Class Notes: Introduction to the Finite Element Methods

Juan David Gómez Cataño
jgomezc1@eafit.edu.co
Nicolás Guarín-Zapata
nicoguarin@gmail.com

Grupo de Investigación en Mecánica Aplicada
Civil Engineering Department
School of Engineering
Universidad EAFIT
Medellín, Colombia
2017

Contents

Todo list

Summary

Class Notes for the Course Introduction to the Finite Element Methods

Keywords: Scientific Computing, Computational Mechanics, Finite Element Methods, Numerical Methods.

Chapter 1

Introduction

In an introductory graduate course in the Finite Element Method (FEM), the purpose is to develop a basic understanding of the discrete or numerical strategy of Finite Element-(FE) Algorithms when a closed form solution to a Boundary Value Problem (BVP) is not possible due to complexities existing probably: in the boundary conditions, material behavior or in the kinematic description used in the model. In such a course, in order to master and identify the main mathematical and algorithmic aspects of the method at the beginners level, the studied problem is kept lineal. In this sense 4 key aspects make the core of the introductory course (at least for the case of the linearized theory of elasticity boundary value problem):

- Formulation and identification of the strong form of the Boundary Value Problem (BVP) where the continuum mechanics governing equations are revisited and particularized to the case of linear elastic material behavior. The BVP is completed by identifying the correct prescription of boundary conditions for a well posed mathematical problem.
- Formulation and identification of the weak form of the BVP where the governing equations and natural boundary conditions representing equilibrium at the material point level are replaced by an equivalent but weaker form of equilibrium now valid at the global level. In the case of a Continuum Mechanics problem this so-called weak form can be shown to be equivalent to the principle of virtual power and lends itself for the partition of the problem into sub-domains or finite size elements.
- Introduction of the idea of discretization dividing the problem into subdomains and using interpolation theory within each subdomain-In the context of the FE method this is the subject of shape functions. Once the computational specimen has been divided into subdomains (or a mesh of finite elements) the selected primary variable is approximated within each element via interpolation of the known response at selected predefined points or nodes.
- Computational aspects grouped also into 5 points:
  - Formulation of elemental matrices in the physical space.

- Formulation of elemental matrices in the natural domain-Isoparametric transformation.
- Numerical integration: Gauss quadrature.
- Assembly of system matrices and imposition of boundary conditions.
- Solution of the discrete equilibrium statement and calculation of elemental results.

These 4 key points are perfectly well documented in numerous nicely written textbooks and it could be argued that is not even worth to register for an introductory course since a moderately dedicated student can accomplish the task via self-study. Unfortunately, linearity is scarce–although useful to grasp the basic understanding of the problem–and the real world is full of non-linear behavior and understanding the needed algorithms can be easily justified. In this Advanced Finite Element Methods course the goal is then to understand the basic aspects of non-linear finite element analysis. Like in the linear case there is also a vast amount of literature for the non-linear problem. However, in the non-linear case the kinematic problem itself may take different routes leading to a wide variety of FEM formulations that will difficult a self-study strategy. Considering the above this brief set of class notes is intended as a guide for self-study and more important represents a help towards the implementation of the algorithm for the consideration of Material and Geometric Non-linearities in Solids.

Check name of the "course"

The basic reference is Professor Bathe's textbook [1] but we will also follow closely ABAQUS Theory Manual [2]. ABAQUS is a multi-physics oriented commercially available finite element analysis tool. Its strength resides in the effective non-linear algorithms and on the capability of taking user subroutines written in Fortran or in C++. The possibility of implementing user subroutines makes it a very powerful research tool. In the particular case of a stress/displacement analysis problem with non-linearities these are considered through the kinematics contribution or through the material contribution. In the first case the non-linear behavior must be considered at the element level while in the second it corresponds to the response of a material point which in the context of the FEM algorithm corresponds to an integration point. In ABAQUS those two sources of non-linearity can be independently controlled by the user via user subroutines UEL and UMAT. In both cases the non-linearity is primarily solved by the classical Newton-Raphson scheme and that will be the approached followed herein. Although the notes are mainly written for an advanced course the specific problem of a linear solid usually studied in the introductory course can be derived like a particular case of the most general non-linear algorithm.

The current set of Class Notes is organized as follows. First and since we will be dealing with history dependent non-linear problems the most powerful (at least when it works) solution algorithm, namely the Newton-Raphson iteration is studied. The technique is first illustrated for the simple 1D-case and then generalized into the multi-degree of freedom system. In both cases pseudo-codes will be presented preparing the way for the Finite Element Algorithm. The presentation however is not exhaustive in mathematical terms and the reader is referred to excellent treatments like Burden [3] and Press et al. [4] for the mathematical aspects of the Newton method.

In the next section the briefly introduced Newton-Raphson technique is contextualized to the case of a system of equations representing equilibrium between internal and external forces as

typically found in a finite element model. Moreover, the non-linearities come into play through a dependence of the internal forces into displacements. At this stage the details of the formulation of the finite element equations via discretization into nodal variables is not presented but emphasis is laid down into the solution algorithm. Interest is then given to the particular form taken by the Newton-Raphson algorithm into the commercial finite element code ABAQUS. That code can be used as a powerful non-linear equation solver where the coefficient matrix and the excitation can be directly controlled by the user. Moreover the solver can be used into a multi-physics context in terms of generalized forces and fluxes. In the particular case of the stress analysis finite element method the user can control the elemental contribution to the coefficient matrix and the contribution of each material point to obtain that element contribution. This is achieved through the so-called user subroutines UEL for element and UMAT for material. Having introduced the Newton-Raphson method the notes concentrate next on general discretization aspects starting from the physical strong form of the equations in the deformed configuration and passing to an arbitrary weak form in the reference configuration. The resulting algorithm is therefore a Total Lagrangian (TL) method. Once the general equations are introduced a particular work conjugate stress-strain pair is chosen and the discrete equations, including kinematic interpolators, are described.

Notation

In a non-linear algorithm the bookkeeping is involved since we have to simultaneously record 4 different fields as follows:

- The physical fields in terms of tensorial descriptions.
- The time field since the problem is solved incrementally and time may appear as an artificial chronological variable or as a real quantity in a dynamic problem.
- The interpolation field. Since all the involved variables will be interpolated we will need to keep track of the way this interpolation is being performed.
- The iterations field needed in the solution of the non-linear problem.

In order to keep this bookkeeping simple we use the following indicial notation with subscripts and superscripts

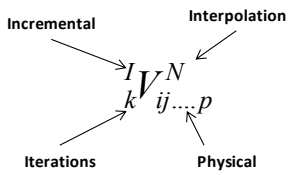


Figure 1.1. General notation to study non-linear finite element problems

Capital superscripts will be reserved for the incremental time description and for the interpolation scheme. For instance an expression like refers to the time instant while refers to . Similarly a variable refers to interpolation over the node. Left and right subscripts will be used to make reference to the iteration being performed and the order of the tensor variable. For instance refers to a second order tensor corresponding to the iteration.

Review notation.

Chapter 2

The Finite Element Method

2.1 Boundary value problems

In this section we will define a general initial boundary value problem (I-BVP). In the first part we will introduce the differential formulation given in terms of a set of governing equations and properly specified boundary conditions. The resulting equations are obtained after using a generalized balance law. Following this classical and well known approach we formally re-state these equations in the so-called strong form. Subsequently we re-write and prove an equivalent form of the balance law in the form of an integral representation highly friendly for a numerical solution. Since in the integral description of the problem the order of the derivatives in the field functions decreases by one, the resulting statement is called a weak formulation.

2.1.1 Differential formulation-Generalized balance law

Let ds be a differential surface element; $\mathrm{d}V$ a differential volume element; $u(\mathbf{x}, t)$ a scalar (or vector) function of space and time. The flux or rate of flow of the quantity $u(\mathbf{x}, t)$ through $\mathrm{d}S$ at time t is defined like

$$p(\mathbf{x})\boldsymbol{\nabla} u \cdot \hat{n}\, \mathrm{d}S \quad ,$$

where $p(\mathbf{x})$ is a positive function, assumed known and time independent. Similarly, the time rate of change of $u(\mathbf{x}, t)$ in an element $\mathrm{d}V$ is given by

$$\rho(\mathbf{x})\frac{\partial u}{\partial t}\, \mathrm{d}V \quad ,$$

where once again $\rho(\mathbf{x})$ is a known, given, time independent positive function. Additional effects occurring in the element $\mathrm{d}V$ at the time time can be expressed like

$$H(\mathbf{x}, t)\, \mathrm{d}V \equiv -q(\mathbf{x})u(\mathbf{x}, t) + \hat{F}(\mathbf{x}, t)$$

where $\hat{F}(\mathbf{x},t) = \rho(\mathbf{x})F(\mathbf{x},t)$. In the above the term qu represent internal effects due to changes proportional to u while $\hat{F}(\mathbf{x},t)$ are other external influences in the medium.

Balancing the internal and external changes yields

$$\int_V \rho(\mathbf{x})\frac{\partial u}{\partial t}\,\mathrm{d}V = \int_S p(\mathbf{x})\boldsymbol{\nabla}u\cdot\hat{n}\,\mathrm{d}S + \int_V H(\mathbf{x},t)\,\mathrm{d}V \quad ,$$

or equivalently

$$\int_V \rho(\mathbf{x})\frac{\partial u}{\partial t}\,\mathrm{d}V = \int_S p(\mathbf{x})\boldsymbol{\nabla}u\cdot\hat{n}\,\mathrm{d}S - \int_V q(\mathbf{x})u(\mathbf{x},t)\,\mathrm{d}V + \int_V \rho(\mathbf{x})F(\mathbf{x},t)\,\mathrm{d}V \quad .$$

Using the divergence theorem as

$$\int_S p(\mathbf{x})\boldsymbol{\nabla}u\cdot\hat{n}\,\mathrm{d}S = \int_V \boldsymbol{\nabla}\cdot(p\boldsymbol{\nabla}u)\,\mathrm{d}V$$

yields after substitution

$$\int_V \left[\rho(\mathbf{x})\frac{\partial u}{\partial t} - \boldsymbol{\nabla}\cdot(p(\mathbf{x})\boldsymbol{\nabla}u) + q(\mathbf{x})u(\mathbf{x},t) - \rho(\mathbf{x})F(\mathbf{x},t)\right]\mathrm{d}V = 0$$

Assuming a continuous integrand, the arbitrariness of V implies

$$\rho(\mathbf{x})\frac{\partial u}{\partial t} - \boldsymbol{\nabla}\cdot(p(\mathbf{x})\boldsymbol{\nabla}u) + q(\mathbf{x})u(\mathbf{x},t) - \rho(\mathbf{x})F(\mathbf{x},t) = 0$$

Letting

$$\mathcal{L} \equiv -\boldsymbol{\nabla}\cdot p(\mathbf{x})\boldsymbol{\nabla} + q(\mathbf{x})$$

the generalized set of partial differential equations can be written like

$$\rho(\mathbf{x})\frac{\partial u(\mathbf{x},t)}{\partial t} + \mathcal{L}u(\mathbf{x},t) = \rho(\mathbf{x})F(\mathbf{x},t) \quad . \tag{2.1}$$

And they are categorized as:

- Hyperbolic;
$$\rho(\mathbf{x})\frac{\partial^2 u(\mathbf{x},t)}{\partial t^2} + \mathcal{L}u(\mathbf{x},t) = 0$$
- Parabolic; and
$$\rho(\mathbf{x})\frac{\partial u(\mathbf{x},t)}{\partial t} + \mathcal{L}u(\mathbf{x},t) = 0$$

- Elliptic

$$\mathcal{L}u(\mathbf{x},t) = \rho(\mathbf{x})F(\mathbf{x},t) \ .$$

It can be shown that $\mathcal{L}$ satisfies the *symmetry* condition

$$\int_V \mathcal{L}(u)v \, \mathrm{d}V = \int_V \mathcal{L}(v)u \, \mathrm{d}V$$

and positive definiteness [5, 6]

$$\int_V \mathcal{L}(u)u \, \mathrm{d}V > 0, \quad \forall u \ .$$

Strong form

Given $\rho(\mathbf{x})$, $q(\mathbf{x})$, $p(\mathbf{x})$, $F(\mathbf{x},t)$ and $\bar{u}$ find $u(\mathbf{x},t) : V \to \mathbb{R}$ such:

$$\rho(\mathbf{x})\frac{\partial u}{\partial t} - \boldsymbol{\nabla} \cdot [p(\mathbf{x})\boldsymbol{\nabla} u] + q(\mathbf{x})u(\mathbf{x},t) - \rho(\mathbf{x})F(\mathbf{x},t) = 0 \quad \forall \mathbf{x} \in V$$

and

$$\begin{aligned} u &= \bar{u} \quad \forall \mathbf{x} \in S_u \\ p(\mathbf{x})u_{,i}\hat{n}_i &= B(\mathbf{x},t) \quad \forall \mathbf{x} \in S_t \ . \end{aligned}$$

In the FEM we will look for approximate solutions to u subject to the following conditions:

$$u = \bar{u} \quad \forall \mathbf{x} \in S_u \quad \text{(Essential boundary conditions)}$$

and

$$\int_S \left(\frac{\partial u}{\partial x_j}\right)^2 \mathrm{d}S < \infty \ ,$$

which corresponds to the functions being square integrable. We will denote this space by $\mathbb{H}$.

The space of functions satisfying the above two conditions will be denoted by ζ and termed the space of trial functions, formally defined like:

$$\zeta = \{u \mid u \in \mathbb{H}, u = \bar{u} \quad \forall \mathbf{x} \in S_u\}$$

On the other hand, to validate (or test) the correctness of the approximated or proposed trial functions u it is also necessary to introduce test functions w which are arbitrary except that they satisfy the following conditions:

$$w = 0 \quad \forall \mathbf{x} \in S_u$$

and

$$\int_S \left(\frac{\partial w}{\partial x_j}\right)^2 \mathrm{d}S < \infty \ ,$$

which corresponds to the functions being square integrable. The space of functions satisfying the above two conditions will be denoted by $\mathcal{L}$ and termed the space of test functions, formally defined like:

$$\mathcal{L} = \{w \mid w \in \mathbb{H}, w = 0 \quad \forall \mathbf{x} \in S_u\}$$

Weak form

Given $\rho(\mathbf{x})$, $q(\mathbf{x})$, $p(\mathbf{x})$, $F(\mathbf{x},t)$ and $\bar{u}$ find $u(\mathbf{x},t) : V \to \mathbb{R}$ and $\forall w \in \mathcal{L}$ such:

$$\int_V p(\mathbf{x}) u_{,i}\, w_{,i}\, \mathrm{d}V - \int_{S_t} B(\mathbf{x},t) w\, \mathrm{d}S + \int_V q(\mathbf{x}) u(\mathbf{x},t) w\, \mathrm{d}V + \int_V \rho(\mathbf{x}) \frac{\partial u}{\partial t} w\, \mathrm{d}V - \int_V \rho(\mathbf{x}) F(\mathbf{x},t) w\, \mathrm{d}V = 0$$

and

$$u = \bar{u} \quad \forall \mathbf{x} \in S_u \ .$$

Equivalence between the strong and weak forms

$$-\int_V [p(\mathbf{x}) u_{,i}]_{,i} w\, \mathrm{d}V + \int_{S_t} [p(\mathbf{x}) u_{,i}] \hat{n}_i w\, \mathrm{d}S - \int_{S_t} B(\mathbf{x},t) w\, \mathrm{d}S + \int_V q(\mathbf{x}) u(\mathbf{x},t) w\, \mathrm{d}V + \int_V \rho(\vec{x}) \frac{\partial u}{\partial t} w\, \mathrm{d}V - \int_V \rho(\mathbf{x}) F(\mathbf{x},t) w\, \mathrm{d}V = 0 \quad (2.2)$$

Grouping together common terms yields

$$\int_V \left\{ \rho(\mathbf{x}) \frac{\partial u}{\partial t} - [p(\mathbf{x}) u_{,i}]_{,i} + q(\mathbf{x}) u(\mathbf{x},t) - \rho(\mathbf{x}) F(\mathbf{x},t) \right\} w\, \mathrm{d}V + \int_{S_t} \left\{ [p(\mathbf{x}) u_{,i}] \hat{n}_i - B(\mathbf{x},t) \right\} w\, \mathrm{d}S = 0$$

from which

$$\rho(\mathbf{x}) \frac{\partial u}{\partial t} - [p(\mathbf{x}) u_{,i}]_{,i} + q(\mathbf{x}) u(\mathbf{x},t) - \rho(\mathbf{x}) F(\mathbf{x},t) = 0$$

and

$$p(\mathbf{x}) u_{,i} \hat{n}_i = B(\mathbf{x},t) \quad \forall \mathbf{x} \in S_t \ .$$

2.2 Brief review of the linearized theory of elasticity model

Here we present a brief description of the boundary value problem governing the response of an elastic body. For a full discussion of the model and its mathematical aspects the reader is referred to [7].

The governing equations (in terms of stresses) stem from the principle of conservation of linear momentum and conservation of moment of linear momentum. The former leads to a set of 3 partial differential equations in the components of the stress tensor while the latter leads to the symmetries in the stress tensor.

$$\begin{aligned} &\sigma_{ij,j} + f_i = \rho \ddot{u}_i \quad \forall\, \mathbf{x} \in V,\, t \in \mathbb{R}^+ \\ &\sigma_{ij} = \sigma_{ji}. \end{aligned} \tag{2.3}$$

In eq. (2.3) σ_{ij} is the stress tensor; f_i is the vector of body forces; and u_i is the displacements vector.

Denoting the tractions vector associated with a surface with normal direction $\hat{n}_j$ by $t_i^{\hat{n}}$ we have the complete BVP as follows:

$$t_i^{\hat{n}} = \sigma_{ij}\hat{n}_j \quad \forall \in \mathbf{x} \in S. \tag{2.4}$$

Equation (2.3) correspond to 6 equations with 12 unknowns (the 9 components of the stress tensor and the 3 components of the displacements vector) and the system is undetermined. In order to have a solvable BVP we must introduce kinematic strain-displacement relations and a stress-strain law. In the case of infinitesimal theory of elasticity the strain-displacement relation is given by:

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2.5}$$

where the term ϵ_{ij} is the symmetric component of the displacements gradient tensor. The components of the strain tensor describe the distortions and changes in magnitude (volumetric changes) of the material point in the continuum model. Now the simplest stress-strain (constitutive) relationship is given by Hooke's law[1]

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} \ . \tag{2.6}$$

[1]Despite the name of *law* used, this relation is not always valid, but is a good approximation for small strains.

where μ and λ are material constants. The problem involves now a total of 18 equations and 18 unknowns can be solved if subjected to properly specified boundary conditions.

Displacement formulation

Substituting eq. (2.5) in eq. (2.6) and the result in eq. (2.3) yields after some manipulation:

$$(\lambda + \mu)u_{j,ij} + \mu u_{i,jj} + f_i = \rho \ddot{u}_i \quad \forall \mathbf{x} \in V,\, t \in \mathbb{R}^+. \tag{2.7}$$

Since eq. (2.7) (simultaneously describing equilibrium, kinematic relations and constitutive response) is a second order equation governing the displacement field possible boundary conditions are in terms of the variable itself or its first oder derivatives. For a well-posed problem the following is a set of valid boundary conditions:

$$\begin{aligned} t_i^{\hat{n}} &= \sigma_{ij}\hat{n}_{ij} \quad \forall\ \mathbf{x} \in S_t \\ u_i &= \bar{u}_i \quad \forall \mathbf{x} \in S_u \end{aligned} \tag{2.8}$$

and where $S_t \cup S_u = S$ and $S_t \cap S_u = \emptyset$.

In the particular case in which u_i is not a function of time, we obtain the static version of the BVP, i.e,

$$\begin{aligned} &(\lambda + \mu)\, u_{j,ij} + \mu u_{i,jj} + f_i = 0 \quad \forall \mathbf{x} \in V \\ &t_i^{\hat{n}} = \sigma_{ij}\hat{n}_{ij} \quad \forall\ \mathbf{x} \in S_t \\ &u_i = \bar{u}_i \quad \forall \mathbf{x} \in S_u \end{aligned} \tag{2.9}$$

Notice that the tractions BC

$$t_i^{\hat{n}} = \mu(u_{i,j} + u_{j,i})\hat{n}_j + \lambda u_{k,k}\delta_{ij}\hat{n}_j \ ,$$

actually involves first order displacements derivative and as such it is a Neumann boundary condition on u_i.

2.2.1 Equivalence between strong and weak forms

Strong form

The strong form corresponds to the differential formulation of the problem, it is denoted by $\{S\}$ and it reads:

Given f_i, $t_i^{\hat{n}}$ and $\bar{u}_i$ find $u_i : V \to \mathbb{R}$ such:

$$\begin{aligned} &(\lambda+\mu)u_{j,ij} + \mu u_{i,jj} + f_i = 0 \quad \forall \mathbf{x} \in V \\ &t_i^{\hat{n}} = \sigma_{ij}\hat{n}_{ij} \quad \forall \mathbf{x} \in S_t \\ &u_i = \bar{u}_i \quad \forall \mathbf{x} \in S_u \end{aligned} \tag{2.10}$$

In eq. (2.10) the boundary conditions specified by the traction vector $t_i^{\hat{n}}$ correspond to the natural boundary conditions, while those specified in terms of the displacements vector $\bar{u}_i$ represent the essential boundary conditions.

- We are interested in developing methods to obtain approximate solutions to $\{S\}$.
- The FEM is formulated starting from a statement equivalent to $\{S\}$ in which we use trial functions until certain prescribed conditions are met.
- We will look for solutions u_i subject to the following conditions:

$$u_i = \bar{u}_i \qquad \text{in} \qquad S_u$$
$$\int_S \left(\frac{\partial u_i}{\partial x_j}\right)^2 dS < \infty$$

The first condition corresponds to the satisfaction of the essential boundary condition, while the second corresponds to the functions being square integrable. The space of functions satisfying the above two conditions is denoted by ς and formally defined like

$$\varsigma = \{u_i \,|u_i \in H, u_i = \bar{u}_i \in S_u\} \quad .$$

On the other hand, in order to validate the introduced trial functions we also need testing functions w_i also called in the FEM literature weighting or distribution functions. These functions are arbitrary apart from having to satisfy the following conditions:

$$w_i = 0 \quad in \quad S_u$$
$$\int_S \left(\frac{\partial w_i}{\partial x_j}\right)^2 dS < \infty$$

In what follows we formally denote the space of these functions by V and define it like

$$V = \{w_i \,|w_i \in H, u_i = w_i = 0 \in S_u\} \quad .$$

Weak form

Here we will show that the equilibrium statement represented in the differential formulation can be described in alternative forms. In such description the continuity requirement for the trial functions is weaker than in the strong form leading to the term "weak" statement. Here this alternative representation will be denoted like $\{W\}$ and it reads;

Given f_i, $t_i^{\hat{n}}$ and $\bar{u}_i$ find $u_i : V \to \mathbb{R}$ and $\forall w_i \in V$ such:

$$\int_V \sigma_{ij} w_{i,j}\, dV - \int_V f_i w_i\, dV - \int_{S_t} t_i^{\hat{n}} w_i\, dS = 0$$

Proof 1:

Let $u_i \in \varsigma$ be a solution to $\{S\}$ and let $w_i \in V$. Forming the inner product of the equilibrium statement given in eq. (2.3) with w_i and forcing the integral over the domain to be zero we have

$$\int_V (\sigma_{ij,j} + f_i) w_i\, dV = 0 \ ,$$

expanding the terms in the integrand and integrating by parts the first term on the left we have

$$\int_V \sigma_{ij,j} w_i\, dV + \int_V f_i w_i\, dV = 0 \ .$$

$$-\int_V w_{i,j}\sigma_{ij}\, dV + \int_S \sigma_{ij}\hat{n}_j w_i\, dS + \int_V w_i f_i\, dV = 0$$

since $w_i \in V$ it follows that $w_i = 0$ in S_u from which

$$\int_V \sigma_{ij} w_{i,j}\, dV - \int_V f_i w_i\, dV - \int_{S_t} t_i^{\hat{n}} w_i\, dS = 0 \tag{2.11}$$

Now, considering that u_i is solution of the strong form $\{S\}$ it must satisfy $u_i = \bar{u}_i \quad \in \quad S_u$ and as a result $u_i \in \varsigma$. On the other hand, since u_i satisfies eq. (2.11) $\forall w_i \in V$ we have that u_i satisfies the definition of weak solution specified in $\{W\}$.

Proof 2:

Let u_i be a solution of $\{W\}$ and thus $u_i \in \varsigma$ which means that

$$u_i = \bar{u}_i \quad \in \quad S_u$$

and that it satisfies

$$\int_V \sigma_{ij} w_{i,j}\, dV - \int_V f_i w_i\, dV - \int_{S_t} t_i^n w_i\, dS = 0 \ ,$$

integrating by parts,

$$-\int_V \sigma_{ij,j} w_i dV + \int_S \sigma_{ij} n_j w_i dS - \int_V f_i w_i dV - \int_{S_t} t_i^n w_i dS = 0$$

Since $w_i \in V$ we have that $w_i = 0$ in S_u and therefore

$$\int_V w_i(\sigma_{ij,j} + f_i) dV + \int_{S_t} w_i(\sigma_{ij} n_j - t_i^n) dS = 0$$

from which

$$\begin{aligned} &\sigma_{ij,j} + f_i = 0 \quad \mathbf{x} \in V \\ &t_i^n = \sigma_{ij} n_j \quad \forall \mathbf{x} \in S_t \\ &u_i = \bar{u}_i \quad \forall \mathbf{x} \in S_u \end{aligned} \tag{2.12}$$

which is once again the strong form of the problem given in eq. (2.3).

Simple wedge under self-equilibrated loads

Consider the double wedge of side ℓ and internal angle 2ϕ shown in fig. 2.1. It is assumed to be contained in the $X - Y$ plane, with loading conditions satisfying a plane strain (or plane stress) idealization. The material is elastic with Lame constants λ and μ. The wedge is loaded by uniform tractions of intensity S applied over its four faces in such a way that the wedge is self-equilibrated. We wish to find the closed-form elasticity solution for the stress, strain and displacement fields throughout the problem domain.

