Lecture 7

THE METHOD OF MOMENTS (MoM)

CHOICE OF THE BASIS FUNCTIONS / 1

Ideally, the basis functions should represent a complete orthogonal base in the definition domain of the unknown function f.

$$< f_n, f_m >= \int_S f_n \cdot f_m^* \, dS = \delta_{mn} \qquad \forall\, m, n$$

However, the determination of an orthogonal base is not an easy task, especially in the case of arbitrary domains.

From a practical point of view, the MoM results efficient when the basis functions exhibit a high degree of linear independency.

Choice of the Basis Functions / 2

The choice of the basis functions is based on various factors:

- Accuracy of the solution
- Easy computation of **A** and **B** matrix entries
- Size of matrix **A** (number of needed functions)
- Condition number of matrix **A**

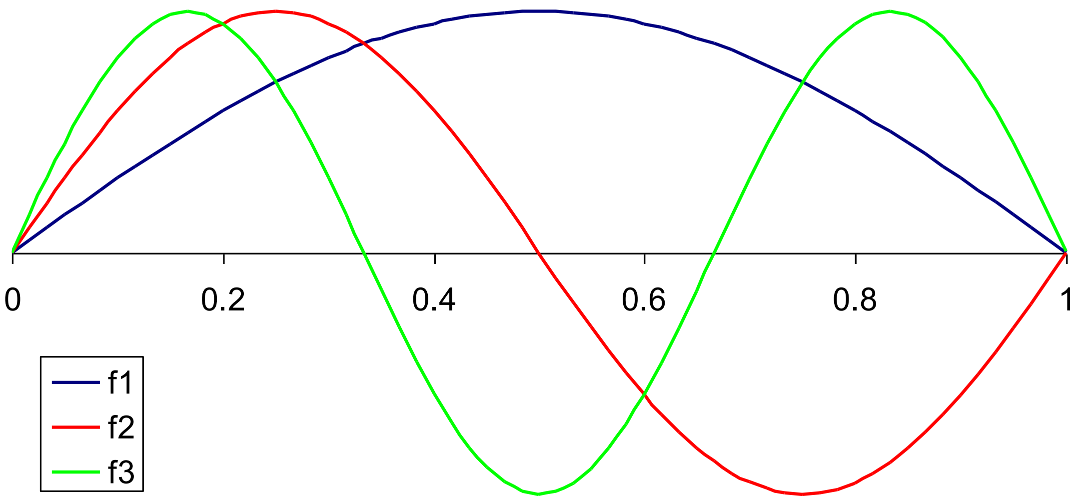
Basis functions can be subdivided in two big classes:

- entire-domain basis functions (defined in the entire domain of functions *f*)
- sub-domain basis functions (defined in the small portion of the domain)

EXAMPLES OF BASIS FUNCTIONS / 1

An example of entire-domain basis functions (in 1D case) is represented by sinusoidal functions.

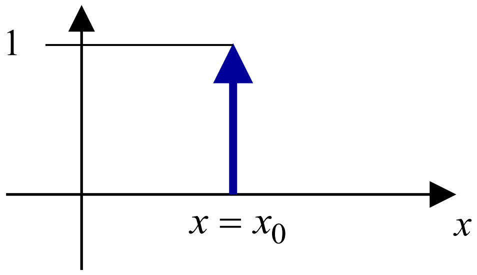
$$f_n = \begin{cases} \sin(nx) \\ \cos(nx) \\ \exp(jnx) \end{cases}$$



Examples of Basis Functions / 2

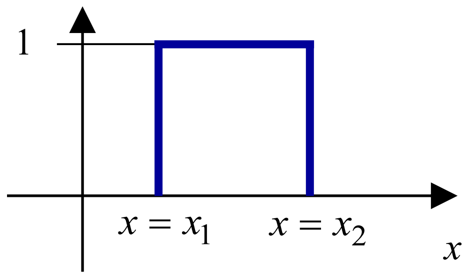
The sub-domain basis functions exist only on one of the N non-overlapping segments into which the domain is divided. Examples of sub-domain basis functions (in 1D case) are:

- delta functions



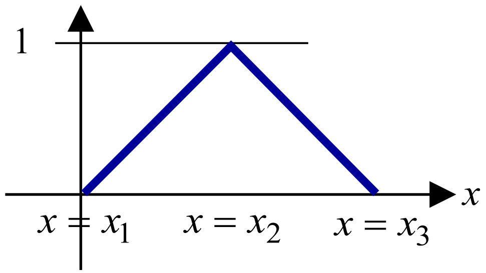
$$\mathrm{D}(x) = \delta(x - x_0)$$

- piecewise constant functions



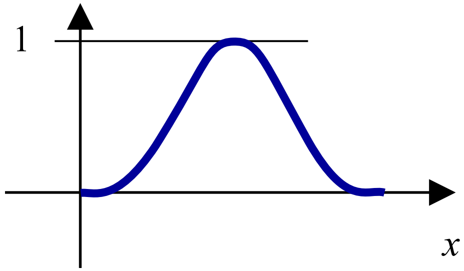
$$\Pi(x, x_1, x_2) = \begin{cases} 1 & x_1 \le x \le x_2 \\ 0 & \text{otherwise} \end{cases}$$

- triangular functions



$$\Lambda(x, x_1, x_2, x_3) = \begin{cases} \dfrac{x - x_1}{x_2 - x_1} & x_1 \le x \le x_2 \\ \dfrac{x_3 - x}{x_3 - x_2} & x_2 \le x \le x_3 \\ 0 & \text{otherwise} \end{cases}$$

- spline functions



NUMERICAL INTEGRATION / 1

The integrals are typically computed in an approximate way, by using techniques for numerical integration (also called numerical quadrature).

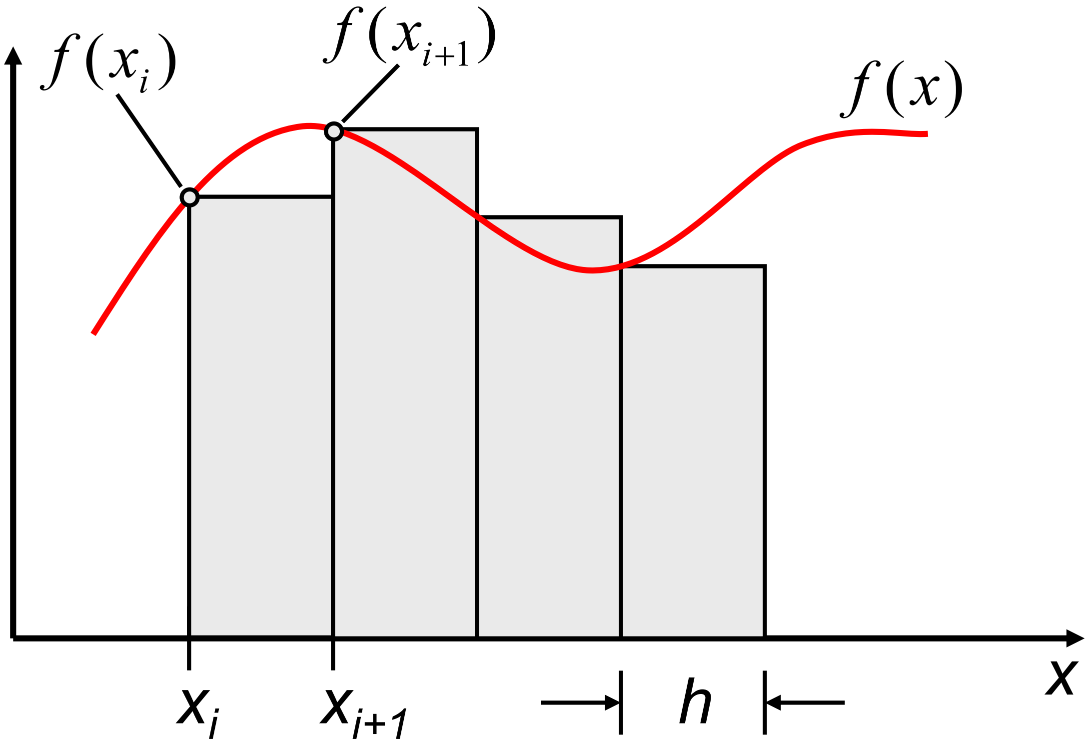
$$\int_a^b f(x)\,dx \cong \sum_{i=1}^{N} \omega_i \, f(x_i)$$

