# Part I
# The Numerical Methods

## 1. Finite Element Method

### *1.1. Basic principle of finite element method*

The finite element method (FEM) is a numerical method for the approximate solution of problems which are described by differential equations. The practical use of FEM began in the 1950s in construction specially in the aerospace and car industry. The technique is based on the fact that the calculation area will be divided into simple sub-areas, the finite elements. Due to their simple geometry, their physical behavior can be calculated well with known shape functions. This shape functions contain parameters that have a physical meaning at a certain point in component and time. The accuracy of the approximate solution can be improved by using more parameters, that means smaller elements or using approach functions of a higher grade. The physical behavior of the entire body is realized through the transition from one element to the next element. Since this method requires considerable computing power, the process thus benefits from the development of powerful computers.

### *1.2. Procedure of the FEM*

To summarize how the finite element method works in general, it will outline the main steps of the finite element method Element solution methods listed below [1].

1. Discretion of the calculation domain: The first step is to divide a solution region into finite elements.
2. Select interpolation functions (shape functions): The second step is to choose the interpolation function to interpolate the field variables over the elements.
3. Define the element properties: Set the matrix equation for the finite elements, which relates the nodal values of the unknown function to other parameters. The most convenient method is the Galerkin method.
4. Assemble the element equations: It should combine local equations for all elements which are used for discretization and the boundary conditions should also be considered.
5. Solve the global equation system: It is a system of linear equations with a symmetric matrix. For the solution, direct or iterative methods can be used to find the unknown parameters.

### *1.3. Illustration of the method using an example*

It It is considered a simple one-dimensional example to explain the finite element formulation using Galerkin method. This is about the principle of using FEM to solve differential equations with boundary conditions, the extension to higher dimensions is similar and can be found in the literature[2]. The following differential equation is given

$$a\frac{d^2u}{dx^2} + b = 0, \quad 0 \leq x \leq 2L \tag{1}$$

with Dirichlet and Neuman boundary condition

$$\begin{aligned} u\,|_{x=0} &= 0 \\ a\frac{du}{dx}\,|_{x=2L} &= q \end{aligned} \tag{2}$$

The domain is one-dimensional and consists of two linear elements and three nodes Fig.1.

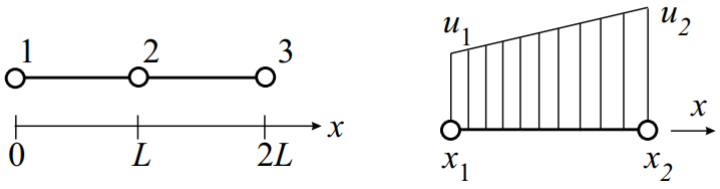


Figure 1: One-dimensional case with two linear elements and an interpolation of a function for the domain[1]

The approach taken is that the solution can be approximated in this way

$$\begin{aligned} u_h \approx u = \sum_i N_i u_i = N_1 u_1 + N_2 u_2 = [N]\,\{u\} \\ [N] = \begin{bmatrix} N_1 & N_2 \end{bmatrix} \\ \{u\} = \begin{Bmatrix} u_1 & u_2 \end{Bmatrix} \end{aligned} \tag{3}$$

where $N_i$are the shape function and are developed using the Lagrangian interpolation polynomial

$$\begin{aligned} N_1 &= \frac{x - x_2}{x_1 - x_2} \\ N_2 &= \frac{x - x_1}{x_2 - x_1}. \end{aligned} \tag{4}$$

Substituting this into the equation gives us an error called the residual

$$a\frac{d^2}{dx^2}\,[N]\,\{u\} + b = R. \tag{5}$$

To minimize the error we use the weighted residuals method and in the case of Galerkin method for weighting $w$ bis also the shape function used

$$\int_{\Omega} w(x)R(x)dx = \int_{x1}^{x2} [N]^T a\frac{d^2}{dx^2}[N]\{u\}dx + \int_{x1}^{x2} [N]^T bdx = 0 \tag{6}$$

Here is the problem that the second derivative of the linear function vanishes $\frac{d^2}{dx^2}[N] = 0$. The remedy is partial integration, which leads to a weak formulation of the equation.

$$\int_{x1}^{x2} \left[\frac{dN}{dx}\right]^T a\left[\frac{dN}{dx}\right] dx\{u\} - \int_{x1}^{x2} [N]^T bdx - \left\{\begin{array}{c} 0 \\ 1 \end{array}\right\} a\frac{du}{dx}\mid_{x=x2} + \left\{\begin{array}{c} 1 \\ 0 \end{array}\right\} a\frac{du}{dx}\mid_{x=x1} = 0 \tag{7}$$