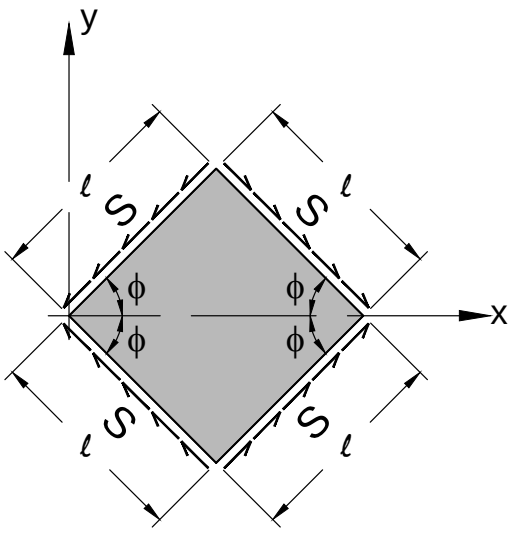


Figure 2.1. 2D Self-equilibrated wedge.

Under plane strain conditions the general 3D stress equilibrium equations (see eq. (2.3)) reduce to:

$$\begin{aligned} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} &= 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} &= 0 \end{aligned} \tag{2.13}$$

while the kinematic relation (eq. (2.5)) reads

$$\begin{aligned} \epsilon_{xx} &= \frac{\partial u}{\partial x} \\ \epsilon_{yy} &= \frac{\partial v}{\partial y} \\ \gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{aligned} \tag{2.14}$$

where u and v are the horizontal and vertical displacements respectively.

Stress field

The stress field can be obtained by simple inspection from the traction boundary conditions prescribed over the inclined surfaces yielding;

$$\sum F_x = 0 \longrightarrow -\ell S\cos(\phi) + \sigma_{xx}\ell\sin(\phi) = 0$$
$$\sum F_y = 0 \longrightarrow -\ell S\sin(\phi) - \sigma_{yy}\ell\cos(\phi) = 0$$

and the following stress solution:

$$\begin{aligned}\sigma_{xx} &= S\cot(\phi)\\ \sigma_{yy} &= -S\tan(\phi)\\ \tau_{xy} &= 0.\end{aligned} \tag{2.15}$$

In eq. (2.15) the condition $\tau_{xy} = 0$ is due to the symmetries in the problem.

Traction boundary conditions

Let us verify that the above stress solution satisfies the traction BC using the expression:

$$t_i^{\hat{n}} = \sigma_{ij}\hat{n}_{ij}.$$

Denoting the outward normals to the inclined surfaces of the wedge by $\hat{n}^1$, $\hat{n}^2$, $\hat{n}^3$, $\hat{n}^4$ these are given by;

$$\begin{aligned}\hat{n}^1 &= -\sin(\phi)\hat{e}_x + \cos(\phi)\hat{e}_y\\ \hat{n}^2 &= -\sin(\phi)\hat{e}_x - \cos(\phi)\hat{e}_y\\ \hat{n}^3 &= +\sin(\phi)\hat{e}_x + \cos(\phi)\hat{e}_y\\ \hat{n}^4 &= +\sin(\phi)\hat{e}_x - \cos(\phi)\hat{e}_y\end{aligned}$$

where $\hat{e}_x$ and $\hat{e}_y$ are the reference unit vectors. Now, the components of the traction vector follow directly like

$$t_i = \sigma_{ij}\hat{n}_j$$

then over the face with normal $\hat{n}^1$ we have

$$\begin{aligned}t_x &= -S\cos(\phi)\\ t_y &= -S\sin(\phi)\end{aligned}$$

similarly, over the face with normal $\hat{n}^2$

$$\begin{aligned} t_x &= -S\cos(\phi) \\ t_y &= +S\sin(\phi) \end{aligned}$$

over the face with normal $\hat{n}^3$

$$\begin{aligned} t_x &= +S\cos(\phi) \\ t_y &= -S\sin(\phi) \end{aligned}$$

and finally, over the face with normal $\hat{n}^4$;

$$\begin{aligned} t_x &= +S\cos(\phi) \\ t_y &= +S\sin(\phi) \ . \end{aligned}$$

Strain field

The strain field can be obtained after using the stress solution found in eq. (2.15) together with the constitutive law given by eq. (2.6) which for a plane strain idealization takes the form:

$$\begin{aligned} \epsilon_{xx} &= \frac{1}{E}(\sigma_{xx} - \nu\sigma_{yy}) \\ \epsilon_{yy} &= \frac{1}{E}(\sigma_{yy} - \nu\sigma_{xx}) \\ \gamma_{xy} &= \frac{\tau_{xy}}{\mu} \end{aligned} \tag{2.16}$$

which for the particular case yields;

$$\begin{aligned} \epsilon_{xx} &= +\frac{S}{E}\left[\cot(\phi) + \nu\tan(\phi)\right] = +\frac{S}{E}K_1(\nu,\phi) \\ \epsilon_{yy} &= -\frac{S}{E}\left[\tan(\phi) + \nu\cot(\phi)\right] = -\frac{S}{E}K_2(\nu,\phi) \\ \gamma_{xy} &= 0. \end{aligned} \tag{2.17}$$

Displacement field

The displacement field is obtained after direct integration of the strains after using the fact that:

$$du_i = \epsilon_{ij}dx_j + \omega_{ij}dx_j$$

and the condition $\omega_{xy} = 0$ also due to symmetries , as follows:

$$u = +\frac{S}{E}K_1(\nu, \phi)x + A$$
$$v = -\frac{S}{E}K_2(\nu, \phi)y + B$$

and where A and B are integration constants.

From the condition $u = 0$ at $x = \ell\cos(\phi)$ we have that $A = -\frac{S}{E}K_1(\nu, \phi)\ell\cos(\phi)$ then it follows that

$$u = \frac{S}{E}K_1(\nu, \phi)(x - \ell\cos(\phi)).$$

Similarly, from the condition $v = 0$ at $y = 0$ we have that $B = 0$ from which

$$v = -\frac{S}{E}K_2(\nu, \phi)y$$

2.2.2 Variational formulation

In this section we formulate the boundary value problem using the approach of the calculus of variations in which the governing PDEs and boundary conditions are obtained after finding the minimum (or maximum) of a functional according to a variational principle[2].

We will see that the weak form (and therefore also the strong form) can be obtained alternatively through the process of finding extreme values for a functional. We will illustrate this idea for the general case of theory of elasticity and then we will present particular examples.

[2]According to Wikipedia [8]

> A variational principle is a scientific principle used within the calculus of variations, which develops general methods for finding functions which minimize or maximize the value of quantities that depend upon those functions. For example, to answer this question: "What is the shape of a chain suspended at both ends?" we can use the variational principle that the shape must minimize the gravitational potential energy.
>
> According to Cornelius Lanczos, any physical law which can be expressed as a variational principle describes an expression which is self-adjoint. These expressions are also called Hermitian. Such an expression describes an invariant under a Hermitian transformation.

Some vague definitions in the calculus of variations

In variational calculus a **functional** can be understood as a "function" having as independent variables or arguments a space of vector functions and producing as a result (or dependent variable) a scalar. For instance, in the particular case of the theory of elasticity such a "function" corresponds to the total potential energy functional Π given by;

$$\Pi(u_i) = \frac{1}{2}\int_V \sigma_{ij}\varepsilon_{ij}\,\mathrm{d}V - \int_V f_i\,u_i\,\mathrm{d}V - \int_S t_i^{(n)}u_i\,\mathrm{d}S \tag{2.18}$$

and where the first term in the left hand side corresponds to the internal strain energy, while the last two terms are the work done by the external body and traction forces. The above functional has as independent variables the displacement vector and its spatial derivatives. This is indicated by the presence of the displacement vector u_i in the expression $\Pi(u_i)$.

In variational calculus we are interested in finding a function u_i that renders the functional Π a maximum or a minimum. In loose terms, the analogous to the differential operator in calculus of functions is now termed the variational operator δ (i.e., δ is analogous to $\frac{\partial}{\partial x_i}$). As such $\delta\Pi$ acts over the function u_i and its derivatives as follows

$$\delta\Pi = \frac{\partial\Pi}{\partial u_i}\,\delta u_i + \frac{\partial\Pi}{\partial\left(\frac{\partial u_i}{\partial x_j}\right)}\delta\left(\frac{\partial u_i}{\partial x_j}\right) + \cdots + \frac{\partial\Pi}{\partial\left(\frac{\partial^n u_i}{\partial x_j\cdots\partial x_k}\right)}\delta\left(\frac{\partial^n u_i}{\partial x_j\cdots\partial x_k}\right) \quad .$$

The following rules apply to the variational operator δ:

- For functionals Π and Φ it follows that

$$\delta(\Pi + \Phi) = \delta\Pi + \delta\Phi$$

- For functionals Π and Φ it follows that

$$\delta(\Pi\Phi) = \delta\Pi\,\Phi + \Pi\,\delta\Phi$$

- For a functional Π and an integer n it follows that

$$\delta(\Pi^n) = n(\Pi^{n-1})\,\delta\Pi$$

- For a functional Π it follows that

$$\delta\int \Pi\,\mathrm{d}x = \int \delta\Pi\,\mathrm{d}x$$

If the variational operator is applied to the functional $\Pi(u_i)$ it produces functions or variations in u_i which are arbitrary and such $\delta u_i \in V$ and $\delta u_i = 0$ in S_u.

To find an extreme function in the calculus of variations we proceed like in differential calculus. Here we compute the first variation of the functional $\delta\Pi$ and solve the variational equation

$$\delta\Pi = 0 \tag{2.19}$$

in the unknown function u_i.

In the particular case of the total potential energy functional Π this yields

$$\int_V \sigma_{ij}\,\delta\varepsilon_{ij}\,\mathrm{d}V - \int_V f_i varu_i\,\mathrm{d}V - \int_{S_t} t_i^{(n)}\,\delta u_i\,\mathrm{d}S = 0 \tag{2.20}$$

where we recognize the weak form of the BVP stated previously. It becomes evident that the functions δu_i in eq. (2.20) play the role of the test functions w_i introduced in the weak form. On the other hand, since we have already shown that the weak and strong forms are equivalent we conclude that having the functional and the essential boundary conditions is equivalent to having the strong form of the problem.

Principle of minimum potential energy

In the theory of elasticity the total potential energy Π is the result of adding the elastic strain energy which is stored in the body upon deformation and the potential energy (work) imparted to the body by the applied forces. The principle states that the body is in equilibrium when this total potential energy reaches a minimum. This is equivalent to stating that an equilibrium configuration is attained when an infinitesimal variation from the position of minimum potential energy involves null changes in energy. This implies the variational condition:

$$\delta\Pi = 0. \tag{2.21}$$

The above principle leads to the so-called principle of virtual displacements stated as follows[3]: "The equilibrium of the body requires that for any compatible small virtual displacements satisfying the condition of being zero at S_u, imposed on the body in its state of equilibrium, the total internal virtual work is equal to the total external virtual work"

$$\int_V \sigma_{ij}\,\delta\varepsilon_{ij}\,\mathrm{d}V - \int_V f_i\delta u_i\,\mathrm{d}V - \int_{S_t} t_i^{(n)}\,\delta u_i\,\mathrm{d}S = 0$$

where δu_i are the virtual displacements and $\delta\varepsilon_{ij}$ are the corresponding virtual strains.

Comparing the virtual work principle with the weak formulation given in eq. (2.11) we identify δu_i with the test functions w_i. As such the PVW takes the form of a powerful tool to test if a body is in equilibrium for a given solution (represented by the trial functions). In what follows we illustrate the use of the principle through some examples corresponding to problems in Bathe's textbook.

[3] see Bathe pp 156

Problem[4]

Establish the differential equation of equilibrium of the problem shown and the boundary conditions. Determine whether the differential operator of the problem is symmetric and positive definite and prove your answer. [1]

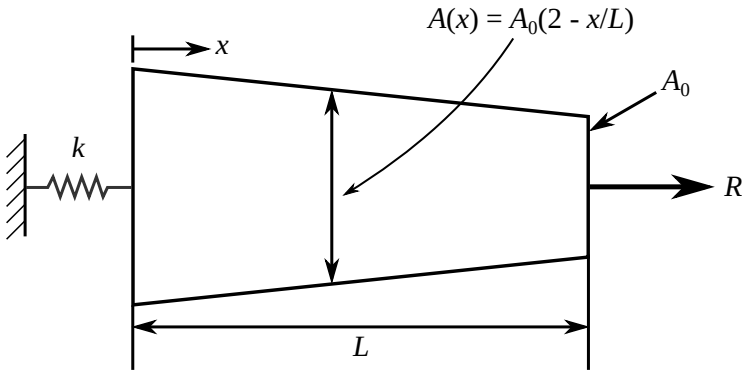


Figure 2.2. Rod with varying cross-sectional area. The Young's modulus is E.

$$\Pi = \frac{1}{2}\int_0^L \sigma_{xx}\varepsilon_{xx}A(x)\,\mathrm{d}x + \frac{1}{2}ku_0^2 - Ru_L$$
$$= \frac{1}{2}\int_0^L EA(x)\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)^2 \mathrm{d}x + \frac{1}{2}ku_0^2 - Ru_L \quad .$$

The first variation for this functional is

$$\delta\Pi = \int_0^L EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}\,\delta u}{\mathrm{d}x}\,\mathrm{d}x + ku_0\,\delta u_0 - R\,\delta u_L \quad .$$

If we integrate by parts, we obtain

$$\delta\Pi = -\int_0^L \frac{\mathrm{d}}{\mathrm{d}x}\left[EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right]\delta u\,\mathrm{d}x + EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\,\delta u\Big|_0^L + ku_0\,\delta u_0 - R\,\delta u_L$$

[4] 3.15 from Finite Element Procedures

from which

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right] = 0$$
$$EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\bigg|_{x=0} = ku_0$$
$$EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\bigg|_{x=L} = R$$

Problem: The Hu-Washizu Variational Principle[5]

Consider the Hu-Washizu functional:

$$\Pi^* = \Pi - \int_V \lambda_{ij}^{\varepsilon}(\varepsilon_{ij} - L_{ijk}u_k)\,\mathrm{d}V - \int_{S_u} \lambda_i^u(u_i^{S_u} - \bar{u}_i)\,\mathrm{d}S \tag{2.22}$$

where

- Π: is the potential energy functional.
- L_{ijk} is a differential operator such $\varepsilon_{ij} = L_{ijk}u_k$.
- S_u surface where essential boundary conditions are prescribed.
- $\lambda_{ij}^{\varepsilon}$ and λ_i^u are Lagrange multipliers.

Using the condition $\delta\Pi = 0$ derive for the interior of the body the equilibrium equations

$$\sigma_{ij,j} + f_i = 0$$

the strain-displacement relationship

$$\varepsilon_{ij} = L_{ijk}u_k$$

and the constitutive equation

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl}$$

and at the surface of the body the relation between the stress tensor and the applied tractions vector at S_t

$$t_i = \sigma_{ij}n_j$$

the relation between the stress tensor and the unknown tractions vector (or reactions) at S_u

$$t_i = \tilde{\sigma_{ij}}n_j$$

[5] 4.35 from Finite Element Procedures

and the essential boundary condition at S_u

$$u_i = \tilde{u}_i$$

We want to determine the so-called Euler equations resulting from the condition $\delta\Pi^* = 0$. Applying the variational operator we have:

$$\begin{aligned}\delta\Pi^* = \delta\Pi &- \int_V \delta\lambda_{ij}^{\varepsilon}\,(\varepsilon_{ij} - L_{ijk}u_k)\,\mathrm{d}V - \int_V \lambda_{ij}^{\varepsilon}(\delta\varepsilon_{ij} - L_{ijk}\,\delta u_k)\,\mathrm{d}V \\ &- \int_V \delta\lambda_i^u\,(u_i^{S_u} - \bar{u}_i)\,\mathrm{d}S - \int_V \lambda_i^u\,\delta u_i^{S_u}\,\mathrm{d}S\end{aligned} \tag{2.23}$$

and

$$\begin{aligned}\delta\Pi^* = &\int_V C_{ijkl}\varepsilon_{kl}\,\delta\varepsilon_{ij}\,\mathrm{d}V - \int_{S_t} t_i\,\delta u_i\,\mathrm{d}S - \int_V f_i\,\delta u_i\,\mathrm{d}V - \int_V \lambda_{ij}^{\varepsilon}\,\delta\varepsilon_{ij}\,\mathrm{d}V + \int_V \lambda_{ij}^{\varepsilon}L_{ijk}\,\delta u_k\,\mathrm{d}V \\ &- \int_V \delta\lambda_{ij}^{\varepsilon}(\varepsilon_{ij} - L_{ijk}u_k)\,\mathrm{d}V - \int_S \delta\lambda_i^u\,(u_i^{S_u} - \bar{u}_i)\,\mathrm{d}S - \int_{S_u} \lambda_i^u\,\delta u_i\,\mathrm{d}S = 0\end{aligned} \tag{2.24}$$

using

$$\int_V (\lambda_{ij}^{\varepsilon}\,\delta u_i)_{,j}\,\mathrm{d}V = \int_V \lambda_{ij}^{\varepsilon}\,\delta u_{i,j}\,\mathrm{d}V + \int_V \lambda_{ij,j}^{\varepsilon}\,\delta u_i\,\mathrm{d}V$$

In the above we can write

$$\begin{aligned}\int_V \lambda_{ij}^{\varepsilon}L_{ijk}\,\delta u_k\,\mathrm{d}V &= \int_V (\lambda_{ij}^{\varepsilon}\,\delta u_i)_{,j}\,\mathrm{d}V - \int_V \lambda_{ij,j}^{\varepsilon}\delta u_i\,\mathrm{d}V \\ &= \int_{S_t} \lambda_{ij}^{\varepsilon}\,\delta u_i\,\hat{n}_j\,\mathrm{d}S - \int_V \lambda_{ij,j}^{\varepsilon}\,\delta u_i\,\mathrm{d}V\end{aligned}$$

therefore

$$\begin{aligned}\delta\Pi^* &= \int_V (C_{ijkl}\varepsilon_{kl} - \lambda_{ij}^{\varepsilon})\,\delta\varepsilon_{ij}\,\mathrm{d}V - \int_{S_t} t_i\,\delta u_i\,\mathrm{d}S - \int_V f_i\,\delta u_i\,\mathrm{d}V + \int_{S_t} \lambda_{ij}^{\varepsilon}\,\delta u_i\,\hat{n}_j\,\mathrm{d}S - \int_V \lambda_{ij,j}^{\varepsilon}\delta u_i\,\mathrm{d}V \\ &- \int_V \delta\lambda_{ij}^{\varepsilon}(\varepsilon_{ij} - L_{ijk}u_k)\,\mathrm{d}V - \int_{S_u} \delta\lambda_i^u\,(u_i^{S_u} - \bar{u}_i)\,\mathrm{d}S - \int_{S_u} \lambda_i^u\,\delta u_i\,\mathrm{d}S = 0 \\ &= \int_V (C_{ijkl}\varepsilon_{kl} - \lambda_{ij}^{\varepsilon})\,\delta\varepsilon_{ij}\,\mathrm{d}V + \int_{S_t} (\lambda_{ij}^{\varepsilon}\hat{n}_j - t_i)\delta u_i\,\mathrm{d}S \\ &- \int_V (\lambda_{ij,j}^{\varepsilon} + f_i)\,\delta u_i\,\mathrm{d}V - \int_V (\varepsilon_{ij} - L_{ijk}u_k)\,\delta\lambda_{ij}^{\varepsilon}\,\mathrm{d}V - \int_{S_u} (u_i^{S_u} - \bar{u}_i)\,\delta\lambda_i^u\,\mathrm{d}S - \cancel{\int_{S_u} \lambda_i^u\,\delta u_i^{S_u}\,\mathrm{d}S} = 0\end{aligned}$$

Now, imposing the conditions $\delta\varepsilon_{ij} \neq 0$, $\delta\lambda^{\varepsilon}_{ij} \neq 0$, $varu_i \neq 0$ in S_t, $\delta u_i \neq 0$ in V and $\delta\lambda^{u}_{i} \neq 0$ in S_u we have

$$\lambda^{\varepsilon}_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2.25}$$
$$\varepsilon_{ij} = L_{ijk}u_k \tag{2.26}$$
$$t_i = \lambda^{\varepsilon}_{ij}\hat{n}_j \tag{2.27}$$
$$\lambda^{\varepsilon}_{ij,j} + f_i = 0 \tag{2.28}$$
$$u_i^{S_u} = \bar{u}_i \ . \tag{2.29}$$

2.2.3 Weighted residual methods

This section introduces the concept of residual or difference from zero in a differential equation once its solution is aproximated numerically. For that purpose we will take as prototype equation the one obtained as our general model of BVP (see eq. (2.1)) and recalled here for completeness:

$$\rho(\mathbf{x})\frac{\partial u(\mathbf{x},t)}{\partial t} + \mathcal{L}u(\mathbf{x},t) = \rho(\mathbf{x})F(\mathbf{x},t) \ . \tag{2.30}$$

We will assume that the actual solution to the generalized BVP given by eq. (2.30) is approximated by the numerically derived solution $\tilde{u}(\mathbf{x})$ through a superposition like

$$\tilde{u}(\mathbf{x}) = N^I(\mathbf{x})u^I \tag{2.31}$$

where $N^I(\mathbf{x})$ are interpolating functions and I denotes a superposition index varying like $I = 1, 2, ..., K$ with K being the number of points where the solution is known. In what follows we will use $u(\mathbf{x})$ instead of $\tilde{u}(\mathbf{x})$ but will keep in mind that we are actually using the approximation given by eq. (2.31). Similarly, in order to keep the discussion simple for the time being we will drop the time effects reducing the generailzed PDE to the simple form:

$$\mathcal{L}u(\mathbf{x}) = \rho(\mathbf{x})F(\mathbf{x}) \ . \tag{2.32}$$

Now, since we are using the approximation given by eq. (2.31) this equation is not strictly satisifed but instead we will have the following "unbalanced" condition

$$Lu(\mathbf{x}) - \rho(\mathbf{x})F(\mathbf{x}) \equiv R \neq 0$$

where the term R corresponds to a residual error which is to be distributed throughout the solution domain. The so-called weighted residual methods differ in the form in which they distribute

the residual between the different K points conforming the computational domain.

Using eq. (2.31) in eq. (2.30) and the linearity in the differential operator yields;

$$R = \mathcal{L}(N^P)u^P - \rho F.$$

In what follows we will consider different strategies to distribute or weight the residual R over the computational domain.

Galerkin method

In the Galerkin scheme the interpolation functions are used also as weighting functions leading to:

$$\int_V N^Q R \, \mathrm{d}V = 0$$

or explicitly

$$\int_V N^Q \mathcal{L}(N^P) \, \mathrm{d}V \, u^P = \int_V N^Q \rho F \, \mathrm{d}V \tag{2.33}$$

Imposing eq. (2.33) in the K points conforming the computational domain or equivalently ranging Q from 1 to K leads to the following system of algebraic equation

$$K^{QP} U^P = f^Q \tag{2.34}$$

where U^P is a vector that stores the point values of the function u along te K points of the computational domain, while f^Q stores the corresponding point excitations.

Least squares method

In this method the integral of the square of the residual is minimized with respect to the K point parameters or nodal values of the function. Accordingly,

$$\frac{\partial}{\partial u^I}\int_V R^2\,\mathrm{d}V = 0$$
$$\frac{\partial}{\partial u^I}\int_V [\mathcal{L}(N^P u^P) - \rho F]^2\,\mathrm{d}V = 0$$
$$\int_V [\mathcal{L}N^P u^P - \rho F]\mathcal{L}(N^I)\,\mathrm{d}V = 0$$
$$\int_V \mathcal{L}(N^I)\mathcal{L}(N^P)\,\mathrm{d}V\,u^P - \rho\int_V \mathcal{L}(N^I)F\,\mathrm{d}V = 0$$

which can be written like

$$K^{IP}U^P = f^I \tag{2.35}$$

Collocation method

Zero values of the residual are directly imposed over the K points of the computational domain;

$$\mathcal{L}(N^I)u^I - \rho F = 0$$

$$L[N^I(x^J)]u^I - \rho F(x^J) = 0$$

where J ranges between 1 and K leading to the system;

$$K^{IP}U^P = f^I \tag{2.36}$$

Subdomain method

The zero value of the residual is imposed upon K subdomains.

$$\int_{V^I} \mathcal{L}(N^P)dV^I u^P - \rho\int_{V^I} F dV^I = 0 \qquad I = 1, ..., K.$$

For instance, for the N-th element it follows that ;

$$\int_{V^N} \mathcal{L}(N^P) dV^N u^P - \rho \int_{V^N} F dV^N = 0 \qquad P = 1, ..., K.$$

Applying the equation over the K subdomains leads to the discrete system;

$$K^{IP} U^P = f^I \qquad I = 1, ..., K. \tag{2.37}$$

Ritz method

It operates directly upon the variational statement of the problem. For a given functional

$$\Pi = \Pi(N^Q u^Q)$$

the variational equation reads

$$\delta\Pi \equiv \frac{\partial \Pi}{\partial u^Q} \delta u^Q = 0$$

from which

$$\frac{\partial \Pi}{\partial u^Q} = 0.$$

Problem: Discretization of the generalized parabolic equation.

Let us consider the case of the generalized parabolic equation and its discretization following the Galerkin method

$$\rho(\mathbf{x}) \frac{\partial u(\mathbf{x}, t)}{\partial t} + \mathcal{L} u(\mathbf{x}, t) = 0$$

which can also be written using indicial notation

$$\frac{\partial}{\partial x_i} \left[p(x) \frac{\partial u}{\partial x_i} \right] + q(x) u + \rho \frac{\partial u}{\partial t} = \rho F.$$

Assuming for simplicity that $p(x) = 1$ yields

$$-\int_V N^P N^Q_{,ii} dV u^Q + \int_V q N^P N^Q dV u^Q + \rho \int_V N^P N^Q dV v^Q - \rho \int_V N^P F dV = 0$$

$$\int_V N^P_{,i} N^Q_{,i} dV u^Q - \int_S N^P N^Q_{,i} \hat{n}_i dS u^Q + \int_V q N^P N^Q dV u^Q + \rho \int_V N^P N^Q dV v^Q - \rho \int_V N^P F dV = 0$$

$$\int_V \left(N^P_{,i} N^Q_{,i} + q N^P N^Q \right) dV u^Q + \rho \int_V N^P N^Q dV V^Q = \int_S N^P N^Q_{,i} \hat{n}_i dS u^Q + \rho \int_V N^P F dV$$

which can be written in discrete form;

$$K^{PQ} U^Q + C^{PQ} V^Q = f^p$$

Problem: Discretization of Navier equations.

