r} \int_{t_{i-1}}^{t_i} f(x)dx \approx \frac{h}{2}\sum_{i=1}^{r}\left[f(t_{i-1}) + f(t_i)\right] \\
&= \frac{h}{2}\left(\sum_{i=1}^{r} f(t_{i-1}) + \sum_{i=1}^{r} f(t_i)\right) \\
&= \frac{h}{2}\left(f(t_0) + f(t_1) + \cdots + f(t_{r-1})\right) \\
&+ \frac{h}{2}\left(\qquad\qquad f(t_1) + \cdots + f(t_{r-1}) + f(t_r)\right) \\
&= \frac{h}{2}\left[f(a) + f(b)\right] + h\sum_{i=1}^{r-1} f(t_i)
\end{aligned}$$

Error of composite trapezoidal rule

From

$$\int_{t_{i-1}}^{t_i} f(x)dx = \frac{h}{2}\left[f(t_{i-1}) + f(t_i)\right] - \frac{f''(\eta_i)}{12}h^3$$

we have

$$\int_a^b f(x)dx = \underbrace{\sum_{i=1}^{r} \frac{h}{2}\left[f(t_{i-1}) + f(t_i)\right]}_{\text{composite}} \underbrace{- \sum_{i=1}^{r} \frac{f''(\eta_i)}{12}h^3}_{\text{error}}$$

Assuming $f''(x)$ continuous on $[a, b]$,

$$\min_{x\in[a,b]} f''(x) \leq f''(\eta_i) \leq \max_{x\in[a,b]} f''(x)$$

Then

$$\min_{x\in[a,b]} f''(x) \leq \frac{1}{r}\sum_{i=1}^{r} f''(\eta_i) \leq \max_{x\in[a,b]} f''(x)$$

Error of composite trapezoidal rule cont.

From the Intermediate Value Theorem, there exists μ, such that

$$f''(\mu) = \frac{1}{r}\sum_{i=1}^{r} f''(\eta_i)$$

Then the error is

$$\begin{aligned} -\sum_{i=1}^{r} \frac{f''(\eta_i)}{12} h^3 &= -\frac{1}{12}\left[\frac{1}{r}\sum_{i=1}^{r} f''(\eta_i)\right] r \cdot h \cdot h^2 \\ &= -\frac{f''(\mu)}{12}(b-a)h^2, \end{aligned}$$

$h = (b-a)/r$, and $r \cdot h = b - a$

Composite Simpson & midpoint rules

Simpson:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 2\sum_{i=1}^{r/2-1} f(t_{2i}) + 4\sum_{i=1}^{r/2} f(t_{2i-1}) + f(b)\right]$$

Error

$$-\frac{f^{(4)}(\zeta)}{180}(b-a)h^4$$

Midpoint:

$$\int_a^b f(x)dx \approx h\sum_{i=1}^{r} f\left(a + (i-1/2)h\right)$$

Error

$$\frac{f''(\xi)}{24}(b-a)h^2$$

Linear Least Squares

CS/SE 4X03

Ned Nedialkov

McMaster University

November 6, 2023

Outline

Formulation

In linear least squares, we have $n+1$ basis functions and $m+1$ data points (x_k, y_k), $k = 1, \ldots, m$, where $m > n$

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x), \quad v(x_k) \approx y_k, \qquad k = 0, \ldots, m$$

Find the c_j such that the sum

$$\sum_{k=0}^{m} (v(x_k) - y_k)^2 = \sum_{k=0}^{m} \left(\sum_{j=0}^{n} c_j \phi_j(x_k) - y_k \right)^2$$

is minimized

Least squares vs. interpolation

In interpolation, given $(n+1)$ data points (x_k, y_k), we find a function $v(x)$ such that

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x), \quad v(x_k) = y_k, \qquad k = 0, \ldots, n$$

In real-life applications, the data points may not be accurate, e.g. may come from measurements

May not make sense to interpolate inaccurate data

With least squares, may want to pick up a trend in the data, e.g. average temperature over last 10 years, is it warming or cooling down?

Linear fit

Suppose we search for a linear fit: $y = ax + b$, i.e. find a and b

Error or residual

$$r_k = ax_k + b - y_k$$

Find a and b such that

$$\phi(a, b) = \sum_{k=0}^{m} r_k^2 = \sum_{k=0}^{m} (ax_k + b - y_k)^2$$

is minimized

Necessary conditions for minimum:

$$\frac{\partial \phi}{\partial a} = 0, \quad \frac{\partial \phi}{\partial b} = 0$$

$$
\begin{aligned}
0 &= \frac{\partial \phi}{\partial a} = 2\sum_{k=0}^{m}(ax_k + b - y_k)x_k \\
0 &= a\sum_{k=0}^{m} x_k^2 + b\sum_{k=0}^{m} x_k - \sum_{k=0}^{m} y_k x_k
\end{aligned}
$$

from which

$$
\left(\sum_{k=0}^{m} x_k^2\right) a + \left(\sum_{k=0}^{m} x_k\right) b = \sum_{k=0}^{m} x_k y_k \tag{1}
$$

$$0 = \frac{\partial \phi}{\partial b} = 2\sum_{k=0}^{m}(ax_k + b - y_k)$$

$$0 = a\sum_{k=0}^{m} x_k + b\sum_{k=0}^{m} 1 - \sum_{k=0}^{m} y_k$$

from which

$$\left(\sum_{k=0}^{m} x_k\right) a + (m+1)b = \sum_{k=0}^{m} y_k \tag{2}$$

From (1) and (2) we have the linear system

$$\begin{bmatrix} \sum_{k=0}^{m} x_k^2 & \sum_{k=0}^{m} x_k \\ \sum_{k=0}^{m} x_k & m+1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{m} x_k y_k \\ \sum_{k=0}^{m} y_k \end{bmatrix}$$

Denote

$$p = \sum_{k=0}^{m} x_k, \qquad q = \sum_{k=0}^{m} y_k$$

$$r = \sum_{k=0}^{m} x_k y_k, \qquad s = \sum_{k=0}^{m} x_k^2$$

Then the system is

$$\begin{bmatrix} s & p \\ p & m+1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ q \end{bmatrix}$$

Solve for a and b

This system can be also obtained as follows.

Write $ax_k + b = y_k$, $k = 1, \dots, m$ as

$$Az = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{bmatrix} = f$$

Multiply both sided by A^T, $A^T Az = A^T f$

$$A^T A = \begin{bmatrix} x_0 & x_1 & \cdots & x_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^m x_k^2 & \sum_{k=0}^m x_k \\ \sum_{k=0}^m x_k & m+1 \end{bmatrix}$$

$$= \begin{bmatrix} s & p \\ p & m+1 \end{bmatrix}$$

$$A^T f = \begin{bmatrix} x_0 & x_1 & \cdots & x_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^m x_k y_k \\ \sum_{k=0}^m y_k \end{bmatrix}$$

$$= \begin{bmatrix} r \\ q \end{bmatrix}$$

$Az = f$ is overdetermined, more equations than unknowns

In MATLAB, find z by `A\f`

Example

- Assume a program runs in αn^β, where α and β are real constants we don't know
- How to determine them?
- Run the program with sizes $n_1, n_2, \ldots, n_m$ and measure the corresponding CPU times $t_1, t_2, \ldots, t_m$, $m > 2$
- Write $\alpha n_i^\beta = t_i$, $i = 1, \ldots, m$
- Then

$$\ln \alpha + \beta \ln n_i = \ln t_i, \quad i = 1, \ldots, m$$

- Let $x = \ln \alpha$
- Then

$$1 \cdot x + \ln n_i \cdot \beta = \ln t_i, \quad i = 1, \ldots, m$$

Write

$$
\begin{aligned}
1 \cdot x + \ln n_1 \cdot \beta &= \ln t_1 \\
1 \cdot x + \ln n_2 \cdot \beta &= \ln t_2 \\
&\vdots \\
1 \cdot x + \ln n_m \cdot \beta &= \ln t_m
\end{aligned}
$$

