Notions of Numerical Analysis

Antonio Mucherino

University of Rennes 1

www.antoniomucherino.it

last update: August 2013

Numerical Analysis

Linear systems

Aircraft and wind

Suppose an aircraft flies from Paris to Rio and then it comes back. Suppose the wind is constant during the whole travel and it is able to influence the speed of the aircraft.

- Paris - Rio, time $t_1 = 5.1$ hours, aircraft flying *against* wind
- Rio - Paris, time $t_2 = 4.7$ hours, aircraft flying *with* wind
- distance: 5700 miles

How can we find the average speed of the aircraft and the average speed of the wind?

How to solve this problem?

Let x be the average speed of the aircraft,
and let y be the average speed of the wind:

- the actual aircraft speed is $x - y$ when it flies *against* the wind
- the actual aircraft speed is $x + y$ when it flies *with* the wind
- the distance d for each travel can be computed as the product between the time (t_1 or t_2) and the actual speed

We can define the following system of equations:

$$\begin{cases} t_1(x - y) = d \\ t_2(x + y) = d \end{cases}$$

This is a linear system:

- t_1, t_2 and d are *parameters* (already known)
- x and y are *variables*

Linear systems

General form of a linear system with 2 equations:

$$\begin{cases} a_{11}x + a_{12}y = b_1 \\ a_{21}x + a_{22}y = b_2 \end{cases}$$

And, in matrix form:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

The coefficient matrix $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ is able to provide information

- about the existence of solutions
- about the number of solutions

(out of the scope of this course).

The solution for our example

For the system

$$\begin{cases} t_1 x - t_1 y = d \\ t_2 x + t_2 y = d \end{cases}$$

we can find a solution (x, y) analytically:

$$\begin{cases} x = \dfrac{d}{t_1} + y \\[2ex] y = \dfrac{d\left(1 - \dfrac{t_2}{t_1}\right)}{2t_2} \end{cases}$$

The solution for our example

So, in our example:

- $t_1 = 5.1$
- $t_2 = 4.7$
- $d = 5700$

and therefore:

$$\begin{cases} x = 1165.2 \ miles/hours \\ y = 47.6 \ miles/hours \end{cases}$$

Can we program a computer to make this work for us?

Note that, in this simple example, we did not consider the health rotation.

Back substitution

Suppose the coefficient matrix of our linear system is an upper triangular matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

In this situation, we can compute:

$$x_4 = \frac{b_4}{a_{44}}$$

Back substitution

Suppose the coefficient matrix of our linear system is an upper triangular matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

In this situation, we can compute:

$$x_3 = \frac{b_3 - a_{34}x_4}{a_{33}}$$

Back substitution

Suppose the coefficient matrix of our linear system is an upper triangular matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

In this situation, we can compute:

$$x_2 = \frac{b_2 - a_{23}x_3 - a_{24}x_4}{a_{22}}$$

Back substitution

Suppose the coefficient matrix of our linear system is an upper triangular matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

In this situation, we can compute:

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4}{a_{11}}$$

C function for back substitution

Numerical Analysis

A. Mucherino

Linear systems
Finding the speed of the wind
A simple algorithm
Roots of functions
Functions and roots
The bisection method
Interpolation
A reaction equilibrium constant
Interpolation ...
... and regression?
Numerical integration
The area of a circle
Trapezoidal rule
Optimization
Definition and commom methods

```
void back(int n,double **a,double *x)
{
   // n is the system dimension
   // a is the coefficient matrix
   //       (must be upper triangular)
   // x is
   //       the vector of known terms (input)
   //       the solution (output)

   int i,j;

   for (i = n - 1; i >= 0; i--)
   {
      for (j = i+1; j < n; j++)
      {
         x[i] = x[i] - a[i][j]*x[j];
      };
      x[i] = x[i]/a[i][i];
   };
};
```

Gaussian elimination

How to solve linear systems whose coefficient matrix is not in triangular form?

Gaussian elimination method: *transform the system in an equivalent system whose coefficient matrix is in triangular form*.