where:

x_i represent the points where the function is computed

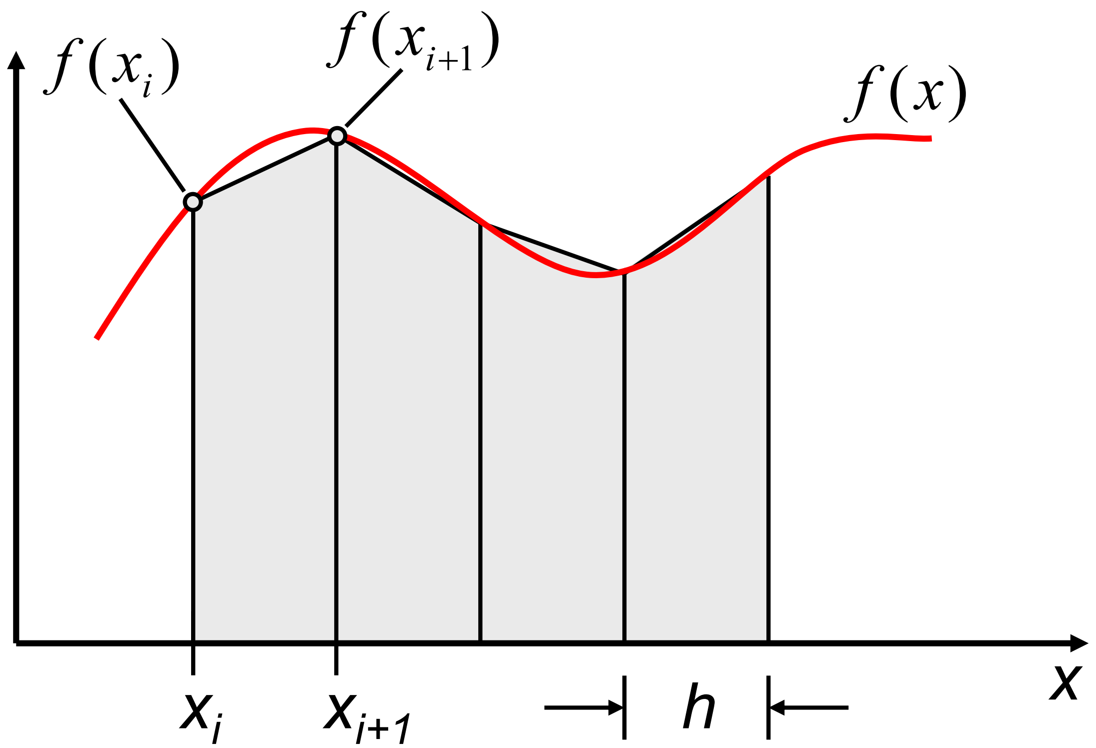
ω_i are the weights used to multiply the samples

EULER'S RULE



$$\int_a^b f(x)\,dx \cong \sum_{i=1}^{N} h\, f(x_i) = h \sum_{i=1}^{N} f(x_i)$$

TRAPEZOIDAL RULE



$$\int_a^b f(x)\,dx \cong \sum_{i=1}^{N} h\left[\frac{f(x_{i-1})+f(x_i)}{2}\right] = \frac{h}{2}f(x_0)+\left[h\sum_{i=1}^{N-1} f(x_i)\right]+\frac{h}{2}f(x_N)$$

SIMPSON'S RULE

Simpson's rule gives a more accurate result than the trapezoidal rule, as the integrand function is approximated by a second-degree polynomial (i.e., a parabola) in each sub-interval.

$$\int_a^b f(x)\,dx \cong \sum_{i=1}^{N} h\left[\frac{f(x_{i-1}) + f(x_i) + f(x_{i+1})}{2}\right] =$$

$$= \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 4f(x_{N-2}) + 2f(x_{N-1}) + f(x_N)\right]$$

where N is an even number.