Then

$$
Ay = \begin{bmatrix} 1 & \ln n_1 \\ 1 & \ln n_2 \\ \vdots & \vdots \\ 1 & \ln n_m \end{bmatrix} \begin{bmatrix} x \\ \beta \end{bmatrix} = \begin{bmatrix} \ln t_1 \\ \ln t_2 \\ \vdots \\ \ln t_m \end{bmatrix} = b
$$

Solve in Matlab as `y = A\b`; $\alpha =$ `exp(y(1))` $\beta =$ `y(2)`

Solving overdetermined systems

- $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
  $m > n$
- $Ax = b$ is an overdetermined system: more equations than variables
- Find x that minimizes $\|b - Ax\|_2$
- $r = b - Ax$
- $\|r\|_2^2 = \sum_{i=1}^m r_i^2 = \sum_{i=1}^m \left(b_i - \sum_{j=1}^n a_{ij}x_j\right)^2$
- Let

$$\phi(x) = \frac{1}{2}\|r\|_2^2 = \frac{1}{2}\sum_{i=1}^m \left(b_i - \sum_{j=1}^n a_{ij}x_j\right)^2$$

- We want to find the minimum of $\phi(x)$
- Necessary conditions are

$$\frac{\partial \phi}{\partial x_k} = 0, \quad \text{for } k = 1, \ldots, n$$

$$\begin{aligned}
0 = \frac{\partial \phi}{\partial x_k} &= \frac{\partial}{\partial x_k} \left(\frac{1}{2} \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)^2 \right) \\
&= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial x_k} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)^2 \\
&= \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) (-a_{ik})
\end{aligned}$$

$$
\begin{aligned}
0 &= \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) (-a_{ik}) \\
&= -\sum_{i=1}^{m} a_{ik} b_i + \sum_{i=1}^{m} a_{ik} \sum_{j=1}^{n} a_{ij} x_j
\end{aligned}
$$

We have

$$
\sum_{i=1}^{m} a_{ik} \sum_{j=1}^{n} a_{ij} x_j = \sum_{i=1}^{m} a_{ik} b_i, \quad k = 1, \ldots, n
$$

This is the same as $A^T A x = A^T b$, as explained below

A is $m \times n$. A^T is $n \times m$.

Let $y = Ax$. $y_i = \sum_{j=1}^{n} a_{ij} x_j$, $i = 1, \ldots, m$.

The kth component of $A^T A x = A^T y$ is

$$(A^T A x)_k = (A^T y)_k = \sum_{i=1}^{m} (A^T)_{ki} y_i = \sum_{i=1}^{m} a_{ik} y_i = \sum_{i=1}^{m} a_{ik} \sum_{j=1}^{n} a_{ij} x_j$$

The kth component of $A^T b$ is

$$(A^T b)_k = \sum_{i=1}^{m} (A^T)_{ki} b_i = \sum_{i=1}^{m} a_{ik} b_i$$

$(A^T A x)_k = (A^T b)_k$, $k = 1, \ldots, n$ is

$$\sum_{i=1}^{m} a_{ik} \sum_{j=1}^{n} a_{ij} x_j = \sum_{i=1}^{m} a_{ik} b_i$$

Normal equations

- $A^T Ax = A^T b$ are called *normal equations*
- If A has a full-column rank (all columns are linearly independent),

$$\min_x \|b - Ax\|_2$$

has a unique solution which is the solution to $(A^T A)x = A^T b$:

$$x = (A^T A)^{-1} A^T b = A^\dagger b$$

- $A^\dagger = (A^T A)^{-1} A^T$ is the *pseudo inverse* of A

Adaptive Simpson

CS/SE 4X03

Ned Nedialkov

McMaster University

October 31, 2023

Outline

Derivation of Simpson's rule

Simpson's rule can be derived using the method of undetermined coefficients

- Seek integration formula of the form

$$\int_a^b f(x)dx \approx Af(a) + Bf\left(\frac{a+b}{2}\right) + Cf(b)$$

- Find A, B, C such that for quadratic polynomials the formula is exact:

$$\int_a^b f(x)dx = Af(a) + Bf\left(\frac{a+b}{2}\right) + Cf(b)$$

Derivation of Simpson's rule cont.

- Let $a = -1$, $b = 1$. We should integrate exactly 1, x, x^2:

$$
\begin{aligned}
f(x) = 1: &\quad \int_{-1}^{1} dx = 2 = A + B + C \\
f(x) = x: &\quad \int_{-1}^{1} x dx = 0 = -A + C \\
f(x) = x^2: &\quad \int_{-1}^{1} x^2 dx = \frac{2}{3} = A + C
\end{aligned}
$$

  from which $A = 1/3$, $C = 1/3$, $B = 4/3$
- Hence

$$
\int_{-1}^{1} f(x)dx \approx \frac{1}{3}[f(-1) + 4f(0) + f(1)]
$$

Derivation of Simpson's rule cont.

- Let $y(x) = 0.5(b-a)x + 0.5(b+a)$, $y(-1) = a$, $y(1) = b$
- Changing variables:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Adaptive Simpson

- Given a function $f(x)$ on $[a, b]$ and tolerance tol
- find Q such that

$$|Q - I| \leq \text{tol},$$

where

$$I = \int_a^b f(x)dx$$

Adaptive Simpson cont.

Denote $h = b - a$. Then

$$I = \int_a^b f(x)dx = S(a,b) + E(a,b),$$

where

$$S(a,b) = \frac{h}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

$$E(a,b) = -\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi), \quad \xi \text{ between } a \text{ and } b$$

Denote $S_1 = S(a,b)$ and $E_1 = E(a,b)$

Adaptive Simpson cont.

- Let $c = (a+b)/2$ and apply Simpson on $[a,c]$ and $[c,b]$:

$$I = \int_a^b f(x)dx = \underbrace{S(a,c) + S(c,b)}_{S_2} + \underbrace{E(a,c) + E(c,b)}_{E_2}$$

- We can compute S_1 and S_2
- How to estimate the error? If $f^{(4)}$ does not change much on $[a,b]$

$$\begin{aligned} E(a,c) &= -\frac{1}{90}\left(\frac{h/2}{2}\right)^5 f^{(4)}(\xi_1) = \frac{1}{32}\left[-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi_1)\right], \quad \xi_1 \in [a,c] \\ &\approx \frac{1}{32}\left[-\frac{1}{90}\left(\frac{h}{2}\right)^5 f^{(4)}(\xi)\right] \\ &= \frac{1}{32}E_1 \end{aligned}$$

Adaptive Simpson cont.

Similarly $E(c,b) \approx \frac{1}{32}E_1$

- Hence

$$E_2 = E(a,c) + E(c,b) \approx \frac{1}{16}E_1$$

- From $I = S_1 + E_1 = S_2 + E_2$,

$$S_1 - S_2 = E_2 - E_1 \approx E_2 - 16E_2 = -15E_2$$

$$E_2 \approx \widetilde{E}_2 = \frac{1}{15}(S_2 - S_1)$$

- Then

$$I = \int_a^b f(x)dx = S_2 + E_2 \approx S_2 + \widetilde{E}_2$$

Method outline

Given f, $[a, b]$ and tol:

- $c = (a + b)/2$
- Compute $S_1 = S(a, b)$ and $S_2 = S(a, c) + S(c, b)$
- $\widetilde{E}_2 = (S_2 - S_1)/15$
- If $|\widetilde{E}_2| \leq \text{tol}$ return $S_2 + \widetilde{E}_2$
  else apply recursively on $[a, c]$ and $[c, b]$ with $\text{tol}/2$

Adaptive Simpson cont.

