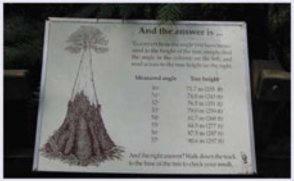
TEXTBOOKS IN MATHEMATICS

INTRODUCTION TO COMPUTATIONAL MATHEMATICS

AN OUTLINE


And the answer is ...
Measured angle
Tree height
And the right answer? Walk down the track to the base of the tree to check your result.

William C. Bauldry

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Introduction to Computational Mathematics

Textbooks in Mathematics

Series editors:

Al Boggess, Kenneth H. Rosen

Modeling Change and Uncertainty
Machine Learning and Other Techniques
William P. Fox and Robert E. Burks

Abstract Algebra
A First Course, Second Edition
Stephen Lovett

Multiplicative Differential Calculus
Svetlin Georgiev, Khaled Zennir

Applied Differential Equations
The Primary Course
Vladimir A. Dobrushkin

Introduction to Computational Mathematics: An Outline
William C. Bauldry

https://www.routledge.com/Textbooks-in-Mathematics/book-series/CANDHTEXBOOMTH

Introduction to Computational Mathematics: An Outline

William C. Bauldry

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

First edition published 2022
by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

© 2023 William C. Bauldry

CRC Press is an imprint of Taylor & Francis Group, LLC

Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, access www.copyright.com or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. For works that are not available on CCC please contact mpkbookspermissions@tandf.co.uk

Trademark notice: Product or corporate names may be trademarks or registered trademarks and are used only for identification and explanation without intent to infringe.

ISBN: 9781003299257 (ebk)
DOI: 10.1201/9781003299257

Publisher's note: This book has been prepared from camera-ready copy provided by the authors.

Cover photos: Mount Field National Park, Tall Trees Walk, Tasmania. © WmCB, 2018

Introduction to Computational Mathematics: An Outline

Photo © WmCB, 2015

"$1 + 1 = 3$
for large enough
values of 1."

Introduction to Computational Mathematics: An Outline

Contents

Note: Light blue text links inside this *Outline*; grey text links to a web page.

Introduction to Computational Mathematics: An Outline

Photo © WmCB, 2015

On two occasions I have been asked, "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" ... I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question. — Charles Babbage *Passages from the Life of a Philosopher*, p. 67.

Preface: Is Computation Important?

A simple computational error can cause very serious problems.

The *Mars Climate Orbiter* Crash

Two teams used different units of measure when writing the Orbiter's software: NASA used the metric system, and Lockheed Martin used the Imperial system. Read NASA's Mars Climate Orbiter page.

The *Ariane 5* Explosion

The European Space Agency's *Ariane 5* rocket exploded 37 seconds after takeoff; the explosion was caused by an integer overflow in the software used for launching the rocket. Watch the launch video.

The Gulf War Dhahran "Scud" Missile Attack

The Patriot missile software measured time in binary and decimal based on the system clock that used 10th's of a second, a non-terminating fraction in binary. After running for over 100 hours, the system was inaccurate. Twenty-eight U.S. soldiers died when the system failed to track a Scud missile fired from Iraq. Read Michael Barr's "Lethal Software Defects: Patriot Missile Failure."

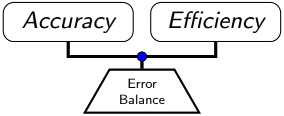
Preface: Three Factors of Computation

There are three aspects to keep in mind while studying computation. The first two are always in tension with each other:

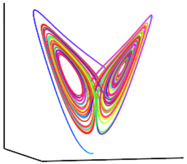
Accuracy versus **Efficiency**

Accuracy concerns how much error occurs and how to control the error.

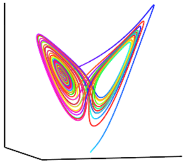
Efficiency concerns how much computation is needed to produce a result.



Stability is the third aspect to always keep in mind. In a stable computation, a small change in inputs produces only a small change in outputs. As a counterpoint, investigate instability in a "Lorenz attractor."



Starting at $[1,1,1]$.



Starting at $[-1,-2,-3]$.

Preface: A Computational Mathematics Course

This slide deck was developed while teaching Appalachian State's MAT 2310, Computational Mathematics (3 cr). The course was designed to introduce numerical analysis and give students experience with basic programming structures in a mathematical environment. A sample syllabus is as follows:

1. Computer Arithmetic . . . . . . 2 weeks
2. Control Structures . . . . 2 weeks
3. Numerical Differentiation . . 3 weeks

   Midterm Exam
4. Root-Finding Algorithms . . . 3 weeks
5. Numerical Integration . . . . . 2 weeks
6. Polynomial Interpolation . . . 2 weeks

   Team Project Posters . . . . . . 1 week

   Final Exam

Programming was done with the computer algebra system Maple, to easily adjust precision, and with Python. The Special Topics and Case Study come in as time allows. A selection of group projects appears at the end of this *Outline*.

Thanks go to the students of MAT 2310 who lived through the development of both a new course and these slides. Many thanks also go to my colleagues Greg Rhoads, René Salinas, and Eric Marland who co-designed the course and gave great feedback during the adventure.

— *WmCB, Aug 2022*

I. Computer Arithmetic

Sections

DOI: 10.1201/9781003299257-1

I. Computer Arithmetic: Scientific Notation

Definitions of Scientific Notation

Normalized: Any numeric value can be written as

$$d_0.d_1d_2d_3\ldots d_n \times 10^p$$

where $1 \le d_0 \le 9$.

Engineering: Any numeric value can be written as

$$n.d_1d_2d_3\ldots d_m \times 10^q$$

where $1 \le n \le 999$ and q is a multiple of 3.

Examples (NIST's Values of Constants)

- Speed of light in a vacuum: $2.997\,924\,58 \times 10^8$ m/s
- Newtonian constant of gravitation: $6.673\,84 \times 10^{-11}$ $\mathrm{m}^3/(\mathrm{kg}\cdot\mathrm{s}^2)$
- Avogadro's number: $6.022\,141 \times 10^{-23}$ mol^{-1}
- Mass of a proton: $1.672\,621\,777 \times 10^{-27}$ kg
- Astronomical unit: $92.955\,807\,27 \times 10^6$ mi

Conversions

Basic Base Transmogrification: Integers

Binary → Decimal
(*Vector version*)
Think of the binary number as a vector of 1's and 0's. Use a dot product to convert to decimal.

1. $x_2 = 101110$
2. $x_{10} =$
$$\begin{array}{r} \langle\ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \rangle \\ \bullet \langle\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ \rangle \end{array}$$
3. $x_{10} = 2^5 + 2^3 + 2^2 + 2^1$
$= 46$

Decimal → Binary
(*Algebra version*)
Successively compute the bits (from right to left)

1. $bit = x \mod 2$
then set $x = \lfloor x/2 \rfloor$
2. Repeat until $x = 0$

E.g., $x_{10} = 46$

$b_0 = 0$; then set	$x = 23$
$b_1 = 1$;	$x = 11$
$b_2 = 1$;	$x = 5$
$b_3 = 1$;	$x = 2$
$b_4 = 0$;	$x = 1$
$b_5 = 1$;	$x = 0$

Whence $x_2 = 101110$

Conversions

Basic Base Transmogrification: Fractions

Binary → Decimal

(*Vector version*)

Think of the binary number as a vector of 1's and 0's. Use a dot product to convert to decimal.

1. $x_2 = 0.10111$
2. $x_{10} =$
$$\begin{array}{c} \langle \; 1 \quad 0 \quad 1 \quad 1 \quad 1 \; \rangle \\ \bullet \langle\, 2^{-1}\, 2^{-2}\, 2^{-3}\, 2^{-4}\, 2^{-5} \,\rangle \end{array}$$
3. $x_{10} = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5}$
$= 0.71875$

Decimal → Binary

(*Algebra version*)

Successively compute the bits (from left to right)

1. $bit = \lfloor 2x \rfloor$
then set $x = \text{frac}(2x)$
2. Repeat until $x = 0$ (or when reaching maximum length)

E.g., $x_{10} = 0.71875$

$$\begin{array}{ll} b_{-1} = 1; \text{ then set} & x = 0.43750 \\ b_{-2} = 0; & x = 0.87500 \\ b_{-3} = 1; & x = 0.75000 \\ b_{-4} = 1; & x = 0.50000 \\ b_{-5} = 1; & x = 0.0 \textit{ Stop} \end{array}$$

Whence $x_2 = 0.10111$

Conversions

Terminating Expansions?

When does a fraction's expansion terminate?

Base 10: A decimal fraction terminates when $r = \dfrac{n}{10^p} = \dfrac{n}{2^p \cdot 5^p}$.

Base 2: A binary fraction terminates when $r = \dfrac{m}{2^p}$.

Examples

1. $\frac{1}{10} = 0.1_{10} = 0.0\overline{0011}_2$
2. $\frac{1}{3} = 0.\overline{3}_{10} = 0.\overline{01}_2$
3. $\sqrt{2} \doteq 1.414\,213\,562\,373\,095\,048\,8_{10} \doteq 1.0110\,1010\,0000\,1001\,111_2$
4. $\pi \doteq 3.141\,592\,653\,589\,793\,238\,5_{10} \doteq 11.0010\,0100\,0011\,1111\,01_2$

Conversions

Examples (Convert a Repeating Binary Expansion)

Convert $n = 0.0101\,101\,101\,01\cdots = 0.0\overline{101}_2$ to decimal.

1. Convert the repeating block to decimal:

$$101_2 = 5_{10}$$

2. Rewrite n in "powers-of-two" notation:

$$n = 5 \cdot 2^{-4} + 5 \cdot 2^{-7} + 5 \cdot 2^{-10} + 5 \cdot 2^{-13} + \cdots$$

3. Express n as a geometric series:

$$n = 5 \cdot 2^{-4} \cdot \sum_{k=0}^{\infty} 2^{-3k}$$

4. And sum the series:

$$n = 5 \cdot 2^{-4} \cdot \frac{1}{1 - 2^{-3}} = \frac{5}{14}$$

Binary Coded Decimal

BCD

The digits 0 to 9 can be represented with four binary bits:

$$\underset{8}{\boxed{\text{x}}}\underset{4}{\boxed{\text{x}}}\underset{2}{\boxed{\text{x}}}\underset{1}{\boxed{\text{x}}}$$

For example, 93_{10} would be

$$\text{BCD: } \overbrace{\underset{8}{\boxed{1}}\underset{4}{\boxed{0}}\underset{2}{\boxed{0}}\underset{1}{\boxed{1}}}^{9}\overbrace{\underset{8}{\boxed{0}}\underset{4}{\boxed{0}}\underset{2}{\boxed{1}}\underset{1}{\boxed{1}}}^{3} \quad vs. \quad \text{Binary: } \overbrace{\underset{128}{\boxed{0}}\underset{64}{\boxed{1}}\underset{32}{\boxed{0}}\underset{16}{\boxed{1}}\underset{8}{\boxed{1}}\underset{4}{\boxed{1}}\underset{2}{\boxed{0}}\underset{1}{\boxed{1}}}^{93}$$

Advantages

- Eliminates some repeating expansions
- Rounding is simpler
- Displaying values is easier

Disadvantages

- Fewer numbers per 8 bits $(100/256 \approx 39\%)$
- Complicated arithmetic routines
- Slower to compute

Nearly all calculators use BCD formats.

Floating Point Numbers

Definition (Floating Point Representation)

A number x is represented (and approximated) as

$$x \doteq \sigma \times f \times \beta^{e-p}$$

where

σ: **sign** ± 1, f: **mantissa**, β: **base**, usually 2, 10, or 16

e: **biased exponent** (shifted), p: exponent's **bias** (shift)

The standard floating point storage format is

σ	e	f

Exponent Bias

The bias value is chosen to give equal ranges for positive and negative exponents without needing a sign bit. E.g., for an exponent with

- 8 bits: $0 \le e \le 255 = 2^8 - 1$. Use $p = 2^8/2 - 1$ gives an exp range of
$$-127 \le e - 127 \le 128.$$
- 11 bits: $0 \le e \le 2047 = 2^{11} - 1$. Use $p = 2^{11}/2 - 1 = 1023$ gives
$$-1023 \le e - 1023 \le 1024.$$

Samples

Examples

1. $-3.95 = (-1)^1 \times 0.1234375 \times 2^{21-16}$
   So $\sigma = 1$, $f = 0.1234375$, $\beta = 2$, $e = 21$, and $p = 16$.

   Note: $(-1)^1 \times 0.1234375 \times 2^{21-16} = -3.950$, so $err = 0$.

   Storage format: $\boxed{1}\boxed{21}\boxed{0.1234375}$

---

2. $11/3 = (-1)^0 \times 0.2291666667 \times 16^{16384-16383}$
   So $\sigma = 0$, $f = 0.2291666667$, $\beta = 16$, $e = 16384$, and $p = 16383$.

   Note: $(-1)^0 \times 0.2291666667 \times 16^{16384-16383} = 3.6666666672$,
   so $err = 5.\overline{3} \cdot 10^{-10}$.

   Storage format: $\boxed{0}\boxed{16384}\boxed{0.2291666667}$

---

3. $2^{10} = 1024 = (-1)^0 \times 0.250 \times 16^{66-63}$
   So $\sigma = 0$, $f = 0.250$, $\beta = 16$, $e = 66$, and $p = 63$.

   Note: $(-1)^0 \times 0.250 \times 16^{66-63} = 1024.0$, so $err = 0$.

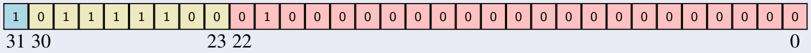
   Storage format: $\boxed{0}\boxed{66}\boxed{0.2500000000}$

IEEE Standard for Floating-Point Arithmetic

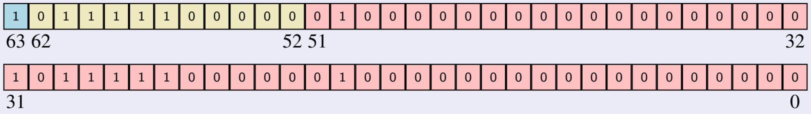
Definition (IEEE-754)

Normalized Floating Point Representation (Binary)

Single precision: $x \doteq (-1)^{\sigma} \times (1. + f_{[23]}) \times 2^{e_{[8]}-127}$ (32 bit)



Double precision: $x \doteq (-1)^{\sigma} \times (1. + f_{[52]}) \times 2^{e_{[11]}-1023}$ (64 bit)



Online Floating-Point Converter

Original IEEE-754-1985, 2008 revision, 2019 revision ($)

IEEE Standard for Floating-Point Arithmetic, II

Single Precision Bit Patterns

Pattern			**Value**
$0 < e < 255$			$n = (-1)^{\sigma} \times 2^{e-127} \times 1.f$ normal number
$e = 0$	$f = 0$ *all bits are zero*		$n = (-1)^{\sigma} \times 0.0$ signed zero
	$f \neq 0$ *at least 1 nonzero bit*		$n = (-1)^{\sigma} \times 2^{-126} \times 0.f$ subnormal number
$e = 255$	$f = 0$	$\sigma = 0$	+INF plus infinity
		$\sigma = 1$	−INF minus infinity
	$f \neq 0$		NaN Not-a-Number

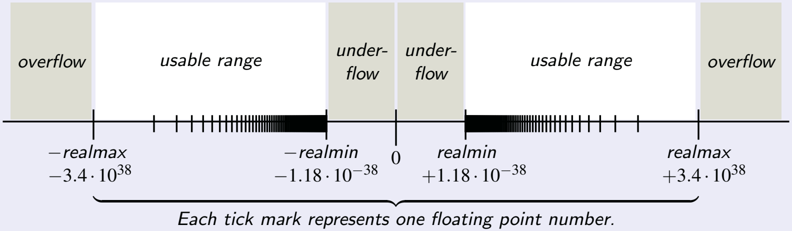
Big and Small & Gaps

IEEE-754 Largest and Smallest Representable Numbers

Precision	*Digits*	*Max Exp*	*Smallest #*	*Largest #*
Single	≈ 9	≈ 38.2	$\approx 1.18 \cdot 10^{-38}$	$\approx 3.4 \cdot 10^{38}$
Double	≈ 17	≈ 307.95	$\approx 2.225 \cdot 10^{-307}$	$\approx 1.798 \cdot 10^{308}$

Gaps in the Floating Point Number Line

The size of the gap between consecutive floating point numbers gets larger as the numbers get larger.



Machine Epsilon

Definition

The *machine epsilon* ε is the largest value such that

$$1 + \varepsilon = 1$$

for a given numeric implementation.

Example (Single Precision [*using Java*])

```
wmcb:▶ cat machineEpsilon.java
class mEps {
public static void main(String[] args) {
    float machEps = 1.0f;
    do {
      machEps /= 2.0f;
    } while ((float)(1.0+(machEps/2.0)) != 1.0);
    System.out.println("Calculated machine epsilon: "+machEps);
    }
}
wmcb:▶ javac machineEpsilon.java
wmcb:▶ java mEps
Calculated machine epsilon:  1.1920929E-7
```

$\Longrightarrow \varepsilon_s \approx 1.192 \cdot 10^{-7}$

Machine Epsilon, II

Example (Double Precision [*using Java*])

```
wmcb:▶ cat machineEpsilonD.java

class mEpsD {
public static void main(String[] args) {
    double machEps = 1.0d;
    do {
      machEps /= 2.0d;
    } while ((double)(1.0 + (machEps/2.0)) != 1.0);
    System.out.println("Calculated machine epsilon: " + machEps);
    }
}

wmcb:▶ javac machineEpsilonD.java

wmcb:▶ java mEpsD
Calculated machine epsilon: 2.220446049250313E-16
```

$\Longrightarrow \varepsilon_d \approx 2.22 \cdot 10^{-16}$

Single Precision	Double Precision
$\varepsilon_s \approx 1.192 \cdot 10^{-7}$	$\varepsilon_d \approx 2.22 \cdot 10^{-16}$

Machine Epsilon, III

Example (Using Python 3)

```
>>> macEps = 1.0
>>> while (1.0 + macEps) != 1.0:
      macEps = macEps/2.0

>>> macEps
1.1102230246251565e-16
>>>
>>> import numpy as np
>>> macEpsL = np.longdouble(1.0)
>>> while (1.0 + macEpsL) != 1.0:
      macEpsL = macEpsL/2.0

>>> macEpsL
5.42101086242752217e-20
>>>
>>> np.finfo(np.longdouble)
finfo(resolution=1.00000000000000000715e-18,
min=-1.18973149535723176e+4932, max=1.18973149535723176e+4932,
dtype=float128)
```

Note: Python's `float` defaults to double precision.

Large Value Floating Point Gap

Example (Double Precision [*using Java*])

- Approximate the gap to the next floating point value above 10^{30}.

```
wmcb:▶ cat FPGap.java

class BigGap {
public static void main(String[] args) {
    float gap = 1e23f;
    float n = 1e30f;
    do {
      gap /= 2.0;
     } while ((float)(n+(gap/2.0)) != n);
    System.out.println("Approximate gap: " + eps);
    }
}
wmcb:▶ javac FPGap.java

wmcb:▶ java BigGap
  Approximate gap: 5.0E22
```

Properties

Floating Point Arithmetic Properties

Commutative: Addition is commutative:

$$n_1 + n_2 = n_2 + n_1$$

Multiplication is commutative:

$$n_1 \times n_2 = n_2 \times n_1$$

Nonassociative: Addition is **not** associative:

$$(n_1 + n_2) + n_3 \neq n_1 + (n_2 + n_3)$$

Multiplication is **not** associative:

$$(n_1 \times n_2) \times n_3 \neq n_1 \times (n_2 \times n_3)$$

Nondistributive: Multiplication does **not** distribute over addition:

$$n_1 \times (n_2 + n_3) \neq (n_1 \times n_2) + (n_1 \times n_3)$$

Maple's Floating Point Representation

Maple's Floating Point Implementation

$$\begin{aligned} \text{Maximum exponent} &= 9223372036854775806 \\ \text{Minimum exponent} &= -9223372036854775806 \\ \text{Maximum ``float''} &= 1. \times 10^{9223372036854775806} \\ \text{Minimum ``float''} &= 1. \times 10^{-9223372036854775806} \\ \text{Maximum digits} &= 38654705646 \\ \text{Maximum binary power} &= 4611686018427387903 \end{aligned}$$

Example (Maple's Floating Point Structure)

```
> N := evalf(Pi, 20):
  dismantle(N)
```

```
    FLOAT(3):  3.1415926535897932385
        INTPOS(6):  31415926535897932385
        INTNEG(2):  -19
```

Floating Point Rounding

IEEE-754 Rounding Algorithms[1]

Rounding to Nearest

- Round to nearest, ties to even (*default for binary floating-point*)
- Round to nearest, ties to odd
- Round to nearest, ties away from zero (*used by Maple and MATLAB*)

Directed Roundings

- Round toward 0 — *truncation*
- Round toward $+\infty$ — *rounding up* or *ceiling*: $\lceil x \rceil$
- Round toward $-\infty$ — *rounding down* or *floor*: $\lfloor x \rfloor$

Team Project: Implement *Stochastic Rounding* in Maple or Python.

[1] See *EE Times*, "Rounding Algorithms"

Error

Defining Error

Absolute Error: The value $err_{abs} = |actual - approximate|$

Relative Error: The ratio $err_{rel} = \dfrac{|actual - approximate|}{|actual|} = \dfrac{err_{abs}}{|actual|}$

Example (Weighty Thoughts)

A *long-tailed field mouse* normally weighs up to about 50 g. Suppose a lab-tech makes an error of 2.5 g ($\approx$ *a penny*) when weighing a mouse. The relative error is

$$err_{rel} = \frac{2.5\,\mathrm{g}}{50\,\mathrm{g}} = 5\%.$$

A mature *African bush elephant* normally weighs about 6.5 tons. Suppose a zookeeper makes an error of 50# ($\approx$ 7-*yr-old boy*) weighing an elephant. The relative error is

$$err_{rel} = \frac{50\#}{13000\#} \doteq 0.4\%.$$

Error Accumulates

Adding Error

Add $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{10^6}$ forward and backward with 6 digits.