In this case the differential equations are written as

$$(\lambda + \mu) u_{j,ij} + \mu u_{i,jj} + f_i = 0 \ .$$

We can write the differential operator as

$$L_{ij} \equiv (\lambda + \mu) \frac{\partial^2}{\partial x_i \partial x_j} + \mu \frac{\partial^2}{\partial x_k \partial x_k} \delta_{ij}$$

$$\begin{aligned}
&r_i = -f_i \\
&u_i = N^Q_i u^Q \\
&\mathcal{L}_{ij}(u_j) \equiv (\lambda + \mu)(N^Q_j u^Q)_{,ij} + \mu (N^Q_j u^Q)_{,kk} \delta_{ij} \\
&\mathcal{L}_{ij}(u_j) \equiv (\lambda + \mu) N^Q_{j,ij} u^Q + \mu N^Q_{i,kk} u^Q \\
&\mathcal{L}_{ij}(u_j) \equiv \mathcal{L}_{ij}(N^Q_j) u^Q \ .
\end{aligned}$$

In the Galerkin scheme we use the trial function as weighting function.

$$R_i \equiv L_{ij}(N^Q_j) u^Q + f_i$$

and we state ...

$$\int_V N^P_i R_i dV = 0 \qquad P = 1, 2, ..., N.$$

$$\int_V N^P_i \mathcal{L}_{ij}(N^K_j) dV u^K + \int_V N^P_i f_i dV = 0$$

$$(\lambda+\mu)\int_V N_i^P N_{j,ij}^K dV u^K + \mu\int_V N_i^P N_{i,kk}^K dV u^K + \int_V N_i^P f_i dV = 0$$

integrating by parts;

$$-(\lambda+\mu)\int_V N_{i,j}^P N_{j,i}^K dV u^K + (\lambda+\mu)\int_S N_i^P N_{j,i}^K \hat{n}_j dS u^K - \mu\int_V N_{i,k}^P N_{i,k}^K dV u^K$$
$$+\mu\int_S N_i^P N_{i,k}^K \hat{n}_k dS u^K + \int_V N_i^P f_i dV = 0$$

which can be written like;

$$K^{PQ}U^Q = F^P$$

where

$$K^{PQ} = (\lambda+\mu)\int_V N_{i,j}^P N_{j,i}^Q dV + \mu\int_V N_{i,k}^P N_{i,k}^Q dV$$

$$F^P = \int_S N_i^P t_i^{(\hat{n})} dS + \int_V N_i^P f_i dV = 0$$

Problem: Discretization of the acoustic wave equation.

$$\vec{\nabla}\cdot\left[\frac{1}{\rho}\vec{\nabla}p(\mathbf{x})\right] - \frac{\partial}{\partial t}\left(\frac{1}{\lambda}\frac{\partial\rho}{\partial t}\right) - q(\mathbf{x}) = 0$$

where we recognize;

$$\mathcal{L}() \equiv \vec{\nabla}\cdot\left(\frac{1}{\rho}\vec{\nabla}\right) - \frac{\partial}{\partial t}\left(\frac{1}{\lambda}\frac{\partial}{\partial t}\right)$$

Let

$$p(x) = N^K p^K$$

then

$$\mathcal{L}(p) \equiv \vec{\nabla} \cdot \left(\frac{1}{\rho} \vec{\nabla} N^K p^K \right) - \frac{\partial}{\partial t} \left(\frac{1}{\lambda} \frac{\partial N^K p^K}{\partial t} \right)$$

or in indicial notation

$$\mathcal{L}(p) \equiv \left(\frac{1}{\rho} N^K_{,i} \right)_{,i} p^K - \frac{\partial}{\partial t} \left(\frac{1}{\lambda} \frac{\partial N^K}{\partial t} \right) p^K$$

which is equivalent to having;

$$\mathcal{L}(p) \equiv \mathcal{L}(N^K) p^K$$

using the trial functions as weighting functios and recalling the definition of the residual which in this case reads;

$$R = \mathcal{L}(N^K) p^K - q$$

yields;

$$\int_V N^J R dV = 0 \qquad J = 1, 2, ..., K$$

$$\int_V N^J L(N^K) dV p^K - \int_V N^J q dV = 0$$

$$\int_V N^J \left(\frac{1}{\rho} N^K_{,i} \right)_{,i} dV p^K - \int_V N^J \frac{\partial}{\partial t} \left(\frac{1}{\lambda} \frac{\partial N^K}{\partial t} \right) dV p^K - \int_V N^J q dV = 0$$

Integrating by parts the first term on the R.H.S gives us;

$$-\int_V N^J_{,i}\frac{1}{\rho}N^K_{,i}dVp^K+\int_S N^J\frac{1}{\rho}N^K_{,i}\hat{n}_i dSp^K=\int_V N^J\frac{\partial}{\partial t}\left(\frac{1}{\lambda}\frac{\partial N^K}{\partial t}\right)dVp^K+\int_V N^J qdV$$

$$\int_V N^J_{,i}\frac{1}{\rho}N^K_{,i}dVp^K+\int_V N^J\frac{1}{\lambda}N^K dV\ddot{p}^K=\int_S N^J\frac{1}{\rho}N^K_{,i}\hat{n}_i dSp^K+\int_V N^J qdV$$

$$K^{JK}P^K+M^{JK}\ddot{P}^K+f^J=0$$

2.3 Discretization of the PVW via the FEM

A simple spring-mass system

The simple problem of a spring-mass system considered next resembles most of the algorithmic aspects of a finite element code with the advantage that the problem is already a discrete mechanical system. The problem consists of an assemblage of masses joined by different springs submitted to time varying loads. Each spring plays the role of a finite element and each mass is analogous to a nodal point in a finite element algorithm. For instance the full system may be like the one shown in fig. 2.3

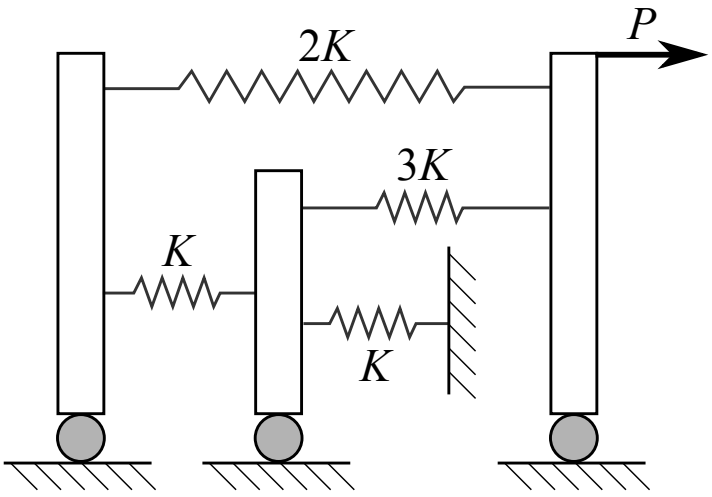


Figure 2.3. Typical assemblage of springs and masses.

Consider a typical spring (finite element) like the one shown in fig. 2.4

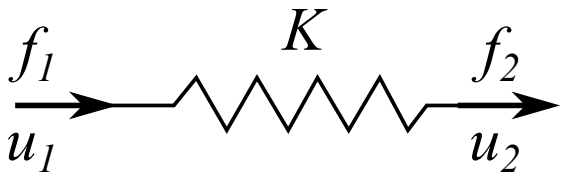


Figure 2.4. Typical spring element.

The relation between the force and the relative displacement can be written like

$$f_1 = K(u_1 - u_2)$$

and from equilibrium we have

$$f_1 + f_2 = 0$$

which yields the following force-displacement relationship for a typical spring element:

$$\begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = K \begin{bmatrix} 1.0 & -1.0 \\ -1.0 & 1.0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{2.38}$$

On the other hand, the equilibrium equation for a typical mass with displacement u_j (see fig. 2.5) and attached to springs i and $i+1$ reads

$$f_2^i + f_1^{i+1} + m_j \frac{\mathrm{d}V_j}{\mathrm{d}t} = P_j. \tag{2.39}$$

which can be written in terms of displacements using eq. (2.38) like

$$(K^i + K^{i+1})u_j - K^i u_{j-1} - K^{i+1} u_{j+1} + m_j \frac{\mathrm{d}V_j}{\mathrm{d}t} = P_j.$$

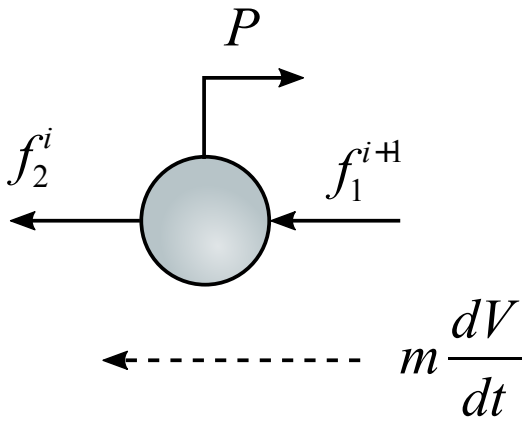


Figure 2.5. Free body diagram for a typical mass connected to springs i and $i+1$.

Writing the elemental equilibrium equations in terms of u_{j-1}, u_j and u_{j+1} we have:

$$\left\{ \begin{array}{c} f_1^i \\ f_2^i \end{array} \right\} = \left[\begin{array}{cc} k_{11}^i & k_{12}^i \\ k_{21}^i & k_{22}^i \end{array} \right] \left\{ \begin{array}{c} u_{j-1} \\ u_j \end{array} \right\}$$

and

$$\left\{ \begin{array}{c} f_1^{i+1} \\ f_2^{i+1} \end{array} \right\} = \left[\begin{array}{cc} k_{11}^{i+1} & k_{12}^{i+1} \\ k_{21}^{i+1} & k_{22}^{i+1} \end{array} \right] \left\{ \begin{array}{c} u_j \\ u_{j+1} \end{array} \right\}$$

which gives for the equilibrium equation of the m_j mass:

$$k_{21}^i u_{j-1} + (k_{22}^i + k_{11}^{i+1}) u_j + k_{12}^{i+1} u_{j+1} + m_j \frac{dV_j}{dt} = P_j.$$

Considering also the contributions from the springs K^i and K^{i+1} to the equilibrium of masses m_{j-1} and m_{j+1} respectively we have the following matrix block:

$$\left[\begin{array}{ccc} & & \\ k_{11}^i & k_{12}^i & \\ k_{21}^i & k_{22}^i + k_{11}^{i+1} & k_{12}^{i+1} \\ & k_{21}^{i+1} & k_{22}^{i+1} \end{array} \right]$$

Considering now the complete system of masses and springs leads to a system of linear equations of the form

$$[K_G] \{U_G\} + [M] \{A_G\} = \{F_G\}. \tag{2.40}$$

where each equation represents the equilibrium of a given mass. The coefficient matrix in eq. (2.40) can be assembled in a systematic way by establishing the connection between the global and local degrees of freedom. This can be accomplished through an operator storing in each row the global degrees of freedom corresponding to each element. For instance, fig. 2.6 shows elements K^i and K^{i+1} and the global degrees of freedom corresponding to masses m_{j-1}, m_j and m_{j+1}. The corresponding entries of the DME operator for these elements are given by:

$$DME = \left[\begin{array}{cc} j-1 & j \\ j & j+1 \\ & \end{array} \right]$$

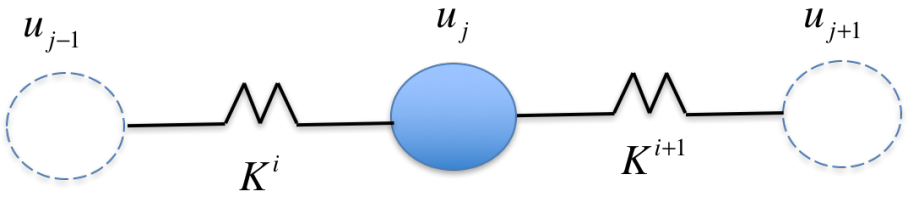


Figure 2.6. Global degrees of freedom connected to the spring elements K^i and K^{i+1} respectively.

and the contribution from these elements to the global coefficient matrix for elements i and $i+1$ reads respectively:

$$
\begin{aligned}
K_{j-1,j-1} &\leftarrow K_{j-1,j-1} + k^i_{11} \\
K_{j-1,j} &\leftarrow K_{j-1,j} + k^i_{12} \\
K_{j,j-1} &\leftarrow K_{j,j-1} + k^i_{21} \\
K_{j,j} &\leftarrow K_{j,j} + k^i_{22}
\end{aligned}
$$

and

$$
\begin{aligned}
K_{j,j} &\leftarrow K_{j,j} + k^{i+1}_{11} \\
K_{j,j+1} &\leftarrow K_{j,j+1} + k^{i+1}_{12} \\
K_{j+1,j} &\leftarrow K_{j+1,j} + k^{i+1}_{21} \\
K_{j+1,j+1} &\leftarrow K_{j+1,j+1} + k^{i+1}_{22}
\end{aligned}
$$

The system given by eq. (2.40), assembled with the aid of the DME operator can be solved for the global displacements U_G. The pseudo-code shown in line 1 presents all the steps required to solve the problem in the context of the finite element method. In that code the so-called DME operator is an equation assembly array indicating how each element contributes to the global stiffness and mass matrix.

Algorithm 1: Springs Algorithm.

Data: Problem parameters; NUMNP, NUMEL, NMATP
Result: Displacements and spring forces
Create DME operator;
Assemble K^G, F^G;
while $j \leq 1, NUMEL$ **do**
  $K^G \leftarrow K^G + K^i$
  $F^G \leftarrow F^G + F^i$
end
Solve $[K^G]U = F^G$
Find internal forces

```
"""
Computes the displacement solution for an assembly
of 1D spring elements under point loads.
"""
import numpy as np
import starter as sta
import assembler as ass
from datetime import datetime
start_time = datetime.now()
"""
   PRE-PROCESSING
"""
nodes , mats , elements , loads = sta.readin()
ne , nn , nm , nl = sta.proini(nodes , mats , elements , loads)
DME , IBC , neq = ass.DME(nn , ne , nodes , elements)
"""
   ASSEMBLY
"""
KG = np.zeros([neq, neq])
for i in range(ne):
    kloc , ndof  = ass.retriever(elements , mats  , nodes , i)
    KG = ass.assembler(KG , neq , kloc , ndof , DME , i)
RHSG = ass.loadasem(loads, IBC, neq, nl)
"""
   SYSTEM SOLUTION
"""
UG = np.linalg.solve(KG, RHSG)
if not(np.allclose(np.dot(KG, UG), RHSG)):
    print("The system is not in equilibrium!")
end_time = datetime.now()
print('Duration for system solution: {}'.format(end_time - start_time))
```

2.3.1 Basic elements of interpolation theory

Let $f(x)$ be a function whose values are known at n discrete points $x_1, x_2, ..., x_n$. We want to know (interpolate) the value of $f(x)$ at an arbitrary point $x \in [x_1, x_n]$.

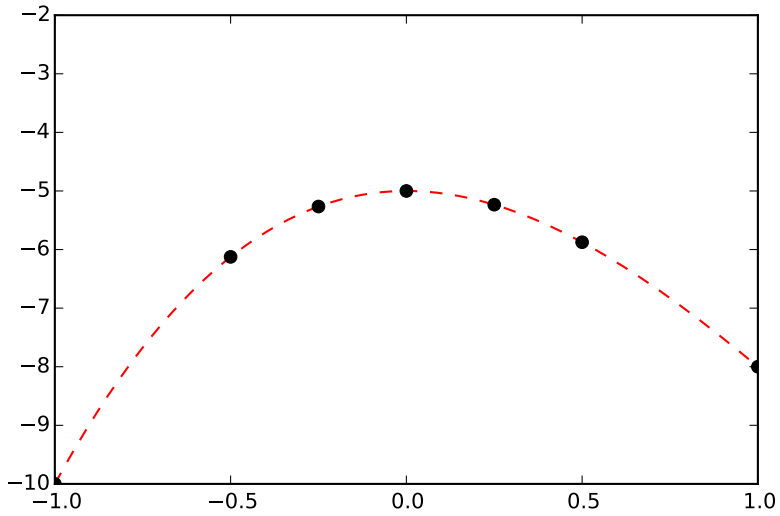


Figure 2.7. Global interpolation of a function

The process of interpolation or computation of the unknown value of $f(x)$ using the known values $\{f^1, f^2, ..., f^n\}$ involves two steps:

i Fitting an interpolating function to the known data points.

ii Evaluating the function at the arbitrary point.

We can (i) use all the n-data points and fit an $(n-1)$-th order polynomial (which is cumbersome and difficult to code) or (ii) split the domain in sub-intervals and use local polynomials within each sub-interval. This last approach involves only a couple of polynomials and it is easy to code, however it may have some continuity issues.

In finite element analysis local interpolation is used in order to proceed systematically. Local interpolation uses a finite number of nearest-neighbors and generates interpolated values $f(x)$ that do not in general have continuous first or higher derivatives.

Lagrange interpolation theorem

Given a set of n-points $\{(x^1, y^1), \cdots, (x^n, y^n)\}$ where $y^n \equiv f(x^n)$ then: "there exists a unique polynomial $p(x)$ of order at most $(n-1)$ such $p(x^I) = f(x^I)$ for $I = 1, 2, \cdots, n$". The polynomial

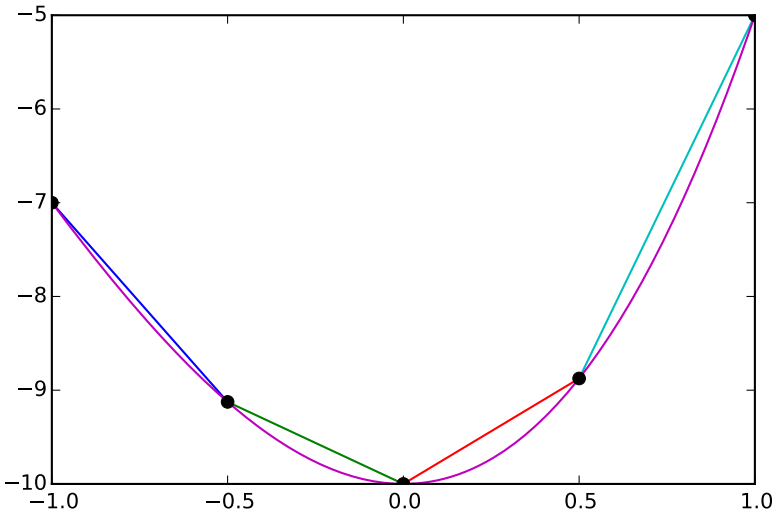


Figure 2.8. Local or piecewise interpolation of a function

is given by;

$$p(x^I) = L^I(x) f(x^I) \tag{2.41}$$

for $I = 1, 2, ..., n$ where

$$L^I(x) = \prod_{\substack{J=1 \\ I \neq J}}^{n} \frac{(x - x^J)}{(x^I - x^J)} \tag{2.42}$$

and where it should be noticed that

$$L^I(x^J) = \delta^{IJ}.$$

Example for n=3

Consider the domain $[-1, 1]$ and the data points at $x^1 = -1.0$, $x^2 = +1.0$ and $x^3 = 0.0$. We have

$$L^1(x) = \frac{(x - x^2)(x - x^3)}{(x^1 - x^2)(x^1 - x^3)} \equiv -\frac{1}{2}(1-)x$$
$$L^2(x) = \frac{(x - x^1)(x - x^3)}{(x^2 - x^1)(x^2 - x^3)} \equiv +\frac{1}{2}(1 + x)x$$

and

$$L^3(x) = \frac{(x - x^1)(x - x^2)}{(x^3 - x^1)(x^3 - x^2)} \equiv 1 - x^2.$$

The resulting interpolating polynomials $L^I(x)$ and the interpolating function are shown in fig. 2.9 below

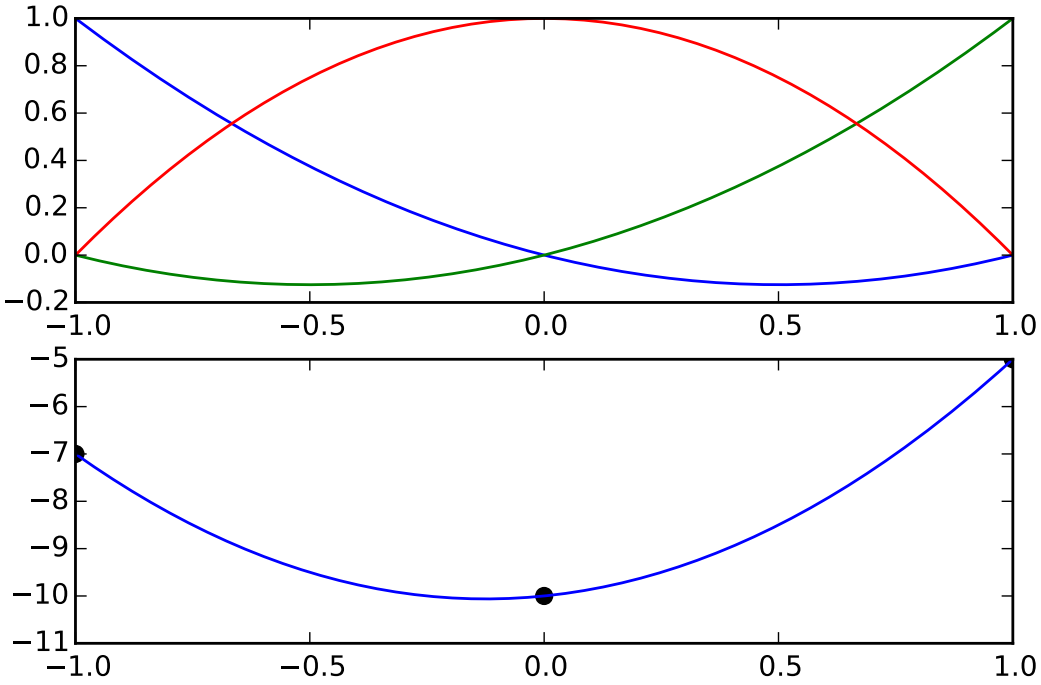


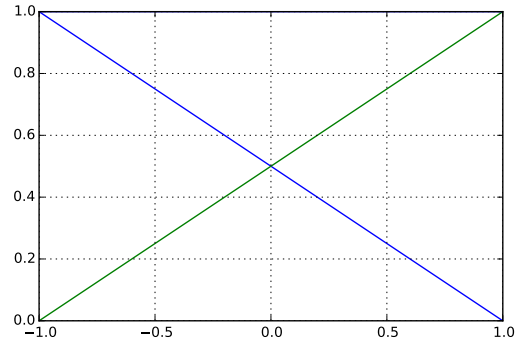
Figure 2.9. Interpolating polynomials and the resulting interpolating function

Example: Interpolation of a function using a global and a local scheme

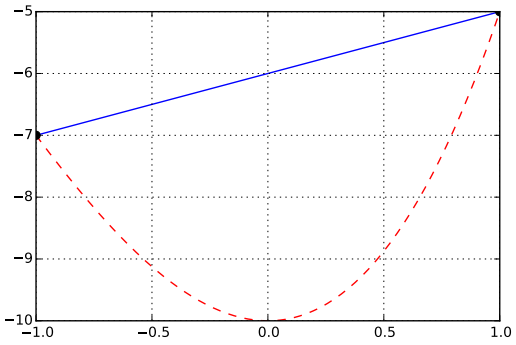
Assume we have known values of the function:

$$f(x) = x^3 + 4x^2 - 10$$

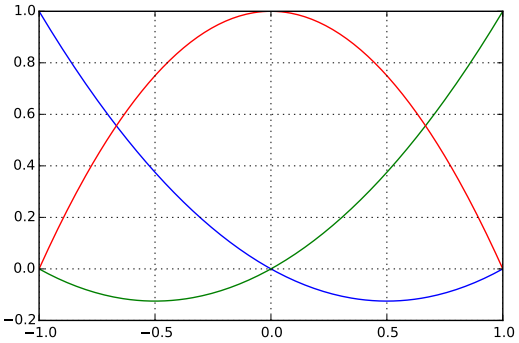
in the interval $[-1.0, 1.0]$ and we wish to obtain an interpolated version of the function using different schemes.



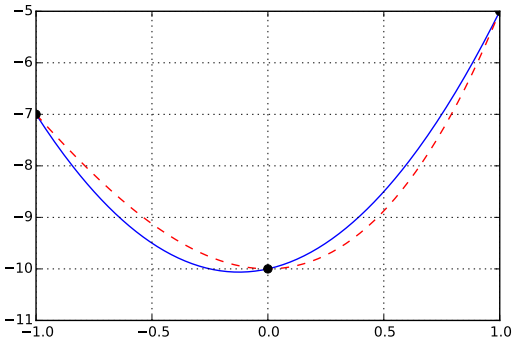
(a) Interpolation polynomials.



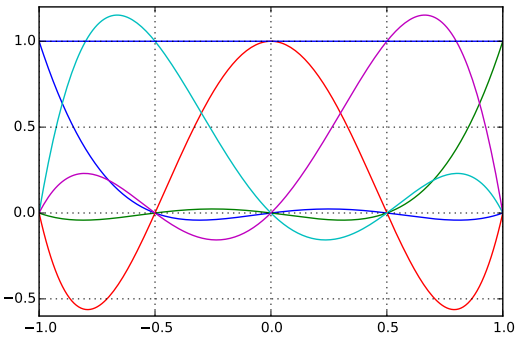
(b) Actual and interpolated function.



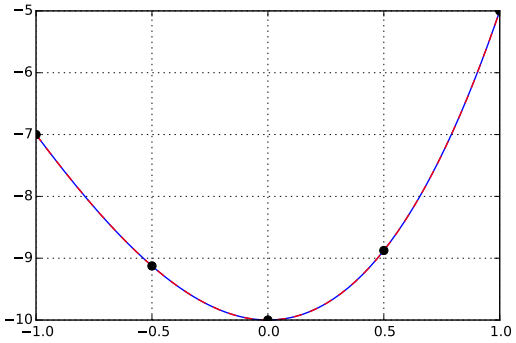
(c) Interpolation polynomials.



(d) Actual and interpolated function.



