Numerical Algorithms

Dov Kruger

Department of Electrical and Computer Engineering
Rutgers University

January 23, 2024

Numerical methods involve floating point

First, define floating point and how it works

- Exact and inexact representations
- Roundoff error
- Subtractive Cancellation

Then we will focus on two kinds of algorithms:

- Root finding
- Numerical Integration

Properties of Floating Point

- Floating point is an approximation to real numbers
- $a + b = b + a$ Commutivity still holds
- $a + (b + c) \neq (a + b) + c$ Associativity does NOT hold
- Some numbers are not exactly representable
- Errors are relative to size of the number
- Subtractive cancellation

Single precision: 1 sign bit, 8 exponent bits, 23 mantissa bits
seeeeeeeemmmmmmmmmmmmmmmmmmmmmmm
Exponent: -127..127

$$s * 2^{exponent-128} * mantissa$$

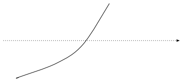
Special values for $\pm\infty$ and NaN (Not A Number)
Double precision: 1 sign bit, 11 exponent bits, 52 mantissa

Exact and Inexact Representations

Numbers which are an exact power of 2 are exactly representable
Whole numbers are exact until they exceed the size of the mantissa

- exact: $1, 2, 50, 500, 1234567$
- inexact: $1.2345678e+20, 5.671e-31$
- exact: $0.5, 0.25, 0.125, 0.375$
- inexact: $\frac{1}{3}, 0.1, 0.2, 0.01, 0.001$

Find where a function crosses the x axis
Solution for $f(x) = 0$



Bisection: Safe But Slow

Given

- Continuous function
- Function evaluates to opposite signs on both sides
- Function must cross zero

Approach: divide and conquer

1. Start with an interval [a,b]
2. Compute

```
bisection(f, a, b, ϵ)
  while b − a > ϵ
    x ← (a + b) / 2
    y ← f(x)
    if y < 0
      b ← x
    else if y > 0
      a ← x
    else
      return x
    end
  end
  return x
end
```

$$f(x) = x^2 - 3$$

We know the answer: $\pm\sqrt{3}$

Pick an initial range bracketing the root: $[1, 5]$

a	b	mid	$f(mid)$
1	5	3	6
1	3	2	1
1	2	1.5	$-.75$
1.5	2	1.75	$.0625$
1.5	1.75	1.625	

Analysis: Complexity of Bisection

Bisection gets twice as accurate for each iteration
If the initial range is guessed badly, then $O(\log n)$ Once the range is not wrong by orders of magnitude, then

- One more bit for each iteration
- number of iterations = number of bits in representation
- double = 53 bits
- On the order of 53 iterations once in the right neighborhood

Newton-Raphson

Newton's algorithm can be much faster than bisection
Quadratic in numerical methods means the number of correct digits doubles each time
Often, a good initial guess can require just 2-3 Newton "polishing steps"
Newton is somewhat dangerous, can fail to converge to an answer
Requires the derivative be available

Given

- $f(x) = x^2 - 3$
- $f'(x) = 2x$
- an initial guess x_0

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

Example:

x	$f(x)$	$f'(x)$	$x_{n+1} = x_n - f(x_n)/f'(x_n)$
5	22	10	$5 - 22/10$
3.8	11.44	7.6	$3.8 - 11.44/7.6$
1.505	$-.73418$	3.01	$\ldots$

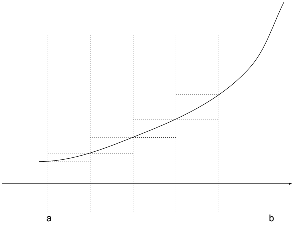
Numerical Integration

Integration is computing the area under a curve
Symbolic integration uses manipulation of the equation
Numerical integration involves a finite number of function evaluations

- Euler
- Trapezoidal
- Adaptive Quadrature
- Gauss Quadrature
- Romberg

Unfair to blame the famous mathematician by attaching his name to the worst method

Poor but instructive



Given: f(x) and n, the number of slices

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

I_n = approximation to the integral with n slices

$$f(x) = x^2$$

$$I = \int_0^1 f(x)dx = \frac{1}{3}x^3$$

Arbitrarily evaluate the function on the right side of each slice

n	h	$I_n = h \sum f(x)$	error
1	1	$1(1) = 1$	$.66666$
2	0.5	$0.5(.25 + 1) = 0.625$	$.33333$
4	0.25	$0.25(.25 + 1) = 0.625$	$.33333$

