Numerical Analysis: The Definitive Edition

Alexander Ng

Monday, April 14, 2025

Foreword

Hi reader, the following is a collection of notes I took while taking SFU's MACM 316 (Numerical Analysis) in Spring 2025. Many of the notes are heavily based on the lecture notes given by Professor Steven Ruuth, which can be found in this repository at `./steve's notes`. I have tried to make the notes as readable as possible, and every file is compiled in a machine-readable format so you can easily stick them into an AI knowledge base to help you with your studies.

If you need help getting back up to speed on Linear Algebra, I highly recommend taking a look at [An Infinitely Large Napkin](#) by Venhance. It's a great book that covers all of the theory of mathematics that you will need to understand the material in this course. Overtime, I will be updating these notes to be better structured based on the chapters of the textbook that the course is structured around.

Current Progress

Lectures 1 - 4 are cleaned up, corrected and usable to learn off of. I have separated out the sections into their appropriate chapters. I am currently working on refactoring the "Lecture 00n" files into "Chapter 00n" files. Next up, I will be cleaning up Lecture 5.

How to use this book

Until the notes are perfected, I recommend using them purely as a reference for studying the material. You can very easily use text search to find any specific information that you are looking for. When I wrote them, they were not meant to be used as a textbook, instead, they were meant to be thrown into ChatGPT and used as a knowledge base to help ChatGPT help you with your studies. Good luck!

Contents

Chapter 1

An Introduction to Numerical Analysis

> The first four lectures of this course are an introduction to numerical analysis and the main recurring concepts that will be used throughout the course.

Many real-world problems stem from numerical analysis, particularly poor execution. Rounding errors, insufficient representation problems, and other such problems represent the significant impact of computation in the real world.

Check out the following resources for more information:

1. https://www-users.cse.umn.edu/ arnold/disasters/

2. https://web.ma.utexas.edu/users/arbogast/misc/disasters.html

1.1   Computer Arithmetic

In Spring 2025, this was the start of Lecture 1.

We often want to work with the real number system, which consists of all integers, rational and irrational numbers

$$2, \sqrt{2}, e, \pi, 10^6, \text{ etc.}$$

Because we have a finite space limitation for numbers, **not all numbers can be represented exactly.** This can cause problems with arithmetic.

1.1.1   Bases - Binary and Decimal

We typically use the decimal (base 10) system, e.g.

$$427.325 = 4 \times 10^2 + 2 \times 10^1 + 7 \times 10^0 + 3 \times 10^{-1} + \dots$$

However, when we work with a computer, we use the binary (base 2) system:

$$(1001.11101)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + \dots$$

In this class, we will only be working in the decimal system in order to make computations simpler, since the application of concepts is identical.

1.1.2   Base Conversion and Error

Because it is impossible to represent some finite decimal fractions in binary, we will (definitely) encounter **error** when converting from base-10 to base-2. In the following example, we will convert $\frac{1}{10}$ to binary, and show how there exists some numbers which cannot be represented exactly within the binary floating-point system.

Example

To convert a decimal fraction like $\frac{1}{10}$ into its binary representation, we repeatedly multiply the fractional part by 2 and extract the integer part at each step.

We begin with $1/10 = 0.1_{10}$, and assume $0.1_{10} = (a_1.a_2a_3 \ldots a_n)_2$.

$$\frac{1}{10} = 0.1_{10} = (0.a_1a_2a_3 \ldots a_n)_2$$

Multiply by 2:

$$2 \times 0.1 = 0.2 = (a_1.a_2a_3 \ldots)_2$$

Keep the fractional part and repeat:

$$a_1 = \lfloor 0.2 \rfloor = 0$$
$$2 \times 0.2 = 0.4 \implies a_2 = \lfloor 0.4 \rfloor = 0$$
$$2 \times 0.4 = 0.8 \implies a_3 = \lfloor 0.8 \rfloor = 0$$
$$2 \times 0.8 = 1.6 \implies a_4 = \lfloor 1.6 \rfloor = 1$$
$$1.6 - 1 = 0.6$$
$$2 \times 0.6 = 1.2 \implies a_5 = \lfloor 1.2 \rfloor = 1$$
$$1.2 - 1 = 0.2$$

We have now returned to 0.2, which was the value after the very first multiplication. This means the process will repeat indefinitely. Thus, the binary representation of $\frac{1}{10}$ is:

$$\frac{1}{10} = (0.0\mathbf{0011}00110011 \ldots)_2$$

The repeating part is $\mathbf{0011}\ldots$, and this pattern continues forever. Using a finite number of digits, $\frac{1}{10}$ **cannot** be represented exactly in binary. Just as $\frac{1}{3} = 0.\overline{3}$ cannot be perfectly represented in decimal, the same is true for $\frac{1}{10}$ in binary. This is why certain decimal fractions lead to rounding errors in binary- based floating-point arithmetic.

1.2 Hypothetical Storage Scheme (32-bit)

We will use a hypothetical decimal computer since the concept is identical. (By the way, this is very close to IEEE-754 floating point representation, except that we are using a decimal representation instead of binary.)

Suppose we have the decimal number 423.7. Since we always want to represent numbers in proper scientific notation, we normalize the **mantissa**. We write our number as follows:

$$423.7 = + \underbrace{0.4237}_{\text{mantissa}} \times 10^{+3}$$

Notice the $+$ is relevant because we require an explicit representation of the sign of the number. We call the bits following from the decimal point (4273) the **mantissa**. We include 1 bit for the sign, which is 1 for positive numbers, 1 bit for the exponent sign, 7 bits for the exponent, and the remaining 23 bits for the mantissa.
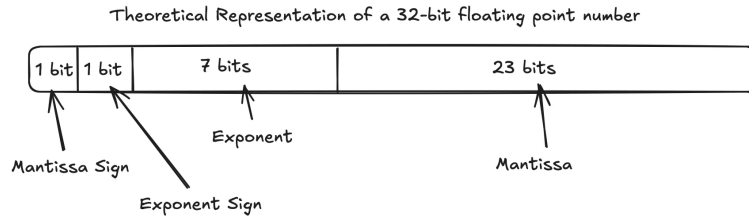

Figure 1.1: Hypothetical Storage Scheme (32-bit)

1.2.1 Problems with Floating Point

Because our storage format is finite, the biggest problems we will encounter are:

1. **bit overflow**: the maximum magnitude of our exponent (in binary) is **127**, so our number can only range from 2^{-127} to 2^{+127}.

2. **rounding error**: because our mantissa only has 23 bits of precision, the precision will decrease as our numbers get larger because we use exponentiation to represent the actual number.

Error Example

Consider the number $2^{25} = 33,554,432$. This number can be represented exactly in binary. However, the number $2^{25} + 1 = 33,554,433$ cannot be represented exactly, since it can't fit within the 23 bits of precision available.

From this, we find that all numbers (including fractions) from $2^{25} - 1$ through $2^{25} + 2$ are represented with the same mantissa in binary. Only when you reach $2^{25} + 3$ does the mantissa change.

Remarks

Within IEEE-754-style floating point representation, the number of representable values within a given exponent is the same, regardless of the exponent. This may seem obvious, but it's interesting nonetheless. This comes from the fact that the number of bits in the mantissa is fixed. The number of representable values is exactly $2 \times 2^{23} = 2^{24}$, since each positive value has a negative counterpart.

Actual IEEE-754 Floating Point

There are a few key differences betewen our hypothetical storage scheme and the actual IEEE-754 specification.

1. Instead of an exponent sign bit, IEEE-754 specifies a **biased exponent** which just stores the exponent as an 8-bit unsigned integer and interprets it as 2^{e-127}. **Ex.** we want 2^2 to be the exponent part of our number, so we set the biased exponent to $e = 2 + 127 = 129$. When we decode the number, we multiply the mantissa by $2^{129-127=2}$ to get the actual number.

2. The hidden bit (implicit leading 1) is a feature of IEEE-754 that allows us to "fake" 24 bits of mantissa precision. We save a bit by specifying the zeroth bit of the mantissa as always being a 1, and we can use this fact to save one bit of space.

In IEEE-754, the number is represented as

$$(-1)^s \times 1.f \times 2^{e-127}.$$

where s is the sign bit, f is the mantissa, and e is the biased exponent.

1.3   Floating Point Decimal Normalization

In Spring 2025, this was the start of Lecture 2.

Can we write all real numbers in normalized scientific notation? Well, the short answer is yes.

$$732.5051 \rightarrow +0.7325051 \times 10^{+3}$$
$$-0.005612 \rightarrow -0.5612 \times 10^{-2}$$

For $x \in \mathbb{R}$, we can express it as:

$$x = \pm r \times 10^{\pm n}, \quad \text{where } \frac{1}{10} \leq r \leq 1.$$

In binary, we write:

$$x = \pm q \times 2^{\pm m}, \quad \text{where } \frac{1}{2} \leq q < 1.$$

Here, q is the mantissa and m is the integer exponent.

Let b be the base. We limit r and q because when $k < 1/b$, we can shift the decimal place and normalize the number further. When we have $1.d_1$, we rewrite it as $0.1d_1 \times b^1$.

1.4   What if we have too many digits?

Oftentimes, we will be doing computations with a finite number of digits, but using operations that increase the number of digits. For example, if we multiply $\frac{1}{8} = 0.125$ (3 significant digits) by $\frac{1}{7} = 0.142857$ (5 significant digits) we get 0.017857125 (8 significant digits). This is a problem, since at a certain point, we will have too many digits to represent the number accurately. To solve this problem, we have rounding and truncation.

1.4.1   Rounding

Given $x = 0.a_1 a_2 \ldots a_n a_{n+1} \ldots a_m$ using m digits, rounding to n places follows:

- If $0 \leq a_{n+1} < 5$, then $x = 0.a_1a_2 \ldots a_n$.

- If $5 \leq a_{n+1} \leq 9$, then $x = 0.a_1a_2 \ldots (a_n + 1)$.

As you will see in the following example, this definition of rounding is different from the way we traditionally round numbers. We only consider the magnitude of the smallest significant digit, not the sign, so that when you round a number and its additive inverse, they will still cancel out.

Example

$$\text{round}(0.125) = 0.13,$$
$$\text{round}(-0.125) = -0.13.$$

Suppose, instead, that we define rounding classically, where we round to the nearest digit. Then, we would round 0.125 to 0.12 and -0.125 to -0.13, and adding $0.125 + (-0.125) = 0$ would result in $0.12 + (-0.13) = -0.01$, which is definitely wrong.

1.4.2 Chopping (Truncation)

Compared to rounding, truncation/chopping to n decimal places follows:

$$x = 0.a_1a_2 \ldots a_n a_{n+1} \ldots a_m,$$
$$\text{chop}_n(x) = 0.a_1a_2 \ldots a_n.$$

Truncation introduces larger errors but is computationally cheaper than rounding. In general, we prefer rounding because of the higher accuracy, because the computational cost is generally considered negligible.

1.5 Error

Because this class is called Numerical **Analysis**, we care a lot about the accuracy of our computations, and therefore we need to be able to quantify the error of the methods we use. Make sure to memorize the following definitions, especially if your professor does not allow you to have a formula sheet on the midterm(s)/final exam.

1.5.1   Definitions

- Actual error: $p - \hat{p}$. Use case: when the direction of the error matters (e.g. bias detection, error cancellation).

- Absolute error: $|p - \hat{p}|$. Use case: you only care about the magnitude of the error (e.g. reporting precision).

- Relative error: $\frac{|p-\hat{p}|}{|p|}$. Use case: we need a scale-independent measure of error (**i.e.** how big is the error relative to the magnitude of the value).

*The notes use p and p^* but I will use them interchangeably with p and $\hat{p}$.

1.5.2   Significant Digits

An approximation $\hat{p}$ has t significant digits if

$$\frac{|p - \hat{p}|}{|p|} \leq 5 \times 10^{-t}.$$

we call 5×10^{-t} the **error bound**.

1.5.3   Example

Our exact value is 0.1 and we approximate it with 0.099. The relative error is:

$$\frac{|0.1 - 0.099|}{|0.1|} = 0.01.$$

t	5×10^{-t}	error $\leq$ bound?
0	5	✓
1	0.5	✓
2	0.05	✓
3	0.005	✗

Since $0.01 < 5 \times 10^{-2}$ but not 5×10^{-3}, we have two significant digits.

1.6   Computations and Machine Representation

Let $\mathrm{fl}(x)$ denote the machine representation of x. Computations on a machine follow:

$$\mathrm{fl}(\mathrm{fl}(x) + \mathrm{fl}(y)).$$

Each step introduces an error.

1.6.1   Example

$$p = 0.54617, \quad q = 0.54601,$$
$$r = p - q = 0.00016.$$

With 4-digit rounding,

$$\hat{p} = 0.5462, \quad \hat{q} = 0.5460,$$
$$\hat{r} = \hat{p} - \hat{q} = -0.0002.$$

Relative error:

$$\frac{|r - \hat{r}|}{|r|} = 0.25.$$

A high relative error results when subtracting close numbers.

1.7   Roundoff Error

Consider computing $f(x) = \dfrac{1 - \cos x}{x^2}$ for $\bar{x} = 1.2 \times 10^{-5}$. With 10-digit rounding:

$$c = \mathrm{fl}(\cos \bar{x}) = 0.9999999999,$$
$$1 - c = 0.0000000001.$$

In this case, the roundoff error causes a **catastrophic cancellation**, which results in a large error. Instead, if we use an alternative formula, such as $\cos x = 1 - 2\sin^2(x/2)$:

$$f(x) = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2,$$

We obtain a more accurate computation.

Conclusion: Avoid subtracting close numbers. Use alternative representations like Taylor series, trigonometric identities or rationalized approximations.

1.7.1   Reducing Roundoff Error

In Spring 2025, this was the start of Lecture 3.

One way to reduce roundoff error is to minimize the number of floating-point operations.

Polynomial Evaluation Using Nested Multiplication

Consider evaluating the polynomial:

$$f(z) = 1.01z^4 - 4.62z^3 - 3.11z^2 + 12.2z - 1.99.$$

We can rewrite this expression using nested multiplication:

$$\begin{aligned}
f(z) &= (1.01z^3 - 4.62z^2 - 3.11z + 12.2)z - 1.99 \\
&= ((1.01z^2 - 4.62z - 3.11)z + 12.2)z - 1.99 \\
&= (((1.0z - 4.62)z - 3.11)z + 12.2)z - 1.99
\end{aligned}$$

By factoring out z as much as possible, we reduce the total number of floating-point operations, minimizing roundoff error accumulation.

1.8   Cancellation Errors

Otherwise known as catastrophic cancellation.

1.8.1   Quadratic Formula and Cancellation Errors

Consider solving the quadratic equation:

$$ax^2 + bx + c = 0.$$

Using the quadratic formula, the roots are:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad\qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Suppose $b = 600$, $a = c = 1$. Then we have

$$x_1 = \frac{-600 + \sqrt{600^2 - 4 \cdot 1 \cdot 1}}{2 \cdot 1}, \qquad x_2 = \frac{-600 - \sqrt{600^2 - 4 \cdot 1 \cdot 1}}{2 \cdot 1}$$

$$x_1 = \frac{-600 + \sqrt{359996}}{2}, \qquad x_2 = \frac{-600 - \sqrt{359996}}{2}$$

You'll notice that $\sqrt{359996} = 599.9966666574$ is very close to 600. This is a pretty big problem.

Since b is large and a, c are small, the term $\sqrt{b^2 - 4ac}$ is very close to b. This causes **catastrophic cancellation** in the computation of x_1, where subtracting two nearly equal quantities leads to significant loss of precision in floating-point arithmetic.

1.8.2 Reformulating to Reduce Cancellation

We rationalize the numerator:

$$x_1 = \frac{(-b + \sqrt{b^2 - 4ac})}{2a} \times \frac{(-b - \sqrt{b^2 - 4ac})}{(-b - \sqrt{b^2 - 4ac})}.$$

This simplifies to:

$$x_1 = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})} = \frac{2c}{-b - \sqrt{b^2 - 4ac}}.$$

Now, the cancellation error is eliminated. If, suppose, $b = -600$, the same issue would occur with x_2, and we could apply the same rationalization technique.

You should write this down on your cheat sheet.

1.9 Review of Taylor Series

Taylor's theorem is fundamental for numerical approximations.

1.9.1   Definition of Taylor Series

Given a function $f(x)$ that is sufficiently smooth on $[a, b]$, we can approximate $f(x)$ with a Taylor polynomial $P_n(x)$:

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

Here, $x_0, x \in [a, b]$.

1.9.2   Conditions for Taylor Series Expansion

For $f(x)$ to have a valid Taylor series expansion:

- $f \in C^n[a, b]$

  - This reads "f is continuous to the nth derivative on the interval $[a, b]$". *Remember this
  - **i.e.** f, f', f'', ..., f^n must be continuous

- $f^{(n+1)}$ must exist on $[a, b]$.

1.9.3   Error in Taylor Approximation

The error in a Taylor series approximation is given by:

$$f(x) = P_n(x) + R_n(x),$$

where the remainder term $R_n(x)$ satisfies:

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n + 1)!}(x - x_0)^{n+1},$$

for some $c \in (x_0, x)$. The approximation is most accurate when x is close to x_0.

1.9.4   Example: Third-Order Taylor Polynomial

Find $P_3(x)$ for $f(x) = \sin(x)$ centered at $x_0 = 0$.

$$P_3(x) = f(0) + f'(0)(x - 0) + \frac{f''(0)}{2!}(x - 0)^2 + \frac{f'''(0)}{3!}(x - 0)^3$$

$$= x - \frac{x^3}{6}.$$

Error Analysis

The remainder term for $n = 3$ is:

$$R_3(x) = \frac{f^{(4)}(c)}{4!}x^4.$$

substituting $f^{(4)}(x) = \sin(x)$,

$$R_3(x) = \frac{\sin(c)}{24}x^4.$$

and finally, taking $x = 0.1$:

$$|R_3(0.1)| \leq \frac{|\sin(0.1)|}{24}(0.1)^4 < 4.2 \times 10^{-7}.$$

A 10^{-7} relative error is more than enough to be considered high accuracy.

1.9.5 Example: Linear Approximation of $\sqrt{16.1}$

In Spring 2025, this was the start of Lecture 2.

We use P_1 (linear approximation) to find an approximation of $\sqrt{16.1}$ without using the square root algorithm.

Solution

Let $f(x) = \sqrt{x}$ and choose an expansion point $x_0 = 16$, since $\sqrt{16} = 4 \in \mathbb{Z}$ is easily computable. Using the first-order Taylor approximation:

$$f(x_0 + h) \approx f(x_0) + hf'(x_0),$$

where $h = 0.1$.

Computation

$$f(16) = \sqrt{16} = 4,$$
$$f'(x) = \frac{1}{2\sqrt{x}}, \quad f'(16) = \frac{1}{8},$$
$$f(16.1) \approx 4 + 0.1 \times \frac{1}{8} = 4.0125.$$

The exact value is $4.01248052955\ldots$, with a small truncation error. Since the machine error is on the order of 10^{-14}, it is negligible compared to the Taylor approximation error.

1.10   Algorithm Quantification

Numerical methods construct a sequence of better and better approximations, converging to a solution α.

Given a sequence $\{\alpha_n\}$:

$$\lim_{n \to \infty} \alpha_n = \alpha.$$

We quantify convergence speed by analyzing $|\alpha - \alpha_n| \leq c$, where c is a target error.

1.10.1   Example: Convergence of $\sin(1/n)$

Consider $\alpha_n = \sin(1/n)$, which converges to $\alpha = 0$ as $n \to \infty$. We rewrite:

$$\lim_{n \to \infty} \sin(1/n) = \lim_{h \to 0} \sin(h),$$

which is easier to analyze. Expanding $\sin(h)$ in a Taylor series:

$$\sin(h) = h - \frac{h^3}{3!} + \frac{h^5}{5!} + \ldots.$$

For small h, $\sin(h) \approx h$, implying:

$$|\alpha_n - \alpha| \leq \frac{1}{n}.$$

Thus, α_n converges to $\alpha = 0$ with rate of convergence $O(1/n)$.

1.11   Big-O Notation

For a sequence $\{A_n\}$, if:

$$|A_n - A| \leq k|B_n| \quad \text{for sufficiently large } n,$$

where k is a constant, then we say:

$$A_n = A + O(B_n).$$

1.11.1   Example: $\sin(1/n)$ Convergence

From before, $|\alpha_n - \alpha| \leq 1/n$, so:

$A_n = \sin(1/n)$ converges to $A = 0$ with rate of convergence $O(1/n)$.

1.11.2   Example: Convergence of $n\sin(1/n)$

We evaluate:
$$\lim_{n\to\infty} n\sin(1/n) = 1.$$

Changing variables, $h = 1/n$, we obtain:

$$\lim_{h\to 0} \frac{\sin(h)}{h} = 1.$$

Expanding $\sin(h)/h$ in Taylor form:

$$\frac{\sin(h)}{h} = 1 - \frac{h^2}{6} + O(h^4).$$

For small h:
$$\frac{\sin(h)}{h} - 1 \approx -\frac{h^2}{6},$$
so α_n converges to $\alpha = 1$ with rate $O(1/n^2)$.

1.11.3   Takeaways

Big-O notation quantifies algorithm efficiency by ignoring constants and focusing on convergence trends. Constants vary across systems, so we care about general convergence patterns rather than specific values.

1.12   Review

Why did we expand around $h_0 = 0$?

Because we want to know what happens with h_0, choosing 0 as our expansion point makes sense. The Taylor series approximation is more accurate the closer you are to your expansion point.

1.13   Another Example

Find the rate of convergence of the following $h \to 0$.

$$\lim_{h \to 0} \cos h + \frac{1}{2}h^2 = 1$$

$\alpha = 1$
$\alpha_h = \cos h + \frac{1}{2}h^2$

$$\alpha_h - \alpha = \cos h + \frac{1}{2}h^2 - 1$$
$$= 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + O(h^6) + \frac{1}{2}h^2 - 1$$
$$= \frac{h^4}{4!} + O(h^6)$$
$$= O(h^4)$$

Chapter 2

Direct Methods for Solving Linear Systems

<div style="background-color:#ccffcc">

Chapter 6 of the textbook.

</div>

In MATH 240, we learned methods for solving linear systems of equations where our $n \times m$ matrix has few rows and few columns. In Numerical Analysis, we will learn methods for solving linear systems where our matrix has thousands or millions of rows and columns.

We will first study methods that give an answer in a fixed number of steps, subject only to roundoff errors. This method only has error from the accumulation of numerical representation errors.

2.1 A Review of Linear Algebra

2.1.1 Linear Systems of Equations

Linear Systems of Equations

$$
\begin{bmatrix}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n
\end{bmatrix}
$$

2.1.2 Elementary Row Operations

1. Multiply row i by a constant $\lambda \neq 0$, denoted by $\lambda E_i \rightarrow E_i$

2. Add a multiple of row i to row j, denoted by $E_i + \lambda E_j \rightarrow E_i$

3. Interchange rows i and j, denoted by $E_i \leftrightarrow E_j$

2.1.3 Notes on Gaussian Elimination

The idea behind Gaussian Elimination is to transform the matrix into a 'triangular' equivalent matrix problem of the upper triangular or lower triangular form.

2.1.4 Example 1

$$
\begin{bmatrix}
1 & -1 & 2 & -1 & -8 \\
2 & -2 & 3 & -3 & -20 \\
1 & 1 & 1 & 0 & -2 \\
1 & -1 & 4 & 3 & 12
\end{bmatrix}
$$

2.2 Complexity of Gaussian Elimination

How does the number of operations change with the size of the matrices?

We can count up the number of multiplications and divisions to go to upper triangular form.

$$
\begin{bmatrix}
* & \cdots & * & x_1 \\
\vdots & \ddots & \vdots & \vdots \\
* & \cdots & * & x_n
\end{bmatrix}
\rightarrow
\begin{bmatrix}
* & * & \cdots & * & x_1 \\
0 & * & & \vdots & \vdots \\
\vdots & \vdots & & * & x_n \\
0 & * & \cdots & * & x_{n+1}
\end{bmatrix}
$$

This proceeds as follows:

Provided $a_{11} \neq 0$, the operation corresponding to

$$
E_j - \frac{a_{ji}}{a_{11}} E_i \rightarrow E_j
$$

2.3   Partial Pivoting

Parital pivoting is the simplest technique to avoid generating massive round-off errors in the Gaussian Elimination algorithm.

The idea is to find the largest element beneath the pivot and swap its row with the pivot row.

Parital pivoting is sufficient for most linear systems. However, it can be inadequate for certain problems.

2.3.1   Example

Consider the linear system:

$E_1 : 30.00x_1 + 591400x_2 = 491700$

$E_2 : 5.291x_1 - 6.130x_2 = 46.78$

No row exchanges are carried out during partial pivoting.

Now, the multiplier is $m_{21} = \frac{5.291}{30.000} = 0.1764$ and $(E_2 - m_{21}E_1 \to E_1)$ gives the system

$30.00x_1 + 591400x_2 \approx 591700$

$-104300x_1 \approx -104400$

2.4   Scaled Partial Pivoting

We can improve the accuracy of the partial pivoting algorithm if we scale coefficients before deciding on row exchanges.

The scaling factor is the largest absolute value of any coefficient in the current row. The idea is to select the largest scaled value a_{ik}/S_i corresponding to elements that are below the pivot.

The extra work to apply the partial pivoting algorithm is some constant times $O(n^2)$.

In rare instances, complete pivoting may be needed, which is $O(n^3)$ extra work.

2.4.1   An excerpt on Time Complexity

Total cost of partial pivoting is $O(n^3) + O(n^2) = O(n^3)$

Complete pivoting is $O(n^3) + O(n^3) = O(n^3)$

In the end, the dominant term of complete pivoting is $(c_1 + c_2)O(n^3)$, which is technically more than $c_1O(n^3)$ for partial pivoting, but it is the extra work is insignificant compared to the total work of the actual algorithm.

