NuMe Noter

Magnus Goltermann

Indhold

1 Mathematical preliminaries and computer arithmetic

1.1 Number systems

Form of any number can be given on the form

$$\pm(d_N \dots d_0)_b = s(\sum_{i=0}^{N} d_i \cdot b^i)_{10}$$

Where s is the sign, d_i is how many of the i'th number that is given and b is the base. $b = 2$ is binary and $b = 10$ i decimal. d_i can always take any number between $[0, b-1]$

1.2 Decreasing power method

Converts decimal to binary.
Choosing the largest n where $d_{10} - 2^n \geq 0$, subtract to that and repeat. Example with 175:

$$d_{10} - 2^n \geq 0 \iff 175 - 128 \geq 0$$

Flip the bit at $2^7 = 128$ and repeat with $175 - 128 = 47$

$$d_{10} - 2^n \geq 0 \iff 47 - 32 \geq 0$$

Flip the bit at $2^5 = 32$ and repeat with $47 - 32 = 15$

$$d_{10} - 2^n \geq 0 \iff 15 - 8 \geq 0$$

Flip the bit at $2^3 = 8$ and repeat with $15 - 8 = 7$

$$d_{10} - 2^n \geq 0 \iff 7 - 4 \geq 0$$

Flip the bit at $2^2 = 4$ and repeat with $7 - 4 = 3$

$$d_{10} - 2^n \geq 0 \iff 3 - 2 \geq 0$$

Flip the bit at $2^1 = 2$ and repeat with $3 - 2 = 1$

$$d_{10} - 2^n \geq 0 \iff 1 - 1 \geq 0$$

Flip the bit at $2^0 = 1$ and repeat with $1 - 1 = 0$ The method is done and we have $175 = 128 + 32 + 8 + 4 + 2 + 1 = (10101111)_2$

1.3 Long division method

Divide by 2, flip the bit if remainder is 1 and continue until it reaches 0. The bits are "reversed"so we find the least significant first and the most significant last (smallest to largest.) Example with 175:

$$175 = 87 \cdot 2 + 1$$

$$87 = 43 \cdot 2 + 1$$

$$43 = 21 \cdot +1$$

$$21 = 10 \cdot +1$$

$$10 = 5 \cdot 2 + 0$$

$$5 = 2 \cdot 2 + 1$$

$$2 = 1 \cdot 2 + 0$$

$$1 = 0 \cdot 2 + 1$$

Therefore $175 = (10101111)_2$ (reading from the bottom and up!).

1.4 Floats

Floats can be given on the form

$$\pm(d_n \dots d_0, d_{-1}, d_{-2} \dots d_{-m})_b = s \sum_{i=-m}^{n} d_i \cdot b^i$$

Example: $(101, 01)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0, 0 \cdot 2^-1 + 1 \cdot 2^-2 = 4 + 1, \frac{1}{4} = 5,25$

1.4.1 Method to convert binary float to decimal float

Multiply the binary number m times with 2, until it is not a float. Convert it to decimal and then divide by 2^m. Example

$$(101,01)_2 \cdot 2 = (1010,1)_2$$

$$(1010,1)_2 \cdot 2 = (10101)_2$$

Then $(101,01)_2 = \frac{(10101)_2}{2^2} = (5,25)_{10}$

1.4.2 Multiplication algorithm (Decimal to binary)

Multiply by 2, if the new number is > 1 then flip the bit, and reapeat with the number after the comma until its 0 after the comma. Example:

$$0,8125 \cdot 2 = 1,625 \rightarrow 1$$

$$0,625 \cdot 2 = 1,25 \rightarrow 1$$

$$0,25 \cdot 2 = 0,5 \rightarrow 0$$

$$0,5 \cdot 2 = 1,0 \rightarrow 1$$

Therefore $(0,8125)_{10} = (0,1101)_2$

1.4.3 Infinity long comma numbers

When using the Multiplication algorithm we can sometimes run into an infinite loop (e.g. $0,1$ to binary).

1.4.4 Infinity long comma numbers to fraction

Find how many times, m, you have to multiply 2, to get the repeating sequence to right after the comma.
Then find how many time, k, you have to multiply 2 with, to get the repeating sequence right before the comma.
The fraction will then be given as

$$\frac{x}{y}$$

Where x is the repeating sequence in binary and y is $2^k - 2^m$. Example:

$\frac{x}{y} = (0,000\ \overline{1100})_2$

$\times 2^3 = (0,\overline{1100})_2 \cdot y$

$\times 2^7 = (1100,\overline{1100})_2 \cdot y$

$(2^7 - 2^3)x = (1100)_2 \cdot y$

$\frac{x}{y} = \frac{(1100)_2}{(2^7-2^3)} = \frac{(12)_{10}}{(120)_{10}} = \frac{(1)_{10}}{(10)_{10}} = (\frac{1}{10})_{10}$
$= (0,1)_{10}$

1.5 Truncating error

When floats are to large we can truncate (cut of) at a certain point. The truncating error is smaller than the point just before where we cut of:

$$|(x)_b - (\hat{x})_b| < b^{-n}$$

Where x is the part we keep, $\hat{x}$ is the part we loose and n is digits truncation (how many numbers after the comma did we keep).

1.6 Rounding

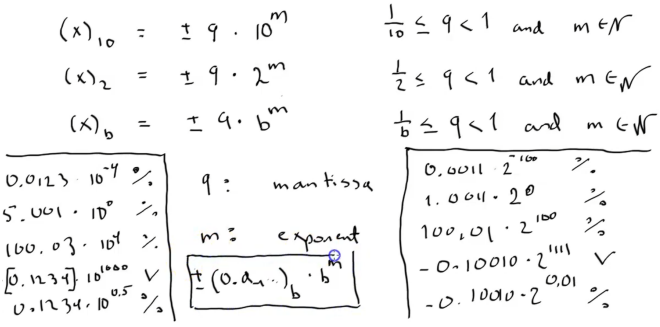
If the number after the point we want to truncate at is larger than 5, then add one to the number we truncate at.
The rounding error is

$$|(x) - (\tilde{x})| \leq \frac{1}{2} b^{-n}$$

Which is half of the truncating error, since the value at $n+1$ lost is at most $0,5$ with rounding where with truncating it is at most 1.

1.7 Normalized scientific notation

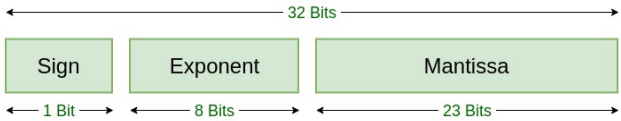
Consists of a mantissa and an exponent. The mantissa is normalized, so can only have a given number.



It is easy to get the mantissa to a normalized form just by adding of subtracting from the exponent. Example:
$5,001 \cdot 10^0$ can be normalized to $0,5001 \cdot 10^1$

1.8 Machine number

Machine numbers are used to represent binary floats in a computer. using normalized scientific notation for binay



Single Precision
IEEE 754 Floating-Point Standard

The mantissa is now f where we remember the to put $(1, f)$ to improve accuracy instead of just $(0, mantissa)$.
Converting machine number to decimal is given as

$$x = (-1)^s \cdot (1, f) \cdot 2^e$$

Where $m = e - 1023$, e is the exponent, f is the mantissa and s is the sign bit.

1.8.1 Absolute error when converting to closest machine number

With a 64 bit machine number and using the defintion for rounding error the error is given as

$$|x - \tilde{x}| \leq 2^{m-53}$$

1.8.2 Relative error when converting to closest machine number

If the machine number is "rounding down"to x^- then it is

$$|\frac{x - x^-}{x}| \leq \frac{2^{m-53}}{q \cdot 2^m} = \frac{1}{q} \cdot 2^{-53}$$

and since $1 \leq q < 2$ the relative error is

$$|\frac{x - x^-}{x}| \leq 2^{-53}$$

When rounding up to x^+ it is the same

$$|\frac{x - x^+}{x}| \leq 2^{-53}$$

Relative error is always the same, but the absolute error is larger when the number is smaller.