NUMERICAL INTEGRATION / 5

More sophisticated techniques are based on higher order polynomial interpolation: the integrand function is interpolated by using a polynomial, which is subsequently integrated analytically.

NEWTON-COTES RULES

- Equally spaced sample points
- Weights are computed in such a way, that the quadrature rule with N points exactly integrates polynomials with order N-1
  (N weights represent N degrees of freedom)

GAUSSIAN RULES

- Sample points are not equally spaced
- Points and weights are computed in such a way, that the quadrature rule with N points exactly integrates polynomials with order 2N-1
  (N points + N weights represent 2N degrees of freedom)

NUMERICAL INTEGRATION / 6

Points and weights for Gaussian integration, in the normalized interval (-1,1)

rule	weights	points
2	ω_1 = 1.000000000 ω_2 = 1.000000000	x_1 = -0.577350269 x_2 = 0.577350269
3	ω_1 = 0.555555556 ω_2 = 0.888888889 ω_3 = 0.555555556	x_1 = -0.774596669 x_2 = 0.000000000 x_3 = 0.774596669
4	ω_1 = 0.347854845 ω_2 = 0.652145155 ω_3 = 0.652145155 ω_4 = 0.347854845	x_1 = -0.861136312 x_2 = -0.339981044 x_3 = 0.339981044 x_4 = 0.861136312

SOLUTION OF A LINEAR SYSTEM / 1

The solution of a set of simultaneous equations (linear system)

$$\mathbf{A}\,\mathbf{X} = \mathbf{B}$$

can be based on:

- direct methods (Gauss's elimination method, LU decomposition), in the case of matrices with moderate size (up to 100x100).

- iterative methods, in the case of matrices with larger dimension.

GAUSS'S ELIMINATION METHOD

By a number of transformations, matrix $\mathbf{A}$ is converted into a triangular matrix.

The set of equations is then solved by back-substitution.

computational effort ⇨ $O(N^3)$

Drawbacks of this method:

- the procedure to transform matrix $\mathbf{A}$ into a triangular matrix changes also vector $\mathbf{B}$ (therefore, if $\mathbf{B}$ changes, the procedure has to be repeated).
- in some cases, a further re-ordering is needed (pivoting)

SOLUTION OF A LINEAR SYSTEM / 3

LU DECOMPOSITION (CHOLESKY'S METHOD)

Through a number of transformations, which do not affect vector $\mathbf{B}$, matrix $\mathbf{A}$ is factorized in the form $\mathbf{A}=\mathbf{LU}$, i.e., as the product of a lower triangular matrix ($\mathbf{L}$) and an upper triangular matrix ($\mathbf{U}$).

By replacing $\mathbf{A}=\mathbf{LU}$ in the matrix equation $\mathbf{AX}=\mathbf{B}$, it results: $\mathbf{LUX}=\mathbf{B}$.
An auxiliary matrix $\mathbf{Y}$ is defined, thus obtaining:

$$\begin{cases} \mathbf{U}\,\mathbf{X} = \mathbf{Y} \\ \mathbf{L}\,\mathbf{Y} = \mathbf{B} \end{cases}$$

The computational effort is $O(N^3)$, but with a weight of 1/3 compared to the Gauss's elimination method.

SOLUTION OF A LINEAR SYSTEM / 4

ITERATIVE METHODS

Instead of solving directly the system of equations, a tentative solution $\mathbf{X}_0$ is adopted, and iteratively updated according to the formula

$$\mathbf{X}_n = \mathbf{C}\,\mathbf{X}_{n\text{-}1} + \mathbf{D} \qquad (n = 1,2,...)$$

until the convergence is achieved, which is defined by $\left|\mathbf{X}_n - \mathbf{X}_{n\text{-}1}\right| < \varepsilon$

The convergence process can be improved by adopting a preconditioner **P**, thus solving the matrix equation

$$\mathbf{P}\,\mathbf{A}\,\mathbf{X} = \mathbf{P}\,\mathbf{B}$$

where **P** is an approximation of the inverse of matrix **A** $(\mathbf{P} \approx \mathbf{A}^{-1})$