Computational Seismology

Lecture 4: Finite-element Method

March 10, 2021

University of Toronto

TABLE OF CONTENTS

FD vs. FEM

FD vs. FEM

The key of **FD** method (pseudospectral methods are an extreme example of FD where derivatives involve all global points): discretize the continuous PDE into discrete form by replacing space/time derivatives with finite difference based on regular grids

FEM is a complete paradiam shift from FD:

- originated in solid mechanics and structural engineering (stiffness, mass matrix), based on solid mathematical foundation: weak form (variational form) of the PDE
- divide structures into **elements** linked at **element corners** and can be all joined into a complete system (**assembly**).
- suited for problems with **geometrically complex** structures (e.g., surface topography or internal structures); free-surface boundary condition is implicitly fulfilled (accurate for surface waves)
- more compute-intensive than other low-order methods such as finite differences

History of FEM

History of FEM in seismology

- First introduced for surface-wave propagation (Lysmer and Drake, 1972, Schlue, 1979), and seismic scattering problem (Day 1977)
- introduced to exploration seismology (Marfurt, 1984), later low-order implementation for more efficiency (Seron et al., 1990)
- seismic hazard analysis and engineering seismology: ground motion simulation (Bielak et al., 1998, Bielak and Xu 1999), FWI (Askan and Bielak, 2008, Epanomeritakis et al., 2008).
- hybrid methods of FEM and FD: Moczo et al. (2010b).
- Large linear system requiring global communications $\rightarrow$ high-order variations of FEM using Lagrange/Chebyshev polynomials as basis functions $\rightarrow$ **SEM**

FEM in a nutshell

Finite-element: basis functions

We again take the 1D elastic wave equation for the displacement field $u(x, t)$ as an example

$$\rho \partial_t^2 u = \partial_x(\mu \partial_x u) + f \tag{1}$$

Instead of solving u by discretization, we replace it by a **finite sum over basis function** $\phi_i(x)$, $i = 1, \cdots, N$

$$u(x, t) \approx \overline{u}(x, t) = \sum_{i=1}^{N} u_i(t)\phi_i(x) \tag{2}$$

For a particular time t, we solve the coefficients $u_i(t)$ corresponding to basis function $\phi_i(x)$. These local basis functions are often related to the discrete displacement value at node x_i, i.e., $u_i = u(x_i)$.

FEM: Weak form

Weak form of the wave equation can be formed by multiplying the original strong form with a **test function** $\phi_j(x)$ the same as the basis functions (**Galerkin principle**) and integrate over space.

$$\int_D \rho \partial_t^2 \overline{u}\, \phi_j \, dx + \int_D \mu \partial_x \overline{u}\, \partial_x \phi_j \, dx = \int_D f \phi_j \, dx + \text{b.c. (0 for free surface)}$$

To solve the approximate field $\overline{u}$ for the given model parameterization, it can be rewritten into a **linear system** for vector field **u** ($N \times 1$)

$$\mathbf{M}\, \ddot{\mathbf{u}}(t) = -\mathbf{K}^T \mathbf{u} + \mathbf{f} \tag{3}$$

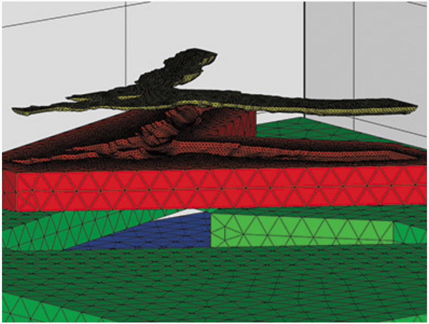
where $M_{N \times N}$ is the **mass matrix** and $K_{N \times N}$ is the **stiffness matrix**