Maple

```
Digits := 6:
N := 10^6:
```

```
Sf := 0;
for i from 1 to N do
  Sf := Sf + (1.0/i);
  end do:
Sf;
```

10.7624

```
Sb := 0;
for j from N to 1 by -1 do
  Sb := Sb + (1.0/j);
  end do:
Sb;
```

14.0537

The correct value of $\displaystyle\sum_{k=1}^{10^6} \frac{1}{k}$ to 6 significant digits is $\boxed{14.3927}$.

$$\text{relative error}(S_f) \approx 25.2\%, \qquad \text{relative error}(S_b) \approx 2.4\%$$

What happened?

Error Accumulates

Subtracting Error

Solve for x: $1.22x^2 + 3.34x + 2.28 = 0$ *(3-digit, 2-decimal precision)*

The quadratic formula $r_\pm = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ can lead to problems.

Using the formula directly:

$$\begin{aligned} b^2 &= 11.2 \\ 4ac &= 11.1 \\ \sqrt{b^2 - 4ac} &= 0.32 \\ r_+, r_- &= -1.24, -1.50 \end{aligned}$$

But the exact roots are:

$$\begin{aligned} R_\pm &= \frac{-167 \pm \sqrt{73}}{122} \\ &\doteq -1.30, -1.44 \end{aligned}$$

The *relative error* is $\approx 5\%$.

"Rationalize the numerator" to eliminate a bad subtraction:

$$R_- = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

More Error Accumulates

Even Worse Subtraction Error

Solve for x: $0.01x^2 - 1.00x + 0.02 = 0$ *(3-digit, 2-decimal precision)*

Again, using the quadratic formula directly:

$$4ac = 0.0008 \doteq 0.00$$

$$\sqrt{b^2 - 4ac} \doteq 1.00$$

$$r_\pm \doteq 100.,\ 0.00$$

But the real roots are:

$$R_\pm \doteq 99.98,\ 0.02$$

The *relative errors* are

$$err_{rel} \approx 0.02\% \ \&\ 100\%\,!$$

Again, "rationalize the numerator" to eliminate a bad subtraction:

$$R_- = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

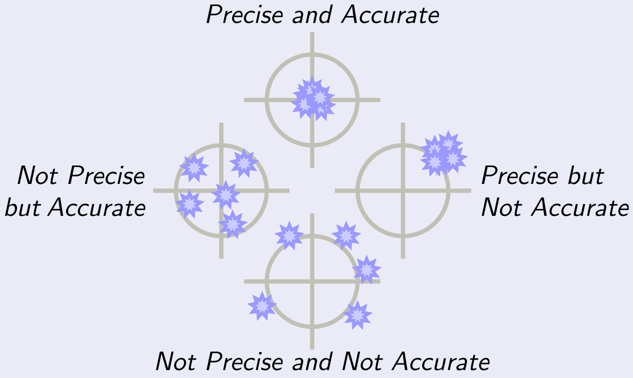
$$\frac{-b - \sqrt{b^2 - 4ac}}{2a} \doteq 0.00 \qquad \text{but} \qquad \frac{2c}{-b + \sqrt{b^2 - 4ac}} \doteq 0.02$$

Accuracy v. Precision

On-Target v. Grouping

Accuracy: How closely computed values agree with the true value.

Precision: How closely computed values agree with each other.


Precise and Accurate
Not Precise but Accurate
Precise but Not Accurate
Not Precise and Not Accurate

Roundoff v. Truncation

Computational v. Formulaic

Roundoff: Error from floating point arithmetic (*fixed number of digits*)

Truncation: Error from formula approximation (*dropping terms*)

Examples

- Roundoff

$$\tfrac{6}{10}+\tfrac{5}{10}=\tfrac{11}{10} \Longleftrightarrow 1.+1. \overset{?}{=} 1.$$

$$\tfrac{4}{10}+\tfrac{5}{10}+\tfrac{6}{10}+\tfrac{7}{10}+\tfrac{8}{10}=3 \Longleftrightarrow 0.+1.+1.+1.+1. \overset{?}{=} 3.$$

- Truncation

$$\sin(\theta)=\sum_{k=1}^{\infty}\tfrac{\theta^{2k-1}}{(2k-1)!} \Longleftrightarrow \sin(\theta)\approx\theta-\tfrac{1}{6}\theta^3$$

$$\tan(\theta)=\sum_{k=1}^{\infty}(-1)^n\,\tfrac{B_{2n}4^n(1-4^n)}{(2n)!}\,\theta^{2n-1} \Longleftrightarrow \tan(\theta)\approx\theta+\tfrac{1}{3}\theta^3$$

Landau Notation

"Big-O"

We use Landau's notation to describe the *order* of terms or functions:

Big-$\mathcal{O}$: If there is a constant $C > 0$ such that $|f(x)| \leq C \cdot |g(x)|$ for all x near x_0, then we say $f = \mathcal{O}(g)$ [*that's "f is 'big-O' of g"*].[2]

Examples

1. For x near 0, we have $\sin(x) = \mathcal{O}(x)$ and $\sin(x) = x + \mathcal{O}(x^3)$.
2. If $p(x) = 101x^7 - 123x^6 + x^5 - 15x^2 + 201x - 10$, then
   - $p(x) = \mathcal{O}(x^7)$ as $x \to \infty$.
   - $p(x) = \mathcal{O}(1)$ for x near 0.
3. As $x \to \infty$, is $x^n = \mathcal{O}(e^x)$ for every $n \in \mathbb{N}$?
4. As $x \to \infty$, is $\ln(x) = \mathcal{O}(x^{1/n})$ for every $n \in \mathbb{N}$?

[2] Link to ▶ *Further big-O info*.

Exercises, I

Problems

Scientific Notation

1. Convert several constants at NIST to engineering notation.

Converting Bases

2. Convert to decimal: 101110, 101×2^{10}, 101.0111, $1110.0\overline{01}$
3. Convert to binary (to 8 places): 10^5, $1/7$, 1234.4321, π, $\sqrt{2}$.
4. Express 831.22 in BCD form.
5. Write the BCD number 1001 0110 0011 . 1000 0101 in decimal.
6. Investigate converting bases by using synthetic division.

Floating Point Numbers

7. Convert 31.3875_{10} to floating point format with base $\beta = 10$ and bias $p = 49$.
8. Convert from floating point format with base $\beta = 2$ and bias $p = 127$:

1	126_{10}	5141_{10}

9. Why is the gap between successive values larger for bigger numbers when using a fixed number of digits?
10. Give an example showing that floating point arithmetic is not distributive (mult over add).

Exercises, II

Problems

IEEE-754 Standard

11. Write $20/7$ in single-precision format. In double-precision.

12. Convert the single-precision #

0	10000111	0010010...0

    to decimal.

13. Chart double-precision bit patterns.

14. Describe a simple way to test if a computation result is either infinite or NaN.

15. What is the purpose of using *round to nearest, ties to even?*

16. Explain the significance of the machine-epsilon value.

Error

17. The US Mint specifies that quarters weigh 5.670 g. What is the largest acceptable weight, if the relative error must be no more than 0.5%?

18. Find the relative error when adding $1+\frac{1}{2}+\cdots+\frac{1}{10^5}$ using 5-digit arithmetic.

19. Show that $\cos(x) = \mathcal{O}(1)$ for x near 0.

20. Let p be a polynomial with $n = \text{degree}(p)$. Find k so that

    a. $p(x) = \mathcal{O}(x^k)$ as $x \to \infty$.
    b. $p(x) = \mathcal{O}(x^k)$ for $x \approx 0$.

II. Control Structures

Sections

DOI: 10.1201/9781003299257-2

II. Control Structures: Flow

Flow Control

Conditional Statements. A *condition* determines an action:

If the condition is true, then do an action.

If the condition is not true,[3] then do a different action.

E.g.,

- If a number is even, divide it by 2. Otherwise mult by 3 & add 1.
- If error is less than 10^{-5}, stop. If not, reapply Newton's method.

Repeating Blocks / Loops: Repeat an action a specified number of times (NB: *Loops embed a conditional*):

Count to a value doing an action each time.

E.g.,

- Add the first 20 prime numbers.
- Starting at $t = 0; y_0 = 0$, use Euler's method to find $y(1)$ when $y' = t$.

[3] Is there a difference between *not true* and *false*? See, e.g., *Intuitionistic Logic* at the Stanford Encyclopedia of Philosophy.

Types of Conditional Statements

Basic Conditional Types

```
Simple:   IF condition THEN do action for true
          IF condition THEN do action for true ELSE do action for false

Compound: IF $condition_1$ THEN $do\ action_1$
              ELSE IF $condition_2$ THEN $do\ action_2$
              ELSE IF $condition_3$ THEN $do\ action_3$
                 ...
              ELSE $do\ action_n$ when all conditions are false
```

Example (NC 2011–2013 Tax Rate Schedule[4])

IF your filing status is single;

and taxable income is more than:	**but not over:**	**your tax is:**
\$0	\$12,750	6% OF THE NC TAXABLE INCOME AMOUNT ON FORM D-400
\$12,750	\$60,000	\$765 + 7% OF THE AMOUNT OVER \$12,750
\$60,000	...	\$4,072.50 + 7.75% OF THE AMOUNT OVER \$60,000

[4] In 2014, NC converted to a regressive "flat tax" currently at 5.25% (2021).

Types of Loops

Loop Types

Counting Loops: `For` loops perform an action a pre-specified number of times.

Condition Loops: `While` loops perform an action as long as a given condition is true.

Examples

- `For` each employee, calculate their monthly pay.
- `For` each integer i less than n, compute the ith number in the Fibonacci sequence.

- `While` the current remainder in the Euclidean algorithm is greater than 1, calculate the next remainder.
- `While` the game isn't over, process the user's input.

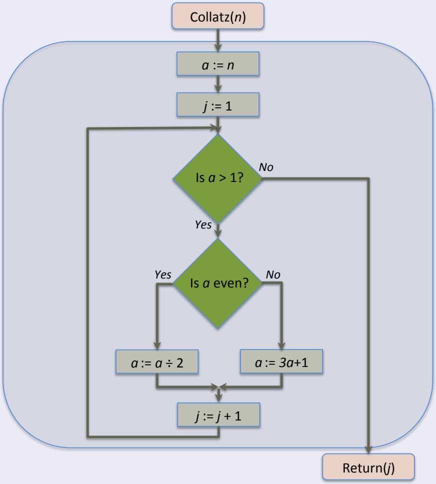
Example: Collatz Flow Chart

Collatz Function

- Start with an integer greater than 1. If it's even, divide it by 2. Otherwise, multiply it by 3 then add 1. Repeat until the value reaches 1 counting the number of steps.

A program to calculate the number of steps requires a loop with a conditional inside.

XKCD: Collatz Conjecture



Example: Collatz — A Loop and a Conditional

Pseudocode

We see a conditional loop in the Collatz function's flow chart:

```
while (the term $> 1$) do
    Calculate the next term
    end do
```

There is an "if" statement inside the loop to calculate the new term:

```
If (the term is even) then
    divide by $2$
else
    multiply by $3$, then add $1$
end if
```

Putting these together gives:

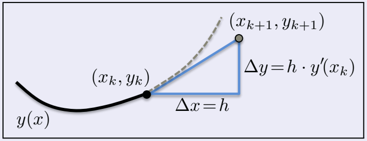
```
Get the first term
Set the term counter to $1$
while (the term $> 1$) do
    If (the term is even) then divide by $2$
    else multiply by $3$, then add $1$
    end if
    Increment the term counter
    end do
Return the term counter
```

Example: Euler's Method as a Loop

Euler's Method

The solution to a differential equation $y' = f(x,y)$ can be approximated using the differential triangle. Calculate the next point from the current point (x_k, y_k) by following the tangent line for a step $\Delta x = h$. Then the new point is

$$(x_{k+1}, y_{k+1}) = (x_k + h, y_k + h \cdot y'(x_k, y_k)).$$



Implement Euler's method as a loop:

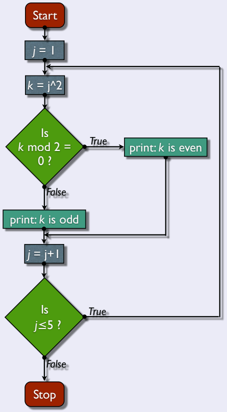
```
Define the derivative function $y' = f(x,y)$
Get the initial point $(x_0, y_0)$
Get the stepsize $h$
Determine the number of steps $n = (b-a)/h$
for i from 1 to n do
    Compute $x_{k+1} = x_k + h$ and $y_{k+1} = y_k + h \cdot y'(x_k, y_k)$
    end do
Return the collection of points $\{(x_i, y_i)\}_{i=0}^{n}$
```

Control Structures Example: Flowchart

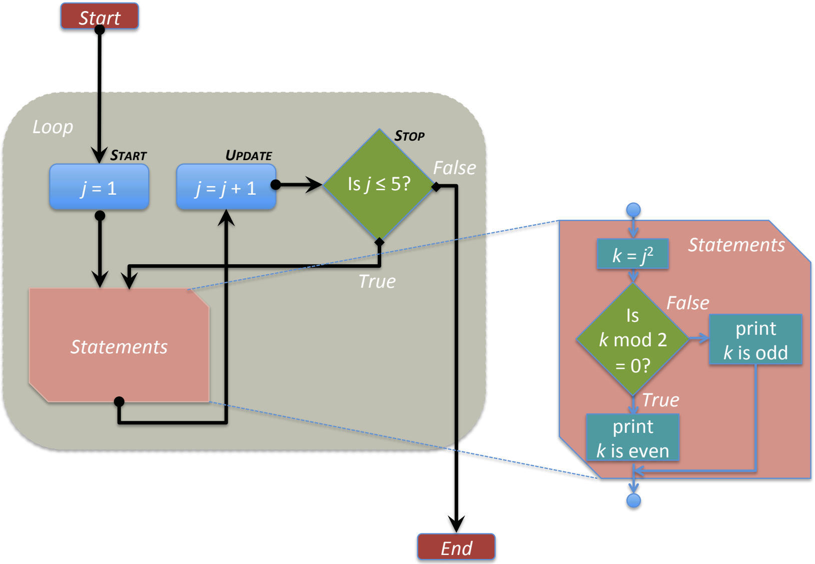
A Common Sample Problem

1. List the squares of the first 5 integers showing which are even and which are odd.

- Use a `for` loop to step through the integers and an `if-then` conditional to test for even or odd.



Control Structures Examples: Diagram



Excel Control Structures

Conditional Statements

If: =`IF`(*condition*, *true action*, *false action*)

[*Can nest up to 7 IFs*: =`IF`(*condition*$_1$, `IF`(*condition*$_2$, ..., ...), ...)]

[*But I've nested 12 deep without problems...*]

Case: =`CHOOSE`(*index*, *case_1*, *case_2*, ...)

[*Maximum of 29 cases*) (*see also:* `LOOKUP`)]

Note

Many versions of Excel include *Visual Basic for Applications* (VBA), a small programming language for macros. VBA includes a standard `if-then-else/elseif-end if` structure. (See the *Excel Easy* web tutorial.)

Excel Control Structures

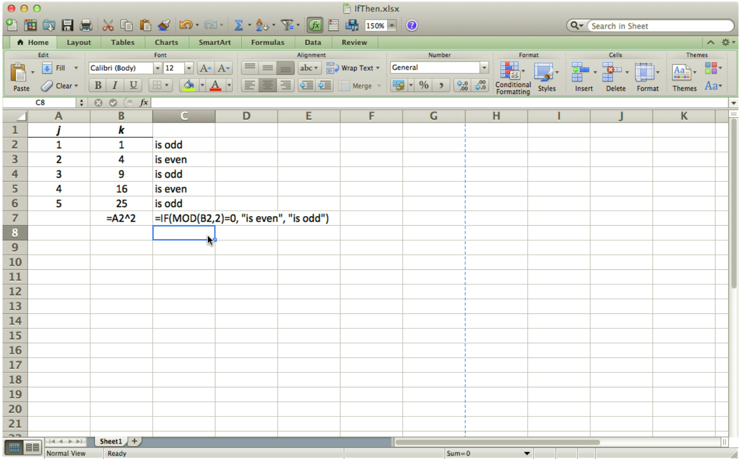
Loop Structures

For:	N/A *(must be programmed in VBA)*
While:	N/A *(must be programmed in VBA)*

View:

- Excel (Mac) website
- Excel (Win) website

Excel Control Structures Example



	A	B	C
1	*j*	*k*	
2	1	1	is odd
3	2	4	is even
4	3	9	is odd
5	4	16	is even
6	5	25	is odd
7		=A2^2	=IF(MOD(B2,2)=0, "is even", "is odd")

Maple Control Structures

Conditional Statements

If: `if` *condition* `then` *statements*;
`else` *statements*;
`end if`

`if` *condition* $_1$ `then` *statements*;
`elif` *condition* $_2$ `then` *statements*;
`else` *statements*;
`end if`

Case: N/A (*use* `piecewise` *or* `if-elif-end if`)

Maple Control Structures

Loop Structures

For: `for` *index* `from` *start_value* $|1$ `to` *end_value*

```
  by increment|1 do
      statements;
      end do
```

```
for index in expression_sequence do
      statements;
      end do
```

While:

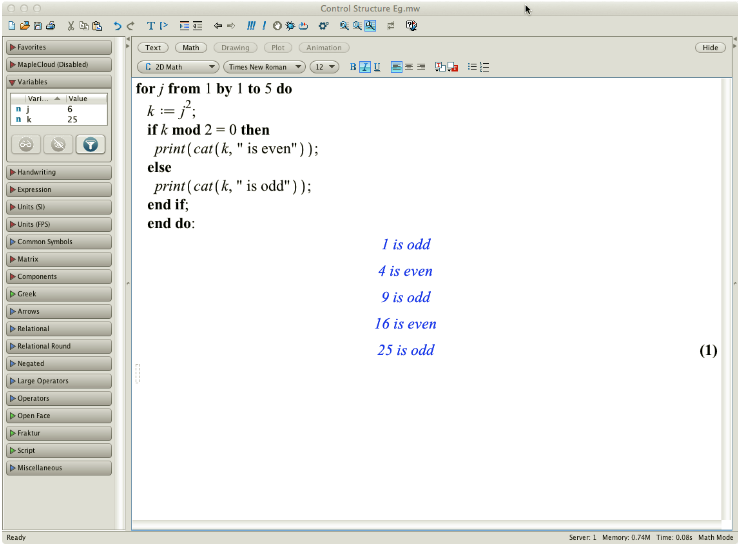
```
while condition do
      statements;
      end do
```

View:

- Maple website
- Maple Online Help

(*See also:* the Sage, Xcas, and TI Nspire, Maxima, or Mathematica website.)

Maple Control Structures Example



Vari...	Value
j	6
k	25

for *j* **from** 1 **by** 1 **to** 5 **do**
  $k := j^2$;
  if *k* **mod** 2 = 0 **then**
   print(*cat*(*k*, " is even"));
  else
   print(*cat*(*k*, " is odd"));
  end if;
  end do:

```
1 is odd
4 is even
9 is odd
16 is even
25 is odd                (1)
```

MATLAB[1] Control Structures

Conditional Statements

If: `if` *condition*; *statements*; `else` *statements*; `end`

`if` *condition*; *statements*; `elseif` *condition*; *statements*;
`else` *statements*; `end`

Case: (Scilab uses `select`)

```
switch index
  case value1
    statements;
  case value2
    statements;
        ⋮
  otherwise
    statements;
  end
```

[1] FreeMat, Octave, and Scilab are FOSS clones of MATLAB. Also see GDL and R.

MATLAB / Octave / Scilab Control Structures

Loop Structures

For:
```
for index = startvalue:increment:endvalue
    statements
end
```

While:
```
while condition
    statements
end
```

View:

- MATLAB website
- Octave website
- Scilab website
- FreeMat website

MATLAB Control Structures Example

```
octave-3.4.0:1>
> for j=1:1:5;
> k=j*j;
> if mod(k,2)==0;
>    printf("%d is even\n", k);
> else
>    printf("%d is odd\n", k);
> end;  % of if
> end;  % of for
1 is odd
4 is even
9 is odd
16 is even
25 is odd
octave-3.4.0:2>
```

C / Java Control Structures

Conditional Statements

If: `if (`*condition*`) {`*statements*`}`

```
if (condition) {statements}
else {statements}
```

Case:

```
switch (index) {
    case 1: statements ; break;
    case 2: statements ; break;
      ⋮
    case n: statements ; break;
    default: statements }
```

(*See also:* Lua.)

C / Java Control Structures

Loop Structures

For: `for (`*initialize*`;` *test*`;` *update*`)` `{`*statements*`}`

While: `while (`*condition*`)` `{`*statements*`}` ← *"entrance condition" loop*

`do {`*statements*`} while (`*condition*`)` ← *"exit condition" loop*

View:

- C reference card
- Java reference card

C Control Structures Example

```c
#include <stdio.h>

main()
{ int i, j;

  for (i=1; i<= 5; i++)
  { j = i*i;
    if ((j%2)==0)
       printf("%d is even\n", j);
    else
       printf("%d is odd\n", j);
  }

  return 0;
}
```

```
wmcb> gcc -o cs_eg cs_eg.c
wmcb> ./cs_eg
1 is odd
4 is even
9 is odd
16 is even
25 is odd
```

Java Control Structures Example

```
class cs_eg {
 public static void main(String[] args)
 {
 int i, j;

 for (i=1; i<= 5; i++)
 { j = i*i;
   if ((j % 2)==0)
       System.out.println(j+" is even");
   else
       System.out.println(j+" is odd");
 }

 }
}
```

```
wmcb> javac cs_eg.java
wmcb> java cs_eg
1 is odd
4 is even
9 is odd
16 is even
25 is odd
```

TI-84 Control Structures

Conditional Statements

If: `If` *condition*: *statement*

```
If condition
Then
statements
Else
statements
End
```

Case: N/A (*Use a piecewise function or nested* `if` *statements.*)

TI-84 Control Structures

Loop Structures

For:
```
For(index, start_value, end_value [, increment])
statements
End
```

While:

`While`* *condition*	`Repeat`** *condition*
statements	*statements*
`End`	`End`

* *Loop while the condition is true; test condition at the beginning*

** *Loop until the condition is true; test condition at the end*

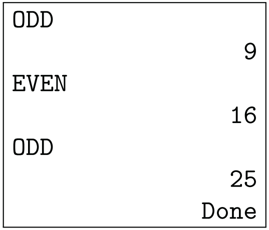
View:

- TI Calculator website
- TI-84 Guidebook links

TI-84 Control Structures Example

$\boxed{\texttt{PRGM}}$ ▶ NEW ▶ 1:Create New

```
PROGRAM
Name= CONTROL
:ClrHome
:For(J,1,5)
:J^2→K
:If gcd(K,2)=2
:Then
:Disp "EVEN",K
:Else
:Disp "ODD",K
:End
:End
```



```
ODD
                 9
EVEN
                16
ODD
                25
              Done
```

TI-84+ SE *Screen Capture*

R Control Structures

Conditional Statements

If: `if`(*condition*) {*statements*}

`if`(*condition*)
{*statements*}
`else`
{*statements*}

Case: `switch` (*index*, *list*)

R Control Structures

Loop Structures

For: `for` (*variable* `in` *sequence*)
{*statements*}

While: `while`[*] (*condition*)
{*statements*}

`repeat`[**]
{*statements*
`if` (*exit_condition*) `break`
statements}

[*] *Loop while the condition is true; test condition at the beginning*
[**] *Loop until the condition is true; test condition inside the loop*

View:

- The R Project for Statistical Computing homepage
- The Comprehensive R Archive Network — *CRAN*

R Control Structures Example

```
> for (j in 1:5){
+ k=j^2
+ if (k%%2==0) {
+  cat(k,"is even\n")}
+ else {
+  cat(k,"is odd\n")}
+ }
1 is odd
4 is even
9 is odd
16 is even
25 is odd
>
```

Python Control Structures

Conditional Statements

if	`if`(*condition*) {*statements*}
	`if`(*condition*) {*statements*} `else` {*statements*}
case:	`switch` (*index*, *list*)

View the Python website.