1.8.3 Roundoff error with operations

$$fl(x) = x(1 + \delta_x)$$

where fl is the function that converts the float to a machine number with a relative error of δ_x.

The operations of x and y therefore has the error given as:

$$\delta_{add} = \frac{x\delta_x + y\delta_y}{x+y}$$

$$\delta_{sub} = \frac{x\delta_x - y\delta_y}{x-y}$$

$$\delta_{mul} = \delta_x + \delta_y$$

$$\delta_{div} = \delta_x - \delta_y$$

1.9 Significant digits

Numbers after the comma we are interested in are significant digits.
Relative error when truncating at the significant digits can be calculated as:

$$\frac{x_{Trunc} - x_{real}}{x_{real}}$$

1.10 Loss of significance theorem

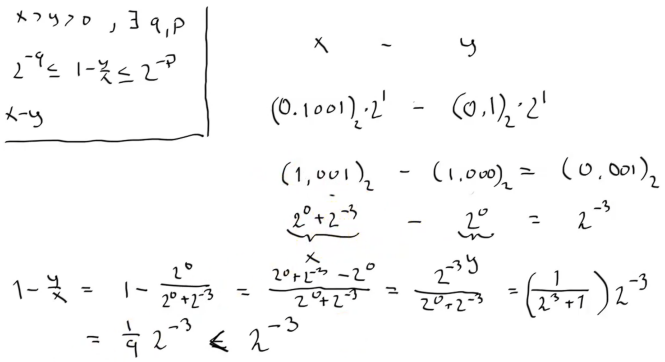
We will lose at least p bits and at most q bits in precision when doing the operation $x - y$ Where

$$x = r \cdot 2^n, \text{ where } \frac{1}{2} \leq r < 1$$

$$y = s \cdot 2^m, \text{ where } \frac{1}{2} \leq s < 1$$

$$2^{-q} \leq 1 - \frac{x}{y} \leq 2^{-p}$$

Example of use:



Therefore the operation $(0,1001)_2 \cdot 2^1 - (0,1)_2 \cdot 2^1$ looses at most 2^{-3} bits.

2 Non-Linear Equations

2.1 Bisection method

To find root of a function.

Given an interval of between a and b, compare the point in the middle,c $c = \frac{a+b}{2}$, with $f(a) \cdot f(c)$ and $f(b) \cdot f(c)$, and then choose the interval where $f(c) \cdot x$ is < 0. Repeat for a given number of iterations or if a condition/stop criteria is reached like $|f(c)| < \epsilon$ (y going close to 0) or $|b - a| < \delta$ (interval getting very small).
Precondition: $f(a) \cdot f(b) > 0$ (has to be a root between a and b). If there are more roots in the interval, it will only find one.

2.1.1 Global convergence

The interval will always get halved each iteration, and therefore will the size of the interval after n iteration be

$$(b_n - a_n) = (\frac{1}{2})^n (b_0 - a_0)$$

We can therefore solve how many iterations to do to recieve a given precision δ:

$$(\frac{1}{2})^n (b_0 - a_0) < \delta$$

And since the interval goes towards 0, it will always find a root.

2.1.2 Absolute error

The error must be less than or equal to half of the interval after n iterations, using c as the estimate:

$$|e_n| = |c_n - root| \leq \frac{1}{2}(b_n - a_n)$$

And from previous we knoow the size after n iterations, and therefore the error will just be that times a half:

$$(\frac{1}{2})^{1+n}(b_0 - a_0) \leq \delta$$

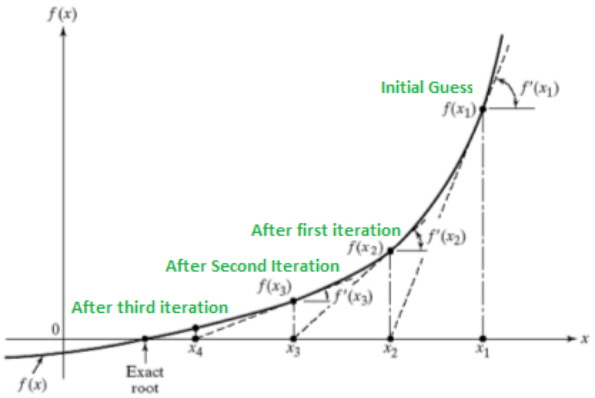
2.1.3 Convergence rate

If we know the error after k iterations, then the k+1 step must have half the error. This follows a linear convergence rate.

2.2 Newtons method

Starting at an x and corresponding $f(x)$ value, get the tangent at $f(x)$, and then use the intercept between the tangent and the x-axis as the new x, and repeat. The update is given as

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



The algorithm is to stop at either max iterations, $f(x) < \epsilon$ (close to 0) or if $|x_{n+1} - x_n| < \delta$ (if it does not get much closer).
Precondition: $f'(x) \neq 0$

2.2.1 Error

The error of the Newtons method is defind as

$$e_n = x_n - root$$

Using the newton update we know that

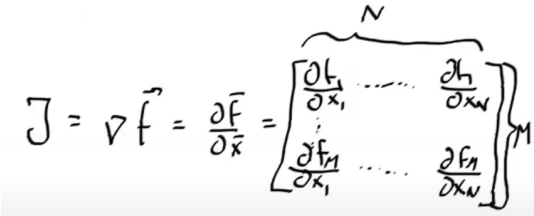
$$e_{n+1} = e_n - \frac{f_n}{f'_n}$$

2.2.2 Convergence

Newtons method converges quadratically $O(n^2)$.

2.2.3 For rooting non-linear vector function

Given a vector function $\bar{f}$, a function that calculated the gradient (Jacobian matrix) of the function $\Delta\bar{f}$ and a vector containing the start values $\vec{x_0}$. First calculate the J matrix given by



Then solving the system of equations

$$J\bar{h} = -\bar{f}(x_0)$$

The new estimation is then given by

$$\bar{x_1} = \bar{x_0} + \bar{h}$$

Continue until max iteration, $|\bar{f}(\bar{x_1})| < \epsilon$ or if $|\bar{x_1}) - \bar{x_0})| < \delta$

2.3 Evaluation of polynomials

A polynomial of degree n is given by

$$p(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

Where a is a list of coefficients.
Calculating this naively will use $\frac{n^2+n}{2} + (n-1)$ flops (operations), which makes it $O(n^2)$.

2.3.1 Nested multiplication

Take a polonium, factorize by taking an x out of the parenthesis and adding the constant, and repeat. Example:

$$
\begin{aligned}
P(z) &= 3z^4 + 2z^3 - 3z^2 + 2z - 1 \\
&= z\underbrace{(3z^3 + 2z^2 - 3z + 2)}_{=b_0} - 1 \\
&= z(z\underbrace{(3z^2 + 2z - 3)}_{=b_1} + 2) - 1 \\
&= z(z(z\underbrace{(3z + 2)}_{=b_2} - 3) + 2) - 1 \\
&= z(z(z(z\underbrace{(3)}_{=b_3} + 2) - 3) + 2) - 1
\end{aligned}
$$

$$
\begin{aligned}
b_3 &= a_4 \\
b_2 &= z \cdot b_3 + a_3 \\
b_1 &= z \cdot b_2 + a_2 \\
b_0 &= z \cdot b_1 + a_1 \\
P(z) = b_{-1} &= z \cdot b_0 + a_0
\end{aligned}
$$

2.4 Horners method

Horners method is defined as

$$
\begin{aligned}
b_{n-1} &= a_n \\
b_{n-2} &= z\, b_{n-1} + a_{n-1} \\
&\vdots \\
b_0 &= z\, b_1 + a_1 \\
P(z) = b_{-1} &= z\, b_0 + a_0
\end{aligned}
$$