2.5   Some Review (Definitions)

1. Two matrices A and B are equal if they have the same size and if each element a_{ij} in A is equal to b_{ij} in B.

2. $A + B$ for two similarly sized matrices A and B is defined as the $n \times n$ matrix whose entries are $(a_{ij} + b_{ij})$ for all $i = 1..n$ and $j = 1..m$.

3. λA for a scalar λ and a matrix A is defined as the matrix whose entries are λa_{ij} for all $i = 1..n$ and $j = 1..m$.

2.6   Determinants

A very useful concept of linear algebra is the determinant of a matrix. The determinant of a matrix A is denoted by $\det(A)$ or $|A|$.

Determinants are important, in part, because of the following theorem:

2.6.1 Theorem

The following statements are equivalent for any $n \times n$ matrix A.

1. $\det(A) \neq 0$

2. The equatoin $Ax = 0$ has a unique solution $x = 0$.

3. The system $Ax = b$ has a unique solution for any n-dimensional column vector b.

4. The matrix A is nonsingular.
   i.e. A^{-1} exists.

2.6.2 Definition of the Determinant

The definition of the determinant is somewhat involved:

(a) If $A = [a]$, then $\det(A) = a$

(b) If A is some $n \times n$ matrix, the minor M_{ij} of A is the determinant of the $(n-1) \times (n-1)$ submatrix of A obtained by removing the ith row and j^{th} column.

(c) The <u>cofactor</u> A_{ij} of A associated with M_{ij} is defined by $A_{ij} = (-1)^{i+j} \det(M_{ij})$.

(d) The determinant of the $n \times n$ matrix A, when $n > 1$ is given either by

$$\det(A) = \sum_{j=0}^{n} a_{ij} A_{ij} \text{ (Row cofactor expansion) or}$$

$$\det(A) = \sum_{j=0}^{n} a_{ij} A_{ij} \text{ (Column cofactor expansion)}$$

2.6.3 Complexity of the Determinant Algorithm

Assume there are no free zero entries in A, and you must compute the fully expanded determinant of A.

Work (in general) for a 4x4 matrix

$$
\begin{aligned}
&= 4 \times (\text{work to compute } \det(3 \times 3)) \\
&= 4 \times 3 \times (\text{work to compute } \det(2 \times 2)) \\
&= 4 \times 3 \times 2 \times (\text{work to compute } \det(1 \times 1)) \\
&= 4 \times 3 \times 2 \times 1 \\
&= \boxed{4!}
\end{aligned}
$$

Which implies that the determinant of a $n \times n$ matrix can be computed in $O(n!)$ time.

2.7 More ways of computing the determinant

If $A = [a_{ij}]$ is an $n \times n$ matrix that is either upper or lower triangular form, then $\det(A) = \prod_{i=1}^{n} a_{ii}$.

2.8   LU-Decomposition

Suppose that we wanted to solve the system

$$Ax = b_k \qquad A \in \mathbb{R}^{n \times n}$$

for several different $b_k \in \mathbb{R}^n$.

This type of matrix problem arises frequently in initial value problems. If we apply Gaussian Elimination to solve the system, then $O(n^3)$ operations are required for each b_k. On the other hand, suppose we could factor A into the form $A = LU$ where L is lower triangular and U is upper triangular.

Then, we can let $y = Ux \implies Ly = b_k$, and solving for y via forward substitution is $O(n^2)$ operations.

Next, we have to solve for $Ux = y$ via backward substitution, which is also $O(n^2)$ operations.

The total number of operations is $O(n^2) + O(n^2) = O(n^2)$, which is much better than the $O(n^3)$ operations required for Gaussian Elimination.

Finding L and U is $O(n^3)$ requires $O(n^3) + O(n^3)$ operations **HOW-EVER**, by doing this expensive operation once, we can save a lot of operations when there are many b_k.

2.8.1   LU-Factorization

We will proceed to derive this LU-factorization using Gaussian Elimination.

Example:

$$\begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 39 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 14 \end{bmatrix}$$

We want to zero out entries below the pivot element a_{11} in the first row. So we want to take $E_2 - 3E_1 \to E_2$. This same effect can be obtained by multiplying A by the elementary matrix E_{21}.

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 39 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 4 & -8 & 30 \end{bmatrix}$$

An elementary matrix is equal to the edentity matrix except for one nonzero entry off of the main diagonal.

Now we want to zero out the remaining entry below the pivot.

This can be done by taking $E_3 - 2E_1 \to E_3$ or by left-multiplying A by the elementary matrix E_{31}.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 4 & -8 & 30 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & -4 & 24 \end{bmatrix}$$

To obtain an upper triangular system, we would finally zero out the remaining entry below the pivot in E_2, taking $E_3 - 4E_2 \to E_3$. Or, we can left multiply by the elementary matrix E_{32}.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & -4 & 24 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & 0 & 4 \end{bmatrix}$$

Notice that $E_{32}E_{31}E_{21}A$ is upper triangular.

Set $U = E_{32}E_{31}E_{21}$ and $L = A$.

Now $E_{32}^{-1}E_{31}^{-1}E_{21}^{-1}U = L$.

To find the inverses of elementary matrices, we have to recall their meaning.

For example, $E_{21} = \begin{vmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$ subtracts 3 times the first row from the second.

The inverse will need to add 3 times the first row to the second.

$$\implies (E_{21})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Which costs $O(1)$ operations to flip the sign of the single entry.

Furthermore, notice that

$L \cong E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}$

$$
= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 4 & 1 \end{bmatrix}
$$

So the matrix multiplication can be performed by putting the nonzero offdiagonal elements of the elementary matrices intot he appropriate positions in the matrix L.

This means that the matrix L is easily constructed during the Gaussian Elimination process just by storing the multipliers.

We conclude that

$$
\underbrace{\begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 30 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 4 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & 0 & 4 \end{bmatrix}}_{U}
$$

2.9 LU Decomposition (contd.)

THM: If Gaussian elimination can be performed without row exchanges, then the matrix A has a unique LU factorization where L is lower triangular and with all diagonal entries equal to 1 and U is an upper triangular matrix.

Once the matrix factorization is complete, the solution to

$$LUx = b$$

is found by first setting

$$y = Ux$$

then determining the vector y from $Ly = b$ using forward substitution. The variable x is found from $Ux = y$ using backward substitution.

Notice: in the theorem, **we need to be able to perform Gaussian Elimination without row exchanges**. This means that we can't use row exchanges to solve the system.

2.10 Matrix Factorization

If A is any nonsingular matrix, $Ax = b$ can be solved by Gaussian Elimination with the possibility of row exchanges. If we can find the row interchanges required to solve the system by Gaussian Elimination, then we can re-arrange the equations in an order that would ensure that no row interchanges are required.

$\implies$ There is a rearrangement of the equations that permits Gaussian Elimination without row exchanges.

But WHY does the rearrangement of the equations break LU decomposition?

2.11 Permutation Matrix

An $n \times n$ permutation matrix P is obtained by rearranging the rows of the identity matrix. This gives a matrix with precisely one nonzero entry in each row and column. The nonzero entries are all 1's.

On the other hand, multiplying A on the right by P will exchange the second and third **columns** of A.

We will be using the following two properties of permutation matrices:

1. If $K_1, ..., K_n$ is a permutation of the integers $1, ..., n$, and the permutation matrix $P = [p_{ij}]$ is defined by

$$p_{ij} = \begin{cases} 1 & i = K_j \\ 0 & \text{otherwise} \end{cases}$$

   Then, PA permutes the rows of A according to (BIGASS MATRIX MISSING HERE) data from chapter6-part2.pdf page 3.

2. If P is a permutation matrix, then P^{-1} exists and $P^{-1} = P^T$

Now our approach will be to left multiply the system

$$Ax = b$$

by the appropriate permutiation matrix P_1 so that the system

$$(PA)x = Pb$$

can be solved without row exchanges. Then PA can be factorized into

$$PA = LU$$

This tells us that

$$P^{-1}LUx = b$$
$$LUx = Pb$$

which can be solved rapidly for x.

Chapter 3

Iterative Techniques in Matrix Algebra

Chapter 7 of the textbook.

We are interested in solving large linear systems $Ax = b$. We can use certain properties of A, such as A being sparse (**i.e.** having a high percentage of zeroes), to reduce the amount of computational work required to solve the system. Unfortunately, the classic Gaussian Elimination does not include any computational optimizations that can take advantage of most of the special properties we are interested in. For this reason, we will be considering iterative techniques.

3.1   Speical Types of Matrices

Where can Gauassian Elimination be performe without row exchanges?

3.1.1   Strictly Diagonally Dominant Matrices

Def. An $n \times n$ matrix A is <u>strictly diagonally dominant</u> if

$$|a_{ii}| > \sum_{j=1;j\neq i}^{n} |a_{ij}| \text{ for all } i = 1, \ldots, n$$

Example

$$\begin{bmatrix} 3h & h & -h \\ 4 & 10 & 4 \\ 1 & 1 & -3 \end{bmatrix}$$

This matrix is strictly diagonally dominant when $h \neq 0$.

Thm. 1

A strictly diagonally dominant matrix A is nonsingular.

Thm. 2

Let A be a strictly diagonally dominant matrix. Then Gaussian Elimination can be performed on any linear system of the form $Ax = b$ to obtain its unique solution without row or column interchanges, and the computations are stable to the growth of roundoff error.

3.1.2   Symmetric, Positive Definite Matrices

Def. A matrix A is <u>positive definite</u> if

$$\forall x \neq 0 (x^T A x > 0)$$

Example

Find all values of α for which the matrix

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

is positive definite.

Ans.

$$x^T A x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= x_1^2 + x_2^2 + \alpha x_3^2 - 2x_1 x_3 + 2x_2 x_3$$

$$= (x_1 - x_3)^2 + (x_2 + x_3)^2 + (\alpha - 2)x_3^2$$

A is positive definite $\iff$ $\alpha > 2$.

Some necessary conditions for an $n \times n$ matrix to be positive definite u include:

(a) A is nonsingular

(b) $a_{ii} > 0$ for each $i = i...n$

(c) ???

(d) $(a_{ij})^2 < a_{ii}a_{jj}$ for each $i \neq j$

HOWEVER: These are not sufficient conditions for positive definiteness. they are necessary but not sufficient.

We would like necessary and sufficient conditions for a matrix to be positive definite.

3.1.3   Leading Principle Submatrices

A <u>leading principle submatrix</u> of a matrix A is a matrix of the form

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{bmatrix}$$

for some $1 \leq k \leq n$.

Based on this definition, we have the following theorem:

Thm. 3

A symmetrix matrix A is positive definite if and only if each of its leading principle submatrices has a positive determinant.

As it turns out, we don't need to carry out row exchanges when Gaussian Elimination is used on a symmetric, positive definite matrix.

Thm. 4

A symmetric matrix A is positive definite if and only if Gaussian Elimination withour row exchanges can be performed on the linear system $Ax = b$ with all the pivot elements positive. Moreover, in this case, the computations are stable with respect to the growth of roundoff error.

Corollary: The matrix A is symmetric positive definite if and only if A can be factored in the form LDL^T where L is lower triangular with 1's on its diagonal and D is a diagonal matrix with positive diagonal entries.

A modification of the LU factorization algorithm can be made to factor a symmetric positive definite matrix into the form

$$A = LDL^T$$

This LDL^T factorization only requires $\frac{n^3}{6} + n^2 - \frac{7n}{6}$ multiplications/divisions, and $\frac{n^3}{6} - \frac{n}{6}$ additions/subtractions. This is only half the number of operations as LU factorization.

A version of this algorithm can also be constructed for matrices that are symmetric but not positive definite.

Corollary 2: The matrix A is positive definite if and only if A can be factored int he form LL^T where L is lower triangular with nonzero diagonal entries. Once again, a modification of the LU factorization algorithm can be made. This method, called <u>Choleski's Algorithm</u>, factors a symmetric positive definite matrix into the form

$$A = LL^T$$

Choleski's Algorithm only requires $\frac{n^3}{6} + \frac{n^2}{2} - \frac{2n}{3}$ multiplications/divisions, and $\frac{n^3}{6} - \frac{n}{6}$ additions/subtractions, which is even less than the LDL^T factorization. However, for small n, Choleski's Algorithm may be slower because it requires n square roots to be computed.

3.1.4   Band Matrices

Another important class of matrices that arise in a wide variety of applications are <u>band matrices</u>. A band matrix is a matrix of the form

3.2   More Special Matrices

3.2.1   Band Matrices

Another important class of matrices that arise in a wide variety of applications are called band matrices. These matrices concentrate all their nonzero entries about the diagonal.

Definition

An $n \times n$ matrix is called a band matrix if there exist integers p and q such that $1 < p, q < n$ having the property that $a_{ij} = 0$ whenever $i + p \leq j$ or $j + q \leq i$.

The bandwidth of a band matrix is defined as $w = p + q - 1$. We subtract 1 from our count because we do not want to double count the diagonal.

We will focus on the important case of tridiagonal matrices, which are band matrices with $p = q = 2$, i.e., the matrix is tridiagonal if the nonzero entries are on the main diagonal and the diagonals above and below the main diagonal.

Suppose A can be factored into the triangular matrices L and U. Suppose that the matrices can be found in the form:

$$
L = \begin{bmatrix}
l_{11} & 0 & \ldots & 0 \\
l_{21} & l_{22} & & \\
& & \ddots & \\
0 & \ldots & l_{n,n-1} & l_{nn}
\end{bmatrix}
$$

$$
U = \begin{bmatrix}
u_{11} & u_{12} & \ldots & u_{1n} \\
0 & u_{22} & \ldots & u_{2n} \\
& & \ddots & \\
0 & 0 & \ldots & u_{nn}
\end{bmatrix}
$$

The zero entries of A are automatically generated by LU. Multiplying $A = LU$, we also find the following conditions:

1.

$$
a_{11} = l_{11}, \quad a_{i,i-1} = l_{i,i-1}, \quad i = 2, 3, \ldots, n
$$

2.

$$a_{ii} = l_{i,i-1}u_{i-1,i} + l_{ii}, \quad i = 2, 3, \ldots, n$$

3.

$$a_{i,i+1} = l_{ii}u_{i,i+1}, \quad i = 1, 2, \ldots, n-1$$

This system is straightforward to solve: (1) gives us l_{11} and the off-diagonal entries of L. (2) and (3) are used alternately to obtain the remaining entries of L and U. This solution technique is often referred to as **Crout Factorization**.

If we count up the number of operations, we find:

- $(5n - 4)$ multiplications/divisions - $(3n - 3)$ additions/subtractions

Crout Factorization can be applied to a matrix that is positive definite or one that is strictly diagonally dominant. See the text for another general case where it can be applied.

3.3   Iterative Techniques in Matrix Algebra

We are interested in solving large linear systems $Ax = b$.

Suppose the matrix A has a high ($> 99.9\%$) sparsity, i.e., most of the entries are zeros. We would like to take advantage of this spare structure to reduce the amount of computational work required. Unfortunately, Gaussin Elimination is often unable to take advantage of the sparse structure. For this reason, we consider iterative techniques.

3.4   Vector Norms

To estimate how well a particular iterative technique approximates the true solution, we will need some measurement of distance. This motivates the notion of the vector norm.

Def. A vector norm on $\mathbb{R}^n$ is a function $||\cdot||$ from $\mathbb{R}^n \to \mathbb{R}^n$ satisfying the following properties:

- $||x|| \geq 0$ for all $x \in \mathbb{R}^n$

- $||x|| = 0 \iff x = \mathbf{0}$

- $||\alpha x|| = |\alpha| ||x||$ for all $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$

- $||x + y|| \leq ||x|| + ||y||$ for all $x, y \in \mathbb{R}^n$

Def. 2. The l_2 or Euclidean norm of the vector x is given by

$$||x||_2 = \left\{ \sum_{i=1}^{n} x_i^2 \right\}^{1/2}$$

This represents the usual notion of distance.

Def. 3. The infinity or max norm of a vector x is given by

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

Def. 4. If $x, y \in \mathbb{R}^2$, then the l_2 distance between x and y is given by

$$||x - y||_2 = \left\{ \sum_{i=1}^{n} (x_i - y_i)^2 \right\}^{1/2}.$$

and the l_∞ distance between x and y is given by

$$||x - y||_\infty = \max_{i=1}^{n} |x_i - y_i|.$$

3.5 Sequences?

Iterative techniques generate a sequence of vectors.

Def. A sequence $\left\{x^k\right\}_{k=1}^{\infty}$ of vectors in $\mathbb{R}^n$ is said to converge to x with respect to the norm $|| \cdot ||$ if, given any $\epsilon > 0$, there exists an integer $N('\epsilon)$ such that

$$||x^{(k)} - x|| < \epsilon \text{ for all } k \geq N$$

The notation $N('\epsilon)$ is used to emphasize that N is dependent on ϵ, however, N is not a function of ϵ.

3.5.1 Thm.

The sequence of vectors $\left\{x^k\right\}$ converges to x in $\mathbb{R}^n$ with respect to $|| \cdot ||_\infty$ if and only if $\lim_{k\to\infty} x_i^{(k)} = x_i$. for each i.

3.5.2 Thm. 2

For each $x \in \mathbb{R}^n$

$$||x||_\infty \leq ||x||_2 \leq \sqrt{n}||x||_\infty$$

This theorem relates the infinity norm to the Euclidean norm, which is very useful in the context of iterative techniques.

Example 1

Prove that $x^{(k)} = \left(\frac{1}{k}, 1 + e^{1-k}, -\frac{2}{k^2}\right)^t$ is convergent w.r.t. $||\cdot||_2$.

We know $0 \leq ||x^{(k)} - x||_2 \leq \sqrt{3}||x^{(k)} - x||_\infty$

$$\lim_{k\to\infty} ||x^{(k)} - x||_\infty = 0 \implies \lim_{k\to\infty} ||x^{(k)} - x||_2 = 0$$

So, $\left\{x^{(k)}\right\}$ is convergent to x w.r.t. $||\cdot||_2$.

It can be shown that <u>all</u> norms on $\mathbb{R}^n$ are equivalent with respect to convergence.

i.e. If $||\cdot||_a$ and $||\cdot||_b$ are norms on $\mathbb{R}^n$, and $\left\{x^{(k)}\right\}_{k=1}^\infty$ has the limit x with respect to $||\cdot||_a$, then $\left\{x^{(k)}\right\}_{k=1}^\infty$ also has the limit x with respect to $||\cdot||_b$.

3.6   Matrix Norms

Def. A <u>matrix norm</u> on the set of all $n \times n$ matrices $(R^{n\times n})$ is a real-valued function $\overline{||\cdot||}$ defined on this set satisfying for all $n \times n$ matries A and B and all real numbers α :

1. $||A|| \geq 0$

2. $||A|| = 0 \iff A = 0$

3. $||\alpha A|| = |\alpha|||A||$

4. $||A + B|| \leq ||A|| + ||B||$

5. $||AB|| \leq ||A||||B||$

Def. A distance between two $n \times n$ matrices A and B is

$$||A - B||$$

3.6.1 Thm.

If $||\cdot||$ is a vector norm on $\mathbb{R}^n$, then

$$||A|| = \max_{||x||=1} ||Ax||.$$

is a matrix norm.

This is called the natural or induced matrix norm associated with the vector norm.

The following result gives a bound on the value of $||Ax||$:

3.6.2 Thm.

For any vector $x \neq 0$, matrix A, and any natural norm $||\cdot||$, we have

$$||Ax|| \leq ||A|| \cdot ||x||.$$

Notes on the Infinity Norm

The infinity norm is defined as

$$||x||_\infty = \max_i |x_i|$$

So, it's the maximum absolute value of every entry in x.

Example: $x = \begin{pmatrix} 1 \\ -2 \\ 1.5 \end{pmatrix}$

$||x||_\infty = 2$

Because the largest element (in magnitude) is -2.

3.7 Computing the Infinity Norm and the 1-Norm

Computing the ∞-norm of a matrix is straightforward:

3.7.1   Thm.

If $A = (a_{ij})$ is a $n \times n$ matrix, then

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

Example:

Find

$$\left\lVert \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \right\rVert_\infty$$

$$\sum_{j=1}^{n} |a_{1j}| = |2| + |-1| + |0| = 3$$

$$\sum_{j=1}^{n} |a_{2j}| = |-1| + |2| + |-1| = 4$$

$$\sum_{j=1}^{n} |a_{3j}| = |0| + |-1| + |2| = 3$$

$$\implies ||A||_\infty = \max\{3, 4, 3\} = 4$$

3.7.2 Visualizing the Infinity Norm

Images courtesy of ChatGPT. I got really confused by the concept of the infinity norm, so I asked the bot for help.
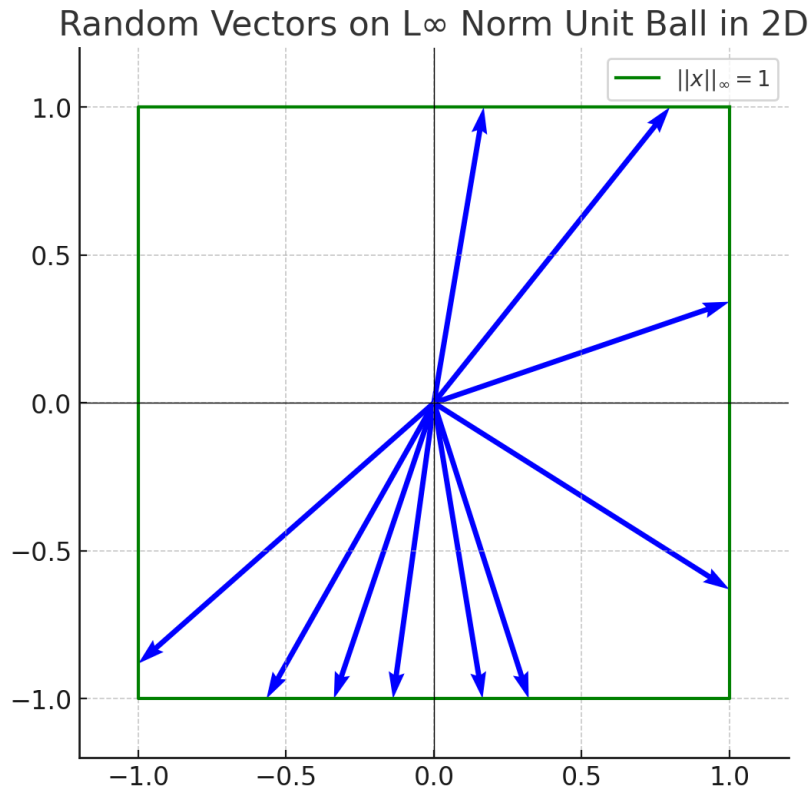

Figure 3.1: Visualizing the Infinity Norm

So the reason why the infinity norm, visualized this way, looks like a square, is because the equation $||x||_\infty = 1$ is equivalent to saying "the set of all vectors x such that the the largest component magnitude of the vector is 1". Meaning this is the set of all vectors that have $x = 1$ or $y = 1$

3.8 Eigenvalues and Eigenvectors

To calculate the l_∞-norm of a matrix, we did not need to directly appy the definition. This is also true for the l_2-norm, however, we will need to

introduce eigenvalues and eigenvectors to apply this technique.

First we will need the following definition:

Def. If A is a square matrix, the polynomial defined by

$$p(\lambda) = \det(A - \lambda I)$$

is called the <u>characteristic polynomial</u> of A. It is easily shown that p is an nth degree polynomial.

Now we can introduce eigenvalues and eigenvectors.

Def. If p is the characteristic polynomial of the matrix A, the zeros of p are called eigenvalues, or characteristic values of A. If λ is an eigenvalue of A and $x \neq 0$ has the property that $(A - \lambda I)x = 0$ then x is called an eigenvector, or characteristic vector, of A corresponding to the eigenvalue λ.

The professor does not specifically discuss eigenvectors and eigenvalues in the context of Numerical Analysis, but they will be important for future problems. He also does not mention how to compute them.

However, he did suggest that we review how to compute them by hand and study the related theorems.

3.9   Finding the l_2-norm of a matrix

Def. The spectral radius $\rho(A)$ of a matrix A is defined as

$$\rho(A) = \max |\lambda| \ \text{ where } \lambda \text{ is an eigenvalue of } A$$

ex.

$$\rho\left(\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \right) = \max \{|3|, |3|, |1|\} = 3$$

And now, we can consider the following:

Thm. If A is a $n \times n$ matrix, then

1. $||A||_2 = \sqrt{\rho(A^T A)}$

2. $\rho(A) \leq ||A||$ for any natural norm $|| \cdot ||$

My editor broke in the middle so you should look at the Chapter 7 PDF notes for the proofs and examples. The end of this section is around page 19 of the PDF. (35.13)

When we use iterative matrix techniques, we want to know when powers of a matrix become small.

Def. We call an $n \times n$ matrix A convergent if

$$\lim_{k \to \infty} \left(A^k \right)_{ij} = 0, \ \text{ for all } i, j$$

ex. Consider $A = \begin{bmatrix} \frac{1}{2} & 0 \\ 16 & \frac{1}{2} \end{bmatrix}$

$$A^2 = \begin{bmatrix} \frac{1}{4} & 0 \\ 16 & \frac{1}{4} \end{bmatrix}$$

$$A^3 = \begin{bmatrix} \frac{1}{8} & 0 \\ 12 & \frac{1}{8} \end{bmatrix}$$

$$A^4 = \begin{bmatrix} \frac{1}{16} & 0 \\ 8 & \frac{1}{16} \end{bmatrix}$$

$$A^k = \begin{bmatrix} \frac{1}{2^k} & 0 \\ P_k & \frac{1}{2^k} \end{bmatrix}$$

where $P_k = \begin{cases} 16 & k = 1 \\ \frac{16}{2^{k-1}} + \frac{1}{2} P_{k-1} & k > 1. \end{cases}$

Since $\lim_{k \to \infty} P_k = 0$, we also know that $\lim_{k \to \infty} P_k = 0$. $\therefore$ A is a convergent matrix.