Problem: error goes down linearly. To reduce error to the 6th digit we need $O(10^6)$ iterations

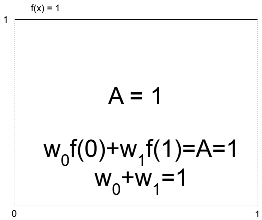
Roundoff error will destroy the answer

Trapezoidal

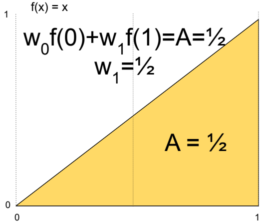
Major improvement: Average the answers

Deriving Trapezoidal for Polynomials

Trapezoidal is exact for polynomials up to x^2



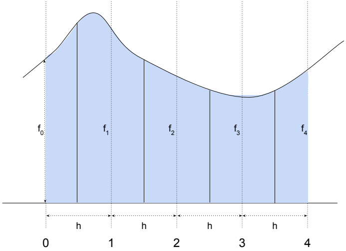
- constant



- linear

Error is $O(h^2)$

Given trapezoidal result I_n computing I_{2n} means evaluating the function at the midpoint of each slice.



Trapezoidal Pseudocode

```
trapezoidal(f, a, b, n, ϵ)
  h ← (b−a)/n
  sum ← (a+b)/2
  x ← a
  for i ← 1 to n
    sum ← sum + f(x)
    x ← x + h
  end
end
```

```
trapezoidal(f, a, b, n, ϵ)
  ...
  h2 ← h / 2
  sum2 ← sum
  x ← h2
  for i ← 1 to n
    sum2 ← sum2 + f(x)
    x ← x + h
  end
end
```

Determining Convergence

We don't generally know the answer, or we would not be computing it

```
do
    ...
while |I_n - I_2n| > ε
```

do

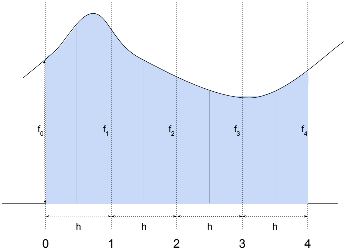
$\dots$

while $|I_n - I_{2n}| > \epsilon$

When the last two answers are close enough, we assume I_{2n} is good enough

Midpoint Method

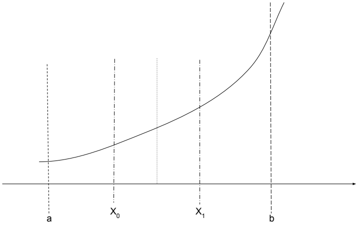
The midpoint method is similar to trapezoidal
Collect only a single value at the center of each slice
Just as accurate with one less sample
Not practically important but leads to a different method



Trapezoidal is an example of an equally spaced algorithm
Why assume that points should be equally spaced?
If we allow any x and any weight, there are 4 unknowns for a 2nd order fit
Gauss Quadrature is the result
For gauss we use different normalization:

$$I_n = \int_{-1}^{1} f(x)dx$$

Gauss 2nd order uses two samples symmetrically distributed about the midpoint of the region



Gauss 2nd order is exact for polynomials up to degree 3

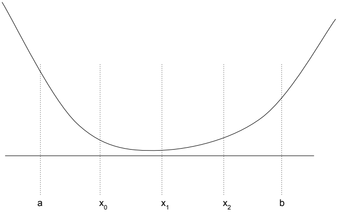
$$x_0 = mid - \frac{h}{2}\sqrt{3}, x_1 = mid + \frac{h}{2}\sqrt{3}$$

$$w_0 = 0.5, w_1 = 0.5$$

For all the following $x_0 = -\frac{\sqrt{3}}{2}, x_1 = \frac{\sqrt{(3)}}{2}$

$$\begin{array}{c|c}
f(x) = 1 & I = w_0 f(x_0) + w_1 f(x_1) = 0.5 + 0.5 = 1 \\
f(x) = x & I = w_0 f(x_0) + w_1 f(x_1) = 0 \\
f(x) = x^2 & I = w_0 f(x_0) + w_1 f(x_1) = \frac{2}{3} \\
f(x) = x^3 & I = w_0 f(x_0) + w_1 f(x_1) = 0
\end{array}$$