Or more intuitively in a table form:

	a_n	a_{n-1}	...	a_1	a_0
z		$z\,b_{n-1}$		$z\,b_1$	$z\,b_0$
	b_{n-1}	b_{n-2}		b_0	b_{-1}

Example for finding $p(2)$ when $p(z) = 3z^4 + 2z^3 - 3z^2 + 2z - 1$:

	3	2	-3	2	-1
2		6	16	26	56
	3	8	13	28	55

The method uses nested multiplication and only uses a linear number of flops, and is therefore $O(n)$

2.4.1 Polynomial division relationship

Given a polynomial p with degree n and q with degree n-1 given with

$$p(z) = \sum_{i=0}^{n} a_i \cdot z^i \text{ and } q(z) = \sum_{i=0}^{n-1} b_i \cdot z^i$$

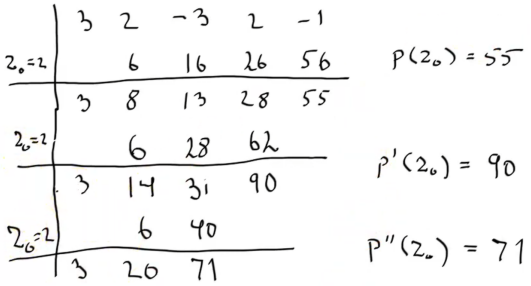
Then the relationship is

$$p(z) = (z - z_0)q(z) + p(z_0)$$

Where $(z - z_0)$ is how many times $q(z)$ can be multiplied by to get $p(z)$ with $p(z_0)$ as the remainder.

2.4.2 Horner derivatives

From the relation above we have $p'(z_0) = q(z_0)$ (z_0 is given as a value).
When using Horners method, the bottom of the table (b values) can be used to find the derivative value of p, and can be continued to find second derived and so on.
Example:



2.5 Localization theorem

Roots of a given polynomial is within the disc at $(0, 0)$ with radius ρ which is given as

$$\rho = 1 + \frac{1}{|a_n|} max_{0 \leq k \leq n}\{|a_k|\}$$

Where $c = max_{0 \leq k \leq n}\{|a_k|\}$, which is just the largest coefficient in absolute value.

2.5.1 Using the "reversed"

The "reversed"polynomial is defined by a polynomial which have the exponents reversed:
If $p(z) = \sum_{i=0}^{n} a_i \cdot z^i$ then the "reversed is given by

$$s(z) = z^n \cdot p(\frac{1}{z}) = a_n + a_{n-1}z + \ldots a_0 z^n$$

The two polynomials have the same roots, which means we can extend the localization theorem such that the roots can not be with the circle with a radius of $\frac{1}{\rho}$:

3 Systems of Linear Equations

3.1 Solving linear systems

Solving the equation $Ax = b$. If A has numbers at all positions, it is dense, otherwise it is sparse.
If dense, then use a factorization method, and if sparse use a iterative method. Factorization methods are exact, but costly time and memory wise. Iterative methods are fast and does not cost much memory, but are approximations.

3.2 Solve using factorization

3.2.1 With LU factorization

If $A = LU$ then we have to solve $LUx = b$. We can then set $Ux = z$, and then solve $Lz = b$ for z using forward substitution and then solve $Ux = z$ using backwards substitution.

3.2.2 With QR factorization

If $Ax = b$ and $A = QR$ then we have to solve $QRx = b$.
Set $Rx = z$, and solve $Qz = b$. This can be done easily since Q is orthonormal, which has the property $Q^TQ = I$, and therefore the solution is

$$Qz = b \iff Q^TQz = Q^Tb \iff z = Q^Tb$$

Then solve $Rx = z$ using backwards substitution.

3.3 LU factorization - Crout

Factorized into a lower (L) and an upper (U) matrix. In crout the diagonal of the U matrix is 1

Using matrix multiplication to make a system of equations to solve in specific order. e.g for A_{00} to get U_{00}:

$$A_{00} = L_{00} \cdot U_{00} + 0 \cdot 0 + 0 \cdot 0 \cdots \iff A_{00} = L_{00} \cdot 1$$

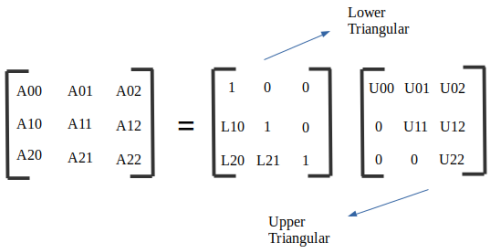
Then we can solve for L_{10}:

$$A_{10} = L_{10} \cdot U_{00} + 1 \cdot 0 + 0 \cdot 0 \cdots \iff A_{10} = L_{10}$$

The general method is to first solve a column in L, then a row in U and so on. You will get more and more parts in every equation along the way, but always only one unknown.

3.3.1 Doolittle

Like crout but with 1 on the diagonal of the L matrix.



Opposite "direction"than crout, so find a row in U first, then a column in L and so on.

3.4 Cholesky factorization

Works for a quadratic symmetric matrix (and positive definite)

$$A = LL^T$$

L is a lower triangular matrix. Using same principle as LU factorization with matrix multiplication.

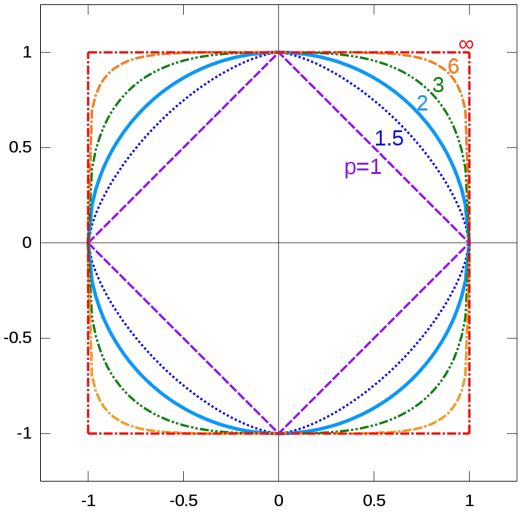
$$A_{00} = L_{00}L_{00} \iff \sqrt{A_{00}} = L_{00}$$

If A is an 3x3 matrix then L is defined with

$$\mathbf{L} = \begin{pmatrix} \sqrt{A_{11}} & 0 & 0 \\ A_{21}/L_{11} & \sqrt{A_{22} - L_{21}^2} & 0 \\ A_{31}/L_{11} & \left(A_{32} - L_{31}L_{21}\right)/L_{22} & \sqrt{A_{33} - L_{31}^2 - L_{32}^2} \end{pmatrix}$$

3.5 Norm

All points when a norm of a vector (when equal to 1, e.g. $||\bar{x}||_2 = 1$) can be illustrated as



3.5.1 2 norm of vector

Defined by

$$||\bar{x}||_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$$

Corresponds to the euclidean length of the vector.

3.5.2 Infinity norm of vector

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

Largest absolute element.

3.5.3 1 norm of vector

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

Sum of all the values in absolute termns

3.6 Matrix norm

3.6.1 Infinity norm of a matrix

Given as

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

Take the sum of all rows (using absolute values) in a matrix, then the infinity norm is the largest of the row sums.

3.6.2 1 norm of a matrix

Given as

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

Take the sum of all columns (using absolute values) in a matrix, then the infinity norm is the largest of the column sums.