Notice that this convergent matrix has a spectral radius (see Lecture 12 notes , page 5) less than 1.

This generalizes:

Thm. The following statements are equivalent:

1. A is a convergent matrix.

2. $\rho(A) < 1$

3. $\lim_{n \to \infty} A^n x = 0$ for every x

4. $\lim_{n \to \infty} ||A^n|| = 0$ for all natural norms $|| \cdot ||$

Iterative techniques convert the system $Ax = b$ into an equivalent system of the form $x = Tx + c$ where T is a fixed matrix and c is a vector. An initialv vector $x^{(0)}$ is chosen, and then a sequence of approximate solution vectors is generated:

$$x^{(k)} = Tx^{(k-1)} + c$$

Iterative techniques are rarely used in very small systems (i.e. when n^3 is small). In these cases, iterative techniques may be slower since they require several iterations to obtain the desired accuracy.

IDEA: It is possible to "split" the matrix A :

$$Ax = b$$
$$[M + (A - M)]\, x = b$$
$$Mx = b + (M - A)x$$
$$x = (I - M^{-1}A)x + M^{-1}b$$

Iteration becomes

$$x^{(k+1)} = (I - M^{-1}A)x^{(k)} + M^{-1}b$$

We set $T \cong I - M^{-1}A$ (the amplification matrix) and $c \cong M^{-1}b$.

$$x^{(k+1)} = Tx^{(k)} + c$$

How do we choose M?
We want:

1. M easy to "invert"

2. M "close to A" in the sense that $\rho(T)$ is small.

ex. Let $M = D = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{nn} \end{bmatrix}$

This gives the Jacobi Iterative Method.
In the text's notation,

$$A = D - L - U$$

Where D is diagonal, L is lower triangular, and U is upper triangular.

$$Ax = b$$
$$(D - L - U)x = b$$
$$Dx = (L + U)x + b$$
$$x = D^{-1}(L + U)x + D^{-1}b$$

Which results in the iteration

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$$

Let $T = D^{-1}(L + U)$ and $c = D^{-1}b$.

$$x^{(k+1)} = Tx^{(k)} + c$$

See example in the notes, there are too many matrices to type out in LaTeX. See "Chapter 7.pdf" page 27 (7-36.7)

Comments on Jacobi's Method

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$$

1. The algorithm requirs $a_{ii} \neq 0$ for $i = 1, \ldots, n$ If one of the $a_{ii} = 0$, and the system is nonsingular, then a reordering of the equations can be performed so that no $a_{ii} = 0$.

2. To speed convergence, the equations should be arranged such that $|a_{ii}$ is as large as possible.

3. A possible stopping criterion is to iterate until $\frac{||x^{(k)} - x^{(k-1)}||}{||x^{(k-1)}||} \leq \epsilon$

If we write out Jacobi's Method

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$$

we find that

$$x_i^{(k+1)} = \frac{\sum_{j=1;j\neq i}^n (-a_{ij}x_j^{(k)}) + b_i}{a_{ii}}$$

Notice that to compute $x_i^{(k+1)}$, the components $x_i^{(k)}$ are used. But, for $i > 1$, $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$ have already been computed and are likely better approximations to the actual solutions than

$$x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$$

So it seems reasonable to compute with these most recently computed values.

i.e.:

$$x_i^{(k+1)} = \frac{-\sum_{j=1}^{i=1}(a_{ij}x_j^{(k+1)}) - \sum_{j=i+1}^n(a_{ij}x_j^{(k)} + b_i}{a_{ii}}$$

This is called the Gauss-Seidel iterative technique, and it also has a matrix formulation with $M \cong (D - L)$:

$$Ax = b$$
$$(D - L - U)x = b$$
$$(D - L)x = Ux + b$$
$$x = (D - L)^{-1}Ux + (D - L)^{-1}b$$

$\implies$ iteration becomes

$$x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b$$

*Notice that $D - L$ is lower triangular. It is invertible $\iff$ each $a_{ii} \neq 0$

Lemma If the spectral radius $\rho(T)$ satisfies $\rho(T) < 1$ then $(I - T)^{-1} = I + T + T^2 + \ldots$

And we will prove the following theorem:

Thm. For any $x^{(0)} \in \mathbb{R}^n$, $\left\{ x^{(k)} \right\}_{k=0}^{\infty}$ the sequence defined by

$$x^{(k)} = Tx^{(k-1)} + c$$

converges to the unique solution of

$$x = Tx + c \text{ if and only if } \rho(T) < 1$$

Proof ($\Longleftarrow$).: assume $\rho(T) < 1$

$$
\begin{aligned}
x^{(k)} &= Tx^{(k-1)} + c \\
&= T(Tx^{(k-2)} + c) + c \\
&= T^2 x^{(k-2)} + (T + I)c \\
&\vdots \\
&= T^k x^{(0)} + (T^{k-1} + \cdots + T + I)c
\end{aligned}
$$

Since $\rho(T) < 1$, the matrix T is convergent and

$$\lim_{k \to \infty} T^k x^{(0)} = 0$$

Proof ($\Longrightarrow$).
HAS NOT BEEN WRITTEN DOWN YET
The *Lemma* implies that

$$\lim_{k \to \infty} x^{(k)} = \lim_{k \to \infty} T^k x^{(0)} + \lim_{k \to \infty} \left(\sum_{j=0}^{k-1} T^j \right) c = 0 + (1 - T)^{-1} c$$

$\Longrightarrow \left\{ x^{(k)} \right\}$ converges to the unique solution of $x = Tx + c$
i.e. $(I - T)x = c \implies x = (I - T)^{-1}c$
This allows us to derive some related results on the rates of convergence.

Corollary: If $||T|| < 1$ for any natural matrix norm and c is a given vector, then the sequence $\left\{ x^{(k)} \right\}_{k=0}^{\infty}$ defined by

$$x^{(k)} = Tx^{(k-1)} + c$$

converges for any $x^{(0)} \in \mathbb{R}^n$ to a vector $x \in \mathbb{R}^n$ and the following error bounds hold:

(i) $||x - x^{(k)}|| \leq ||T|^k||x^{(0)} - x||$

(ii) $||x - x^{(k)}|| \leq \frac{||T||^k}{-||T||}||x^{(1)} - x^{(0)}||$

Note, however, that $\rho(A) \leq ||A||$ for <u>any</u> natural norm. In practice,

$$||x - x^{(k)}|| \approx \rho(T)^k||x^{(0)} - x||$$

so it is desirable to have $\rho(T)$ as small as possible.

Some results for Jacobi's and Gauss-Seidel methods:

Thm. If A is strictly diagonally dominant, then for any choice of $x^{(0)}$, both the Jacobi and Gauss-Seidel methods give sequences $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ that converge to the unique solution of $Ax = b$.

No general results exist tot tell which of the two methods will converge more quickly, but the following result applies in a variety of examples:

Thm. Stein Rosenberge

If $a_{ij} \leq 0$ for each $i \neq j$ and $a_{ii} > 0$ for each $i = 1, 2, \ldots, n$, then exactly one of the following holds.

(a) $0 \leq \rho(T_g) < \rho(T_j) < 1$

3.10   Successive Over-Relaxation (SOR)

To define, suppose $\tilde{x}^{(k+1)}$ is the iterate from Gauss-Seidel using $x^{(k)}$ as the initial guess. The $(k + 1)^{st}$ iterate of SOR is defined by

$$x^{(k+1)} = \omega\tilde{x}^{(k)} + (1 - \omega)x^{(k)}$$

where $1 < \omega < 2$. It can be difficult to select ω optimally. Indeed, the answer to this question is not known for general $n \times n$ linear systems.

However, we do have the following results:

Thm. (kahan): If $a_{ii} \neq 0$ for each i, then

$$\rho(T_{SOR}) \geq |\omega - 1|$$

$\implies$ SOR can converge only if $0 < \omega < 2$

Thm. (ostrowski-reich): If A is a positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of initial approximate vector $x^{(0)}$

Thm.. If A is positive definite and tridiagonal, then

$$\rho(T_g) = [\rho(T_j)]^2 < 1$$

and the optimal choice of ω for the SOR method is

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(T_j)^2}}$$

with this choice of ω, we have $\rho(T_{SOR}) = \omega - 1$

Chapter 4

Solutions of Nonlinear Equations

We are interested in finding a root $x \in \mathbb{R}$ of an equation of the form $f(x) = 0$ for a given continuous function f. In most cases, it is impossible to solve this type of problem analytically, so we will consider iterative methods to approximate the solution.

4.1   The Bisection Method

We must start with an interval $[a, b]$ with $f(a)$ and $f(b)$ of opposite signs.

By the intermediate value theorem, there exists a number c in (a, b) such that $f(c) = 0$.

Thm. If $f \in C[a, b]$ and k is any number between $f(a)$ and $f(b)$, then there exists a number c in (a, b) such that $f(c) = k$.

Now set $a_1 = a$ and $b_1 = b$ and let p_1 be the midpoint of $[a_1, b_1]$.

1. Compute the midpoint:

$$p_1 = \frac{1}{2}(a_1 + b_1)$$

2. If $f(p_1) = 0$, then we are done. Set $p = p_1$.

3. If $f(p_1)$ and $f(a_1)$ are of opposite signs, then there must exist a root $p \in (a_1, p_1)$ such that $f(p) = 0$. Set $a_2 = a_1$ and $b_2 = p_1$.

4. Otherwise, if $f(p_1)$ and $f(b_1)$ are of opposite signs, then there must exist a root $p \in (p_1, b_1)$ such that $f(p) = 0$. Set $a_2 = p_1$ and $b_2 = b_1$.

5. Reapply the process to the new interval $[a_2, b_2]$.

6. Once the stopping criteria are satisfied, set the midpoint of the interval as the estimate for the root.

4.1.1   Possible Stopping Criteria

Every stopping criterion given here (except the first) are good stopping criteria for other methods as well.

1. $\dfrac{b_n - a_n}{2} < \text{ TOL} \quad \underline{\text{or}} \quad |p_n - p_{n-1}| < \text{ TOL}$

    GOOD: Ensures that the returned root value p_n is within tolerance of the exact value p

    GOOD: Easy error analysis

    BAD: Does not ensure that $f(p_n)$ is small.

    BAD: An absolute rather than relative measure of error.

2. $\dfrac{|p_n - p_{n-1}|}{p_n} <$ TOL, $p_n \neq 0$

   Usually preferred over (1) if nothing is known about $f(\cdot)$ or p

3. $|f(p_n)| <$ TOL

   Ensures that $f(p_n)$ is small, but p_n may differ significantly from the true root p.

4. We can also carry out a fixed number of iterations $N-$ This is closely related to (1)

The best stopping criteria will depend on what is known about f and p and on the type of problem. It's often useful to use criteria 2 (relative error test) and criteria 4 (fixed number of steps) together.

Bisection Method Example

He starts with an example on the bisection method. I will provide one later on.

Lowkey I missed notes from page 5.7 to 5.10 but I'll add them later. (page 5 of chapter 2 part 1 notes)

4.2   Fixed Point Iteration

We wish to find the roots of an equation $f(p) = 0$. We will be focusing on methods that <u>iterate</u> to find the root:

$$p_{n+1} = g(p_n)$$

We start by considering the fixed point problem.
Def. A fixed point p is the value of p such that $g(p) = p$.
We can form a fixed pt. problem from a root finding problem.

$$f(p) = 0$$

Find a fixed point problem.
e.g. set $g(x) = f(x) + x$
$g(p) = f(p) + p$
$p = g(p)$
there are many choices of g
Try $g(x) = f^3(x) + x$
Notice that fixed point problems and root finding problems are equivalent. (???)

4.2.1   Example

We are given $g(p) = p$. Formulate a root finding problem.

$$f(x) = g(x) - x$$

Now we have $f(p) = 0$. We now have a root finding problem.

$$f(p) = 0 \implies g(p) = p$$

i.e. f has a root p implies g has a fixed point p.

$$g(p) = p \implies f(p) = 0.$$

i.e. g has a fixed point p implies f has a root p.
There are many possible choices for g: <u>example</u> $g(x) - x = (f(x))^3$
Our ultimate goal is to find functions with fixed points.

4.3 Theorems

Existence and Uniqueness
    ⋆If $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$ then $g(x)$ has a fixed point in $[a,b]$.
    ⋆⋆ Suppose, in addition, that $g'(x)$ exists on (a,b) and that a positive constant $k < 1$ exists with

$$|g'(x)| \le k < 1 \text{ for all } x \in (a,b).$$

then the fixed point in $[a,b]$ is unique

Proof. (⋆): Existence
    If $g(a) = a$ or $g(b) = b$ then g has a fixed point at an endpoint.
    Suppose not, then it must be true that $g(a) > a$ and $g(b) < b$.
    Define $h(x) = g(x) - x$. Then h is continuous on $[a,b]$ (adding two continuous functions yields a continuous function) and

$$h(a) = g(a) - a > 0 \text{ and } h(b) = g(b) - b < 0.$$

The **IVT** implies that there exists a $p \in (a,b)$ for which $h(p) = 0$
This $g(p) - p = 0 \implies p$ is a fixed point of g

Proof. (⋆⋆): Uniqueness
    Suppose, in addition,

$$\forall x \in (a,b), |g'(x)| \le k < 1.$$

and that p and q are both fixed points in $[a,b]$ with $p \ne q$.
By the **MVT**, a number c eixsts between p and q such that

$$\frac{g(p) - g(q)}{p - q} = g'(c).$$

then

$$
\begin{aligned}
|p - q| &= |g(p) - g(q)| \\
&= |g'(c)||p - q| \\
&\leq k|p - q| \\
&< |p - q|
\end{aligned}
$$

Which is a contradiction

This contradiction must come from the assumption that $p \neq q$

$\therefore p = q$ and the fixed point is unique.

We want to approximate the fixed point of a function g.

IDEA

- choose an initial approximation p_0

- generate a squence $\{p_n\}_{n=0}^{\infty}$ such that $p_n = g(p_{n-1}); n \geq 1$

If the sequence converges to p and g is continuous;

$$
\begin{aligned}
p &\equiv \lim_{n \to \infty} p_n \\
&= \lim_{n \to \infty} g(p_{n-1}) \\
&= g(\lim_{n \to \infty} p_{n-1}) \\
&= g(p)
\end{aligned}
$$

This gives the Fixed Point Algorithm:

See next notes package

We want to convert from $0 = f(x)$ to $x = g(x)$, which is to convert from a root finding problem to a fixed point problem.

One way to do this, very simply, is to just add x to both sides of the equation.

$$x^3 + 4x^2 - 10 = 0$$
$$x^3 + 4x^2 - 10 + x = x$$

In general, if your iterative method converges very quickly, you will not have a guarantee of convergence. Therefore, you should use a mix of methods to get a good initial guess and then quickly converge to the fixed point.

4.4 Convergence

Why do some methods converge and some diverge? Why do they converge with different rates?

Consider a simple example:

ex.

$$g(x) = ax + b.$$

We have

$$x_1 = ax_0 + b$$
$$x_2 = ax_1 + b = a(ax_0 + b) + b = a^2 x_0 + (1 + a)b$$
$$x_3 = ax_2 + b = a^3 x_0 + (a + a + a^2)b$$

and by induction,

$$x_n = \begin{cases} a^n x_0 + \left(\frac{1-a^n}{1-a}\right)b & a \neq 1 \\ x_0 + nb & a = 1. \end{cases}$$

$$\therefore \lim_{n\to\infty} x_n = \begin{cases} \frac{1}{1-a}b & |a| < 1 \\ x_0 & a = 1, b = 0. \end{cases}$$

No proper limit exists for all other values of a, b.

4.5   Fixed Point Theorem

When does a fixed point iteration converge? How quickly does it converge?

For this, we turn to the **fixed point theorem**.

Thm. Let $g \in C[a, b]$ and suppose $g(x) \in [a, b]$ for all $x \in [a, b]$.

Suppose, in addition, that g' exists on (a, b) and a positive constant $k < 1$ exists with

$$|g'(x)| \leq k \text{ for all } x \in (a, b).$$

Then for any number p_0 in (a, b), the sequence defined by

$$p_n = g(p_{n-1}) \qquad n \geq 1.$$

converges to the unique fixed point p in $[a, b]$

Proof. Our earlier **Thm.** (existence + uniqueness) tells us that a unique fixed point exists in $[a, b]$. Notice that g maps $[a, b]$ into itself, so the sequence $\{p_n\}_{n=0}^{\infty}$ is defined for all $n \geq 0$ and $p_n \in [a, b]$ for all n.

We may apply the **mean value theorem** to g to show that for any n

$$\begin{aligned}
|p_n - p| &= |g(p_{n-1}) - g(p)| \\
&= |g'(c)||p_{n-1} - p| \\
&\leq k|p_{n-1} - p|
\end{aligned}$$

Where $c \in (a, b)$. Applying the inequality inductively gives

$$some garbage i haven't writ ten down yet.$$

Since $k < 1$, $\lim_{n \to \infty} |p_n - p| \leq \lim_{n \to \infty} k^n |p_0 - p| = 0$

$$\therefore \{p_n\}_{n=0}^{\infty} \text{ converges to } p.$$

This proof also gives us a natural bound for the error.

4.6 Newton's Method

(or Newton-Raphson Method)

One of the most powerful and well-known methods for solving a root-finding problem

$$f(x) = 0.$$

Pros: Much faster than bisection
Cons: Needs $f'(x)$, not guaranteed to converge
Want: $x = p$ s.t. $f(x) = 0$
Idea: Use slope as well as function values

4.6.1 Derivation (by Taylor's Thm.)

Want: $x = p$ s.t. $f(x) = 0$

Suppsoe $f \in C^2[a, b]$. Let $\overline{x} \in [a, b]$ be an approximation to p s.t. $f'(\overline{x}) \neq 0$ and $|\overline{x} - p|$ is sufficiently small. Then

$$f(x) = f(\overline{x}) + f'(\overline{x})(x - \overline{x}) + \frac{1}{2}f''(\xi(x))(x - \overline{x})^2 \qquad \xi \text{ lies between } x, \overline{x}.$$

Set $x = p$

$$0 = f(\overline{x}) + f'(\overline{x})(p - \overline{x}) + \frac{1}{2}f''(\xi(p))(p - \overline{x})^2.$$

$p - \overline{x}$ is very small $\implies |p - \overline{x}|$ is even smaller. So we just drop the error term $\frac{1}{2}f''(\xi(p))(p - \overline{x})^2$

$$0 \approx f(\overline{x}) + f'(\overline{x})(p - \overline{x}).$$

Solve for p:

$$\tilde{p} = \overline{x} - \frac{f(\overline{x})}{f'(\overline{x})}.$$

Take $p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$

As a stopping criterion, we might use

$$|p_n - p_{n-1}| \leq TOLERANCE \quad \epsilon.$$

Called "absolute error approximation".

We might also use "relative error approximation"

$$\frac{|p_n - p_{n-1}|}{|p_{n-1}|} \le \epsilon.$$

(1) Newton's method fails:

$$f'(p_n) = 0.$$

$\implies$ method is not effective if f' is equal to zero at p. It will also not perform well if f' is close to 0.

Also we see in the derivation that $|p - \bar{x}|$ needs to be small, which implies we need a good initial guess.

ex.:

Use Newton's Method to compute the square root of a number R. We want to find the roots of $p^2 - R = 0$.

Let

$$f(x) = x^2 - R$$
$$f'(x) = 2x$$

Newton's Method takes the form

$$\begin{aligned}
p_n &= p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \\
&= p_{n-1} - \frac{p_{n-1}^2 - R}{2p_{n-1}} \\
&= \frac{1}{2}(p_{n-1} + \frac{R}{p_{n-1}})
\end{aligned}$$

Method is credited to Heron, a Greek Engineer circa 100BC -> 100AD.

Try $R - 2$:

$$p_0 = 2$$
$$p_1 = 1.5$$
$$p_2 = 1.416666$$
$$p_3 = 1.41425162$$
$$p_4 = 1.414211356 \qquad \text{12 digits correct}$$

Newton's method can be shown to converge under reasonable assumptions (smoothness of $f(\cdot)$, $f'(p) \neq 0$ and a good initial guess)

Thm. (Convergence):

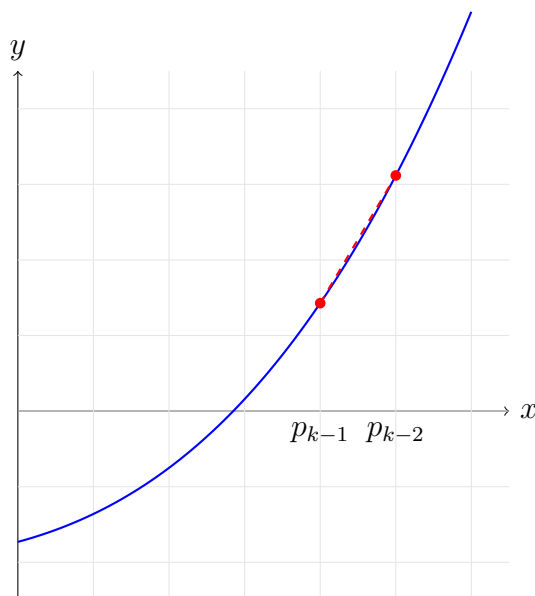## 4.7   Newton's Method

Needs $f'(p_k)$ to exist.

We might have that:

- f' is not available

- f' is expensive

We might choose to approximate:

$$f'(p_k) \approx \frac{f(p_{k-1}) - f(p_{k-2})}{p_{k-1} - p_{k-2}}.$$

This gives the secant method:

both newton's method and the secant method have the limitation that they may diverge when the initial guess is not sufficiently close to the root. In bisection, we used the idea of bracketing the root at each step to ensure convergence.

4.8 False Position

If instead of considering a midpoint approximation for the root
$(p_k \approx p_0 + \dfrac{p_1 - p_0}{2})$, we consider a secant approximation for the root (based on the endpoints of the interval). This is called the **Method of False Position**.

$$p_k \approx p_1 - f(p_1)\frac{p_1 - p_0}{f(p_0) - f(p_1)}.$$

4.9 oops, skipped a bunch of pages

4.10 Error Analysis

We want to be able to give a more precise description of how a method converges to a the solution. For example, consider finding a root for the polynomial

$$x^3 + 4x^2 - 10 = 0.$$

with two different methods (A and B). Suppose that the errors produced by these methods are as given below.

	Method A	**Method B**
$|p - p_0|$	0.134769987	0.134769987
$|p - p_1|$	0.078276245	0.008103332
$|p - p_2|$	0.037310791	0.000003811
$|p - p_3|$	0.019771639	0.000000000
$|p - p_4|$	0.009940240	to all significant digits

Table 4.1: Comparison of Methods A and B

Notice that the error for method A decreases by a constant factor (about 2) at each iteration. For method B, the error drops off much more quickly. The error at step n is roughly proportional to the error at step $n-1$, squared.
Both of these behaviours can be quantified.

Def. Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n. If positive constants λ and α exist with $\lim\limits_{n\to\infty} \dfrac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda$ then $\{p_n\}_{n=0}^{\infty}$ converges to p of order α, with asymptotic error constant λ.

Notice that:

- a sequence with high order of convergence converges more rapidly than a sequence with a lower order.

- the constant λ affects the speed of convergence, but is not as important as the order (α).

- We want a large α. $\alpha \geq 1$ is sufficient.
  If $\alpha = 0.5$, for example, then the error at k is proportional to the square root of the error at $k - 1$. This is not a good behaviour.

4.10.1   Common Cases

For different values of α, we observe the following convergence behaviors:

- $\alpha = 1$, $\lambda < 1$: Linearly convergent.

$$\mathcal{E}_{n+1} \approx \lambda \mathcal{E}_n$$

- $\alpha = 2$: Quadratically convergent.

$$\mathcal{E}_{n+1} \approx \lambda \mathcal{E}_n^2$$

- α does not have to be an integer. For example, the secant method has:

$$\alpha = \frac{1 + \sqrt{5}}{2} < 2.$$

In order to truly understand the behaviour of a method, we need to find both the order and the asymptotic error constant.

4.11 Thm (2.7 of Text)

Let $g \in C[a, b]$ s.t. $g(x) \in [a, b]$ for all $x \in [a, b]$.

Suppose, in addition, that g' is continuous on (a, b) and a constant $0 \leq k < 1$ exists with $|g'(x)| \leq k$ for all $x \in (a, b)$.

If $g'(p) \neq 0$, then for any number p_0 in $[a, b]$ the sequence

$$p_n = g(p_{n-1}) \text{ for } n \geq 1.$$

converges only linearly to the unique fixed point p in $[a, b]$.

Proof. We know from the fixed point theorem that the sequence converges to p. since g' exists on $[a, b]$ we can apply the mean value theorem to g:

$$\underbrace{g(p_n) - g(p)}_{p_{n+1} - p} = g'(\xi_n)(p_n - p).$$

where ξ_n is between p_n and p. Thus,

$$\frac{p_{n+1} - p}{p_n - p} = g'(\xi_n).$$

and fixed point iteration gives linear convergence with asymptotic error constant $|g'(p)|$ whenever $g'(p) \neq 0$.

4.12　Thm (2.7 of Text)

Let $g \in C[a, b]$ s.t. $g(x) \in [a, b]$ for all $x \in [a, b]$.

Suppose, in addition, that g' is continuous on (a, b) and a constant $0 \leq k < 1$ exists with $|g'(x)| \leq k$ for all $x \in (a, b)$.

If $g'(p) \neq 0$, then for any number p_0 in $[a, b]$ the sequence

$$p_n = g(p_{n-1}) \text{ for } n \geq 1.$$

converges only linearly to the unique fixed point p in $[a, b]$.