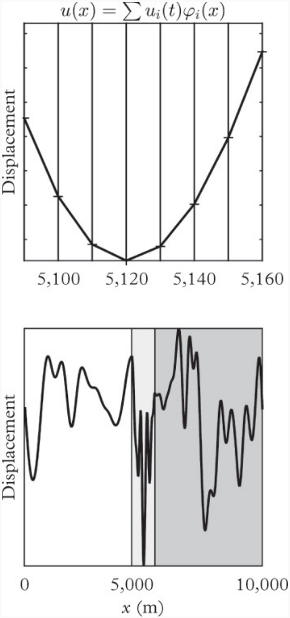
$$M_{ij} = \int_D \rho\, \phi_i \phi_j \, dx, \quad K_{ij} = \int_D \mu\, \dot{\phi}_i \dot{\phi}_j \, dx \tag{4}$$

which are computed (analytically) at elemental level and then assembled, even for arbitrary element shapes.

Challenge: solving a large linear system. **Remedy**: a specific choice of basis functions and a numerical integration scheme (SEM)

FEM examples





- (up) Tetrahedral FEM mesh for velocity structure of the Grenoble basin (sedimentary basin, bedrocks are meshed separately)
- (right) FEM simulation of 1D elastic waves in a domain with 3 velocities.

Static elasticity

FEM: Static elasticity

By assuming displacement does not depend on time, $\partial_t^2 u(x,t) = 0$, 1D elastic wave equation becomes the 1D static elasticity equation (Possion's equation) for $u(x)$

$$-\partial_x(\mu\partial_x u) = f, \tag{5}$$

equivalent to the displacement distribution along a string when pulled with forcing f. Multiplying the test vector $v(x)$, integrate over space (domain D on a 1D line) and apply integration by parts:

$$-\int_D v\,\partial_x(\mu\partial_x u)\,dx = \int_D \mu\partial_x u\partial_x v\,dx - [\mu v\partial_x u]_{xmin}^{xmax} = \int_D fv\,dx \tag{6}$$

The term in [] is related to B.C.. For free surface B.C., stress $\sigma = \mu\partial_x u$ vanishes at on the boundary, which means the term in [] vanishes. We get the free-surface boundary condition **FOR FREE** (also known as natural boundary condition).

FEM: static elasticity

After applying free-surface boundary condition, the weak form of 1D static elasticity equation becomes

$$\int_D \mu \partial_x u \partial_x v \, dx = \int_D f v \, dx \tag{7}$$

Now we discretize this continuous integral form by expanding the displacement field over the basis functions:

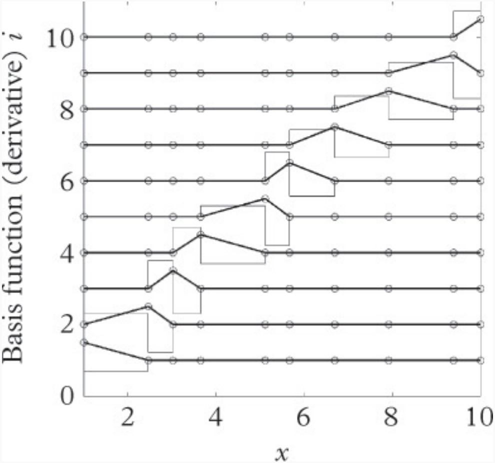
$$u(x,t) \approx \overline{u}(x,t) = \sum_{i=1}^{N} u_i(t)\phi_i(x) \tag{8}$$

and by the **Galerkin method**, we choose N number of test functions the same as the basis functions $v(x) \rightarrow \phi_j(x)$. A simple choice of $\phi_i(x)$ can be local: $\phi_i(x_j) = \delta_{ij}$, and linear inside an element

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } x_{i-1} \le x \le x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } x_i \le x \le x_{i+1} \\ 0 & \text{elsewhere} \end{cases} \tag{9}$$

FEM: Linear basis function

For example, discretize the domain into $N = 10$ points and 9 elements. An arbitrary function can be exactly interpolated at the element boundary points $x_j, j = 1, \cdots, N$