Algorithm 2.1 (Adaptive Simpson).

S = quadSimpson(f, a, b, tol)

> $h = b - a$, $c = (a + b)/2$
>
> $S_1 = \frac{h}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$
>
> $S_2 = \frac{h}{12}[f(a) + 4f(\frac{a+c}{2}) + 2f(c) + 4f(\frac{c+b}{2}) + f(b)]$
>
> $\widetilde{E}_2 = \frac{1}{15}(S_2 - S_1)$
>
> if $|\widetilde{E}_2| \leq$ tol
>
> > return $Q = S_2 + \widetilde{E}_2$
>
> else
>
> > Q_1 = quadSimpson(f, a, c, tol/2)
> > Q_2 = quadSimpson(f, c, b, tol/2)
> > return $Q = Q_1 + Q_2$

Why it works

- If $|E_2| \approx |\widetilde{E}_2| \leq \text{tol}$, we can return $Q = S_2$. Then

$$|I - Q| = |I - S_2| = |E_2| \approx |\widetilde{E}_2| \leq \text{tol}.$$

- However, but adding the error estimate, we can obtain a more accurate approximation as

$$I = S_2 + E_2 \approx Q = S_2 + \widetilde{E}_2.$$

- Otherwise, let $I_1 = \int_a^c f(x)dx$, $I_2 = \int_b^c f(x)dx$
  If
  $$|I_1 - Q_1| \leq \text{ tol}/2 \quad \text{and} \quad |I_2 - Q_2| \leq \text{ tol}/2,$$
  then
  $$\begin{aligned} |I - Q| &= |I_1 + I_2 - (Q_1 + Q_2)| \\ &= |I_1 - Q_1 + I_2 - Q_2| \\ &\leq |I_1 - Q_1| + |I_2 - Q_2| \\ &\leq \text{ tol}/2 + \text{ tol}/2 \\ &= \text{ tol} \end{aligned}$$

Subtleties

- The error estimate assumes $f^{(4)}$ does not vary much, but it may, and then this estimate may not be accurate.
  That is, $\widetilde{E}_2$ may not be a good approximation to E_2.
- The recursion may run "deep" if tol is too small or $f^{(4)}$ varies a lot
  Insert a counter to stop the recursion when the depth exceeds some number, e.g. 20

Introduction to Machine Learning

CS/SE 4X03

Ned Nedialkov

McMaster University

November 13, 2023

Outline

Example

Activation function

A simple network

Training

Steepest descent

Stochastic gradient descent

This is a summary of Sections 1-4 from
C. F. Higham, D. J. Higham, Deep Learning: An Introduction for Applied Mathematicians

Figures are cropped from this article

Example

- Points in $\mathbb{R}^2$ classified in two categories A and B
- This is labeled data

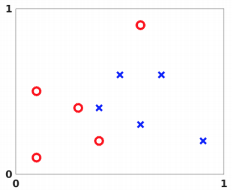
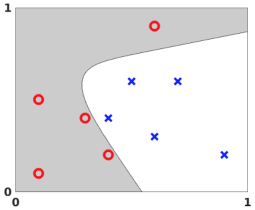


Figure 1: Labeled data points in $\mathbb{R}^2$. Circles denote points in category A. Crosses denote points in category B.

- Given a new point, how to use the labeled data to classify this point?
  Possible classification

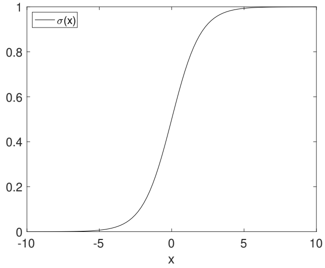


Activation function

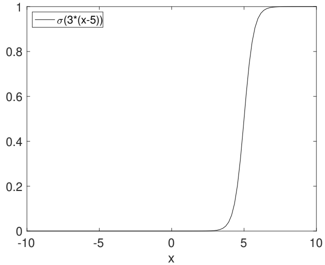
- A neuron fires or is inactive
- Activation can be modeled by the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- $\sigma(0) = 0.5$, $\sigma(x) \approx 1$ when x large, $\sigma(x) \approx 0$ when x small



- Steepness can be changed by scaling
- Location can be changed by shifting
- Useful property $\sigma'(x) = \sigma(x)(1 - \sigma(x))$



A simple network

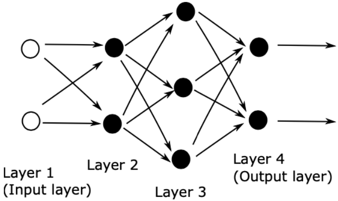


Figure 3: A network with four layers.

- Each neuron
  - outputs a real number
  - sends to every neuron in next layer
- Neuron in next layer
  - forms a linear combination of inputs + bias
  - applies activation function

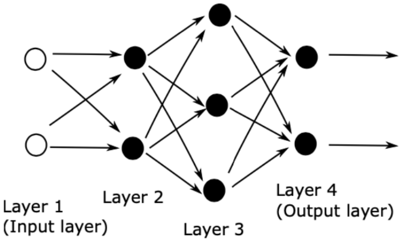
Consider layers 2 and 3

Layer 2: neurons 1 and 2 output real a_1 and a_2, respectively, and send to neurons $1, 2, 3$ in layer 3
Layer 3:

- neuron
  1 combines a_1 and a_2 and ads bias b_1:

  $$w_{11}a_1 + w_{12}a_2 + b_1$$

  outputs
  $\sigma\left(w_{11}a_1 + w_{12}a_2 + b_1\right)$
- neuron 2 outputs
  $\sigma\left(w_{21}a_1 + w_{22}a_2 + b_2\right)$
- neuron 3 outputs
  $\sigma\left(w_{31}a_1 + w_{32}a_2 + b_3\right)$



Denote

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}, \qquad a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$z = Wa + b$$

W is a matrix with weights, b is a bias vector

For a vector z, apply σ component wise

$$(\sigma(z))_i = \sigma(z_i)$$

The output of layer 3 is

$$\sigma(z) = \sigma(Wa + b)$$

- Denote the input by x, the W and b at layer i by $W^{[i]}$ and $b^{[i]}$, and the output of layer i by $a^{[i]}$
- Output of layer 2 is

$$a^{[2]} = \sigma\left(W^{[2]}x + b^{[2]}\right) \in \mathbb{R}^2, \qquad W^{[2]} \in \mathbb{R}^{2\times 2},\ b^{[2]} \in \mathbb{R}^2$$

- Output of layer 3 is

$$a^{[3]} = \sigma\left(W^{[3]}a^{[2]} + b^{[3]}\right) \in \mathbb{R}^3, \qquad W^{[3]} \in \mathbb{R}^{3\times 2},\ b^{[3]} \in \mathbb{R}^3$$

- Output of layer 4 is

$$a^{[4]} = \sigma\left(W^{[4]}a^{[3]} + b^{[4]}\right) \in \mathbb{R}^2, \qquad W^{[4]} \in \mathbb{R}^{2\times 3},\ b^{[4]} \in \mathbb{R}^2$$

- Write the above as

$$F(x) = \sigma\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) + b^{[4]}\right)$$

- Layer i:
  - $W^{[i]}$ is of size (# outputs)$\times$(# inputs)
  - $b^{[i]}$ is of size (# outputs)