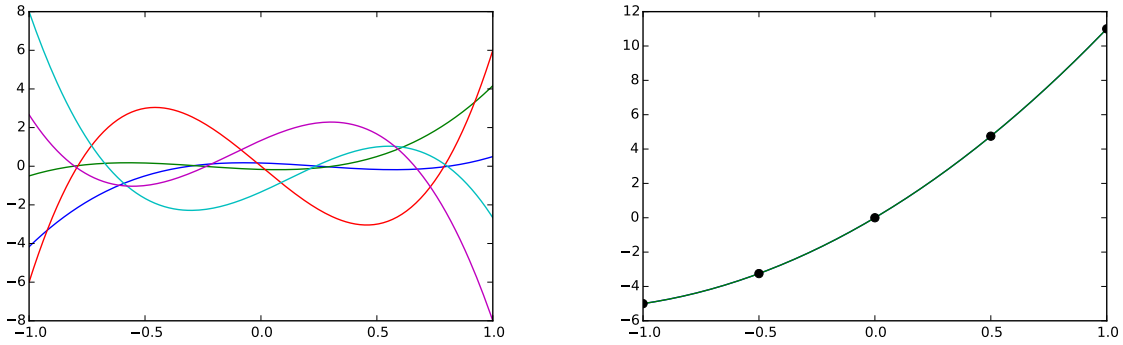
(e) Interpolation polynomials.



(f) Actual and interpolated function.

Figure 2.10. Lineal interpolation of the function $f(x) = x^3 + 4x^2 - 10$.

Figure 2.10 shows the interpolating polynomials and the resulting function for the case of first, second and fourth order polynomials respectively. Similarly, the first order derivative obtained out of the interpolated function is displayed in fig. 2.11.



(a) First order derivatives of the interpolation polynomials.

(b) Interpolated first order derivative of the function.

Figure 2.11. Lineal interpolation of the function $f(x) = x^3 + 4x^2 - 10$.

We now proceed like in the finite element method and use the local first order polynomials shown in fig. 2.12 to interpolate the function under study.

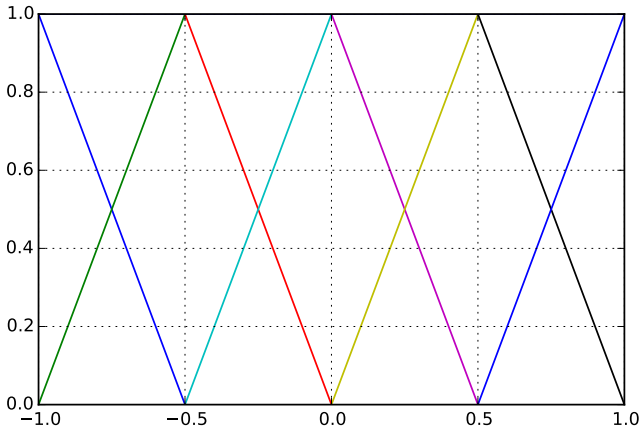
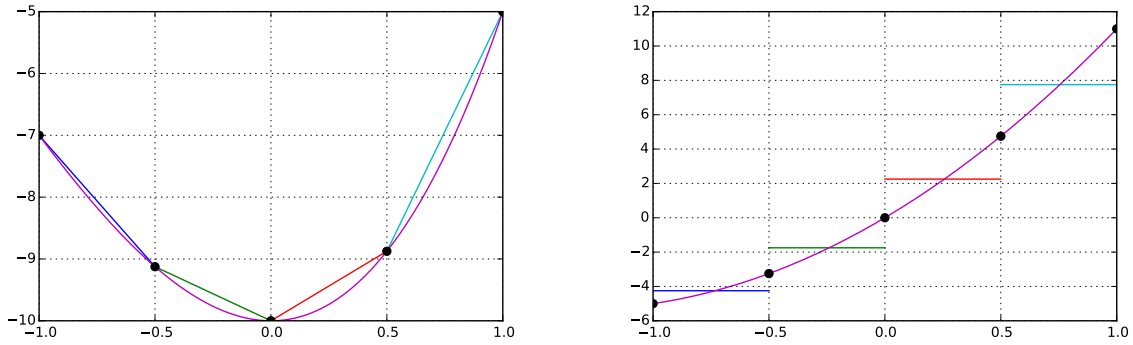


Figure 2.12. Local interpolating polynomials

The resulting function and its numerically obtained first order derivateive are shown in fig. 2.13. It is clear how the local interpolation destroys the global continuity in the first order derivative while the function itself remains continous.



(a) Function interpolated with local first order polynomials.

(b) Interpolated first order derivative of the function.

Figure 2.13. Lineal interpolation of the function $f(x) = x^3 + 4x^2 - 10$.

Extension to 2D domains

Assume we are now interested in conducting interpolation of a function over a spatial 2-dimensional domain where every point is specified by a position vector of the form $\mathbf{x} = x\hat{\imath} + y\hat{\jmath}$. We want to know, via interpolation, the value of a function $f(\mathbf{x})$ at an arbitrary point $\mathbf{x}$ provided we know the set of n-points $\{(\mathbf{x}^1, f^1), \cdots, (\mathbf{x}^n, f^n)\}$.

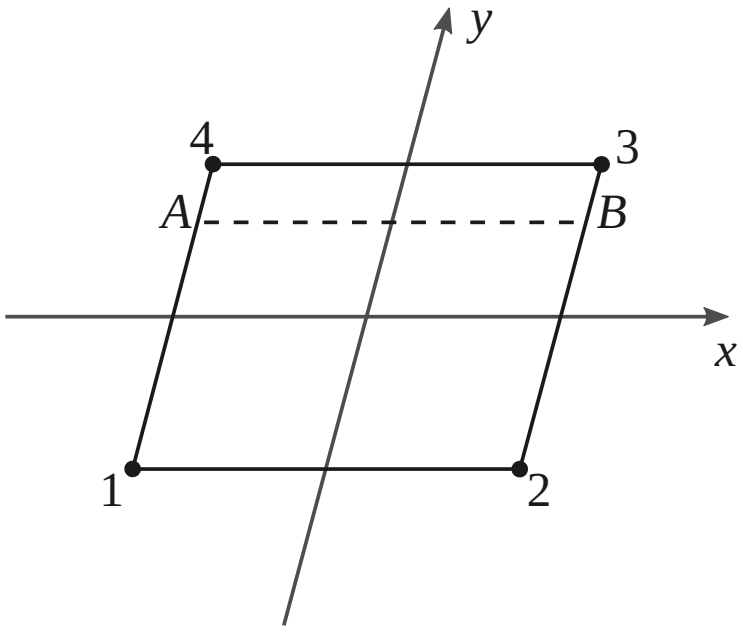


Figure 2.14. Basic square domain

We first fix $x = x^A$ and conduct 1-dimensional interpolation along the y direction as discussed in the previous section as follows (see fig. 2.15)

$$f(x^A, y) = L^1(y)f^1 + L^4(y)f^4$$

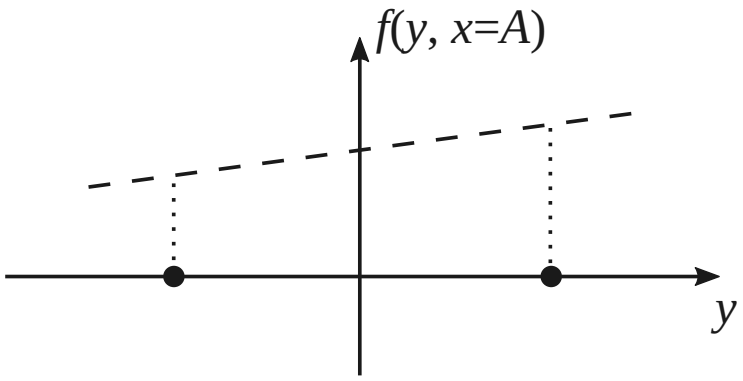


Figure 2.15. Interpolation along the y-direction

Similarly, we can fix $x = x^B$ and interpolate once again along the y direction

$$f(x^B, y) = L^2(y)f^2 + L^3(y)f^3.$$

We now conduct the interpolation along the x-direction using the functions $f(x^A, y)$ and $f(x^B, y)$ respectively as follows

$$\begin{aligned}
f(x,y) &= L^A(x)f(x^A,y) + L^B(x)f(x^B,y) \\
f(x,y) &= L^A(x)\{L^1(y)f^1 + L^4(y)f^4\} + L^B(x)\{L^2(y)f^2 + L^3(y)f^3\} \\
f(x,y) &= L^A(x)L^1(y)f^1 + L^A(x)L^4(y)f^4 + L^B(x)L^2(y)f^2 + L^B(x)L^3(y)f^3 \ ,
\end{aligned}$$

where

$$\begin{aligned}
L^A(x) &\equiv L^1(x) \\
L^B(x) &\equiv L^2(x) \\
L^1(y) &\equiv L^1(y) \\
L^2(y) &\equiv L^1(y) \\
L^3(y) &\equiv L^2(y) \\
L^4(y) &\equiv L^2(x) \ .
\end{aligned}$$

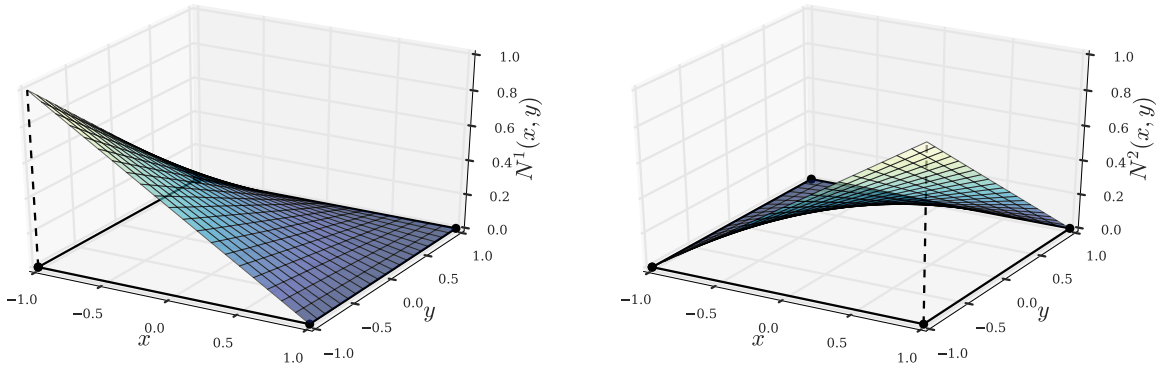
The function of two variables is then written as the product of one-dimensional interpolations

$$f(x,y) = N^1(x,y)f^1 + N^2(x,y)f^2 + N^3(x,y)f^3 + N^4(x,y)f^4$$

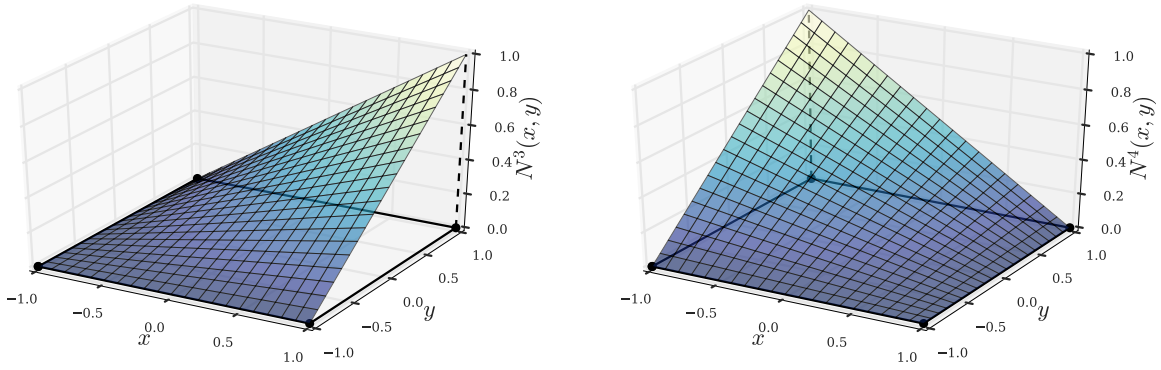
with

$$
\begin{aligned}
N^1(x,y) &= L^1(x)L^1(y) \\
N^2(x,y) &= L^2(x)L^1(y) \\
N^3(x,y) &= L^2(x)L^2(y) \\
N^4(x,y) &= L^1(x)L^2(y) \quad .
\end{aligned}
$$

See Figure 2.16 for the shape functions of a 4-nodes element.



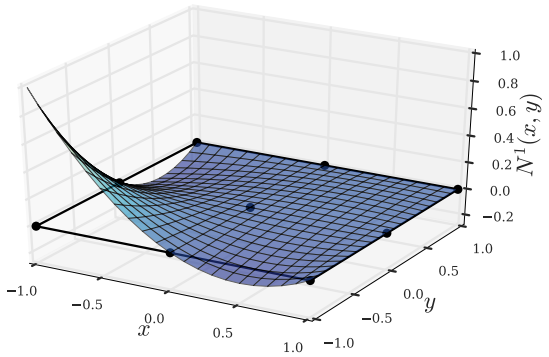
(a) Shape function $N^1(x,y) = \frac{1}{4}(1-x)(1-y)$. **(b)** Shape function $N^2(x,y) = \frac{1}{4}(1+x)(1-y)$.



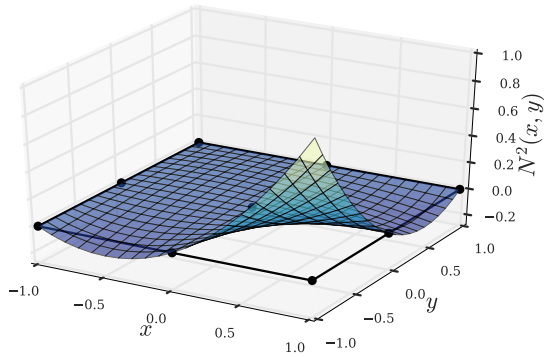
(c) Shape function $N^3(x,y) = \frac{1}{4}(1+x)(1+y)$. **(d)** Shape function $N^4(x,y) = \frac{1}{4}(1-x)(1+y)$.

Figure 2.16. Shape functions for a 4-nodes element.

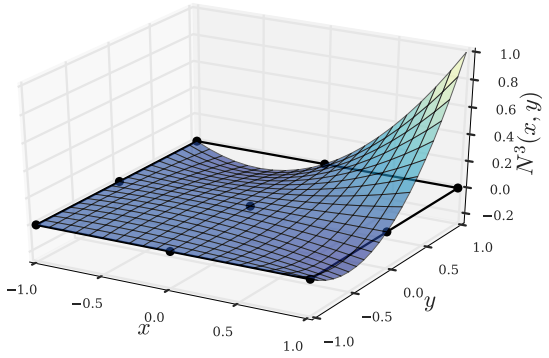
See Figure 2.17 for the shape functions of a 9-nodes element.



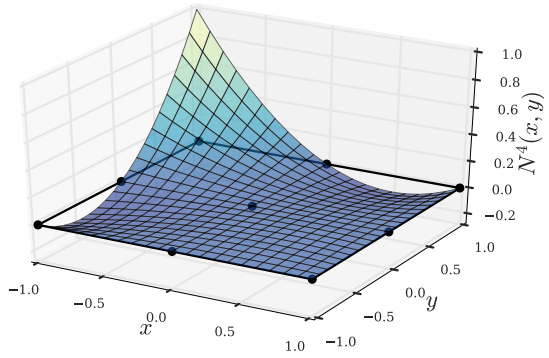
(a) Shape function $N^1(x, y)$.



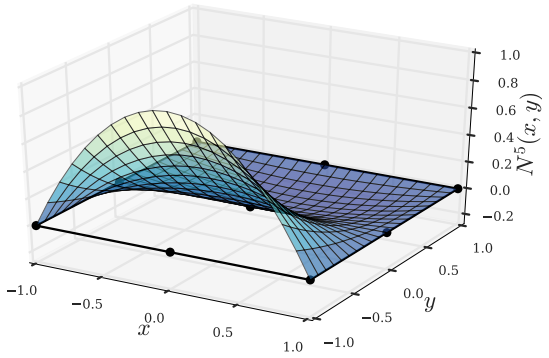
(b) Shape function $N^2(x, y)$.



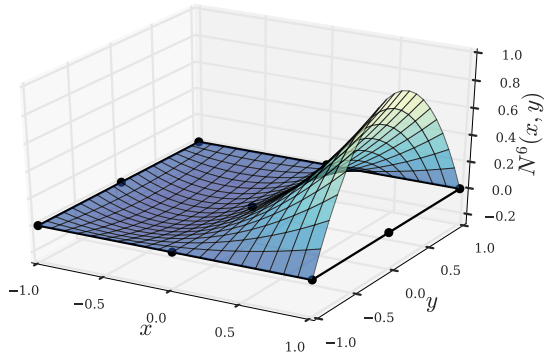
(c) Shape function $N^3(x, y)$.



(d) Shape function $N^4(x, y)$.

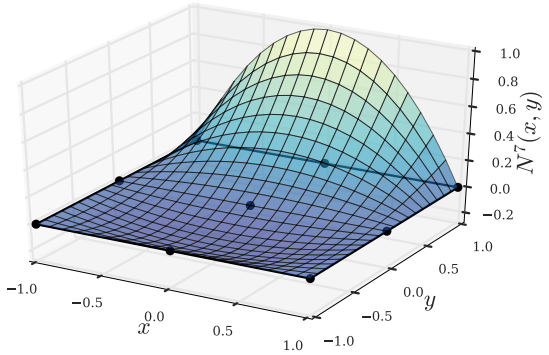


(e) Shape function $N^5(x, y)$.

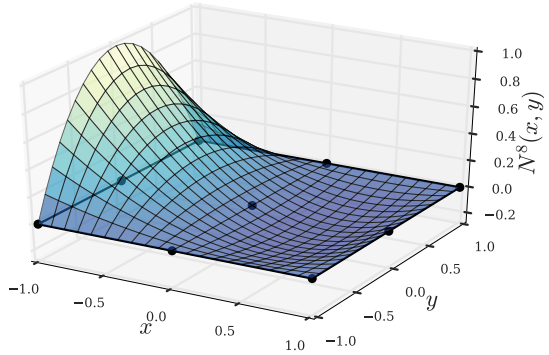


(f) Shape function $N^6(x, y)$.

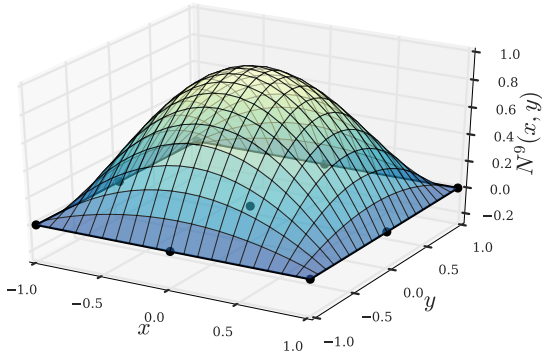
Figure 2.17. Shape functions for a 9-nodes element.



(g) Shape function $N^7(x, y)$.



(h) Shape function $N^8(x, y)$.



(i) Shape function $N^9(x, y)$.

Figure 2.17. Shape functions for a 9-nodes element. (Continued)

2.3.2 Formulation of the finite element matrices

We now discretize the principle of virtual work repeated below for completeness:

$$\int_V \sigma_{ij}\delta u_{i,j}\,\mathrm{d}V - \int_V f_i\delta u_i\,\mathrm{d}V - \int_{S_t} t_i^n\delta u_i\,\mathrm{d}S = 0. \tag{2.43}$$

For that purpose we will divide the complete domain V into N-finite non-overlapping sub-domains over each one of which we will approximate the solution in terms of local interpolating functions (see fig. 2.13). Since the PVW (or weak form of the BVP) has been cast into an integral representation, it is possible to build the total integral considering the contribution of the N-sub-domains like:

$$\sum_{e=1}^{NEL}\int_{V^e} \sigma_{ij}\,\delta u_{i,j}\,\mathrm{d}V^e - \int_V f_i\,\delta u_i\,\mathrm{d}V^e - \int_{S_t} t_i^n\,\delta u_i\,\mathrm{d}S^e = 0 \tag{2.44}$$

For simplicity we consider only a single element or sub-domain, thus;

$$\int_V \sigma_{ij}\delta\epsilon_{i,j}dV - \int_V f_i\delta u_i dV - \int_{S_t} t_i^n \delta u_i dS = 0 \tag{2.45}$$

where it is assumed that the discretized version of each element is later added up (or assembled) into the global equations for the complete model and where we have already used the fact that σ_{ij} is symmetric impliying that $\sigma_{ij}\delta u_{i,j} = \sigma_{ij}\delta\epsilon_{i,j}$.

The involved functions (e.g., displacements, strain, stresses) will be approximated via interpolation of the solution over a determined number of points termed in what follows nodes. Assume for instance that over element e containing n such nodes we know the displacements vector u_i. For instance, fig. 2.18 below shows a typical square 4-noded element of side $2h$. The rectangular components of the displacement vector at each node are labeled like $[u_1, u_2, ..., u_7, u_8]$.

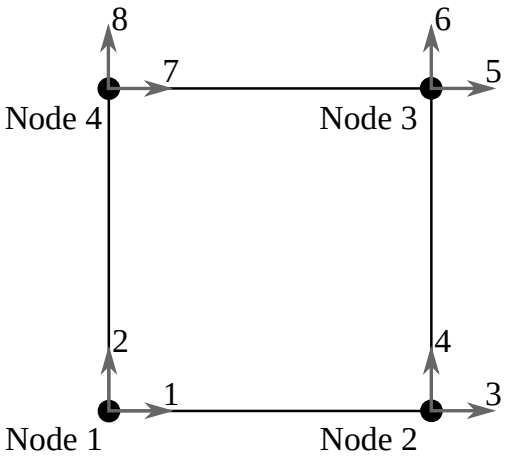


Figure 2.18. Square element of side $2h$.

Furthermore, in a more general treatment, we let the displacements for an arbitrary p-node of a $3D$ problem $u^P = [u^P, v^P, w^P]$. Using ideas from interpolation theory it is now possible to approximate the displacements vector over an arbitrary point $\mathbf{x}$ inside the element by;

$$u_i(\mathbf{x}) = N_i^1(\mathbf{x})u^1 + N_i^2(\mathbf{x})u^2 + \cdots + N_i^P(\mathbf{x})u^P + \cdots + N_i^n(\mathbf{x})u^n$$

or in more general form

$$u_i(\mathbf{x}) = N_i^Q(\mathbf{x})u^Q \tag{2.46}$$

and where the caption superscripts indicate summation over the number of nodes of the element while the subscript refers to the physical character of the variable being interpolated.

Now let us use the kinematic relationshio between strains and displpacements:

$$\varepsilon_{ij}(\mathbf{x}) = \frac{1}{2}(u_{i,j} + u_{j,i})$$

together with eq. (2.46). Carrying the derivatives into the shape functions yields;

$$\varepsilon_{ij}(\mathbf{x}) = \frac{1}{2}\left(\frac{\partial N_i^Q}{\partial x_j} + \frac{\partial N_j^Q}{\partial x_i}\right) u^Q$$

which can be written like

$$\varepsilon_{ij}(\mathbf{x}) = B_{ij}^Q(\mathbf{x}) u^Q \tag{2.47}$$

after letting

$$B_{ij}^Q = \frac{1}{2}\left(\frac{\partial N_i^Q}{\partial x_j} + \frac{\partial N_j^Q}{\partial x_i}\right).$$

Proceeding analogously for the virtual fields after using

$$\delta u_i = N_i^Q(\mathbf{x})\delta u^Q$$

gives us:

$$\int_V C_{ijkl} B_{kl}^P u^P B_{ij}^Q \delta u^Q \,\mathrm{d}V - \int_V f_i N_i^Q \delta u^Q \,\mathrm{d}V - \int_{S_t} t_i^n N_i^Q \delta u^Q \,\mathrm{d}S = 0$$

which can be re-organized into

$$\delta u^Q \int_V B_{ij}^Q C_{ijkl} B_{kl}^P \,\mathrm{d}V \, u^P - \delta u^Q \int_V N_i^Q f_i \,\mathrm{d}V - \delta u^Q \int_{S_t} N_i^Q t_i^n \,\mathrm{d}S = 0$$

which is the generalized discrete version of the PVW for a single element consistent with eq. (2.45). Defining the terms corresponding to the integrals like:

$$\begin{aligned} K^{QP} &= \int_V C_{ijkl} B_{kl}^P B_{ij}^Q dV \\ f_c^Q &= \int_S N_i^Q t_i^n dS \\ f_V^Q &= \int_V N_i^Q f_i dV \end{aligned} \tag{2.48}$$

allows us to write:

$$\delta u^Q f_\sigma^Q - \delta u^Q f_V^Q - \delta u^Q f_c^Q = 0.$$

This equation is once again the generalized discrete version of the PVW corresponding to the Q-displacement degree of freedom. The equation quantifies the work of the forces along the Q-th degree of freedom over the virtual displacements δu^Q. If we now use the arbitrary character of the virtual field δu^Q we can write;

$$f_\sigma^Q - f_V^Q - f_c^Q = 0. \tag{2.49}$$

This is now a mechanical equilibrium equation relating the forces along the Q-th degree of freedom associated to the internal element stresses f_σ^Q; the external body forces f_V^Q and the applied external tractions f_c^Q. Introducing the stiffness of the element gives:

$$K^{QP} u^P = f_V^Q + f_c^Q. \tag{2.50}$$

Formulation for square elements

In the previous section we recognized:

$$\begin{aligned} K^{QP} &= \int_V C_{ijkl} B_{kl}^P B_{ij}^Q dV \\ f_c^Q &= \int_S N_i^Q t_i^n dS \\ f_V^Q &= \int_V N_i^Q f_i dV \end{aligned} \tag{2.51}$$

as the elemental stiffness matrix and the vectors of consistent contact and body forces for a single element. Let us assume that the domain of the single element is a perfect square of side $2h$ (see fig. 2.19), thus;

$$
\begin{aligned}
K &= \int_{-h}^{+h}\int_{-h}^{+h} B^T C B dxdy \\
f_c &= \int_{-h}^{+h} N^T t^n ds \\
f_v &= \int_{-h}^{+h}\int_{-h}^{+h} N^T f dxdy
\end{aligned}
\tag{2.52}
$$

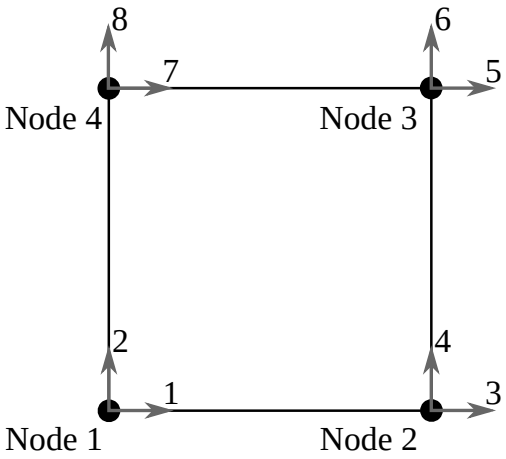


Figure 2.19. Square element of side $2h$.