The whole equation can now be summarized more briefly in matrix and vector form

$$\begin{aligned} &[K]\{u\} = \{f\} \\ &[K] = \int_{x1}^{x2} \left[\frac{dN}{dx}\right]^T a\left[\frac{dN}{dx}\right] dx \\ &\{f\} = \int_{x1}^{x2} [N]^T bdx + \left\{\begin{array}{c} 0 \\ 1 \end{array}\right\} a\frac{du}{dx}\mid_{x=x2} - \left\{\begin{array}{c} 1 \\ 0 \end{array}\right\} a\frac{du}{dx}\mid_{x=x1}. \end{aligned} \tag{8}$$

In mechanics, $K$ means the stiffness matrix, $u$ is the displacement vector and $f$ is the force vector, these terms have become established in FEM.

For a domain with two elements of length L we get the following matrix and vector equations

$$\begin{aligned} &[K_1] = [K_2] = \frac{a}{L}\left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array}\right] \\ &\{f_1\} = \frac{bL}{2}\left\{\begin{array}{c} 1 \\ 1 \end{array}\right\},\ \{f_2\} = \frac{bL}{2}\left\{\begin{array}{c} 1 \\ 1 \end{array}\right\} + \left\{\begin{array}{c} 0 \\ q \end{array}\right\}. \end{aligned} \tag{9}$$

This two element equation will be assembled because of interact of the elements in node number 2.

$$\begin{aligned} [K]\{u\} &= \left[\begin{array}{ccc} k_1 & -k_1 & 0 \\ -k_1 & k_1+k_2 & k_2 \\ 0 & -k_2 & k_2 \end{array}\right]\left\{\begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}\right\} = \frac{a}{L}\left[\begin{array}{ccc} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{array}\right]\left\{\begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}\right\} \\ &= \left\{\begin{array}{c} f_{11} \\ f_{12}+f_{21} \\ f_{22} \end{array}\right\} = \frac{bL}{2}\left\{\begin{array}{c} 1 \\ 2 \\ 1 \end{array}\right\} + \left\{\begin{array}{c} 0 \\ 0 \\ q \end{array}\right\} \end{aligned} \tag{10}$$

Consider the boundary condition the final appearance equation system is

$$\frac{a}{L}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \frac{bL}{2}\begin{Bmatrix} 0 \\ 2 \\ 1 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ q \end{Bmatrix}. \tag{11}$$

The exact solution and the approximate solution are shown in Fig.2 for the Parameters $a = b = q = L = 1$.

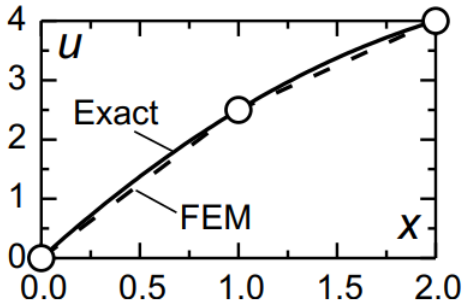


Figure 2: Comparison between analytical and numerical solution[1]

All these steps are realized for the models in this work in Comsol Multiphysics.

## 2. Genetic Algorithms

### *2.1. Historical Context and Basics*

Genetic algorithms (GA) are machine learning models and optimization methods motivated by the evolutionary processes in nature and so the vocabulary was adopted from genetics. In this method, learning is understood as competition between evolved species in a population, and this makes the algorithms directed probability algorithms and thus much more efficient than random search algorithms. GAs were first proposed by John Holland in the 1960s and further developed by his students and colleagues at the University of Michigan in the 1960s and 1970s [3]. Genetic algorithms are useful for many types of complex optimization tasks well suited:

- No fundamental limitations regarding the function to be optimized, such as continuity, derivability, etc.
- Particularly suitable for problems with a very large search space where an optimal search by listing all possible solutions is no longer possible.
- The global extremum is sought and not just the nearest local extremum.

- Genetic algorithms can be efficiently implemented on parallel computing structures.

One of the weaknesses of the algorithm is that finding the optimal solution cannot be guaranteed. More information on this topic can be found in the literature [4]. Algorithm 1 shows the basic structure of a GA.