Example:

$$\left\{\begin{array}{rrrcr} 2x & +y & -z & = & 8 \\ -3x & -y & +2z & = & -11 \\ -2x & +y & +2z & = & -3 \end{array}\right.$$

$$\Longrightarrow \left\{\begin{array}{rrrcr} 2x & +y & -z & = & 8 \\ & \frac{1}{2}y & +\frac{1}{2}z & = & 1 \\ & & -z & = & 1 \end{array}\right.$$

LAPACK

LAPACK – Linear Algebra PACKage

free library for linear algebra
(including linear systems)

$$\begin{bmatrix} L & A & P & A & C & K \\ L & -A & P & -A & C & -K \\ L & A & P & A & -C & -K \\ L & -A & P & -A & -C & K \\ L & A & -P & -A & C & K \\ L & -A & -P & A & C & -K \end{bmatrix}$$

- it's a freely-available software package (library + sources)
- originally developed in Fortran, there are versions for C and C++
- it's based on another library called BLAS, which contains functions for efficient matrix manipulations (sums, products, . . .)

Some references

- Wikipedia page about linear systems,
  `http://en.wikipedia.org/wiki/System_of_linear_equations`
- Online solution of linear systems,
  `http://karlscalculus.org/cgi-bin/linear.pl`
- LAPACK,
  `http://www.netlib.org/lapack/`
- BLAS,
  `http://netlib.org/blas/`

Numerical Analysis

Roots of functions

Stationary points

In many applications, stationary points of functions are of particular interest.

They might provide:

- the minimum and maximum points of functions
- the equilibrium point of a dynamic system (may be stable or not)
- ...

In order to find a stationary point, the derivative of a function must be computed, and roots of such a derivative must be identified:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = 0$$

Functions and roots

Given a function $f : [a, b] \to Y$, how to find its roots (zeros)?

$$f(x) = 0$$

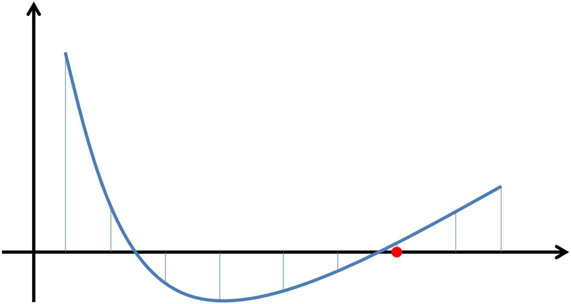
Examples:

$$ax = b \implies x = \frac{b}{a}$$

$$ax^2 + bx + c = 0 \implies \begin{cases} x_1 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a} \\ x_2 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a} \end{cases}$$

A simple method

Simplest method for finding roots:
Extract a predefined number of points from the function domain $[a, b]$ *and evaluate the function in all these points. One of these points can be a root (or be close to a root)*.



This method is not able to provide a good approximation of roots of functions having a more complex shape.

Bisection: basic idea

The **bisection method** is an iterative method which defines a sequence of intervals $\{a_k, b_k\}_{k=1,2,\dots,itmax}$ converging to one function root.

At the beginning, the whole function domain is considered:

$$[a_0, b_0] = [a, b]$$

At each iteration, the following two steps are performed:

- the average point of the current interval is computed:

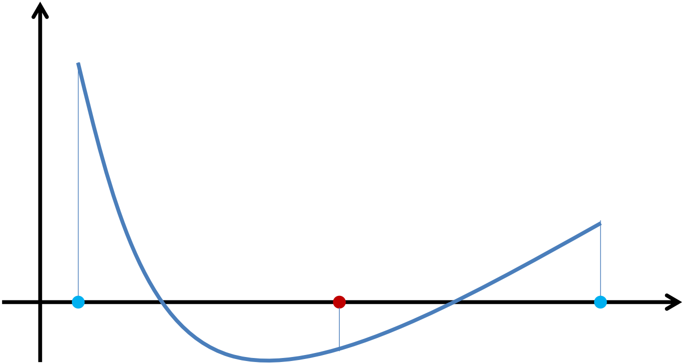
$$x_k = a_k + \frac{b_k - a_k}{2}$$

- the new interval is then defined as:

$$\begin{cases} [a_{k+1}, b_{k+1}] = [a_k, x_k] & \text{if} \quad f(a_k)f(x_k) \leq 0 \\ [a_{k+1}, b_{k+1}] = [x_k, b_k] & \text{otherwise} \end{cases}$$