FEM

Note: adjacent elements share the same value at the boundaries (different from finite-volume and discontinous Galerkin methods)

Assemble the discrete version of the weak form (for test function $\phi_j(x)$

$$\int_D \mu \partial_x \left(\sum_{i=1}^{N} u_i \phi_i \right) \partial_x \phi_j \, dx = \int_D f \phi_j \, dx \tag{10}$$

or

$$\sum_{i=1}^{N} u_i \int_D \mu \partial_x \phi_i \partial_x \phi_j \, dx = \int_D f \phi_j \, dx \tag{11}$$

and form N system of equations in vector form

$$\mathbf{K}^{\mathbf{T}} \mathbf{u} = \mathbf{f} \tag{12}$$

FEM: linear system

N unknowns and N equations form the linear system

$$\mathbf{K}^{\mathbf{T}}\mathbf{u} = \mathbf{f} \tag{13}$$

where

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} \int_D f\phi_1\,dx \\ \int_D f\phi_2\,dx \\ \vdots \\ \int_D f\phi_N\,dx \end{pmatrix}, \quad K_{ij} = \int_D \mu\partial_x\phi_i\partial_x\phi_j\,dx \tag{14}$$

and **K** is the stiffness matrix. If **K** is positive definite, then the solution becomes

$$\mathbf{u} = (\mathbf{K}^{\mathbf{T}})^{-1}\mathbf{f} \tag{15}$$

Solving a large system of linear equations (with non-diagonal system matrix) can be numerically very expensive; to run it on parallel computers is by itself a field of research (LAPACK).

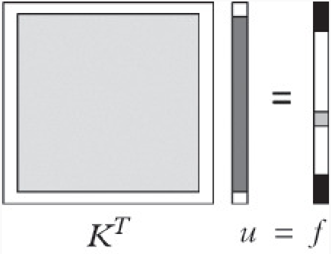
Boundary conditions

- Free surface boundary condition $\rightarrow$ nothing to do as it is implicitly fullfilled
- Fixed boundary condition (e.g., at $i = 1$ and $i = N$)

$$\bar{u} = u_1\phi_1 + \sum_{i=2}^{N-1} u_i\phi_i + u_N\phi_N \tag{16}$$

The weak form becomes

$$\sum_{i=2}^{N-1} u_i \int_D \mu\partial_x\phi_i\partial_x\phi_j\,dx = \int_D f\phi_j\,dx$$
$$- u_1 \int_D \mu\partial_x\phi_1\partial_x\phi_j\,dx - u_N \int_D \mu\partial_x\phi_N\partial_x\phi_j\,dx$$



where B.Cs are turned into source terms and the number of unknowns become $N - 2$ instead of N.

Reference element

All the integrals are carried out on the entire domain D with the global basis function: e.g., $\int_D \mu \partial_x \phi_i \partial_x \phi_j \, dx$, which can be performed at local element level and assembled through a standard procedure.

Introduce coordinate transformations from $x \to \xi$ at elemental level (which is different for different elements), e.g.,

$$x \in D_i \equiv [x_i, x_{i+1}] \to \xi = \frac{x - x_i}{h_i} \in [0, 1], \quad h_i = x_{i+1} - x_i$$

and the basis function becomes locally as

$$\phi_i(\xi) = \begin{cases} \xi & \text{for } x \in D_{i-1} \\ 1 - \xi & \text{for } x \in D_i \end{cases}$$

and derivatives

$$\partial_\xi \phi_i(\xi) = \begin{cases} 1 & \text{for } x \in D_{i-1} \\ -1 & \text{for } x \in D_i \end{cases}$$