3.6.3 2 norm of a matrix

Given as

$$||A||_2 = max_{1 \leq i \leq n} |\sigma_i|$$

Where σ is a singular value of A. or as

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

Where ρ is a function that gives the spectral radius. Where

$$\rho(A^T A) = max_{1 \leq i \leq n} \lambda_i$$

Where λ is an eigenvalue of $A^T A$. It just finds the largest eigenvalue of $A^T A$. In short: The 2 matrix norm is the square root of largest eigenvalue of $A^T A$

3.7 Condition number

When solving a linear system of equation like

$$Ax = b$$

Then when there is an error in either A or b, then the error is "scaled"by a condition number. The bigger the condition number, the less useful are the answers. The condition number is given by

$$K(A) = ||A^{-1}|||A|||$$

3.8 Neumann series

A is quadrtric and $||A|| < 1$ then $I - A$ is invertible and then the following is true

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$$

And for some matrix B where $B = I - A$ then the inverse of B is just

$$B^{-1} = (I - A)^{-1} = \sum_{k=0}^{\infty} A^k$$

3.8.1 Using nested multiplication

To reduce the number of matrix multiplications we can calculate then sum using nested multiplication: Going form $O(n^3)$ to a faster $O(n^2)$, and if A is sparse we can get $O(n)$.

$$(I-A)^{-1} \simeq A^3 + A^2 + A^1 + A^0$$
$$A(A^2 + A^1 - A^0) + I$$
$$A(A(A^1 + A^0) + I) + I$$
$$A(A(A(A^0) + I) + I) + I$$

3.9 Splitting matrix Q

A linear system of equations is given as $Ax = b$ then

$$Ax = b \iff Qx = (Q - A)x + b$$

A can be divided into a lower triangular matrix plus the diagonal plus an upper triangular matrix as

$$A = L + D + U$$

The idea is then to choose a Q matrix and the iterate on

$$Qx^{k+1} = (Q - A)x^k + b \iff x^{k+1} = (I - Q^{-1}A)x^k + Q^{-1}b$$

Where x^k is an approximate solution to $Ax = b$ (we only use the above notation as theoretical, since calculating Q^{-1} is very heavy computationally).

3.9.1 Jacobi method

Using the definitions of splitting matrix, and setting $Q = D$ then we have

$$Qx^{k+1} = (Q-A)x^k+b \iff Dx^{k+1} = (D-L-D-U)x^k+b \iff Dx^{k+1} = b-(L+U)x^k$$

The i'th element in x can then be defined as

$$x_i^{k+1} = \frac{b_i - \sum_{j=0}^{i-1} L_{ij}x_j^k - \sum_{j=i+1}^{n} U_{ij}x_j^k}{D_{ii}}$$

3.9.2 Gauss-Seidel method

Setting $Q = D + L$:

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

The i'th element in x can then be defined as

$$x_i^{k+1} = \frac{b_i - \sum_{j=i+1}^{n} U_{ij}x_j^k - \sum_{j=0}^{i-1} L_{ij}x_j^k}{D_{ii}}$$

3.9.3 SOR method

Setting $Q = \frac{1}{w}(D + wL)$

$$(\frac{1}{w}(D + wL))x^{k+1} = (\frac{1}{w}(D + wL) - L - D - U)x^k + b = b - Ux^k$$

$$\iff (D + wL)x^{k+1} = ((1 - w)D - wU)x^k + wb$$

The i'th element in x can then be defined as

$$x_i^{k+1} = \frac{wb_i + D_{ii}x_i^k - wD_{ii}x_i^k - w\sum_{j=i+1}^{n} U_{ij}x_j^k - w\sum_{j=1}^{i-1} L_{ij}x^k}{D_{ii}}$$

Simplified knowing the residual to be $r = b - Ax^k$:

$$x_i^{k+1} = x_i^k + w\frac{r_i}{D_{ii}}$$

Where $0 < w < 2$. When $w = 1$ it is just the Gauss-Seidel method.

4 Interpolation

4.1 Theorem of polynomial interpolations

If we have n distinct coordinate pairs, then we can find a polynomial of degree $n - 1$ that will pass all points. There exists only one unique polynomial that satisfy this for every case.
But there exists infinitely many polynomials of degree $n \leq$ that also satisfy this, but we always prefer a polynomial with fewest degrees.

4.2 Secret sharing

If 2 people are both given a coordinate and a secret value is hidden at e.g. $f(0) = secret$, where the secret is located such that if the two people make a straight line between their points, it will pass the secret point. They can therefore only know the secret point if they combine their points. This can easily be extended to higher degree polynomials, such that more people need to know a point or such that more people are needed to reveal the secret.

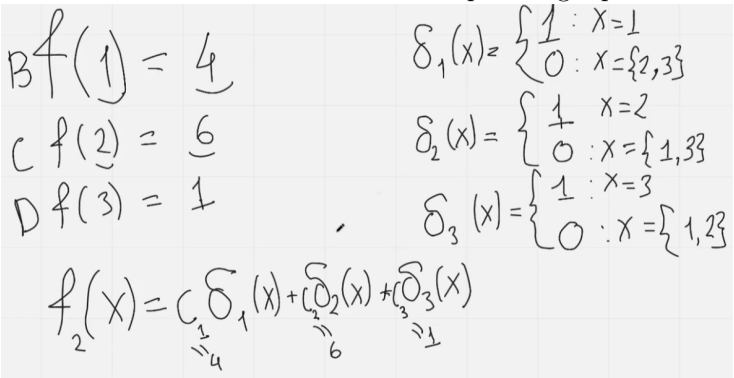
4.3 Lagrange interpolation

The lagrangian interpolation is defined on the form

$$f(x) = c_1\delta_1(x) + c_2\delta_2(x) + c_3\delta_3(x) + \cdots + c_n\delta_n(x)$$

One delta function for every point the polynomial passes trough, and c_1 is the coefficent such that $c_1 = y_1$. For example δ_1 that to equal $\delta_1(x_1) = 1$ and $\delta_1(x_2) = 0$ such that whenever an x value corresponds to a point in the data

set, only one delta function gives the correct y value with the coefficient, and all other delta functions are 0. An example using 3 points:



It can formally be defind as

$$p(x) = \sum c_i \delta_i(x) = \sum_i f(x_i) \cdot \Pi_j \frac{x - x_j}{x_i - x_j} =$$

An example using 4 points:

$$x_0 = 5, x_1 = 6, x_2 = 9, x_3 = 11$$

$$y_0 = 12, y_1 = 13, y_2 = 14, y_3 = 16$$

$$y = f(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} \times y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} \times y_1$$

$$+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \times y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \times y_3$$

$$= \frac{(x-6)(x-9)(x-11)}{(5-6)(5-6)(5-11)}(12) + \frac{(x-5)(x-9)(x-11)}{(6-5)(6-9)(6-9)}(13)$$

$$+ \frac{(x-5)(x-6)(x-11)}{(9-5)(9-6)(9-11)}(14) + \frac{(x-5)(x-6)(x-9)}{(11-5)(11-6)(11-9)}(16)$$

4.4 Newton divided difference method

The newton divided difference method uses the idea that when given n points to interpolate, that can result in a system of equations with n equations and n unknowns. To solve this system the method uses a diveded difference method, formally as

$$f_n(x) = c_0 + c_(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots c_n(x - x_0)(x - x_1)\ldots(x - x_n)$$

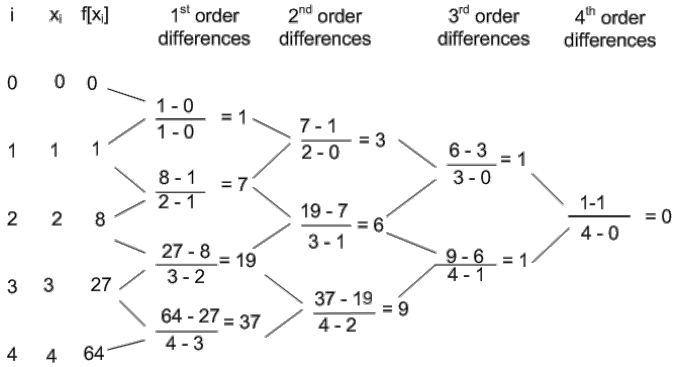
$$c_0 = f[x_0] = f(x_0) = y_0$$

$$c_1 = f[x_1, x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$c_2 = f[x_2, x_1, x_0] = \frac{f[x_1, x_1] - f[x_1, x_0]}{x_2 - x_0}$$

$$c_3 = f[x_3, x_2, x_1, x_0] = \frac{f[x_3, x_2, x_1] - f[x_2, x_1, x_0]}{x_3 - x_0}$$

And so on.
When doing the method, it is best to do it in a table, like the following example:



To find the correct denominator, it can easily be found by first going down and left as far possible from the current position and then minus that by the x value going back an up as far as possible. As an example in the example above in the first 2nd order difference to get the denominator $2-0$: Go to left-down-left-down and then 2 i reached. Go up-left-up-left and then 0 is reached.