**Algorithm 1** Basic Genetic Algorithm [5]

$t \leftarrow 0$
Generate population $P(t)$ randomly
evaluate fitness for $P(t)$
**while** ( **not** termination condition )
**do**
$t \leftarrow t + 1$
create new population $P(t)$
select
recombine
mutate
evaluate fitness for$(P(t))$;
**end while**

The GA starts with a randomly generated population (a set of chromosomes) and the fitness is evaluated by the fitness function. Evolutionary it means that the fittest in the population survive and mathematically the fitness function is the function that is trying to optimize the solution. While the termination condition is not achieved, the GA performs a fitness based selection, recombination and mutation process to create a succeeding population and subsequently the next generation. During recombination, parent chromosomes are selected and their genetic material is recombined to produce child chromosomes. These then advance into the next population. As this process is repeated, a sequence of successive generations develops and the average fitness of the chromosomes tends to increase until the termination criterion is reached.

*2.2. The detailed structure*

A GA consists of a number of different components. This is a special strength because it means that standard components can be reused in many different GAs with trivial customization, thus easing implementation[6]. The vocabulary for genetic algorithms was taken from genetics.

*2.2.1. Population*

In order to use a genetic algorithm, a solution space must first be specified. From the solution space an initial population is randomly generated which is a set of individuals or chromosomes. At this stage one can apply heuristics to give the algorithm a better starting position. A chromosome represents a point in the search space and consists of genes that represent the elements of a solution 3. The number of genes represents a measure of the size of a solution.

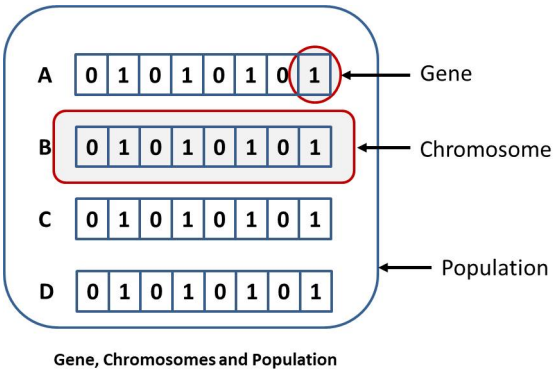


Figure 3: Population construction in GA [7]

*2.2.2. Fitness and Objective Function*

The fitness function of an individual $f(i_k)$ in a population $P = \{i_1, i_2, ..., i_n\}$ gives the probability whether that individual can survive and thus pass on their genes. Since the fitness function is a positive real number, it is obtained by scaling from the objective function $f(i) = scale(o(i))$, which corresponds to the optimization goal of the optimization problem.

*2.2.3. Selection*

The GA uses the selection to appoint individuals for reproduction as a new generation on the principle of biological evolution "survival of the fittest". Individuals with higher fitness have a greater chance of selection than those with lower fitness, thus creating selection pressure on the individuals in the direction of optimal solution. Choosing the right selection pressure is crucial, if the selection pressure is too high, the solution converges prematurely to the local optimum. However, if the selection pressure is too low, the optimal individuals are hardly favored. Therefore, the solution will either not converge or will only converge at a much slower rate.

There are several selection methods and the choice is made according to the task. An example of such a method presented here is the selection scheme Stochastic Universal Sampling (SUS). In this process each Individual is assigned a segment on the roulette wheel. The size of the segment is proportional to fitness of the individual. The wheel is turned and then $n$ pointers arranged evenly around the wheel choose $n$ individuals for reproduction. A detailed description of the various selection schemes can be found in Handbook of Evolutionary Computation [8].

*2.2.4. Crossover*

The crossover operator allows mixing of genetic material from selected initial chromosomes to create offspring chromosomes. After selecting two parent chromosomes for recombination, a random number is generated in the interval $[0, 1]$ with uniform probability and compared with a predetermined "crossover rate". If the random number is greater than the crossover rate, no crossover occurs and one or both parents will proceed unchanged to the

next level or recombination. If the crossover rate is greater than or equal to the random number, the crossover operator is applied Fig 4. There are many other ways to perform the crossover, see the following literature [9].