Proof. We know from the fixed point theorem that the sequence converges to p. since g' exists on $[a, b]$ we can apply the mean value theorem to g:

$$\underbrace{g(p_n) - g(p)}_{p_{n+1}-p} = g'(\xi_n)(p_n - p).$$

where ξ_n is between p_n and p. Thus,

$$\frac{p_{n+1} - p}{p_n - p} = g'(\xi_n).$$

and fixed point iteration gives linear convergence with asymptotic error constant $|g'(p)|$ whenever $g'(p) \neq 0$.

Proof. ...

Thus $\displaystyle\lim_{n \to \infty} \frac{|p_{n+1} - p|}{|p_n - p|} = |g'(p)|$ and fixed-point iteration gives linear convergence with asymptotic error constant $|g'(p)|$ whenever $g'(p) \neq 0$.

Method A was the fixed-point iteration method defined by the iteration function

$$g(x) = \frac{1}{2}(10 - x^3)^{1/2}.$$

Notice that:

$$g'(p = 1.365230013) = -\frac{3}{4}x^2(10 - x^3)^{-1/2}$$
$$= -0.51 \neq 0$$

so the theorem applies if we consider the interval $[1, 1.5]$ and we see that linear convergence is obtained. On the other hand, Method B was the fixed point iteration method defined by the iteration function

$$g(x) = x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x}.$$

This method gave quadratic convergence, but the theorem cannot be applied because

$$g'(p) = 0.$$

We saw last day that higher order convergence for fixed point method can occur only when $g'(p) = 0$. It is possible under certain reasonable conditions to obtain quadratic convergence...

4.13   Theorem (2.8 of Text)

Let p be a solution of the equation $x = g(x)$.

Suppose $g'(p) = 0$ and g'' is continuous and strictly bounded by M on an open interval I containing p. Then, there exists a $\delta > 0$ such that for $p_0 \in [p-\delta, p+\delta]$, the sequence defined by $p_n = g(p_{n-1})$ when $n \geq 1$ converges at least quadratically to p.

Moreover, for sufficiently large values of n,

$$|p_{n+1} - p| < \frac{M}{2}|p_n - p|^2.$$

Proof. ... (see lecture notes)

Thus the sequence $\{p_n\}_{n=0}^{\infty}$ converges quadratically if $g''(p) \neq 0$ and higher order convergent if $g''(p) = 0$. Also, we know $|g''| < M$ so

$$|p_{n+1} - p| < \frac{M}{2}|p_n - p|^2.$$

So the idea behind finding iteration methods with a high order of convergence is to look for schemes whose derivatives are zero at the fixed point.

4.14   Newton's Method

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

$$g'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2}$$

$$= \frac{f(x)f''(x)}{[f'(x)]^2}$$

$\therefore g'(p) = 0$ provided $f'(p) \neq 0$.

$\therefore$ Newton's Method satisfies the derivative condition for **Thm.** 2.8.

Let's take another look at Newton's Method. Consider using Newton's Method to find the roots of

$$p^3 - p^3 - p + 1 = 0.$$

Newton's Method here is

$$p_{n+1} = p_n - \frac{p_n^3 - p_n^2 - p_n + 1}{3p_n^2 - 2p_n - 1}.$$

Starting from $p_0 = 1.1$ we find

Iteration	Value
p_0	1.1
p_1	1.05116...
p_2	1.02589...
p_3	1.01303...
p_4	1.00653...
p_5	1.00327...
$\vdots$	$\vdots$

Table 4.2: Numerical Iterations

Which is very slow (Linear) convergence to the root (which is $p = 1$).
Why is this?
In Newton's Method, we need to find $f'(p) \neq 0$ to obtaine quadratic convergence. Notice that

$$f'(p) = 3p^2 - 2p - 1|_{p=1} = 0.$$

So the theorem doesn't hold. Moreover, factoring f:

$$f(x) = (x-1)^2(x+1).$$

we see that $x = 1$ is a zero with multiplicity of 2.
Def. A solution p of $f(x) = 0$ is a zero of multiplicity m of f if for $x \neq p$ we can write $f(x) = (x-p)^m q(x)$ where $\lim_{x \to p} q(x) \neq 0$.
Simple zeros are those that have multiplicity 1.

Thus Newton's Method can only be applied to simple zeros of a function. Identification fo the multiplicity of a zero is often made easier by the two following theorems.

Thm. 2.10

$f \in C'[a, b]$ has a simple zero at p in (a, b) if and only if $f(p) = 0$ but $f'(p) \neq 0$.

Thm. 2.11

The function $f \in C^m[a, b]$ has a zero of multiplicity m at p if and only if

$$0 = f(p) = f'(p) = f''(p) = \cdots = f^{(m-1)}(p).$$

but $f^{(m)}(p) \neq 0$.

We want to obtain quadratic convergence with Newton's Method for multiple roots.

One approach is to define a new function

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

We assume p is a zero of multiplicity m and $f(x) = (x - p)^m q(x)$ where $q(p) \neq 0$. Then,

$$\mu(x) = \frac{(x - p)^m}{m(x - p)^{m-1}q(x) + q'(x)(x - p)^m}$$

$$= \frac{(x - p)q(x)}{mq(x) + q'(x)(x - p)}$$

$$= (x - p)\frac{q(x)}{mq(x) + q'(x - p)}$$

$q(p) \neq 0$ therefore, $\mu(p)$ has a simple root at $x = p$

so $\mu(p) = 0$, but $\dfrac{q(p)}{mq(p) + q'(p)(p - p)} = \dfrac{1}{m} \neq 0$. and p is a zero of multiplicity 1 of $\mu(x)$.

Good:

- Quadratic convergence for all roots

Bad:

- Need f''

- μ is more expensive to work with

- μ might give more roundoff error

Newton's Method can only be applied to simple zeros of a function. Identification fo the multiplicity of a zero is often made easier by the two following theorems.

Thm. 2.10

$f \in C'[a, b]$ has a simple zero at p in (a, b) if and only if $f(p) = 0$ but $f'(p) \neq 0$.

Thm. 2.11

The function $f \in C^m[a, b]$ has a zero of multiplicity m at p if and only if

$$0 = f(p) = f'(p) = f''(p) = \cdots = f^{(m-1)}(p).$$

but $f^{(m)}(p) \neq 0$.

We want to obtain quadratic convergence with Newton's Method for multiple roots.

One approach is to define a new function

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

We assume p is a zero of multiplicity m and $f(x) = (x - p)^m q(x)$ where $q(p) \neq 0$. Then,

$$\mu(x) = \frac{(x - p)^m}{m(x - p)^{m-1}q(x) + q'(x)(x - p)^m}$$

$$= \frac{(x - p)q(x)}{mq(x) + q'(x)(x - p)}$$

$$= (x - p)\frac{q(x)}{mq(x) + q'(x - p)}$$

$q(p) \neq 0$ therefore, $\mu(p)$ has a simple root at $x = p$

so $\mu(p) = 0$, but $\dfrac{q(p)}{mq(p) + q'(p)(p - p)} = \dfrac{1}{m} \neq 0$. and p is a zero of multiplicity 1 of $\mu(x)$.

Good:

- Quadratic convergence for all roots

Bad:

- Need f''

- μ is more expensive to work with

- μ might give more roundoff error

We return to finding the roots of

$$p^3 - p^3 - p + 1 = 0.$$

Apply:

	Newton's Method	Modified Newton's Method
p_0	1.1	1.1
p_1	1.05116...	0.997735...
p_2	1.02589...	0.999999...
p_3	1.01303...	
p_4	1.00653...	
p_5	1.00327...	

This table shows that Newton's Method has linear convergence, and the Modified Newton's Method has quadratic convergence.

4.15 Congervence Speed and Acceleration

Suppose that we are given a linearly convergent sequence, and that we want to speed up the convergence. We want to analyze the behaviour of the error and use this knowledge to greatly reduce the error.

For example, we saw last day that the iterator method with

$$p_{n+1} = g(p_n)$$
$$\text{and } g(x) = \frac{1}{2}\sqrt{10 - x^3}$$

gives only linear convergence and the limit $p = 1.3652$

Notice that the <u>ratio</u> of the errors is fairly constant- we can use this idea to accelerate the convergence of the method.

Iteration	$p - p_n$
$p - p_0$	$-0.13476\ldots$
$p - p_1$	$+0.07827\ldots$
$p - p_2$	$-0.03710\ldots$
$p - p_3$	$+0.01977\ldots$
$p - p_4$	$-0.00994\ldots$

Suppose $\dfrac{p_{n+1} - p}{p_n - p} \approx \dfrac{p_{n+2} - p}{p_{n+1} - p}$

$$\therefore p \approx \frac{p_n p_{n+1} - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n}.$$

or,

$$p \approx p_n - \frac{p_{n+1}^2 - 2p_n p_{n+1} + p_n^2}{p_{n+2} - 2p_{n+1} + p_n}.$$

which is derived by adding and subtracting p_n to the right hand side.

The corresponding sequence

$$\hat{p}_{n+1} = p_n - \left[\frac{p_{n+1}^2 - 2p_n p_{n+1} + p_n^2}{p_{n+2} - 2p_{n+1} + p_n} \right].$$

is known as Aitken's Method.

Applying Aitken's method to our previous sequence,

Fixed Point Iteration		Aitken's Method	
Iteration	$p - p_n$	Iteration	$\hat{p} - p_n$
$p - p_0$	$-0.13476\ldots$	$\hat{p} - p_0$	0.00090088
$p - p_1$	$+0.07827\ldots$	$\hat{p} - p_1$	0.00023088
$p - p_2$	$-0.03731\ldots$	$\hat{p} - p_2$	0.00006107
$p - p_3$	$+0.01977\ldots$	$\hat{p} - p_3$	0.00001592
$p - p_4$	$-0.00994\ldots$	$\hat{p} - p_4$	0.00000418

we notice that Aitken's Method is much faster. It can be shown that the following theorem holds:

4.16   Theorem (2.13 of Text)

Suppose that $\{p_n\}$ is a sequence that converges linearly to the limit p and that for all sufficiently large values of n we have $(p_n - p)(p_{n+1} - p) > 0$.

Then the sequence $\{\hat{p}_{n=0}^\infty\}$ converges faster than $\{p_n\}_{n=0}^\infty$. to p in the sense that $\lim\limits_{n\to\infty} \dfrac{\hat{p}_n - p}{p_n - p} = 0$

The theorem does not apply to alternating sequences, but as we saw from our example it is often very useful even there for accelerating convergence.

Finally, we remark that this method is often written using difference operations to simplify the notation.

... sorry i missed a bunch here

4.17   Zeros of Polynomials

We want to compute the zeros (roots) of polynomials. A polynomial of degree n has the form

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

where $a_i, i = 0, \ldots, n$ are constants and $a_n \neq 0$.

4.17.1   Fundamental Theorem of Algebra

If P is a polynomial of degree $n \geq 1$, then $P(x)$ has at least one (possibly complex) root.

Corollary:

If $P(x)$ is a polynomial of degree $n \geq 1$, then there exist unique constants $x_1, x_2, \ldots, x_k$ (possibly complex), and positive integers $m_1, m_2, \ldots, m_k$ such that

$$\sum_{i=1}^{k} m_i = n$$

and

$$P(x) = a_n(x - x_1)^{m_1}(x - x_2)^{m_2} \ldots (x - x_k)^{m_k}.$$

Corollary:

Let P and Q be polynomials of degree at most n. If $x_1, x_2, \ldots, x_{n+1}$ with $n + 1 > n$ are distinct numbers with

$$P(x_i) = Q(x_i) \quad \text{for } i = 1, 2, \ldots, n + 1,$$

then

$$P(x) = Q(x) \quad \text{for all } x.$$

We want to use Newton's Method to locate the approximate roots of P. It will be necessary to evaluate P and its derivative at specified values. We now direct our attention to efficient methods for this task. The idea is to use nesting to evaluate an arbitrary n^{th} degree polynomial using only

$$n \text{ multiplications and } n \text{ additions.}$$

For illustration, consider $n = 4$. To evaluate $P(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, we write:

$$P(x) = (((a_4x + a_3)x + a_2)x + a_1)x + a_0.$$

Algorithmically,

1. set $b_4 = a_4$

2. set $b_3 = b_4 x + a_3$

3. set $b_2 = b_3 x + a_2$

4. set $b_1 = b_2 x + a_1$

5. set $b_0 = b_1 x + a_0$

Now b_0 gives the value $P(x)$.
For general polynomials of degree n :

$$b_n = a_n$$
$$b_k = a_k + b_{k+1} x \qquad 0 \le k \le n - 1$$
$$\text{and } b_0 = P(x)$$

we also want $P'(x)$. Differentiate each of the steps listed above, keeping in mind that b_i is a function of x. We get:

$$b_n' = 0$$
$$b_k' = b_{k+1}' x + b_{k+1}$$
$$\text{and } b_0' = P'(x)$$

Relabel: $c_{n+1} = b_n'$. Then an efficient method for computing $P'(x)$ is

$$c_n = a_n$$
$$c_k = c_{k+1} x + b_k$$
$$\text{and then } c_1 = P'(x)$$

This is called Horner's Method for evaluating a polynomial.

4.18   Horner's Method

Honer's Method has another useful property.

Consider $P(x)$

$$
\begin{aligned}
&= a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \\
&= b_n x^n + (b_{n-1} - b_n x_0) x^{n-1} + \cdots + (b_1 - b_2 x_0) x + (b_0 - b_1 x_0) \\
&= (b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1) x \\
&\quad - (b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1) x_0 \\
&\quad + b_0 \\
&= (b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1)(x - x_0) + b_0
\end{aligned}
$$

This property is useful because it gives us a way to find the next approximate zero after we have found our first zeros.

i.e. If x_0 is a root $P(x_0) = b_0 = 0 \implies Q(x)(x - x_0) = P(x)$
so we can find the next root by examining roots of $Q(x)$.

Suppose we want additional roots of P. If x_0 is an approximate root of P,

$$
P(x) \approx Q(x)(x - x_0).
$$

and the next step is toa pply Newton's Method to

$$
Q(X) = \frac{P(x)}{(x - x_0)}.
$$

this process is called deflation.

Chapter 5

Approximation and Interpolation

Chapter 3 of the textbook.

This is the section about splines! You are going to love splines.

5.1   Approximation and Interpolation

It is often useful or necessary to approximate a complicated or expensive function, or a function only known at a discrete set of points, by a smpler function which can be computed or evaluated more easily over a whole interval. When the function in question is known accurately at a discrete set of points, we are inclined to use an interpolation procedure- where the graph of the approximating function runs exactly through the points of the discrete set.

If the dataset is expected to contain error, which is the case for measurements or observations in experimental studies, a better strategy is to allow the graph of the approximating function to stray from the data points.

A useful and well known class of functions for mapping the set of real numbers into itself is the class of algebraic polynomials.

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

where n is a non-negative integer and a_i are real constants. Polynomials have the desirable property that they can approximate <u>any</u> function over a closed, bounded interval.

This desired property is precisely captured by the **Weierstrass Approximation Theorem**

5.1.1 The Weierstrass Approximation Theorem

Suppose $f \in C[a, b]$.
  $\forall \mathcal{E} > 0 \exists P \in \{\mathbb{P}_n, C[a, b]\}$ such that

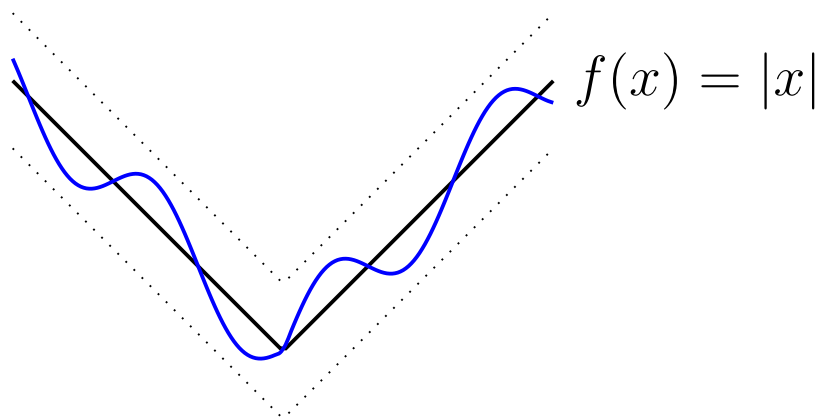$$|f(x) - P(x)| < \epsilon \forall x \in [a, b].$$

Figure 5.1: Polynomial approximation to $f(x) = |x|$.

This is a very strong theorem, as it only requires $f(x)$ to be continuous on the interval, and not necessarily differentiable.

Unfortunately, the Weierstrass Approximation Theorem does not tell us how to select such a polynomial. Many would immediately jump to using Taylor series polynomials for polynomial interpolation, however, Taylor series' concentrate their accuracy at the point a rather than over the entire interval, and are typically poorly suited for interpolation.

Taylor Series Polynomials

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

A particularly clear demonstration of this drawback is seen for

$$f(x) = \frac{1}{x} \qquad \text{expanded about } x_0 = 1.$$

Then,

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(1)}{k!}(x-1)^k$$

$$= \sum_{k=0}^{n} (-1)^k (x-1)^k$$

To approximate $f(3) = \frac{1}{3}$ by $P_n(3)$ for increasing values of n, we see a dramatic and catastrophic failure:

n	0	1	2	3	4	5	6	7
$P_n(3)$	1	-1	3	-5	11	-21	43	-85

Table 5.1: Values of $P_n(3)$ for increasing n.

We will insetad focus on methods which use information throughout the entire interval to approximate f.

5.2 Polynomial Interpolation

We now assume that the given dataset is exact and represents values of some unknown function. We want to find the polynomial $P_n(x)$ of the smallest possible degree n such that

$$P_n(x_k) = f_k \qquad k = 1, 2, \ldots, N.$$

for $N+1$ distinct interpolation points $x_0, \ldots, x_N$ and $N+1$ values $\underbrace{f_0, \ldots, f_N}_{\text{data points}}$.

To solve this problem, we will first investigate the simpler problem where all the data equals zero, except at one point.

We are looking for a polynomial $L_m(x)$ of degree $\leq N$ such that

$$L_m(x_k) = \delta_{mk} \qquad = \begin{cases} 1 & k = m \\ 0 & k \neq m \end{cases}.$$

Kronecker Delta

This is easy to find. Since the polynomial must vanish at the points $x_k, k \neq m$, it must contain the factors
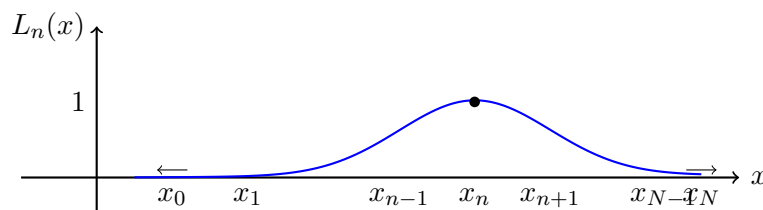
$$(x - x_k) \qquad \text{for } k \neq m.$$

$$\therefore L_m(x) = c \cdot \prod_{k=0;k\neq m}^{N} (x - x_k).$$

The constant is determined by the condition $L_m(x_m) = 1$.

$$\implies L_m(x) = \prod_{k=0;k\neq m}^{N} \frac{x - x_k}{x_m - x_k}.$$

Typically,

These polynomials are the building blocks for deriving a polynomial interpolating a general function. It is easy to see that

$$P(x) = \sum_{m=0}^{N} f_m < L_m(x).$$

is a polynomial of degree $\leq N$ satisfying the interpolation conditions.

This polynomial is called the nth **Lagrange Interpolating Polynomial**

Recall that we wanted to find the polynomial $P_n(x)$ of smallest degree such that

$$P_n(x_k) = f_k \qquad k = 0, \ldots, N.$$

5.2.1   Uniqueness

Is this polynomial unique?

Yes.

Proof. Assume there are two different polynomials p and q of degree $\leq N$ which both satisfy the interpolation conditions. Their difference, $d = p - q$, is also a polynomial of degree $\leq N$ and vanishes at the $N + 1$ distinct points $x_0, \ldots, x_N$.

However, a nonzero polynomial of degree $\leq N$ has at most N zeros, this

$$d = p - q = 0 \implies \begin{cases} p = q \\ \text{uniqueness} \end{cases}$$

$\therefore$ the nth Lagrange Interpolating Polynomial is the unique interpolating polynomial satisfying the interpolation conditions.

5.2.2   Example

Fit a cubic through the first four points of the table

i	x^i	$f(x_i)$
0	3.2	22.0
1	2.7	17.8
2	1.0	14.2
3	4.8	38.3
4	5.6	51.7

and use it to find the interpolated value for $x = 3.0$.

Soln. The 3rd Lagrange Interpolating Polynomial is given by

$$\begin{aligned} P(x) = {} & \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} f_0 \\ & + \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} f_1 \\ & + \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} f_2 \\ & + \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} f_3 \end{aligned}$$

Substituting the values from the table and evaluating at $x = 3.0$ gives

$$P(3.0) = \frac{(3.0 - 2.7)(3.0 - 1.0)(3.0 - 4.8)}{(3.2 - 2.7)(3.2 - 1.0)(3.2 - 4.8)}(22.0)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 1.0)(3.0 - 4.8)}{(2.7 - 3.2)(2.7 - 1.0)(2.7 - 4.8)}(17.8)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 2.7)(3.0 - 4.8)}{(1.0 - 3.2)(1.0 - 2.7)(1.0 - 4.8)}(14.2)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 2.7)(3.0 - 1.0)}{(4.8 - 3.2)(4.8 - 2.7)(4.8 - 1.0)}(38.3)$$
$$= 20.21$$

Our next task is to develop estimates for the error. As it turns out, the form of the error (but not necessarily the magnitude) resembles that of the nth Taylor Polynomial.

5.2.3   Error Estimates

The lecture ended just before this section

Thm. (3.3 of Text)

Suppose $x_0, \ldots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then for each $x \in [a, b]$, a number $\xi(x) \in (a, b)$ exists with the property

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n).$$

n-th Langrange Interpolating Polynomial
*do you guys like my diagrams with arrows?*

5.3   Continued from Lecture 20

Our next task is to develop estimates for the error. As it turns out, the form of the error (but not necessarily the magnitude) resembles that of the nth Taylor Polynomial.

5.3.1   Error Estimates

Thm. (3.3 of Text)

Suppose $x_0, \ldots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then for each $x \in [a, b]$, a number $\xi(x) \in (a, b)$ exists with the property

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \ldots (x - x_n).$$

n-th Langrange Interpolating Polynomial

Recall: If f has $(n + 1)$ continuous derivatives on $[a, b]$ and $P(x)$ is the interpolating polynomial of degree $\leq n$ for f at the points $x_0, \ldots, x_n$, then,

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{k=0}^{n}(x - x_k).$$

Ex. suppose you need to construct six-decimal-place tables for the common, or base-10, logarithm function from $x = 1$ to $x = 10$ in a way that linear interpolation is accurate within 10^{-6} of the true value. Determine a bound for the step size for this table.

Based on the following data:

i	x_i	$f(x_i)$
0	3.2	22.0
1	2.7	17.8
2	1.0	14.2
3	4.8	38.3
4	5.6	51.7

find approximations to $f(3)$ using the 2nd and 3rd Lagrange interpolating polynomials.

Soln. we will use x_0, x_1 and x_3 to build the 2nd Lagrange interpreting polynomial.

$$P_2(3) = \frac{(3 - 2.7)(3 - 4.8)}{(3.2 - 2.7)(3.2 - 4.8)}(22.0)$$
$$+ \frac{(3 - 3.2)(3 - 4.8)}{(2.7 - 3.2)(2.7 - 4.8)}(17.8)$$
$$+ \frac{(3 - 3.2)(3 - 2.7)}{(4.8 - 3.2)(4.8 - 2.7)}(38.3)$$
$$\approx 20.27$$

We will use x_0, x_1, x_2, x_3 to build the 3rd Lagrange interpolating polynomial.

$$P_3(3) = \frac{(3.0 - 2.7)(3.0 - 1.0)(3.0 - 4.8)}{(3.2 - 2.7)(3.2 - 1.0)(3.2 - 4.8)}(22.0)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 1.0)(3.0 - 4.8)}{(2.7 - 3.2)(2.7 - 1.0)(2.7 - 4.8)}(17.8)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 2.7)(3.0 - 4.8)}{(1.0 - 3.2)(1.0 - 2.7)(1.0 - 4.8)}(14.2)$$
$$+ \frac{(3.0 - 3.2)(3.0 - 2.7)(3.0 - 1.0)}{(4.8 - 3.2)(4.8 - 2.7)(4.8 - 1.0)}(38.3)$$
$$\approx 20.21$$

Notice that:

1. We do not know the derivative values of f, therefore we cannot apply the error formula. However, we can make an estimate for the error by examining polynomials of different degrees by using different nodes.

2. The P_2-calculation was not used to reduce the work in calculating P_3. We want to find a way to use previous degrees of P, especially since the previous point implies that we will examine the results for Lagrange polynomials of varying degrees.

We want to examine polynomials based on different nodes. In the last example, we considered the polynomial based on the nodes x_0, x_1 and x_3

We will call this polyomial $\mathbf{P_{013}(x)}$

We also considered the polynomial based on the nodes x_0, x_1, x_2, x_3

We will call this polyomial $\mathbf{P_{0123}(x)}$

Similarly, we make the following definition:

Let f be a function defined at $x_0, x_1, x_2, \ldots, x_n$ and suppose that $m_1, m_2, \ldots, m_k$ are k distinct integers with $0 \le m_i \le n$ and for each i.