and where N and B are the displacements interpolation matrix and the strain-displacements interpolation matrix respectively. For the linear 4-noded element shown in fig. 2.19 these matrices have the form defined in the following interpolation equations:

$$
\left\{ \begin{array}{c} u(\vec{x}) \\ v(\vec{x}) \end{array} \right\} = \left[\begin{array}{cccccccc} N^1(\vec{x}) & 0 & N^2(\vec{x}) & 0 & N^3(\vec{x}) & 0 & N^4(\vec{x}) & 0 \\ 0 & N^1(\vec{x}) & 0 & N^2(\vec{x}) & 0 & N^3(\vec{x}) & 0 & N^4(\vec{x}) \end{array} \right] \left\{ \begin{array}{c} u^1 \\ u^2 \\ u^3 \\ u^4 \\ u^5 \\ u^6 \\ u^7 \\ u^8 \end{array} \right\}
$$

$$\begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \end{Bmatrix} = \begin{bmatrix} \frac{\partial N^1(\vec{x})}{\partial x} & 0 & \frac{\partial N^2(\vec{x})}{\partial x} & 0 & \frac{\partial N^3(\vec{x})}{\partial x} & 0 & \frac{\partial N^4(\vec{x})}{\partial x} & 0 \\ 0 & \frac{\partial N^1(\vec{x})}{\partial y} & 0 & \frac{\partial N^2(\vec{x})}{\partial y} & 0 & \frac{\partial N^3(\vec{x})}{\partial x} & 0 & \frac{\partial N^4(\vec{x})}{\partial x} \\ \frac{\partial N^1(\vec{x})}{\partial x} & \frac{\partial N^1(\vec{x})}{\partial y} & \frac{\partial N^2(\vec{x})}{\partial x} & \frac{\partial N^2(\vec{x})}{\partial y} & \frac{\partial N^3(\vec{x})}{\partial x} & \frac{\partial N^3(\vec{x})}{\partial y} & \frac{\partial N^4(\vec{x})}{\partial x} & \frac{\partial N^4(\vec{x})}{\partial y} \end{bmatrix} \begin{Bmatrix} u^1 \\ u^2 \\ u^3 \\ u^4 \\ u^5 \\ u^6 \\ u^7 \\ u^8 \end{Bmatrix}$$

with the independent shape functions being:

$$\begin{aligned} N^1(x) &= \frac{1}{4}(1-x)(1-y) \\ N^2(x) &= \frac{1}{4}(1+x)(1-y) \\ N^3(x) &= \frac{1}{4}(1+x)(1+y) \\ N^4(x) &= \frac{1}{4}(1-x)(1+y) \end{aligned}$$

Notice that in the computation of the stiffness matrix we actually require the spatial derivatives of the shape functions given by;

$$\begin{aligned} \frac{\partial N^1(x)}{\partial x} &= -\frac{1}{4}(1-y) & \qquad \frac{\partial N^1(x)}{\partial y} &= -\frac{1}{4}(1-x) \\ \frac{\partial N^2(x)}{\partial x} &= +\frac{1}{4}(1-y) & \qquad \frac{\partial N^2(x)}{\partial y} &= -\frac{1}{4}(1+x) \\ \frac{\partial N^3(x)}{\partial x} &= +\frac{1}{4}(1+y) & \qquad \frac{\partial N^3(x)}{\partial y} &= +\frac{1}{4}(1+x) \\ \frac{\partial N^4(x)}{\partial x} &= -\frac{1}{4}(1+y) & \qquad \frac{\partial N^4(x)}{\partial y} &= +\frac{1}{4}(1-y) \end{aligned}$$

The contribution to the element matrix from a typical nodal point is thus given by;

$$K = \int_{-h}^{+h}\int_{-h}^{+a} \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \frac{\partial N^K(x)}{\partial x} & 0 & \frac{\partial N^K(x)}{\partial y} & \cdots & \cdots \\ \cdots & 0 & \frac{\partial N^K(x)}{\partial y} & \frac{\partial N^K(x)}{\partial x} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} A & B & 0 \\ B & A & 0 \\ 0 & 0 & C \end{bmatrix} \begin{bmatrix} \cdots & \frac{\partial N^Q(x)}{\partial x} & 0 & \cdots \\ \cdots & 0 & \frac{\partial N^Q(x)}{\partial y} & \cdots \\ \cdots & \frac{\partial N^Q(x)}{\partial y} & \frac{\partial N^Q(x)}{\partial x} & \cdots \end{bmatrix} dxdy$$

while a typical element of the resultant stiffness matrix takes the general form:

$$K \equiv A \int_{-h}^{+h}\int_{-h}^{+h} \frac{\partial N^K}{\partial x}\frac{\partial N^Q}{\partial x}\,\mathrm{d}x\,\mathrm{d}y + C \int_{-h}^{+h}\int_{-h}^{+h} \frac{\partial N^K}{\partial y}\frac{\partial N^Q}{\partial x}\,\mathrm{d}x\,\mathrm{d}y\,.$$

It should be noticed that for the considered perfectly square element the resulting stiffness matrix can be partitioned into factors which depend upon the material properties only and into a single factor which depends on the size h. This last contribution can be obtained analytically.

Once the elemental matrices are obtained these are assembled into the final global system of equations. Details of the assembly process are discussed in chapter 6. The following Python script computes the stiffness matrix for the 4-noded element.

```
"""
Compute the stiffness matrix for a 4-noded square element
"""
from __future__ import division
import numpy as np
from sympy import *


def umat(nu,E):
    """2D Elasticity constitutive matrix"""
    C = zeros(3, 3)
    G = E/(1 - nu**2)
    mnu = (1 - nu)/2.0
    C[0, 0] = G
    C[0, 1] = nu*G
    C[1, 0] = C[0, 1]
    C[1, 1] = G
    C[2, 2] = G*mnu

    return C


def stdm4(x,y):
    """Four noded element strain-displacement matrix"""
    N = zeros(4)
    B = zeros(3,8)
    N = S(1)/4*Matrix([
         (1 - x)*(1 - y),
         (1 + x)*(1 - y),
         (1 + x)*(1 + y),
         (1 - x)*(1 + y)])
```

```
    dhdx=zeros(2,4)
    for i in range(4):
        dhdx[0,i]=diff(N[i],x)
        dhdx[1,i]=diff(N[i],y)

    for i in range(4):
        B[0, 2*i] = dhdx[0, i]
        B[1, 2*i+1] = dhdx[1, i]
        B[2, 2*i] = dhdx[1, i]
        B[2, 2*i+1] = dhdx[0, i]

    return B


# Assign symbols
x , y = symbols('x y')
nu, E = symbols('nu, E')
h=symbols('h')

K = zeros(8,8)

# Symbolically compute matrices
C = umat(nu, E)
B = stdm4(x, y)
K_int = B.T*C*B

# Integrate final stiffness
for i in range(8):
    for j in range(8):
        K[i,j] = integrate(K_int[i,j], (x,-h,h), (y,-h,h))

knum = K.subs([(E, S(1)), (nu, S(1)/3.0), (h, S(2))])

print knum
```

Formulation in the natural space: the continuum mechanics analogy

In typical finite element equilibrium equations we need to perform integration over the reference element domain $V_0(\mathbf{x})$ corresponding to originally arbitrarily shaped sub-domains as created during the meshing process. In order to proceed with this integration it is useful to consider the following continuum mechanics analogy.

First assume that the actual physical domain $V_0(\mathbf{x})$ is the result of a deformation process

imparted upon the natural domain as shown in fig. 2.20. In this analogy, the physical domain $V_0(\mathbf{x})$ is regarded like a "deformed" configuration at an imaginary time $t = t$, while the natural "undeformed" domain $V(\mathbf{r})$ is treated like a reference undeformed configurations at time $t = 0$. Both configurations are assumed to be connected through a deformation process

$$\begin{aligned} \mathbf{X} &= \mathbf{X}(\mathbf{r}) \\ \mathbf{r} &= \mathbf{r}(\mathbf{X}) \end{aligned} \tag{2.53}$$

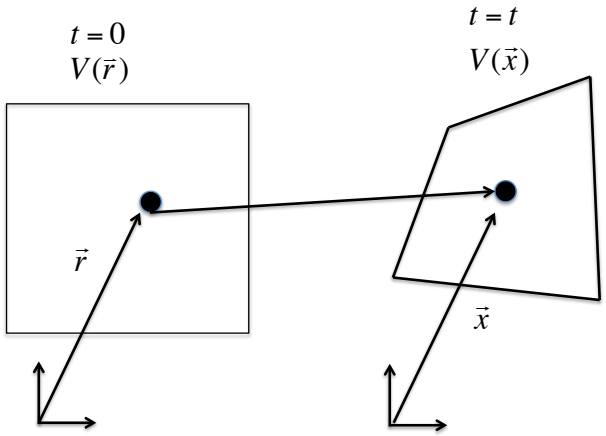


Figure 2.20. Definition of the natural domain

In eq. (2.53) we can understand $\mathbf{r}$ like a material (Lagrangian) variable and $\mathbf{X}$ like a spatial (or Eulerian) variable. Using the continuum mechanics analogy it is clear that the "deformation" process at the continuum level is fully characterized by the "deformation" gradient or Jacobian of the transformation eq. (2.53) and defined according to

$$\mathrm{d}X_i = \frac{\partial X_i}{\partial r_J}\,\mathrm{d}r_J \equiv J_{iJ}\,\mathrm{d}r_J \tag{2.54}$$

where $\mathrm{d}r_J$ and $\mathrm{d}X_i$ represent material vectors in the original and deformed configuration. From eq. (2.54) it is evident that the Jacobian contains all the information describing the change of the physical sub-domain with respect to the natural element. For the element integration process we will assume that every element $V(\mathbf{r})$ in the natural domain deforms into the physical element $V_0(\mathbf{X})$, thus allowing us to write typical terms like the ones in the material stiffness matrix

$$\int_{V(\mathbf{X})} \hat{B}^K_{ij}(\mathbf{X}) C_{ijkl} \hat{B}^P_{kl}(\mathbf{X}) dV(\mathbf{X}) \equiv \int_{V_0(\mathbf{r})} \hat{B}^K_{ij}(\mathbf{r}) C_{ijkl} \hat{B}^P_{kl}(\mathbf{r}) J dV_0(\mathbf{r}) \tag{2.55}$$

where we have used $dV(\mathbf{X}) = J dV(\mathbf{r})$, with J being the determinant of the deformation gradient and in general we transform functions between the natural and physical space making use of eq. (2.53) according to

$$f(\mathbf{r}) = F[\mathbf{X}(\mathbf{r})] \tag{2.56}$$

Interpolation scheme

Having identified the fact that the integration process will take place in the natural domain, we will approach the interpolation process directly in this natural space. In the case of the displacement based finite element method all the involved variables will then be obtained via interpolation of nodal displacements. For instance, assume that a given problem variable is defined in the physical space by the tensor $\Phi_{ik\ldots p}(\mathbf{X})$. The interpolated variable is then written like;

$$\Phi_{ij\ldots p}(\mathbf{X}) = H^K_{ij\ldots p}(\mathbf{r})\hat{u}^K \tag{2.57}$$

where $\hat{u}^K$ represents a vector of nodal points displacements, see fig. 2.21, and $H^K_{ij\ldots p}(\mathbf{r})$ is an interpolator which keeps the tensorial character of the original physical variable $\Phi_{ik\ldots p}(\mathbf{X})$ and where the super-index makes reference to a nodal identifier (with the summation convention in place).

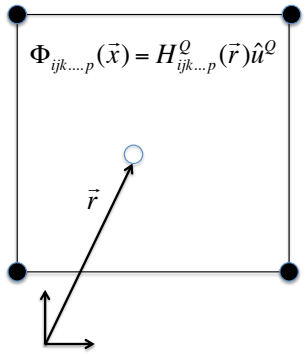


Figure 2.21. General interpolation strategy in the natural domain

Since the primary variable corresponds to displacements it must be kept in mind that $H^K_{ij\ldots p}(\mathbf{r})$ corresponds to combinations of derivatives (or other arbitrary combinations) of the basic element shape functions defined in;

$$u_i(\mathbf{X}) = N^K_i(\mathbf{r})\hat{u}^K \tag{2.58}$$

For the general interpolation process we need two kinds of transformations. First we need to transform integrals over the physical space into integrals into the natural space which corresponds to

$$\int\limits_{V(\mathbf{X})} F(\mathbf{X})dV(\mathbf{X}) \equiv \int\limits_{V_0(\mathbf{r})} f(\mathbf{r})JdV_0(\mathbf{r}) \tag{2.59}$$

Second we need to relate spatial differentiation in both, the physical and spatial domains. Let

us define these operators like ∇_i^X and ∇_I^r respectively. It then follows from eq. (2.56) that

$$\frac{\partial F}{\partial X_i} = \frac{\partial f}{\partial r_J}\frac{\partial r_J}{\partial X_i} \tag{2.60}$$

from where we can establish the connection between the two operators like

$$\nabla_i^X = J_{iJ}^{-1}\nabla_J^r \tag{2.61}$$

The fundamental interpolator

We further define the fundamental interpolator giving rise to gradients of the primary displacement variable in the physical space according to

$$u_{i,j}(\mathbf{X}) = L_{ij}^K(\mathbf{r})\hat{u}^K \tag{2.62}$$

This fundamental interpolator $L_{ik}^K(\mathbf{r})$ is derived after using eq. (2.58) and eq. (2.61) in the physical displacement gradient definition as shown next

$$\begin{aligned} u_{i,j}(\mathbf{X}) &= \nabla_j^X u_i(\mathbf{X}) \\ u_{i,j}(\mathbf{X}) &= \nabla_j^X N_i^K(\mathbf{r})\hat{u}^K \\ u_{i,j}(\mathbf{X}) &= J_{jQ}^{-1}\nabla_Q^r N_i^K(\mathbf{r})\hat{u}^K \\ u_{i,j}(\mathbf{X}) &= J_{jQ}^{-1}N_{i,Q}^K(\mathbf{r})\hat{u}^K \end{aligned}$$

then

$$L_{ij}^K(\mathbf{r}) = J_{jQ}^{-1}N_{i,Q}^K(\mathbf{r}) \tag{2.63}$$

Elemental stiffness matrix

The elemental material stiffness matrix computed in the natural domain of fig. 2.22 reads

$$K^{KP} = \int\limits_{V_0(\mathbf{r})} \hat{B}_{ij}^K(\mathbf{r})C_{ijkl}\hat{B}_{kl}^P(\mathbf{r})JdV_0(\mathbf{r}) \equiv \int\limits_{r=-1}^{r=+1}\int\limits_{s=-1}^{s=+1} \hat{B}_{ij}^K(r,s)C_{ijkl}\hat{B}_{kl}^P(r,s)J(r,s)\mathrm{d}r\mathrm{d}s \tag{2.64}$$

Once the interpolator $\hat{B}_{ij}^K(\mathbf{r})$ has been identified the elemental stiffness matrix is obtained via numerical integration (quadrature) as described in (2.65)

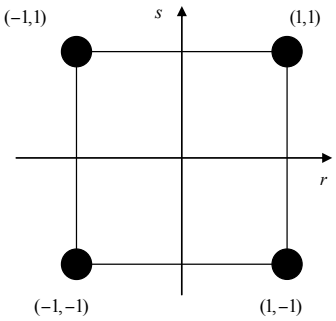


Figure 2.22. Natural domain of integration

$$\int_{r=-1}^{r=+1}\int_{s=-1}^{s=+1} \hat{B}_{ij}^{K}(r,s)C_{ijkl}\hat{B}_{kl}^{P}(r,s)J(r,s)\mathrm{d}r\mathrm{d}s \approx \sum_{i,j=1}^{\mathrm{NGPTS}} \alpha_i\alpha_j \hat{B}_{kl}^{K}(r_i,s_j)C_{ijkl}\hat{B}_{kl}^{P}(r_i,s_j)J(r_i,s_j) \quad (2.65)$$

and where NGPTS corresponds to the number of integration points, α_j is a weighting factor and r_i, s_j are the coordinates of a typical point $\mathbf{r}$ in the natural space of fig. 2.22.

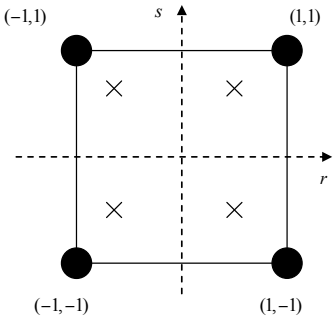


Figure 2.23. Natural integration domain showing quadrature evaluation nodes

One important aspect of the numerical integration that has to be kept in mind is accuracy. Depending on the particularly selected integration scheme, the number of introduced integration points fixes the maximum polynomial order of the considered functions that can be integrated accurately. In the case of the integrand in eq. (2.65), it is clear that this order increases as the distortion of the physical element with respect to the natural element increases. One way of dealing with this dependency of accuracy with element distortion is to make use of adaptative integration techniques which are numerically expensive. What is actually done in standard FEM analysis is to choose the number of quadrature points beforehand and introduce distortion related error criteria inside the code in such a way that some sort of validation is performed before the numerical integration process is started.

Strain displacement interpolator for the infinitesimal strain tensor

The Q-th nodal contribution to the infinitesimal strain-displacement interpolator can be obtained in explicit form as follows. Let L_x^Q and L_y^Q be the spatial differential operators in x and y respectively. We have after expanding eq. (2.63)

$$L_x^Q = J_{xP}^{-1}\frac{\partial N^Q}{\partial r_P} \equiv J_{xr}^{-1}\frac{\partial N^Q}{\partial r} + J_{xs}^{-1}\frac{\partial N^Q}{\partial s}$$
$$L_y^Q = J_{yP}^{-1}\frac{\partial N^Q}{\partial r_P} \equiv J_{yr}^{-1}\frac{\partial N^Q}{\partial r} + J_{ys}^{-1}\frac{\partial N^Q}{\partial s}$$

or in matrix form

$$\begin{Bmatrix} L_x^Q \\ L_y^Q \end{Bmatrix} = \begin{bmatrix} J_{xP}^{-1} & J_{xs}^{-1} \\ J_{yr}^{-1} & J_{ys}^{-1} \end{bmatrix} \begin{Bmatrix} \frac{\partial N^Q}{\partial r} \\ \frac{\partial N^Q}{\partial s} \end{Bmatrix} \tag{2.66}$$

The Q-th nodal contribution is then assembled as follows;

$$\begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix} = \begin{bmatrix} & L_x^Q & 0 & \\ \cdots & 0 & L_y^Q & \cdots \\ & L_y^Q & L_{xy}^Q & \end{bmatrix} \begin{Bmatrix} \vdots \\ u^Q \\ v^Q \\ \vdots \end{Bmatrix} \tag{2.67}$$

Algorithm 2: Strain-displacement interpolator

Data: Nodal coordinates x^Q
Result: Strain-displacement interpolator B_{ij}^Q
Compute Jacobian $J_{iJ} = \frac{\partial^2 N_i^Q}{\partial r_J \partial \hat{x}}^Q$
Invert Jacobian $J_{iJ} \rightarrow J_{iJ}^{-1}$
Compute fundamental interpolator $L_{ij}^Q = J_{jP}^{-1}\frac{\partial N_i^Q}{\partial r_P}$
Assemble $B_{ij}^Q = \frac{1}{2}\left(L_{ij}^Q + L_{ji}^Q\right)$

Chapter 3

Wave propagation problems in the frequency domain

3.1 Problem formulation

3.2 Discretization in complex algebra

3.3 Discretization in real algebra

3.4 Scattering by a micropolar solid

3.5 Coupling with the BEM

Chapter 4

Wave propagation problems in the time domain

4.1 Problem formulation

4.2 Explicit time integration algorithm

4.3 Statement of the Problem

Consider the discrete dynamic equilibrium equations at time t

$$M\,{}^{t}A + C\,{}^{t}V + K\,{}^{t}U = {}^{t}F \tag{4.1}$$

where M, C, K are the assembled mass, damping and stiffness matrix respectively and similarly ${}^{t}A$, ${}^{t}V$, ${}^{t}U$, ${}^{t}F$ are the nodal accelerations, velocities, displacements and external loads vectors at time t. In terms of forces, eq. (4.1) can be written like;

$${}^{t}F^{I} + {}^{t}F^{D} + {}^{t}F^{s} = {}^{t}F \tag{4.2}$$

where ${}^{t}F^{I}$, ${}^{t}F^{D}$ and ${}^{t}F^{s}$ are inertial, damping and elastic nodal forces respectively.

Expanding the acceleration and velocity terms at time t in a consistent finite central differences scheme we have;

$$
\begin{aligned}
{}^{t}A &= \frac{1}{\Delta t^2}\left({}^{t-\Delta t}U - 2{}^{t}U + {}^{t+\Delta t}U\right) \\
{}^{t}V &= \frac{1}{2\Delta t}\left(-{}^{t-\Delta t}U + {}^{t+\Delta t}U\right)
\end{aligned}
\tag{4.3}
$$

It is convenient to write eq. (4.3) like;

$$
\begin{aligned}
{}^{t}A &= {}^{t}\hat{A} + \frac{1}{\Delta t^2}{}^{t+\Delta t}U \\
{}^{t}V &= {}^{t}\hat{V} + \frac{1}{2\Delta t}{}^{t+\Delta t}U
\end{aligned}
\tag{4.4}
$$

where:

$$
\begin{aligned}
{}^{t}\hat{A} &= \frac{1}{\Delta t^2}\left({}^{t-\Delta t}U - 2{}^{t}U\right) \\
{}^{t}\hat{V} &= -\frac{1}{2\Delta t}{}^{t-\Delta t}U
\end{aligned}
\tag{4.5}
$$

are termed predictors, while $\frac{1}{\Delta t^2}{}^{t+\Delta t}U$ and $\frac{1}{2\Delta t}{}^{t+\Delta t}U$ are termed correctors.

Using eq. (4.3) in eq. (4.1) yields;

$$
\left(\frac{1}{\Delta t^2}M + \frac{1}{2\Delta t}C\right)^{t+\Delta t}U = {}^{t}F - M{}^{t}\hat{A} - C{}^{t}\hat{V} - K{}^{t}U \tag{4.6}
$$

or after letting;

$$
\hat{K} = \frac{1}{\Delta t^2}M + \frac{1}{2\Delta t}C
$$

and

$$
{}^{t}\hat{F} = {}^{t}F - M{}^{t}\hat{A} - C{}^{t}\hat{V} - K{}^{t}U
$$

we have;

$$
\hat{K}{}^{t+\Delta t}U = {}^{t}\hat{P}
$$

which can be solved for ${}^{t+\Delta t}U$.

Algorithm 3: Explicit algorithm

Data: Values at $t = 0$, Geometry, Material Paramters
Result: Displacements, Velocity and Acceleration at any time $t = t$
Compute predictors ${}^{t}\hat{A}$, ${}^{t}\hat{V}$, ${}^{t}\hat{U}$
Assemble $\hat{K}$, ${}^{t}\hat{P}$
Solve $\hat{K}^{t+\Delta t}U = {}^{t}\hat{P}$
Perform Corrections ${}^{t}A \leftarrow {}^{t}\hat{A} + \frac{1}{2\Delta t^2}{}^{t+\Delta t}U$ and ${}^{t}V \leftarrow {}^{t}\hat{V} + \frac{1}{2\Delta t}{}^{t+\Delta t}U$

Redefine forces as follows

$$\begin{aligned} {}^{j}F^{I} &= \frac{1}{\Delta t^2} M {}^{j}U \\ {}^{j}F^{D} &= \frac{1}{2\Delta t} C {}^{j}U \\ {}^{j}F^{S} &= K {}^{j}U \end{aligned} \tag{4.7}$$

and write (4.6) as

$${}^{t+\Delta t}F^{I} + {}^{t+\Delta t}F^{D} = {}^{t}F - {}^{t}F^{s} + 2{}^{t}F^{I} - {}^{t-\Delta t}F^{I} + {}^{t-\Delta t}F^{D} \tag{4.8}$$

- (4.8) is an equilibrium equation at time $t = t$ allowing to predict the displacements at time $t = t + \Delta t$ in terms of previously known values at times t and $t = t - \Delta t$.
- The equation is exact within the error introduced by the expansion used in (4.3).

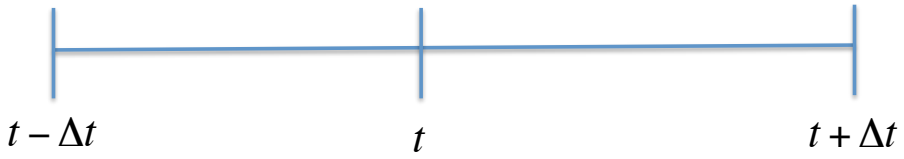


Figure 4.1. Definition of the general iteration

- The first predicted solution is at $t = \Delta t$ and we require data at $t = -\Delta t$ and at $t = 0$.

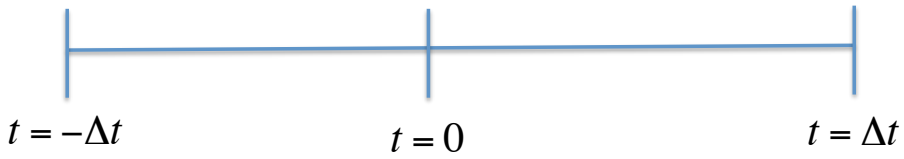


Figure 4.2. Definition of the general iteration

4.3.1 Damping Assumptions

1 Use Rayleigh Damping and retain the velocity expansion used in (4.3). That is;

$$C = \alpha M + \beta K \tag{4.9}$$

then we have (in terms of forces);

$$(1 + \beta \Delta t^2)^{t+\Delta t} F^I + \frac{\alpha}{2\Delta t}{}^{t+\Delta t} F^S = {}^t \hat{F} \tag{4.10}$$

where;

$${}^t \hat{F} = {}^t R - {}^t F^S + 2^t F^I - {}^{t-\Delta t} F^I + {}^{t-\Delta t} F^D$$

Solution in equation (4.10) requires the full assembly and factorization of an effective stiffness matrix.