When the table is done, the polynomial can be made from the first definition by using the top row as c_i. Using the example above
$c_0 = 0$, $c_1 = 1$, $c_2 = 3$, $c_3 = 1$ and $c_4 = 0$

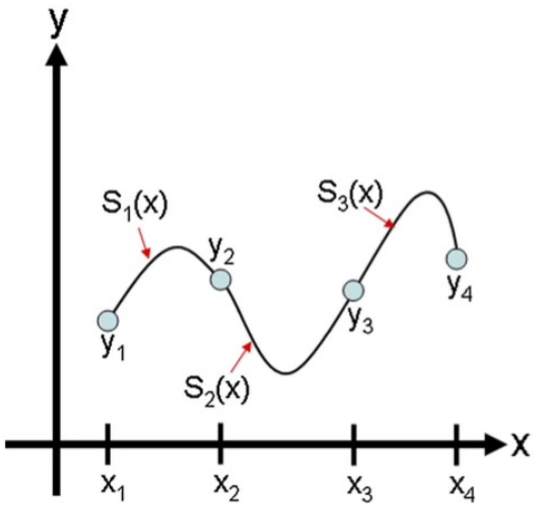
$$p(x) = 0+1(x-0)+3(x-0)(x-1)+1(x-0)(x-1)(x-2)+0(x-0)(x-1)(x-2)(x-3)$$

4.5 Spline interpolation

Instead of making one polynomial to interpolate all points, we just make a straight line between all points to make a linear spline interpolation. The line between every two points ($x_i \to x_{i+1}$) are defined as

$$f(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

This makes in non-differentiable.
Instead we can make quadratic spline interpolation using second degree polynomials. In order for it to be differentiable we want the connection between two lines (at every point) to have the same slope. E.g. the splines $x_{i-1} \to x_i \to x_{i+1}$ then the 2 splines meeting at x_i should have the same slope at (x_i, y_i)



To calculate the two splines, we need to do it simultanous to have the same slope. We can define the two splines as

$$y_{-1} = a_{-1,0} + a_{-1,1}x + a_{-1,2}x^2$$

$$y_1 = a_{1,0} + a_{1,1}x + a_{1,2}x^2$$

Then we need the following to be true

$$\frac{d}{dx}(a_{-1,0} + a_{-1,1}x + a_{-1,2}x^2) = \frac{d}{dx}(a_{1,0} + a_{1,1}x + a_{1,2}x^2)$$

And we know it has to be true at x_i therefore $x = x_i$

$$\Longleftrightarrow$$

$$a_{-1,1} + 2a_{-1,2}x_i = a_{1,1} + 2a_{1,2}x_i$$

To be able to have a system with as many unknowns as equations, we have to set one of the parameters to begin with (usually $a_{2,0}$ such that the first spline is linear.)

4.5.1 Example of quadratic spline interpolation

Example with the points $(5, 6), (6, 5), (9, 5)$:
First we choose to make $a_{0,2} = 0$ to get the correct number of unknowns (can be any number, but for simplicity we choose 0).

The splines meet at $(6, 5)$ so we can make two the two equations from point one $(5, 6)$ and two $(6, 5)$

$$6 = a_{0,0} + a_{0,1} \cdot 5$$

$$5 = a_{0,0} + a_{0,1} \cdot 6$$

Because first spline must pass through those two points. As must the second spline pass through point two $(6, 5)$ and three $(9, 5)$ therefore

$$5 = a_{1,0} + a_{1,1} \cdot 6 + a_{1,2} \cdot 6^2$$

$$5 = a_{1,0} + a_{1,1} \cdot 9 + a_{1,2} \cdot 9^2$$

And to make the slopes equal

$$\frac{d}{dx}(a_{0,0} + a_{0,1} \cdot x_i) = a_{0,1}$$

$$\frac{d}{dx}(a_{1,0} + a_{1,1} \cdot x + a_{1,2} \cdot x^2) = a_{11} + 2a_{1,2} \cdot x_i$$

We know they meet at $x = 6$ therefore $x = x_i = 6$ and we set them equal

$$a_{0,1} = a_{11} + 2a_{1,2} \cdot 6$$

Now we have the following equation 5

$$6 = a_{0,0} + a_{0,1} \cdot 5$$

$$5 = a_{0,0} + a_{0,1} \cdot 6$$

$$5 = a_{1,0} + a_{1,1} \cdot 6 + a_{1,2} \cdot 6^2$$

$$5 = a_{1,0} + a_{1,1} \cdot 9 + a_{1,2} \cdot 9^2$$

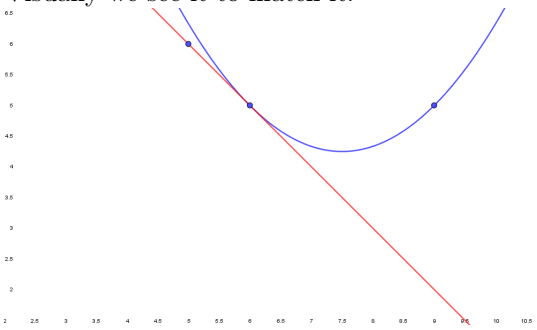
$$a_{0,1} = a_{11} + 2a_{1,2} \cdot 6$$

And with 5 unknowns. We solve them and get the two splines to

$$y_0 = 11 - x$$

$$y_1 = 13 - 5x + \frac{1}{3}x^2$$

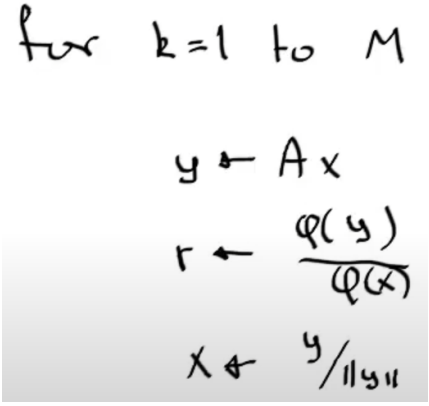
Visually we see it to match it:



5 Numerical Linear Algebra

5.1 Power method

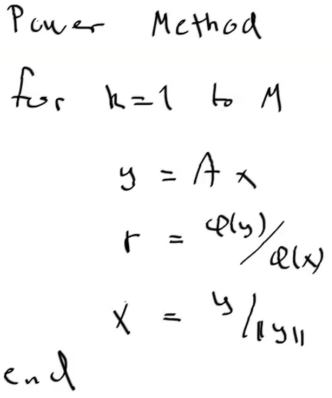
Used to find the largest absolute unique eigenvalue and its eigenvector ($Av = \lambda v$), A is quadratic and has an unique largest eigenvalue. Usually we make the characteristic polynomial ($det(A - \lambda I)$) and solve for $det(A - \lambda I) = 0$, but that is difficult for large matrices.
If given a matrix A and a startvector (can be any), then an algorithm can be made as.