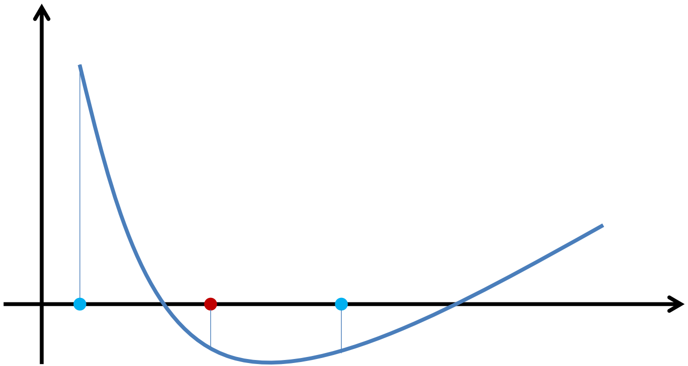
Bisection: basic idea

In the bisection method, intervals $[a_k, b_k]$ are reduced in size at each iteration, and they are supposed to converge to a function root.



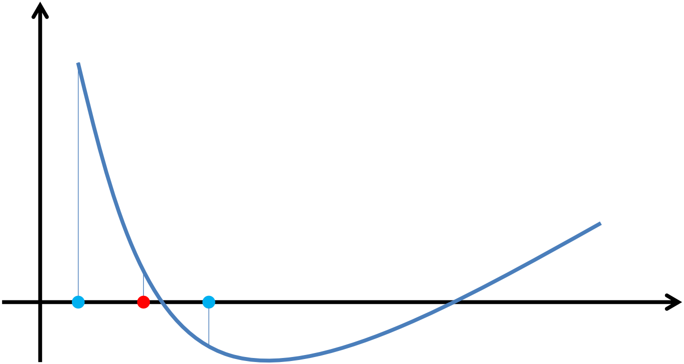
Bisection: basic idea

In the bisection method, intervals $[a_k, b_k]$ are reduced in size at each iteration, and they are supposed to converge to a function root.



Bisection: basic idea

In the bisection method, intervals $[a_k, b_k]$ are reduced in size at each iteration, and they are supposed to converge to a function root.



Applicability

The bisection method can be applied to functions $f : [a, b] \to Y$:

- if all points in $[a, b]$ can be evaluated
- if f is a continuous function

Moreover, if at least one of the intervals $[a_k, b_k]$ is such that

$$f(a_k)f(b_k) < 0$$

then, the method converges toward one of the roots contained in the interval.

The method can be stopped when

$$|a_k - b_k| < \varepsilon \qquad \text{or} \qquad |f(a_k) - f(b_k)| < \varepsilon$$

where ε is a small real number (tolerance).

C function for the bisection algorithm

```
double bisection(double a,double b,double (*f)(double),double eps,int itmax)
{
   // [a,b], function domain
   // double (*f)(double), pointer to a function
   // eps, tolerance
   // itmax, maximum number of iterations

   int it;
   double ca,cb,cx,fa,fb,fx;

   ca = a; cb = b; fa = f(a); fb = f(b);
   cx = (ca+cb)/2.0; fx = f(cx);

   it = 0;
   while (it <= itmax && fabs(fx) > eps && fabs(cb-ca) > eps && fabs(fb-fa) > eps)
   {
      it = it + 1;
      if (fa*fx < 0)
      {
         cb = cx; fb = fx;
      }
      else
      {
         ca = cx; fa = fx;
      };
      cx = (ca+cb)/2.0; fx = f(cx);
   };

   return cx;
};
```

Pointers to functions

A **pointer to a function** can make reference to any function of a predefined type:

```
double (*f)(double)
```

In the main function:

```
double xcube(double x);
double polynomial(double x);
double bisection(double a,double b,double (*f)(double),double eps,int itmax);

main()
{
   int i,j;
   double a,b,root;
   ...
   double *f(double);
   ...

   f = xcube; root = bisection(a,b,f,0.001,100);
   ...

   f = polynomial; root = bisection(a,b,f,0.001,100);
   ...

};
```

Other methods for finding roots

- Newton's method
  it is based on the computation of the tangent to the function in the current root approximation
- Secant method
  similar to the Newton's method, but the tangent is replaced by a secant (the function does not have to be differentiable in the whole domain)
- Lehmer-Schur method
  extension of the bisection method
- Brent's method
  combination of different methods, including the bisection method, with the aim of speeding up the search