Python Control Structures Example

```
>>> for j in range(1, 6):
        k=j*j
        if (k%2) == 0:
            print(k, 'is even')
        else:
            print(k, 'is odd')

1 is odd
4 is even
9 is odd
16 is even
25 is odd
>>>
```

Note: Indentation is critical in Python.

From Code to a Flow Chart

Maple Loops

Build flow charts for the Maple code shown below:

```
▼ Algorithm 1.
  $n := 12;$
  $r := 1;$
  for $i$ from $2$ to $n$ do
      $r := r \cdot i;$
      end do:
  $r;$
```

```
▼ Algorithm 2.
  $n := 12;$
  $R := n;$
  $j := n - 1;$
  while $j > 1$ do
      $R := R \cdot j;$
      $j := j - 1;$
      end do:
  $R;$
```

What mathematical function are these routines calculating?

Exercises, I

Problems

1. Write an `If-Then-ElseIf` statement that calculates tax for a married couple filing jointly using the 2011 NC Tax Table (before the "Flat Tax").
   a. In natural language
   b. In Excel
   c. In Maple
   d. In MATLAB (Octave or Scilab)
   e. In C (Java, Python, or R)
   f. On a TI-84
2. Implement the Collatz Flow Chart
   a. In pseudocode
   b. In Maple (as a function)
3. Write code that, given a positive integer n, prints the first n primes.
4. Give a Maple version of Euler's method.
5. Write nested `for` loops that fill in the entries of an $n \times n$ Hilbert matrix
   a. In Maple
   b. On a TI-84
6. How can a `while` loop be redesigned as a `for` loop?
7. How can a `for` loop be redesigned as a `while` loop?

Exercises, II

Problems

8. Make a flow chart for implementing the Euclidean algorithm to find the GCD of two positive integers p and q.

9. Write code using the Euclidean algorithm to find the GCD of two positive integers p and q.

10. Write a Maple or MATLAB function that applies the Extended Euclidean algorithm to two positive integers p and q to give the greatest common divisor $\gcd(p,q)$ and to find integers a and b such that $a\,p + b\,q = \gcd(p,q)$.

11.
    a. Make a flow chart for the Maple code shown in the Flow Chart Problem worksheet.
    b. What does the code do?
    c. Convert the Maple statements to
        i. MATLAB
        ii. TI-84+

Exercises, III

Problems

12. The "9's-complement" of a number x is the value needed to add to x to have 9's. E.g., the 9's-complement of 3 is 6; of 64 is 35; etc.

    a. Write a statement to calculate the 9's-complement of an n-digit number y; call the result y_9.
    b. Write an `if-then` statement that performs *carry-around*: if the sum of two n-digits numbers has an $n+1$st carry digit, drop that digit and add 1 to the sum.
    c. Let $r > s$ be two n-digit integers. Find s_9 with a. Now perform carry-around on $(r + s_9)$ with b.
    d. What simple arithmetic operation is equivalent to the result of c?

13. The compass heading CH for going from $P_1 = (lat_1, lon_1)$ to $P_2 = (lat_2, lon_2)$ (other than the North or South poles) is given by

$$\mathrm{CH}(P_1, P_2) = \begin{cases} L & \cos^{-1}(\mathrm{lon}_2 - \mathrm{lon}_1) < 0 \\ 2\pi - L & \text{otherwise} \end{cases}$$

where $L = \cos^{-1}\left(\frac{\sin(\mathrm{lat}_2) - \sin(\mathrm{lat}_1)\cos(d)}{\sin(d)\cos(\mathrm{lat}_1)}\right)$ and
$d = \cos^{-1}(\sin(\mathrm{lat}_1)\sin(\mathrm{lat}_2) + \cos(\mathrm{lat}_1)\cos(\mathrm{lat}2)\cos(\mathrm{lon}_1 - \mathrm{lon}_2))$.

Quick Reference Cards

Quick Reference Card Collection

• Maple 15 (Mac) • Maple 15 (Win) • Maplesoft's Online help	• Excel 2011 (Mac) • Excel 2010 (Win) • Microsoft's Online help
• MATLAB • Scilab • Octave	• Wikiversity's Control Structures • C • Java
• TI-84 +: Algebra; Trigonometry	• R • Python 2.7, Python 3

S I. Special Topics: Computation Cost and Horner's Form

Sections

DOI: 10.1201/9781003299257-3

Special Topic: Computation Cost & Horner's Form

Introduction to Cost

The arithmetic *cost of computation* is a measure of how much mathematical work a particular expression takes to compute. We will measure an expression in terms of the number of arithmetic operations it requires. For example, we'll measure the cost of computing the expression

$$\sin(2x^4 + 3x + 1)$$

as

$$2 \textit{ additions} + 5 \textit{ multiplications} + 1 \textit{ function call}$$

for a total of 7 arithmetic operations plus a function call.

At a lower level, the time cost of a CPU instruction is the number of clock cycles taken to execute the instruction. Current CPUs[5] are measured in *FLoating-point OPerations per Second* or FLOPS. For example, the eight-core Intel® Core™ i9 processor used in an iMac (19/2019) can achieve over 235 gigaFLOPS $= 10^{11}$ floating-point operations per second.

[5] Currently computed as FLOP per cycle per core. Also see Moore's Law.

Horner's Form

Partial Factoring

William Horner studied solving algebraic equations and efficient forms for computation. Horner observed that *partial factoring* simplified a polynomial calculation. Consider:

$$\begin{array}{rcl}
\textit{Standard Form} & \Leftrightarrow & \textit{Horner's Form} \\
\underbrace{1+2x}_{1\,add+1\,mult} & = & \underbrace{1+2x}_{1\,add+1\,mult} \\
\underbrace{1+2x+3x^2}_{2\,add+3\,mult} & = & \underbrace{1+x\cdot(2+3x)}_{2\,add+2\,mult} \\
\underbrace{1+2x+3x^2+4x^3}_{3\,add+6\,mult} & = & \underbrace{1+x\cdot(2+x\cdot[3+4x])}_{3\,add+3\,mult} \\
\underbrace{1+2x+3x^2+4x^3+5x^4}_{4\,add+10\,mult} & = & \underbrace{1+x\cdot(2+x\cdot[3+x\cdot(4+5x)])}_{4\,add+4\,mult} \\
\underbrace{1+2x+3x^2+4x^3+5x^4+6x^5}_{5\,add+15\,mult} & = & \underbrace{1+x\cdot(2+x\cdot[3+x\cdot(4+x\cdot[5+6x])])}_{5\,add+5\,mult}
\end{array}$$

What are the patterns?

Patterns

Two Patterns

If $p(x)$ is an nth-degree polynomial, the cost of computation in standard form is $O(n^2)$. Using Horner's form reduces the cost to $O(n)$.

EXAMPLE: Let $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6$.

Horner's form: $p(x) = a_0 + x(a_1 + x[a_2 + x(a_3 + x[a_4 + x(a_5 + a_6x)])])$.

This factored form significantly reduces the work needed to evaluate p at a given value of x.

Modified: $?(x) = a_1 + x[2a_2 + x(3a_3 + x[4a_4 + x(5a_5 + 6a_6x)])]$.

- What does this modification calculate in terms of p?
- What is the cost of this modification versus using its standard form?

Further Reductions

Chebyshev's Polynomials

Pafnuty L. Chebyshev worked in number theory, approximation theory, and statistics. The special polynomials named for him are the Chebyshev Polynomials $T_n(x)$, which have many interesting properties. For example, T_n is even or odd with n, oscillates between -1 and 1 on the interval $[-1,1]$, and also has all its zeros in $[-1,1]$. The Horner form of T_n is quite interesting. Let $u = x^2$, then:

$$\begin{aligned}
-3x+4x^3 &\iff x(-3+4x^2) = x(-3+4u) \\
1-8x^2+8x^4 &\iff 1+u(-8+8u) \\
5x-20x^3+16x^5 &\iff x(5+u[-20+16u]) \\
-1+18x^2-48x^4+32x^6 &\iff -1+u(18+u[-48+32u]) \\
-7x+56x^3-112x^5+64x^7 &\iff \quad ? \\
1-32x^2+160x^4-256x^6+128x^8 &\iff \quad ?
\end{aligned}$$

Exercises, I

Problems

1. Make a flow chart for evaluating a polynomial using Horner's form.
2. Write Maple or MATLAB code implementing Horner's form.
3. How does *synthetic division* relate to Horner's form?
4. Write a Maple or MATLAB function that performs synthetic division with a given polynomial at a given value.
5. Calculate the number of additions and multiplications required for evaluating an nth-degree polynomial
   a. in standard form.
   b. in Horner's form.
   c. Look up the sequence $\{0, 2, 5, 9, 14, 20, 27, 35, 44, \ldots\}$ at The On-Line Encyclopedia of Integer Sequences.
6. Prove that Horner's form reduces cost from $O(n^2)$ to $O(n)$.
7. Analyze the reduction of cost when using Horner's form to evaluate Chebyshev polynomials.

III. Numerical Differentiation

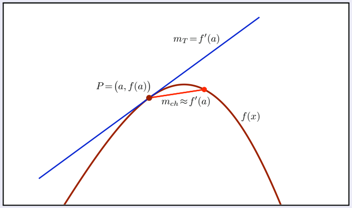
Sections

DOI: 10.1201/9781003299257-4

III. Numerical Differentiation

What is *Numerical Differentiation?*

Numerical Differentiation is the approximation of the derivative of a function at a point using numerical formulas, not the algebraic rules for differentiation. The basic form uses the slope of a short chord rather than the tangent line.



Since we are subtracting numbers that are close together, loss of computational precision can be a serious problem.

Taylor series expansions will be our basic tool for developing formulas and error bounds for numerical derivatives. The errors will have two main components: truncation errors from Taylor polynomials and round-off errors from finite-precision floating-point arithmetic.

Taylor's Theorem

Definition (Taylor Polynomials (1712/1715[6]))

If f has sufficiently many derivatives at $x=a$, the *Taylor polynomial of degree n (or order n)* is

$$p_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

where $f^{(0)}(a) = f(a)$.

Theorem (Taylor's Theorem)

Suppose f has $n+1$ derivatives on a neighborhood of a. Then $f(x) = p_n(x) + R_n(x)$ where

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{n+1}$$

for some c between x and a.

[6] Actually, discovered by Gregory in 1671, ≈ 14 *years before Taylor was born!*

Proving Taylor's Theorem

Proof. (Taylor's Theorem — *Outline*).

1. The FToC $\Rightarrow f(x) = f(a) + \int_0^{x-a} f'(x-t)\,dt.$
2. Integrate by parts with $u = f'(x-t)$ and $dv = dt$:

$$f(x) = f(a) + f'(a)(x-a) + \int_0^{x-a} f''(x-t)\cdot t\,dt.$$

3. Repeat the process: choose $u = f^{(k)}(x-t)$ and $dv = t^{k-1}/(k-1)!$ to arrive at

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(a,x)$$

where

$$R_n(a,x) = \frac{1}{n!}\int_0^{x-a} f^{(n+1)}(x-t)\cdot t^n\,dt.$$

□

Tailored Expressions

Forms of the Remainder

Lagrange (1797): $R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{n+1}$

Cauchy (1821): $R_n(x) = \frac{f^{(n+1)}(c)}{n!}(x-c)^n(x-a)$

Integral Form: $R_n(x) = \frac{1}{n!}\int_a^x f^{(n+1)}(t)(x-t)^n\,dt$

Uniform Form: $|R_n(x)| \le \frac{|x-a|^{n+1}}{(n+1)!} \cdot \max\left|f^{(n+1)}(x)\right| = O\left(|x-a|^{n+1}\right)$

Two Useful Taylor Expansions

Set $x = a+h$ in the Taylor polynomial. Then

$$f(a+h) = f(a) + f'(a)\cdot h + \frac{1}{2!}f''(a)\cdot h^2 + \frac{1}{3!}f'''(a)\cdot h^3 + \cdots \tag{1}$$

And now set $x = a-h$. Then

$$f(a-h) = f(a) - f'(a)\cdot h + \frac{1}{2!}f''(a)\cdot h^2 - \frac{1}{3!}f'''(a)\cdot h^3 \pm \cdots \tag{2}$$

Forward Difference Approximation

Forward Difference

Subtract $f(a)$ from both sides of Eq (1), then divide by h to obtain

$$\frac{f(a+h)-f(a)}{h} = f'(a) + \frac{O(h^2)}{h}.$$

The Forward Difference Formula is

$$f'(a) = \frac{f(a+h)-f(a)}{h} + O(h). \tag{FD}$$

Examples

1. Suppose $f(x) = 1 + x\,e^{\sin(x)}$. For $a = 0$ and $h = 0.1$, we have
$$f'(x) \approx (1.1105 - 1.0000)/0.1 = 1.1050.$$
2. Suppose $P_0 = (1.000, 3.320)$ and $P_1 = (1.100, 3.682)$. Then
$$f'(x) \approx (3.682 - 3.320)/(1.100 - 1.000) = 3.620.$$

Backward Difference Approximation

Backward Difference

Subtract $f(a)$ from both sides of Eq (2), then divide by h to obtain

$$\frac{f(a-h)-f(a)}{h} = -f'(a) + \frac{O(h^2)}{h}.$$

The Backward Difference Formula is

$$f'(a) = \frac{f(a)-f(a-h)}{h} + O(h). \tag{BD}$$

Examples

1. Again, suppose $f(x) = 1 + x\,e^{\sin(x)}$. For $a = 0$ and $h = 0.1$, we have
$$f'(x) \approx (1.0000 - 0.910)/0.1 = 0.900.$$

2. Suppose $P_0 = (1.000, 3.320)$ and $P_1 = (0.900, 2.970)$. Then
$$f'(x) \approx (3.320 - 2.970)/(1.000 - 0.900) = 3.500.$$

Centered Difference Approximation

Centered Difference

Subtract $O(h^3)$ versions of Eqs (1) and (2).

$$\begin{array}{r}
f(a+h) = f(a) + f'(a)\cdot h + \frac{1}{2!}f''(a)\cdot h^2 + O\big(h^3\big) \\
f(a-h) = f(a) - f'(a)\cdot h + \frac{1}{2!}f''(a)\cdot h^2 + O\big(h^3\big) \\
\hline
f(a+h) - f(a-h) = 2f'(a)\cdot h + O(h^3) \qquad\qquad\qquad\qquad\qquad
\end{array}$$

Solve for $f'(a)$ to obtain:
The Centered Difference Formula

$$f'(a) = \frac{f(a+h) - f(a-h)}{2h} + O(h^2). \tag{CD}$$

Example

1. Once more, suppose $f(x) = 1 + x\,e^{\sin(x)}$. For $a = 0$ and $h = 0.1$, we have
$$f'(x) \approx (1.110 - 0.910)/0.2 = 1.000.$$

The Chart

A Table of Differences from a Function

Let $f(x) = 1 + xe^{\sin(x)}$ and $a = 1.0$. Then

$$f'(1) = e^{\sin(1)}\,(1+\cos(1)) \approx 3.573157593.$$

h	FD(h)	*error*	BD(h)	*error*	CD(h)	*error*
$1/2^1$	3.494890	0.078268	3.024408	0.548750	3.259649	0.313509
$1/2^2$	3.636316	0.063158	3.347764	0.225394	3.492040	0.081118
$1/2^3$	3.628464	0.055306	3.476944	0.096214	3.552704	0.020454
$1/2^4$	3.606368	0.033210	3.529696	0.043462	3.568032	0.005126
$1/2^5$	3.591104	0.017946	3.552640	0.020518	3.571872	0.001286
$1/2^6$	3.582464	0.009306	3.563200	0.009958	3.572832	0.000326
$1/2^7$	3.577600	0.004442	3.568256	0.004902	3.572928	0.000230
$1/2^8$	3.575296	0.002138	3.570688	0.002470	3.572992	0.000166

Another Chart

A Table of Differences from Data

Estimate the derivatives of a function given the data below ($h = 0.4$).

x_i	-2.00	-1.60	-1.20	-0.80	-0.40	0.00	0.40	0.80	1.20	1.60	2.00
y_i	-1.95	-0.29	0.56	0.81	0.65	0.30	-0.06	-0.21	0.04	0.89	2.55

Forward Differences

x_i	-2.00	-1.60	-1.20	-0.80	-0.40	0.00	0.40	0.80	1.20	1.60	2.00
y_i	4.16	2.12	0.625	-0.375	-0.90	-0.90	-0.375	0.625	2.12	4.16	$\bowtie$

Backward Differences

x_i	-2.00	-1.60	-1.20	-0.80	-0.40	0.00	0.40	0.80	1.20	1.60	2.00
y_i	$\bowtie$	4.16	2.12	0.625	-0.375	-0.90	-0.90	-0.375	0.625	2.12	4.16

Centered Differences

x_i	-2.00	-1.60	-1.20	-0.80	-0.40	0.00	0.40	0.80	1.20	1.60	2.00
y_i	$\bowtie$	3.138	1.374	0.125	-0.637	-0.90	-0.6375	0.125	1.374	3.138	$\bowtie$

Actual Derivatives (from the function's formula — a cubic polynomial)

x_i	-2.00	-1.60	-1.20	-0.80	-0.40	0.00	0.40	0.80	1.20	1.60	2.00
y_i	5.325	3.057	1.293	0.033	-0.723	-0.975	-0.723	0.033	1.293	3.057	5.325

Appendix I: Taylor's Theorem

Methodus Incrementorum Directa et Inversa (1715)[1]

In 1712, Taylor wrote a letter containing his theorem without proof to Machin. The theorem appears with proof in *Methodus Incrementorum* as Corollary II to Proposition VII. The proposition is a restatement of "Newton's [interpolation] Formula." Maclaurin introduced the method (*undet coeffs; order of contact*) we use now to present Taylor's theorem in elementary calculus classes in *A Treatise of Fluxions* (1742) §751.

Corollary (Maclaurin's Corollary II (pg 23))

If for the evanescent increments, the fluxions that are proportional to them are written, the quantities $\overset{\prime\prime}{v}$, $\overset{\prime}{v}$, v, $\underset{\prime}{v}$, $\underset{\prime\prime}{v}$, *&c. being now made all equal to the time* z *uniformly flows to become* $z+v$, *then* x *will become*

$$x+\dot{x}\,\frac{v}{1\dot{z}}+\ddot{x}\,\frac{v^2}{1\,.\,2\dot{z}^2}+\overset{\therefore}{x}\,\frac{v^3}{1\,.\,2\,.\,3\dot{z}^3}+\ \&c.$$

- *Investigate the connection between synthetic division and Taylor expansions.*

[1]See Ian Bruce's annotated translation.

Appendix II: Centered Difference Coefficients Chart

Centered Finite Difference Formula Coefficients[1]

Derivative	$O(h^a)$	$x-4h$	$x-3h$	$x-2h$	$x-h$	x	$x+h$	$x+2h$	$x+3h$	$x+4h$
1	2				$-1/2$	0	$1/2$			
	4			$1/12$	$-2/3$	0	$2/3$	$-1/12$		
	6		$-1/60$	$3/20$	$-3/4$	0	$3/4$	$-3/20$	$1/60$	
	8	$1/280$	$-4/105$	$1/5$	$-4/5$	0	$4/5$	$-1/5$	$4/105$	$-1/280$
2	2				1	-2	1			
	4			$-1/12$	$4/3$	$-5/2$	$4/3$	$-1/12$		
	6		$1/90$	$-3/20$	$3/2$	$-49/18$	$3/2$	$-3/20$	$1/90$	
	8	$-1/560$	$8/315$	$-1/5$	$8/5$	$-205/72$	$8/5$	$-1/5$	$8/315$	$-1/560$
3	2			$-1/2$	1	0	-1	$1/2$		
	4		$1/8$	-1	$13/8$	0	$-13/8$	1	$-1/8$	
	6	$-7/240$	$3/10$	$-169/120$	$61/30$	0	$-61/30$	$169/120$	$-3/10$	$7/240$
4	2			1	-4	6	-4	1		
	4		$-1/6$	2	$-13/2$	$28/3$	$-13/2$	2	$-1/6$	
	6	$7/240$	$-2/5$	$169/60$	$-122/15$	$91/8$	$-122/15$	$169/60$	$-2/5$	$7/240$

E.g., the third derivative's centered difference approximation with second-order accuracy is

$$f^{(3)}(x_0) \approx \frac{-\frac{1}{2}f(x_0-2h)+1f(x_0-h)+0f(x_0)-1f(x_0+h)+\frac{1}{2}f(x_0+2h)}{h^3}+O\left(h^2\right).$$

[1]See Fornberg, "Generation of Finite Difference Formulas on Arbitrarily Spaced Grids," *Math of Comp* **51** (184), pp 699–706.