  Number of parameters is 23:

layer i	inputs	outputs	$W^{[i]}$	$b^{[i]}$
2	2	2	2×2	2
3	2	3	3×2	3
4	3	2	2×3	2
			16	7

- $F(x)$ is a function from $\mathbb{R}^2 \to \mathbb{R}^2$ with 23 parameters
- Training is about finding parameters

Training

Residual

- Denote the input points by $x^{\{i\}}$
- Let

$$y\left(x^{\{i\}}\right)=\begin{cases}\begin{bmatrix}1\\0\end{bmatrix} & \text{if } x^{\{i\}}\in A\\ \begin{bmatrix}0\\1\end{bmatrix} & \text{if } x^{\{i\}}\in B\end{cases}$$

- Suppose we have computed $W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}$ and evaluate $F\left(x^{\{i\}}\right)$
- Residual

$$\left\|y\left(x^{\{i\}}\right)-F\left(x^{\{i\}}\right)\right\|_2$$

Training

Cost function

- Cost function

$$\text{Cost}\left(W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}\right)$$
$$= \frac{1}{10} \sum_{i=1}^{10} \frac{1}{2} \left\| y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right) \right\|_2^2$$

- Training: find the parameters that minimize the cost function
- Nonlinear least squares problem

Classifying

- Suppose we have computed values for the parameters
- Given $x \in \mathbb{R}^2$, compute $y = F(x)$
- If $y_1 > y_2$ classify x as A, y closer to $[1, 0]^T$
- If $y_1 < y_2$ classify x as B, y closer to $[0, 1]^T$
- Tie breaking when $=$

Steepest descent

- Consider the parameters in a vector $p \in \mathbb{R}^s$. Here $s = 23$
- Cost function is $\text{Cost}(p)$
- Find Δp such that

$$\text{Cost}(p + \Delta p) < \text{Cost}(p)$$

- For small Δp,

$$\begin{aligned}\text{Cost}(p + \Delta p) &\approx \text{Cost}(p) + \sum_{r=1}^{s} \frac{\partial \text{Cost}(p)}{\partial p_r} \Delta p_r \\ &= \text{Cost}(p) + \nabla \text{Cost}(p)^T \Delta p \\ \nabla \text{Cost}(p) &= \left[\frac{\partial \text{Cost}(p)}{\partial p_1}, \frac{\partial \text{Cost}(p)}{\partial p_2}, \cdots, \frac{\partial \text{Cost}(p)}{\partial p_s} \right]^T\end{aligned}$$

Example

- To illustrate the above, suppose

$$\mathsf{Cost}(p) = p_1^2 + p_2^2 + 2p_1 + 3$$

- Gradient is

$$\nabla\mathsf{Cost}(p) = [2p_1 + 2, 2p_2]^T$$

$$\begin{aligned}\mathsf{Cost}(p + \Delta p) &\approx \mathsf{Cost}(p) + \nabla\mathsf{Cost}(p)^T\Delta p \\ &= \mathsf{Cost}(p) + (2p_1 + 2)\Delta p_1 + 2p_2\Delta p_2\end{aligned}$$

Steepest descent cont

- $\text{Cost}(p) \geq 0$
- From
$$\text{Cost}(p + \Delta p) \approx \text{Cost}(p) + \nabla \text{Cost}(p)^T \Delta p,$$
we want to make $\nabla \text{Cost}(p)^T \Delta p$ as negative as possible
- Given $\nabla \text{Cost}(p)$ how to choose Δp?
- For $u, v \in \mathbb{R}^s$,
$$u^T v = \|u\| \cdot \|v\| \cos \theta$$
is most negative when $v = -u$
- Chose Δp in the direction of $-\nabla \text{Cost}(p)$
That is move along the direction of steepest descent

$$\Delta p = p_{\mathsf{new}} - p = -\eta \nabla \mathsf{Cost}(p)$$
$$p_{\mathsf{new}} = p - \eta \nabla \mathsf{Cost}(p)$$

η is learning rate

Steepest descent:

chose initial p
repeat
    $p \leftarrow p - \eta \nabla \mathsf{Cost}(p)$
until stopping criterion is met or max # of iterations is reached

- In general N input points

$$\text{Cost}(p) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{1}{2}\left\| y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right)\right\|_2^2}_{C_i(p)}$$

$$= \frac{1}{N}\sum_{i=1}^{N} C_i(p)$$

$$\nabla\text{Cost}(p) = \frac{1}{N}\sum_{i=1}^{N} \nabla C_i(p)$$

- N can be large
- Number of parameters can be very large
- Evaluating $\nabla\text{Cost}(p)$ can be very expensive

Stochastic gradient descent

- Idea: replace $\frac{1}{N}\sum_{i=1}^{N} \nabla C_i(p)$ by random $\nabla C_i(p)$
- Iterate until a stopping criterion is met or max # of iterations is reached:
  - pick a random integer i from $\{1, 2, \ldots, N\}$
  - $p \leftarrow p - \eta \nabla C_i(p)$

Newton's Method for Nonlinear Equations

CS/SE 4X03

Ned Nedialkov

McMaster University

November 20, 2023

Outline

Scalar case

- Given a scalar function f find a zero/root of f, i.e. an r such that $f(r) = 0$
- f may have no zeros, one, or many
- Let r be a root of f and let $x_n \approx r$
  From

$$
\begin{aligned}
0 &= f(r) = f(x_n) + f'(x_n)(r - x_n) + O(|r - x_n|^2) \\
0 &= f(r) \approx f(x_n) + f'(x_n)(r - x_n)
\end{aligned}
$$

we find x_{n+1} by solving

$$
f(x_n) + f'(x_n)(x_{n+1} - x_n) = 0 \tag{1}
$$

Scalar case cont.

- That is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

- We start with an initial guess x_0 and compute $x_1, x_2, \dots$
- How to choose x_0, does it converge to a root, when to stop iterating...?

Interpretation

Given x_0, we compute

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

The tangent line at $(x, f(x_0))$ is

$$l(x) = f(x_0) + f'(x_0)(x - x_0)$$

We find x_1 such that $l(x)$ crosses the x axis, $l(x_1) = 0$:

$$0 = l(x_1) = f(x_0) + f'(x_0)(x_1 - x_0)$$

Similarly for x_2, x_3, ...

Examples

Square root

- Given $a > 0$, compute $\sqrt{a}$
- Write $x = \sqrt{a}$, $f(x) = x^2 - a$
- Apply (2):

$$
\begin{aligned}
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} \\
&= x_n - \frac{x_n}{2} + \frac{a}{2x_n} \\
&= 0.5\left(x_n + \frac{a}{x_n}\right)
\end{aligned}
$$

- Let $a = 2$ and $x_0 = 3$
- We compute

i	x_i	$\lvert x_i - \sqrt{2} \rvert$
1	1.833333333333333	4.19e-01
2	1.462121212121212	4.79e-02
3	1.414998429894803	7.85e-04
4	1.414213780047198	2.18e-07
5	1.414213562373112	1.67e-14
6	1.414213562373095	2.22e-16

Examples cont.

Dividing without division operation

- How to obtain a/b without division?
- $a/b = a * (1/b)$
- Find $1/b$. Write $f(x) = 1/x - b$ and apply (2)

$$
\begin{aligned}
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{1/x_n - b}{-1/x_n^2} \\
&= x_n + x_n - b x_n^2 \\
&= x_n(2 - b x_n)
\end{aligned}
$$

Examples cont.