Numerical Analysis

Polynomial interpolation

A reaction equilibrium constant

The equilibrium constant for ammonia reacting in hydrogen and nitrogen gases depends upon the hydrogen-nitrogen mole ratio, the pressure, and the temperature.
For a 3-to-1 hydrogen-nitrogen mole ratio, the equilibrium constant K_p for a range of pressures and temperatures is given by:

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.004409	0.004873	0.005408	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

Encyclopedia of Chemical Technology, vol. 2, 2^{nd} edition, New York, Wiley, 1963.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.004409	0.004873	0.005408	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.004409	0.004873	0.005408	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.005311	0.004873	0.005408	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.005311	0.004873	0.005408	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.005311	0.004873	0.006618	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

A reaction equilibrium constant

Suppose that values for K_p related to 500°C and 300 atm are not available, and that, for same reason, we cannot perform any experiment to find them.

	100 atm	200 atm	300 atm	400 atm	500 atm
400°C	0.014145	0.015897	0.018060	0.020742	0.024065
450°C	0.007222	0.008023	0.008985	0.010134	0.011492
500°C	0.004013	0.005311	**0.005965**	0.006618	0.006013
550°C	0.002389	0.002598	0.002836	0.003102	0.003392
600°C	0.001506	0.001622	0.001751	0.001890	0.002036

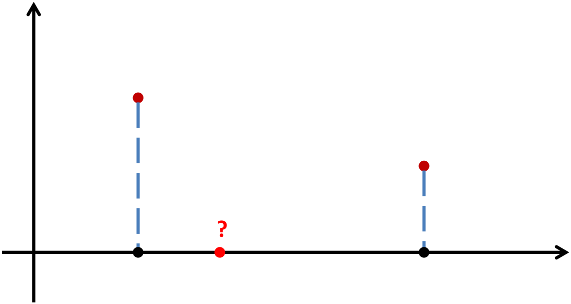
How can we find the needed values for the constant K_p?

Easiest solution: linear interpolation.

Linear interpolation

Let $f : [a,b] \to Y$ be a function such that

- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a,b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a,b]$ and $x_2 > x_1$
- $f(x)$ is not known for any $x \in (x_1, x_2)$

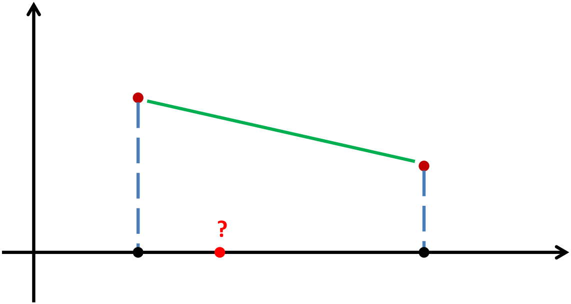


Linear interpolation: assign to the interval (x_1, x_2) of $f(x)$ the equation of the *line* between x_1 and x_2.

Linear interpolation

Let $f : [a, b] \to Y$ be a function such that

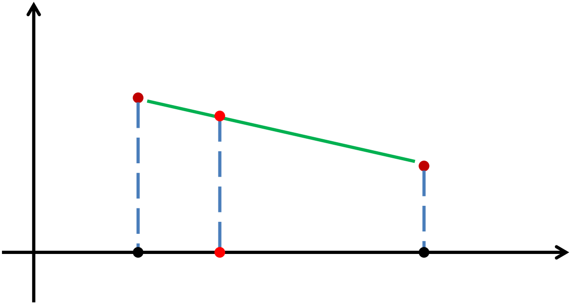
- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a, b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a, b]$ and $x_2 > x_1$
- $f(x)$ is not known for any $x \in (x_1, x_2)$



Linear interpolation: assign to the interval (x_1, x_2) of $f(x)$ the equation of the *line* between x_1 and x_2.

Let $f : [a,b] \to Y$ be a function such that

- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a,b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a,b]$ and $x_2 > x_1$
- $f(x)$ is not known for any $x \in (x_1, x_2)$

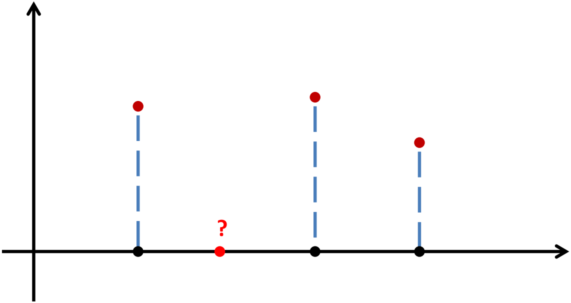


Linear interpolation: assign to the interval (x_1, x_2) of $f(x)$ the equation of the *line* between x_1 and x_2.