Integral evaluations over elements: analytical

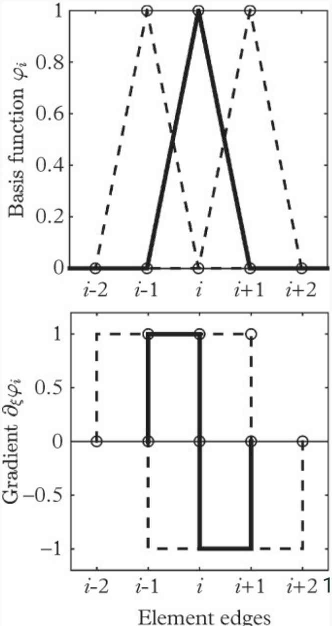
The stiffness matrix (assuming μ=const)

$$K_{ij} = \mu \int_D \partial_x \phi_i \partial_x \phi_j \, dx \tag{17}$$

For example:

$$\begin{aligned} K_{11} &= \mu \int_D \partial_x \phi_1 \partial_x \phi_1 \, dx = \frac{\mu}{h_1} \int_{D_1} \partial_\xi \phi_1 \partial_\xi \phi_1 \, d\xi \\ &= \frac{\mu}{h_1} \int_0^1 (-1)^2 d\xi = \frac{\mu}{h_1} \end{aligned} \tag{18}$$

$$\begin{aligned} K_{22} &= \mu \int_D \partial_x \phi_2 \partial_x \phi_2 \, dx = \frac{\mu}{h_1} \int_{D_1} \partial_\xi \phi_2 \partial_\xi \phi_2 \, d\xi \\ &+ \frac{\mu}{h_2} \int_{D_2} \partial_\xi \phi_2 \partial_\xi \phi_2 \, d\xi = \frac{\mu}{h_1} \int_0^1 1^2 d\xi \\ &+ \frac{\mu}{h_2} \int_0^1 (-1)^2 d\xi = \frac{\mu}{h_1} + \frac{\mu}{h_2} \end{aligned} \tag{19}$$



Integration and Assembly

And

$$K_{12} = \mu \int_D \partial_x \phi_1 \partial_x \phi_2 \, dx = \frac{\mu}{h_1} \int_{D_1} \partial_\xi \phi_1 \partial_\xi \phi_2 \, d\xi$$
$$= \frac{\mu}{h_1} \int_0^1 (-1) \cdot 1 d\xi = -\frac{\mu}{h_1} \tag{20}$$

If we assume uniform grid size h, then

$$K_{ij} = \frac{\mu}{h} \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \cdots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix} \tag{21}$$

very similar to FD coefficients for 2nd order spatial derivatives.

1D FEM Example

Example: $x \in [0, 1]$, unit forcing at $x = 0.75$, $h = 0.0526$, $N = 20$, fixed b.c. $u(0) = 0.15$, $u(x = 1) = 0.05$.

FEM: solve $\mathbf{u} = (\mathbf{K^T})^{-1}\mathbf{f}$

Recall how it is solved in FD: $-\mu\partial_x^2 u = f$

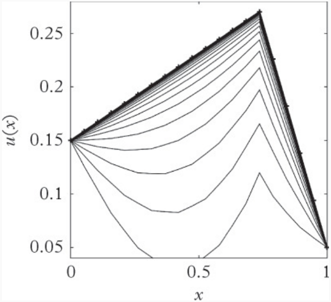
$$-\mu\frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = f \quad (22)$$

$$u(x) = \frac{u(x-h) + u(x+h)}{2} + \frac{h^2}{2\mu}f \quad (23)$$

Solved through iterative procedures $k \to k+1$ (**relaxation method**)

$$u_i^{k+1} = \frac{u_{i+1}^k + u_{i-1}^k}{2} + \frac{h^2}{2\mu}f_i \quad (24)$$

with an initial guess of $u_i^{k=1} = 0$.