2 Neglect damping (This is however inconvenient for finite domains)

$${}^{t+\Delta t} F^I = {}^t F - {}^t F^S + 2^t F^I - {}^{t-\Delta t} F^I \tag{4.11}$$

3 Use Rayleigh damping but modify the velocity expansion introduced in (4.3). Using

$${}^t V = \frac{1}{\Delta t}({}^t U - {}^{t-\Delta t} U) \tag{4.12}$$

yielding;

$${}^{t+\Delta t} F^I = {}^t F - {}^t F^S + 2^t F^I - {}^t F^D + {}^{t-\Delta t} F^D - {}^{t-\Delta t} F^I \tag{4.13}$$

Defining a set of forces associated to the initial conditions like;

$${}^{t-\Delta t} F^{IC} = {}^{t-\Delta t} F^I - {}^{t-\Delta t} F^D$$

we have

$${}^{t+\Delta t} F^I = {}^t F - {}^t F^S + 2^t F^I - {}^t F^D - {}^{t-\Delta t} F^{IC} \tag{4.14}$$

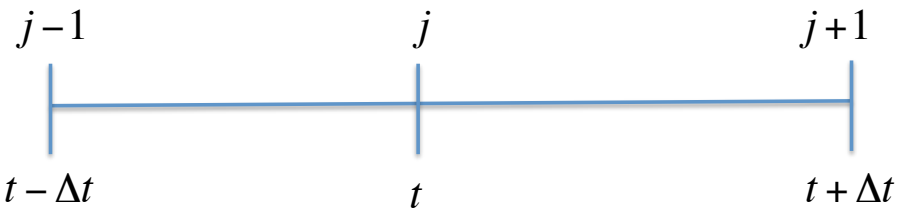


Figure 4.3. Definition of the general iteration

4.3.2 Algorithm corresponding to the damping assumption 3

Let us write (4.14) like

$$ {}^{j+1}F^I = {}^{j}F - {}^{j}F^S + 2^{j}F^I - {}^{j}F^D - {}^{j-1}F^{IC} \tag{4.15} $$

where the initialization process corresponds to;

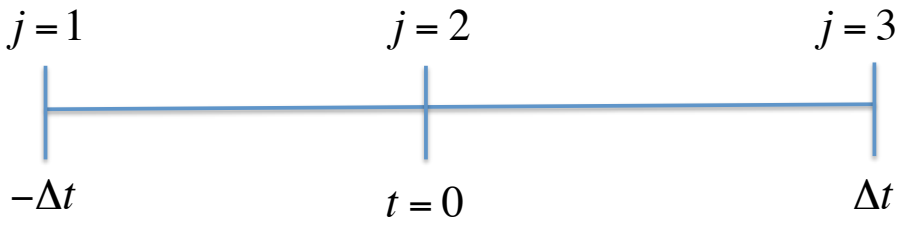


Figure 4.4. Definition of the initial iteration

Applying (4.15) for $t = 0$ we have;

$$ {}^{0}F^I = {}^{0}F - {}^{0}F^S + 2^{0}F^I - {}^{0}F^D - {}^{-\Delta t}F^{IC} $$

from which it is clear that we require ${}^{-\Delta t}U$. Applying the central difference expansion at $t = 0$ and solving for ${}^{-\Delta t}U$ yields;

$$ {}^{-\Delta t}U = {}^{0}F - \Delta t\,{}^{0}V + \frac{\Delta t^2}{2}\,{}^{0}A \tag{4.16} $$

Using (4.16) in (4.15) allows us to start up the algorithm.

Particulars

In what follows we concentrate on this last algorithm and in order to study some details we return to its standard displacements form. Writing (4.15) in terms of displacements and re-arranging yields;

$$\frac{1}{\Delta t^2}M^{t+\Delta t}U =^t F-(1+\frac{\beta}{\Delta t})K^tU+(\frac{2}{\Delta t^2}-\frac{\alpha}{\Delta t})M^tU-(\frac{1}{\Delta t^2}-\frac{\alpha}{\Delta t})M^{t-\Delta t}U+(\frac{\beta}{\Delta t})K^{t-\Delta t}U \quad (4.17)$$

Let;

$$\begin{aligned} a_1 &= 1+\frac{\beta}{\Delta t} \\ a_2 &= \frac{2}{\Delta t^2}-\frac{\alpha}{\Delta t} \\ a_3 &= \frac{1}{\Delta t^2}-\frac{\alpha}{\Delta t} \\ a_4 &= \frac{\beta}{\Delta t} \end{aligned}$$

$${}^{t+\Delta t}F^I =^t F - a_1K^tU + a_2M^tU - a_3M^{t-\Delta t}U + a_4K^{t-\Delta t}U \quad (4.18)$$

4.3.3 Decoupling

Consider the equation for the i-th d.o.f;

$$\frac{1}{\Delta t^2}M_{ij}^{t+\Delta t}U_j =^t F_i - a_1K_{ij}^tU_j + a_2M_{ij}^tU_j - a_3M_{ij}^{t-\Delta t}U_j + a_4K_{ij}^{t-\Delta t}U_j \quad (4.19)$$

where we keep i fixed in (4.19). For a lumped mass matrix we can write;

$$M_{ij} = m_I\delta_{ij}$$

then (4.19) becomes;

$$\frac{1}{\Delta t^2}m_I^{t+\Delta t}U_i =^t F_I - a_1K_{ij}^tU_j + a_2m_I^tU_i - a_3m_I^{t-\Delta t}U_i + a_4K_{ij}^{t-\Delta t}U_j \quad (4.20)$$

Let;

$$\begin{aligned} {}^{t+\Delta t}F_i^I &= \frac{1}{\Delta t^2} m_I^{t+\Delta t} U_i \\ {}^{t}\hat{F}_i^S &= a_1 K_{ij}^t U_j \\ {}^{t}\hat{F}_i^I &= a_2 m_I^t U_i \\ {}^{t-\Delta t}\hat{F}_i^I &= a_3 m_I^{t-\Delta t} U_i \\ {}^{t-\Delta t}\hat{F}_i^S &= a_4 K_{ij}^{t-\Delta t} U_j \end{aligned} \tag{4.21}$$

so the recursive equation takes the form;

$${}^{t+\Delta t}F_i^I = {}^{t}\ F_i - {}^{t}\ \hat{F}_i^S + {}^{t}\ \hat{F}_i^I - {}^{t-\Delta t}\ \hat{F}_i^I + {}^{t-\Delta t}\ \hat{F}_i^S \tag{4.22}$$

and the algorithm then reduces to;

Algorithm 4: Summarized Algorithm

Data: Time span, Geometry, Material Parameters
Result: Displacements, Velocity and Acceleration time histories
Compute ${}^{t+\Delta t}F_i^I$
Solve for ${}^{t+\Delta t}U_i = \left(\frac{\Delta t^2}{m_I}\right)^{t+\Delta t} F_i^I$
Update ${}^{t}V_i$, ${}^{t}A_i$

To initialize the algorithm we apply the FD's equations at $t = 0$

$$\frac{1}{\Delta t^2} m_I^{\Delta t} U_i = {}^{0}\ F_i - a_1 K_{ij}^0 U_j + a_2 m_I^0 U_i - a_3 m_I^{-\Delta t} U_i + a_4 K_{ij}^{-\Delta t} U_j$$

where ${}^{-\Delta t}U_i$ is obtained from (4.16)

$${}^{-\Delta t}U_i = {}^{0}\ U_i - \Delta t^0 V_i + \frac{\Delta t^{2\,0}}{2}\ A_i \tag{4.23}$$

The initial acceleration is obtained after assuming homogeneous IC's;

$$m_I^0 A_i + C_{ij}^0 V_j + K_{ij}^0 U_j = {}^{0}\ F_i$$

therefore

$$m_I^0 A_i = {}^0F_i$$
$$-^{\Delta t}U_i = \frac{\Delta t^2}{2m_I}{}^0 F_i$$

Moreover, neglecting the damping effects on the prediction of $^{\Delta t}U_i$ yields;

$$^{\Delta t}U_i = \frac{\Delta t^2}{2m_I}{}^0 F_i$$

Algorithm 5: Full Algorithm

Data: Time span, Geometry, Material Parameters
Result: Displacements, Velocity and Acceleration time histories
Initialize solution vectors ($j = 1$);
$^0U_i \longrightarrow {}^1U_i = 0$, $^0V_i = 0$, $^1A_i = \dfrac{^1R_i}{m_I}$;
Select Δt and integration constants a_1,a_2, a_3, a_4;
Fix 1-st predicted value (let $j = 2$);

$$^{\Delta t}U_i \longleftarrow \frac{\Delta t^2}{2m_I}{}^0 F_i \longleftrightarrow \left[{}^2U_i \longleftarrow \frac{\Delta t^2}{2m_I}{}^1 F_i\right]$$

Time Integration Phase;
while $j \leq N$ **do**

$$^{j+1}F_i^I \longleftarrow {}^jF_i - a_1K_{ij}^jU_j + a_2m_I^jU_j - a_3m_I^{j-1}U_i + a_4K_{ij}^{j-1}U_j$$
$$^{j+1}U_i \longleftarrow \frac{\Delta t^2}{2m_I}{}^0 F_i^I$$
$$^jA_i \longleftarrow \frac{1}{\Delta t^2}\left({}^{j-1}U_i - 2^jU_i + {}^{j+1}U_i\right)$$
$$^jV_i \longleftarrow \frac{1}{2\Delta t}\left(-^{j-1}U_i + {}^{j+1}U_i\right)$$
$$j \longleftarrow j+1$$

end

Nodal Assembler

In the uncoupled explicit finite element formulation the equation solving process proceeds one degree of freedom at a time. This implies a different assembly process to the one used in an implicit algorithm where a formal coefficient matrix is assembled and inverted. Now the mass,

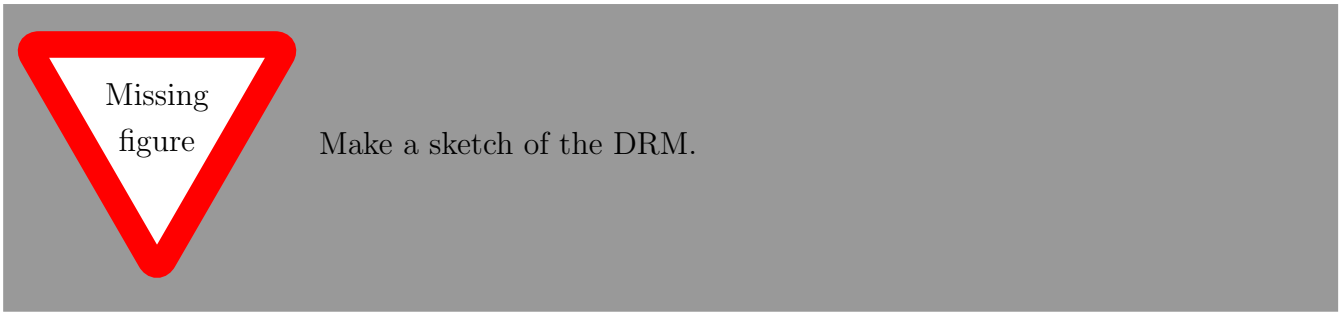
damping and stiffness elemental matrices are used to obtain effective nodal forces at each degree of freedom. In summary the mesh is not covered in an element by element basis, but in a node by node basis. In the following algorithm we discuss this nodal assembly process where in order to solve the displacement at a given degree of freedom prior knowledge of the element contributing to the given node is necessary. In the nodal assembler algorithm the following arrays are needed.

ILIST(): Stores the elements connected to the current node.
LPLIST():Stores the local position of the current node in each one of the elements of ILIST().
NIEL():Number of elements at the current node. This array is used to access ILIST().

Algorithm 6: Nodal Assembler

Data: Number of nodal points, number of elements, Model
Result: Displacements, Velocity and Acceleration time histories
while $i \leq NUMNP$ **do**
| $K \leftarrow NIEL(i)$
end

4.4 The DRM approach



Make a sketch of the DRM.

Chapter 5

Periodic media problems

A periodic material is defined as the repetition of a given motif in one, two or three dimensions. This motif refers to heterogeneities at micro-structural level and it can contains several materials and diverse geometries. These materials can be solids or fluids.

A periodic material is completely described by a lattice and a elementary unit, that is termed *elementary cell*. The lattice is defined by a vector base, and the dimension of the base is the number of direction for the periodicity. This set is called lattice *base vectors*, and they allow to build the whole material applying successive translation operations [9]. For periodic material in 3D, we will denote the base vectors as $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ (see Figure 5.1). In the case of 2D periodicity, we can have a material that is infinite or finite in the third dimension (see Figure 5.2), we denote the vectors as $\mathbf{a}_1$ and $\mathbf{a}_2$. There is also possible to have periodic materials in one direction, and this one can be finite or infinite in the other two directions.

The mesh generator `gmsh` provides an option to link elements in opposite sides. [10]. Finite Element Method Magnetic (FEMMM) provides the option of *periodic* boundary conditions, where Multipoint constraints are imposed, this are used for approximation of infinite boundary conditions [11].

Describe periodic boundary conditions and Bloch theorem as boundary conditions.

Add Periodic media info.

Periodic boundary conditions are common in molecular dynamics and micromechanics/homogenization.

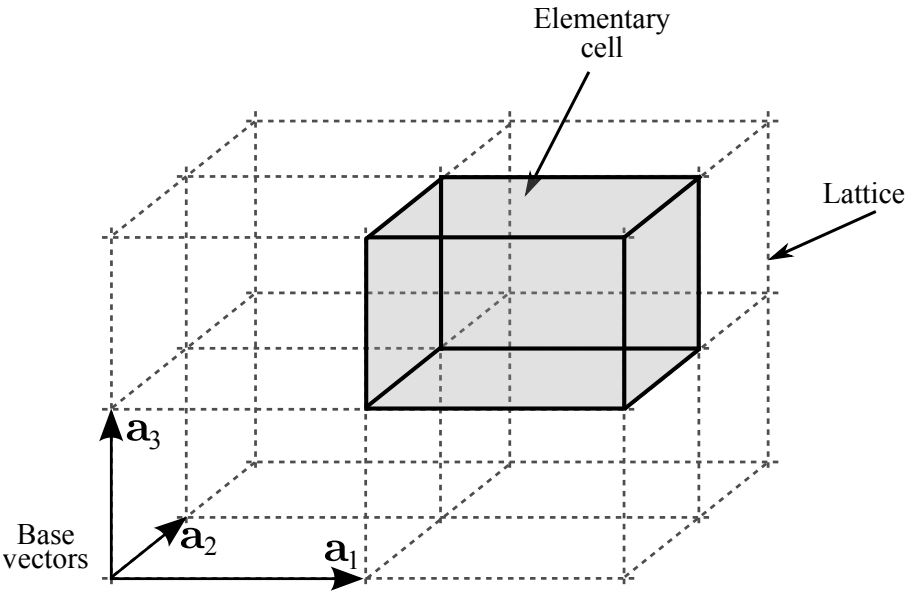


Figure 5.1. Depiction of a periodic material in 3D.

5.1 Bloch theorem

Let us start from an equation of the form

$$Lu(\mathbf{x}) = \omega^2 u(\mathbf{x}) \ , \tag{5.1}$$

where L is a differential operator with algebraic properties that are similar to the wave equation [12]. The Bloch theorem establishes that the solution for (5.1) is given by

$$u(\mathbf{x}) = w(\mathbf{x}) e^{i\mathbf{k}\cdot\mathbf{x}} \ , \tag{5.2}$$

where $w(\mathbf{x})$ is function with the same periodicity of the material and is termed Bloch function. The theorem according to [13] states

> *The eigenfunctions of the wave equation for a periodic potential are the product of a plane wave* $\exp(i\mathbf{k}\cdot\mathbf{r})$ *times a function* $u_{\mathbf{k}}(\mathbf{r})$ *with the periodicity of the crystal lattice.*

If we write the solution (5.2) for the point $\mathbf{x}+\mathbf{a}$, we obtain

$$u(\mathbf{x}+\mathbf{a}) = w(\mathbf{x}+\mathbf{a}) e^{i\mathbf{k}\cdot(\mathbf{x}+\mathbf{a})} \ ,$$

being $\mathbf{a} = n_i \mathbf{a}_i$ a vector with the periodicity of the lattice, and $n_i \in \mathbb{Z}$. Since$w(\mathbf{x})$ should present the same periodicity of the lattice

$$u(\mathbf{x}+\mathbf{a}) = w(\mathbf{x}) e^{i\mathbf{k}\cdot(\mathbf{x}+\mathbf{a})} \ ,$$

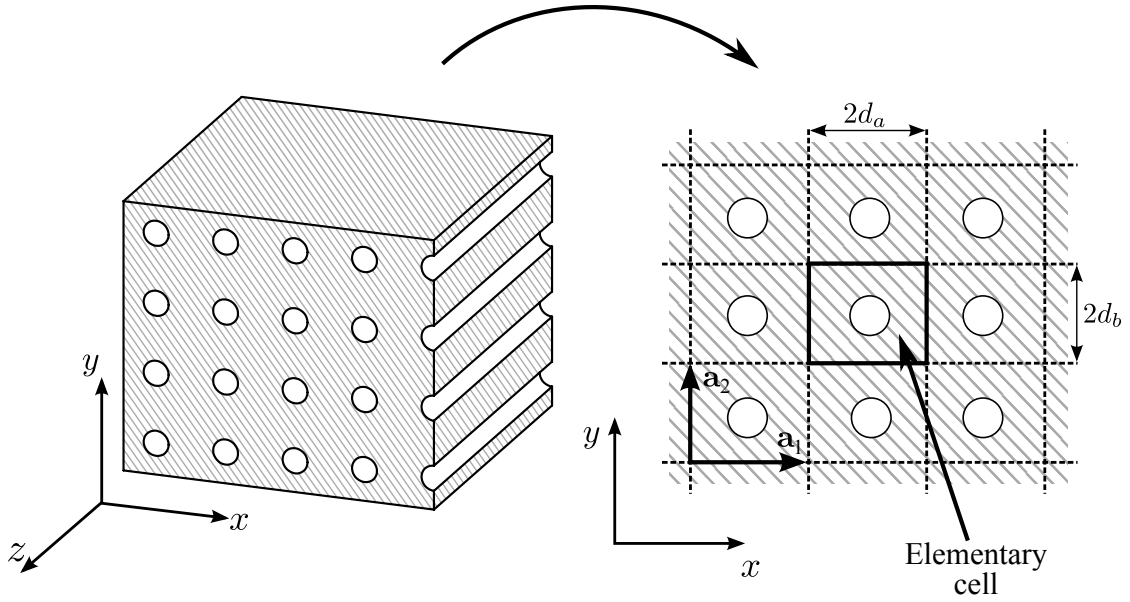


Figure 5.2. Depiction of a periodic material in 2D.

implying

$$w(\mathbf{x}) = u(\mathbf{x}+\mathbf{a})e^{-i\mathbf{k}\cdot(\mathbf{x}+\mathbf{a})} \ ,$$

and substituting this in (5.2), we get

$$u(\mathbf{x}+\mathbf{a}) = u(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{a}} \ . \tag{5.3}$$

Equation (5.3) is termed Bloch periodicity or Bloch-periodic condition. Besides the form in equation (5.1), we can also have an equation of the form

$$Lu(\mathbf{x}) = \omega^2 u(\mathbf{x}) + \mathbf{f}(\omega) \ , \tag{5.4}$$

where $\mathbf{f}(\omega)$ is a harmonic body force with circular frequency ω that presents the same periodicity that the lattice, see [14].

5.1.1 Bloch theorem in elastodynamics

Let us consider the wave propagation in an elastic, linear isotropic material. This is described using the Navier-Cauchy equation in the frequency domain (neglecting body forces)

$$(\lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) + \mu\nabla\times(\nabla\times\mathbf{u}) = -\omega^2\rho\mathbf{u} \ .$$

The displacement vector can be expressed as $\mathbf{u} = \nabla\varphi + \nabla\times\boldsymbol{\psi}$ due to Helmholtz decomposition theorem [5, 6], and the potentials $\varphi, \boldsymbol{\psi}$ are the scalar and vector potential, respectively. These

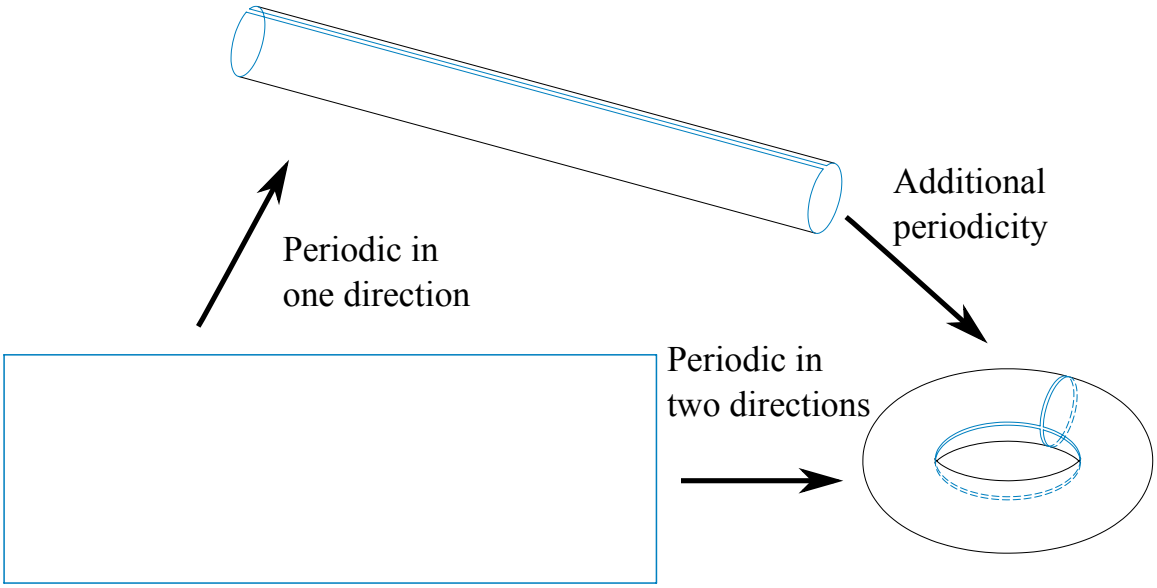


Figure 5.3. Topology for a rectangular region with periodic boundary conditions.

potential verify

$$\nabla^2\varphi = -\frac{\omega^2}{\alpha^2}\varphi \ ,$$
$$\nabla^2\boldsymbol{\psi} = -\frac{\omega^2}{\beta^2}\boldsymbol{\psi} \ ,$$

where α, β correspond with the speed for the longitudinal and transverse waves.

Since the equations for the potentials $\varphi, \boldsymbol{\psi}$ are wave equations, they satisfy Bloch theorem and then

$$\varphi(\mathbf{x}) = \varphi(\mathbf{x}+\mathbf{a})e^{i\mathbf{k}\cdot\mathbf{a}},$$
$$\boldsymbol{\psi}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x}+\mathbf{a})e^{i\mathbf{k}\cdot\mathbf{a}} \ ,$$

and

$$\mathbf{u}(\mathbf{x}) = \nabla\varphi(\mathbf{x}) + \nabla\times\boldsymbol{\psi}(\mathbf{x}) = e^{i\mathbf{k}\cdot\mathbf{a}}\nabla\varphi(\mathbf{x}) + e^{i\mathbf{k}\cdot\mathbf{a}}\nabla\times\boldsymbol{\psi}(\mathbf{x}),$$
$$= e^{i\mathbf{k}\cdot\mathbf{a}}\left[\nabla\varphi(\mathbf{x}) + \nabla\times\boldsymbol{\psi}(\mathbf{x})\right] = e^{i\mathbf{k}\cdot\mathbf{a}}\mathbf{u}(\mathbf{x}+\mathbf{a}) \ .$$

Thus, the Bloch theorem is satisfied for the displacement field, i.e.

$$\mathbf{u}(\mathbf{x}) = e^{i\mathbf{k}\cdot\mathbf{a}}\mathbf{u}(\mathbf{x}+\mathbf{a}) \ .$$

5.2 Bloch theorem as boundary conditions

If we start from the reduced wave equation [15] the boundary value problem would be given by

$$\begin{aligned}
&(\lambda+2\mu)\nabla(\nabla\cdot\mathbf{u})+\mu\nabla\times(\nabla\times\mathbf{u})=-\omega^2\rho\mathbf{u} && \text{in } \Omega,\\
&\mathbf{u}(\mathbf{x}+\mathbf{a})=\mathbf{u}(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{a}} && \text{in } \Gamma_u,\\
&\mathbf{t}(\mathbf{x}+\mathbf{a})=-\mathbf{t}(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{a}} && \text{in } \Gamma_t\ .
\end{aligned}$$

We will call this type of boundary conditions *Bloch-periodicity*[1]. The operator of this partial differential equation is self-adjoint and positive definite for Bloch-periodic conditions [16,17], thus the eigenvalues are real and positive. After discretization (using FEM), the resulting matrices are Hermitian and positive definite. What makes sense from a physical point of view, the energy of the system is always bigger than zero and the eigenvalues (squared frequencies) must be positive.

The problem can be equivalently formulated for the Bloch function $\mathbf{w}(\mathbf{x})$, where the Bloch-periodic conditions are replace by (plane) periodic conditions [18]. The equivalent boundary value problem is

$$\begin{aligned}
&B\mathbf{w}=-\omega^2\rho\mathbf{w} && \text{in } \Omega,\\
&\mathbf{w}(\mathbf{x}+\mathbf{a})=\mathbf{w}(\mathbf{x}) && \text{in } \Gamma_u,\\
&\frac{\partial\mathbf{w}}{\partial\hat{\mathbf{n}}}(\mathbf{x}+\mathbf{a})=\frac{\partial\mathbf{w}}{\partial\hat{\mathbf{n}}}(\mathbf{x}) && \text{in } \Gamma_t\ ,
\end{aligned}$$

where B is the operator

$$B=(\lambda+\mu)\left[\nabla\nabla\cdot+i\{\mathbf{k}\nabla\cdot+\mathbf{k}\cdot\nabla+\mathbf{k}\times\nabla\times\}\right]()-\mu\left[\nabla^2+2i\mathbf{k}\cdot\nabla-\|\mathbf{k}\|^2\right]()\ .$$

5.3 Variational form for elastodynamics with Bloch-periodicity

Let us start from the momentum equation in the frequency domain

$$\sigma_{rs,s}+f_r(\omega)=-\rho\omega^2u_r,\ \text{in } \Omega,$$

where f_r is a harmonic force in the direction r, $\sigma_{rs}=C_{rskl}\epsilon_{kl}$ and $\epsilon_{kl}=1/2(u_{k,l}+u_{l,k})$. The boundary conditions are

$$u_r(x_s+a_s)=u_r(x_s)e^{ik_sa_s}\ \text{in } \Gamma_u, \tag{5.5a}$$
$$t_r(x_s+a_s)=-t_r(x_s)e^{ik_sa_s}\ \text{in } \Gamma_t, \tag{5.5b}$$

being t_r the traction on the surface of the cell.