Quadratic interpolation

Let $f : [a,b] \rightarrow Y$ be a function such that

- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a,b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a,b]$ and $x_2 > x_1$
- the pair $(x_3, f(x_3))$ is known, with $x_3 \in [a,b]$ and $x_3 > x_2 > x_1$
- $f(x)$ is not known for any $x \in [a,b] \setminus \{x_1, x_2, x_3\}$

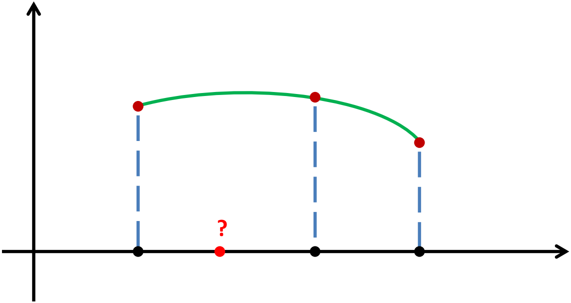


Quadratic interpolation: assign to the interval (x_1, x_3) of $f(x)$ the equation of the *parabola* passing through x_1, x_2 and x_3.

Quadratic interpolation

Let $f : [a, b] \rightarrow Y$ be a function such that

- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a, b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a, b]$ and $x_2 > x_1$
- the pair $(x_3, f(x_3))$ is known, with $x_3 \in [a, b]$ and $x_3 > x_2 > x_1$
- $f(x)$ is not known for any $x \in [a, b] \setminus \{x_1, x_2, x_3\}$

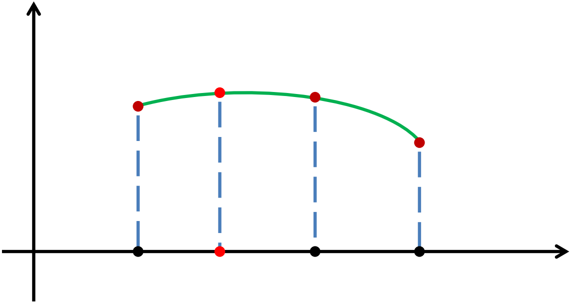


Quadratic interpolation: assign to the interval (x_1, x_3) of $f(x)$ the equation of the *parabola* passing through x_1, x_2 and x_3.

Quadratic interpolation

Let $f : [a, b] \to Y$ be a function such that

- the pair $(x_1, f(x_1))$ is known, with $x_1 \in [a, b]$
- the pair $(x_2, f(x_2))$ is known, with $x_2 \in [a, b]$ and $x_2 > x_1$
- the pair $(x_3, f(x_3))$ is known, with $x_3 \in [a, b]$ and $x_3 > x_2 > x_1$
- $f(x)$ is not known for any $x \in [a, b] \setminus \{x_1, x_2, x_3\}$



Quadratic interpolation: assign to the interval (x_1, x_3) of $f(x)$ the equation of the *parabola* passing through x_1, x_2 and x_3.

Lagrangian interpolation

Let $f : [a, b] \rightarrow Y$ be a function such that

- the pairs $(x_i, f(x_i))$ are known, with

$$x_i \in \{x_1, x_2, \ldots, x_n\} \subset [a, b]$$

- $f(x)$ is not known for any $x \in [a, b] \setminus \{x_1, x_2, \ldots, x_n\}$

Lagrangian interpolation:

assign to the interval (x_1, x_n) of $f(x)$ the equation of the polynomial of degree $n - 1$ passing through the n points x_1, x_2, $\ldots x_n$.