- With $b = 3$ and $x_0 = 0.3$, we compute

i	x_i	$\lvert x_i - 1/3\rvert$
1	0.330000000000000	3.33e-03
2	0.333300000000000	3.33e-05
3	0.333333330000000	3.33e-09
4	0.333333333333333	5.55e-17

Convergence

Theorem 1. If f, f', and f'' are continuous in a neighbourhood of a root r of f and $f'(r) \neq 0$, then $\exists \delta > 0$ such that if $|r - x_0| \leq \delta$, then all x_n satisfy

$$|r - x_n| \leq \delta, \tag{3}$$

$$|r - x_{n+1}| \leq c(\delta)|r - x_n|^2, \tag{4}$$

where $c(\delta)$ is defined in (6), and x_n converges to r

Let $e_n = r - x_n$. (4) is

$$|e_{n+1}| \leq c(\delta)|e_n|^2 \tag{5}$$

If e.g. $|e_n| \approx 10^{-4}$, $|e_{n+1}| \lessapprox c(\delta)10^{-8}$

If sufficiently close to r, each iteration $\approx$ doubles the number of accurate digits

Quadratic convergence $|e_{n+1}| \leq \text{constant} \cdot |e_n|^2$

Order of convergence is 2

Convergence cont.

Proof. From Taylor series,

$$
\begin{aligned}
0 = f(r) &= f(x_n) + f'(x_n)(r - x_n) + \frac{f''(\xi)}{2}(r - x_n)^2 \\
&= f(x_n) + f'(x_n)e_n + \frac{f''(\xi)}{2}e_n^2
\end{aligned}
$$

$$
f(x_n) + f'(x_n)e_n = -\frac{f''(\xi)}{2}e_n^2, \quad \xi \text{ is between } r \text{ and } x_n
$$

The error in x_{n+1} is

$$
\begin{aligned}
e_{n+1} = r - x_{n+1} = r - \left(x_n - \frac{f(x_n)}{f'(x_n)}\right) &= r - x_n + \frac{f(x_n)}{f'(x_n)} \\
&= e_n + \frac{f(x_n)}{f'(x_n)} = \frac{f(x_n) + e_n f'(x_n)}{f'(x_n)} \\
&= -\frac{1}{2}\frac{f''(\xi)}{f'(x_n)}e_n^2
\end{aligned}
$$

Convergence cont.

For a $\delta > 0$, let

$$c(\delta) = \frac{1}{2} \frac{\max_{|r-x|\leq\delta} |f''(x)|}{\min_{|r-x|\leq\delta} |f'(x)|} \tag{6}$$

Then (4) follows from

$$\begin{aligned} |e_{n+1}| &= \frac{1}{2} \frac{|f''(\xi)|}{|f'(x_n)|} e_n^2 \leq \frac{1}{2} \frac{\max_{|r-x|\leq\delta} |f''(x)|}{\min_{|r-x|\leq\delta} |f'(x)|} e_n^2 \\ &\leq c(\delta) e_n^2 \end{aligned}$$

There exists δ such that $c(\delta)\delta < 1$ since

$$c(\delta) \to \frac{1}{2} \left| \frac{f''(r)}{f'(r)} \right| \quad \text{as } \delta \to 0$$

and $f'(r) \neq 0$ by assumption

Convergence cont.

If $|e_n| = |r - x_n| \le \delta$, then

$$\begin{aligned} |e_{n+1}| &\le c(\delta)e_n^2 = c(\delta) \cdot e_n \cdot e_n \le c(\delta)\delta \cdot e_n \\ &< \rho e_n, \quad \text{where } \rho = \delta c(\delta) < 1 \end{aligned}$$

and (3) follows

Hence

$$|e_n| \le \rho |e_{n-1}| \le \rho^2 |e_{n-2}| \le \cdots \le \rho^n |e_0|$$

Since $\rho < 1$, $|e_n| \to r$ as $n \to \infty$

Subtleties

We require $f'(r) \neq 0$

If $f'(r) = 0$ and $f''(r) \neq 0$, r is a double root, e.g. $f(x) = (x-1)^2$

A root r is of multiplicity m if $f^{(k)}(r) = 0$ for all $k = 1, 2, \dots m-1$ and $f^{(m)}(r) \neq 0$. In this case

$$x_{n+1} = x_n - m\frac{f(x_n)}{f'(x_n)}$$

is quadratically convergent

If $f'(x_n)$ is not available, we can approximate $f'(x_n) \approx \frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$
Then

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

This is the secant method. Order of convergence is $(1+\sqrt{5})/2 \approx 1.618$ (golden ratio)

Newton for systems of equations

- Consider a system of n equations in n variables

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

- Denote $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ and $F = (f_1, f_2, \ldots, f_n)$
- Find $\mathbf{x}^*$ (if it exists) such that $F(\mathbf{x}^*) = 0$

Newton for systems of equations cont.

- Assume $\mathbf{x}^*$ is such that $F(\mathbf{x}^*) = 0$ and $\mathbf{x}^{(k)} \approx \mathbf{x}^*$
- From

$$0 = F(\mathbf{x}^*) \approx F(\mathbf{x}^{(k)}) + F'(\mathbf{x}^{(k)})(\mathbf{x}^* - \mathbf{x}^{(k)})$$

  find $\mathbf{x}^{(k+1)}$ by solving (cf. (1))

$$F(\mathbf{x}^{(k)}) + F'(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = 0 \tag{7}$$

- $F'(\mathbf{x}^{(k)})$ is the Jacobian of F at $\mathbf{x}^{(k)}$, an $n \times n$ matrix

Newton for systems of equations cont.

- Let $s = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$
- Solve (assuming $F'(\mathbf{x}^{(k)})$ nonsingular) linear system

$$F'(\mathbf{x}^{(k)})s = -F(\mathbf{x}^{(k)}) \tag{8}$$

and set

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + s \tag{9}$$

- (8,9) is basic Newton for systems of equations

Example

- Consider

$$0 = F(\mathbf{x}) = \begin{cases} x_1^2 + x_2^2 - 25 \\ x_1^2 - x_2 - 1 \end{cases}$$

- Jacobian is

$$F'(\mathbf{x}) = \begin{pmatrix} 2x_1 & 2x_2 \\ 2x_1 & -1 \end{pmatrix}$$

- Let $x_0 = (5, 1)^T$

- Then

$$F(\mathbf{x}^{(0)}) = (1, 23)^T$$
$$J(\mathbf{x}^{(0)}) = \begin{pmatrix} 10 & 2 \\ 10 & -1 \end{pmatrix}$$

- Solve $J(\mathbf{x}^{(0)})s = -F(\mathbf{x}^{(0)})$
- $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + s$ and so on
- We compute

i	x_1	x_2	$\|F(\mathbf{x})\|$
1	3.433333333333334	8.333333333333332	5.63e+01
2	2.632585333089088	5.289308176100628	9.93e+00
3	2.358810087435537	4.489032143454986	7.19e-01
4	2.329316858408983	4.424847176309882	5.06e-03
5	2.329040359270796	4.424428918660463	2.63e-07
6	2.329040339044829	4.424428900898053	7.11e-15

Numerical Methods for IVP ODEs

CS/SE 4X03

Ned Nedialkov

McMaster University

November 28, 2023

Outline

The problem

- Given
$$y' = f(t, y), \qquad y(a) = c$$
compute $y(t)$ on $[a, b]$
- $y' \equiv y'(t) \equiv \frac{dy}{dt}$
- This is an Initial Value Problem (IVP) in Ordinary Differential Equations (ODEs)
- We approximate $y(t)$ at points t_i in $[a, b]$ using a numerical method

ODE examples

$$y' = -y + t$$

- Solution is $y(t) = t - 1 + \alpha e^{-t}$:

$$\begin{aligned} y'(t) &= 1 - \alpha e^{-t} \\ -y + t &= -(t - 1 + \alpha e^{-t}) + t = 1 - \alpha e^{-t} \end{aligned}$$