The Lagrange Interpolating Polynomial that agrees with f at $x_m, x_{m2}, \ldots, x_{mk}$ is denoted

$$P_{m_1, m_2, \ldots, m_k}.$$

Using this notation,

$$P_0(x) = f(x_0)$$

$$P_1(x) = \frac{(x - x_1)f(x_0) + (x - x_0)f(x_1)}{x_0 - x_1}$$

and

$$P_{0,1}(x) = \left(\frac{x - x_1}{x_0 - x_1}\right) f(x_0) + \left(\frac{x - x_0}{x_1 - x_0}\right) f(x_1)$$
$$= \frac{(x - x_1)P_0(x) - (x - x_0)P_1(x)}{x_0 - x_1}$$

So $P_{0,1}(x)$ can be recursively defined in terms of $P_0(x)$ and $P_1(x)$.

More generally:

**Thm.** Let f be defined at $x_0, x_1, \ldots, x_k$ and x_j and x_i be two distinct numbers in this set. Then

$$P_{0,1,\ldots,k}(x) = \frac{(x - x_j)P_{0,1,\ldots,j,j+1,\ldots,n}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,\ldots,n}(x)}{x_i - x_j}.$$

Proof. I left out the proof.

The corresponding procedure is called Neville's Method. Here, values for each interpolating polynomial are generated using previous calculations.

Ex. $P_{0,1}(x) = \dfrac{(x - x_1)P_0 - (x - x_0)P_1}{(x_0 - x_1)}$ is derived from $P_0 + P_1$. Correspondingly, $P_{1,2}(x)$ is derived from P_1 and P_2.

Written as a table:
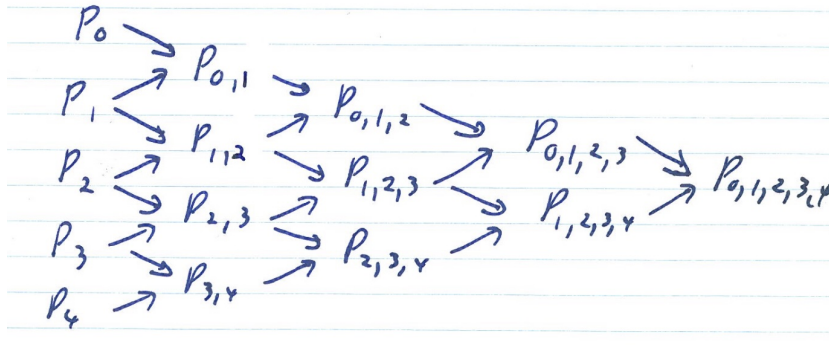
Ex. suppose $x_j = j$ for $j = 0, 1, 2, 3$ and it is known that

$$P_{0,1}(x) = 2x + 1$$
$$P_{0,2}(x) = x + 1$$
$$P_{1,2,3}(2.5) = 3$$

Find $P_{0123}(2.5)$.

We have P_{123} and we need another quadratic to find our P_{0123}

$$P_{0,1,2} = \frac{(x - x_1)P_{02}(x) - (x - x_2)P_{01}(x)}{x_2 - x_1}.$$

We evaluate $P_{02}(2.5) = 2.25$ and $P_{01}(2.5) = 6$
We have $x_1 = 1, x_2 = 2, x = 2.5$

$$P_{012}(2.5) = 2.25.$$

$$P_{0123}(2.5) = \frac{(2.25 - x_0)P_{123}(2.25) - (2.25 - x_3)P_{012}(2.25)}{x_3 - x_0} = 2.875$$

5.4   Review

5.4.1   Polynomial Interpolation

We have data for a function f aat $x_0, x_1, \ldots, x_n$

5.4.2   Lagrange Interpolation

$$\sum_{j=0}^{n} f(x_j) \underbrace{L_j(x)}_{0 \text{ at } x_i \text{ when } i \neq j} .$$

Reminder: a degree n polynomial interpolant for $n + 1$ points is unique. This means any polynomial interpolation method will give the same polynomial, just in a different form.

5.4.3   Divided Differences

Divided differences is an easy way to add data points.

Assume we know $f(x)$ at several values for x.

The x_i points do <u>not</u> need to be evenly spaced, or in any particular order.

We choose to represent our degree n interpolating polynomial as follows:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \ldots (x - x_{n-1}).$$

For every term we add, we interpolate another data point. We choose a_i such that $P_n(x) = f(x)$ at the points $x_0, x_1, \ldots, x_n$.

The coefficients a_i are determined by divided differences

$$P_n(x_0) = f(x_0).$$

$$a_0 = f(x_0) = f[x_0].$$

Define the zeroth divided difference

$$f[x_j] = f(x_j).$$

$$a_0 + a_1(x_1 - x_0) = f(x_1).$$

$$f[x_0] + a_1(x_1 - x_0) = f[x_1].$$

$$a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0}.$$

Define first divided difference

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

$$a_0 = f[x_0].$$

$$a_1 = f[x_0, x_1].$$

$$a_2 = f[x_0, x_1, x_2].$$

Define the second divided difference

$$f[x_i, x_{i+1}, x_{i+2}]$$
$$= \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

Define the kth divided difference for $x_i, \ldots, x_{i+k}$

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

$$a_k = f[x_0, x_1, \ldots, x_k].$$

This is Newton's interpolating divided difference formula.

$$P(x) = f[x_0] + f_[x_0, x_1](x - x_0)$$
$$+ f[x_0, x_1, x_2](x - x_0)(x - x_1)$$
$$+ \ldots$$
$$+ f[x_0, x_1, \ldots, x_n](x - x_0) \ldots (x - x_{n-1})$$

To compute an extra datapoint, we only need to compute one more term:

$$P_n(x) = P_{n-1}(x) + f[x_0, x_1, \ldots, x_n](x - x_0) \ldots (x - x_{n-1}).$$

Note that while using this method, we do NOT need to have our x_i points in any particular order.

Example

$$(x_0, \ldots, x_4) = (0.3, 1.0, 0.7, 0.6, 1.9).$$

Note that our points are not in order, or evenly spaced.
We are going to interpolate

$$f(x) = 2x^3 - x^2 + x - 1.$$

We can use a table!
Divided differences table:

$x_1 = 0.3$ —— $f[x_1] = -0.736$
    2.48
$x_2 = 1.0$ —— $f[x_2] = 1.0$
    3.68
$x_3 = 0.7$ —— $f[x_3] = -0.104$
    2.24
$x_4 = 0.6$ —— $f[x_4] = -0.328$
    8.72
$x_5 = 1.9$ —— $f[x_5] = 11.008$
    ?

The first entry in each column gives the coefficients

$$P_4(x) = -0.736$$
$$+ 2.48(x - 0.3)$$
$$+ 3(x - 0.3)(x - 1.0)$$
$$+ 2(x - 0.3)(x - 1.0)(x - 0.7)$$
$$+ 0(x - 0.3)(x - 1.0)(x - 0.7)(x - 0.6)$$
$$+ 0(x - 0.3)(x - 1.0)(x - 0.7)(x - 0.6)(x - 1.9).$$

5.5 KEY TAKEAWAY

Basically, using the table, we can get all the constants we need to compute the next datapoint by just looking back and grabbing the appropriate column.

5.6 Midterm Review

Find the rate of convergence as $h \to 0$:

$$\lim_{h \to 0}(e^h + e^{-h}) = 2.$$

1. expand $e^h + e^{-h} - 2$ using taylor series

$$e^h = 1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + O(h^4).$$

$$e^h + e^{-h} - 2 = \left(1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + O(h^4)\right)$$
$$+ \left(1 - h + \frac{h^2}{2!} - \frac{h^3}{3!} + O(h^4)\right)$$
$$- 2$$
$$= h^2 + O(h^3) = O(h^2)$$

For midterm, know:

- common taylor series

- Partial Pivoting w/ or w/0 scaling

5.6.1   Another Problem

Consider

$$x_1 + 30x_2 = 50$$
$$5x_1 - 10x_2 = 3$$

Use Gaussian Elimination w/ scaled partial pivoting to take the system to upper triangular form.

$$\frac{1}{30} < \frac{5}{10} \implies \text{we need to exchange rows}$$

$$5x_1 - 10x_2 = 3$$
$$x_1 + 30x_2 = 50$$
$$\overline{5x_1 - 10x_2 = 3}$$
$$32x_2 = 50 - \frac{3}{5}$$

Lecture Review

In this lecture, he begins with a review of somet things that we covered in lecture 22 and 21, which he was absent for and had a TA teach for him.

5.6.2   Divided Differences

Assume that $f(x)$ is known at several points along the x-axis. We do not assume that the x's are evenly spaced or even that the values are arranged in any particular order.

We write the (any arbitrary) n^{th} degree polynomial in the following way:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \ldots (x - x_{n-1}).$$

Def. $P_n(x)$ is an interpolating polynomial for $f(x)$ at $x_0, x_1, \ldots, x_n$ if we choose a_i such that $P_n(x) = f(x))$ at the $n + 1$ known points.

We want to find our a_i to make $P_n(x)$ an interpolating polynomial, and we can determine our a_I by using what are called the "divided differences."

The Zeroth Divided Difference

First, we determine $a_0 : P_n(x_0) = f(x_0) = a_0$, and define the zero$^{\text{th}}$ (degree) divided difference as

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

which is just the value of f at x_i.

The First Divided Difference

Now we need to determine a_1:

$$P_n(x_1) = f(x_1) = a_0 + (x_1 - x_0)a_1.$$

Rearranging, we get

$$a_i = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f[x_1] - f[x_0]}{x_1 - x_0}.$$

which allows us to determine a_1 by using the zero[th] divided difference. We define the first divided difference of f with respect to x_i and x_{i+1} as

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

The Second Divided Difference

Similarly, the second divided difference is defined as

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}.$$

The Kth Divided Difference

In a similar fashion to the evaluation of a_0 and a_1, we can show

$$a_k = f[x_0, x_1, \ldots, x_k].$$

This gives Newton's Interpolatory divided difference formula

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

Disadvantages of Divided Differences

- $P_n(x)$ could be expensive to evaluate

- $P_n(x)$ does go through the datapoints, but between each datapoint, we could have large oscillations, which we expect as the generic behaviour of large degree polynomials.

  This is especially true near endpoints, where our datapoints are sparse and sections where our function is not smooth.

5.7 Better Interpolating Polynomials - Ch2. Pt3. (14.1)

The Lagrange Polynomial can oscillate wildly, except where contained between datapoints that are in close proximity. We want our interpolating polynomial to have the same "shape" as the function at the datapoints. In other words, we want the tangent lines (first derivative) and the function itself to agree at (x_i, f_1)

5.7.1 Proof from (14.2).

This section is directly copied from the Chapter 3 Part 2 pdf.

Thm. If $f \in C^1[a, b]$ and $x_0, \ldots, x_n \in [a, b]$ are distinct, the interpolating polynomial of least degree agreeing with f at $x_0, \ldots, x_n$ is the Hermite polynomial of degree at most $2n + 1$ given by

$$H(x) = \sum_{j=0}^{n} f(x_j) H_j(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_j(x)$$

where

$$H_j(x) = \left[1 - 2(x - x_j) L'_j(x_j)\right] L_j^2(x)$$

and

$$\hat{H}_j(x) = (x - x_j) L_j^2(x)$$

where $L_j(x)$ denotes the Lagrange coefficient polynomial of degree n.

Proof. See text for a derivation showing that $H(x)$ agrees with f and $H'(x)$ agrees with f' at $x_0, x_1, \ldots, x_n$.

To show the uniqueness of this polynomial:

Suppose that $P(x)$ is another polynomial with

$$P(x_j) = f(x_j), \quad P'(x_j) = f'(x_j) \quad \text{for } j = 0, \ldots, n$$

and that the degree of $P(x)$ is at most $2n + 1$.

Let $D(x) = H(x) - P(x)$. Then $D(x)$ is a polynomial of degree at most $2n + 1$.

The zeros at each $x_0, x_1, \ldots, x_n$ are equal to

$$D(x) = (x - x_0)^2 \cdots (x - x_n)^2 Q(x)$$

Either $D(x)$ is of degree $2n + 2$ or more, which would be a contradiction, or

$$Q(x) \equiv 0 \Rightarrow D(x) \equiv 0 \Rightarrow P(x) = H(x)$$

which implies that the polynomial is unique. $\square$

5.7.2   The Newton Interpolatory divided difference formula

Unfortunately, a direct application of the theorem requires us to evaluate the Lagrange Polynomials and their derivatives, which is tedious even for small values of n. Instead, we use the Newton Interpolatory divided difference formula:

Select a new sequence of nodes $z_0, z_1, \ldots, z_{2n+1}$

$$z_{2i} = z_{2i+1} = x_i.$$

We have a small problem.

$$f[z_{2i}] = f[z_{2i+1}] = f(x_i).$$

$$f[z_{2i}, z_{2i+1}] = \frac{f[z_{2i+1}] - f[z_{2i}]}{z_{2i+1} - z_{2i}}.$$

This gives us $\frac{0}{0}$, which is not defined. Instead, we think of taking the limit as z_{2i} approaches z_{2i+1}:

$$f[z_{2i}, z_{2i+1}] = \lim_{z_{2i} \to z_{2i+1}} \frac{f[z_{2i+1}] - f[z_{2i}]}{z_{2i+1} - z_{2i}}.$$

Thus, the derivative values are used in place of the undefined first divided differences, otherwise Newton's divided differences are produced as usual.

z	f(z)	first divided differences
$z_0 = x_0$	$f[z_0] = f[x_0]$	$f[z_0, z_1] = f'(x_0)$
$z_1 = x_0$	$f[z_1] = f[x_0]$	$f[z_1, z_2]$
$z_2 = x_1$	$f[z_2] = f[x_1]$	$f[z_2, z_3] = f'(x_1)$
$z_3 = x_1$	$f[z_3] = f[x_1]$	

5.7.3   The Hermite Polynomial

The Hermite polynomial is then defined as

$$H(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \ldots, z_k] \prod_{j=0}^{k-1} (x - z_j).$$

5.8   Splines

Previously, we saw how to compute an approximation to a function over some finite interval using a single polynomial. We increased the degree of the polynomial if we wanted more accuracy.

Even if we enforce the constraint that f and f' agree at the nodes, we still have the problem that P_n will be expensive to compute for large n and that for high n, the interpolating polynomial can still oscillate wildly.

5.9   Splines (15.1)

Previously, we used a single polynomial to compute an approximation to a function over some finite interval. We increased the degree of the polynomial if we wanted more accuracy. However, increasing the degree of the polynomial causes oscillatory behaviour, and high degree polynomials have the property that a fluctuation over a small portion of the interval can induce large fluctuations over the entire interval.

An alternative approach is to divide the interval into subintervals and use a different, lower degree polynomial in each subinterval. These polynomials are patched together to give a piecewise polynomial approximation.

5.9.1   Possible Choices

Piecewise Linear Interpolation

We join a set of data points by a series of straight lines.

The biggest disadvantage of this approach is that the approximation is typically not smooth. **i.e.** it is not differentiable at these points, which may not be satisfactory.

Piecewise Polynomails of Hermite Type

*Hermite is pronounced "her-meat"

If we know the value of a function **and** it's derivative at each of the data points, then we could use a Hermite cubic polynomial on each interval $[x_i, x_{i+1}]$ to approximate the function. Often, however, the derivative of the function is not known at the data points.

Piecewise Quadratic Polynomials

Alternatively, we could join a set of polynomials with quadratic polynomials. This gives us 3 arbitrary constants. There will be 2 conditions to fit the curve

through the endpoints of each interval, but there isn't enough flexibility to set conditions on the derivative at both endpoints x and x_n.

We have enough flexibility to set conditions for the first $n-1$ data points, but the data point x_n requires four conditions, when we can only provide three degrees of freedom.

- P_0: 2 endpoint conditions $+$ 1 slope condition

- P_1: 2 endpoint conditions $+$ 1 slope conditions

- P_n: 2 endpoint conditions $+$ 2 slope conditions

Cubic Spline Interpolation

If we use cubic polynomials between each successive pair of nodes, then there are four constants and it is possible to ensure that the interpolant

- agrees with the function at all the nodes

- is continuously differentiable

- has continuous second derivatives

Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a cubic spline interpolant S for f is a function that satisfies the following:

- $S(x) = S_j(x)$ on $[x_j, x_{j+1}]$

- $S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_j)$ splines must pass through the data points

- $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$ first derivative is continuous

- $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$ second derivative is continuous

We also need a boundary condition. Typically one of the following hold in the problems we will consider:

- $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ "Clamped Boundary Conditions"

- $S''(x_0) = 0 = S''(x_n)$ "Free or Natural Boundary Conditions"
  *Natural Boundary Conditions tell us that S is not curved at the endpoints.

Of course, for the clamped case, we need derivative information, either from the physics or from some assumption. If there are $(n + 1)$ points, the number of intervals and the number of $S_i(x)$'s are n :

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3.$$

Letting $h_i = x_{i+1} - x_i$ we can simplify by substituting x_{i+1} and x_i into $S_i(x)$, $S_i'(x)$ and $S_i''(x)$. Eliminating the b_i and d_i gives a linear system of equations:

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_0c_{i+1}$$
$$= \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$$
$$1 \le i \le n - 1$$

where the a_i are known

$$a_i = f(x_i) \qquad 0 \le i \le n - 1.$$

and we define

$$a_n \equiv f(x_n)$$
$$b_n \equiv f'(x_n)$$
$$c_n \equiv \frac{f''(x_n)}{2}$$

The b_i and d_i are easily found:

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1}) \quad 0 \le i \le n - 1.$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}.$$

We still need to impose the boundary conditions:

Natural Boundary Conditions Consider the Natural BC's: $S''(x_0) = S''(x_n) = 0$.

$$\therefore C_n = 0 \quad \text{(by definition)}.$$

$$S_0''(x) = 2C_0 + 6d_0(x - x_0) \therefore S''(x_0) = 2c_0 + 6d_0(x_0 - x_0) = 0 = c_0.$$

We can derive a matrix equation for the $[c_i]$:

$$Ax = b.$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & & \vdots \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix}$$

$$x = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

$$b = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-l}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix}.$$

There exists a unique solution for the c_i's.

The matrix is strictly diagonally dominant, so it is invertible and a unique solution for the c_i exists.

First we need a definition and a theorem.

Def. The $n \times n$ matrix A is strictily diagonally dominant if

$$|a_{ii}| > \sum_{j=1; j \neq i}^{n} |a_{ij}|.$$

holds for each $i = 1, \ldots, n$.

Thm. A strictly diagonally dominant matrix is invertible.

Also note that the matrix A is **tridiagonal**: all entires are zero except for a band which is 3 entries wide centred on the main diagonal.

Solutions to tridiagonal linear systems can be found very efficiently:

*Only O(n) operations are needed using methods we shall discuss later

Clamped Boundary Conditions We also want to treat clamped boundary conditions:

$$f'(a) = S'(a) = S_0'(x_0) = b_0 + 2c_0(x_0 - x_0) + 3d_0(x_0 - x_0)^2 = b_0.$$

but

$$b_{i-1} = \frac{a_i - a_{i-1}}{h_{i-1}} - \frac{h_{i-1}}{3}(2c_{i-1} + c_i).$$

$$\therefore b_0 = \frac{a_1 - a_0}{h_0} - \frac{h_0}{3}(2c_0 + c_1).$$

$$\implies 2h_0 c_0 + h_0 c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a).$$

Similarly,

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_{n-1} - a_n).$$

Once again, we obtain a strictly diagonally dominant linear system

$$\implies \text{A unique solution exists for the } c_i\text{'s}$$

Furthermore, the system is tridiagonal so it can be solved efficiently for the coefficients of the spline.

Chapter 6

Numerical Differentiation

Chapter 4 of the textbook. This section is also about Numerical Integration, Richardson's Extrapolation, Romberg Integration, Adaptive Quadrature and Gaussian Quadrature.

6.1   Parametric Curves (C3*2-16.3)

The interpolating polynomials and splines that we have previously discussed can only be used to interpolate functions. The techniques can be extended to represent general curves in space that aren't necessarily functions, or even curves which self-intersect.

Suppose we wish to determine a polynomial or a pieceewise polynomial to connect the points
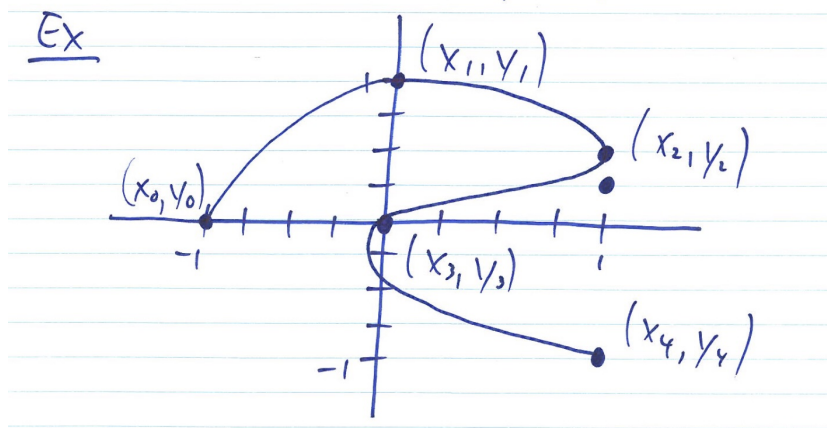
$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n).$$

in the order given.

We can define a parameter t with the interval $[t_0, t_n]$ with

$$t_0 < t_1 < \cdots < t_n.$$

and construct approximation functions for x and y separately:

$$x_i = x(t_i), \quad y_i = y(t_i).$$

Ex.

There is flexibility in choosing the points t_i. Suppose we choose them to be evenly spaced over $[0, 1]$. Then we can write

i	0	1	2	3	4
t_i	0	.25	.5	.75	1
x_i	-1	0	1	0	1
y_i	0	1	.5	0	-1

We could apply Lagrange interpolation for x (or y) as a function of t. The Lagrange Interpolating polynomials for x and y are

$$x(t) = \left(\left(\left(64\,t - \frac{352}{3}\right)t + 60\right)t - \frac{1}{3}\right)t - 1$$

$$y(t) = \left(\left(\left(-\frac{64}{3}t + 48\right)t - \frac{116}{3}\right)t + 11\right)t$$

Alternatively, we could use a spline-based interpolation for x and y.

Both of these approaches have the disadvantage that moving a single data point affects the entire curve. We want the geometric property that changing one of the points on the curve only changes one portion of the curve. **i.e.** we want the curve to only be affected locally by changes in the data.

6.1.1   Piecewise Cubic Hermite Polynomials

Because we want to avoid global changes, we might use a piecewise cubic Hermite polynomial, which is just a spline with specific properties. We use one polynomial for x and one for y, both with respecto to t.

In other words, we specify the endpoints and the derivatives and the endpoints to specify each portion of the curve. Thus, changin ga datapoint will only change the two portions adjacent to the point, which means that smooth curves can be easily and quickly modified.

Uniqueness

Quick answer: No.

The Derivatives and Tangent Lines (C3*2-16.7)

Suppose that the endpoints are at $t = 0$ and $t = 1$, then we only need to satisfy the conditions on the quotients

$$\frac{dy}{dx}(t = 0) = \frac{y'(0)}{x'(0)}, \quad \frac{dy}{dx}(t = 1) = \frac{y'(1)}{x'(1)}.$$

The actual values of $x'(0)$ and $y'(0)$ can be scaled by a common factor and still satisfy these conditions. The larger the scaling factor, the closer the curve comes to satisfying the tangent line near $(x(0), y(0))$.

A similar situation holds for

$$(x(1), y(1)).$$

Guide Points (Cubic Bezier Curves)

To simplify the process of specifying the slopes and to obtain a unique curve, commercial software commonly specifies a second point called a "guidepoint" which lies on the desired tangent line.

Suppose the endpoints are

$$(x(0), y(0)), \text{ and } (x(1), y(1)).$$

and let the guidepoints be

$$(x_0 + \alpha_0, y_0 + \beta_0), \text{ and } (x_1 + \alpha_1, y_1 + \beta_1).$$

we will insist that x satisfies

$$x(0) = x_0 \qquad\qquad x'(0) = \alpha_0$$
$$x(1) = x_1 \qquad\qquad x'(1) = \alpha_1$$

and that y satisfies

$$y(0) = y_0 \qquad\qquad y'(0) = \beta_0$$
$$y(1) = y_1 \qquad\qquad y'(1) = \beta_1$$

Now there is a unique solution for x and y:

$$x(t) = [2(x_0 - x_1) + (\alpha_0 + \alpha_1)]\, t^3$$
$$+ [3(x_1 - x_0) - (\alpha_1 + 2\alpha_0)]\, t^2$$
$$+ \alpha_0 t + x_0$$

$$y(t) = [2(y_0 - y_1) + (\beta_0 + \beta_1)]\, t^3$$
$$+ [3(y_1 - y_0) - (\beta_1 + 2\beta_0)]\, t^2$$
$$+ \beta_0 t + y_0$$

Popular graphics programs will typically use a slightly modified form for x and y. They use Bézier polynomials which scale the derivatives x' and y' by a factor of **3** at the endpoints.

$$x(t) = [2(x_0 - x_1) + (\alpha_0 + \alpha_1)]\, t^3$$
$$+ [3(x_1 - x_0) - (\alpha_1 + 2\alpha_0)]\, t^2$$
$$+ \mathbf{3}\alpha_0 t + x_0$$

$$y(t) = [2(y_0 - y_1) + (\beta_0 + \beta_1)]\, t^3$$
$$+ [3(y_1 - y_0) - (\beta_1 + 2\beta_0)]\, t^2$$
$$+ \mathbf{3}\beta_0 t + y_0$$

Fun Notes

Generally, given that you don't have two points on the same point, i.e. you don't have $(x(0), y(0)) = (x(1), y(1))$, you can guarantee that the output curve will be continuous with a continuous derivative. (i.e. $P \in C^1[a, b]$).

6.2   Numerical Differentiation (C4*1-17.1)

We also need to approximate the derivatives of functions. One approach is to differentiate Lagrange polynomial approximations.