Lagrangian interpolation

A general polynomial of degree $n-1$ can be written as:

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2x^2 + a_1x + a_0$$

For the polynomial to pass through the n points (x_i, y_i), we need to solve the following system of linear equations:

$$\begin{cases} y_1 = a_{n-1}x_1^{n-1} + a_{n-2}x_1^{n-2} + \cdots + a_2x_1^2 + a_1x_1 + a_0 \\ y_2 = a_{n-1}x_2^{n-1} + a_{n-2}x_2^{n-2} + \cdots + a_2x_2^2 + a_1x_2 + a_0 \\ y_3 = a_{n-1}x_3^{n-1} + a_{n-2}x_3^{n-2} + \cdots + a_2x_3^2 + a_1x_3 + a_0 \\ \ldots \\ y_n = a_{n-1}x_n^{n-1} + a_{n-2}x_n^{n-2} + \cdots + a_2x_n^2 + a_1x_n + a_0 \end{cases}$$

Lagrangian interpolation

It can be proved that

- the system of linear equations has only one solution:

$$\{a_0, a_1, a_2, \ldots, a_{n-1}\}$$

- the polynomial of degree $n-1$ and having as coefficients the found a_i's is such that:

$$y_i = f(x_i), \qquad \forall i = 1, 2, \ldots, n$$

The general formula for Lagrangian interpolation is:

$$f(x) = \sum_{i=1}^{n} y_i \prod_{j=1, j\neq i}^{n} \left(\frac{x - x_j}{x_i - x_j} \right)$$

C function for interpolation

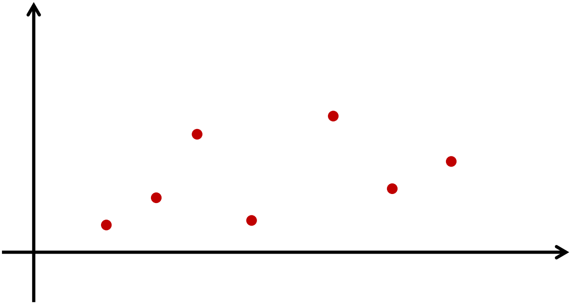
```
double interpol(int n,double *x,double *y,double p)
{
   // n, number of available (x,y)
   // x, vector containing all x's
   // y, vector containing all y's
   // p, point where to evaluate lagrangian polynomial

   int i,j;
   double sum,prod;

   sum = 0.0;
   for (i=0; i<n; i++)
   {
      prod = 1.0;
      for (j=0; j<n; j++)
      {
         if (j!=i)
         {
               prod = prod * ((p-x[j])/(x[i]-x[j]));
         };
      };
      sum = sum + y[i]*prod;
   };

   return sum;
};
```

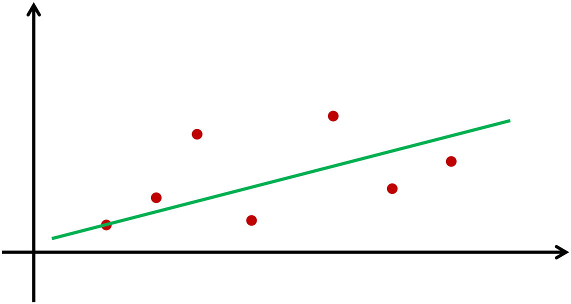
Suppose that the form of $f(x)$ is known a priori.



If $f(x)$ is linear, would the lagrangian polynomial be a good model?

Regression

Suppose that the form of $f(x)$ is known a priori.



If $f(x)$ is linear, would the lagrangian polynomial be a good model? No!

Solution: **regression** models.

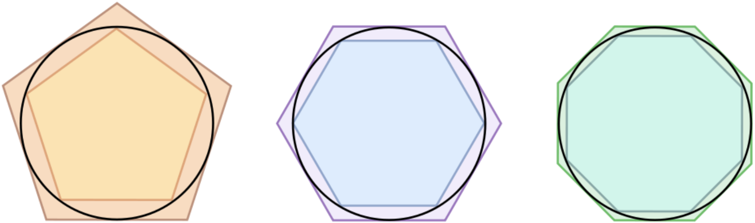
Numerical Analysis

Numerical integration

Archimedes

Archimedes (287BC–212BC) was a Greek mathematician, physicist, engineer, inventor, and astronomer. He is generally considered to be the greatest mathematician of antiquity.

Archimedes was able to approximate the area of a circle with polygons converging to the shape of the circle.