- Given $y(0) = c$, e.g. $c = 5$,

$$y(0) = -1 + \alpha = c = 5, \qquad \alpha = 6$$

$$y(t) = t - 1 + 6e^{-t}$$

is the solution with this initial condition

Motion of a pendulum

$$\theta'' = -g \sin \theta, \qquad \theta'' = \frac{d^2\theta(t)}{dt^2}$$

- ball of mass 1 attached to the end of a rigid, massless rod of length $r = 1$
- $g \approx 9.81$ is gravity
- t is time
- This is a second-order ODE. To write as a first-order ODE, set $y_1 = \theta$, $y_2 = \theta' = y_1'$:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -g \sin(y_1) \end{aligned}$$

- Needed initial conditions are $y_1(0)$ and $y_2(0)$

ODEs

System of n first-order equations in n variables

$$y' = f(t, y), \qquad f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$$

Nonlinear: if f is nonlinear in y, linear otherwise

Autonomous ODE

$$y' = f(y), \qquad y(a) = c$$

is an autonomous ODE, does not depend on time explicitly

$$y' = f(t, y), \qquad y(a) = c$$

is non-autonomous

To convert a non-autonomous ODE to an autonomous set $x = t$ and then

$$\begin{aligned} x' &= 1 \\ y' &= f(x, y), \qquad x(a) = a,\ y(a) = c \end{aligned}$$

Set $z = (z_1, z_2)^T = (x, y)^T$. Then $z' = f(z)$:

$$\begin{aligned} z_1' &= 1 \\ z_2' &= f(z) \end{aligned}$$

High-order ODEs

$$y^{(n)} = f\left(t, y, y', \ldots, y^{(n-1)}\right)$$

can be converted to first-order by setting

$$\begin{aligned}
y_1 &= y \\
y_2 &= y' = y_1' \\
y_3 &= y'' = y_2' \\
&\vdots \\
y_n &= y^{(n-1)} = y_{n-1}'
\end{aligned}$$

Then

$$
\begin{aligned}
y_1' &= y_2 \\
y_2' &= y_3 \\
&\vdots \\
y_n' &= f(t, y_1, y_2, \ldots, y_n)
\end{aligned}
$$

To solve, we need initial values for

$$y_1(a), y_2(a), \ldots, y_n(a)$$

Euler's method

- Let $h = (b-a)/N$, $N > 1$ is an integer
- h is stepsize
- Let $t_0 = a$, $t_i = a + ih$, $i = 0, 1, \ldots, N$
- From $y'(t_i) = f(t_i, y(t_i))$, we write

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(\xi_i), \quad \xi_i \text{ between } t_i \text{ and} t_{i+1} \\ &= y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} y''(\xi_i) \\ &\approx y(t_i) + hf(t_i, y(t_i)) \end{aligned}$$

- Euler's method:

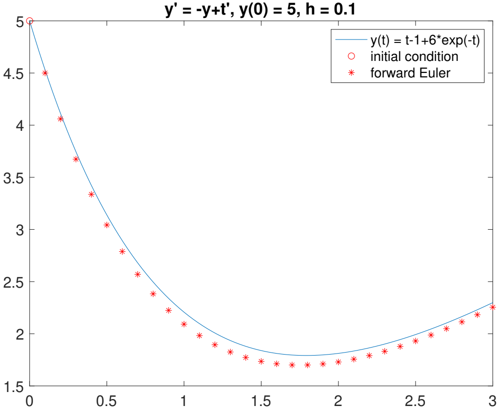
$$y_0 = c$$
$$y_{i+1} = y_i + hf(t_i, y_i), \qquad i = 0, 1, \ldots, N-1$$

- Example: Euler's method on $y' = -y + t$, $y(0) = y_0 = 5$, with $h = 0.1$:

$$\begin{aligned} y_{i+1} &= y_i + hf(t_i, y_i) = y_i + h(-y_i + t_i) \\ y_1 &= y_0 + h(-y_0 + t_0) = 5 + 0.1(-5 + 0) = 4.5 \\ y_2 &= y_1 + h(-y_1 + t_1) = 4.5 + 0.1(-4.5 + 0.1) = 4.06 \\ y_3 &= y_2 + h(-y_2 + t_2) = 4.06 + 0.1(-4.06 + 0.2) = 3.674 \end{aligned}$$

- Exact solution is $y(t) = t - 1 + 6e^{-t}$
- The corresponding exact values are $y(0.1) \approx 4.5290$, $y(0.2) \approx 4.1124$, $y(0.3) \approx 3.7449$

Example: Forward Euler on $y' = -y + t$



Backward Euler

- We can write

$$\begin{aligned} y(t_i) &= y(t_{i+1}) - hy'(t_{i+1}) + \frac{h^2}{2} y''(\eta_i) \\ &\approx y(t_{i+1}) - hf(t_{i+1}, y(t_{i+1})) \\ y(t_{i+1}) &\approx y(t_i) + hf(t_{i+1}, y(t_{i+1})) \end{aligned}$$

- Backward Euler

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1})$$

- This is an implicit method; forward Euler is explicit

- Example: Backward Euler method on $y' = -y + t$, $y(0) = y_0 = 5$, with $h = 0.1$:

$$\begin{aligned} y_{i+1} &= y_i + hf(t_{i+1}, y_{i+1}) \\ &= y_i + h(-y_{i+1} + t_{i+1}) \end{aligned}$$

- We need to solve for y_{i+1}:

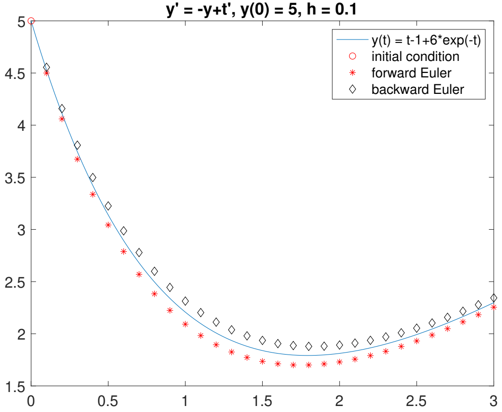
$$\begin{aligned} y_{i+1} &= y_i - hy_{i+1} + ht_{i+1} \\ y_{i+1} + hy_{i+1} &= y_i + ht_{i+1} \\ y_{i+1} &= \frac{y_i + ht_{i+1}}{1 + h} \end{aligned}$$

- We compute

$$
\begin{aligned}
y_1 &= \frac{y_0 + ht_1}{1+h} = \frac{5 + 0.1 \cdot 0.1}{1 + 0.1} \approx 4.5545 \\
y_2 &= \frac{y_1 + ht_2}{1+h} \approx \frac{4.5545 + 0.1 \cdot 0.2}{1 + 0.1} \approx 4.1586 \\
y_3 &= \frac{y_2 + ht_3}{1+h} \approx \frac{4.1586 + 0.1 \cdot 0.3}{1 + 0.1} \approx 3.7987
\end{aligned}
$$

- The corresponding exact values are $y(0.1) \approx 4.5290$, $y(0.2) \approx 4.1124$, $y(0.3) \approx 3.7449$
- Here it was easy to solve for y_{i+1}: $f(t, y) = -y + t$ is linear in y
- In general, it is non-linear: apply Newton's method

Example: FE and BE on $y' = -y + t$



Stability

Forward Euler

- Consider $y' = \lambda y$, $y(0) = y_0$
- The exact solution is $y(t) = e^{\lambda t} y_0$
- Forward Euler with constant stepsize h is

$$
\begin{aligned}
y_{i+1} &= y_i + hf(t_i, y_i) = y_i + h\lambda y_i \\
&= (1 + h\lambda) y_i \\
&= (1 + h\lambda)^2 y_{i-1} \\
&\quad \vdots \\
&= (1 + h\lambda)^{i+1} y_0
\end{aligned}
$$

- If $\lambda < 0$, $y(t)$ is decaying. Since $|y(t_{i+1})| < |y(t_i)|$, we want $|y_{i+1}| \leq |y_i|$

Stability cont.