Suppose $x_0, x \in (a, b)$ and $f \in C^2[a, b]$.

Now

$$
\begin{aligned}
f(x) &= P_{0,1}(x) + \frac{1}{2!}(x - x_0)(x - x_1)f''(\xi(x)) \\
&= \frac{f(x_0)(x - x_1)}{x_0 - x_1} + \frac{f(x_1)(x - x_0)}{x_1 - x_0} + \frac{(x - x_0)(x - x_1)}{2!}f''(\xi(x)) \\
&\text{where } \xi(x) \in [a, b]
\end{aligned}
$$

Now differentiate:

$$
\begin{aligned}
f'(x) &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} + D_x\left[\frac{(x - x_0)(x - x_1)}{2!}f''(\xi(x))\right] \\
&= \frac{f(x_1) - f(x_0)}{x_1 - x_0} + \frac{2(x - x_0)(x - x_1)}{2}f''(\xi(x)) \\
&\quad + \frac{(x - x_0)(x - x_1)}{2}D_x\left(f''(\xi(x))\right)
\end{aligned}
$$

6.2.1   More remarks

Chapter 4 also considers numerical integration. We know how to integrate polynomials, so we can just integrate the lagrange polynomial and call it a day.

6.3   Numerical Differentiation (C4*1-17.1)

We also need to approximate the derivatives of functions. One approach is to differentiate Lagrange polynomial approximations.

Suppose $x_0, x \in (a, b)$ and $f \in C^2[a, b]$.

Now

$$
\begin{aligned}
f(x) &= P_{0,1}(x) + \frac{1}{2!}(x - x_0)(x - x_1)f''(\xi(x)) \\
&= \frac{f(x_0)(x - x_1)}{x_0 - x_1} + \frac{f(x_1)(x - x_0)}{x_1 - x_0} + \frac{(x - x_0)(x - x_1)}{2!}f''(\xi(x)) \\
&\quad \text{where } \xi(x) \in [a, b]
\end{aligned}
$$

Now differentiate:

$$
\begin{aligned}
f'(x) &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} + D_x\left[\frac{(x - x_0)(x - x_1)}{2!}f''(\xi(x))\right] \\
&= \frac{f(x_1) - f(x_0)}{x_1 - x_0} + \frac{2(x - x_0)(x - x_1)}{2}f''(\xi(x)) \\
&\quad + \frac{(x - x_0)(x - x_1)}{2}D_x\left(f''(\xi(x))\right)
\end{aligned}
$$

We only care about the derivatives at the nodes x_0, x_1. The last error term becomes zero.

For example, at $x = x_0$,

$$f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} - \frac{(x_0 - x_1)}{2} f''(\xi).$$

Typically, we set $x_1 = x_0 + h$, then

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''(\xi(x_0)) \qquad (6.1)$$

This is known as a forward difference formula if $h > 0$ and a backward difference formula if $h < 0$. We can derive more general approximation formulas:

6.3.1   More General Approximation Formulas (17.3)

Suppose $x_0, x_1, \ldots x_n \in (a, b)$ and $f \in C^{n+1}[a, b]$. Now

$$f(x) = \sum_{k=0}^{n} f(x_k) L_k(x)$$
$$+ \frac{(x - x_0) \ldots (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi(x))$$

for some $\xi(x) \in [a, b]$
Differentiate and evaluate at $x = x_j$:

$$f'(x_j) = \sum_{k=0}^{n} f(x_k) L_k'(x_j)$$
$$+ \frac{f^{(n+1)}(\xi(x_j))}{(n + 1)!} \prod_{k=0;k \neq j}^{n} (x_j - x_k)$$

This is an $(n + 1)$ point formula for $f'(x_j)$ since we use the $(n + 1)$ values $f(x_k); k = 0, \ldots, n$. Two, three and five point formulas are hte most commonly used formulas.

Three Point Formulas (17.4)

Consider 3 point formulas with x_0, x_1 and x_2.
  Given $n = 2$:

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

Taking the derivative:

$$L_0'(x) = \frac{2x - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)}$$

Similarly,

$$L_1'(x) = \frac{2x - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)}$$

$$L_2'(x) = \frac{2x - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)}$$

and

$$f'(x_j) = f[x_0] \left[\frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)} \right]$$

$$+ f[x_1] \left[\frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)} \right]$$

$$+ f[x_2] \left[\frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)} \right]$$

$$+ \frac{1}{6} f^{(3)}(\xi_j) \prod_{\substack{k=0 \\ k \neq j}}^{2} (x_j - x_k)$$

These simplify considerably when nodes are equally spaced:

Equally Spaced Three Point Formulas (17.5)

Given:

$$x_1 = x_0 + h, \quad x_2 = x_0 + 2h$$

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0 + h) - \frac{1}{2}f(x_0 + 2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0) \qquad (6.2)$$

$$f'(x_1) = \frac{1}{h}\left[-\frac{1}{2}f(x_1 - h) + \frac{1}{2}f(x_1 + h)\right] - \frac{h^2}{6}f^{(3)}(\xi_1) \qquad (6.3)$$

$$f'(x_2) = \frac{1}{h}\left[\frac{1}{2}f(x_2 - 2h) - 2f(x_2 - h) + \frac{3}{2}f(x_2)\right] + \frac{h^2}{3}f^{(3)}(\xi_2) \qquad (6.4)$$

For convenience, replace x_1 and x_2 by x_0. This gives 3 formulas for approximating $f'(x_0)$.

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0 + h) - \frac{1}{2}f(x_0 + 2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0)$$

$$f'(x_0) = \frac{1}{h}\left[-\frac{1}{2}f(x_0 - h) + \frac{1}{2}f(x_0 + h)\right] - \frac{h^2}{6}f^{(3)}(\xi_1)$$

$$f'(x_0) = \frac{1}{h}\left[\frac{1}{2}f(x_0 - 2h) - 2f(x_0 - h) + \frac{3}{2}f(x_0)\right] + \frac{h^2}{3}f^{(3)}(\xi_2)$$

Formula 1 uses two points ahead, formula 2 uses one point in front and one point behind, and formula 3 uses two points behind.

Example 1 (17.5.1)

Ex. use the most appropriate three-point formula to determine approximatinos that will complete the following table:

x	$f(x)$	$f'(x)$
1.1	9.025013	
1.2	11.02318	
1.3	13.46374	
1.4	16.44465	

ANSWER: (17.6)

We only have data in $[1.1, 1.4]$, and no data before or after this interval. So, for $f'(1.1)$ we use formula 1. For $f'(1.2)$ we can use formula 2 or formula 1, but since formula 2 has a smaller error, we choose it. We do the same for $f'(1.3)$. For $f'(1.4)$ we use formula 3, since we only have data behind, and no available data ahead.

$$f'(1.1) \approx \frac{1}{2(0.1)} \left[-3f(1.1) + 4f(1.2) - f(1.3) \right] = 17.769705$$

$$f'(1.2) \approx \frac{1}{2(0.1)} \left[f(1.3) - f(1.1) \right] = 22.193635$$

$$f'(1.3) \approx \frac{1}{2(0.1)} \left[f(1.4) - f(1.2) \right] = 27.107350$$

$$f'(1.4) \approx \frac{1}{2(0.1)} \left[f(1.2) - 4f(1.3) + f(1.4) \right] = 32.510850$$

these are OCR'd through ChatGPT. They may be wrong.

Notice that at the endpoints we must use one sided difference formulas. In the interior, we used centered differencing. Centered differences often have a smaller error constant when f is smooth, and they require fewer operations to compute, but they cannot be used at the endpoints.

The Second Derivative of f (17.8)

Approximations to higher order derivatives may also be found based on function values. Consider finding the second derivative of f.

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_1)h^4$$

$$f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_{-1})h^4$$

where $x_0 - h < \xi_{-1} < x_0 < \xi_1 < x_0 + h$.

Summing:

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + f''(x_0)h^2 + \frac{h^4}{24}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_2)\right] \quad (6.5)$$

We assume $f^{(4)}$ is continuous on $[x_0 - h, x_0 + h]$.

Since $\frac{1}{2}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right]$ is between $f^{(4)}(\xi_1)$ and $f^{(4)}(\xi_{-1})$, the *intermediate value theorem* implies that there is a number ξ between ξ_1 and ξ_2 such that

$$f^{(4)}(\xi) = \frac{1}{2}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right].$$

Thus,

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + f''(x_0)h^2 + \frac{h^4}{24}f^{(4)}(\xi).$$

Therefore,

$$f''(x_0) = \frac{1}{h^2}\left[f(x_0 - h) - 2f(x_0) + f(x_0 + h)\right] - \frac{h^2}{24}f^{(4)}(\xi).$$

Remarks on Error (17.9)

Notice that all the differentiation formulas divide by some power of h. Division by small numbers tends to exaggerate roundoff error, but this is an effect that cannot be entirely avoided in numerical differentiation. Thus we do not want to take h to be too small because then the roundoff errors will dominate the calculation.

The exact value at which h becomes too small depends on the scaling of your problem and your specific function f. However, for most problems, scaled nicely, you will start to see roundoff errors at around $h = 10^{-6}$.

6.4   Richardson's Extrapolation (C4*1-17.10)

When the error depends on some parameter such as the step size h and the dependency is predictable, we can often derive higher order accuracy from low order formulas. To illustrate the procedure, assume we have an approximation $N(h)$ to some quantity M. Assume this approximation has an order h truncation error and that we know the expression for the first few terms of the trunction error,

$$M = N(h) + k_1 h + k_2 h^2 + k_3 h^3 + \ldots \tag{6.6}$$

where the k_i's are constants, h is a positive parameter and $N(h)$ is an $O(h)$ approximation to M. We can repeat the calculation with a parameter $\frac{h}{2}$. Now:

$$M = N(\frac{h}{2}) + \frac{k_1}{2}h + \frac{k_2}{4}h^2 + \frac{k_3}{8}h^3 + \ldots \tag{6.7}$$

We want to obtain a higher order method by using some combination of these results.

Subctracting (**??**) from twice (**??**) gives:

$$M = [2N(\frac{h}{2}) - N(h)] + k_2(\frac{h^2}{2} - h^2) + k_3(\frac{h^3}{4} - h^3) + \ldots \tag{6.8}$$

which is an $O(h^2)$ approximation formula for M. For ease of notiation,

$$\text{Let } N_2(h) = 2N(\frac{h}{2}) - N(h).$$

Now

$$M = N_2(h) - \frac{1}{2}k_2 h^2 - \frac{3}{4}k_3 h^3 - \ldots \tag{6.9}$$

We can repeat this calculation with $\frac{h}{2}$:
Now

$$M = N_2(\frac{h}{2}) - \frac{1}{8}k_2 h^2 - \frac{3k_3}{32}h^3 - \ldots \tag{6.10}$$

We want to eliminate the h^2 term. We can do this by subtracting four times (**??**) from (**??**), which gives:

$$3M = 4N_2(\frac{h}{2}) - N_2(h) + \frac{3k_3}{8}h^3 + \dots \tag{6.11}$$

which gives an $O(h^3)$ formula for approximating M.

$$M = N_3(h) + \frac{1}{3}J_3 h^3 + \dots$$

where

$$N_3(h) = \frac{4}{3}N_2\left(\frac{h}{2}\right) - \frac{1}{3}N_2(h).$$

Similarly, an $O(h^4)$ approximation can be derived as:

$$N_4(h) = N_3\left(\frac{h}{2}\right) + \frac{N_3(h/2) - N_3(h)}{7}.$$

And an $O(h^5)$ approximation:

$$N_5(h) = N_4\left(\frac{h}{2}\right) + \frac{N_4(h/2) - N_4(h)}{15}.$$

Generally, if M can be written as:

$$M = N(h) + \sum_{j=1}^{m-1} K_j h^j + O(h^m),$$

then for each $j = 2, 3, \dots, m$, we have an $O(h^j)$ approximation of the form:

$$N_j(h) = N_{j-1}\left(\frac{h}{2}\right) + \frac{N_{j-1}\left(\frac{h}{2}\right) - N_{j-1}(h)}{2^{j-1} - 1}.$$

6.4.1   More remarks - things that Steve said during lecture

Chapter 4 also considers numerical integration. We know how to integrate polynomials, so we can just integrate the lagrange polynomial and call it a day.

6.5 Overview

The most important parts of this lecture are the following:

1. **Richardson's Extrapolation**: A method for obtaining higher order accuracy from lower order formulas. link: **??**

   (a) **??** illustrates the procedure of building the richardson extrapolation table.

   (b) **??** illustrates the use of Richardson's Extrapolation to obtain a higher order approximation to the integral of $\sin x$.

2. **Numerical Integration**: A method for evaluating the definite integral of a function that has no explicit antiderivative or whose antiderivative is not easy to obtain. link: **??**

   (a) **??** illustrates the procedure of using a 2 point integration formula.

   (b) **??** illustrates the procedure of using the weighted mean value theorem for integrals.

   (c) **??** illustrates the procedure of using a 3 point integration formula.

6.6   Richardson's Extrapolation (C4*1-17.10)

When the error depends on some parameter such as the step size h and the dependency is predictable, we can often derive higher order accuracy from low order formulas. To illustrate the procedure, assume we have an approximation $N(h)$ to some quantity M. Assume this approximation has an order h truncation error and that we know the expression for the first few terms of the trunction error,

$$M = N(h) + k_1 h + k_2 h^2 + k_3 h^3 + \ldots \tag{6.12}$$

where the k_i's are constants, h is a positive parameter and $N(h)$ is an $O(h)$ approximation to M. We can repeat the calculation with a parameter $\frac{h}{2}$:

$$M = N\left(\frac{h}{2}\right) + \frac{k_1}{2}h + \frac{k_2}{4}h^2 + \frac{k_3}{8}h^3 + \ldots. \tag{6.13}$$

We want to obtain a higher order method by using some combination of these results. Subtracting (**??**) from twice (**??**) gives:

$$M = \left[2N\left(\frac{h}{2}\right) - N(h)\right] + k_2\left(\frac{h^2}{2} - h^2\right) + k_3\left(\frac{h^3}{4} - h^3\right) + \ldots \tag{6.14}$$

which is an $O(h^2)$ approximation formula for M. For ease of notation,

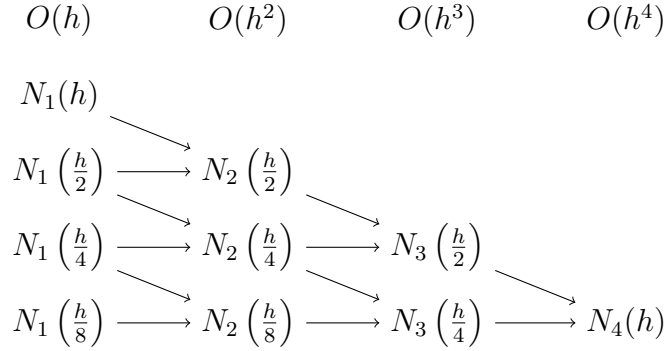$$\text{Let } N_2(h) = 2N\left(\frac{h}{2}\right) - N(h).$$

Generally, if M can be written as

$$M = N(h) + \sum_{j=1}^{m-1} K_j h^j + O(h^m), .$$

then for each $j = 2, 3, \ldots, m$, we have an $O(h^j)$ approximation of the form

$$N_j(h) = N_{j-1}\left(\frac{h}{2}\right) + \frac{N_{j-1}\left(\frac{h}{2}\right) - N_{j-1}(h)}{2^{j-1} - 1}..$$

In practice, because of how N_j is defined, higher order approximations can be systematically derived from lower order approximations.

Building the Extrapolation Table (18.1)

$$O(h) \qquad O(h^2) \qquad O(h^3) \qquad O(h^4)$$

$$
\begin{array}{llll}
N_1(h) & & & \\
N_1\left(\frac{h}{2}\right) \longrightarrow N_2\left(\frac{h}{2}\right) & & & \\
N_1\left(\frac{h}{4}\right) \longrightarrow N_2\left(\frac{h}{4}\right) \longrightarrow N_3\left(\frac{h}{2}\right) & & \\
N_1\left(\frac{h}{8}\right) \longrightarrow N_2\left(\frac{h}{8}\right) \longrightarrow N_3\left(\frac{h}{4}\right) \longrightarrow N_4(h) &
\end{array}
$$

The cost of building the extrapolation table is $O(n)$ where n is the inverse degree of the error term ($O(h^n)$). Each subsequent iteration of N is defined solely by its previous iteration, therefore, the cost of building the extrapolation is the number of initial calculations of N_1 required to build out the table. Some pitfalls of this approach is that the calculation of N_{j+1} requires a subtraction of two numbers that get closer and closer as the degree of the error term increases. This is a problem for higher order calculations, as you may run out of precision and introduce substantial machine error.

Extrapolation can be used whenever the truncation error for a formula has the form

$$\sum_{j=1}^{m-1} K_j h^{\alpha_j} + O(h^{\alpha_m}).$$

for constants K_J and $\alpha_1 < \alpha_2 < \cdots < \alpha_m$.

When $N_j(h)$ is an $O(h^{2j})$ approximation

Suppose $N_j(h)$ is an $O(h^{2j})$ approximation of M. Then, from the definition:

$$
\begin{align}
M &= N_j(h) + O(h^{2j}) \qquad \text{we add another term of } M: \tag{6.15} \\
&= N_j(h) + k_j(h^{2j}) + O(h^{2j+2}) \tag{6.16} \\
&= N_j(\frac{h}{2}) + k_j h^{2j} + O(h^{2j+2}) \tag{6.17}
\end{align}
$$

2^{2j} (**??**) $-$ (**??**) gives:

$$M = N_j(\frac{h}{2}) + \frac{N_j(\frac{h}{2}) - N_j(h)}{2^{2j-1}} + O(h^{2j+2})$$

$$\therefore N_{j+1}(h) \equiv N_j\left(\frac{h}{2}\right) + \frac{N_j\left(\frac{h}{2}\right) - N_j(h)}{4^{j-1}}.$$

is an $O(h^{2j+2})$ approximation of M. Then, the table becomes

$$O(h^2) \quad O(h^4) \quad O(h^6) \quad O(h^8)$$

6.6.1   Richardson's Extrapolation Example (18.1)

Ex. The following data gives approximations to the integral

$$M = \int_0^\infty \sin x \, dx \tag{6.18}$$

$$N_1\left(\frac{h}{2}\right) = 1.570796, \quad N_1\left(\frac{h}{2}\right) = 1.896119$$

$$N_1\left(\frac{h}{4}\right) = 1.974232, \quad N_1\left(\frac{h}{8}\right) = 1.993570$$

Assuming

$$M = N_1(h) + K_1 h^2 + K_2 h^4 + k_3 h^6 + k_4 h^8 + O(h^{10}).$$

construct an extrapolation table to determine $N_4(h)$.

Soln. (from text)

6.7 Numerical Integration (18.4)

We often need to evaluate the definite integral of a function that has no explicit antiderivative or whose antiderivative is not easy to obtain. The usual strategy in developing formulas for numerical integration is similar to that for numerical differentiation. We pass a polynomial through points defined by the function and then integrate this polynomial approximation for the function. This permist us to use a function known only as a table of values. We get an expression for the error by integrating the error for our interpolating polynomial. As we progress, we will eventually turn to splitting our function into sub-intervals and using methods similar to spline interpolation to integrate each sub-interval.

6.7.1 2-Point Integration Formula (18.5)

Suppose we use a 2 point integration formula:
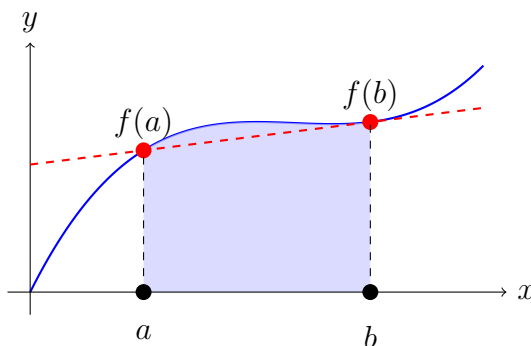
Let $x_0 = a$, $x_1 = b$, $h = b - 1$

The linear Lagrange polynomial passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is

$$P_1(x) = \frac{(x - x_1)(x - x_0)}{f(x_0)} + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1).$$

and

$$\int_a^b f(x)\, dx = \int_{x_0}^{x_1} P_1(x)\, dx + \frac{1}{2} \int_{x_0}^x f''\left(\xi(x)\right) (x - x_0)(x - x_1)\, dx$$

$$= \frac{(x - x_1)^2}{2(x_0 - x_1)} f(x_0) + \frac{(x - x_0)^2}{2(x_1 - x_0)} f(x_1) \bigg|_{x_0}^{x_1} + \text{error}$$

$$= \frac{h}{2} \left(f(x_0) + f(x_1) \right) + \text{error}.$$

To evaluate the error we will need the **Weighted Mean Value Theorem for Integrals**.

Weighted Mean Value Theorem for Integrals (18.6)

If $f \in C[a, b]$, the Riemann Integral of g exists on $[a, b]$ and $g(x)$ does not change sign on $[a, b]$ then there exists a number $c \in (a, b)$ such that
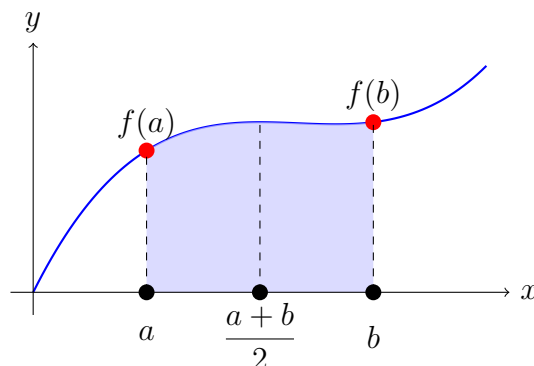
$$\int_a^b f(x)g(x)\, dx = f(c) \int_a^b g(x).$$

$$\text{error} = \frac{1}{2} \int_{x_0}^{x_1} f''\left(\xi(x)\right) (x - x_0)(x - x_1)\, dx$$

$$= \frac{1}{2} f''(\xi) \int_{x_0}^{x_1} (x - x_0)(x - x_1)\, dx \quad [\xi \text{ some number in } (x_0, x_1)]$$

$$= \frac{1}{2} f''(\xi) \left[\frac{x^3}{3} - \frac{(x_1 + x_0)}{2} x^2 + x_0 x_1 x \right]_{x_0}^{x_1}$$

$$= -\frac{h^3}{12} f''(\xi)$$

$$\text{Thus} \quad \boxed{\int_a^b f(x)\, dx = \frac{h}{2} \left[f(x_0) + f(x_1) \right] - \frac{h^3}{12} f''(\xi)}$$

This is known as the **Trapezoid Rule** since the integral is approximated by the area of a trapezoid.

6.7.2 Three Point Integration Formula (18.7)

We might also consider a 3 point integration formula based on equally spaced points:

If we use the usual strategy of integrating the error term for the Lagrange polynomial then we get an $O(h^4)$ error. A sharper estimate can be obtained using an alternative approach.

Expand f about x, using the third Taylor polynomial:

$$
f(x) = f(x_1) + f'(x_1)(x - x_1) + \frac{f''(x_1)(x - x_1)^2}{2}
$$
$$
+ \frac{f'''(x_1)(x - x_1)^3}{6} + \frac{f^{(4)}(\xi(x))(x - x_1)^4}{24}
$$

$$
\int_{x_0}^{x_2} P_3(f(x)) \, dx = \left[f(x_1)(x - x_1) + \frac{f'(x_1)}{2}(x - x_1)^2 + \frac{f''(x_1)}{6}(x - x_1)^3 \right.
$$
$$
\left. + \frac{f'''(x_1)}{24}(x - x_1)^4 \right]_{x_0}^{x_2}
$$
$$
+ \frac{1}{24} \int_{x_0}^{x_2} f^{(4)}(\xi(x))(x - x_1)^4 \, dx
$$

$$\text{Consider} \quad \frac{1}{24} \int_{x_0}^{x_2} f^{(4)}(\xi(x))(x-x_1)^4 \, dx$$

$$= \frac{f^{(4)}(\xi_1)}{24} \int_{x_0}^{x_2} (x-x_1)^4 \, dx$$

$$= \frac{f^{(4)}(\xi_1)}{120} (x-x_1)^5 \Big|_{x_0}^{x_2}$$

$$= \frac{f^{(4)}(\xi_1)}{60} h^5$$

$$\therefore \int_{x_0}^{x_2} f(x) \, dx = 2hf(x_1) + \frac{h^3}{3} f''(x_1) + \frac{f^{(4)}(\xi_1)h^5}{60}.$$

But, we know that $f''(x_1) = \dfrac{1}{h^2} \left[f(x_0) - 2f(x_1) + f(x_2) \right] + \dfrac{h^2}{12} f^{(4)}(\xi_1)$.

$$\int_{x_0}^{x_2} f(x) \, dx = 2hf(x_1) + \frac{h^3}{3} \left[\frac{1}{h^2}(f(x_0) - 2f(x_1) + f(x_2)) - \frac{h^2 f^{(4)}(\xi_2)}{12} \right]$$

$$+ \frac{f^{(4)}(\xi_1)}{60} h^5$$

$$= \frac{h}{3} \left[f(x_0) + 4f(x_1) + f(x_2) \right] + O(h^5)$$

This integration rule is known as **Simpson's Rule**:

$$\int_{x_0}^{x_2} f(x) \, dx = \frac{h}{3} \left[f(x_0) + 4f(x_1) + f(x_2) \right] - \underbrace{\frac{h^5}{90} f^{(4)}(\xi)}_{\text{part of assignment 4}}$$

Lecture Outline

1. Starts with a review of the previous lecture, including:

   (a) Interpolation with two points (Trapezoidal Rule)

   (b) Interpolation with three points (Simpson's Rule)

2. The **error analysis** of numerical integration methods. **(??)**.

   (a) Simpson's Rule, despite using a quadratic approximation, can integrate up to a cubic. (Integrating a quadratic gives a cubic, and the method specifically finds a nice cancellation that allows higher order approximations)

   (b) Definition of the degree of accuracy **??**

3. It turns out that Simpson's Rule and the Trapezoidal Rule are not the only methods of numerical integration. In fact, there are infinitely many methods of numerical Integration: The **Newton-Cotes Formulas (??)**.