Exercises, I

Problems

1. Show that the centered difference formula is the average of the forward and backward difference formulas.
2. Explain why the centered difference formula is $O(h^2)$ rather than $O(h)$.
3. Add $O(h^4)$ versions of Eqs (1) and (2) to find a centered difference approximation of $f''(a)$.
4. Investigate the ratio of error in the function's difference chart as h is successively divided by 2 for
    a. forward differences
    b. backward differences
    c. centered differences
5. Examine the ratios of error to h in the data difference chart for
    a. forward differences
    b. backward differences
    c. centered differences

Exercises, II

Problems

6. Derive the 5-*point difference formula* for $f'(a)$ by combining Taylor expansions to $O(h^5)$ for $f(a \pm h)$ and $f(a \pm 2h)$.
7. Write a Maple or MATLAB function that uses the backward difference formula (BD) in Euler's method of solving differential equations.
8. Collect the temperatures (with a CBL) in a classroom from 8:00 am to 6:00 pm.
   a. Estimate the rate of change of temperatures during the day.
   b. Compare plots of the rates given by forward, backward, and centered differences.
9. a. Find Taylor expansions for $\sin$ and $\cos$ to $O(x^6)$. Estimate $\cos(1.0)$.
   b. Since $\frac{d}{dx}\sin(x) = \cos(x)$, we can estimate $\cos$ with the derivative of $\sin$. Use your expansion of $\sin$ and $h = 0.05$ to approximate $\cos(1.0)$ with
      i. forward differences
      ii. backward differences
      iii. centered differences

   Discuss the errors.

IV. Root-Finding Algorithms

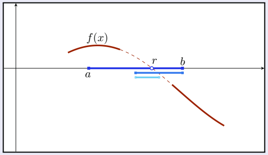
Sections

DOI: 10.1201/9781003299257-5

IV. Root Finding Algorithms: The Bisection Method

The Bisection Method

If a continuous function f has a root r in an interval, then r is in either the interval's left half or right half. Suppose r is in the right half interval. Then r must be in either this smaller interval's left half or right half. Find which half and continue the procedure.



This process depends on the *Intermediate Value Theorem*

Theorem (Bolzano's Intermediate Value Theorem (1817))

Let f be continuous on $[a,b]$. Suppose that y^* is between $f(a)$ and $f(b)$.
Then there is a point $c \in (a,b)$ such that $f(c) = y^*$.
In particular, if $f(a) \cdot f(b) < 0$, then f has a root r with $a < r < b$.

The Bisection Error

Theorem (Bisection Algorithm)

Let $[a,b]$ *be an interval on which* f *changes sign. Define*

$$x_n = c_n = \frac{1}{2}(a_{n-1} + b_{n-1})$$

with $[a_n, b_n]$ *chosen by the algorithm. Then* f *has a root* $\alpha \in [a,b]$, *and*

$$|\alpha - x_n| \le (b-a) \cdot \left(\frac{1}{2}\right)^n.$$

For an error tolerance $\varepsilon > 0$, *set*

$$n = \left\lceil \frac{\log(b-a) - \log(\varepsilon)}{\log(2)} \right\rceil$$

to obtain $|\alpha - x_n| \le \varepsilon$. *(This is called* linear convergence.*)*

Theorem (Cauchy's Bound for Real Roots[1] (1829))

Suppose that r *is a root of* $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$. *Let* $M = \max_{k=0..n-1} |a_k|$. *Then* $|r| \le M + 1$.

[1] A-L Cauchy, *Exercices de mathématiques*, De Bure frères, Paris (1829).

The Bisection Method Algorithm

Algorithm (Bisection Method (Basic Outline))

Given f and $[a,b]$.

```
1. Set $k = 0$ and $[a_0, b_0] = [a, b]$.
2. Calculate $c = \frac{1}{2}(a_k + b_k)$
3. if $f(c) \approx 0$, then $c$ is a root; quit
4. if $f(c) \cdot f(a_k) < 0$, then set $[a_{k+1}, b_{k+1}] = [a_k, c]$
5. else if $f(c) \cdot f(b_k) < 0$, then set $[a_{k+1}, b_{k+1}] = [c, b_k]$
6. Set $k = k + 1$ and (if $k$ isn't too big) go to 2.
```

The Bisection Method Pseudocode

```
input  : a, b, eps
extern: f(x)
fa ← f(a);
fb ← f(b);
if $fa \cdot fb > 0$ then
    stop ;                                        /* Better:  sign(fa) ≠ sign(fb) */
n ← ceiling((log(b − a) − log(eps))/log(2));
for $i \leftarrow 1$ to $n$ do
    c ← a + 0.5·(b − a);
    fc ← f(c);
    if abs(fc) < eps then
        return: c
    if $fa \cdot fc < 0$ then
        b ← c;
        fb ← fc;
    else
        if $fa \cdot fc > 0$ then
            a ← c;
            fa ← fc;
        else
            return: c
return: c
```

Newton-Raphson Method

Newton-Raphson Method[2]

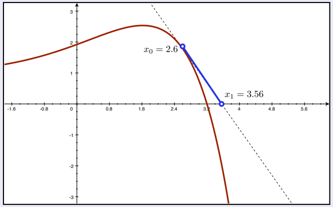
If a function f is "nice," use the tangent line to approximate f. The root of the tangent line — easy to find — approximates the root of f.

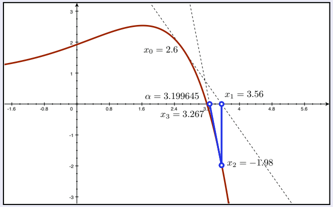
1. $f(x) = f(a) + f'(a)(x - a)$
2. Set $f(x) = 0$; solve for x:

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

3. Set $N(x) = x - \frac{f(x)}{f'(x)}$. Then

$$x_{n+1} = N(x_n)$$





[2]The general method we use was actually developed by Simpson; Newton worked with polynomials; Raphson iterated the formula to improve the estimate of the root.

Newton's Method Error

Theorem

Let $f \in \mathscr{C}^2(I)$ on some interval $I \subset \mathbb{R}$. Suppose $\alpha \in I$ is a root of f. Choose $x_0 \in I$ and define

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

Then

$$|\alpha - x_{n+1}| \leq M \cdot |\alpha - x_n|^2$$

or, with $\varepsilon_n = |\alpha - x_n|$,

$$\varepsilon_{n+1} \leq M \cdot \varepsilon_n^2$$

where M is an upper bound for $\frac{1}{2}|f''(x)/f'(x)|$ on I.

This is called quadratic or "order 2" convergence.

Newton's Method Algorithm

Algorithm (Newton's Method (Basic Outline))

Given f and x_0.

```
1. Set $k = 1$ and $N(x) = x - \frac{f(x)}{f'(x)}$.
2. Compute $x_k = N(x_{k-1})$.
3. If $f(x_k) \approx 0$, then $x_k$ is a root; quit
4. else if $|f(x_k)|$ or $|x_k - x_{k-1}|$ is very small, then $x_k \approx$ a root; quit
5. Set $k = k + 1$ and (if $k$ isn't too big) go to 2.
```

Newton's Method Pseudocode

```
input  : x0, eps, n
extern: f(x), df(x)=f′(x)

fx0 ← f(x0);
dfx0 ← df(x0);
for $i \leftarrow 1$ to $n$ do
    x1 ← x0 − fx0/dfx0;
    fx1 ← f(x1);
    if $|fx1| + |x1 - x0| < eps$ then
        return: x1
    x0 ← x1;
    fx0 ← fx1;
    dfx0 ← df(x0);
return: x1
```

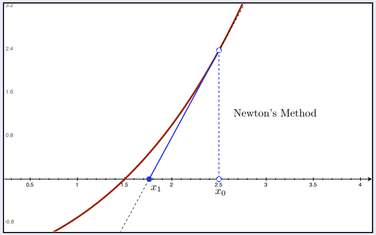
Secant Method

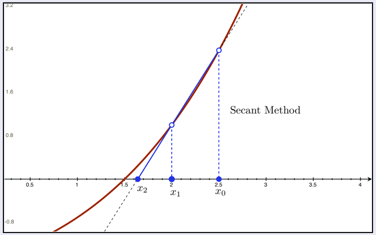
Secant Method

Newton's method requires evaluating the derivative — this can be from difficult to impossible in practice. Approximate the derivative in Newton's method with a secant line[3]:

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

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





[3]Historically, the methods developed the opposite way: Viète used discrete steps of 10^{-k} (1600); Newton used secants (1669), then truncated power series (1687); Simpson used fluxions/derivatives (1740) with general functions.

Secant Method Error

Theorem

Let $f \in \mathscr{C}^2(I)$ for some interval $I \subset \mathbb{R}$. Suppose $\alpha \in I$ is a root of f. Choose x_0 and $x_1 \in I$, and define

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

Then

$$|\alpha - x_{n+1}| \leq M \cdot |\alpha - x_n| \cdot |\alpha - x_{n-1}|$$

or, with $\varepsilon_n = |\alpha - x_n|$,

$$\varepsilon_{n+1} \leq M \cdot \varepsilon_n \cdot \varepsilon_{n-1}$$

where M is an upper bound for $\frac{1}{2}|f''(x)/f'(x)|$ on I.

This is superlinear convergence of "order 1.6." (Actually, it's order $\frac{1+\sqrt{5}}{2}$.)

Secant Method Algorithm

Algorithm (Secant Method (Basic Outline))

Given f, x_0, and x_1:

```
1. Set $k = 2$ and $S(x_0, x_1) = x_1 - f(x_1) \cdot \frac{x_1 - x_0}{f(x_1) - f(x_0)}$.
2. Compute $x_k = S(x_{k-1}, x_{k-2})$.
3. If $f(x_k) \approx 0$, then $x_k$ is a root; then quit
4. else if $|x_k - x_{k-1}|$ is very small, then $x_k \approx$ a root; quit
5. Set $k = k + 1$ and (if $k$ isn't too big) go to 2.
```

Secant Method Pseudocode

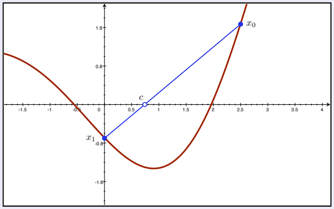
```
input  : x0, x1, eps, n
extern: f(x)
f0 ← f(x0);
f1 ← f(x1);
for $i \leftarrow 1$ to $n$ do
    c ← x1 − f1·(x1−x0) / (f1−f0);
    fc ← f(c);
    x0 ← x1;                                  /* update parameters */
    x1 ← c;
    f0 ← f1;
    f1 ← fc;
    if $|x1 - x0| < eps$ then
        return: x1 ;                          /* Or:  |fc|< eps */
return: x1
```

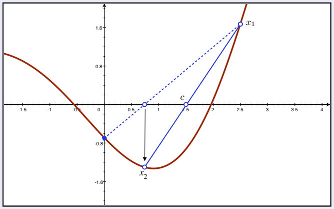
Regula Falsi

Regula Falsi

The *regula falsi*, or "false position," method[4] is very old; the Egyptians used the concept. The method appears in the *Vaishali Ganit* (India, 3rd century BC), *Book on Numbers and Computation* & *Nine Chapters on the Mathematical Art* (China, 2nd century BC), *Book of the Two Errors* (Persia, c. 900), and came to the west in Fibonacci's *Liber Abaci* (1202).

Regula falsi combines the secant and bisection techniques: Use the secant to find a "middle point," then keep the interval with a sign change, i.e., that brackets the root.





[4]Also called *Modified Regula Falsi*, *Double False Position*, *Regula Positionum*, *Secant Method*, *Rule of Two Errors*, etc. My favourite name is *Yíng bù zú: "Too much and not enough."*

Regula Falsi Method Error

Theorem

Let $f \in \mathscr{C}^2(I)$ for some interval $I \subset \mathbb{R}$. Suppose $\alpha \in I$ is a root of f. Choose a and $b \in I$ such that $\operatorname{sign}(f(a)) \neq \operatorname{sign}(f(b))$, and define

$$c = b - f(b) \cdot \frac{b-a}{f(b) - f(a)}.$$

Then

$$|\alpha - c| \leq M \cdot |\alpha - a|$$

or, with $\varepsilon_n = |\alpha - x_n|$,

$$\varepsilon_{n+1} \leq M \cdot \varepsilon_n$$

where $0 < M < 1$ is a constant depending on $|f''(x)|$ and $|f'(x)|$ on I.

This is linear or "order 1" convergence. (The same as the bisection method.)

Regula Falsi Algorithm

Algorithm (Regula Falsi (Basic Method))

Given f, a, and b:

1. *Set $k = 1$ and* $S(a,b) = b - f(b) \cdot \dfrac{b-a}{f(b)-f(a)} \qquad = \dfrac{a \cdot f(b) - b \cdot f(a)}{f(b)-f(a)}$
2. *Compute $c = S(a,b)$.*
3. *If $f(c) \approx 0$, then c is a root; quit*
4. *If $f(c) \cdot f(a) < 0$, then $b \leftarrow c$*
5. *else $a \leftarrow c$*
6. *If $|b-a|$ is very small compared to $|a|$, then a is a root; quit*
7. *Set $k = k+1$, and (if k isn't too big) go to 2.*

Regula Falsi Pseudocode

```
input  : a, b, eps, n
extern: f(x)
1:  fa ← f(a);
2:  fb ← f(b);
3:  if $fa \cdot fb > 0$ then
4:      stop ;                                    /* Better:  sign(fa) ≠ sign(fb) */
5:  for $i \leftarrow 1$ to $n$ do
6:      c ← (a·fb − b·fa)/(fb − fa) ;              /* Better:  c ← b - fb*(b-a)/(fb-fa) */
7:      fc ← f(c);
8:      if $|fc| < eps$ then
            return: c
9:      if $fa \cdot fc < 0$ then
10:         b ← c;
11:         fb ← fc;
12:     else
13:         if $fb \cdot fc < 0$ then
14:             a ← c;
15:             fa ← fc;
16:         else
                return: c
return: c
```

Halley's Method

Halley's Method

Halley's method[5] of 1694 extends Newton's method to obtain cubic convergence. Halley was motivated by de Lagny's 1692 work showing algebraic formulas for extracting roots. Using the quadratic term in Taylor's formula obtains the extra degree of convergence. Just as Newton did not recognize the derivative appearing in his iterative method, Halley also missed the connection to calculus — it was first seen by Taylor in 1712. The version of Halley's method we use comes from applying Newton's method to the auxiliary function

$$g(x) = \frac{f(x)}{\sqrt{|f'(x)|}}.$$

Then the iterating function for Halley's method is

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

[5]For historical background and a nice development, see Scavo and Thoo, "On the Geometry of Halley's Method," *Am. Math. Mo.*, Vol. 102, No. 5, pp. 417–426.

Halley's Method Error

Theorem

Let $f \in \mathscr{C}^3(I)$ on some interval $I \subset \mathbb{R}$. Suppose $\alpha \in I$ is a root of f. Choose $x_0 \in I$ and define

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

Then

$$|\alpha - x_{n+1}| \le M \cdot |\alpha - x_n|^3$$

or, with $\varepsilon_n = |\alpha - x_n|$,

$$\varepsilon_{n+1} \le M \cdot \varepsilon_n^3$$

where M is an upper bound for $\left|\frac{3f''(x)^2 - 2f'(x)f'''(x)}{12f'(x)^2}\right|$ on I.

This is called cubic or "order 3" convergence.

Halley's Method Algorithm

Algorithm (Halley's Method (Basic Outline))

Given f and x_0.

0. *Compute f' and f''.*
1. *Set $k = 1$ and* $H(x) = x - \dfrac{2f(x)f'(x)}{2f'(x)^2 - f(x)f''(x)}$.
2. *Compute $x_k = H(x_{k-1})$.*
3. *If $f(x_k) \approx 0$, then x_k is a root; quit*
4. *else if $|f(x_k)|$ or $|x_k - x_{k-1}|$ is very small, then $x_k \approx$ a root; quit*
5. *Set $k = k+1$ and (if k isn't too big) go to 2.*

Halley's Method Pseudocode

```
input  : x0, eps, n
extern: f(x), df(x)=f'(x), ddf(x)=f''(x)

fx0 ← f(x0);
dfx0 ← df(x0);
ddfx0 ← ddf(x0);
for $i \leftarrow 1$ to $n$ do
    x1 ← x0 − 2·fx0·dfx0 / (2·dfx0^2 − fx0·ddfxo);
    fx1 ← f(x1);
    if $|fx1| + |x1 - x0| < eps$ then
        return: x1
    x0 ← x1;
    fx0 ← f(x1);
    dfx0 ← df(x1);
    ddfx0 ← ddf(x1);
return: x1
```

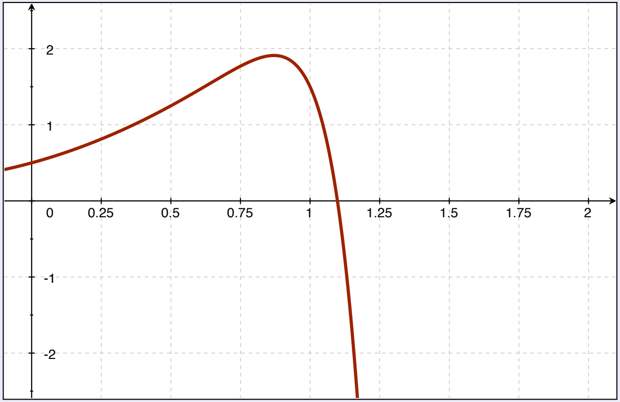
A Sample Problem

Polynomial Root

Find the real root of $f(x) = -x^{11} + x^2 + x + 0.5$ in $\left[\frac{1}{2}, \frac{3}{2}\right]$. $(r \doteq 1.098282972)$
(This is f's only real root.)

Bisection	Newton		Secant	Regula Falsi	Halley	
[0.5000, 1.5000]	0.5000	1.500	[+0.500, +1.500]	[0.500, 1.5]	0.5000	1.500
[1.0000, 1.5000]	−0.1280	1.370	[+0.515, +0.500]	[0.515, 1.5]	−0.3752	1.268
[1.0000, 1.2500]	−0.6500	1.258	[−0.124, +0.515]	[0.530, 1.5]	−0.0538	1.128
[1.0000, 1.1250]	−0.0236	1.171	[−0.406, −0.124]	[0.545, 1.5]	−1.2080	1.099
[1.0625, 1.1250]	−0.5241	1.119	[−0.956, −0.406]	[0.561, 1.5]	−0.9812	1.098
[1.0938, 1.1250]	3.315	1.100	[−0.230, −0.956]	[0.576, 1.5]	−0.6556	1.098
[1.0938, 1.1094]	3.014	1.098	[+0.085, −0.230]	[0.592, 1.5]	−0.9822	·
[1.0938, 1.1016]	2.740	1.098	[−0.607, +0.085]	[0.607, 1.5]	−0.6588	·
[1.0977, 1.1016]	2.491	·	[−1.171, −0.607]	[0.623, 1.5]	−0.9934	·
[1.0977, 1.0996]	2.264	·	[−0.583, −1.170]	[0.639, 1.5]	−0.6846	·
[1.0977, 1.0986]	2.059	·	[−0.558, −0.583]	[0.654, 1.5]	−1.085	·

The Sample Problem's Graph



A plot of $f(x) = -x^{11} + x^2 + x + 0.5$

Þ^e Chartes

Method	*Type*	*Update Function*
Bisection	Bracketing (2 pts)	$B(a,b) = \frac{1}{2}(a+b)$
Regula Falsi	Bracketing (2 pts)	$R(a,b) = \frac{a\,f(b) - b\,f(a)}{f(b) - f(a)}$
Secant method	Approximating (1 pt)	$S(x_n, x_{n-1}) = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$
Newton's method	Approximating (1 pt)	$N(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$
Halley's method	Approximating (1 pt)	$H(x_n) = x_n - \frac{2f(x_n)f'(x_n)}{2f'(x_n)^2 - f(x_n)f''(x_n)}$

Method	*Error*	*Convergence Speed*	*Computation Cost*
Bisection	$\varepsilon_{n+1} \leq \frac{1}{2}\varepsilon_n$	Linear (order 1)	Low
Regula Falsi	$\varepsilon_{n+1} \leq C\varepsilon_n$	Linear (order 1)	Medium
Secant method	$\varepsilon_{n+1} \leq C\varepsilon_n\varepsilon_{n-1}$	Superlinear (order ≈ 1.6)	Medium
Newton's method	$\varepsilon_{n+1} \leq C\varepsilon_n^2$	Quadratic (order 2)	High
Halley's method	$\varepsilon_{n+1} \leq C\varepsilon_n^3$	Cubic (order 3)	Very High

Appendix III: Rate of Convergence

Definition (Rate of Convergence)

Let $x_n \to x^*$ and set $\varepsilon_n = |x^* - x_n|$. Then x_n *converges to* x^* *with rate* r iff there is a positive constant C (*the asymptotic error constant*) such that

$$\lim_{n\to\infty} \frac{\varepsilon_{n+1}}{\varepsilon_n^r} = C.$$

Terminology

Rate	*Parameters*
Sublinear	$r = 1$ and $C = 1$
Linear	$r = 1$ and $0 < C < 1$
Superlinear	$r > 1$
Quadratic	$r = 2$
Cubic	$r = 3$

NB: Quadratic and cubic are special cases of superlinear convergence.