Gauss 3rd order uses the midpoint, and two points symmetrically to each side:



Gauss 3rd order is exact for polynomials up to degree 6

$x_0 = mid$

$x_1 = mid - h\sqrt{\frac{3}{5}}$

$x_2 = mid + h\sqrt{\frac{3}{5}}$

$w_0 = 8/9, w_1 = w_2 = 5/9$

The error of the trapezoidal method is $O(h^2)$

$$I = I_n + \alpha_1 h^2 + \dots$$

Moreover, Euler-MacLaurin theorem: only even error term

$$I = I_n + \alpha_1 h^2 + \alpha_2 h^4 + \dots$$

Calculate successive trapezoidal approximations: $I_1, I_2, I_4, \dots$
The error from each one is $O(h^2), O(\frac{h}{2}^2), O(\frac{h}{4}^2), \dots$

Romberg Integration, contd

Romberg relies on the ingenious observation that the errors can cancel

$$R_1 = \frac{4I_2 - I_1}{3}$$

$$4\alpha O(\frac{h}{2}^2) - \alpha O(h^2) = 0$$

This leaves only the next term which is $O(h^4)$.

Romberg Integration, contd

Romberg can be extended by canceling successive Rombergs to achieve $O(h^6)$ accuracy

$$Q_1 = \frac{16R_2 - 1R_1}{15}$$

This cancels the $O(h^4)$ terms, but cannot continue forever

Due to roundoff error, the terms are not really equal, so 2 times works

The third time gains some accuracy but not a full step.

We will see this in a live demo.

Future Additions

The remaining slides are not taught in the current course

If you are interested in differential equations, these will show you which algorithms to look for

Differential Equations

- Functions that relate functions and their derivatives
- Correspond to physics where local behavior is known but there is no global solution

Examples

- Real artillery problem (with friction)
- n-Body problem (gravity simulator)

Methods used are analogous to numerical integration
Euler: the simple method $O(h)$ accuracy, for understanding principle only
Runge-Kutta methods:

- compute intermediate results
- compute a combined higher-order result further in the future

Predictor-Corrector

- Using the last n values
- Construct an interpolating polynomial
- Extrapolate forward
- Use the forward point to compute backwards and apply a correction

Overview: Benefits

Generally, predictor-corrector methods are more efficient, but require multiple starting values

Runge-Kutta methods are more computationally expensive, so used to compute the initial values, then predictor corrector is used to continue

Adaptive methods vary stepsizes so if there is little curvature, the method can take great leaps, and when there is little, it slows down

RKF-45

Runge-Kutta-Fehlberg simultaneously solves a 4th order and 5th order equation

Comparing the two gives an error estimate as well at each step

RKF-45

$$k_1 = hf(x, y)$$
$$k_2 = hf(x + \frac{2}{9}h, y + \frac{2}{9}k_1)$$
$$k_3 = hf(x + \frac{1}{3}h, y + \frac{1}{12}k_1 + \frac{1}{4}k_2)$$
$$k_4 = hf(x + \frac{3}{4}h, y + \frac{69}{128}k_1 + \frac{-243}{128}k_2) + \frac{135}{64}k_3)$$
$$k_5 = hf(x + h, y + \frac{-17}{12}k_1 + \frac{27}{4}k_2) + \frac{-27}{5}k_3) + \frac{16}{15}k_4$$
$$k_6 = hf(x + \frac{5}{6}h, y + \frac{65}{432}k_1 + \frac{-5}{16}k_2) + \frac{13}{16}k_3) + \frac{4}{27}k_4 + \frac{5}{144}k_5$$
$$y_{x+h} = y_x + \frac{47}{450}k_1 + \frac{12}{25}k_3 + \frac{32}{225}k_4 + \frac{1}{30}k_5 + \frac{6}{25}k_6$$
$$T_e = \frac{-1}{150}k_1 + \frac{3}{100}k_3 + \frac{-16}{75}k_4 + \frac{-1}{20}k_5 + \frac{6}{25}k_6$$
$$h_{new} = 0.9(\frac{\epsilon}{T_e})^{\frac{1}{5}}$$

See for further details.

Predictor corrector methods

- Fit a polynomial to the last few points
- Extrapolate a new point with step size h
- Use the new point with a backward polynomial to determine error and correct

Just like Runge-Kutta methods, there are different order approximations

Wikipedia: Multistep Methods