- For the method to be numerically stable, we require

$$|y_{i+1}| = |1 + h\lambda| \cdot |y_i| \leq |y_i|$$

- That is $|1 + h\lambda| \leq 1$, or

$$\begin{aligned} -1 \leq 1 + h\lambda &\leq 1 \\ -2 \leq h\lambda &\leq 0 \\ h &\leq \frac{2}{|\lambda|} \end{aligned}$$

- If $|\lambda|$ is large, we can have a severe restriction on the stepsize
If e.g. $y' = -10^6 y$, $h \leq 2 \cdot 10^{-6}$

Stability cont.

Example 1.

- Consider $y' = -10y$, $y(0) = y_0$
- Euler's method is

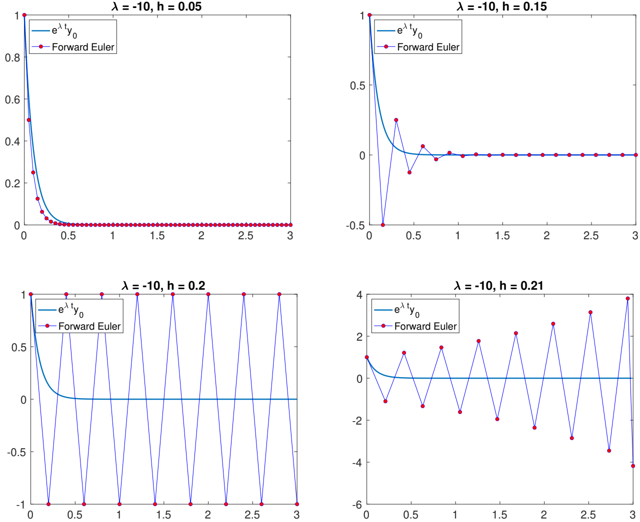
$$y_{i+1} = y_i + h\lambda y_i = (1 - 10h)y_i$$

- For stability $h \leq 0.2$
- If e.g. $h = 0.21$ then

$$\begin{aligned}
y_1 &= (1 - 10 \cdot 0.21)y_0 = -1.1y_0 \\
y_2 &= -1.1y_1 = 1.21y_0 \\
y_3 &= -1.1y_2 = -1.331y_0 \\
&\vdots \\
y_i &= (-1.1)^i y_0
\end{aligned}$$

Stability

Example 1. cont.



Stability

Backward Euler

- Consider the backward Euler on $y' = \lambda y$, where $\lambda < 0$

$$\begin{aligned} y_{i+1} &= y_i + h\lambda y_{i+1} \\ y_{i+1} &= \frac{1}{1 - h\lambda} y_i \\ |y_{i+1}| &= \frac{1}{|1 - h\lambda|} |y_i| \\ &\le |y_i| \qquad \text{for any } h > 0 \end{aligned}$$

Stability

Example 2.

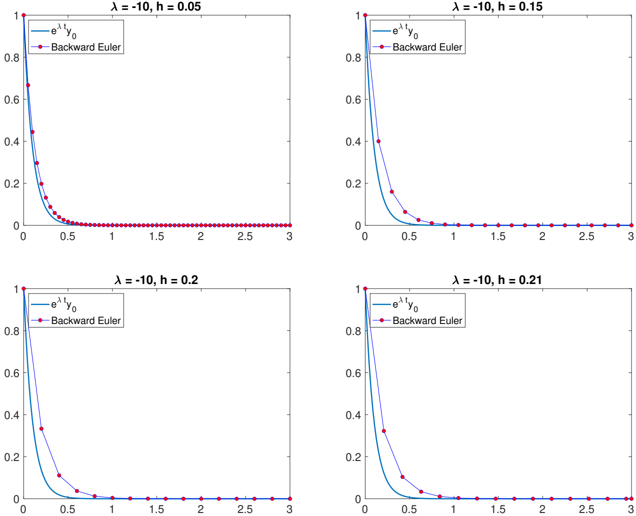
- $y' = -10y$
- Backward Euler is

$$y_{i+1} = \frac{1}{1+10h} y_i$$

- Stable for any $h > 0$
- Backward Euler is absolutely (for any $h > 0$) stable

Stability

Example 2. cont.



Errors, Convergence, Stiffness

CS/SE 4X03

Ned Nedialkov

McMaster University

November 28, 2023

Outline

Local truncation error and order

- *Local truncation error* is the amount by which the exact solution fails to satisfy the numerical method
- Forward Euler $y_{i+1} = y_i + hf(t_i, y_i)$
  Using the exact solution $y(t)$ in this formula

$$d_i = \frac{y(t_{i+1}) - y(t_i)}{h} - f(t_i, y(t_i)) = \frac{h}{2} y''(\eta_i)$$

- Backward Euler $d_i = -\frac{h}{2} y''(\xi_i)$
- A method is of *order* q, if q is the lowest positive integer such that for any sufficiently smooth exact solution $y(t)$

$$\max_i |d_i| = O(h^q)$$

- Forward and backward Euler are of order $q = 1$

Local and global error

- Global error is

$$e_i = y(t_i) - y_i, \quad i = 0, 1, \ldots, N,$$

  where $y(t_i)$ is the exact solution at t_i and y_i is the computed approximation

- Consider

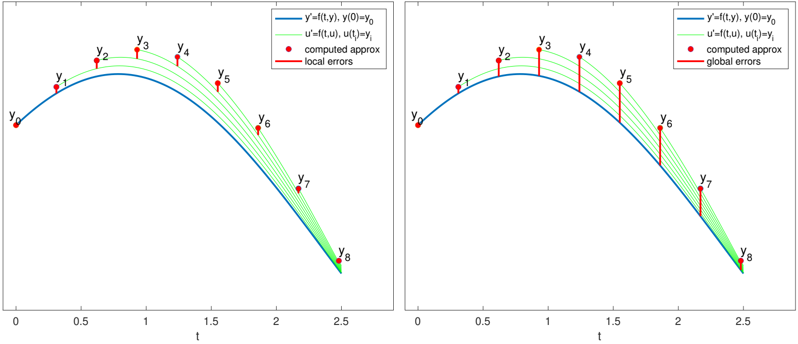
$$u' = f(t, u), \quad u(t_{i-1}) = y_{i-1}$$

  The local error is

$$l_i = u(t_i) - y_i$$

  where $u(t_i)$ is the exact solution to $u' = f(t, u)$ with initial condition u_i at t_i

Local vs global error



- Numerical methods control the local error
- That is, select a stepsize such that the local error is within a given tolerance
- Typically the global error is proportional to the tolerance

Convergence

- A methods is said to *converge* if the maximum global error goes to 0 as $h \to 0$
- That is

$$\max_i e_i = \max_i [y(t_i) - y_i] \to 0 \quad \text{as } h \to 0$$

Stiffness

- When the stepsize is restricted by stability rather than accuracy
- When an explicit solver takes very small steps
- Matlab: nonstiff solvers `ode45`, `ode113`,...
  stiff solvers: `ode15s`, `ode23s`

Stiffness cont.