Exercises, I

Exercises

For each of the functions given in 1. to 7. below:

a. *Graph f in a relevant window.*

b. *Use Maple's `fsolve` to find f's root to 10 digits.*

c. *Use each of the five methods with a maximum of 15 steps and fill the table:*

Method	Approx Root	Relative Error	No. of Steps

1. *The "Newton-era" test function* $T(x) = x^3 - 2x - 5$.
2. $f(x) = \frac{1}{13} - \frac{1}{7}x - x^{11}$
3. $g(x) = \int_0^x \sin(t^2/2)\,dt - 1$
4. $h(x) = x - 8e^{-x}$
5. $R(x) = \dfrac{30x - 31}{29(x-1)}$
6. $S(x) = \dfrac{\sin(x^2)+1}{\cos(x)+2}$ *for* $x \in [0,4]$
7. *The intransigent function* $\psi(x) = 10 \cdot e^{-x} \cdot \left(1 + \frac{\ln(x^2)}{x}\right)$.
8. *Explain why the bisection method has difficulties with two roots in an interval.*

Exercises, II

Exercises

For Exercises 9. to 15., generate your *personal polynomial* $p(x)$ in Maple by entering:

```
> randomize(your phone number, no dashes or spaces):
  deg := 1+2*rand(4..9)():
  p := randpoly(x, degree=deg, coeffs=rand(-2..2)):
  p := unapply(sort(p), x);
```

9. Use `fsolve` to find the roots of your polynomial $p(x)$.
10. Compare the results of the four root-finding methods applied to $p(x)$.
11. Report on *stopping conditions*: function value, step/interval size, maximum number of iterations.
12. Find any bad initial points for Newton's method for $p(x)$.
13. Verify Fibonacci's root[6] $x = 1.368808107$ of the equation $x^3 + 2x^2 + 10x = 20$.
14. Determine the convergence rates of the following sequences.
    a. 2^{-n} b. $1 + 2^{(1-2n)}$ c. $(n+1)/n$ d. $\sin(k)/k$
15. Solve this problem from *The Nine Chapters on the Mathematical Art* (c. 200 BCE): "Now an item is purchased jointly; everyone contributes 8 coins, the excess is 3; everyone contributes 7, the deficit is 4. Tell: The number of people, the item price, what is each?"

[6] Fibonacci found this root in 1225 — *no one knows how he did it!*

Exercises, III

Exercises

16. Set $f(x) = x^3 \cdot e^x - \sqrt{2}$ and $x_0 = 1.0$.
    a. Plot f over $0 \leq x \leq 1$ to see the root.
    b. Use Newton's method to find the root; print x_n for each iteration.
    c. Alter Newton's method:
        i. Replace $f'(x)$ in Newton's method with a *centered difference formula* approximation Eq (CD).
        ii. Execute the altered Newton's method beginning with $h = 1.0$.
        iii. Decrease h by 0.1 and rerun the altered Newton's method.
        iv. Keep decreasing h until the results are very close to Newton's method.
        v. Determine practical reasons for using this modification rather than the original method.

17. [Group Exercise] Redo the previous problem finding all roots in $[-1, 1]$ of the 8th Chebyshev polynomial

$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1.$$

Links and Others

Wikipedia entries:

- Bisection Method
- Newton's Method
- Secant Method
- Regula Falsi Method

More information:

- Massoud Malek's Root-Finding Methods
- Chad Higdon-Topaz's Iterative methods for root finding (video)

See also: Interactive Educational Modules in Scientific Computing (U of I) and MathWorld (Wolfram Research)

Investigate:

- Müller's method
- Brent's method
- Bernoulli's method
- Jenkins-Traub method
- Laguerre's method
- Durand-Kerner method
- Fibonacci Search method
- Ridder's method
- Splitting Circle method
- Maple's `fsolve`
- MATLAB's `fzero`
- Wilkinson's *Perfidious Polynomial*:

$$w(x) := \prod_{k=1}^{20} (x - k)$$

S II. Special Topics: Modified Newton's Method

Sections

DOI: 10.1201/9781003299257-6

II. Special Topic: Modified Newton's Method

Newton's Method Revisited

Newton's method uses the iteration function

$$N(x) = x - \frac{f(x)}{f'(x)}.$$

A *fixed point* of N, which is a value x^* where $N(x^*) = x^*$, is a zero of $f(x)$. It was really Simpson who realized the connection of Newton's method with calculus; Newton had developed an algebraic method in 1669 (not publishing it until 1711); Simpson's generalized version appeared in 1740 in his text *A New Treatise of Fluxions*. In 1690, midway between Newton and Simpson, Raphson published a simplified version of Newton's method that was based on iterations, much like ours today.

Modified Newton's Method

Modified Newton's Method

To modify Newton's method, replace the "correcting factor" quotient f/f' with f'/f''. Our new iterator is

$$M(x) = x - \frac{f'(x)}{f''(x)}.$$

Choose an initial value x_0. Then calculate the values $x_{n+1} = M(x_n)$ for $n = 1, 2, \ldots$. The question is: Does x_n have a limit?

Convergence?

Use Newton's example function: $y = x^3 - 2x - 5$. Then

$$M(x) = x - \frac{3x^2 - 2}{6x}.$$

Starting with $x_0 = 1$ gives the sequence $x_1 = 0.8\bar{3}$, $x_2 = 0.81\bar{6}$, $x_3 = 0.81649659$, $x_4 = 0.81649658$. Where are these points going?

Exercises

Problems

1. Use Maple to generate a personalized random polynomial with:

```
    > randomize(your phone number, no dashes or spaces):
      deg := 1+2*rand(4..9)():
      p := randpoly(x, degree=deg, coeffs=rand(-2..2)):
      p := unapply(sort(p), x);
```

2. Apply the Modified Newton's Method to your polynomial with a selection of starting points.
3. Produce a chart of your intermediate results.
4. Graph your polynomial and the trajectories using your data chart.
5. Can you determine a specific value where the trajectories change from one to another target point?
6. Investigate the Modified Newton's Method's rate of convergence.

V. Numerical Integration

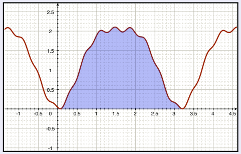
Sections

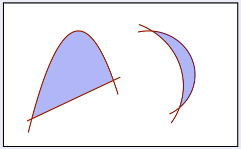
DOI: 10.1201/9781003299257-7

V. Numerical Integration

What Is *Numerical Integration*?

Numerical integration or *(numerical) quadrature* is the calculation of a definite integral using numerical formulas, not the fundamental theorem. The Greeks studied quadrature: given a figure, construct a square that has the same area. The two most famous are Hippocrates of Chios' *Quadrature of the Lune* (c. 450 BC) and Archimedes' *Quadrature of the Parabola* (c. 250 BC). Archimedes used the method of exhaustion — a precursor to calculus — invented by Eudoxus.





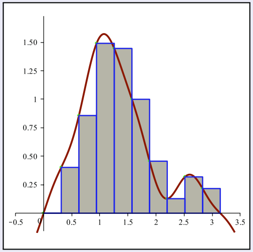
Squaring the circle is one of the classical problems of constructing a square with the area of a given circle — it was shown to be impossible by Lindemann's theorem (1882).[1]

[1] Lindemann was the first to prove that π is transcendental.

Methods of Elementary Calculus

Rectangle Methods

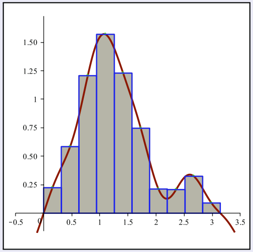
Left endpoint sum



$$A_n \approx \sum_{k=1}^{n} f(x_{k-1})\,\Delta x_k$$

$$\varepsilon_n \leq \frac{(b-a)^2}{2} \cdot M_1 \cdot \frac{1}{n}$$

Midpoint sum

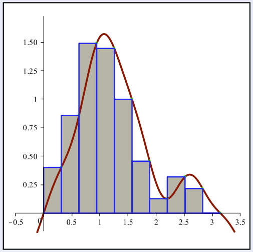


$$A_n \approx \sum_{k=1}^{n} f(m_k)\,\Delta x_k$$

$$m_k = \tfrac{1}{2}(x_{k-1} + x_k)$$

$$\varepsilon_n \leq \frac{(b-a)^3}{24} \cdot M_2 \cdot \frac{1}{n^2}$$

Right endpoint sum



$$A_n \approx \sum_{k=1}^{n} f(x_k)\,\Delta x_k$$

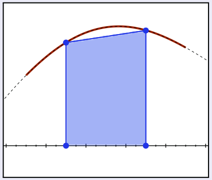
$$\varepsilon_n \leq \frac{(b-a)^2}{2} \cdot M_1 \cdot \frac{1}{n}$$

where $M_i = \max\left|f^{(i)}(x)\right|$

$$\int_0^{\pi} f(x)\,dx = 2 \approx [1.984, 2.008, 1.984]_{n=10}$$

Trapezoid Sums

Trapezoid Sums



Instead of the degree 0 rectangle approximations to the function, use a linear degree 1 approximation. The area of the trapezoid is given by

$$A_T = \tfrac{1}{2}[f(x_{k-1}) + f(x_k)]\,\Delta x_k.$$

This gives an approximation for the integral

$$\int_a^b f(x)\,dx \approx \sum_{k=1}^{n} \tfrac{1}{2}[f(x_{k-1}) + f(x_k)]\,\Delta x_k.$$

[Midpoint: measure height at average x v. trapezoid: average the height measures]

The formula is often written as

$$T_n \approx \left[f(x_0) + \left(2\sum_{k=1}^{n-1} f(x_k) \right) + f(x_n) \right] \frac{\Delta x_k}{2}.$$

Error for the trapezoid rule is

$$\varepsilon_n \le \frac{(b-a)^3}{12} \cdot M_2 \cdot \frac{1}{n^2}.$$

Sample Trapezoid

Example

Let $f(x) = \sin(x) + \frac{1}{2}\sin(2x) - \frac{1}{4}\sin(4x) + \frac{1}{16}\sin(8x)$ over $[0,\pi]$.

With an equipartition,

$$\Delta x = \pi/10 \approx 0.314.$$

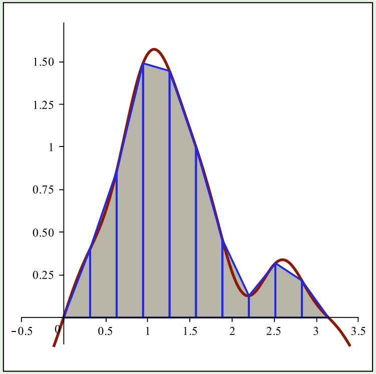
Then

$$T_{10} = \left[f(0) + \left(2\sum_{k=1}^{10} f\left(\tfrac{k}{9}\pi\right) \right) + f(\pi) \right] \frac{\Delta x}{2}$$

which gives

$$T_{10} = 1.984$$

with absolute error of 0.016.

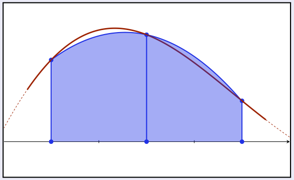


Simpson's Rule

Simpson's Rule

We now move to a degree 2 approximation. The easiest way to have 3 data pts is to take the panels in pairs: instead of rectangle base $[x_i, x_{i+1}]$, use $[x_i, x_{i+1}, x_{i+2}]$. So we require an even number of panels. The area under the parabola is

$$A_S = \frac{1}{3}\left[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\right]\Delta x.$$



This gives a $2n$-panel approximation for the integral

$$\int_a^b f(x)\,dx \approx \left[\sum_{k=1}^{n} f(x_{2k-2}) + 4f(x_{2k-1}) + f(x_{2k})\right]\frac{\Delta x}{3}$$

most often written as

$$S_{2n} = [f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4(f(x_{2n-1}) + f(x_{2n})]\,\frac{\Delta x}{3}.$$

The error is bounded by

$$\varepsilon_n \le \frac{(b-a)^5}{180}\cdot M_4 \cdot \frac{1}{n^4}.$$

Sample Simpson

Example

Let $f(x) = \sin(x) + \frac{1}{2}\sin(2x) - \frac{1}{4}\sin(4x) + \frac{1}{16}\sin(8x)$ over $[0, \pi]$.

With a 10-panel equipartition,

$$\Delta x = \pi/10 \approx 0.3141592654.$$

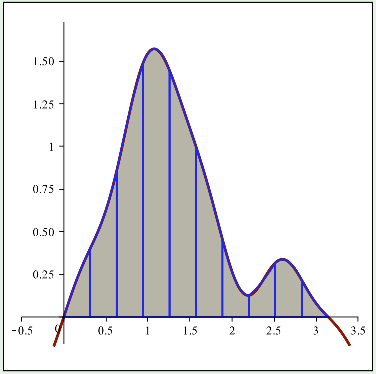
Then, with $y_i = f(x_i)$,

$$S_{10} = \frac{1}{3}\left[y_0 + 4y_1 + 2y_2 + \cdots + 4y_9 + y_{10}\right]\Delta x$$

which gives

$$S_{10} = 2.000006784$$

with absolute error of $6.78 \cdot 10^{-6}$.



A Maple Comparison

Approximating a Difficult Integral

Consider $\int_1^2 \frac{10^{-4}}{(x-\frac{1}{2}\pi)^2+10^{-8}}\,dx$. This integrand has a sharp peak at $\pi/2$.

The exact value of the integral (using the *FToC*) is

$$\arctan(5\cdot 10^3\cdot(4-\pi)) - \arctan(5\cdot 10^3\cdot(2-\pi)) \approx 3.1411844701381.$$

Maple gives

$$n=50 \qquad \begin{bmatrix} \textit{left} & 0.0497133 \\ \textit{right} & 0.0497180 \\ \textit{midpoint} & 3.1210200 \\ \textit{trapezoid} & 0.0497157 \\ \textit{Simpson} & 2.0972500 \end{bmatrix}$$

$$n=500 \qquad \begin{bmatrix} \textit{left} & 0.541336 \\ \textit{right} & 0.541336 \\ \textit{midpoint} & 4.052010 \\ \textit{trapezoid} & 0.541336 \\ \textit{Simpson} & 2.881790 \end{bmatrix}$$

$$n=5000 \qquad \begin{bmatrix} \textit{left} & 3.42282 \\ \textit{right} & 3.42282 \\ \textit{midpoint} & 2.88243 \\ \textit{trapezoid} & 3.42282 \\ \textit{Simpson} & 3.06256 \end{bmatrix}$$

To achieve a relative error below 1% requires approximately $n \geq 6000$.

The Chart

Quadrature	*"Height"*	*Error Bound*[2]
Left end point	$f(x_i)$	$\frac{(b-a)^2}{2} \cdot M_1 \cdot \frac{1}{n} = O(h)$
Right end point	$f(x_{i+1})$	$\frac{(b-a)^2}{2} \cdot M_1 \cdot \frac{1}{n} = O(h)$
Trapezoid Rule	$\frac{f(x_i)+f(x_{i+1})}{2}$	$\frac{(b-a)^3}{12} \cdot M_2 \cdot \frac{1}{n^2} = O(h^2)$
Midpoint	$f\left(\frac{x_i+x_{i+1}}{2}\right)$	$\frac{(b-a)^3}{24} \cdot M_2 \cdot \frac{1}{n^2} = O(h^2)$
Simpson's Rule	$\frac{f(x_i)+4f(x_{i+1})+f(x_{i+2})}{3}$	$\frac{(b-a)^5}{180} \cdot M_4 \cdot \frac{1}{n^4} = O(h^4)$

where $M_i \geq \max \left| f^{(i)}(x) \right|$ and $h = 1/n$.

[2] *An approximation without an error bound has little to no value!*

Gaussian Quadrature

Johann Carl Friedrich Gauss

About 1815, while Gauss was finishing constructing an astronomical observatory, he wrote a paper[3] on approximating integrals. Gauss's technique was studied and extended by Christoffel in 1858. There are several good ways to develop this method. We'll use the easiest . . .

In Search of Improvements

Write the rules we've seen as sums:

Left endpt: $L_n = \frac{1}{n} f(x_0) + \frac{1}{n} f(x_1) + \cdots + \frac{1}{n} f(x_{n-1})$

Right endpt: $R_n = \frac{1}{n} f(x_1) + \frac{1}{n} f(x_2) + \cdots + \frac{1}{n} f(x_n)$

Midpoint: $M_n = \frac{1}{n} f(x_{m_1}) + \frac{1}{n} f(x_{m_2}) + \cdots + \frac{1}{n} f(x_{m_n})$

Trapezoid: $T_n = \frac{1}{2n} f(x_0) + \frac{1}{n} f(x_1) + \cdots + \frac{1}{n} f(x_{n-1}) + \frac{1}{2n} f(x_n)$

Simpson's: $S_n = \frac{1}{3n} f(x_0) + \frac{4}{3n} f(x_1) + \frac{2}{3n} f(x_2) + \cdots + \frac{4}{3n} f(x_{n-1}) + \frac{1}{3n} f(x_n)$

[3] "Methodus nova integralium valores per approximationem inveniendi," *Comment Soc Regiae Sci Gottingensis Recentiores, v. 3, 1816*.

Patterns

Observations

- Each of the formulas has the same form: a *weighted sum*

  $$A_n = w_1 \cdot f(x_1) + w_2 \cdot f(x_2) + \cdots + w_n \cdot f(x_n)$$

  with different sets of weights w_i and different sets of nodes x_i.

- Any closed interval can be mapped to and from $[-1, 1]$, so we can focus just on $\displaystyle\int_{-1}^{1} f(x)\,dx$. $\left[T(t) = 2\frac{t-a}{b-a} - 1\,;\;\; T^{-1}(t) = a\left(\frac{1-t}{2}\right) + b\left(\frac{1+t}{2}\right)\right]$

- Gauss posed the question: Is there a “best choice” of weights $\{w_i\}$ and nodes $\{x_i\}$? Do nodes have to be equidistant?

- The answer depends on what “best” means.

Since we have $2n$ unknowns w_i and x_i, let's look for a set that integrates a $2n-1$ degree polynomial exactly. (*Remember: a $2n-1$ degree polynomial has $2n$ coefficients.*)

Sampling 3

Example (Third Degree)

Set $n = 3$. Determine the choice of w_i and of x_i so that

$$\int_{-1}^{1} x^p \, dx = \sum_{k=1}^{3} w_k \cdot x_k^p$$

exactly for $p = 0, 1, \ldots, 5 = 2 \cdot 3 - 1$.

The range for the power p gives us six equations:

$$\left\{ \begin{array}{l} w_1 \quad + w_2 \quad + w_3 \quad = 2 \\ w_1 x_1 + w_2 x_2 + w_3 x_3 = 0 \\ w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 = \frac{2}{3} \\ w_1 x_1^3 + w_2 x_2^3 + w_3 x_3^3 = 0 \\ w_1 x_1^4 + w_2 x_2^4 + w_3 x_3^4 = \frac{2}{5} \\ w_1 x_1^5 + w_2 x_2^5 + w_3 x_3^5 = 0 \end{array} \right\} \Longrightarrow \left\{ \begin{array}{ccc} x_1 = -\sqrt{\frac{3}{5}}, & x_2 = 0, & x_3 = \sqrt{\frac{3}{5}} \\ w_1 = \frac{5}{9}, & w_2 = \frac{8}{9}, & w_3 = \frac{5}{9} \end{array} \right.$$

Our Gaussian quadrature is $G_3(f) = \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right)$.

Testing Gauss

Random Polynomials

Generate and test a random 5th-degree polynomial.

```
p := unapply(sort(randpoly(x, degree = 5), x), x)
```

$$x \to -7x^5 + 22x^4 - 55x^3 - 94x^2 + 87x - 56$$

```
G3 := 5/9*p(-sqrt(3/5)) + 8/9*p(0) + 5/9*p(sqrt(3/5))
```

$$-\frac{2488}{15}$$

```
Int(p(x), x = -1..1) = int(p(x), x = -1..1)
```

$$\int_{-1}^{1} p(x)\,dx = -\frac{2488}{15}$$

Generate and test a random 7th-degree polynomial.

```
q := unapply(sort(randpoly(x, degree = 7), x), x)
```

$$x \to 97x^7 - 73x^6 - 4x^5 - 83x^3 - 10x - 62$$

```
int(q(x),x = -1..1)= 5/9*q(-sqrt(3/5)) + 8/9*q(0) + 5/9*q(sqrt(3/5))
```

$$\frac{722}{7} = \frac{2662}{25}$$

Gaussian Properties

Theorem (Error Estimate)

Let f have $2n$ continuous derivatives. Then for $\varepsilon_n = \left| G_n - \int_{-1}^{1} f(x)dx \right|$,

$$\varepsilon_n \le \frac{\pi}{2^{2n} \cdot (2n)!} \cdot M_{2n}$$

where $M_{2n} \ge \max \left| f^{(2n)}(x) \right|$.

Values of Gaussian Weights and Nodes

There are numerous sources online, e.g.,:

1. The classic Abramowitz and Stegun *Handbook* (see the entire book)
2. Holoborodko or Kamermans

We could calculate the values directly:

Set $P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} \left[\left(x^2 - 1\right)^n \right]$ (the Legendre polynomials). Then

$$\{x_i\}_{i=1}^{n} = \{\textit{zeros of } P_n\} \qquad \text{and} \qquad w_i = \frac{2}{(1 - x_i^2)\left[P_n'(x_i)\right]^2}.$$

Gauss-Kronrod Quadrature

Aleksandr Kronrod's Idea (1964)

One difficulty in Gaussian quadrature is that increasing the number of nodes requires recomputing all the values of

- nodes
- weights
- function evaluations

Kronrod[4] discovered he could interlace $n+1$ new nodes with n original Gaussian nodes and have a rule of order $3n+1$. A $2n+1$ node Gaussian quadrature would have order $4n+1$, but with significant extra computation for an increase of only n in order over Kronrod's method.

$\Big[$ *Bad news: calculating the nodes and weights is way beyond the scope of our class. The nodes are the roots of the Stieltjes or Stieltjes-Legendre polynomials. (App IV)* $\Big]$

Gauss-Kronrod quadrature is used in Maple, Mathematica, MATLAB, and Sage; it's included in the QUADPACK library, the GNU Scientific Library, the NAG Numerical Libraries, and in R. $\text{GK}_{7,15}$ is the basis of numerical integration in TI calculators. *(Casio uses Simpson's rule; HP, adaptive Romberg.)*

[4]Kronrod, A. S. (1964.) "Integration with control of accuracy" (*in Russian*), *Dokl. Akad. Nauk SSSR* 154, 283–286.