[1] Periodic boundary conditions are a particular case, where the factor $e^{i\mathbf{k}\cdot\mathbf{a}}=1$.

Now, let us multiply the momentum equation by the complex conjugate of an arbitrary function v_r and integrate over the domain[2]

$$\int_\Omega v_r^*[\sigma_{rs,s} + f_r(\omega)]d\Omega = -\omega^2 \int_\Omega \rho v_r^* u_r d\Omega \ ,$$

the complex conjugate is denoted by $*$. Then, using the constitutive relation

$$\int_\Omega v_r^* C_{rskl}\epsilon_{kl,s} d\Omega + \int_\Omega v_r^* f_r d\Omega = -\omega^2 \int_\Omega \rho v_r^* u_r d\Omega \ ,$$

and using divergence theorem

$$-\int_\Omega v_r, s^* C_{rskl}\epsilon_{kl} d\Omega + \int_\Omega v_r^* t_r(n_s) d\Gamma + \int_\Omega v_r^* f_r d\Omega = -\omega^2 \int_\Omega \rho v_r^* u_r d\Omega \ ,$$

with $t_r^{n_s} = C_{rskl}\epsilon_{kl} n_s$, if we take $v_r = u_r$ and consider that the tensor ϵ_{rs} is symmetric, we obtain

$$-\int_\Omega \epsilon_{rs} C_{rskl}\epsilon_{kl} d\Omega + \int_\Gamma u_r^* t_r(n_s) d\Gamma \int_\Omega u_r^* f_r(\omega) d\Omega = -\omega^2 \int_\Omega \rho u_r^* u_r d\Omega \ .$$

And neglecting the body forces, the Energy Functional for this problem is written as

$$\Pi(\omega) = \int_\Omega \epsilon_{rs}^*(\mathbf{x}) C_{rskl}(\mathbf{x})\epsilon_{kl}(\mathbf{x}) d\Omega - \omega^2 \int_\Omega \rho(\mathbf{x}) u_r^*(\mathbf{x}) u_r(\mathbf{x}) d\Omega - \int_\Gamma u_r^*(\mathbf{x}) t_r(\mathbf{x}) d\Gamma \ . \tag{5.6}$$

The first term in the equation (5.6) is the deformation energy inside the domain, the second term corresponds with the kinetic energy for harmonic motion with angular frequency ω, the last term is the deformation energy on the boundary of the domain. The boundary term is related with the imposition of natural boundary conditions (Neumann BCs) in the formulation of the FEM. In this case the Bloch-periodic conditions for the forces are implicitly satisfied in the variational form, as we will see in detail.

Since the domain is a single cell of a lattice, it should form a tessellation [3]. Hence, the boundaries can be grouped by pairs, i.e., each side (*reference side*) should have an opposite side (*image side*), so the last term in (5.6)can be rewritten as

$$\int_\Gamma u_r^*(\mathbf{x}) t_r(\mathbf{x}) d\Gamma = \sum_l \int_{\Gamma_l} [u_r^*(\mathbf{x}) t_r(\mathbf{x}) + u_r^*(\mathbf{x} + \mathbf{a_l}) t_r(\mathbf{x} + \mathbf{a}_l)] d\Gamma \ ,$$

[2] In the deduction of weak forms for FEM it is customary to multiply by a function v_r that is real. In this case the function is complex; in order to satisfy the properties for an inner product, we need to multiply by the complex conjugate (see for example [6, 19]).

[3] A tessellation is a regular pattern that covers all the space with no overlaps and no gaps.

where the index l refers to each one of the pairs of opposite sides in the boundary. Thus

$$\int_\Gamma u_r^*(\mathbf{x}) t_r(\mathbf{x}) d\Gamma = \sum_l \int_{\Gamma_l} [u_r^*(\mathbf{x}) t_r(\mathbf{x}) + e^{-i\mathbf{k}\cdot\mathbf{a}_l} u_r^*(\mathbf{x}) t_r(\mathbf{x}+\mathbf{a}_l)] d\Gamma \ ,$$
$$\int_\Gamma u_r^*(\mathbf{x}) t_r(\mathbf{x}) d\Gamma = \sum_l \int_{\Gamma_l} u_r^*(\mathbf{x}) [t_r(\mathbf{x}) + e^{-i\mathbf{k}\cdot\mathbf{a}_l} t_r(\mathbf{x}+\mathbf{a}_l)] d\Gamma \ ,$$

and the expression in square brackets is the condition (5.5b), that is equal to zero. This variational form can be computed for other operators that have similar algebraic properties to wave equations and are positive definite [12, 16, 17].

5.4 Implementation

5.4.1 Bloch-periodicity in FEM

The Bloch-periodic conditions in a Finite Element problem or, more generally, in a discrete problem can be formulated as a set of relations between the degrees of freedom of opposite sides of the unit cell.

In three dimension, and periodicity or Bloch-periodicity in the three of them, we can depict the original and reduced cell in Figure 5.5. It should be highlighted that in 3D it is possible to have periodicity in 1, 2 or 3 dimensions, depending on the nature of the problem of interest.

Let us start from the original problem

$$[K - \omega^2 M]\{u\} = \{f\} \ , \tag{5.7}$$

where

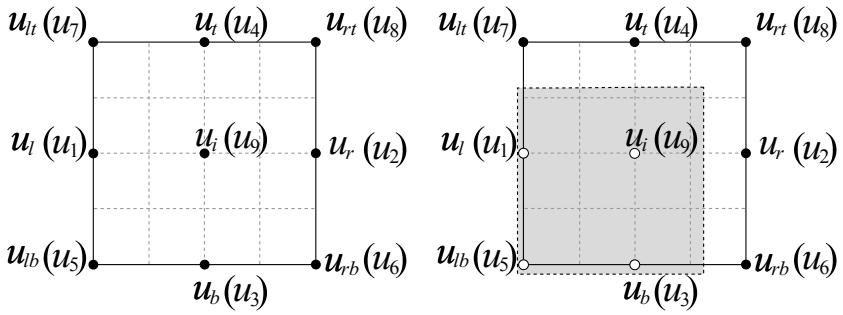
$$u = [u_l \ u_r \ u_b \ u_t \ u_{lb} \ u_{rb} \ u_{lt} \ u_{rt} \ u_i]^T = [u_1 \ u_2 \ u_3 \ u_4 \ u_5 \ u_6 \ u_7 \ u_8 \ u_9]^T \ ,$$
$$f = [f_l \ f_r \ f_b \ f_t \ f_{lb} \ f_{rb} \ f_{lt} \ f_{rt} \ f_i]^T = [f_1 \ f_2 \ f_3 \ f_4 \ f_5 \ f_6 \ f_7 \ f_8 \ f_9]^T \ ,$$

the subindices are describe in Figure 5.4.

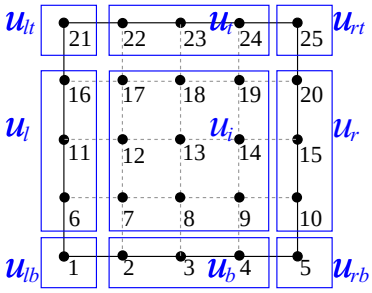
From Bloch theorem we know

$$u_2 = e^{i\psi_x} u_1, \quad u_4 = e^{i\psi_y} u_3, \quad u_6 = e^{i\psi_x} u_5,$$
$$u_7 = e^{i\psi_y} u_5, \quad u_8 = e^{i(\psi_x+\psi_y)} u_5,$$

with $\psi_x = 2k_x d_a$ and $\psi_y = 2k_y d_b$ phase shifts in x and y, and $(k_x, k_y) = \mathbf{k}$ wave vector. If we think



(a) Original cell. **(b)** *Reduced* cell.



(c) The black circles now correspond to single degrees of freedom. Notice that the nodal numbering does not match the original numbering for groups of degrees of freedom.

Figure 5.4. Relevant sets of DOF for an schematic unitary cell in 2D discretized with several finite elements. Each circle represents a family of degrees of freedom for a typical mesh and not a single degree of freedom. In part a) we show all the degrees of freedom for the unitary cell before imposing the relevant Bloch-periodic boundary conditions. Part b) shows the reduced cell where the white circles enclosed by the dark square represent the *reference* nodes containing the information from the *image* nodes which will be eventually deleted from the system. Part c) shows an example of node grouping for a mesh of 4×4 elements.

about this relations as a linear transformation we can write

$$\underbrace{\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{Bmatrix}}_{u} = \underbrace{\begin{bmatrix} I_{nl} & 0 & 0 & 0 \\ I_{nl}e^{i\psi_x} & 0 & 0 & 0 \\ 0 & I_{nb} & 0 & 0 \\ 0 & I_{nb}e^{i\psi_y} & 0 & 0 \\ 0 & 0 & I_2 & 0 \\ 0 & 0 & I_2 e^{i\psi_x} & 0 \\ 0 & 0 & I_2 e^{i\psi_y} & 0 \\ 0 & 0 & I_2 e^{i(\psi_x+\psi_y)} & 0 \\ 0 & 0 & 0 & I_{ni} \end{bmatrix}}_{T} \underbrace{\begin{Bmatrix} u_1 \\ u_3 \\ u_5 \\ u_9 \end{Bmatrix}}_{u_R}, \tag{5.8}$$

being I_n identity matrices of size n (equal to the number of degrees of freedom associated to each group). We can write the equation (5.8) in compact form as $u = Tu_R$, and substituting (5.8) in (5.7) we obtain

$$[KT]u_R = \omega^2[MT]u_R \ . \tag{5.9}$$

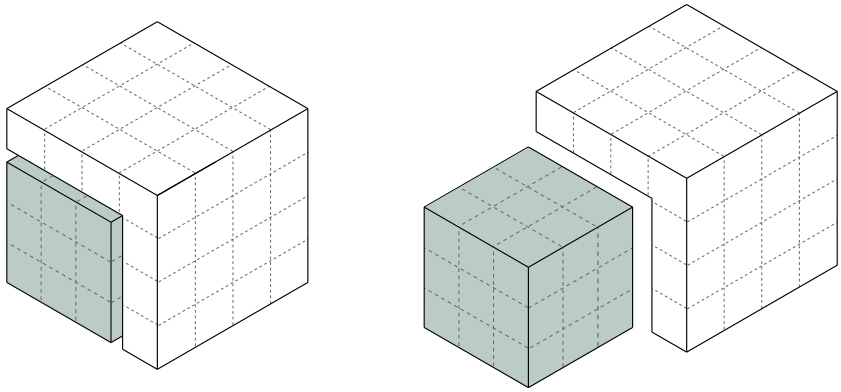


Figure 5.5. Three dimensional unitary cell with a $4 \times 4 \times 4$ mesh. It is partitioned into the image DOF and remaining ones. The reduced cell is highlighted.

Also, from the Bloch theorem we have the following equilibrium conditions

$$f_2 + e^{i\psi_x} f_1 = 0, \quad f_4 + e^{i\psi_y} f_3 = 0,$$
$$f_8 + e^{i\psi_x} f_6 + e^{i\psi_y} f_7 + e^{i(\psi_x+\psi_y)} f_5 = 0 \ .$$

And, again, thinking this relations as another linear transformation

$$\underbrace{\begin{Bmatrix} 0 \\ 0 \\ 0 \\ f_9 \end{Bmatrix}}_{f_R} = \underbrace{\begin{bmatrix} I_{nl} & I_{nl}e^{-i\psi_x} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{nb} & I_{nb}e^{-i\psi_y} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_2 & I_2e^{-i\psi_x} & I_2e^{-i\psi_y} & I_2e^{-i(\psi_x+\psi_y)} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_{ni} \end{bmatrix}}_{T^H} \underbrace{\begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{Bmatrix}}_{f} \ . \tag{5.10}$$

Now, we multiply (5.9) by T^H, where T^H is the Hermitian transpose of T,

$$\left[\underbrace{T^H K T}_{K_R} - \omega^2 \underbrace{T^H K T}_{M_R} \right] \{u_R\} = \{f_R\} \ . \tag{5.11}$$

The new system of equations is of size $n \times n$, being n the number of independent variables in u_R, and the matrices K_R, M_R are Hermitian. This conditions is derived from the fact that the original matrices K, M are real and symmetric. Neglecting the body forces f_R we obtain the generalized eigenvalue problem for the reduced system

$$[K_R]\{u_R\} = \omega^2 [M_R]\{u_R\} \ . \tag{5.12}$$

The implementation of the periodic and Bloch-periodic boundary conditions can be achieved in two different ways:

- changing the connectivity, and the respective interpolation functions (see Figure 5.6); or
- assembling the mass and stiffness matrices assuming they do not have boundary conditions and then, using elementary row operations and elementary column operations, obtain the matrices with the desired conditions.

The modification over the mesh is well illustrated in Figure 5.6, where the continuity is warranted in the periodic case and a phase shift appears for the Bloch-periodic one.

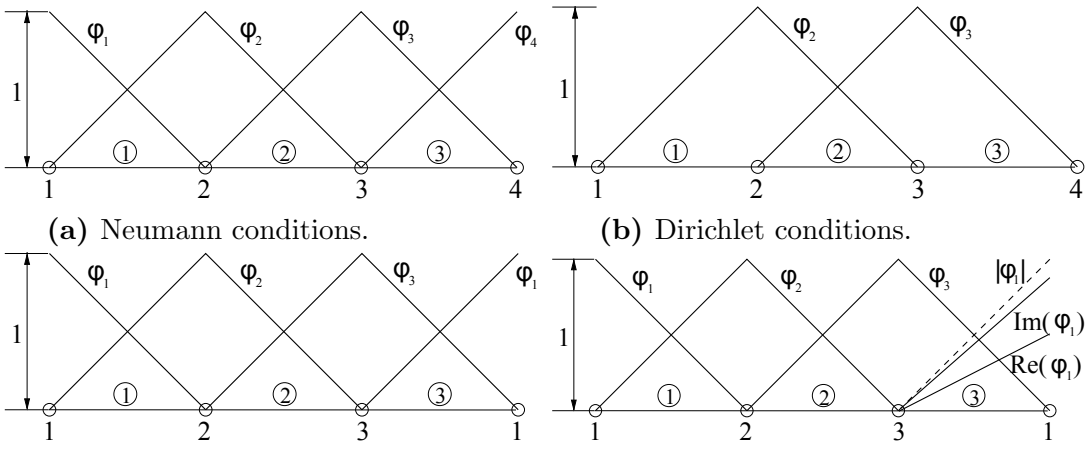


Figure 5.6. Basis functions for linear finite elements. (a) Neumann conditions; (b) Dirichlet conditions; (c) Periodic conditions; and (d) Bloch-periodic conditions. The real and imaginary part of φ_1 are illustrated in the basis with Bloch conditions in (d) for a phase shifts of $ka = \pi/3(\omega = [0, a])$. The modulus of φ_1 is the same at $x = 0$ and at $x = a$.

When obtaining the matrices without boundary conditions, the *reduced* matrices can be obtained in two different ways. One of them is applying elementary operations. We call the nodes of one side *reference nodes* and the corresponding opposite nodes *image nodes*. The procedure to obtain the reduced matrices is [19, 20]

- Multiply the elements of A_{kl} associated with a boundary node by $f_k^* f_l$.
- For each row i of A associated with a reference node, add all the rows k associated with the corresponding image nodes. For each column j associated with a reference node, add all the columns l associated with the corresponding image nodes.
- Eliminate all the rows k and columns l associates with image nodes.

The factor $f_j = e^{i\mathbf{k}\cdot\mathbf{x}_j}$, where $\mathbf{x}_j$ are the coordinates of the jth node and f^* is the complex conjugate of f. A complete description is presented in algorithm 7.

Another way to impose the Bloch-periodic conditions is to *assemble* the transformation matrix T and compute the matrix multiplications shown in (5.11). After applying this procedure the *image nodes* are removed from the system of equations and they are condensed in the corresponding reference nodes [14, 21], and the equation identifiers (equations enumeration) have changed. The procedure to assemble the matrix T is:

- Find the new identifier for the reference nodes after eliminating the image nodes.
- For reference and interior nodes correspond a 1 in the matrix. The row is the current identifier and the column is the future one.
- For image nodes correspond the complex number $e^{ik_x(x_{\text{img}}-x_{\text{ref}})}e^{ik_y(y_{\text{img}}-y_{\text{ref}})}$ in the matrix; being $x_{\text{ref}}, x_{\text{img}}, y_{\text{ref}}, y_{img}$ the coordinates x, y for the reference an image nodes. The row is the identifier for that image node and the column is the future identifier for the corresponding reference node.
- The remaining entries of the matrix are zeros.

This procedure is equivalent to arrange the elementary row/column operations for the procedure described above in a matrix [22]. This procedure is described with further insight in Algorithm 8.

5.4.2 Real algebra implementation

Following [23], we can formulate the problem using just real algebra. This allow to implement the Bloch-periodicity in *standard finite element programs* like Abaqus/Calculix and FEAP [24–26], that do not allow to impose conditions with complex valued quantities. The idea is to duplicate the degrees of freedom, i.e., to have two meshes: one for real values and the other one for imaginary ones (see Figure 5.7)

We can express the values in boundary as

$$\begin{aligned} u_r(x_s) &= u_r^{\text{Re}}(x_s) + iu_r^{\text{Im}}(x_s) \\ t_r(x_s) &= t_r^{\text{Re}}(x_s) + it_r^{\text{Im}}(x_s) \ , \end{aligned}$$

and replacing these in (5.5), we obtain

$$u_r^{\text{Re}}(x_s) = u_r^{\text{Re}}(x_s + a_s)\cos(k_s a_s) + u_r^{\text{Im}}(x_s + a_s)\sin(k_s a_s) \tag{5.13a}$$

$$u_r^{\text{Im}}(x_s) = u_r^{\text{Im}}(x_s + a_s)\cos(k_s a_s) - u_r^{\text{Re}}(x_s + a_s)\sin(k_s a_s) \ , \tag{5.13b}$$

and

$$t_r^{\text{Re}}(x_s + a_s) = -t_r^{\text{Re}}(x_s)\cos(k_s a_s) + t_r^{\text{Im}}(x_s)\sin(k_s a_s) \tag{5.14a}$$

$$t_r^{\text{Im}}(x_s + a_s) = -t_r^{\text{Im}}(x_s)\cos(k_s a_s) - t_r^{\text{Re}}(x_s)\sin(k_s a_s) \ . \tag{5.14b}$$

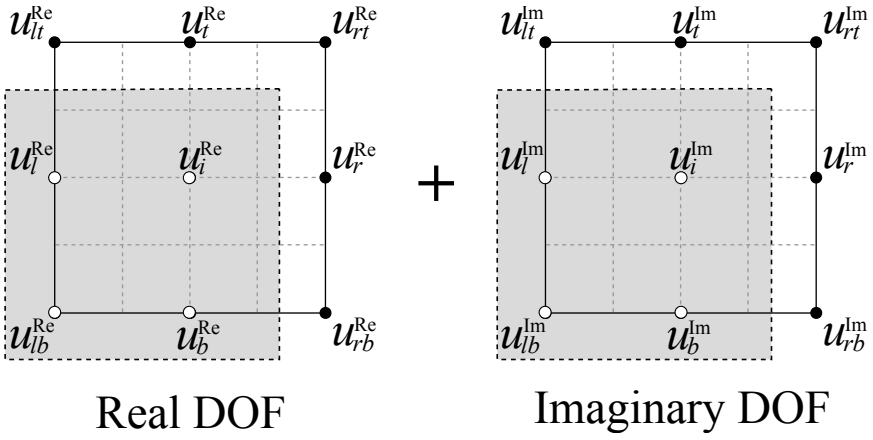


Figure 5.7. Mesh for the real algebra implementation.

In this case, the displacements and forces are arranged in block vectors as

$$\begin{aligned}\mathbf{U} &= \left[\mathbf{U}^{\mathrm{Re}}\ \mathbf{U}^{\mathrm{Im}}\right]^T \\ \mathbf{F} &= \left[\mathbf{F}^{\mathrm{Re}}\ \mathbf{F}^{\mathrm{Im}}\right]^T\ ,\end{aligned}$$

with

$$\begin{aligned}\mathbf{U}^{\mathrm{Re}} &= \left[\mathbf{u}_l^{\mathrm{Re}}\ \mathbf{u}_r^{\mathrm{Re}}\ \mathbf{u}_b^{\mathrm{Re}}\ \mathbf{u}_t^{\mathrm{Re}}\ \mathbf{u}_{lb}^{\mathrm{Re}}\ \mathbf{u}_{rb}^{\mathrm{Re}}\ \mathbf{u}_{lt}^{\mathrm{Re}}\ \mathbf{u}_{rt}^{\mathrm{Re}}\ \mathbf{u}_i^{\mathrm{Re}}\right]^T \\ \mathbf{U}^{\mathrm{Im}} &= \left[\mathbf{u}_l^{\mathrm{Im}}\ \mathbf{u}_r^{\mathrm{Im}}\ \mathbf{u}_b^{\mathrm{Im}}\ \mathbf{u}_t^{\mathrm{Im}}\ \mathbf{u}_{lb}^{\mathrm{Im}}\ \mathbf{u}_{rb}^{\mathrm{Im}}\ \mathbf{u}_{lt}^{\mathrm{Im}}\ \mathbf{u}_{rt}^{\mathrm{Im}}\ \mathbf{u}_i^{\mathrm{Im}}\right]^T \\ \mathbf{F}^{\mathrm{Re}} &= \left[\mathbf{f}_l^{\mathrm{Re}}\ \mathbf{f}_r^{\mathrm{Re}}\ \mathbf{f}_b^{\mathrm{Re}}\ \mathbf{f}_t^{\mathrm{Re}}\ \mathbf{f}_{lb}^{\mathrm{Re}}\ \mathbf{f}_{rb}^{\mathrm{Re}}\ \mathbf{f}_{lt}^{\mathrm{Re}}\ \mathbf{f}_{rt}^{\mathrm{Re}}\ \mathbf{f}_i^{\mathrm{Re}}\right]^T \\ \mathbf{F}^{\mathrm{Im}} &= \left[\mathbf{f}_l^{\mathrm{Im}}\ \mathbf{f}_r^{\mathrm{Im}}\ \mathbf{f}_b^{\mathrm{Im}}\ \mathbf{f}_t^{\mathrm{Im}}\ \mathbf{f}_{lb}^{\mathrm{Im}}\ \mathbf{f}_{rb}^{\mathrm{Im}}\ \mathbf{f}_{lt}^{\mathrm{Im}}\ \mathbf{f}_{rt}^{\mathrm{Im}}\ \mathbf{f}_i^{\mathrm{Im}}\right]^T\ .\end{aligned}$$

For the two dimensional example presented above the equations read

$$\begin{aligned}\mathbf{u}_r^{\mathrm{Re}} &= \mathbf{u}_l^{\mathrm{Re}}\cos(\psi_x) - \mathbf{u}_l^{\mathrm{Im}}\sin(\psi_x) \\ \mathbf{u}_r^{\mathrm{Im}} &= \mathbf{u}_l^{\mathrm{Im}}\cos(\psi_x) + \mathbf{u}_l^{\mathrm{Re}}\sin(\psi_x) \\ \mathbf{u}_t^{\mathrm{Re}} &= \mathbf{u}_b^{\mathrm{Re}}\cos(\psi_y) - \mathbf{u}_b^{\mathrm{Im}}\sin(\psi_y) \\ \mathbf{u}_t^{\mathrm{Im}} &= \mathbf{u}_b^{\mathrm{Im}}\cos(\psi_y) + \mathbf{u}_b^{\mathrm{Re}}\sin(\psi_y) \\ \mathbf{u}_{rb}^{\mathrm{Re}} &= \mathbf{u}_{lb}^{\mathrm{Re}}\cos(\psi_x) - \mathbf{u}_{lb}^{\mathrm{Im}}\sin(\psi_x) \\ \mathbf{u}_{rb}^{\mathrm{Im}} &= \mathbf{u}_{lb}^{\mathrm{Im}}\cos(\psi_x) + \mathbf{u}_{lb}^{\mathrm{Re}}\sin(\psi_x) \\ \mathbf{u}_{lt}^{\mathrm{Re}} &= \mathbf{u}_{lb}^{\mathrm{Re}}\cos(\psi_y) - \mathbf{u}_{lb}^{\mathrm{Im}}\sin(\psi_y) \\ \mathbf{u}_{lt}^{\mathrm{Im}} &= \mathbf{u}_{lb}^{\mathrm{Im}}\cos(\psi_y) + \mathbf{u}_{lb}^{\mathrm{Re}}\sin(\psi_y) \\ \mathbf{u}_{rt}^{\mathrm{Re}} &= \mathbf{u}_{lb}^{\mathrm{Re}}\cos(\psi_x+\psi_y) - \mathbf{u}_{lb}^{\mathrm{Im}}\sin(\psi_x+\psi_y) \\ \mathbf{u}_{rt}^{\mathrm{Im}} &= \mathbf{u}_{lb}^{\mathrm{Im}}\cos(\psi_x+\psi_y) + \mathbf{u}_{lb}^{\mathrm{Re}}\sin(\psi_x+\psi_y)\ ,\end{aligned}$$

where ψ_x and ψ_y are the phase shifts in x and y. Similar expressions hold for the forces. Although the expressions are written explicitly here, they are all contained compactly in (5.13) and (5.14). The phase shift is $\psi = k_j d_j$, with k_j the components of the wave vector and d_j the distance between the image and reference node in j direction.

Algorithm 7: Imposition of Bloch's conditions using elemental row operations.

Input : A: Original matrix
k_x, k_y: wave-vector components
nodes: array with nodal coordinates
image, reference: array with corresponding image and reference nodes
reference2: array with reference nodes without repetition

Output: Matrix A with imposed Bloch's conditions

for *i=1* **to** *n_img* **do** `// Phase shift for image nodes`

img_index ← image[i]
x_img ← nodes[img_index,1]
y_img ← nodes[img_index,2]
fac ← $e^{ik_x \mathrm{x\_img}} e^{ik_y \mathrm{y\_img}}$
A[img_index, :] ← A[img_index, :]fac
A[:, img_index] ← A[:, img_index]fac*

end

for *i=1* **to** *n_ref* **do** `// Phase shift for reference nodes`

ref_index ← reference2[i]
x_ref ← nodes[ref_index,1]
y_ref ← nodes[ref_index,2]
fac ← $e^{ik_x \mathrm{x\_ref}} e^{ik_y \mathrm{y\_ref}}$
A[ref_index, :] ← A[ref_index, :]fac
A[:, ref_index] ← A[:, ref_index]fac*

end

for *i=1* **to** *n_cond* **do** `// Addition of image row/columns to reference row/colums`

img_index ← image[i]
ref_index ← reference[i]
A[ref_index, :] ← A[ref_index, :] + A[img_index, :]
A[:, ref_index] ← A[:, ref_index] + A[:,img_index, :]

end

for *i=1* **to** *n_imag* **do** `// Elimination of image row/columns`

img_index ← image[i]
A[img_index, :] ← erase row img_index from A
A[img_index, :] ← erase column img_index from A

end

Algorithm 8: Impose Bloch-periodic conditions using tranformation matrices.