Comparison of FEM (thick line) with converging FD (thin lines) solutions by relaxation method

Python codes: FEM vs FD

```
# [...]
# Basic parameters
nx = 20           # Number of boundary points
u = zeros(nx)     # Solution vector
f = zeros(nx)     # Source vector
mu = 1            # Constant shear modulus
# Element boundary points
x = linspace(0, 1, nx) # x in [0,1]
h = x[2] - x[1] # Constant element size
# Assemble stiffness matrix K_ij
K = zeros((nx, nx))
for i in range(1, nx-1):
    for j in range(1, nx-1):
        if i == j:
            K[i, j] = 2 * mu/h
        elif i == j + 1:
            K[i, j] = -mu/h
        elif i + 1 == j:
            K[i, j] = -mu/h
        else:
            K[i, j] = 0
# Souce term is a spike at i = 15
f[15] = 1
# Boundary condition at x = 0
u[0] = 0.15 ; f[1] = u[0]/h
# Boundary condition at x = 1
u[nx-1] = 0.05 ; f[nx-2] = u[nx-1]/h
# finite element solution
u[1:nx-1] = linalg.inv(K[1:nx-1, 1:nx-1]) @ f[1:nx-1].T
# [...]
```

```
# [...]
# Forcing
f[15] = 1/h  # force vector
for it in range(nt):
    # Calculate the average of u (omit boundaries)
    for i in range(1, nx-1):
        du[i] = u[i+1] + u[i-1]
    u = 0.5 * (f * h**2/mu + du)
    u[0] = 0.15     # Boundary condition at x=0
    u[nx-1] = 0.05  # Boundary condition at x=1
# [...]
```

u is only solved between 1:N-2 points; (Left) @ is used for conventional matrix multiplication (after Python 3.5).

1D elastic wave equation

1D elastic wave equation

For 1D elastic wave equation

$$\rho \partial_t^2 u = \partial_x(\mu \partial_x u) + f \tag{25}$$

where both ρ and μ are space-dependent. Its weak form using the basis functions as test functions becomes

$$\int_D \rho \partial_t^2 u \phi_j \, dx = -\int_D \mu \partial_x u \partial_x \phi_j \, dx + \int_D f \phi_j \, dx + [\mu \partial_x u \phi_j]_{xmin}^{xmax} \tag{26}$$

Given stress-free b.c., the term in [ ] vanishes. Inserting in the expansion of the displacement field by basis functions

$$u(x,t) \approx \overline{u}(x,t) = \sum_{i=1}^{N} u_i(t)\phi_i(x) \tag{27}$$

$$\sum_{i=1}^{N} \partial_t^2 u_i(t) \int_D \rho \phi_i \phi_j \, dx + \sum_{i=1}^{N} u_i(t) \int_D \mu \partial_x \phi_i \partial_x \phi_j \, dx = \int_D f \phi_j \, dx$$

FEM: Assembly into a linear system

In matrix-vector notation

$$\mathbf{M}^T \partial_t^2 \mathbf{u} + \mathbf{K}^T \mathbf{u} = \mathbf{f} \tag{28}$$

where

$$\begin{aligned} \text{Mass matrix:} \quad & \mathbf{M} \rightarrow M_{ij} = \int_D \rho \phi_i \phi_j \, dx \\ \text{Stiffness matrix:} \quad & \mathbf{K} \rightarrow K_{ij} = \int_D \mu \partial_x \phi_i \partial_x \phi_j \, dx \\ \text{Source vector:} \quad & f_j = \int_D f \phi_j \, dx \end{aligned} \tag{29}$$

and can be solved through time extrapolation

$$\mathbf{M}^T \frac{\mathbf{u}(t+dt) - 2\mathbf{u}(t) + \mathbf{u}(t-dt)}{dt^2} = \mathbf{f} - \mathbf{K}^T \mathbf{u} \tag{30}$$

Global system matrix

It requires the inversion of global mass matrix **M**, which involves global communications and is computationally costly unless **M** is diagonal. It is possible with the right choice of

- basis functions (Lagrange polynomials)
- a corresponding numerical integration scheme (Gauss integration)

which leads to the spectral-element methods (SEM).

Also **M** and **K** do not depend on time. The mass matrix **M** can be computed and inverted once for all before time iterations.