Gauss-Kronrod Quadrature in Practice

$GK_{7,15}$ (1989)

A widely used implementation is based on a Gaussian quadrature with 7 nodes. Kronrod adds 8 to total 15 nodes.

$$G_7 = \sum_{k=1}^{7} w_k f(x_k)$$

$$GK_{7,15} = \sum_{j=1}^{15} w_j f(x_j)$$

$$\varepsilon_{7,15} \approx \left|G_7 - GK_{7,15}\right|$$

or, in practice, use[5]

$$\approx \left[200\left|G_7 - GK_{7,15}\right|\right]^{3/2}$$

$GK_{7,15}$ on $[-1,1]$

Gauss-7 nodes	Weights
0.00000 00000 00000	0.41795 91836 73469
±0.40584 51513 77397	0.38183 00505 05119
±0.74153 11855 99394	0.27970 53914 89277
±0.94910 79123 42759	0.12948 49661 68870

Kronrod-15 nodes		Weights
0.00000 00000 00000	*G*	0.20948 21410 84728
±0.20778 49550 07898	*K*	0.20443 29400 75298
±0.40584 51513 77397	*G*	0.19035 05780 64785
±0.58608 72354 67691	*K*	0.16900 47266 39267
±0.74153 11855 99394	*G*	0.14065 32597 15525
±0.86486 44233 59769	*K*	0.10479 00103 22250
±0.94910 79123 42759	*G*	0.06309 20926 29979
±0.99145 53711 20813	*K*	0.02293 53220 10529

[5] Kahaner, Moler, & Nash, *Numerical Methods and Software*, Prentice-Hall, 1989.

GK Sample

Example

Find $\int_{-1}^{1} e^{-x^2}\,dx$.

Using Maple gives:

$$G_7 = \sum_{k=1}^{7} w_k f(x_k) = 1.4936482888 6941$$

$$GK_{7,15} = \sum_{k=1}^{15} w_k f(x_k) = 1.49364826562485$$

$$\varepsilon_{7,15} \approx \left|G_7 - GK_{7,15}\right| = 2.324456 \cdot 10^{-8}$$

`int(f(x),x=-1..1,numeric)` $\approx 1.49364826562485 = GK_{7,15}$

See Maple's Online Help for int/numeric to see the methods available.

A Class Exercise

Easy, but Hard

Set $f(x) = x - \lfloor x \rfloor$. Calculate $\int_0^{6.4} f(x)\,dx$.

Set $n = 10$. Find

1. The exact value	3.08
2. Left endpoint approximation	
3. Right endpoint approximation	
4. Midpoint approximation	
5. Trapezoid rule approximation	
6. Simpson's rule approximation	
7. Gauss 7 quadrature	
8. Gaussian-Kronrod 7-15 quadrature	

A Menagerie of Test Integrals

Integrals for Testing Numerical Quadratures,[6] I

Lyness:

1. $$I(\lambda) = \int_1^2 \frac{0.1}{(x-\lambda)^2+0.01}\,dx$$

Piessens, de Doncker-Kapenga, Überhuber, & Kahaner:

2. $$\int_0^1 x^\alpha \log\left(\frac{1}{x}\right) dx = \frac{1}{(1+\alpha)^2}$$

3. $$\int_0^1 \frac{4^{-\alpha}}{\left(x-\frac{\pi}{4}\right)^2+16^{-\alpha}}\,dx = \tan^{-1}\left((4-\pi)4^{\alpha-1}\right) + \tan^{-1}\left(\pi 4^{\alpha-1}\right)$$

4. $$\int_0^\pi \cos(2^\alpha \sin(x))\,dx = \pi J_0(2^\alpha)$$

5. $$\int_0^1 \left|x-\tfrac{1}{3}\right|^\alpha dx = \frac{\left(\frac{2}{3}\right)^{\alpha+1}+\left(\frac{1}{3}\right)^{\alpha+1}}{1+\alpha}$$

6. $$\int_0^1 \left|x-\frac{\pi}{4}\right|^\alpha dx = \frac{\left(1-\frac{\pi}{4}\right)^{\alpha+1}+\left(\frac{\pi}{4}\right)^{\alpha+1}}{1+\alpha}$$

7. $$\int_{-1}^{+1} \frac{1}{\sqrt{1-x^2}}\,\frac{1}{1+x+2^{-\alpha}}\,dx = \frac{\pi}{\sqrt{(1+2^{-\alpha})^2-1}}$$

[6] D. Zwillinger, *Handbook of Integration* p. 272, (A K Peters/CRC Press, 1992).

Test Integrals, II

Integrals for Testing Numerical Quadratures, II

Piessens *et al.* (continued):

8. $\displaystyle\int_0^{\pi/2} \sin^{\alpha-1}(x)\,dx = 2^{\alpha-2}\,\frac{\Gamma^2\left(\frac{\alpha}{2}\right)}{\Gamma(\alpha)}$

9. $\displaystyle\int_0^1 \log^{\alpha-1}\left(\frac{1}{x}\right)dx = \Gamma(\alpha)$

10. $\displaystyle\int_0^1 \frac{\cos(2^\alpha x)}{\sqrt{x(1-x)}}dx = \pi\cos(2^{\alpha-1})\,\mathrm{J}_0(2^{\alpha-1})$

11. $\displaystyle\int_0^\infty x^2 e^{-2^{-\alpha}x}\,dx = 2^{3\alpha+1}$

12. $\displaystyle\int_0^\infty \frac{x^{\alpha-1}}{(1+10x)^2}\,dx = \frac{(1-\alpha)\pi}{10^\alpha\sin(\pi\alpha)}$

Berntsen, Espelid, & Sørevik:

13. $\displaystyle\int_0^1 \left|x-\tfrac{1}{3}\right|^{-1/2}dx$ (*singularity*)

14. $\displaystyle\int_{-1}^1 U(x)\,e^{x/2}\,dx$ (*discontinuity*)

15. $\displaystyle\int_0^1 e^{2|x-1/3|}\,dx$ (C_0 *function*)

16. $\displaystyle\int_1^2 \frac{10^{-4}}{\left(x-\frac{\pi}{2}\right)^2+10^{-8}}\,dx$ (*sharp peak*)

Appendix IV: Legendre & Stieltjes Polynomials for $\mathrm{GK}_{7,15}$

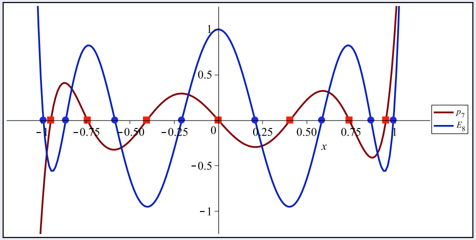
The Polynomials of $\mathrm{GK}_{7,15}$

The Gaussian nodes for G_7 are the roots of the Legendre polynomial p_7

$$p_7(x) = \frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x.$$

The additional nodes Kronrod adds for $\mathrm{GK}_{7,15}$ are the roots of the Stieltjes polynomial E_8 (from solving the system: $\left\{\int_{-1}^{1} p_7(x)E_8(x)x^k\,dx = 0 \text{ for } k = 0..7\right\}$)

$$E_8(x) = c\left[\frac{4854324041}{52932681}x^8 - \frac{1142193892}{5881409}x^6 + \frac{765588166}{5881409}x^4 - \frac{501576364}{17644227}x^2 + 1\right].$$



Problems

Exercises

For each of the following functions, investigate the integrals using: left endpoint, midpoint, trapezoid, and Simpson's rules.

1. $S(x) = \int_0^x \left[\sin(t^2/2) - \sqrt{\pi}/(2x)\right] dt$
2. *Lyness' integral* $I(\lambda) = \int_1^2 \frac{0.1}{(x-\lambda)^2 + 0.01}\, dx$ *for* $\lambda = \pi/2$
3. *Modified Piessens' integral* $\int_{-1}^1 \left| x^2 - \frac{\pi^2}{16} \right|^{0.1} dx$

Investigate Gaussian and Gauss-Kronrod quadrature (after transforming the interval to $[-1,1]$*) of the integral:*

4. $\int_1^2 \frac{10^{-4}}{\left(x - \frac{\pi}{2}\right)^2 + 10^{-8}}\, dx$
5. *Explain why the integrals*

$$\int_{-1}^1 p_7(x)\, dx \qquad \int_{-1}^1 E_8(x)\, dx \qquad \int_{-1}^1 p_7(x) \cdot E_8(x)\, dx$$

are all equal to zero. Use numerical methods to approximate each.

Links and Others

Wikipedia entries:

- Newton-Cotes formulas
- Romberg's method
- Clenshaw-Curtis integration
- Cubature

More information:

- Adaptive Simpson's Rule
- Monte Carlo Integration
- Legendre Polynomials
- Chebyshev Polynomials

See also: MathWorld (Wolfram Research) and Interactive Educational Modules in Scientific Computing (U of I)

Investigate:

- Boole's Rule
- adaptive quadrature
- orthogonal polynomials
- Vandermonde Matrix
- Maple's `evalf/Int` command
- The Maple command `ApproximateInt` in the `Student[Calculus1]` package
- MATLAB's `integral` command
- Cubature

VI. Polynomial Interpolation

Sections

DOI: 10.1201/9781003299257-8

VI. Polynomial Interpolation

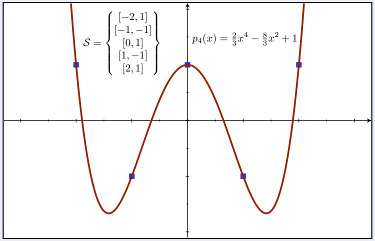
What Is *Polynomial Interpolation*?

An *interpolating polynomial* $p(x)$ for a set of points $\mathscr{S}$ is a polynomial that goes through each point of $\mathscr{S}$. That is, for each point $P_i = (x_i, y_i)$ in the set, $p(x_i) = y_i$.

Typical applications include:

Approximating Functions	TrueType Fonts (2nd deg)	Fast Multiplication
Cryptography	PostScript Fonts (3rd deg)	Data Compression

Since each data point determines the value of one polynomial coefficient, an n-point data set has an $n-1$ degree interpolating polynomial.



Free Sample

Finding an Interpolating Polynomial

Let $\mathscr{S} = \{[-2,1],[-1,-1],[0,1],[1,-1],[2,1]\}$.

1. Since $\mathscr{S}$ has 5 points, we compute a 4th degree polynomial
$$p_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4.$$
2. Substitute the values of $\mathscr{S}$ into p_4; write the results as a system of linear equations.

$$\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 - 2a_1 + 4a_2 - 8a_3 + 16a_4 \\ a_0 - a_1 + a_2 - a_3 + a_4 \\ a_0 \\ a_0 + a_1 + a_2 + a_3 + a_4 \\ a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 4 & -8 & 16 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

3. Solve via your favorite method: $p_4(x) = \frac{2}{3}x^4 - \frac{8}{3}x^2 + 1$.

Toward a Better Way: Lagrange Interpolation

Definition

Knots or Nodes: The x-values of the interpolation points.

Lagrange Fundamental Polynomial: Given a set of $n+1$ knots, define

$$\begin{aligned} L_i(x) &= \prod_{\substack{k=0..n \\ k\neq i}} \frac{x-x_k}{x_i-x_k} \\ &= \frac{x-x_0}{x_i-x_0} \times \cdots \times \frac{x-x_{i-1}}{x_i-x_{i-1}} \times \frac{x-x_{i+1}}{x_i-x_{i+1}} \times \cdots \times \frac{x-x_n}{x_i-x_n}. \end{aligned}$$

Lagrange Interpolating Polynomial: Given a set of $n+1$ data points (x_k, y_k), define

$$p_n(x) = \sum_{k=0}^{n} y_k L_k(x).$$

Sampling a Better Way

Example (Lagrange Fundamental Polynomials)

The set of knots $[-2,-1,0,1,2]$ gives

$$L_0(x) = \underbrace{\cancel{\frac{x-(-2)}{-2-(-2)}}}_{\textit{delete}} \cdot \frac{x-(-1)}{-2-(-1)} \cdot \frac{x-0}{-2-0} \cdot \frac{x-1}{-2-1} \cdot \frac{x-2}{-2-2}$$

$$L_1(x) = \frac{x-(-2)}{-1-(-2)} \cdot \underbrace{\cancel{\frac{x-(-1)}{-1-(-1)}}}_{\textit{delete}} \cdot \frac{x-0}{-1-0} \cdot \frac{x-1}{-1-1} \cdot \frac{x-2}{-1-2}$$

$$L_2(x) = \frac{x-(-2)}{0-(-2)} \cdot \frac{x-(-1)}{0-(-1)} \cdot \underbrace{\cancel{\frac{x-0}{0-0}}}_{\textit{delete}} \cdot \frac{x-1}{0-1} \cdot \frac{x-2}{0-2}$$

$$L_3(x) = \frac{x-(-2)}{1-(-2)} \cdot \frac{x-(-1)}{1-(-1)} \cdot \frac{x-0}{1-0} \cdot \underbrace{\cancel{\frac{x-1}{1-1}}}_{\textit{delete}} \cdot \frac{x-2}{1-2}$$

$$L_4(x) = \frac{x-(-2)}{2-(-2)} \cdot \frac{x-(-1)}{2-(-1)} \cdot \frac{x-0}{2-0} \cdot \frac{x-1}{2-1} \cdot \underbrace{\cancel{\frac{x-2}{2-2}}}_{\textit{delete}}$$

Graph the L_k!

Sampling a Better Way

Example

Let $\mathscr{S} = \{[-2,1],[-1,-1],[0,1],[1,-1],[2,1]\}$.

We have $[x_k] = [-2,-1,0,1,2]$ and $[y_k] = [1,-1,1,-1,1]$. Then $p_4(x) = \sum_{k=0}^{4} y_k L_k(x)$. So

$$\begin{aligned} p_4(x) = {} & (1)\cdot\tfrac{x+1}{-2+1}\cdot\tfrac{x-0}{-2-0}\cdot\tfrac{x-1}{-2-1}\cdot\tfrac{x-2}{-2-2} \\ & +(-1)\cdot\tfrac{x+2}{-1+2}\cdot\tfrac{x-0}{-1-0}\cdot\tfrac{x-1}{-1-1}\cdot\tfrac{x-2}{-1-2} \\ & +(1)\cdot\tfrac{x+2}{0+2}\cdot\tfrac{x+1}{0+1}\cdot\tfrac{x-1}{0-1}\cdot\tfrac{x-2}{0-2} \\ & +(-1)\cdot\tfrac{x+2}{1+2}\cdot\tfrac{x+1}{1+1}\cdot\tfrac{x-0}{1-0}\cdot\tfrac{x-2}{1-2} \\ & +(1)\cdot\tfrac{x+2}{2+2}\cdot\tfrac{x+1}{2+1}\cdot\tfrac{x-0}{2-0}\cdot\tfrac{x-1}{2-1}. \end{aligned}$$

Then simplified $p_4(x) = \frac{2}{3}x^4 - \frac{8}{3}x^2 + 1$.

An "Easier" L_k Formula

Compact Expressions

For a set $[x_k]$ of $n+1$ knots, we defined $L_i(x) = \prod_{\substack{k=0..n \\ k\neq i}} \frac{x-x_k}{x_i-x_k}$. This formula is computationally intensive.

Set $\omega(x) = \prod_{k=0}^{n} (x-x_k)$.

1. The numerator of L_i is $\omega(x)/(x-x_i)$.
2. The denominator of L_i is $\omega(x)/(x-x_i)$ evaluated at x_i. Rewrite as $\dfrac{\omega(x)}{x-x_i} = \dfrac{\omega(x)-\omega(x_i)}{x-x_i}$. Take the limit as $x \to x_i$:

$$\lim_{x\to x_i} \frac{\omega(x)-\omega(x_i)}{x-x_i} = \omega'(x_i).$$

Thus $L_i(x) = \dfrac{\omega(x)}{(x-x_i)\,\omega'(x_i)}$. A very compact formula!

Properties of the L_k's

Proposition

For the Lagrange interpolating polynomial $p_n(x) = \sum_{k=0}^{n} y_k L_k(x)$*:*

1. $p_n(x)$ *is the unique* n*th degree polynomial s.t.* $p(x_k) = y_k$ *for* $k = 0..n$.
2. $L_k(x_j) = \delta_{kj} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$. *(See Kronecker delta.)*
3. $\sum_{k=0}^{n} L_k(x) = 1$.
4. *If* $q(x)$ *is a polynomial of degree* $\leq n$ *with* $y_k = q(x_k)$*, then* $q \equiv p_n$.
5. *The set* $\{L_k(x) : k = 0..(n-1)\}$ *is a basis of* $\mathbb{P}_{n-1}$.

Theorem (Lagrange Interpolation Error)

If $f \in \mathscr{C}^{n+1}[a,b]$ *and* $\{x_k\} \in [a,b]$*, then*

$$\varepsilon_n = |f(x) - p_n(x)| \leq \frac{(b-a)^{n+1}}{(n+1)!} \max |f^{(n+1)}(x)|.$$

Drawbacks

More Knots

To decrease the error, use more knots. But ... all the $L_k(x)$ change.

1. Set $\{x_k\} = \{-2, 1, 2\}$. Then

$$\begin{aligned}
L_0(x) &= \frac{x-1}{-2-1} \cdot \frac{x-2}{-2-2} = \tfrac{1}{12}x^2 - \tfrac{1}{4}x + \tfrac{1}{6} \\
L_1(x) &= \frac{x+2}{1+2} \cdot \frac{x-2}{1-2} \qquad = -\tfrac{1}{3}x^2 + \tfrac{4}{3} \\
L_2(x) &= \frac{x+2}{2+2} \cdot \frac{x-1}{2-1} \qquad = \tfrac{1}{4}x^2 + \tfrac{1}{4}x - \tfrac{1}{2}.
\end{aligned}$$

2. Set $\{x_k\} = \{-2, -1, 1, 2\}$. Then

$$\begin{aligned}
L_0(x) &= \frac{x+1}{-2+1} \cdot \frac{x-1}{-2-1} \cdot \frac{x-2}{-2-2} = -\tfrac{1}{12}x^3 + \tfrac{1}{6}x^2 + \tfrac{1}{12}x - \tfrac{1}{6} \\
L_1(x) &= \frac{x+2}{-1+2} \cdot \frac{x-1}{-1-1} \cdot \frac{x-2}{-1-2} = \tfrac{1}{16}x^3 - \tfrac{1}{6}x^2 - \tfrac{2}{3}x + \tfrac{2}{3} \\
L_2(x) &= \frac{x+2}{1+2} \cdot \frac{x+1}{1+1} \cdot \frac{x-2}{1-2} \qquad = -\tfrac{1}{6}x^3 - \tfrac{1}{6}x^2 + \tfrac{2}{3}x + \tfrac{2}{3} \\
L_3(x) &= \frac{x+2}{2+2} \cdot \frac{x+1}{2+1} \cdot \frac{x-1}{2-1} \qquad = \tfrac{1}{12}x^3 + \tfrac{1}{6}x^2 - \tfrac{1}{12}x - \tfrac{1}{6}.
\end{aligned}$$

Interlude: Bernstein Polynomials

Definition (Bernstein Polynomials of f)

Bernstein Basis Polynomials: $b_{n,k}(x) = \binom{n}{k} x^k (1-x)^{n-k}$ for $k = 0..n$

Bernstein Polynomial of f: Let $f:[0,1] \to \mathbb{R}$. Then

$$B_n(f) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}.$$

Note: If $g:[a,b] \to \mathbb{R}$, then use $f(x) = g\big(a + (b-a)x\big)$.

Example

Let $f(x) = x^3$ for $x \in [0,1]$. Then $B_n(f) = \sum_{k=0}^{n} \frac{k^3}{n^3} \binom{k}{n} x^k (1-x)^{n-k}$.

$$B_1(x) = x \qquad B_2(x) = \tfrac{1}{4}x + \tfrac{3}{4}x^2$$

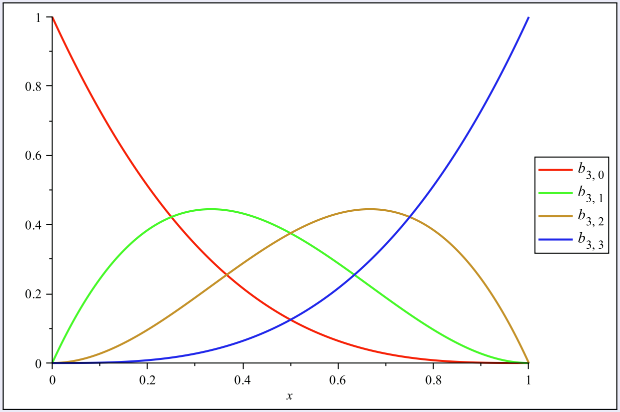
$$B_3(x) = \tfrac{1}{9}x + \tfrac{2}{3}x^2 + \tfrac{1}{9}x^3 \qquad B_4(x) = \tfrac{1}{16}x + \tfrac{9}{16}x^2 + \tfrac{3}{8}x^3$$

Bernstein Basis Functions

Bernstein Basis Functions, $n = 3$

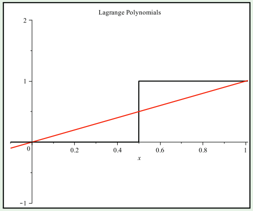
k=0: $b_{3,0}(x) = (1-x)^3$ k=1: $b_{3,1}(x) = 3x(1-x)^2$

k=2: $b_{3,2}(x) = 3x^2(1-x)$ k=3: $b_{3,3}(x) = x^3$

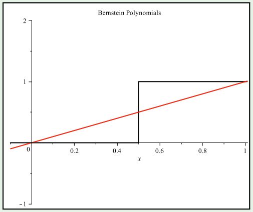


Bernstein and Lagrange

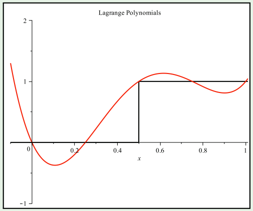
Example $(f(x) = \text{Heaviside}(x - \frac{1}{2}))$



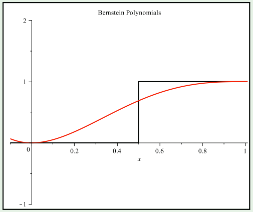
p_1



$B_1(f)$



p_4



$B_4(f)$

Newton Interpolation

Newton Basis Polynomials

In order to make it easy to add a new knot, we change the set of basis polynomials. Given a set of $n+1$ knots, $\{x_k\}$, set

$$\begin{aligned}
N_0(x) &= 1\\
N_1(x) &= (x-x_0)\\
N_2(x) &= (x-x_0)(x-x_1)\\
N_3(x) &= (x-x_0)(x-x_1)(x-x_2)\\
&\;\vdots\\
N_n(x) &= (x-x_0)(x-x_1)(x-x_2)\cdots(x-x_{n-1}).
\end{aligned}$$

Now let

$$P_n(x) = \sum_{k=0}^{n} a_k N_k(x).$$

Note that $\mathscr{B}_N = \{N_k(x) \,|\, k = 0..n\}$ forms a basis for $\mathbb{P}_n$.