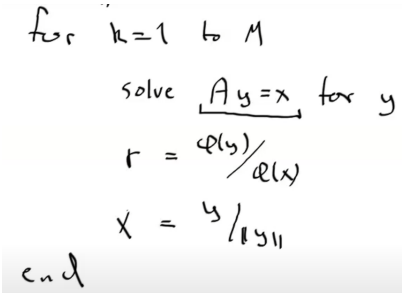
Where r is the approximation of the eigenvalue, x is the normalized eigenvector and φ is a linear function (e.g. $\varphi(x) = 2x + 1$). The algorithm can also include a absolute stop criteria such as $|r^{k+1} - r^k| < \epsilon_r$ or $|x^{k+1} - x^k| < \epsilon_x$ or relative stop criteria such as $|r^{k+1} - r^k| < \delta_r |r^k|$ or $|x^{k+1} - x^k| < \delta_x ||x^k||$

5.1.1 Inverse power method

Can be used to find the smallest eigenvalue and corresponding eigenvector.



But calculating the inverse of A can be very expensive we therefore change it to



5.2 Gershgorins theorem

Discs that contain all the eigenvalues of a quadratic matrix (but complex and real) in the complex plan. Defined as

$$D_i = \{2 \in \mathcal{C} || 2 - a_{ii}| \leq \sum_{j=1, j\neq 1}^{n} |a_{ij}|\}$$

Where the discs radius is $\sum_{j=1,j\neq 1}^{n} |a_{ij}|$ (the absolute sum of the row, not including the diagonal) and the center is at the diagonal point.
Example:

$$\begin{bmatrix} 1+i & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & \frac{1}{2} \end{bmatrix} \begin{matrix} r_1 = 2 \\ r_2 = 3 \\ r_3 = 3 \end{matrix}$$

This matrix has 3 discs, where D_1 has its center at $(1, i)$ and a radius of 2, D_2 has its center at $(0, 0)$ with a radius of 3 and D_3 has its center at $(\frac{1}{2}, 0)$ with a radius of 3.

5.2.1 Using the inverse

A matrix A and its inverse A^T have the same eigenvalues. Using that we can make discs for the inverse as well, and then just choose the smallest disc at every point to minimize the area the eigenvalues can be located at.

5.3 Gram-Schmidt process

The process takes a set of linear independent vectors (matrix A) and makes an orthonormal set (matrix B). The process takes for all vectors and subtracts them by how much they go in the same direction as the other vectors to make them orthogonal and them normalize them to make them orthonormal. In an orthonormal set of vectors all the length are 1 and the dot product between all vectors are 0.

$$B'_k = A_k - \sum_{i<k} < A_k, B_i > B_i$$

$$B_k = \frac{B'_k}{||B'_k||}$$

By using the Gram-Schmidt process we can factorize A by saving the B vectors and by what we correct them by to get a factorization $A = BT$, e.g. with 3 vectors

$$[A_1|A_2|A_3] = [B_1|B_2|B_3] \begin{bmatrix} ||B'_1|| & < A_2, B_1 > & < A_3, B_1 > \\ 0 & ||B'_2|| & < A_1, B_2 > \\ 0 & 0 & ||B'_3|| \end{bmatrix}$$

5.4 Least square solutions

If we have more equations than unknowns, then we can approximate a solution by using least squares method. x^* is the approximate solution to the equation $Ax = b$ where

$$A^T(Ax^* - b) = 0$$

This equation can be solved using a orthonormal factorization of A ($A = BT$, using e.g. Gram-schmidt):

$$Tx = (B^T B)^{-1} B^T b$$

5.5 QR factorization - Householder

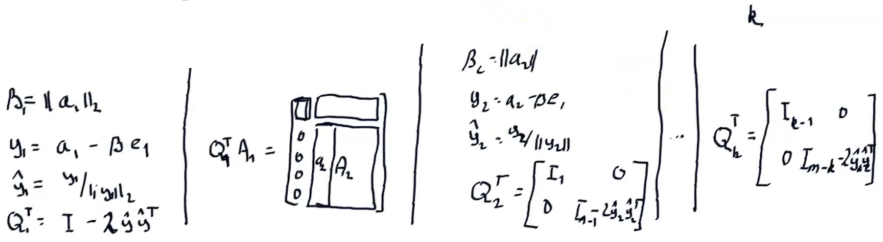
If we have a matrix A with M rows and N columns (MxN) and $M \geq N$ (more or equally as many rows as columns), then A can be factorized as $A = QR$ where Q is a MxM orthonormal matrix and R is a MxN upper triangular matrix. Householders calculates $Q^T A = R$ recursively by calculating

$$(Q_N^T \dots Q_2^T Q_1^T)A = Q^T A = R$$

With every iteration we first calculate the length (norm 2) of the k'th vector in A $B_k = ||a_k||_2$. Then we calculate $y_k = a_k - B_k e_1$ where e_1 is the elementary vector with as many elements as a_k (e.g. $[1, 0, 0]^T$ with 3 elements). Then normalize it $\hat{y_k} \frac{y_k}{||y_k||_2}$, and then Q_k^T is given by

$$I - 2\hat{y}\hat{y}^T$$

By every iteration the dimension of Q^T matrix is reduced by one row and one column, with the identity matrix to fill it out, so they all have the same dimension. Generic example with the first 2 iterations, and then the k'th iteration:



5.6 Singular value decomposition (SVD - A=PDQ form)

Given a general matrix A (MxN), with no specific restrictions. A can be factorized such that $A = PDQ$, then P is a MxM matrix, D is a diagonal MxN matrix with singular values and Q is a NxN matrix.
D contains as many singular value as $r = rank(A)$, given by the relation

$$A^T A u_i = \sigma_i^2 u_i$$

Which means the singular value is the square root of the eigenvalue for $A^T A$. The singular values are sorted, such that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_n \geq 0$. The first r singular value are positive ($\sigma_r > 0$).

The Q matrix contains eigenvectors u_i (as rows) that corresponds to the singular values in D. The $r + 1$ to N eigenvectors are the orthonormal basis for the nullspace of A.

The P matrix has the first r vectors given by the relation

$$v_i = \frac{1}{\sigma_i} A u_i$$

The last $r + 1$ to N vectors are selected such that P is an orthonormal basis for the space $\mathcal{R}^N$. The first r vectors are already an orthonormal set, so we just have to add vectors that match into the set.

5.6.1 Approach for SVD

First calculate the singualar values and eigenvectors

$$A^T A u_i = \sigma_i^2 u_i$$

Where $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_n \geq 0$, which we can use to contruct the D matrix

$$D = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_r \end{bmatrix}$$

Then we make the Q matrix by setting the eigenvectors found to the singular values as rows

$$Q = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

Then we can calculate the vectors v_i for the P matrix using

$$v_i = \frac{1}{\sigma_i} A u_i$$

for all $1 \leq i \leq r = rank(A)$ (same number of positive singular values).

$$P = \begin{bmatrix} v_1 & \dots v_r & | & v_{r+1} & \dots & v_M \end{bmatrix}$$

Then we have to select the last v_{r+1} to v_M vectors such that the matrix is a orthonormal basis.

5.6.2 VSU form

When constructing the full $A = PDQ$ we notice that the D matrix only contains values at the first rxr rows/columns, we therefore dont need the $r+1 \to r+n$ columns in P (the vectors we had to find ourself) and the $r+1 \to r+N$ rows in Q we don't need (the eigenvectors corresponding to the non-positive singular values). The form is called the VSU form

$$A = VSU$$

5.7 Minimal solutions

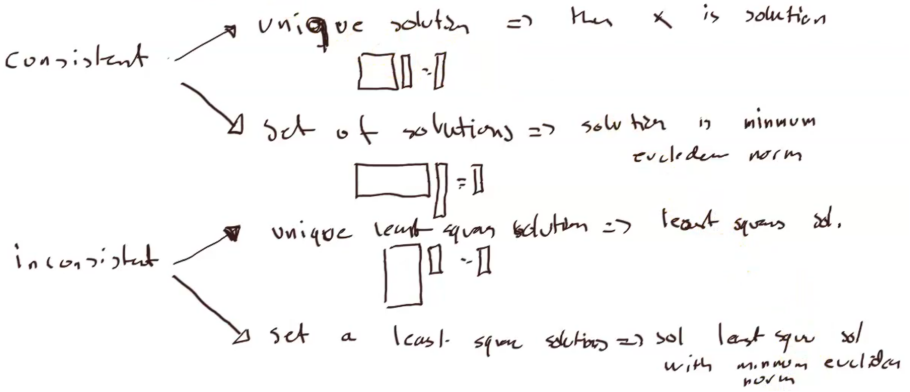
We can either have a consistent system of equations or an inconsistent system when solving $Ax = b$. We can then define what we view as the solution in all given cases.