Van der Pol

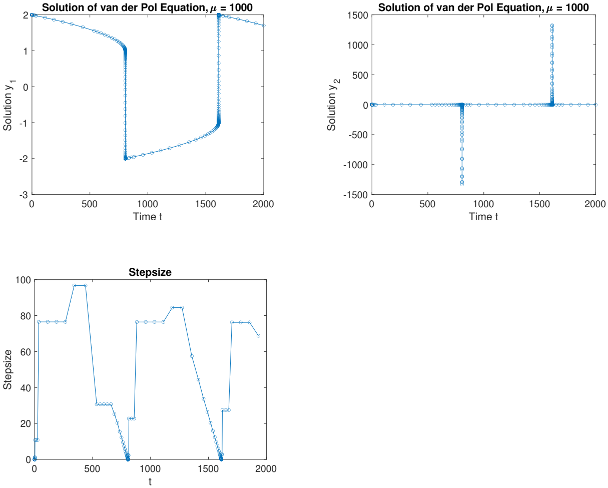
$$
\begin{aligned}
y_1' &= y_2 \\
y_2' &= \mu(1 - y_1^2)y_2 - y_1
\end{aligned}
$$

μ is a constant

$y(0) = (2, 0)^T$, $t \in [0, 2000]$

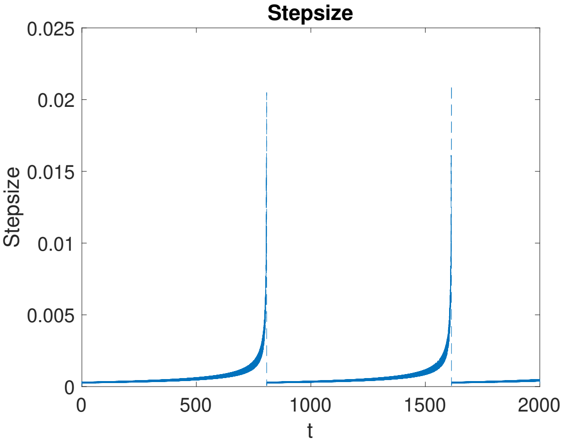
Stiff vs Nonstiff

ode15s on Van der Pol, $\mu = 1000$: integrated in ≈ 0.2 seconds, 408 steps



Stiff vs Nonstiff

ode45 on Van der Pol, $\mu = 1000$: integrated in ≈ 15 seconds, 4,624,409 steps



Runge-Kutta Methods

CS/SE 4X03

Ned Nedialkov

McMaster University

November 28, 2023

Outline

Implicit trapezoidal method

- Consider $y'(t) = f(t, y)$, $y(t_i) = y_i$
- From $y(t_{i+1}) = y(t_i) + \int_{t_i}^{t} f(s, y(s))ds$,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(s, y(s))ds$$

- Use the trapezoidal rule for the integral

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + \int_{t_i}^{t_{i+1}} f(s, y(s))ds \\ &\approx y(t_i) + \frac{h}{2}[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))] \end{aligned}$$

- From

$$y(t_{i+1}) \approx y(t_i) + \frac{h}{2}[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))]$$

write

$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_{i+1})]$$

This is the implicit trapezoidal method

- We have to solve a nonlinear system in general for y_{i+1}

- Local truncation error is

$$d_i = \frac{y(t_{i+1}) - y(t_i)}{h} - \frac{1}{2}[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))]$$

- $d_i = O(h^2)$

Explicit trapezoidal method

- In the implicit trapezoidal rule, we need to solve for y_{i+1}
- We can approximate $y(t_{i+1})$ first using forward Euler:

$$Y = y_i + hf(t_i, y_i)$$

- Then plug Y into the formula for the implicit trapezoidal method

$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, Y)]$$

- This is a two-stage explicit Runge-Kutta method
- Local truncation error is

$$d_i = \frac{y(t_{i+1}) - y(t_i)}{h} - \frac{1}{2}[f(t_i, y(t_i)) + f(t_{i+1}, y(t_i) + hf(t_i, y(t_i)))]$$

$d_i = O(h^2)$, a bit involved to derive it

Implicit midpoint

- Use the midpoint quadrature rule:

$$
\begin{aligned}
y_{i+1} &= y_i + hf\left(t_{i+1/2}, y_{i+1/2}\right) \\
&= y_i + hf\left(t_i + h/2, (y_i + y_{i+1})/2\right)
\end{aligned}
$$

- That is, we solve for y_{i+1}
- Order is 2

Explicit midpoint method

- Take a step of size $h/2$ with forward Euler

$$Y = y_i + \frac{h}{2} f(t_i, y_i)$$

- Plug into the formula from the midpoint quadrature rule:

$$y_{i+1} = y_i + hf\left(t_i + h/2, Y\right),$$

- This is a two-stage explicit Runge-Kutta method
- Order is 2

Classical 4th order Runge-Kutta

- Based on Simpson's quadrature rule
- 4 stages
- Order 4, $O(h^4)$ accuracy

$$
\begin{aligned}
Y_1 &= y_i \\
Y_2 &= y_i + \frac{h}{2} f(t_i, Y_1) \\
Y_3 &= y_i + \frac{h}{2} f(t_i + h/2, Y_2) \\
Y_4 &= y_i + h f(t_i + h/2, Y_3) \\
y_{i+1} &= y_i + \frac{h}{6} \big[f(t_i, Y_1) + 2f(t_i + h/2, Y_2) + 2f(t_i + h/2, Y_3) + f(t_{i+1}, Y_4) \big]
\end{aligned}
$$

Stepsize control

Example 1. Denote $h = t_{i+1} - t_i$. Consider forward Euler and the explicit trapezoidal methods

$$
\begin{aligned}
y_{i+1} &= y_i + hf(t_i, y_i), && \text{local error } O(h^2)\\
\widehat{y}_{i+1} &= y_i + \tfrac{1}{2}h[f(t_i, y_i) + f(t_{i+1}, y_{i+1})], && \text{local error } O(h^3)
\end{aligned}
$$

The error in y_{i+1} is $e = \|y_{i+1} - \widehat{y}_{i+1}\|$. Given tolerance tol,

if $e \leq$ tol
  accept $\widehat{y}_{i+1}$ at t_{i+1}
  predict $\bar{h}$ for the next step
else
  reject the step
  predict $\bar{h} < h$
  repeat the step with $\bar{h}$

Example 1. cont.

The error is $e = ch^2$ for some $c \geq 0$

$$c = \frac{e}{h^2}$$

Suppose $e \leq$ tol. On the next step $\bar{e} = \bar{c}\bar{h}^2$, for some $\bar{c} \geq 0$

Assume $c \approx \bar{c}$. Then

$$\begin{aligned} \bar{e} = \bar{c}\bar{h}^2 \approx c\bar{h}^2 &= \frac{e}{h^2}\bar{h}^2 \\ &= e\left(\frac{\bar{h}}{h}\right)^2 \end{aligned}$$

Example 1. cont.

From

$$\bar{e} \approx e\left(\frac{\bar{h}}{h}\right)^2 = \mathsf{tol}\,,$$

we can select

$$\bar{h} = h\left(\frac{\mathsf{tol}}{e}\right)^{1/2}$$

To reduce the likelihood of stepsize rejections, aim at $0.5\,\mathsf{tol}$ and multiply by 0.9:

$$\bar{h} = 0.9h\left(\frac{0.5\,\mathsf{tol}}{e}\right)^{1/2}$$

0.5 and 0.9 are safety factors

Example 1. cont.

If $e \geq$ tol, one can use the same formula.

How to form tol?

Assume absolute atol and relative rtol tolerances are given. Then

$$\text{tol} = \text{rtol} \cdot \|y_i\| + \text{atol}$$