   (a) Even-degree Newton-Cotes formulas are more accurate than odd.

   (b) The formulas are **exact** if the degree of accuracy is greater than or equal to the degree of the function it is approximating.

   (c) The Newton-Cotes formulas are **closed** if the endpoints of the interval are included as nodes. The formulas are **open (??)** if the nodes are all contained in the open interval (a, b).

4. Finally, we discuss the **composite numerical integration (??)** method. This method uses Newton-Cotes formulas to approximate the integral of a function over an interval.

6.8 Error Analysis of Numerical Integration Methods

$f(x)$	Simpson's	Trapezoidal
x	0	0
x^3	nonzero	0

Simpson's Rule can integrate up to a cubic, even though the function used to approximate the integral is a quadratic. This is because in Simpson's Rule, the quadratic approximation is integrated and there is a cancellation that occurs, which allows the quadratic approximation to be used to approximate the integral.

6.8.1   Degree of Accuracy

Def. The **degree of accuracy** or precision of a quadratic formula is the largest positive integer n such that the formula is exact for x^k when $k = 0, 1, \ldots, n$.

	Degree of Accuracy
Trapezoid Rule	1
Simpson's Rule	3

6.9   Newton-Cotes Formulas

The Trapezoid and Simpson's Rules are examples of **Newton-Cotes Formulas**. The $(n + 1)$-point <u>closed</u> Newton-Cotes formula uses nodes $x_i = x_0 + ih$ for $i = 0, 1, \ldots, n$ where

$$x_0 = a,$$
$$x_n = b,$$
$$h = \frac{b - a}{n}$$

Then,

$$\int_a^b f(x)\, dx \approx \int_a^b P_n(x)\, dx = \int_a^b \sum_{i=0}^n h_i(x) f(x_i)\, dx$$
$$= \sum_{i=0}^n \int_a^b L_i(x) f(x_i)\, dx$$
$$= \sum_{i=0}^n a_i f(x_i)$$

where $a_i = \int_a^b L_i(x)\, dx$

The formula is closed because the endpoints of the interval are included as nodes. An error analysis of the Newton-Cotes formulas gives an interesting result:

Thm. Suppose that $\sum_{i=0}^n a_i f(x_i)$ denotes the $n+1$ point closed Newton-Cotes formula with $x_0 = a$, $x_n = b$ and $h = \frac{b-a}{n}$. If n is even and $f \in C^{n+2}[a, b]$ then there exists $\xi \in (a, b)$ with

$$\int_a^b f(x)\, dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1)\ldots(t-n)\, dt.$$

If n is odd and $f \in C^{n+1}[a, b]$ then there exists $\xi \in (a, b)$ with

$$\int_a^b f(x)\, dx = \sum_{i=0}^n a_i f(x_i) + \frac{h^{n+2} f^{n+1}(\xi)}{(n+1)!} \int_0^n t(t-1)\ldots(t-n)\, dt.$$

Notice that the degree of precision is $n+1$ and the error is $O(h^{n+3})$ if n is even. If n is odd then the degree of precision is only n and hte error is only $O(h^{n+2})$.

n	Name	Error Term
1	Trapezoid Rule	$-\frac{h^3}{12} f''(\xi)$
2	Simpson's Rule	$-\frac{h^5}{90} f^{(4)}(\xi)$
3	Simpson's Three-Eighths Rule	$-\frac{3h^5}{80} f^{(4)}(\xi)$
4		$-\frac{h^7}{945} f^{(6)}(\xi)$

6.9.1   Open Newton-Cotes Formulas

There are also <u>open</u> Newton-Cotes formulas. here,

$$x_i = x_0 + ih \qquad i = 0, 1, \ldots, n$$
$$x_0 = a + h$$
$$h = \frac{b-a}{n+2}$$

Then the open Newton-Cotes formulas are given by

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} a_i f(x_i). \tag{6.19}$$

where $a_i = \displaystyle\int_a^b L_i(x)\,dx$

Note that $x_0 = a + h$ and $x_n = b - h$. The formulas are <u>open</u> because the nodes are all contained in the open interval (a, b). Once again, if n is even, the degree of precision is $(n + 1)$ and the error is $O(h^{n+3})$. If n is odd, the degree of precision is only n and the error is only $O(h^{n+2})$.

Some examples of open Newton-Cotes formulas are:

n	
0	$2hf(x_0) + \dfrac{h^3}{3}f''(\xi), \quad \text{where } \xi \in (a, b)$
1	$\dfrac{3h}{2}\left[f(x_0) + f(x_1)\right] + \dfrac{3h^3}{4}f''(\xi), \quad \text{where } \xi \in (a, b)$
2	$\dfrac{4h}{3}\left[2f(x_0) - f(x_1) + 2f(x_2)\right] + \dfrac{14h^5}{45}f^{(4)}(\xi)$
3	$\dfrac{5h}{24}\left[11f(x_0) + f(x_1) + f(x_2) + 11f(x_3)\right] + \dfrac{95}{144}h^5 f^{(4)}(\xi)$

$n = 0$ is also called the **Midpoint Rule**.

6.10   Composite Numerical Integration

Typically, we do not apply Newton-Cotes formulas directly onto the interval $[a, b]$. If we did, then high degree formulas would be required to obtain accurate solutions, but as we have already seen, these high degree polynomials would give an oscillatory and innacurate interpolation. To avoid this problem, we prefer a piecewise approach to numerical integration that uses low order Newton-Cotes formulas.

Ex. Simpson's Rule
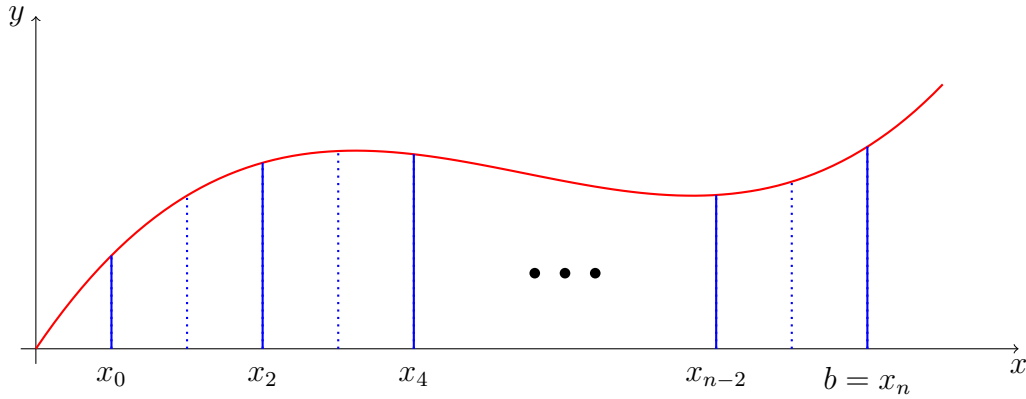
We divide the interval into an even number of subintervals. Simpson's rule is applied on each consecutive pair of subintervals.

$$\frac{h}{3}\left[f(x_i) + 4f(\frac{x_i + x_{i+2}}{2}) + f(x_{i+2})\right].$$

Take $h = \dfrac{(b-1)}{n}$ and $x_j = a + jh$. Then,

$$\int_a^b f(x)\,dx = \sum_{j=1}^{\frac{n}{2}} \int_{x_{2j-2}}^{x_{2j}} f(x)\,dx$$

$$= \sum_{j=1}^{\frac{n}{2}} \left\{ \frac{h}{3}\left[f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})\right] - \frac{h^5}{90}f^{(4)}(\xi_j)\right\}$$

$$\text{where } x_{2j-2} < \xi_j < x_{2j}$$
$$\text{and } f \in C^4[a,b]$$

taking into account that $f(x_{2j}), 0 < j < \frac{n}{2}$ appears in 2 terms, this summation can be simplified somewhat

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n)\right] + \text{error}.$$

got to page 41/80 (32/54 in the published lecture slides)

Lecture Outline

1.

6.11 Error Behavior of Simpson's Rule (19.12)

It is important to understand the stability property of **Composite Newton-Cotes** integration techniques.

Assume $f(x_i)$ is approximated by $\tilde{f}(x_i)$:

$$f(x_i) = \tilde{f}(x_i) + e_i \qquad 0 \leq i \leq n.$$

where e_i is the roundoff associated with using $\tilde{f}$ to approximate f. Then, the accumulated roundoff error in the Composite Simpson's Rule is

$$|e(h)| = \left| \frac{h}{3} \left[e_0 + 2 \sum_{j=1}^{\frac{n}{2}-1} e_{2j} + 4 \sum_{j=1}^{\frac{n}{2}} e_{2j-1} + e_n \right] \right|$$

$$\leq \frac{h}{3} \left[|e_0| + 2 \sum_{j=1}^{\frac{n}{2}-1} |e_{2j}| + 4 \sum_{j=1}^{\frac{n}{2}} |e_{2j-1}| + |e_n| \right]$$

We have a triangle inequality. Assume all e_j are bounded by $\mathcal{E}$.

$$h = \frac{(b-a)}{n}$$

$$\begin{aligned}
|e(h)| &\leq \frac{h}{3} \left[\mathcal{E} + 2(\frac{n}{2} - 1)\mathcal{E} + 4\frac{n}{2}\mathcal{E} + \mathcal{E} \right] \\
&= \frac{h}{3} 3n\mathcal{E} \\
&= nh\mathcal{E} \\
&= (b-a)\mathcal{E}
\end{aligned}$$

Which is indepentent of h which implies the procedure is stable as $h \to 0$

6.12 Romberg Integration

An interesting point concerning the composite Trapezoid Rule:
   If $f =\in C^2[a, b]$ then there exists a $\mu \in [a, b]$ such that

$$\int_a^b f(x)\, dx = \frac{h}{2}\left[f(a) + 2 \sum_{j=1}^{h-1} f(x_j) + f(b) \right] - \frac{b-a}{12} h^2 f''(\mu)$$

$$\text{where } h = \frac{b-1}{n}$$

$$\text{and } x_j = a + jh$$

Thus the error for the composite Trapezoid Rule is $O(h^2)$. In fact we can be more precise. An application of the Euler-MacLaurin summation formula shows that for sufficiently smooth f,

$$\text{error} \quad = c_1 h^2 + c_2 h^4 + \cdots + c_m h^{2m} + O(h^{2m+2})$$

$$\text{where} \quad c_k = \text{const} \times \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right)$$

This shows us that the Composite Trapezoid Rule is extremely accurate for smooth periodic functions, provided h is small enough.
   *Note: The error expansion contains only even powers of h, so eliminating the leading error term improves the accuracy by two additional orders of h.
   Notice that we know the form of the error, so we can obtain higher order accuracy by using Richardson Extrapolation. (To give Romberg Integration)

6.12.1 Richardson Extrapolation to obtain Romberg Integration

We will carry out Composite Trapezoid Rule approximations with

$$m_1 = 1, m_2 = 2, m_3 = 4, \ldots, m_n = 2^{n-1} \text{ intervals..}$$

The values of the step sizes h_k corresponding to m_k are

$$h_k = \frac{(b-1)}{m_k} = \frac{(b-1)}{2^{k-1}}.$$

With this notation, the Composite Trapezoid Rule becomes

$$\int_a^b f(x)\,dx = \frac{h_k}{2}\left[f(a) + 2\left(\sum_{i=1}^{2^{k-1}-1} f(a+ih_k)\right) - \frac{(b-a)}{12}h_k^2 f''(\mu_k)\right].$$

where $\mu_k \in (a,b)$

Let $R_{k,1}$ be the approximation to the integral using $m_k = 2^{k-1}$ intervals.
i.e.

$$R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right] = \frac{(b-1)}{2}\left[f(a) + f(b)\right]$$

$$\begin{aligned}
R_{2,1} &= \frac{h_2}{2}\left[f(a) + f(b) + 2f(a+h_2)\right] \\
&= \frac{(b-a)}{4}\left[f(a) + f(b) + 2f(a + \frac{b-1}{2})\right] \\
&= \frac{1}{2}\left[R_{1,1} + h_1 f(a+h_2)\right]
\end{aligned}$$

notice that when h is halved, all the old points at which the function was evaluated appear in the new computation- we can avoid repeating the evaluations.

$$R_{3,1} = \frac{1}{2}\left\{R_{2,1} + h_2\left[f(a+h_3) + f(a+3h_3)\right]\right\}.$$

$$\vdots$$

$$R_{k,1} = \frac{1}{2}\left\{R_{k-1,1} + h_{k-1}\left[\sum_{i=1}^{2^{k-2}} f(a+(2i-1)h_k)\right]\right\}.$$

We can apply this equation to perform the first step of Romberg Integration for

$$\int_0^1 e^{-x}\,dx = 1 - e^{-1} \approx 0.63212.$$

$$R_{1,1} = \frac{(1-0)}{2}[e^{-0} + e^{-1}] \approx 0.68394$$

$$R_{2,1} = \frac{1}{2}\left[R_{1,1} + \frac{(1-0)}{2}e^{-(0+\frac{1}{2})}\right] = 0.64523$$

$$R_{3,1} = 0.65341$$

$$R_{4,1} = 0.63294$$

We can obtain a faster convergence using Richardson Extrapolation: Notice that:

$$\int_a^b f(x)\,dx - R_{h,1} = \sum_{i=1}^m c_i h^{2i} + O(h^{2m+2})$$

$$= c_1 h^2 + \sum_{i=2}^m c_i h^{2i} + O(h^{2m+2})$$

$$\int_a^b f(x)\,dx - R_{h/2,1} = \sum_{i=1}^m c_i h_{h/2}^{2i} + O(h^{2m+2})$$

$$= \frac{c_1 h^2}{4} + \sum_{i=2}^m \left(\frac{c_i h^{2i}}{4^i}\right) + O(h^{2m+2})$$

Subtracting the first from 4 times the second gives an $O(h_k^4)$ formula:

$$\int_a^b f(x)\,dx - R_{h,2} = \sum_{i=2}^m \frac{c_i}{3}\left(\left(\frac{h^{2i}}{4^i}\right) - h^{2i}\right) + O(h^{2m+2})$$

$$\text{where} \quad R_{h,2} = \frac{R_{h,1} + R_{h/2,1} - R_{h,1}}{3}$$

Of course, this procedure can be repeated to eliminate the $O(h_k^4)$ term from the error. Continuing in this manner, we have an $O(h_k^{2j})$ approximation formula defined by

$$R_{kj} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}.$$

Ex. Use Romberg Integration to approximate

$$\int_0^1 e^{-x}\,dx.$$

to 5 significant digits.

	$R_{h,1}$	$R_{h,2}$	$R_{h,3}$	$R_{h,4}$	$R_{h,5}$
$R_{1,j}$	.6839397				
$R_{2,j}$	.6452352	.6723337			
$R_{3,j}$	.6354094	.6321312	.6321209		
$R_{4,j}$	.6329434	.6321214	.6321206	.6321206	
$R_{5,j}$	.6323263	.6321206	.6321206	.6321206	.6321206

A typical stopping criterion is that both

$$|R_{n-1,n-1} - R_{n,n} \text{ and } |R_{n-2,n-2} - R_{n,n}| < \mathcal{E}.$$

for some error tolerance $\mathcal{E}$.

Note that we may not observe the expected convergence acceleration if

- The integrand f is not sufficiently smooth. We need $f \in C^{2k+2}[a,b]$ to generate the kth row of the table.

- The coefficients $c_1, c_2, \ldots$ are very small. This happens for periodic functions if the interval of integration is an integer multiple of the period or for functions with extremely small derivatives at the endpoints of the interval of integration.

6.13   Adaptive Quadrature

Composative quadrature rules necessitate the use of equally spaced points. This does not take into account that some portions of the curve may have large functional variations that require more attention than other portions of the curve. It is useful to introduce a method that adjusts the step size to be smaller over portions of the curve where a larger functional variation occurs. Thi technique is called adaptive quadrature. We will now discuss an adaptive based on Simpson's Rule. The other composite procedures can be modified in a similar manner.

6.14   Adaptive Quadrature

Composative quadrature rules necessitate the use of equally spaced points. This does not take into account that some portions of the curve may have large functional variations that require more attention than other portions of the curve. It is useful to introduce a method that adjusts the step size to be smaller over portions of the curve where a larger functional variation occurs. Thi technique is called adaptive quadrature. We will now discuss an adaptive based on Simpson's Rule. The other composite procedures can be modified in a similar manner.

We want to approximate

$$\int_a^b f(x)\,dx$$

to within a specified tolerance $\mathcal{E} > 0$.

We will start by applying Simpson's Rule with a step size $h = \frac{(b-a)}{2}$

$$\int_a^b f(x)\,dx = S(a,b) - \frac{h^5}{90}f^{(4)}(\mu)$$

$$\text{where} \quad S(a,b) = \frac{h}{3}\left[f(a) + 4f(a+h) + f(b)\right] \qquad (*)$$

$$\text{and} \quad \mu \in (a,b).$$

We want to know if we should further subdivide the interval, so we need an estimate for the error. Unfortunately, we don't know $f^{(4)}(\mu)$. Instead, we will estimate the error using Simpson's Rule with a step size $\frac{(b-a)}{4}$

$$\int_a^b f(x)\,dx = \frac{h}{6}\left[f(a) + 4f(a + \frac{h}{2}) + 2f(a+h) + 4f(a + \frac{3h}{2}) + f(b)\right]$$
$$- (\frac{h}{2})^4 \frac{(b-a)}{180}f^{(4)}(\tilde{\mu}).$$

for some $\tilde{\mu} \in (a,b)$.

OR

$$\int_a^b f(x)\,dx = S(a, \frac{(a+b)}{2}) + S(\frac{(a+b)}{2}, b) - \frac{1}{16}\frac{h^5}{90}f^{(4)}(\tilde{\mu})$$

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

$$\text{and}\quad S(\frac{(a+b)}{2}, b) = \frac{h}{6}\left[f(a+h) + 4f(a + \frac{3h}{2}) + f(b)\right]$$

Now, we assume $f^{(4)}(\mu) \approx f^{(4)}(\tilde{\mu})$.

Note: if $f^{(4)}$ is continuous, this assumption will hold for sufficiently small h.

Subtract (**) from (*) to obtain

$$0 \equiv S(a,b) - S\left(a, \frac{a+b}{2}\right) - S\left(\frac{a+b}{2}, b\right) - \frac{15}{16}\cdot\frac{h^5}{90}f^{(4)}(\mu)$$

$$\text{or}\,\frac{h^5}{90}f^{(4)}(\mu) \approx \frac{16}{15}\left[\mathcal{I}(a,b) - \mathcal{I}\left(a, \frac{a+b}{2}\right) - \mathcal{I}\left(\frac{a+b}{2}, b\right)\right].$$

$$\therefore\quad |\text{error}| = \left|\int_a^b f(x)\,dx - \mathcal{I}\left(a, \frac{a+b}{2}\right) - \mathcal{I}\left(\frac{a+b}{2}, b\right)\right|$$

$$\equiv \frac{1}{16}\left|\frac{h^5}{90}f^{(4)}(\mu)\right|$$

$$\equiv \frac{1}{15}\left|\mathcal{I}(a,b) - \mathcal{I}\left(a, \frac{a+b}{2}\right) - \mathcal{I}\left(\frac{a+b}{2}, b\right)\right|$$

Thus if we want $|\text{error}| < \mathcal{E}$, we typically insist that

$$\frac{1}{10}|S(a,b) - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)| \le \mathcal{E}.$$

Often, a factor of $\frac{1}{10}$ is used rather than $\frac{1}{15}$ since we had to make the assumption that $f^{(4)}(\mu) \approx f^{(4)}(\tilde{\mu})$ and we prefer to be somewhat conservative in our error estimates.

From last day, we had an error estimate for Simpson's Rule:

$$E = \frac{1}{15}|S(a,b) - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)|.$$

Ex. Compute the Simpson's Rule approximations for

$$\int_a^{1.5} x^2 \ln(x)\, dx = 0.19225935.$$

and calculate the error estimate and the actual error.

In deriving our eror estimate, we had to make the assumption that $f^{(4)}(\mu) \approx f^{(4)}(\tilde{\mu})$, and to compensate for this error we typically insist that

$$\frac{1}{15}|S(a,b) - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)| < \frac{2}{3}\mathcal{E} \equiv \tilde{\mathcal{E}}.$$

Adaptive quadrature methods are built on top of this condition. If the error estimate is less than $\frac{2}{3}$ the desired tolerance $\mathcal{E}$, then

$$S(a, \frac{(a+b)}{2}) + S(\frac{(a+b)}{2}, b).$$

is assumed to be a sufficiently accurate approximation to $\int_a^b f(x)\, dx$

Otherwise, we apply Simpson's Rule to the subintervals $[a, \frac{a+b}{2}]$ and $[\frac{a+b}{2}, b]$. We also use the error estimation in each of the subintervals. If each error estimate is less than $\frac{\tilde{\mathcal{E}}}{2}$, then we sum the approximations to get an approximation of $\int_a^b f(x)\, dx$.

If the approximation on one of the subintervals fails to be within the tolerance $\frac{\tilde{\mathcal{E}}}{2}$ then that subinterval is itself subdivided and the procedure is reapplied to the two subintervals to determine if the approximation on each subinterval is accurate to within $\frac{\tilde{\mathcal{E}}}{2}$.

This recursive procedure is continued until each portion is within the desired tolerance.

6.15 Gaussian Quadrature

Thus far we hvae onlyd ealt with quadrature formulae

$$\int_a^b f(x)\, dx \approx \sum_{j=1}^n a_j f(x_j).$$

that relied on nodes that are equally spaced. This is a nice feature for composite rules because it reduces the number of function evaluations. However, if we allow ourselves to use unequally spaced points, then we can construct quadrature formulas of higher order accuracy.

Notice that with n nodes and n weights we have $2n$ free parameters and we may hope to find an optimal quadrature formula which is exact for polynomials of degree $\leq 2n - 1$.

First, note that an integral

$$\int_a^b f(x)\, dx.$$

over an interval $[a, b]$ can be transformed into an integral over $[-1, 1]$ by using a change of variables

$$\text{let} \quad t = \frac{2x - a - b}{b - a}$$

$$\text{then} \quad x = \frac{1}{2}[(b - a) + a + b]$$

$$\text{and} \quad dx = \frac{1}{2}(b - a)\, dt$$

So

$$\int_a^b f(x)\, dx = \int_{-1}^1 f(\frac{1}{2}[(b - a)t + a + b])\frac{1}{2}(b - a)dt.$$

so without loss of generality, we will consider integrals over the interval $[-1, 1]$.

Suppose that $n = 2$ (2 nodes) and that we want to determine c_1, c_2, x_1, x_2 so that the integration formula

$$\int_{-1}^1 f(x)\, dx = c_1 f(x_1) + c_2 f(x_2).$$

gives the exact result whenever $f(x)$ is a polynomial of degree $2(2) - 1 = 3$.

$$\textbf{i.e.} f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

Since

$$\int_{-1}^1 a_0 + a_1 x + a_2 x^2 + a_3 x^3\, dx$$

$$= a_0 \int_{-1}^1 da + a_1 \int_{-1}^1 x\, dx + a_2 \int_{-1}^1 x^2\, dx + a_3 \int_{-1}^1 x^3\, dx$$

This problem is equivalent to showing that the formula is exact for $f(x) = 1, x, x^2, x^3$.

cases:

Function $f(x)$	Equation
1	$c_1 + c_2 = a_0 \int_{-1}^{1} dx$
x	$c_1 x_1 + c_2 x_2 = \int_{-1}^{1} x\, dx$
x^2	$c_1 x_1^2 + c_2 x_2^2 = \int_{-1}^{1} x^2\, dx$
x^3	$c_1 x_1^3 + c_2 x_2^3 = \int_{-1}^{1} x^3\, dx$

Solving this system gives us that

$$c_1 = 1, c_2 = 1, x_1 = -\frac{\sqrt{3}}{3}, x_2 = \frac{\sqrt{3}}{3}.$$

$\implies$ the approximation formula is

$$\int_{-1}^{1} f(x)\, dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right).$$

This approach can be used to obtain the nodes and coefficients for larger n, but Legendre polynomials can be used to obtain them more easily.

6.15.1 Legendre Polynomials

The Legendre Polynomials are defined according to the following two properties:

1. $P_n(x)$ is a polynomial of degree n

2. $\int_{-1}^{1} P(x) P_n(x)\, dx = 0$ whenever $P(x)$ is a polynomial of degree less than n.

6.16 Initial Value Problems for ordinary Differential Equations

Many natural, scientific and engineering problems can be described in terms of differential equations. Differential equations give us a way to mathematically expres rates of change. We will be considering methods for treating ordinary differential equations (ODEs) in this lecture. ODEs only consider derivatives with respect to one variable.

Ex. Let $y(t)$ denote the number of individuals in a certain population. If this population has a constant growth rate α (the difference between a constant birth rate and death rate), then the differential equation

$$y'(t) = \alpha y(t).$$

with initial condition $y(0) = y_0$ describes the population growth.
Soln.

$$\frac{y'(t)}{y(t)} = \alpha$$
$$D_t[\ln(y(t))] = \alpha$$
$$\ln(y(t)) = \alpha t + \text{const}$$
$$y(t) = ce^{\alpha t} \qquad c = e^{\text{const}}$$

$$y(0) = y_0 \implies y(t) = y_0 e^{\alpha t}.$$

To include other effects such as overcrowding, competition for food, etc, one might introduce a second term into the equation

$$y'(t) = \alpha y(t) - \beta [y(t)]^2.$$

where $\beta > 0$ and β is small. Introducing a nonlinear term makes the problem much more difficult to study analytically.