5.7.1 Consistent

If we have as many unknowns as equations then we have a unique solution.
If we have too many unknowns compared to equations (more columns than rows), then we have a set of solutions, and we choose the one with the minimum euclidean norm.

5.7.2 Inconsistent

A system with too many equations compared to unknowns (Too many rows compared to columns) and one unique least squares solution.
Or we can have a set of least squares solutions, then we choose the solution with the minimum euclidean norm.



5.7.3 Pseudoinverse

A pseudoinverse A^+ will always give an unique minimal solution. Using the SVD, but with the inverse singular values in D instead, such that

$$D^+ = \begin{bmatrix} \sigma_1^{-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_r^{-1} \end{bmatrix}$$

Then $A^+ = Q^T D^+ P^T$ and the minimal solution will be given with $x = A^+ b$

6 Numerical Differentiation

6.1 Connection between analytical and numerical differentiation

Analytically the differentiation is given by

$$\frac{df}{dx} = lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Numerically we can define is as

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

Where the approximation is better the smaller h is. Using this technique the error follows a linear function ($O(n)$). So e.g. every time h gets 10 times smaller ($0.1 \to 0.01$) the error also gets 10 times smaller.

6.1.1 Deriving using Taylor series

We can define $f(x+h)$ using a Taylor series, and then prove the error term and defintion from before

$$f(x + h) = f(x) + \frac{h^1}{1!} f'(x) + \frac{h^2}{2!} f''(x) + \cdots + \frac{h^n}{n!} f^{(n)}(x)$$

Where the Taylor series is exact for $n \to \infty$. We are interested in $f'(x)$, so we isolate that to get

$$h \cdot f'(x) = f(x + h) - f(x) - \frac{h^2}{2} f''(x) - \cdots - \frac{h^n}{n!} f^{(n)}(x)$$

$$\iff$$

$$f'(x) = \frac{f(x + h) - f(x)}{h} - \frac{h}{2} f''(x) - \cdots - \frac{h^{n-1}}{n!} f^{(n)}(x)$$

From there we see that the error term is $-\frac{h}{2} f''(x) - \cdots - \frac{h^{n-1}}{n!} f^{(n)}(x)$, which follows a linear trend, since it can be rewritten as

$$\epsilon = -\frac{h}{2} f''(x) - \cdots - \frac{h^{n-1}}{n!} f^{(n)}(x) = \frac{h}{2}(f''(x) - \frac{h}{3} f'''(x) - \frac{h^2}{12} f''''(x) - \cdots - \frac{h^{n-1}}{n!} f^{(n)}(x))$$

Where the leading term is $\frac{h}{2}$ which is linear.

6.2 Deriving $O(n^2)$ using $f(x + \frac{h}{2})$

Using the same technique, but using $f(x + \frac{h}{2})$ instead of $f(x + h)$ we can have the error follow $O(n^2)$

$$\frac{df}{dx} \approx \frac{f(x + \frac{h}{2}) - f(x)}{h} = \frac{f(x + h) - f(x - h)}{2h}$$

6.2.1 Deriving using Taylor series

We know from before that

$$f(x + h) = f(x) + \frac{h^1}{1!}f'(x) + \frac{h^2}{2!}f''(x) + \cdots + \frac{h^n}{n!}f^{(n)}(x)$$

And now we also have $f(x - h)$ which is

$$f(x - h) = f(x) - \frac{h^1}{1!}f'(x) + \frac{h^2}{2!}f''(x) - \cdots + \frac{h^n}{n!}f^{(n)}(x)$$

All the odd terms are negative, since $(-h)^n$ gives a positive number when n is even.
If we now subtract them from each other

$$f(x + h) - f(x - h) = 2hf'(x) + 2\frac{h^3}{3!}f'''(x) + \cdots + 2\frac{h^n}{n!}f^{(n)}(x)$$

Since all the even terms cancelled out, we are left with all the odd terms. Now isolate $f'(x)$

$$f'(x) = \frac{f(x + h) - f(x - h)}{2h} - \frac{h^2}{6}f'''(x) + \cdots + \frac{h^{n-1}}{2n!}f^{(n)}(x)$$

We now see that the leading term in the error is $\frac{h^2}{6}$ and therefore the error is $O(n^2)$

$$\epsilon = -h^2(\frac{1}{6}f'''(x) + \cdots + \frac{h^{n-3}}{2n!}f^{(n)}(x))$$

6.3 Richardson Extrapolation

We know from before that $f'(x)$ can be given as

$$f'(x) = \frac{f(x + h) - f(x - h)}{2h} - h^2(a \cdot f'''(x) + b \cdot h^2 \cdot f^{(5)}(x) + c \cdot h^4 \cdot f^{(6)}(x) + \dots)$$

Redefining the error as $-t \cdot h^2$ such that

$$f'(x) = \frac{f(x + h) - f(x - h)}{2h} + t \cdot h^2 \tag{1}$$

We want to minimize the error term, so we make a new equation with $\frac{h}{2}$

$$f'(x) = \frac{f(x + \frac{h}{2}) - f(x - \frac{h}{2})}{h} + t \cdot \frac{h^2}{4} \tag{2}$$

Now we subtract (1) with 4 times equation (2) such that (1) − 4(2):

$$-3f'(x) = \frac{f(x+h) - f(x-h)}{2h} - 4\frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h} + 0$$

We see that the error term cancels out since $th^2 - 4t\frac{h^2}{4} = 0$.
Again isolating $f'(x)$

$$f'(x) = \frac{4}{3}\frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h} - \frac{f(x+h) - f(x-h)}{6h}$$

This now approximates the derivative very precisely.

6.4 Differentiation using interpolation

Using interpolation, like Lagrangian interpolation around the around the point we want the derivative of, we can then just differentiate the polynomial we get from the interpolation

7 Numerical Integration

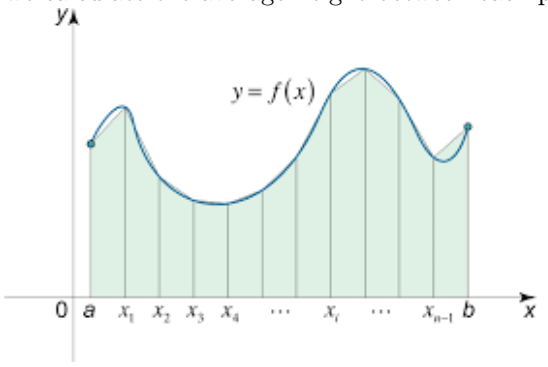
7.1 Average based method

One simple way to approximate the integral is just to calculate the average "height"(y-value) of the interval times the length of the interval $(b-a)$

$$\int_a^b f(x)dx \approx \frac{1}{n}(\sum_{i=0}^{n} f(x_i))(b-a)$$

7.1.1 Dividing segments using trapozoid rule

Instead of calculating just one average height, we can split the integral up, so we calculate the average height between each point pair.