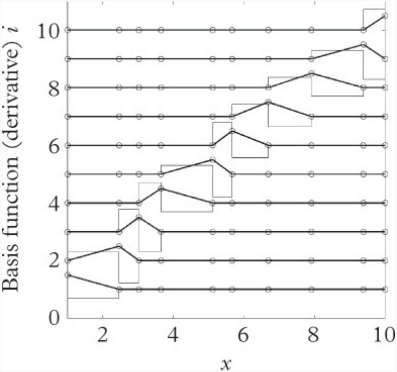
System matrices

Again introduce transformations $x \to \xi$ at elemental level, e.g.,

$$x \in D_i \equiv [x_i, x_{i+1}] \to \xi = \frac{x - x_i}{h_i} \in [0, 1], \quad h_i = x_{i+1} - x_i$$

and the basis function and its derivative become

$$\phi_i(\xi) = \begin{cases} \xi & \text{for } x \in D_{i-1} \\ 1 - \xi & \text{for } x \in D_i \end{cases} \qquad \partial_\xi \phi_i(\xi) = \begin{cases} 1 & \text{for } x \in D_{i-1} \\ -1 & \text{for } x \in D_i \end{cases}$$



Assemble the mass matrix

The mass matrix has only non-zero entries $M_{i,i-1}$, M_{ii}, $M_{i,i+1}$ (M_{11} and M_{NN} treated separately; density const within an element)

$$
\begin{aligned}
M_{ii} &= \int_D \rho\phi_i\phi_i \, dx = \int_{D_{i-1}+D_i} \rho\phi_i\phi_i \, d\xi \\
&= \rho_{i-1}h_{i-1}\int_0^1 \xi^2 d\xi + \rho_i h_i \int_0^1 (1-\xi)^2 d\xi = \frac{1}{3}(\rho_{i-1}h_{i-1} + \rho_i h_i) \\
M_{i,i-1} &= \int_{D_{i-1}} \rho\phi_i\phi_{i-1} \, d\xi = \rho_{i-1}h_{i-1}\int_0^1 \xi(1-\xi) d\xi = \frac{1}{6}\rho_{i-1}h_{i-1} \\
M_{i,i+1} &= \int_{D_i} \rho\phi_i\phi_{i+1} \, d\xi = \rho_i h_i \int_0^1 (1-\xi)\xi d\xi = \frac{1}{6}\rho_i h_i
\end{aligned}
\tag{31}
$$

With constant h and ρ

$$
M = \frac{\rho h}{6}\begin{pmatrix} \ddots & & & & 0 \\ 1 & 4 & 1 & & \\ & 1 & 1 & 1 & \\ & & 1 & 4 & 1 \\ 0 & & & & \ddots \end{pmatrix} \tag{32}
$$

Stiffness matrix

Similarly the stiffness matrix have values at $K_{i,i-1}$, K_{ii} and $K_{i,i+1}$

$$\begin{aligned} K_{ii} &= = \int_D \mu \partial_x \phi_i \partial_x \phi_i \, dx = \int_{D_{i-1}+D_i} \rho \partial_x \phi_i \partial_x \phi_i \, d\xi \\ &= \frac{\mu_{i-1}}{h_{i-1}} \int_0^1 (1)^2 d\xi + \frac{\mu_i}{h_i} \int_0^1 (-1)^2 d\xi = \frac{\mu_{i-1}}{h_{i-1}} + \frac{\mu_i}{h_i} \\ K_{i,i-1} &= \int_{D_{i-1}} \mu \partial_x \phi_i \partial_x \phi_{i-1} \, d\xi = \frac{\mu_{i-1}}{h_{i-1}} \int_0^1 1 \cdot (-1) d\xi = -\frac{\mu_{i-1}}{h_{i-1}} \\ K_{i,i+1} &= \int_{D_i} \mu \partial_x \phi_i \partial_x \phi_{i+1} \, d\xi = \frac{\mu_i}{h_i} \int_0^1 (-1) \cdot 1 d\xi = -\frac{\mu_i}{h_i} \end{aligned} \tag{33}$$

$$K = \frac{\mu}{h} \begin{pmatrix} \ddots & & & & 0 \\ -1 & 2 & 1 & & \\ & -1 & 2 & 1 & \\ & & -1 & 2 & 1 \\ 0 & & & & \ddots \end{pmatrix} \tag{34}$$