Indeed, few problems originating from the study of physical phenomena can be solved exactly. We begin by studying numerical methods for approximating the solution $y(t)$ to a problem.

$$\frac{dy}{dt} = f(t, y) \qquad \text{for } a \le t \le b.$$

subject to the initial condition

$$y(a) = \alpha.$$

6.16.1   The elementary theory of initial value problems

We want/need some theoretical results, in particular, we would like to show that solutions to equations exist and are unique.

Def. A function $f(t, y)$ satisfies a Lipschitz condition in the variable y on a set $D \in \mathbb{R}^2$ if a constant $L > 0$ exists such that

$$|f(t, y_1) - f(t, y_2)| \le L|y_1 - y_2| \qquad \text{for all } (t, y_1), (t, y_2) \in D.$$

The constant L is called a <u>Lipschitz constant</u> for f.

Example

Does $f(t, y) = ty$ satisfy a Lipschitz condition on

$$D = \{(t, y) : 0 \le t \le 1, -\infty < y < \infty\}?$$

Soln.

$$|f(t, y_1) - f(t, y_2)| = |-ty_1 + \frac{4t}{y_1} + ty_2 - \frac{4t}{y_2}|.$$

Consider $y_1 = -1, t = 1$, $y_2 \to 0^+$. Under these conditions, we have $|f(t, y_1) - f(t, y_2)| \to \infty$.

RHS $= L|y_1 - y_2| \to L$

L is finite.

We cannot have $|f(t, y_1) - f(t, y_2)| \le L|y_1 - y_2|$ for any finite L

$\therefore$ Lipschitz condition does not hold.

IDK why but he starts off with this example

6.17 Quadrature Formula Example

Find the constants c_0, c_i and x such that the quadrature formula

$$\int_{-1}^{0} f(x)\, dx = c_0 f(-1) + c_i f(x_1).$$

has the highest degree of precision possible.

<u>ans.</u>

Function f	Equation
$f(x) = 1$	$\int_{-1}^{0} 1\, dx = c_0 + c_i$
$f(x) = x$	$\int_{-1}^{0} x\, dx = \left.\dfrac{x^2}{2}\right\vert_{-1}^{0} = c_0(-1) + c_i x_1 = -\dfrac{1}{2}$
$f(x) = x^2$	$\int_{-1}^{0} x^2\, dx = c_0 + c_i x_1^2 = \dfrac{1}{3}$

Next, we do some substutitions:

1. substitute (1) into (2) to eliminate c_0

2. substitute (1) into (3) to eliminate c_0

We now have 2 equations for $c_i + x_1$.
Solve: $c_0 = \frac{1}{4}, c_i = \frac{3}{4}, x_i = -\frac{1}{3}$
Could it be exact for cubics?
LHS $= \int_{-1}^{0} x^3\, dx = -\frac{1}{4}$
RHS $= -\underbrace{c_0}_{\frac{1}{4}} + \underbrace{c_i}_{\frac{3}{4}} \underbrace{\overset{3}{x_i}}_{-\frac{1}{3}} \neq$ LHS
So, no, it cannot be exact for cubics.

6.18   Legendre Polynomials

This approach can be used to obtain the nodes and coefficients for larger n, but Legendre polynomials can be used to obtain them more easily.

The Legendre Polynomials are defined according to the following two properties:

1. $P_n(x)$ is a polynomial of degree n.
   $\implies P_0(x) = 1$

2. $\int_{-1}^{1} P(x)P_n(x)\,dx = 0$ whenever $P(x)$ is a polynomial of degree less than n.

The first few Legendre polynomials are

$P_n(x)$	
$P_0(x)$	1
$P_1(x)$	x
$P_2(x)$	$x^2 - \frac{1}{3}$
$P_3(x)$	$x^3 - \frac{3}{5}x$
$P_4(x)$	$x^4 - \frac{6}{7}x^2 + \frac{3}{35}$

Some properties:

- The roots of these polynomials are distinct

- The roots of these polynomials lie in $(-1, 1)$

- The P_n's are symmetrical about the origin
  $\implies$ the roots are symmetrical about the origin

- The roots of the nth degree Legendre polynomial have the property that they are the nodes needed to produce an integral approximation formula that gives the exact result for any polynomial of degree less than $2n$.

Thm. Suppose $x_1, x_2, \ldots, x_n$ are the roots of the nth degree Legendre polynomial $P_n(x)$ and that for each $i = 1, 2, \ldots, n$ the numbers c_i are defined by

$$c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \, dx.$$

If $P(x)$ is any polynomial of degree less than $2n$ then

$$\int_{-1}^{1} P(x) \, dx = \sum_{i=1}^{n} c_i P(x_i).$$

6.19   A continuation on the Elementary Theory of Initial Value Problems

6.19.1   Lipschitz Conditions

Def. A function $f(t, y)$ satisfies a Lipschitz condition in the variable y on a set $D \in \mathbb{R}^2$ if a constant $L > 0$ exists such that

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \qquad \text{for all } (t, y_1), (t, y_2) \in D.$$

The constant L is called a Lipschitz Constant for f.
Examples can be found in the lecture notes for Lecture 31-b.

6.19.2   Convex Sets

Def. A set $D \in \mathbb{R}^2$ is said to be convex if whenever (t_1, y_1) and (t_2, y_2) belong to D, the point
$$((1 - \lambda)t_1 + \lambda t_2, (1 - \lambda)y_1 + \lambda y_2).$$

also belongs to D for each $\lambda \in [0, 1]$. Geometrically, a set is convex if, for any two points in the set, a line segment connecting them lies entirely within the set. (**i.e.** every point in the set has line of sight to every other point within the set.)
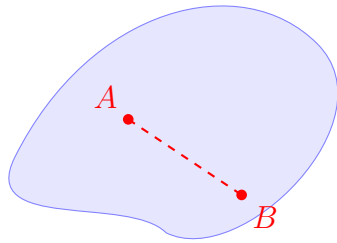
Exercise 1 (22.5)

Show that the set

$$D = \{(t, y) : a \leq t \leq b, -\infty < y < \infty\}.$$

where a and b are constants, is convex.

To prove analytically, we can use the definition of convexity and show that each point falls within the set.

6.19.3 Theorem 1 (22.6)

Suppose $f(t, y)$ is defined on a convex set $D \in \mathbb{R}^2$. If a constant $L > 0$ exists with

$$\left| \frac{\partial f}{\partial y}(t, y) \right| \leq L$$

then for all $(t, y) \in D$ then f satisfies a Lipschitz condition on D in the variable y with Lipschitz constant L.

Proof. Let (t, y_1) and (t, y_2) be in D. Holding t fixed, define $g(y) = f(t, y)$.

Suppose $y_1 \leq y_2$. Since the line joining (t, y_1) to (t, y_2) lies in D and f is continuous on D we have $g \in C[y_1, y_2]$. Furthermore,

$$g'(y) = \frac{\partial f(t, y)}{\partial y}.$$

Using the Mean Value Theorem on g, a number ξ with $y_1 < \xi < y_2$ exists so that

$$g(y_2) - g(y_1) = g'(\xi)(y_2 - y_1)$$
$$\implies f(t, y_2) - f(t, y_1) = \frac{\partial f(t, y)}{\partial y}(y_2 - y_1)$$
$$\implies |f(t, y_2) - f(t, y_1)| \leq L|y_2 - y_1|$$

So f satisfies a Lipschitz condition on D in the variable y with Lipschitz constant L. $\qquad\square$

The previous theorem in combination with the next is particularly fundamental for showing the existence and uniqueness of solutions to ODEs.

6.19.4 Theorem 2 (22.7)

Suppose that $D = \{(t, y) : a \leq t \leq b, -\infty < y < \infty\}$ and that $f(t, y)$ is continuous on D.

If f satisfies a Lipschitz condition on D in the variable y, then the initial value problem

$$y'(t) = f(t, y(t)), \quad a \le t \le b, y(a) = \alpha.$$

has a unique solution $y(t)$ for $a \le t \le b$.

Example (22.8)

Ex. Show that the IVP

$$y' = y \cos t, 0 \le t \le 1, y(0) = 1.$$

has a unique solution.

Soln. Since $f(t, y) = y \cos t$ we have $\frac{\partial f}{\partial y} = \cos t$.

$\implies f$ satisfies a Lipschitz condition in y with $L = 1$ on

$$D = \{(t, y) : 0 \le t \le 1, -\infty < y < \infty\}..$$

Also, f is continuous on D — f is the product of continuous functions and is therefore continuous — so there exists a unique solution.

We also need to know if small changes in the statement of the problem introduce correspondingly small changes in the solution.

6.19.5   Theorem 3 (22.9)

Thm. The initial value problem

$$\frac{dy}{dt} = f(t, y), \quad a \le t \le b, y(a) = \alpha.$$

is said to be a well-posed problem if:

1. A unique solution, $y(t)$, to the problem exists

2. There exist constants $\mathcal{E}_0 \ge 0$ and $k > 0$ such that for any $\mathcal{E}$ with $\mathcal{E}_0 > \mathcal{E} > 0$, whenever $\delta(t)$ is continuous with

   $$|\delta(t)| < \mathcal{E} \quad \text{for all } t \in [a, b]$$

   and when $|\delta_0| < \mathcal{E}$, the initial value problem

   $$\frac{dz}{dt} = f(t, z) + \delta(t), \quad a \le t \le b, \quad z(a) = \alpha + \delta_0$$

has a unique solution $z(t)$ that satisfies

$$|z(t) - y(t)| < k\mathcal{E}$$

for all $t \in [a, b]$.

The perturbed problem assumes the possibility of an error $\delta(t)$ being introduced in the statement of the differential equation as well as an error δ_0 being present in the initial condition. Numerical methods also solve perturbed problems since roundoff errors perturb the original problem. $\implies$ It only makes sense to approximate well-posed problems.

6.19.6   Theorem 4 (22.10)

Thm. Suppose $D = \{(t, y) : a \leq t \leq b, -\infty < y < \infty\}$
   If f is continuous and satisfies a Lipschitz condition int he variable y on the set D, then the initial value problem

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, y(a) = \alpha.$$

is well-posed.

Example (22.11)

Ex. Show that the initial-value problem

$$y' = t^2 y + 1, \quad 0 \leq t \leq 1, y(0) = 1.$$

is well-posed.
   Soln. Since
$$\left| \frac{\partial(t^2 y + 1)}{\partial y} \right| = \left| t^2 \right| \leq 1.$$

and $t^2 y + 1$ is continuous — it's a polynomial in (t, y) — we know that this problem is well-posed.

6.20   Euler's Method (23.1)

Our first numerical scheme for intial value problems will be Euler's Method — a very simple but low order method.

Consider the initial value problem

$$\textbf{IVP} \begin{cases} y' = f(t, y) & a \le t \le b \\ y(a) = y_0 \end{cases}$$

We will compute an approximation to the problem at the mesh points

$$t_k = a + kh, \quad k = 0, 1, \ldots, N.$$

where $h = \frac{(b-a)}{N}$ is called the step size. Here we have assumed h is a constant, although variable step sizes are also useful

Euler's Method can be derived using a Taylor series expansion:

$$y(t_{k+1}) = y(t_k + h) = y(t_k) + hy'(t_k) + \frac{h^2}{2} y''(\xi_k)$$

$$= y(t_k) + hf(t_k, y(t_k)) + \frac{h^2}{2} y''(\xi_k)$$

Euler's Method constructs an approximation

$$w_k \approx y(t_k).$$

by dropping the remainder term.

$$\begin{aligned} w_0 &= y_0 \\ w_k &= w_{k-1} + hf(t_{k-1}, w_{k-1}) \quad 1 \le k \le N \end{aligned}$$

6.21 Euler's Method

This section starts with Euler's Method Error Analysis. The analysis is straightforward, and interesting because it can be extended to the higher order methods that will be discussed in later sections. To derive the proof of convergence, we need the following

Lemma: If s and t are positive real numbers, $\{a_i\}_{i=0}^k$ is a sequence satisfying

$$a_0 \geq -\frac{t}{s}$$
$$a_{i+1} \leq (1+s)a_i + t \quad i = 0, 1, \ldots, k$$

then $a_{i+1} \leq e^{(i+1)s}\left(a_0 + \frac{t}{s}\right) - \frac{t}{s}$

The proof of the lemme is not important, but it is included in section 23.5 of the Chapter 5 lecture notes.

6.21.1 Theorem 1 (23.6)

Suppose f is continuous and satisifes a Lipschitz condition with constant L on

$$D = \{(t,y) : a \leq t \leq b, -\infty < y < \infty\}.$$

and that a constant M exists with the property that

$$|y''(t)| \leq M.$$

Let $y(t)$ denote the unique solution to the initial value problem

$$y' = f(t,y); \quad y(a) = y_0, a \leq t \leq b.$$

and $w_0, w_1, \ldots, w_N$ be the approximations generated by Euler's Method.
Then, for each $i = 0, \ldots, N$,

$$|y(t_i) - w_i| \leq \frac{hM}{2L}\left[e^{L(t_i-a)} - 1\right].$$

Proof. (23.7) (I didn't add it yet)
Note that the theorem <u>requires</u> that

$$|y''(t)| \leq M.$$

The second derivative $y''(t)$ may not be known, but if $\frac{\partial f}{\partial t}$ and $\frac{\partial f}{\partial y}$ exist,

$$y''(t) = \frac{d}{dt}y'(t) = \frac{df}{dt}(t, y(t))$$
$$= \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t))$$

Example (23.8)

What value of h is needed to ensure that $|y(t_i) - w_i| \le 0.1$ for the initial value problem

$$\begin{cases} y' = \frac{2}{t}y + t^2 e^t & 1 \le t \le 2 \\ y = 0 & t = 1. \end{cases}.$$

You are given $y''(t) = (2 + 4t + t^2)e^t - 2e$

Soln. (23.9) $y''(t)$ is increasing and positive on $[1, 2]$, so

$$|y''(t)| \le |y''(2)|$$
$$= 14e^2 - 2e$$
$$= 98.0102$$

$$\text{since} \quad \left| \frac{\partial}{\partial y}\left(\frac{2}{t}y + t^2 e^2 \right) \right|$$
$$\le \left| \frac{2}{t} \right|$$
$$\le 2$$

a Lipschitz Constant for $f(t, y) = \frac{2}{t}y + t^2 e^t$ is $L = 2$.

$$\therefore |y(t_i) - w_i| \le \frac{hM}{2L}\left[e^{L(t_i - 1)} - 1 \right] \le 0.1.$$

we need to choose h so that $\frac{98.0102h}{4}\left[e^{2(2-1)} - 1 \right] \le 0.1.$

$$\implies h \le \frac{0.4}{98.0102(e^2 - 1)} = 0.00064.$$

6.22   The Difference Method

We ned a way to compare the efficiency of different approximation methods. The difference method compares how much the exact solution to the differential equation fails to satisfy the difference equation being used for the approximation.

   Def. The Difference Method

$$w_0 = \alpha$$
$$w_{i+1} = w_i + h\phi(t_i, w_i)$$

has a local trunctation error

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) - (y(t_i) + h\phi(t_i, y(t_i)))}{h}$$
$$= \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i)) \qquad i = 0, 1, \ldots, N-1$$

6.22.1   Example: Euler's Method (23.11)

The difference method for Euler's Method has $\phi = f$.

$$w_0 = \alpha$$
$$w_{i+1} = w_i + h\phi(t_i, w_i) = w_i + hf(t_i, w_i)$$

has local truncation error

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) - y(t_i)}{h} - f(t_i, y(t_i))$$
$$= \frac{y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(\xi)}{den} \qquad \text{we do a taylor expansion}$$

   Local truncation errors are called local because they measure hte accuracy of the method at a specific step, assuming the method was exact at the previous steps. We obviously want the local truncation error to be small. Often, methods for solving ODE's are derived so that the local truncation errors are of the form

$$O(h^p).$$

for the largest possible p, while keeping the number of operations reasonable.

6.22.2   How to obtain improved accuracy?

i.e. a larger p in the $O(h^p)$ local truncation error.

Suppose we want to approximate the solution to the ivp

$$\begin{cases} y' = f(t, y) & a \le t \le b \\ y = \alpha & t = 0. \end{cases}.$$

where $y(t) \in C^{(n+1)}[a, b]$

One approach is to expand the solution in terms of its nth Taylor Polynomial about t_i.

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i) + R$$

$$= y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}f'(t_i, y(t_i)) + \cdots + \frac{h^n}{n!}f^{(n-1)}(t_i, y(t_i)) + R$$

where $\quad R = \dfrac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_i)$

6.23   The Taylor Method of Order n

If we drop the remainder term, we obtain the **Taylor Method of Order n**.

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + hT^{(n)}(t_i, w_i) & i = 0, 1, \ldots, N - 1. \end{cases}$$

where $T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}f'(t_i, w_i) + \cdots + \frac{h^n}{n!}f^{(n)}(t_i, w_i)$ is the nth Taylor Polynomial of f about t_i.

Note: *Euler's Method is equivalent to Taylor's Method of Order* 1.

6.24   The Taylor Method of Order n

If we drop the remainder term, we obtain the **Taylor Method of Order** n.

$$\begin{cases} w_0 = \alpha \\ w_{i+1} = w_i + hT^{(n)}(t_i, w_i) & i = 0, 1, \ldots, N - 1. \end{cases}$$

where $T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}f'(t_i, w_i) + \cdots + \frac{h^n}{n!}f^{(n)}(t_i, w_i)$ is the nth Taylor Polynomial of f about t_i.

Note: *Euler's Method is equivalent to Taylor's Method of Order* 1.

6.24.1   Example: Taylor's Method of Order 2 (24.1)

Use Taylor's Method of order Two to approximate the solution for the IVP

$$\begin{cases} y' = te^{3t} - 2y & 0 \le t \le 1 \\ y = 0 & t = 0. \end{cases}.$$

with $h = 0.5$.

Soln. The first approximation is

$$w_1 = w_0 + h(t_0 e^{3t_0} - 2w + 0) + \frac{h^2}{2}(t_0 e^{3t_0} + 4w_0)$$

$$= 0 + 0.5(0 - 0) + \frac{(0.5)^2}{2}(0 + 1 + 0)$$

$$= 0.125$$

and the second is

$$w_2 = w_1 + h\left(t_1 e^{3t_1 - 2w_1} + \frac{h^2}{2}f\left(t_1, e^{3t_1} + e^{3t_1} + f(w_1)\right)\right)$$

$$= 0.125 + 0.5\left(0.5e^{1.5} - 2(0.125)\right)$$

$$+ \frac{(0.5)^2}{2}\left(0.5e^{1.5} + e^{1.5} + 4(0.125)\right)$$

$$= 2.02323897$$

6.24.2   Intermediate Point Methods (24.2)

If we want to determine an intermediate point (**e.g.** for some $t \in (t_{i-1}, t_i)$), then **Cubic Hermite Interpolation** based on $(y(t_{i-1}), y'(t_{i-1}), y(t_i), y'(t_i))$ is a particularly natural choice for a Taylor Method of degree ≤ 4. Such an interpolation has the advantages that it can be constructed locally and that $y'(t) = f(t, y(t))$ is given.

To interpolate results from very high order Taylor Methods ($n > 4$), we will need higher order oscillating polynomials to preserve the overall accuracy of the results.

6.24.3   Error Analysis fo Taylor's Method (24.3)

The local truncation error for Taylor's Method of Order n is easily derivced:

$$y_{i+1} - y_i - hf(t_i, y_i) - \frac{h^2}{2}f'(t_i, y_i) - \cdots - \frac{h^n}{n!}f^{(n-1)}(t_i, y_i)$$

$$\text{gratuitous cancellations yield} \quad = \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i))$$

$$\text{where} \quad y_i \equiv y(t_i)$$

Thus the local truncation error is

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i) - \frac{h}{2}f'(t_i, y_i) - \cdots - \frac{h^n}{n!}f^{(n-1)}(t_i, y_i)$$

$$= \frac{h^n}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i))$$

Thus, if $y \in C^{(n+1)}[a, b]$
$$\implies y^{(n+1)}(t) = f^{(n)}(t, y(t)) \text{ is bounded}$$
and $\tau_i = \mathcal{O}(h^n)$ for each $i = 1, 2, \ldots, N$.

6.25   Runge-Kutta Methods (24.4)

Taylor Methods are seldom used in practice because they require the computation and evaluation of the derivatives of $f(t, y)$. These evaluations can be complicated and expensive.

Runge-Kutta Methods have the high local truncation error of the Taylor Methods but do not need compute and evaluate the derivatives of $f(t, y)$. To give some idea of how Runge-Kutta methods are developed, we will now show the derivation of a simple second-order method. Here, the increment of w is a weighted average of two estimates of the increment which we will call k_1 and k_2.

$$\begin{cases} w_{n+1} = w_n + ak_1 + bk_2 \\ k_1 = hf(t_n, w_n) \\ k_2 = hf(t_n + \alpha h, w_n + \beta k_1). \end{cases}$$

we can think of k_1 and k_2 as estimates of the change in y when t advances by h because they are the product of the change in t and a value for the slope of the curve.

Runge-Kutta methods often use the simple Euler estimate as the first estimate of δy. Now our problem si to devise a scheme by choosing hte four parameters a, b, α, β. We do so by making the local truncation error of (**??**).

We re-write (**??**) as

$$w_{n+1} = w_n + ahf(t_n, w_n) + bhf(t_n + \alpha h, w_n + \beta h + f(t_n, w_n)).$$

The local truncation error is then

$$\tau_{n+1}(h) = \frac{y_{n+1} - y_n}{n} - af(t_n, y_n) - bf(t_n + \alpha h, y_n + \beta hf(t_n, y_n)).$$

Applying a Taylor Series of degree 2:

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2} \underbrace{f'(t_n, y_n)}_{f_t(t_n, y_n) + f_y(t_n, y_n) \cdot f(t_n, y_n)} + \mathcal{O}(h^3).$$

$$f(t_n + \alpha h, y_n + \beta hf(t_n, y_n))$$
$$= f(t_n, y_n) + f_t(t_n, y_n)\alpha h + f_y(t_n, y_n)f(t_n, y_n)\beta h + \mathcal{O}(h^2)$$

$$\therefore \tau_{n+1}(h) = (1 - a - b)f(t_n, y_n)$$
$$+ h(\frac{1}{2} - \alpha b)f_t(t_n, y_n)$$
$$+ h(\frac{1}{2} - \beta b)f_y(t_n, y_n)f(t_n, y_n)$$

Thus the local truncation error will be $\mathcal{O}(h^2)$ provided

$$a + b = 1$$
$$\alpha b = \frac{1}{2}$$
$$\beta b = \frac{1}{2}$$

but there is not enough flexibility to obtain a third order method. (Proof is left as an exercise.)

6.25.1   Examples of Runge-Kutta Methods (24.7)

The Midpoint Method

$$\left\{ \begin{array}{l} a = 0 \\ b = 1 \\ \alpha = \frac{1}{2} \\ \beta = \frac{1}{2} \end{array} \right. \implies \left\{ \begin{array}{l} w_0 = y(t_0) \\ w_{n+1} = w_n + hf(t_n + \frac{h}{2}, w_n + \frac{h}{2}f(t_n, w_n)) \end{array} \right.$$

The Modified Euler Method

$$\left\{ \begin{array}{l} a = \frac{1}{2} \\ b = \frac{1}{2} \\ \alpha = 1 \\ \beta = 1 \end{array} \right. \implies \left\{ \begin{array}{l} w_0 = y(t_0) \\ w_{n+1} = w_n + \frac{h}{2}[f(t_n, w_n) + f(t_{n+1}w_n + hf(t_nw_n))] \end{array} \right.$$

.

Heun's Method

$$\left\{ \begin{array}{l} a = \frac{1}{4} \\ b = \frac{3}{4} \\ \alpha = \frac{2}{3} \\ \beta = \frac{2}{3} \end{array} \right. \implies \left\{ \begin{array}{l} w_0 = y(t_0) \\ w_{n+1} = w_n + \frac{h}{4}\left[f(t_n, w_n) + 3f\left(t_n + \frac{2}{3}h, w_n + \frac{2}{3}hf(t_nw_n)\right)\right] \end{array} \right.$$

.

6.26 Higher order Runge-Kutta Methods (24.8)

Third order Runge-Kutta methods are not commonly used. However, fourth order Runge-Kutta methods are widely used and derived in a similar fashion. Greater complexity results from having to compare terms through h^4 and this gives a set of 11 equations in 13 unknowns. The set of equations can be solved with 2 unknowns being chosen arbitrarily.

The most commonly used set of values leads to the following algorithm:

$$w_{n+1} = w_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$
$$k_1 = hf(t_n, w_n)$$
$$k_2 = hf\left[t_n + \frac{1}{2}h, w_n + \frac{1}{2}k_1\right]$$
$$k_3 = hf\left[t_n + \frac{1}{2}h, w_n + \frac{1}{2}k_2\right]$$
$$k_4 = hf[t_n + h, w_n + k_3]$$

The main computational effort in applying Runge-Kutta methods is the evaluation of f. In the second order methods, the local truncation error is $\mathcal{O}(h^2)$ and the cost is two functional evaluations per step. The Runge-Kutta method of order four requires four evaluations per step and the local truncation error is $\mathcal{O}(h^4)$.

We may wonder about higher order formulas...

6.26.1 Error Analysis of Higher Order Runge-Kutta Methods (24.9)

Butcher has shown that the following relationship holds:

evaluations	2	3	4	$5 \leq n \leq 7$	$8 \leq n \leq 9$	$10 \leq n$
Best LTE	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^{n-1})$	$O(h^{n-2})$	$O(h^{n-3})$

This indicates why methods of order ≤ 5 are often used rather than higher order methods with a larger step size.