The Newton Coefficients

Calculating the a_k's

For a set of $n+1$ data points $\{[x_k, y_k]\}$, define the *(forward) divided differences* recursively as

$$
\begin{aligned}
[y_0] &= y_0 \\
[y_0, y_1] &= \frac{[y_1] - [y_0]}{x_1 - x_0} \\
[y_0, y_1, y_2] &= \frac{[y_1, y_2] - [y_0, y_1]}{x_2 - x_0} \\
&\vdots
\end{aligned}
$$

Then the Newton interpolating polynomial is

$$
\begin{aligned}
P_n(x) &= [y_0] + [y_0, y_1](x - x_0) + [y_0, y_1, y_2](x - x_0)(x - x_1) \\
&\quad + [y_0, y_1, y_2, y_3](x - x_0)(x - x_1)(x - x_2) + \dots \\
&= \sum_{k=0}^{n} [y_0, \dots, y_k] N_k(x).
\end{aligned}
$$

First Sample

A Used Polynomial

Let $\mathscr{S} = \{[-2,1],[-1,-1],[0,1],[1,-1],[2,1]\}$.

Begin by building a *difference tableau*.

x	-2	-1	0	1	2
y	1	-1	1	-1	1
$[y_0]$	1	-1	1	-1	1
$[y_0,y_1]$		-2	2	-2	2
$[y_0,y_1,y_2]$			2	-2	2
$[y_0,y_1,y_2,y_3]$				$-\frac{4}{3}$	$\frac{4}{3}$
$[y_0,y_1,y_2,y_3,y_4]$					$\frac{2}{3}$

Then $P_4(x) = \sum_{k=0}^{n} [y_0,\ldots,y_k]N_k(x)$

$$= 1-2(x+2)+2(x+2)(x+1)-\tfrac{4}{3}(x+2)(x+1)(x)+\tfrac{2}{3}(x+2)(x+1)(x)(x-1)$$

$$P_4(x) = 1-\tfrac{8}{3}x^2+\tfrac{2}{3}x^4.$$

Second Sample

The Heaviside Function

Set $\mathscr{S} = \{[0,0],[\frac{1}{5},0],[\frac{2}{5},0],[\frac{3}{5},1],[\frac{4}{5},1],[1,1]\}$.

Begin by building a *difference tableau*.

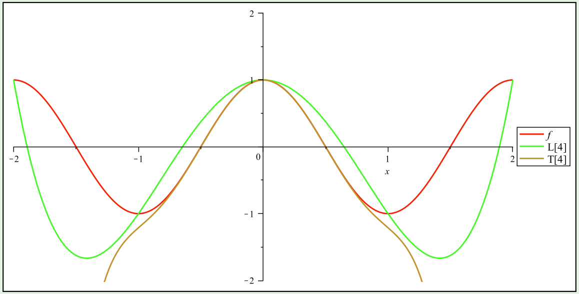
x	0	0.2	0.4	0.6	0.8	1
y	0	0	0	1	1	1
$[y_0,y_1]$		0	0	5	0	0
$[y_0,y_1,y_2]$			0	$\frac{25}{2}$	$-\frac{25}{2}$	0
$[y_0,y_1,y_2,y_3]$				$\frac{125}{6}$	$-\frac{125}{3}$	$\frac{125}{6}$
$[y_0,y_1,y_2,y_3,y_4]$					$-\frac{625}{8}$	$\frac{625}{8}$
$[y_0,y_1,y_2,y_3,y_4,y_5]$						$\frac{625}{4}$

Then $P_5(x) = \sum_{k=0}^{5} [y_0,\ldots,y_k]N_k(x)$

$$P_5(x) = \frac{137}{12}x - \frac{875}{8}x^2 + \frac{1000}{3}x^3 - \frac{3125}{8}x^4 + \frac{625}{4}x^5.$$

Two Comparisons

Example ($\cos(\pi x)$ v. Lagrange, Newton, & Taylor)



Lagrange: $$L_4 = \frac{(x+1)(x)(x-1)(x-2)}{24} + \frac{(x+2)(x)(x-1)(x-2)}{6} + \frac{(x+2)(x+1)(x-1)(x-2)}{4} + \frac{(x+2)(x+1)(x)(x-2)}{6} + \frac{(x+2)(x+1)(x)(x-1)}{24}$$

Newton: $$N_4 = 1 - 2(x+2) + 2(x+2)(x+1) - \tfrac{4}{3}(x+2)(x+1)(x) + \tfrac{2}{3}(x+2)(x+1)(x)(x-1)$$

Taylor: $$T_4 = 1 - \tfrac{\pi^2}{2}x^2 + \tfrac{\pi^4}{24}x^4$$

Second Comparison

Shifted Heaviside: $f(x) = \text{Heaviside}(x - 1/2)$ on $[0,1]$

Lagrange:
$$L_5 = \tfrac{3125}{12}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right)\left(x-\tfrac{4}{5}\right)(x-1) - \tfrac{3125}{24}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right)\left(x-\tfrac{3}{5}\right)(x-1) + \tfrac{625}{24}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right)\left(x-\tfrac{3}{5}\right)\left(x-\tfrac{4}{5}\right)$$

Newton:
$$N_5 = \tfrac{125}{6}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right) - \tfrac{625}{8}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right)\left(x-\tfrac{3}{5}\right) + \tfrac{625}{4}(x)\left(x-\tfrac{1}{5}\right)\left(x-\tfrac{2}{5}\right)\left(x-\tfrac{3}{5}\right)\left(x-\tfrac{4}{5}\right)$$

Bernstein: $B_5 = 10x^3(1-x)^2 + 5x^4(1-x) + x^5$

Taylor: T_5 centered at the middle $a = \frac{1}{2}$: Not possible. (*Why?*)
Centered at $a \in [0, \frac{1}{2})$, $T_5 = 0$
Centered at $a \in (\frac{1}{2}, 1]$, $T_5 = 1$

Interlude: Splines

Splines

Lagrange and Newton polynomials oscillate excessively when there are a number of closely spaced knots. To alleviate the problem, use "splines," piecewise, smaller-degree polynomials with conditions on their derivatives. The two most widely used splines:

Bézier splines are piecewise Bernstein polynomials [Casteljau (1959) and Bézier (1962)].

Cubic B-splines are piecewise cubic polynomials with second derivative equal to zero at the joining knots [Schoenberg (1946)].

Along with engineering, drafting, and CAD, splines are used in a wide variety of fields. TrueType fonts use 2-D quadratic Bézier curves. PostScript and MetaFont use 2-D cubic Bézier curves.

Exercises, I

Exercises

For each of the functions given in 1. to 5.:

- *Find the Lagrange polynomial of order* 6
- *Find the Newton polynomial of order* 6
- *Find the Bernoulli polynomial of order* 6

and plot the interpolation polynomial with the function.

1. $f(x) = \sin(2\pi x)$ *on* $[0,1]$
2. $g(x) = \ln(x+1)$ *on* $[0,2]$
3. $h(x) = \tan(\sin(x))$ *on* $[-\pi,\pi]$
4. $k(x) = \dfrac{x}{x^2+1}$ *on* $[-10,10]$
5. $S(x) = \displaystyle\int_0^x \left[\sin(\tfrac{1}{2}t^2) - \frac{\sqrt{\pi}}{2x}\right] dt$ *for* $x \in [0,10]$

6. *Find an interpolating polynomial for the data given below. Plot the polynomial with the data.*

$$\begin{bmatrix} 0.0 & 1.0 & 2.0 & 3.0 & 4.0 & 5.0 & 6.0 & 7.0 & 8.0 & 9.0 \\ 4.2 & 2.2 & 2.0 & 8.7 & 5.7 & 9.9 & 0.44 & 4.8 & 0.13 & 6.4 \end{bmatrix}$$

Exercises, II

Exercises

7. *An error bound for Newton interpolation with* $n+1$ *knots* $\{x_k\}$ *is*

$$|f(x)-N(x)| \le \frac{1}{(n+1)!} \cdot \max\left|f^{(n+1)}(x)\right| \cdot \prod_{k=0}^{n}(x-x_k).$$

   Show this bound is less than or equal to the Lagrange interpolation error bound. How does this make sense in light of the unicity of interpolation polynomials? (NB: The formula for Newton interpolation also applies to Lagrange interpolation.)

8. *Investigate interpolating Runge's "bell function"* $r(x) = e^{-x^2}$ *on the interval* $[-5,5]$

   a. *with* 10 *equidistant knots,*
   b. *with "Chebyshev knots"* $x_k = 5\cos((n-j)\pi/n)$ *with* $j = 0..10$.

9. *Write a Maple function that produces a difference tableau for a data set. Test your function with the data set produced by*
   > $myData := [seq([k, rand(-9..9)()], k = 1..10)];$

Links and Others

Wikipedia entries:

- Lagrange Polynomial
- Bernstein Polynomials
- Newton Polynomial
- Wavelets

More information:

- Lagrange Interpolation
- Newton Interpolation
- Legendre Polynomial Calculator
- Chebyshev Polynomial Calculator

See also: Interpolation at MathWorld (Wolfram Research)

Investigate:

- Aitken Interpolation
- Extrapolation
- Gauss's Interpolation Formula
- Hermite Interpolation
- Newton-Cotes Formulas
- Thiele's Interpolation Formula
- Vandermonde Matrix
- The Maple command `PolynomialInterpolation` in the `CurveFitting` package
- MATLAB's `fit` command
- Splines and Bézier curves
- Rational Interpolation

S III. Case Study: TI Calculator Numerics

Sections

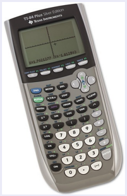
DOI: 10.1201/9781003299257-9

S III. Case Study: TI Calculator Numerics

Introduction

Texas Instruments started a research project in 1965 to design a pocket calculator. The first pocket calculators appeared in the early 1970s from the Japanese companies Sanyo, Canon, and Sharp. The HP-35 (*it had 35 keys*) was the first scientific pocket calculator, introduced by Hewlett Packard in 1972 for \$395. In 1974, HP released the HP-65 (\$795), the first programmable pocket calculator.

Texas Instruments' TI-8x series is based on the Zilog Z-80 processor (1976), an 8-bit CPU originally running at 2 MHz. The TI-81 came out in 1990 with a 2 MHz Z80 and 2.4 KB RAM. The TI-84 Plus CE (released in 2015) has a 48 MHz Z80 with 4 MB Flash ROM and 256 KB RAM. In 2013, the calculators' displays upgraded to 320×240 pixels from the 1990's 96×64 pixels.[1]



[1] TI's Nspire calculators use an ARM processor.

TI-80 Series Calculators

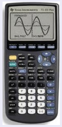
Timeline of the TI-80 Series

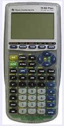
Model	*Year*	Z80 *Processor*	RAM KB / ROM MB
TI-81	1990	2 MHz	2.4 / 0
TI-82	1993	6 MHz	28 / 0
TI-83	1996	6 MHz	32 / 0
TI-83 Plus	1999	6 MHz	32 / 0.5
TI-83 Plus SE	2001	15 MHz	128 / 2
TI-84 Plus	2004	15 MHz	128 / 1
TI-84 Plus SE	2004	15 MHz	128 / 2
TI-84 Plus C	2013	15 MHz	128 / 4
TI-84 Plus CE	2015	48 MHz	256 / 4

TI-81

TI-82

TI-83


TI-83+


TI-83+SE

TI-84+

TI-84+SE

TI-84+CE

TI Floating Point

TI Floating Point Structure

TI's numeric model is not IEEE-754 compliant. The floating point format is

9 Bytes								
0	+1	+2	+3	+4	+5	+6	+7	+8
s/T	EXP	DD	DD	DD	DD	DD	DD	DD

s/T: Sign and Type Byte

8 Bits							
7	6	5	4	3	2	1	0
SIGN	*reserved*		TYPE				

Floating point types: Real: 0; Complex: `0Ch`; (List: `01h`; Matrix: `02h`; etc.)

EXP: Power of 10 exponent coded in binary, biased by `80h`

DD: Mantissa in BCD, 7 bytes of two digits per byte. While the mantissa has 14 digits, only 10 ($+2$ exponent digits) are displayed on the screen. (Many math routines use 9-byte mantissas internally to improve accuracy.)

Examples: $3.14159265 =$ | 00 | 80 | 31 | 41 | 59 | 26 | 50 | 00 | 00 |

$-230.45 =$ | 80 | 82 | 23 | 04 | 50 | 00 | 00 | 00 | 00 |

TI Floating Point Software

TI Floating Point Software

There are six RAM locations called the Floating Point Registers OP1 to OP6; each is 11 bytes (with 9 byte mantissas); they are used extensively for floating point computations. The routines listed below, called in assembly programs, operate on the value in OP1 unless noted.

TI's operating system[2] includes the functions:

Standard Function		*Transcendental*	
FPAdd (OP1+OP2)	Ceiling	Sin	ASin
FPSub (OP1−OP2)	Int	Cos	ACos
FPRecip (1/OP1)	Trunc	Tan	ATan
FPMult (OP1×OP2)	Frac	SinH	ASinH
FPDiv (OP1÷OP2)	Round	CosH	ACosH
FPSquare (OP1×OP1)	RndGuard (to 10 d)	TanH	ATanH
SqRoot	RnFx (to FIX)	LnX	EToX
Factorial ($n \cdot 0.5 \geq -0.5$)	Random	LogX	TenX (10^{OP1})
Max(OP1, OP2)	RandInt		
Min(OP1, OP2)			

[2]See *TI-83 Plus* *Developer Guide* (also covers the TI-84 series).

Numeric Derivatives

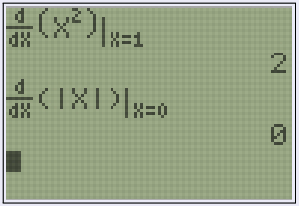
nDeriv

TI uses a centered difference formula

$$f'(a) \approx \frac{f(a+\varepsilon)-f(a-\varepsilon)}{2\varepsilon}.$$

The default step size is $\varepsilon = 0.001$. The command can't be nested and doesn't check whether or not f is differentiable at a.

Syntax: `nDeriv(`*expression*`,`*variable*`,`*value*`[,`ε`])`



```
d/dX(X²)|X=1
                    2
d/dX(|X|)|X=0
                    0
```

(Screen images are from a TI-84+ SE *with the* 2.55MP OS.*)*

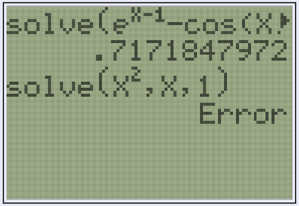
Numerically Finding Roots

solve

TI uses a blend of the bisection and secant root-finding algorithms. (See Appendix V.) The default initial interval is $[-10^{99}, 10^{99}]$. Solve does not find roots of even multiplicity since the algorithm requires a sign change. (solve is available only through CATALOG or the Solver application.)

To find a different root, use a starting value close to the desired new solution; a graph is a good "initial value generator."

Syntax: `solve(`*expression*`,`*variable*`,`*initial_guess*`)`



```
solve(e^(X-1)-cos(X▸
          .7171847972
solve(X²,X,1)
                Error
```

Numeric Quadrature

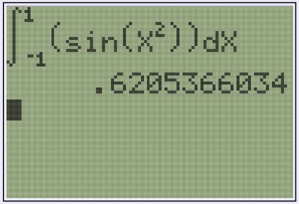
fnInt

TI uses an adaptive Gauss-Kronrod 7-15 quadrature

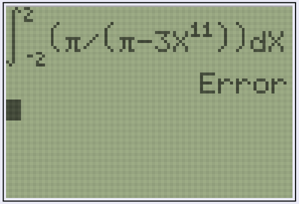
$$\int_{-1}^{1} f(x)\,dx \approx \sum_{k=1}^{15} f(x_k)\cdot w_k.$$

The default error tolerance is $\varepsilon = 10^{-5}$. The command can't be nested and doesn't check if f is integrable over $[a,b]$.

Syntax: `fnInt(`*expression*`,`*variable*`,`*lower*`,`*upper*`[,`ε`])`



```
∫_{-1}^{1}(sin(X²))dX
             .6205366034
```



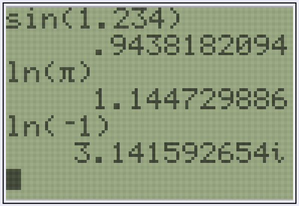
```
∫_{-2}^{2}(π/(π-3X^11))dX
                   Error
```

Transcendental and Other Functions

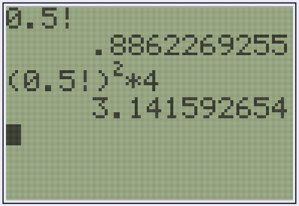
Numeric Function Calculations

- For trigonometric, logarithmic, and exponential functions, TI uses a modified CORDIC algorithm. CORDIC's standard "rotations" of 2^{-k} are replaced with 10^{-k}. (See the CORDIC Project.)
- The factorial $x!$ where x is a multiple of $\frac{1}{2}$ for $-\frac{1}{2} \le x \le 69$ is computed recursively using

$$x! = \begin{cases} x \cdot (x-1)! & x > 0 \\ 1 & x = 0 \\ \sqrt{\pi} & x = -\frac{1}{2} \end{cases}$$



```
sin(1.234)
     .9438182094
ln(π)
      1.144729886
ln(-1)
     3.141592654i
```



```
0.5!
     .8862269255
(0.5!)²*4
     3.141592654
```

Appendix V: TI's Solving Algorithm

Bisection and Secant Combined

The `solve` function and the `Solver` application use a clever, modified combination of the secant method and bisection.[3] The logic is:

1. Order the bracketing points a and b so that $|f(b)| \leq |f(a)|$.
2. Calculate a new point c using the secant method.
3. If c is:
   a. outside the interval, replace c with the midpoint (bisection),
   b. too close to an endpoint (within h), replace c with $c = b \pm h$, a specified minimum step in the interval.
4. The new bracketing points are a & c or b & c, whichever pair has the sign change.
5. If the error tolerance is met or the number of iterations is maximum, then return c, otherwise, go to Step 1.

[3] "Solve() uses a combination of bisection and the secant method, as described in Shampine and Allen *Numerical Computing: An Introduction*, Saunders, 1973" (pp. 96–100 and 244) *according to the* TI 85 *Knowledge Base*.

Exercises, I

Problems

1. Enter `1 + 1`EE$^-$`13`. Now enter ANS $-$ `1`. Explain the result.

2. Enter $\pi - 3.141592654$ on a TI-84 and a TI Nspire CAS. Explain the different results.

3. Explain the result of `nDeriv(`$|x|$`,x,0)`.

4. Define `Y1` to be the function $f(x) = \dfrac{10^{-8} - (x - \pi/2)^2}{10^{-16} + (x - \pi/2)^2}$. Explain the results from using `solve` with an initial guess of:

   a. 0
   b. 1.5

5. Define f by $f(x) = \left|x - \frac{1}{3}\right|^{-8/9}$. Compare evaluating $\displaystyle\int_{-1}^{1} f(x)\,dx$ with

   a. a TI-84+ SE,
   b. Maple.

Exercises, II

Problems

6. Using the Gamma function, we can define $x! = \Gamma(x+1) = \int_0^\infty z^x e^{-z}\,dz$. Compute $(1/4)!$ using the Gamma function with a calculator and with Maple.
7. Investigate the integral $\int_{-1}^{1} \frac{\pi}{\pi - 3x^{39}}\,dx$ numerically and symbolically. First, graph the integrand.
8. Report on how the calculator computes points to draw a graph.
9. Compare graphs of $s(x) = \sqrt[3]{x}$ over the interval $-1 \leq x \leq 1$ from the calculator and from Maple. Explain the differences.
10. Explain the possible sources of error when the calculator computes

$$\int_0^1 \left(\frac{d}{dt} \left(T^{10} \right)_{T=X} \right) dX$$

using `nDeriv` and `fnInt`.

Projects

VII. Projects

The Project List

DOI: 10.1201/9781003299257-10

One Function for All

Project: One Function for All, the Normal Distribution

The mean and standard deviation for total SAT scores for 2021 are $\mu = 1061$ (from Evidence-Based Reading 533 + Math 528) and $\sigma = 217$, respectively.[1] Define the function

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt.$$

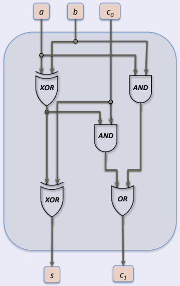
1. Estimate the derivative of $F(x)$ for student scores that are one standard deviation above the mean.
2. The minimum score for students scoring in the top 10% is found by solving $0.10 = 1 - F(x)$. Use a root-finding method to find x.
3. Use a quadrature rule to evaluate $F(1157)$ — Appalachian State University's mean SAT score for entering first-year students in 2021.[2]
4. Use an interpolating polynomial to approximate $F(x)$ for $x \in [1100, 1500]$.

[2]From the *SAT Suite of Assessments Annual Report*, The College Board.
[2]From the College Admission Scores website.

A Bit Adder in Excel

Project: Adding Bits in Excel

1. Implement the one-bit adder in Excel using `IF-THEN` statements and `AND`, `OR`, and `NOT` functions.
2. Test your design with all 8 triples of 1-bit values.
3. Make an eight-bit adder with carries.
4. Test your design with 10 random pairs of 8-bit numbers.
5. Develop a mathematical model for the cost of computing the sum of two eight-bit values.



$F(a,b,c_0) = (s,c_1)$

a	b	c_0	s	c_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

One-Bit Adder with Carry

The Collatz Conjecture Project

Lothar Collatz's Proposition

Lothar Collatz posed a problem in 1937 that is quite simple to state, but that still eludes proof. For any positive integer n, define the sequence

$$a_1 = n \quad \text{and} \quad a_{k+1} = \begin{cases} \frac{1}{2}a_k & \text{if } a_k \text{ is even} \\ 3a_k + 1 & \text{if } a_k \text{ is odd} \end{cases} \text{ for } k > 1.$$

Collatz conjectured the sequence would always reach 1 no matter the starting value $n \in \mathbb{N}$.

Conjecture (Collatz's $3n + 1$ Problem)

For every $n \in \mathbb{N}$, there is a $k \in \mathbb{N}$ such that the sequence above has $a_k = 1$.