The trapoziod rule is given by

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a) - f(b)}{2}$$

Which takes the average height and multiplies by the length between a and b. If we want the integral between a and b, where a nd b is not directly at a given x_i point. We calculate the straight line between the points that e.g. a is between

$$y_0 = a_0 + a_1 \cdot x_0$$

$$y_1 = a_0 +_{a\ 1} \cdot x_1$$

And then solve for a_0 and a_1, so we can calculate the y_i at a (lets call it y_a and at x_a) $y_a = a_0 + a_1 \cdot x_a$

7.2 Integrating using interpolation

As with differentiation, we can just interpolate (e.g. using Lagrangian interpolation) and then integrate that polynomial. Notice that interpolating gets very un-precise outside the interval of given data points

7.3 Method of undetermined coefficients

We know from the trapezoidal rule, that the integral between 2 points can be approximated by taking the integral of a straight line between them. Now we define it to be equal to some linear combination of $f(a)$ and $f(b)$ as

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2} = A_0 \cdot f(a) + A_1 \cdot f(b)$$

And the straight line between a and b can be defined at $c_0 + c_1 x$. And the integral of that line is

$$\int_a^b c_0 + c_1 x dx = \left[c_0 x + \frac{c_1 x^2}{2}\right]_a^b = (c_0 b + \frac{c_1 b^2}{2}) - (c_0 a + \frac{c_1 a^2}{2}) = c_0(b-a) + c_1(\frac{b^2-a^2}{2})$$

We can now define $f(a)$ and $f(b)$ using the straight line just defined

$$A_0 \cdot f(a) + A_1 \cdot f(b) = A_0(c_0 + c_1 \cdot a) + A_1(c_0 + c_1 \cdot b)$$

We rearrange to have it in terms of c_0 and c_1

$$A_0(c_0 + c_1 \cdot a) + A_1(c_0 + c_1 \cdot b) = c_0(A_0 + A_1) + c_1(A_0 \cdot a + A_1 \cdot b)$$

We can now compare that to the true integral we calculated for the straight line, which was $c_0(b-a) + c_1(\frac{b^2-a^2}{2})$, and we can therefore make the 2 equations

$$b - a = A_0 + A_1$$

$$\frac{b^2-a^2}{2} = A_0 \cdot a + A_1 \cdot b$$

We solve for A_0 and A_1 and get

$$A_0 = A_1 = \frac{b-a}{2}$$

Which are the same coefficients as the trapezoid rule.
The same logic can be used to approximate using a higher degree polynomial with more points instead of a straight line with 2 points. We just get more A_n coefficients and a larger true integral to compare.

7.4 Simpsons rule

Using 3 points a, $\frac{a+b}{2}$ and b, and now we define a polynomial of 2nd degree to go through a and b:

$$f_{ab}(x) = a_0 + a_1 x + a_2 x^2$$

And the integral of that function to be

$$\int_a^b f(x)dx = \left[a_0 x + \frac{a_1 x^2}{2} + \frac{a_2 x^3}{3}\right]_a^b = a_0(b-a) + \frac{a_1}{2}(b^2 - a^2) + \frac{a_2}{3}(b^3 - a^3)$$

And we now define $f(a)$, $f(\frac{a+b}{2})$ and $f(b)$ in terms of that polynomial to make 3 equations

$$f(a) = a_0 + a_1 \cdot a + a_2 \cdot a^2$$
$$f(\frac{a+b}{2}) = a_0 + a_1(\frac{a+b}{2}) + a_2(\frac{a+b}{2})^2$$
$$f(b) = a_0 + a_1 \cdot b + a_2 \cdot b^2$$

We now have 4 equations (with the true integral) and 4 unknowns (with the value of the integral). We solve the equations, and the integral is then equal to

$$\int_a^b f(x)dx = \frac{b-a}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$$

$$\iff \int_a^b f(x)dx = \frac{h}{3}(f(a) + 4f(a+h) + f(a+2h))$$

Where h is the distance from $a \to \frac{a+b}{2}$, and there $a + 2h = b$

7.5 Gaussian Quadrature

In the Gaussian quadrature we are no longer restricted to what a and b are, an we therefore define the integral to be

$$\int_{\hat{x}}^{\bar{x}} f(x)dx = A_0 f(\hat{x}) + A_1 f(\bar{x})$$

Where we have 4 unknowns. We can then rewrite to simplify

$$\int_{-1}^{1} f(t)dt = \omega_0 f(t_1) + \omega_1 f(t_2)$$

Where the 4 unkowns are ω_0, ω_1, t_1 and t_2. We then make 4 equations, where $f(t) = 1, t, t^2, t^3$ as a "training set"

$$f(t) = 1 \rightarrow \int_{-1}^{1} f(t)dt = 2 = \omega_0 \cdot 1 + \omega_1 \cdot 1$$

$$f(t) = t \rightarrow \int_{-1}^{1} f(t)dt = 0 = \omega_0 \cdot t_1 + \omega_1 \cdot t_2$$

$$f(t) = t^2 \rightarrow \int_{-1}^{1} f(t)dt = \frac{2}{3} = \omega_0 \cdot t_1^2 + \omega_1 \cdot t_2^2$$

$$f(t) = t^3 \rightarrow \int_{-1}^{1} f(t)dt = 0 = \omega_0 \cdot t_1^3 + \omega_1 \cdot t_2^3$$

Solving the equations we get $\omega_0 = 1$, $\omega_1 = 1$, $t_1 = \pm\sqrt{\frac{1}{3}}$ and $t_2 = -t_1$.
Going back to the original interval a to b, we can do the following

$$x = \frac{(b-a)t + b + a}{2}$$

Since ten when $t = -1 \rightarrow x = a$ and when $t = 1 \rightarrow x = b$. And we set $dx = \frac{(b-a)}{2}dt$ we combine them to make the final Gaussian Quadrature equation

$$\int_a^b f(x)fx = \int_{-1}^{1} f(\frac{(b-a)t + b + a}{2})(\frac{b-a}{2})dt$$

We can easily extend the Gaussian Quadrature to N components using the same logic, we just need 2N equations made from a "training set"as we did above using polynomials from 1 to 2N degree.

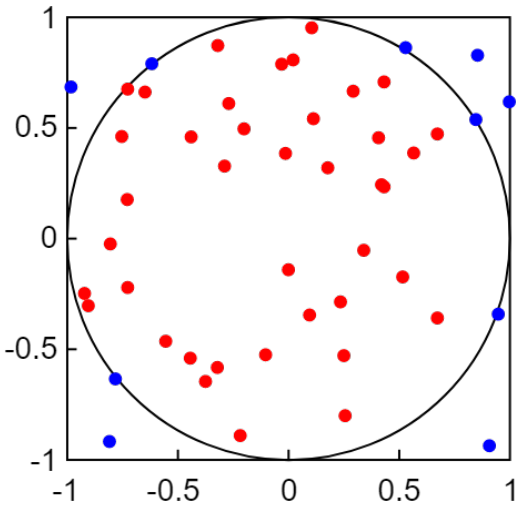
8 Monte-Carlo Simulation

8.1

using a randomized data generating process (DGP) we can simulate many cases of a problem to estimate what is most likely to happen

8.2 Numerical integration

If we plot a figure in a space where we know the size of the space, we can then "drop balls"at random places within the space. The integral of the figure must then be $I = size_{space} \cdot \frac{\text{Balls within the figure}}{\text{Total dropped balls}}$. The more points we drop, the more accurate does the integral get.



The general process can be defined as

Monte Carlo integration - uniform sampling Consider the integral

$$I = \int_0^1 f(x)$$

1. Choose N points x_i at random with uniform probability within the integration interval [0,1].
2. Determine the mean value

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f_i$$

and the variance

$$\sigma_f^2 = \left[\frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} f_i\right)^2\right]$$

3. Approximate the value of the integral as

$$I = I_N \pm \frac{\sigma_f}{\sqrt{N}}$$

Such that the integral is approximated to I_N with an error term of $\pm\frac{\sigma_f}{\sqrt{N}}$.