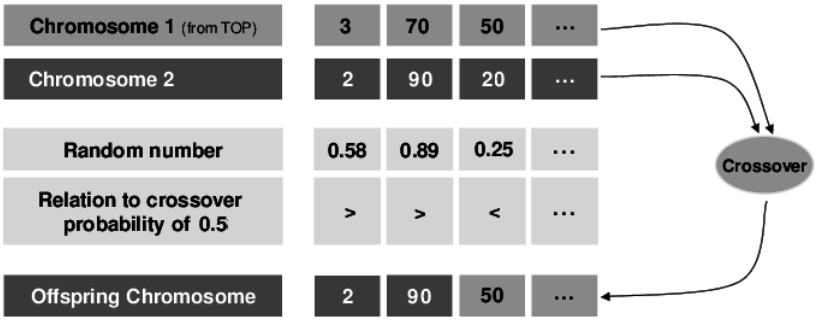


Figure 4: Uniform crossover for crossover rate of 0.5 [10]

*2.2.5. Mutation*

The mutation operator works much like the crossover operator Fig.5. It changes the genes values with less probability $p_{\mathrm{m}}$ and so the solution in smaller step [5].

**Before the mutation:**

Chromosome one: [ 4 50 10 7 9 60 7 600 0.9 3 340 ]

**After the mutation:**

Chromosome one: [ 4 90 10 7 9 60 7 1000 0.9 3 340 ]

Figure 5: Mutation in gene number 2 and 8

The idea is not to get stuck in local minima but , through this method, to reach the global minima.The chosen values for the mutation probability pm are usually 0.01 or 0.001. Some sources [11] prefer the chromosome length $L$ depended mutation probability as the optimal probability $p_{\mathrm{mopt}} = 1/L$, which is used for this implementation.

*2.2.6. Termination*

For the termination of GA there is a criterion needed. Therefore there are several possibilities like time constraints [12] or the consideration of convergence. Another popular variant used for this implementation is a certain number of generations.

## References


[1] Nikishkov, G.P.: *INTRODUCTION TO THE FINITE ELEMENT METHOD.* `https://www.iitg.ac.in/mech/documents/128/introfem.pdf`. Version: 2004

[2] Bathe, Klaus-Jürgen: *Finite-Elemente-Methoden* -. 2nd. Berlin, Heidelberg : Springer, 2016. – ISBN 978–3–540–66806–0

[3] Holland, John: Adaptation in natural and artificial systems: an introductory analysis with application to biology. In: *Control and artificial intelligence* (1975)

[4] Fogel, David B.: *Evolutionary algorithms in theory and practice.* 1997

[5] Buttelmann, Maik ; Lohmann, Boris: Optimierung mit Genetischen Algorithmen und eine Anwendung zur Modellreduktion (Optimization with Genetic Algorithms and an Application for Model Reduction). In: *At-automatisierungstechnik - AT-AUTOM* 52 (2004), April, S. 151–163. http://dx.doi.org/10.1524/auto.52.4.151.29416. – DOI 10.1524/auto.52.4.151.29416

[6] McCall, John: Genetic algorithms for modelling and optimisation. In: *Journal of computational and Applied Mathematics* 184 (2005), Nr. 1, S. 205–222

[7] Kindsonthegenius: *Basics of genetic algorithm*. https://www.kindsonthegenius.com/basics-of-genetic-algor Version: Sep 2020

[8] Back, Thomas (Hrsg.) ; Fogel, David B. (Hrsg.) ; Michalewicz, Zbigniew (Hrsg.): *Handbook of Evolutionary Computation*. 1st. Bristol, UK, UK : IOP Publishing Ltd., 1997. – ISBN 0750303921

[9] Hasançebi, Oğuzhan ; Erbatur, Fuat: Evaluation of crossover techniques in genetic algorithm based optimum structural design. In: *Computers & Structures* 78 (2000), November, Nr. 1-3, 435–448. http://dx.doi.org/10.1016/s0045-7949(00)00089-4. – DOI 10.1016/s0045–7949(00)00089–4

[10] Gonçalves, José ; Resende, Mauricio: A multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. In: *J. Comb. Optim.* 22 (2011), 08, S. 180–201. http://dx.doi.org/10.1007/s10878-009-9282-1. – DOI 10.1007/s10878–009–9282–1

[11] Muhlenbein, Heinz: How genetic algorithms really work: I. mutation and hillclimbing. In: *Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, 1992* Elsevier, 1992

[12] Kramer, Oliver: Genetic algorithms. Version: 2017. http://dx.doi.org/10.1007/978-3-319-52156-5-2. In: *Genetic algorithm essentials*. Springer, 2017. – DOI 10.1007/978–3–319–52156–5–2, 11–19