Currently...

The conjecture has been verified for all starting values up to $87 \cdot 2^{60} \approx 10^{20}$. Read "The $3x + 1$ Problem" by J. Lagarias (January 1, 2011 version) and check Eric Roosendaal's web site http://www.ericr.nl/wondrous/.

The Collatz Conjecture

Project

1. Write a program, Collatz, that determines the *total stopping time* of a starting value n. That is, given $a_1 = n$, find the first k such that $a_k = 1$. Define $\text{Collatz}(n) = k$.
2. Explain why $\text{Collatz}(2^m) = m$.
3. Generate a point-plot of the sequence $[n, \text{Collatz}(n)]$ for $n = 1$ to $100{,}000$.
4. Use your graph to find the maximum value of $\text{Collatz}(n)$ for n from 1 to $100{,}000$.
5. Which initial value $n \leq 10^6$ produces the largest total stopping time? (*Careful!* For example, $\text{Collatz}(159{,}487) = 183$, but before $a_{183} = 1$, this sequence hits $a_{67} = 17{,}202{,}377{,}752$, a value that very much overflows an unsigned 32-bit integer.)
6. Report on the history of Collatz's conjecture.

Extra for Experts: Prove that $\text{Collatz}(2^m n) = m + \text{Collatz}(n)$.

The CORDIC Algorithm

Background

In 1959, Jack Volder designed a way to compute trigonometric functions very quickly, the CORDIC algorithm,[3] while working on digitizing the navigation system of the B58 Hustler, the first Mach 2-capable supersonic bomber. During the '70s, John Walther extended CORDIC to compute exponentials, logs, and hyperbolic trigonometric functions. The algorithm became the standard for calculators (using BCD).

CORDIC Recurrence Equations

$$x_{k+1} = x_k - 2^{-k} m\, \delta_k\, y_k$$
$$y_{k+1} = y_k + 2^{-k} \delta_k\, x_k$$
$$z_{k+1} = z_k - \delta_k\, \sigma_k$$

where $m = 1$ (trig), 0 (arith), or -1 (hypertrig), δ_k is ± 1, and σ_k is a scaling factor.

[2] Jack Volder, "The CORDIC Computing Technique," 1959 Proceedings of the Western Joint Computer Conference, pp. 257–261; and,
—, "The Birth of CORDIC," J. VLSI Signal Proc. 25, 2000, pp. 101–105.

The CORDIC Parameters

Parameter Choices

	Rotation Mode $\delta_k = \operatorname{sgn}(z_k) \quad (z_k \to 0)$	*Vectoring Mode* $\delta_k = -\operatorname{sgn}(y_k) \quad (y_k \to 0)$
$m = 1$ $\sigma_k = \tan^{-1}(2^{-k})$	$\langle x_0, y_0, z_0 \rangle = \langle K, 0, \theta \rangle$ $x_n \to \cos(\theta);\ y_n \to \sin(\theta)$	$\langle x_0, y_0, z_0 \rangle = \langle x, y, 0 \rangle$ $z_n \to \tan^{-1}(y/x)$
$m = 0$ $\sigma_k = 2^{-k}$	$\langle x_0, y_0, z_0 \rangle = \langle x, 0, z \rangle$ $y_n \to x \times z$	$\langle x_0, y_0, z_0 \rangle = \langle x, y, 0 \rangle$ $z_n \to y/x$
$m = -1$ $\sigma_k = \tanh^{-1}(2^{-k})$	$\langle x_0, y_0, z_0 \rangle = \langle K', 0, \theta \rangle$ $x_n \to \cosh(\theta);\ y_n \to \sinh(\theta)$	$\langle x_0, y_0, z_0 \rangle = \langle x, y, 0 \rangle$ $z_n \to \tanh^{-1}(y/x))$
(some σ_k repeated) $(k = 4, 13, 40, 121, \ldots)$	$\langle x_0, y_0, z_0 \rangle = \langle K', 0, \theta \rangle$ $x_n + y_n \to e^{\theta}$	$\langle x_0, y_0, z_0 \rangle = \langle w+1, w-1, 0 \rangle$ $z_n \to \frac{1}{2}\ln(w)$

$$K = \prod_{j=0}^{n} \cos(\sigma_k), \qquad K' = \prod_{j=0}^{n} \cosh(\sigma_k)$$

CORDIC in Maple

CORDIC Trigonometric Functions with Maple

```
CORDIC[Trig] := proc(t)
 local n, K, x, y, z, j, del;
 n := 47;
 K := cos(arctan(1.0));
 for j to n-1 do
   K := K·cos(arctan(2.^(-j)))
   end do;
 (x[1], y[1], z[1]) := (K, 0, evalf(t));
 for j to n+1 do
   del := sign(z[j]);
   if del = 0 then del := 1 end if;
   x[j+1] := x[j] - del·y[j]·2.^(-j+1);
   y[j+1] := y[j] + del·x[j]·2.^(-j+1);
   z[j+1] := z[j] - del·arctan(2.^(-j+1));
   end do;
 return (fnormal([x[j], y[j], z[j]]));
 end proc:
```

The CORDIC Project

Project

1. Modify the Maple program `CORDIC[Trig]` to compute $\arctan(\theta)$.
2. Write a Maple program `CORDIC[HyperTrig]` that computes hyperbolic trigonometric functions using CORDIC.
3. Write a Maple program `CORDIC[Exp]` that computes the exponential function using CORDIC.
4. Write a Maple program `CORDIC[Ln]` that computes the logarithmic function using CORDIC.
5. Report on the complex number basis of the CORDIC algorithm.
6. Create a presentation on the history of the CORDIC algorithm.

Extra for Experts:
Write a single program that computes all possible CORDIC outputs.

The Cost of Computing a Determinant

Project: The Computation Cost of a Determinant

1. a. Define a function in Maple that produces an arbitrary square matrix:
   ```
   > M := n → Matrix(n,n, symbol=a):
   ```
   b. Define a *shortcut function* `det` for determinant:
   ```
   > det := A → LinearAlgebra[Determinant](A):
   ```
   c. Define a function for calculating the computation cost of a determinant, ignoring finding subscripts:
   ```
   > Cost := expr → subs(subscripts = 0, codegen[cost](expr)):
   ```
   d. Test your functions with:
   ```
   > A := M(2):
   > A, det(A), Cost(det(A));
   ```
2. Write a loop that computes the cost of a 1×1 determinant up to a 10×10 determinant. (*A 10×10 determinant can take nearly 20 minutes.*)
3. Develop a mathematical model for the cost of computing a determinant in terms of its dimension n.

Space Shuttle Acceleration

Space Shuttle Acceleration: The Situation[3]

The Space Shuttle had three phases from ignition to achieving orbit. The first phase began by the shuttle lifting off the launch pad using solid rocket boosters (SRBs) to accelerate extremely quickly. At approximately two minutes, the SRBs were separated and fell back to Earth.

The second phase began with SRB separation and lasted approximately 6.5 minutes. The shuttle speed increased to 17,500 mph — the speed needed to achieve orbit. (Note this speed is a good deal less than the Earth's escape velocity of 25,000 mph.)

The third phase began at about 9 minutes when the fuel external tank was jettisoned and the shuttle entered orbit.

NASA's data (next pg) lists the time, altitude, and velocity for STS-121 from July 4, 2006.

Project.

1. Using different divided difference formulas, compute and graph the shuttle's acceleration. Compare the results from each method.
2. Determine the maximum acceleration and its time.
3. Approximate the acceleration when the shuttle entered orbit at 9 minutes.
4. Explain why the shuttle's velocity cannot be computed from $\Delta Altitude/\Delta time$.

[3] Adapted from NASA's *"Space Shuttle Ascent."*

Space Shuttle Acceleration, II

The Data

Time (s)	Altitude (m)	Velocity (m/s)	Time (s)	Altitude (m)	Velocity (m/s)
0	0	0	280	105,321	2651
20	1,244	139	300	107,449	2915
40	5,377	298	320	108,619	3203
60	11,617	433	340	108,942	3516
80	19,872	685	360	108,543	3860
100	31,412	1026	380	107,690	4216
120	44,726	1279	400	106,539	4630
140	57,396	1373	420	105,142	5092
160	67,893	1490	440	103,775	5612
180	77,485	1634	460	102,807	6184
200	85,662	1800	480	102,552	6760
220	92,481	1986	500	103,297	7327
240	98,004	2191	520	105,069	7581
260	102,301	2417			

Commissioner Loeb's Demise

The Situation[4]

Commissioner Loeb was murdered in his office. Dr. "Ducky" Mallard, NCIS coroner, measured the corpse's core temperature to be 90°F at $8{:}00$ pm. One hour later, the core temperature had fallen to 85°F. Looking through the HVAC logs to determine the ambient temperature, Inspector Clouseau discovered that the air conditioner had failed at $4{:}00$ pm; the Commissioner's office was 68°F then. The log's strip chart shows Loeb's office temperature rising at 1°F per hour after the AC failure; at $8{:}00$ pm, it was 72°F.

Inspector Clouseau believes Snidely Whiplash murdered the Commissioner, but Whiplash claims he was being interviewed by Milo Bloom, staff reporter of the *Bloom Beacon*, at the time. Bloom's interview started at $6{:}30$ pm and lasted until $7{:}15$. Whiplash's lawyer, Horace Rumpole, believes he can prove Snidely's innocence.

[4]Adapted from *A Friendly Introduction to Numerical Analysis* by Brian Bradie.

Commissioner Loeb's Demise

First Steps

1. The office temperature is $T_{\text{ambient}} = 72 + t$, where $t = 0$ is $8:00$ pm.

2. *Newton's Law of Cooling* applied to the corpse's temperature gives
$$\frac{dT}{dt} = -k(T - T_{\text{ambient}}) = -k(T - t - 72) \text{ with } T_0 = 90.$$

3. A little differential equation work (with an *integrating factor*) yields
$$T(t) = \left[72 + t - \frac{1}{k}\right] + e^{-kt} \cdot \left[18 + \frac{1}{k}\right].$$

4. To find k, use the other data point. Set $T(1) = 85^\circ$F, then solve for k.

5. Last, solve $T(t_D) = 98.6$ for t_D, the time of death.

Commissioner Loeb's Demise

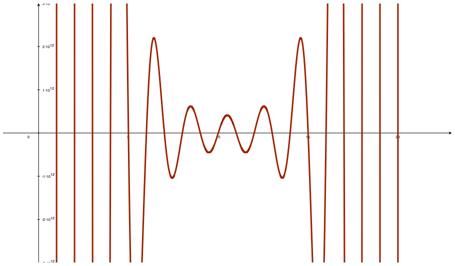
Project

1. Using the four methods, Bisection, Newton, Secant, and Regula Falsi:
   a. Find the value of k using First Step 4.
   b. Find t_d, the time of death of Commissioner Loeb, from First Step 5.
   c. What was the temperature of the Commissioner's office at t_D?
   d. How does error in the value of k effect error in the computation of t_D?
2. Compare the four methods and their results.
3. Graph $T(t)$ over the relevant time period.
4. Chief Inspector Charles LaRousse Dreyfus answers the press's questions:
   - *Is Inspector Clouseau right?*
   - *Could Snidely Whiplash have killed Commissioner Loeb?*
   - *Will Horace Rumpole get another client off?*
   - *Will Milo Bloom win a Pulitzer?*
   - *Will Bullwinkle finally pull a rabbit out of his hat?*

Wilkinson's Perfidious Polynomial

In 1963, James Wilkinson created a polynomial to illustrate numerical difficulties in computing roots.[5] His polynomial is:

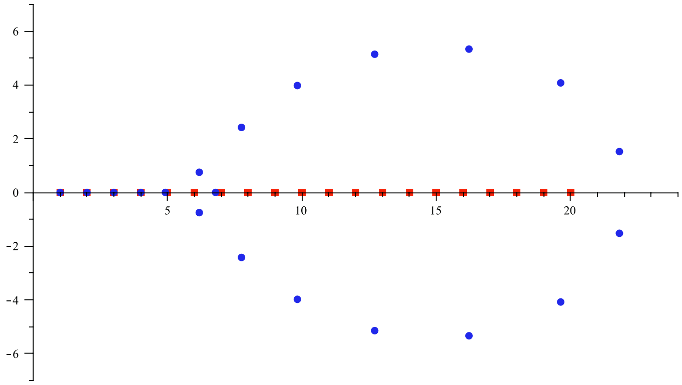
$$\begin{aligned} W(x) = \; & x^{20} \\ & -210\,x^{19} \\ & +20615\,x^{18} \\ & -1256850\,x^{17} \\ & +53327946\,x^{16} \\ & -1672280820\,x^{15} \\ & +40171771630\,x^{14} \\ & -756111184500\,x^{13} \\ & +11310276995381\,x^{12} \\ & -135585182899530\,x^{11} \\ & +1307535010540395\,x^{10} \\ & -10142299865511450\,x^{9} \\ & +63030812099294896\,x^{8} \\ & -311333643161390640\,x^{7} \\ & +1206647803780373360\,x^{6} \\ & -3599979517947607200\,x^{5} \\ & +8037811822645051776\,x^{4} \\ & -12870931245150988800\,x^{3} \\ & +13803759753640704000\,x^{2} \\ & -8752948036761600000\,x \\ & +2432902008176640000 \end{aligned}$$



Plot Window: $[-2,23] \times [-3\cdot 10^{12}, +3\cdot 10^{12}]$

[5]See: James H Wilkinson, "The Perfidious Polynomial," in *Studies in Numerical Analysis*, ed. G. Golub, 1984, MAA, pp. 3–28.

Wilkinson's Polynomial's Roots



Red box: root of $w(x)$
Blue circle: root of $w_p(x) = w(x) + 10^{-23}x^{19}$

Wilkinson's Polynomial's Roots

The Project

1. Describe what happens when trying find the root at $x = 20$ using Newton's method.
2. Describe what happens when trying find the root at $x = 20$ using Halley's method.
3. Discover what happens when the constant term is perturbed, i.e., investigate the roots of $p(x) = w(x) + 10^6$.
4. Discover what happens when the x^1 term is perturbed, i.e., investigate the roots of $p(x) = w(x) + x$.

Bernoulli's Method for Polynomial Roots

Bernoulli's Method (1728)[6]

Let $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ be a polynomial (*wolog* assuming p is monic) and let r be the root of p with the largest magnitude. If r is real and simple, define the sequence $\{x_k\}$ recursively by

$$\begin{aligned} x_k &= -a_{n-1}x_{k-1} - a_{n-2}x_{k-2} - \cdots - a_0 x_{k-n} \qquad k = 1, 2, \ldots \\ x_0 &= 1, \;\; x_{-1} = x_{-2} = \cdots = x_{-n+1} = 0. \end{aligned}$$

Then

$$\frac{x_{n+1}}{x_n} \to r.$$

- Bernoulli's method works best when p has simple roots and r is not "close" to p's next largest root.
- "If the ratio does not tend to a limit, but oscillates, the root of greatest modulus is one of a pair of conjugate complex roots." (Whittaker & Robinson, *The Calculus of Observations*, 1924.)

[6]Daniel Bernoulli, *Commentarii Acad. Sc. Petropol.* III. (1732).

Deflation

Deflating a Polynomial

Let $p(x)$ be a polynomial. If a root r of p is known, then the *deflated polynomial* $q(x)$ is

$$p_1(x) = \frac{p(x)}{x-r}.$$

The coefficients of p_1 are easy to find using synthetic division.

The Technique

1. Use Bernoulli's method to find r, the largest root of p
2. Deflate p to obtain p_1
3. Repeat to find all roots.

Problem: Since there is error in r's computation, there is error in p_1's coefficients. Error compounds quickly with each iteration.

The Bernoulli Project

The Project

1. Expand Wilkinson's "perfidious polynomial" into standard form
$$W(x) = \prod_{k=1}^{20}(x-k) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0.$$
2. Use 50 iterations of Bernoulli's method to find the largest magnitude root. What is the relative error?
3. Determine $W_1(x)$, the deflated polynomial using the root from 2.
4. Use 50 iterations of Bernoulli's method to find the largest magnitude root of $W_1(x)$. What is the relative error?
5. Determine $W_2(x)$, the deflated polynomial using the root from 4.
6. Use 50 iterations of Bernoulli's method to find the largest magnitude root of $W_2(x)$. What is the relative error?
7. Discuss the propagation of error in the deflations.

The Fourier Power Spectrum

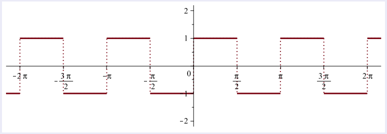
Calculating the Power Spectrum of a Signal

The *Power Spectrum* of a signal f gives the amount of power of a specific frequency component in the signal. The value at the frequency ω is given by

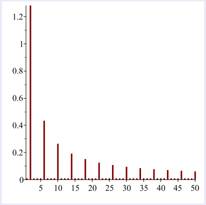
$$P(\omega) = \sqrt{a_\omega^2(f) + b_\omega^2(f)}$$

where $a_\omega(f)$ and $b_\omega(f)$ are the *Fourier coefficients* of f at the frequency ω. The Fourier coefficients a and b are computed with the integrals $(\omega \geq 1)$

$$a_\omega(f) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(\omega t)\,dt \quad \text{and} \quad b_\omega(f) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(\omega t)\,dt.$$



Square Wave $SW(t) = |\sin(2t)|/\sin(2t)$



Spectrum Graph of SW

The *Spectrum Plot* shows the power level at each frequency.

The Fourier Power Spectrum, II

The Project: Produce a Spectrum Plot of the Square Wave

We will use numerical integration to compute the spectrum of the square wave $SW(t) = |\sin(3t)|/\sin(3t)$.

1. Choose a method from: *midpoint, trapezoid, Simpson's*.
2. Use your method to numerically integrate $P(\omega)$ for $\omega = 1..100$.
3. Use Maple to create a list, *Spectrum*, of lines $[[k,0],[k,P(k)]]$ for $k = 1..100$.
4. Graph the list with *plot*$\big[\{$*Spectrum*$\}$, *thickness* $= 3\big]$.
5. Change the numerical integration method to either *Gauss quadrature* or *Gauss-Kronrod quadrature*.
6. Recompute the power spectrum and its graph. Do the spectrum values change? Is it faster or slower?

Extra for Experts

1. Show that $a(k) = 0$ for $k \geq 1$. (HINT: *Integrals of even & odd functions.)*
2. Then conclude that $P(k) = |b(k)|$ for $k \geq 1$.
3. Last, demonstrate that $b(k) = 0$ unless $k = 3 \pmod 6$.[7]

[7] Extended Project: Determine this condition for $SW_n(t) = |\sin(nt)|/\sin(nt)$.

Spline Fit to a Transition Curve

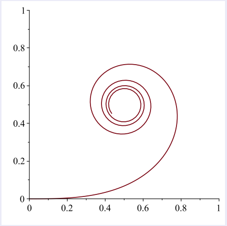
Transition Curves

A *transition curve* is a section of highway or railroad track used to go from a straight section into a curve. A transition curve is designed to prevent sudden changes in lateral acceleration, which would occur with a circular segment, by smoothly transitioning into and out of a curve. The basic transition curve parametric function $\mathscr{T}(t)$ uses the Fresnel sine and cosine integrals

$$\mathrm{C}(t) = \int_0^t \cos\left(\tfrac{1}{2}\pi t^2\right)\,dt \quad \text{and} \quad \mathrm{S}(t) = \int_0^t \sin\left(\tfrac{1}{2}\pi t^2\right)\,dt$$

to give $\mathscr{T}(t) = [\mathrm{C}(t), \mathrm{S}(t)]$. These integrals do not have elementary antiderivatives, and must be evaluated numerically.

A segment from the spiral $\mathscr{T}$ forms a transition curve which will provide a smooth transition without a sudden change in lateral acceleration.
Graph C and S to see their respective behaviors.

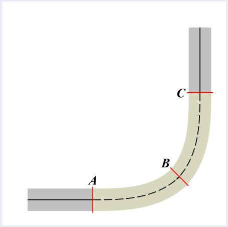


This project will investigate using splines to lay out a roadbed.

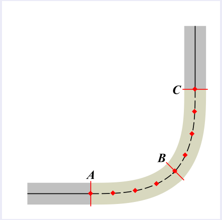
Spline Fit to a Transition Curve, II

Spline Fitting a Transition Curve for a Roadbed

Two roads are perpendicular and need to be connected by a curve. We'll blend two transition curves to make the road. One curve will start at A and turn toward the north until B. The second transition curve will start at C and turn west meeting the first at B. The two curves' tangents match at B.



We will compute waypoints along the centerline using the two transition curves, then fit a cubic spline through the data-points to lay out the roadbed to make surveying and construction reasonable. Use the data table below to determine the cubic spline.



Transition Curve Coordinates

$A = (0.0,\ 0.0)$	$(0.18,\ 0.003)$	$(0.35,\ 0.02)$
$(0.52,\ 0.08)$	$B = (0.66,\ 0.18)$	$(0.75,\ 0.30)$
$(0.80,\ 0.47)$	$(0.82,\ 0.65)$	$C = (0.83,\ 0.82)$

References

Selected References and Further Reading

- Brian Bradie, *A Friendly Introduction to Numerical Analysis*, Pearson, 2006
- Richard Burden, J. Douglas Faires, & Annette Burden, *Numerical Analysis*, 10th ed, Cengage Learning, 2015
- Laurent Demanet, *Introduction to Numerical Analysis*, MIT OpenCourseWare, 2012
- Francis Hildebrand, *Introduction to Numerical Analysis*, 2nd ed, Dover Publications, 1987
- James P Howard, II, *Computational Methods for Numerical Analysis with R*, CRC Press, 2017
- Bookauthority's "100 Best Numerical Analysis Books of All Time"

DOI: 10 1201/9781003299257-11

References

Selected Websites

- Wikipedia's "Computational mathematics"
- Springer's "Encyclopedia of Mathematics" "Computational mathematics"
- Simplicable's "What is Computational Mathematics?"
- University of Waterloo's "Welcome to Computational Mathematics"
- Council on Undergraduate Research's "Computational Mathematics: An Opportunity for Undergraduate Research" (pdf download)

- Perform a Google Scholar search for computational mathematics
- Search on Amazon.com for computational mathematics textbooks

The End

Whew. After all that, it's time to sit down. . .