Input : A: Original matrix
k_x, k_y: wave-vector components
nodes: array with nodal coordinates
image, reference: array with corresponding image and reference nodes
reference2: array with reference nodes without repetition

Output: Matrix A with imposed Bloch's conditions

index_vec $\leftarrow \vec{0}$ `// Array with new equation identifiers`
for *i=1* **to** *ncond* **do**
  index_vec[image[i]] $\leftarrow$ - reference[i] `// Negative of the current identifier for the reference node`
end
for *i=1* **to** *ndof* **do** `// Assign new identifiers`

  cont $\leftarrow$ 0
  for *j=1* **to** *ncond* **do** `// How many image nodes before the current one`

    if *index_vec*[*i*] < 0 **then**
      if *image*[*j*] $<$ |*index_vec*[*i*]| **then**
        cont $\leftarrow$ cont + 1
      end
    else if *image*[*j*] $< i$ **then**
      cont $\leftarrow$ cont + 1
    end
  end
  if *index_vec*[*i*] < 0 **then** `// Assign new identifiers`

    index_vec[i] $\leftarrow$ index_vec[i] + cont
  else
    index_vec[i] $\leftarrow$ i - cont
  end
end
T $\leftarrow$ 0; index_ref $\leftarrow$ - index_vec
for *i=1* **to** *ndof* **do**
  if *index_vec*[*i*] > 0 **then**
    j $\leftarrow$ index_vec[i]
    T[i,j] $\leftarrow$ 1
  else
    j $\leftarrow$ |index_vec[i]|
    i_ref $\leftarrow$ index_ref[i]; i_img $\leftarrow$ i
    x_img $\leftarrow$ coords[i_img,1]; y_img $\leftarrow$ coords[i_img,2]
    x_ref $\leftarrow$ coords[i_ref,1]; y_ref $\leftarrow$ coords[i_ref,2]
    fac $\leftarrow$ $e^{i\psi_x(\text{x\_img - x\_ref})} e^{i\psi_y(\text{y\_img - y\_ref})}$
    T $\leftarrow$ fac
  end
end
A $\leftarrow$ T^HAT

Algorithm 9: Generate the list of Multipoint constraints for the Bloch-periodic imposition using real algebra.

Input : k_x, k_y, k_z: wave-vector components
nodes: array with nodal coordinates
image, reference: array with corresponding image and reference nodes
Output: List of Multipoint constraints

ncond ← Size of image
for *i=1* **to** *ncond* **do**
  x_img, y_img, z_img ← coordinates of image[i]
  x_ref, y_ref, z_ref ← coordinates of reference[i]
  $\psi \leftarrow k_x$(x_img - x_ref) + k_y(y_img - y_ref) + k_z(z_img - z_ref)
  $u^{\mathrm{Re}}_{\mathrm{img}} \leftarrow u^{\mathrm{Re}}_{\mathrm{ref}} \cos\psi + u^{\mathrm{Im}}_{\mathrm{ref}} \sin\psi$
  $u^{\mathrm{Im}}_{\mathrm{img}} \leftarrow u^{\mathrm{Im}}_{\mathrm{ref}} \cos\psi - u^{\mathrm{Re}}_{\mathrm{ref}} \sin\psi$
end

Chapter 6

Computational aspects

6.1 Global assembly

In this section we will discuss the fundamental process of building the system of algebraic equations through the addition or assembly of elemental coefficient matrices as described by eq. (6.1) where K^G is the global coefficient matrix; k^i is the local elemental matrix for the i-th element; $\mathop{\mathbb{A}}_{i=1}^{Numel}$ is the assembly operator and i is an element index ranging between 1 and the total number of elements $Numel$ that conform the finite element model. Notice that the assembly operator is analogous to the sum operator commonly used in the representation of a series of N terms but it contains information indicating the position of each single term from the element coefficient matrix within the global system.

$$K^G = \mathop{\mathbb{A}}_{i=1}^{Numel} k^i. \tag{6.1}$$

In this section we will describe the fundamental algorithmic steps to perform that process. We first summarize the basic general steps and then illustrate the process through a simple example involving a $2D$ mesh of 4-noded elements. The section is complemented with the python scripts **GLOBAL.py** and the secondary module **assemutil.py**.

The assembly algorithm

The process of assembly of the elemental coefficient matrices into the global system involves (i) the identification of the active degrees of freedom (or equation numbers) assigned to each node in the mesh (ii) the identification of the relationship between the elemental degrees of freedom and the global degrees of freedom and (iii) the computation of the coefficient matrix for each element in the model.

The first step is easily accomplished by assigning a boundary condition flag to the degrees of freedom existing at each node. Here we use a 0 value to indicate an active degree of freedom and a -1 to indicate a prescribed degree of freedom. This information is stored in a boundary condition array IBC, of dimension $nn \times MDIM$, where nn corresponds to the total number of nodal points and $MDIM$ represents the problem dimensionality. The IBC array is first input by the user and later modified by the program in a process where the boundary condition flag is read and equations are counted and assigned according to the result. If the boundary condition flag is equal to 0 the program assigns an equation number while it produces a 0 value if the flag is equal to -1.

In the second step the nodes conforming each element are stored in a connectivity array $IELCON$ of dimension $Numel \times MxNNel$ where $Numel$ is the number of elements in the model and $MxNNel$ is the maximum number of nodes in a given element. Thus each row in the $IELCON$ array stores the nodal point data for the element. Each entry in the $IELCON$ array can now be directly translated into equation numbers using the processed IBC array. This results in the discrete version of the assembly operator $\mathop{\mathbb{A}}_{i=1}^{Numel}$ also called the assembly list or the DME operator in our codes.

In the final step the mesh is covered one element at a time and each elemental coefficient matrix is assembled into the global matrix as indicated by the DME operator. The actual computation of the elemental matrix is conducted by an element based subroutine, called here UEL, which may be different for each element in the mesh according to different kinematic o material assumptions. The complete process is summarized in line 10 where we describe for completeness additional steps involved in the finite element algorithm. We have introduced additional global and elemental arrays RHS^G and rhs^i respectively, storing the element nodal excitation and the resulting vector of global excitation. This vector is assembled simultaneously, and using the same data, with the global coefficient matrix. In the final step in the finite element algorithm the system of equations is solved. Notice that the essential boundary conditions were already considered during the assembly process and as a result the global algebraic system of equations is ready to be solved. Finally, after the system has been properly solved the nodal results are scattered through the elements in a process which is inverse to the assembly operation but that uses the same information contained

in the DME operator.

Algorithm 10: Summarized algorithm for the finite element method

Data: Finite element model
Result: Field function

READ IBC and $IELCON$ arrays ;
Compute modified IBC array ;
Compute DME operator (using IBC and $IELCON$ arrays);
$K^G \leftarrow 0.0$;

for $i \leftarrow 1$ *to* $Numel$ **do**
  Call UEL(element parameters;K^i) ;
  $K^G \leftarrow K^G + K^i$ (Assemble each k^i into K^G according to the DME operator);
  $RHS^G \leftarrow RHS^G + rhs^i$ (Assemble each rhs^i into RHS^G);
end

Solve $K^G U^G = RHS^G$;
Scatter solution to the elements

Sample problem

Consider the mesh shown in section 6.1. The problem parameters in this case are $nn = 9$, $Numel = 4$ and $MxNNel = 4$.

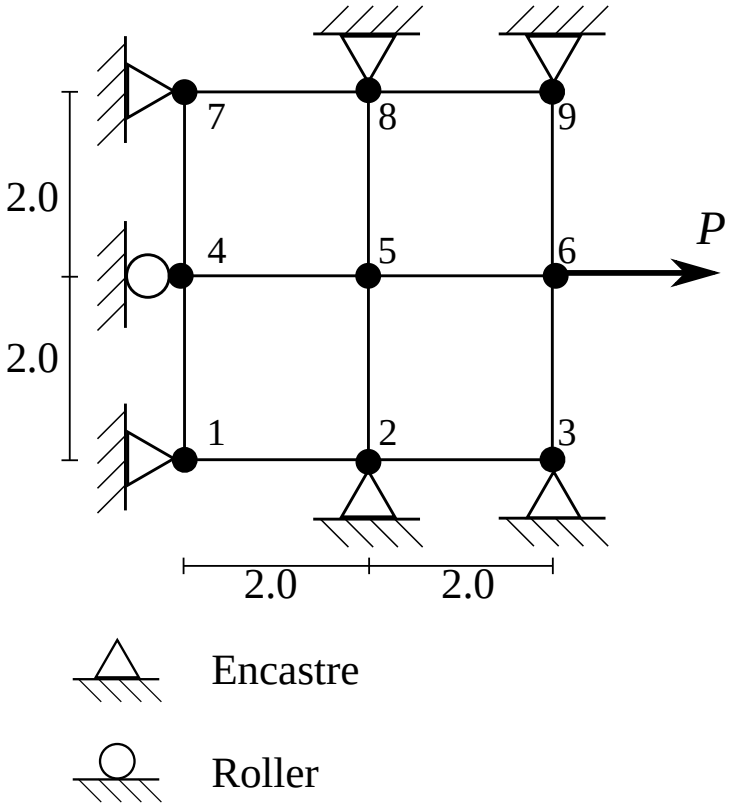


Figure 6.1. Finite element mesh of 4-noded elements. The global matrix can be assembled using the python scripts **GLOBAL.py** and the secondary module **assemutil.py**. The input files required to run the script are called **nodes.txt** and **eles.txt**

The array of boundary conditions as input from the user corresponds to

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

while its code-modified version reads

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

Notice that the program changes each -1 entry to a 0 valued entry and assigns an equation number to the position containing an original 0 entry during an equation-assigning and equation-counting operation. Similarly, the element connectivity for the current example read

$$IELCON = \begin{bmatrix} 1 & 2 & 5 & 4 \\ 2 & 3 & 6 & 5 \\ 4 & 5 & 8 & 7 \\ 5 & 6 & 9 & 8 \end{bmatrix}$$

The so-called DME operator is actually the same $IELCON$ but translated into equation numbers for each element. Accordingly in the current example we have

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

The definition of the element connectivity and its subsequent translation into degrees of freedom is carried out according to a local element definition like the one shown in section 6.1

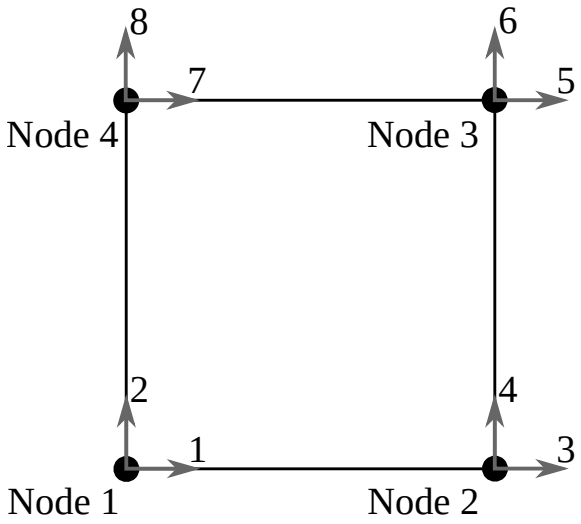


Figure 6.2. Local definition for a 4-noded element.

As a result the (i, j) entry in the DME corresponds to the global equation number associated with the local degree of freedom j of the i element. For instance the value of 4 stored at position $(2, 5)$ in the current DME indicates that the global equation 4 corresponds to the local equation 5 (column index) in element 2 (row index).

The assembly process is then conducted by identifying the relation between the entries in each row of the DME operator and the list of local degrees of freedom for the reference element shown in section 6.1. Accordingly, for element 2 it follows that the assembly of row 5 of the local stiffness matrix proceeds as follows

$$K^G_{4,4} \leftarrow K^G_{4,4} + k^2_{5,5}$$
$$K^G_{4,5} \leftarrow K^G_{4,5} + k^2_{5,6}$$
$$K^G_{4,3} \leftarrow K^G_{4,3} + k^2_{5,8}$$

In line 10 it must be noticed that the local stiffness matrix for each i element is obtained by the call to the local element subroutine UEL. In the script GLOBAL.py these elemental routines produce a fictitious matrix filled out with ones and the actual computation of the elemental coefficient matrix is discussed later.

6.2 Python implementations

6.3 Commercial codes and user subroutines

Chapter 7

Nonlinear Equilibrium Equations in a Finite Element Discretization

7.1 General Equilibrium Equations

In this section we develop the incremental equations needed in a general non-linear problem. First, geometric non-linearities valid for any stress-strain pair are considered. Second, focus is shifted towards the particular case of the Second Piola-Kirchoff-Green Lagrange Strain pair.

Let represent the current (deformed) configuration of a deformable medium with equilibrium equations and boundary conditions

$$\sigma_{ij,j} + f_i = 0 \tag{7.1}$$

$$t_i = \sigma_{ij}\hat{n}_j \tag{7.2}$$

where represents the Cauchy definition of stress (or force per unit of deformed surface) and the corresponding tractions vector at a surface with outward normal . In the equilibrium equations stated in (1) the domain is unknown which makes the problem inherently non-linear. On the other hand this equilibrium statement is mathematically indeterminate since 9 unknown stress components must be solved out of 6 field equations. The indeterminacy is destroyed after the problem is kinematically described and connected to the stress field via constitutive modeling which can involve additional sources of non-linearity. To summarize the problem is non-linear since the domain is unknown, this is what we typically call a Geometric Non-linearity and will be reflected in the mathematical description of changes in configuration.

7.1.1 Lagrangian description of the equilibrium equations

The conceptual definition of Cauchy stress is the only one useful for the engineer since it describes forces per unit deformed surface. This validity is clearly identified in the fact that the equilibrium equations stated in (1) have been formulated in the stressed deformed domain where the body is in fact in equilibrium. The difficulty associated with the unknown domain may be dealt with in alternative ways. In the case of a solid body it is convenient to refer everything to the undeformed or reference configuration (with known domain) and proceed from there using linearization which corresponds to a Total Lagrangian (TL) approach. It is then useful to understand the transformation of the problem to via pullback operations. This process implies the introduction or consideration of mathematically defined stress definitions and newly developed kinematic descriptions. In this section we consider such transformations first from a dynamic point of view and later using thermodynamic principles we address the problem of identifying the proper strain measures.

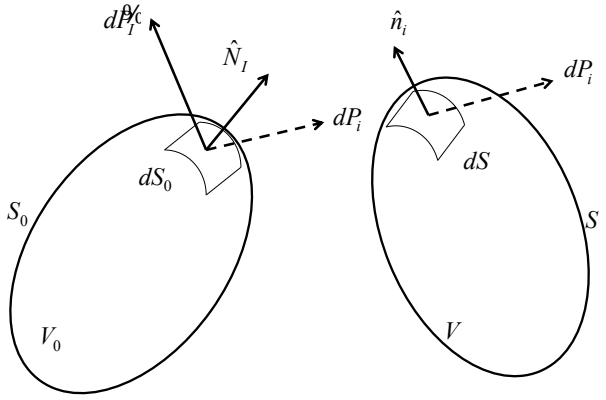


Figure 7.1. Definition of the natural domain

Nanson's formula

For the treatment that follows it will result useful to consider the relation between oriented differential surface elements in the reference and deformed configurations in the so-called Nanson's formula;

$$\hat{n}_i dS = \frac{\rho}{\rho_0} \hat{N}_I f_{Ii} dS_0 \tag{7.3}$$

Lagrangian stress definition

Let be the differential force associated to the Cauchy physical stress such

$$dP_i = t_i dS \equiv \sigma_{ji} n_j dS \tag{7.4}$$

Assume that can also be obtained in the undeformed configuration associated with a new traction definition;

$$dP_i = t_i^0 dS_0 \equiv T_{Ii}^0 N_I dS_0 \equiv \sigma_{ji} n_j dS \tag{7.5}$$

Using Nanson's formula it is possible to write;

$$T_{Ii}^0 N_I dS_0 \equiv \sigma_{ji} \frac{\rho_0}{\rho} N_I f_{Ij} dS_0 \tag{7.6}$$

Which simplifies into

$$T_{Ii}^0 = \frac{\rho_0}{\rho} f_{Ij} \sigma_{ji} \tag{7.7}$$

and this is the asymmetric First Piola-Kirchoff stress tensor.

Assume now that there is a pseudo-force in the undeformed configuration which results from a pullback operation on the physical force . Recalling the kinematic connection between the current and undeformed configurations

$$dX_I = f_{Ii} dx_i \tag{7.8}$$

where is the inverse deformation gradient, we can write for the force and pseudo-force vectors;

$$d\tilde{P}_I = f_{Ii} dP_i. \tag{7.9}$$

Now we can define a tractions vector in the undeformed configuration and associated with the pseudo-force like

$$d\tilde{P}_I = \tilde{t}_I dS_0 \equiv T_{JI} N_J dS_0 \tag{7.10}$$

where we have at the same time introduced the associated stress tensor. It then directly follows that;

$$T_{JI}N_J dS_0 = f_{Ii}dP_i \equiv f_{Ii}\sigma_{ji}n_j dS \tag{7.11}$$

once again using Nanson's formula we can write

$$T_{JI}N_J dS_0 = f_{Ii}\sigma_{ji}\frac{\rho_0}{\rho}N_J f_{Jj}dS_0 \tag{7.12}$$

and

$$T_{JI} = \frac{\rho_0}{\rho}f_{Jj}\sigma_{ji}f_{Ii} \tag{7.13}$$

which corresponds to the symmetric Second Piola-Kirchoff stress tensor.

For the derivations that follow and in the actual computational implementation it will be convenient to have the inverse relationships expressing the Cauchy stress tensor in terms of the First and Second Piola-Kirchoff stress definitions.

Once again recall that and use the definition found for the first PK stress tensor to write

7.2 Non-linear problems

In this section we will study the basic 1-single independent variable problem of numerically finding possible roots to the non-linear equation;

$$f(x) = 0 \tag{7.14}$$

and then we will proceed via generalization to the more realistic case of a system of non-linear equations of the form (29). This last general form will then be converted to the particular case of an equilibrated system of forces as encountered in the Finite Element Method or other numerical techniques appearing in the solution of continuum mechanics problems.

Since the problem is assumed non-linear the root finding process must use iterations. The general idea is that of establishing an initial trial solution and then proposed an efficient algorithm to improve the solution until some desired tolerance is satisfied.

In this section we will concentrate on Newton methods since most of the followed techniques in FEM analysis are either Newton or improved Newton Methods. A general and detailed survey of root finding and non-linear equation solving can be found in Press et al.

The basic idea of the Newton method is that of taking an initial guess ${}_iX$ (the trial solution), extending the tangent line at this point until it crosses zero and then taking as the next approximation ${}_{i+1}X$ the abscissa of that zero crossing, see fig. 7.2.

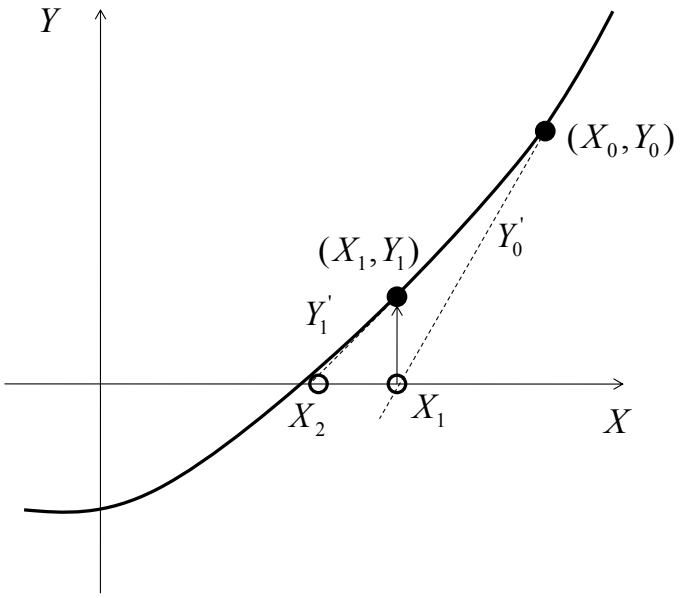


Figure 7.2. Fundamental Newton-Raphson iteration

The main disadvantage of the Newton method is that we formally need to obtain the function derivative $Y_i^{'}$ at every point.

Let δ be the correction to an approximation X to the actual root P such;

$$P = X + \delta \tag{7.15}$$

Using a Taylor series expansion for $f(P)$ in the neighborhood of X we have;

$$\begin{aligned} f(P) \equiv f(X+\delta) &= f(X) + f^{'}(X) + \frac{f^{''}(X)}{2}\delta^2 \\ f(P) \equiv f(X+\delta) &= f^{'}(X)\delta + \frac{f^{''}(X)}{2}\delta^2 \\ f(P) \equiv f(X+\delta) &= f(X) + f^{'}(X)(X-P) + \frac{f^{''}(X)}{2}(X-P)^2 \end{aligned} \tag{7.16}$$

Using $f(P) = 0$ and linearizing the Taylor series yields;

$$P \approx X - \frac{f(X)}{f'(X)} \tag{7.17}$$

From eq. (7.16) it is clear that the correction δ satisfies;

$$\delta = -\frac{f(X)}{f'(X)}$$

and we can write the general Newton-Raphson algorithm like;

$$\begin{aligned} {}_{i+1}X &= {}_{i}X + {}_{i}\delta \\ {}_{i}\delta &= -\frac{f({}_{i}X)}{f'({}_{i+1}X)} \end{aligned} \tag{7.18}$$

In an actual finite element algorithm we are to find an incrementally related history of discrete roots N_{roots} such

$${}^{I+1}X = {}^{I}X + {}^{I+1}\Delta X \tag{7.19}$$

and where

$$\begin{aligned} f({}^{I+1}X) &= 0 \\ f({}^{I}X) &= 0 \end{aligned} \tag{7.20}$$

Bibliography

[1] Bathe, Klaus Jurgen: *Finite Element Procedures*. Prentice Hall, 2nd edition, June 1995.

[2] Hibbitt, HD, BI Karlsson, and P Sorensen: *Abaqus theory manual, version 6.3*, 2006.

[3] Burden, R.L. and J.D. Faires: *Numerical Analysis*. Brooks/Cole, 9th edition, 2011.

[4] Press, William H: *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[5] Sepúlveda, Alonso: *Física Matemática*. Editorial Universidad de Antioquia, 1st edition, 2009.

[6] Arfken, George B., Hans J. Weber, and Frank Harris: *Mathematical Methods for Physicists*. Academic Press, 6th edition, 2005.

[7] Shames, Irving H: *Elastic and inelastic stress analysis*. CRC Press, 1997.

[8] Wikipedia: *Variational principle — wikipedia, the free encyclopedia*, 2015. `http://en.wikipedia.org/w/index.php?title=Variational_principle&oldid=646823082`, [Online; accessed 30-May-2015].

[9] Brillouin, Léon: *Wave propagation in periodic structures: electric filters and crystal lattices*. Courier Dover Publications, 2003.

[10] Geuzaine, C and J Remacle: *Gmsh Reference Manual. Gmsh 2.9: A Three-Dimensional Finite Element Mesh Generator With Built-in Pre-and Post-Processing Facilities*, April 2015. `http://geuz.org/gmsh/doc/texinfo/gmsh.pdf`.

[11] Meeker, David: *Finite Element Method Magnetics: Version 4.2 User's Manual*, 2010. `http://www.femm.info/Archives/doc/manual42.pdf`.

[12] Johnson, Steven G.: *Notes on the algebraic structure of wave equations*. Technical report, Massachusetts Institute of Technology, 2010.

[13] Kittel, Charles and Paul McEuen: *Introduction to solid state physics*, volume 8. Wiley New York, 1986.

[14] Langlet, Philippe: *Analyse de la Propagation des Ondes Acoustiques dans les Materiaux Periodiques a l'aide de la Methode des Elements Finis*. PhD thesis, L'Universite de Valenciennes et du Hainaut-Cambresis, 1993.

[15] Achenbach, J.D.: *Wave Propagation in Elastic Solids*. North Holland Publishing Company, 1973.

[16] Reddy, J.N.: *Applied Functional Analysis and Variational Methods in Engineering*. Krieger Publishing, 1st edition, 1991.

[17] Kreyzsig, Erwin: *Introductory Functional Analysis with Applications*. Wiley, 1st edition, 1989.

[18] Gockenbach, Mark S.: *Partial Differential Equations: Analytical and Numerical Methods*. SIAM, 2002.

[19] Sukumar, N. and J. E. Pask: *Classical and enriched Finite element formulations for Bloch-periodic boundary conditions*. International Journal of Numerical Methods in Engineering, 77(8):1121–1138, February 2009.

[20] Guarín-Zapata, Nicolás: *Simulación Numérica de Problemas de Propagación de Ondas: Dominios Infinitos y Semi-infinitos*. Master's thesis, Universidad EAFIT, 2012.

[21] Hladky-Hennion, Anne Christine and Jean Noël Decarpigny: *Analysis of the scattering of a plane acoustic wave by a doubly periodic structure using the finite element method: Application to Alberich anechoic coatings*. J. Acoust. Soc. Am, 90:3356–3367, 1991.

[22] Poole, David: *Álgebra lineal: una introducción moderna*. Cengage Learning Editores, 2nd edition, 2007. ISBN 9706865950.

[23] Aberg, M. and P. Gudmundson: *The usage of standard finite element codes for computation of dispersion relations in materials with periodic microstructure*. Journal of the Acoustical Society of America, 102:2007–2013, 1997.

[24] Taylor, RL: *FEAP–A Finite Element Analysis Program, Version 8.4 User Manual*, 2013.

[25] Hibbit, Karlsson & Sorensen: *ABAQUS User's Manual: Version 6.12*, 2014.

[26] Dhondt, Guido: *CalculiX CrunchiX User's Manual, version 2.7*, 2014.

[27] Geuzaine, Christophe and Jean François Remacle: *Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities*. International Journal for Numerical Methods in Engineering, 79(11):1309–1331, 2009. `http://geuz.org/gmsh/doc/preprints/gmsh_paper_preprint.pdf`.