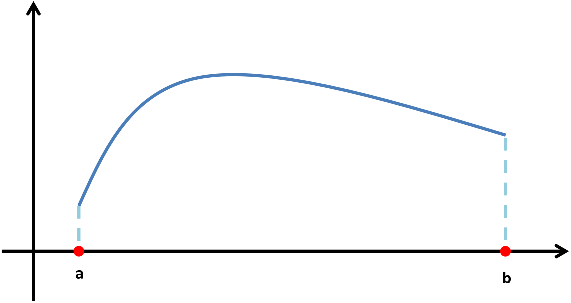


He was able to approximate the value of π to 3.1416.

http://en.wikipedia.org/wiki/Archimedes

Numerical integration

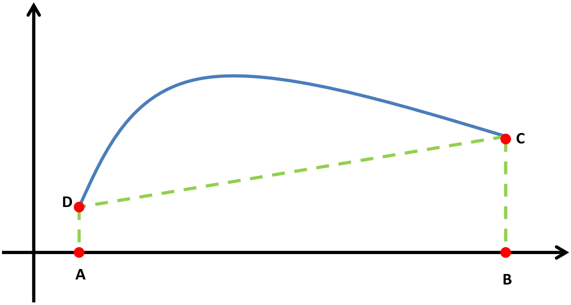
Solving the definite integral $\int_a^b f(x)dx$ equals to finding the area under the curve $y = f(x)$ and between $x = a$ and $x = b$.



We suppose:

- a and b are not $\pm\infty$,
- there are no points $\bar{x} \in [a, b]$ such that $f(\bar{x}) = \pm\infty$.

Idea: approximate the area defined by $f(x)$ with the area of the trapezoid ABCD:

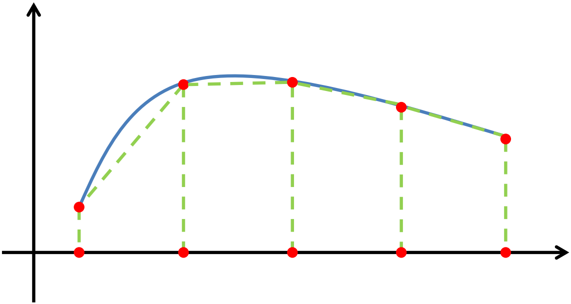


Area of trapezoid: $\frac{1}{2}h\,(f(a)+f(b))$.

The area between $y = f(x)$ and the segment DC corresponds to the error introduced with this approximation.

Trapezoidal rule

Trapezoidal rule: divide $[a, b]$ in n equal parts of length h and approximate each subinterval $[x_i, x_{i+1}]$ with the area of the corresponding trapezoid:



Trapezoidal formula:

$$\frac{1}{2} h \sum_{i=1}^{n} \left(f(x_i) + f(x_{i+1}) \right).$$

Trapezoidal rule: the C function


```
double trapez(double a,double b,int n, double (*f)(double))
{
   // interval [a,b] (input)
   // n, number of subintervals (input)
   // f, pointer to function (input)
   // returning value:  approx.  of the area defined by f(x) in [a,b]

   int i;
   double h,area;
   double ca,cb,fa,fb;

   h = (b - a)/n;
   ca = a; cb = ca + h;
   fa = f(ca); fb = f(cb);
   area = fa + fb;

   for (i = 1; i < n; i++)
   {
      ca = cb; fa = fb;
      cb = cb + h; fb = f(cb);
      area = area + fa + fb;
   };

   return h*area/2.0;
};
```

Other algorithms

- **Simpson rule**: instead of using trapezoids to approximate the areas, parabolas interpolating 3 consecutive points are employed. This simple modification increases the accuracy of the method.
- **Gaussian quadrature**: subintervals of $[a, b]$ do not have the same length but they are chosen so that the global accuracy increases.

W.S. Dorn, D.D. Mc Cracken, *Numerical Methods with Fortran IV Case Studies*, John Wiley & Sons, Inc., 1972.

Numerical Analysis

Optimization

Optimization problems

General form on an optimization problem:

$$\min_{x \in A} f(x)$$

subject to a set of constraints:

$$\begin{cases} \forall x \in B & g(x) = 0 \\ \forall x \in C & h(x) \leq 0 \end{cases}$$

where

- $f(x)$ is the objective function
- $g(x)$ represents the equality constraints
- $h(x)$ represents the inequality constraints

Some methods for optimization

Deterministic methods (may require some assumptions to be satisfied)

- Simplex method
- Branch & Bound
- Branch & Prune
- . . .

Heuristic methods (no guarantees for optimality)

- Simulated Annealing
- Genetic Algorithms
- Tabu Search
- Variable Neighbourhood Search
- . . .