FEM 1D: python codes

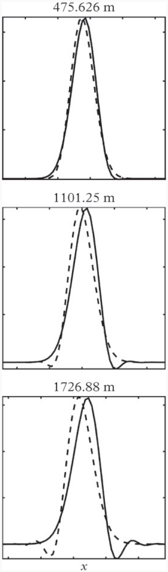
```
# Mass matrix M_ij
M = zeros((nx, nx))
for i in range(1, nx-1):
    for j in range(1, nx-1):
        if i == j:
            M[i, j] = (ro[i-1] * h[i-1]
                       + ro[i] * h[i]) / 3
        elif j == i + 1:
            M[i, j] = ro[i] * h[i]/6
        elif j == i - 1:
            M[i, j] = ro[i-1] * h[i-1]/6
        else:
            M[i, j] = 0
# Corner elements
M[0,0] = ro[0] * h[0] / 3
M[nx-1, nx-1] = ro[nx-1] * h[nx-2] / 3
```

```
# Time extrapolation
for it in range(nt):
    # Finite Element Method
    unew = (dt**2)*Minv @ (f*src[it] - K @ u)
           + 2*u - uold
    uold, u = u, unew
# [...]
```

1

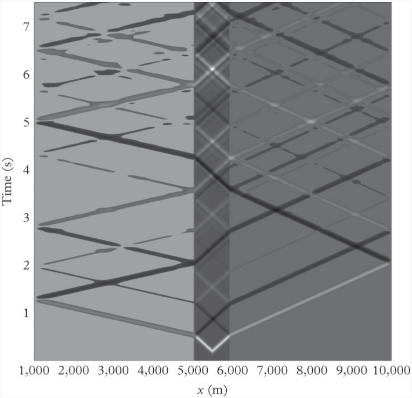
FEM Simulation example: Homogeneous

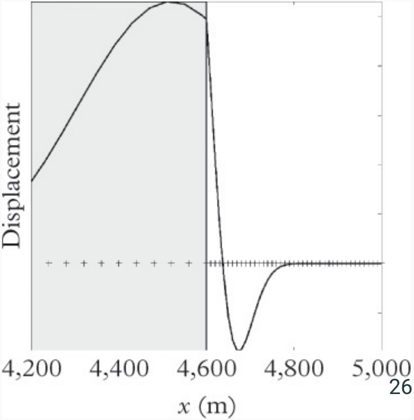
FEM: 1D Domain $[0, 10, 000]$ m, with $nx = 1000$, $h = 10$ m; $V_s = 3000\ m/s$ and $\rho = 2500\ kg/m^3$, $f_0 = 20$ Hz, $\epsilon = 0.5$ (related to CFL). (Right) Snapshots of FEM simulations (solid lines) with FD (dotted lines) as a function of propagation distance. Note the numerical dispersion. The most important advantages of the FEM: **element size can vary** (maintain similar NPW throughout the model).



FEM for Strong heterogeneities: h-adaptive mesh

A fault zone model with central LVZ of damaged zone has three subdomains (Vs=6000/1500/3000 m/s, dx=40/10/20, NPW ~ 30). Injection at the centre of LVZ, free-surface B.C., $f_0 = 5$ Hz. Note wavelength difference and non-differentiable wavefield at boundary.





Shape functions: from 1D to 2D and 3D

Shape functions: 1D linear

Recall the expansion of wavefield by basis functions

$$u(x) = \sum_{i=1}^{N} c_i \phi_i(x) \tag{35}$$

A standard procedure in FEM is to map all elements to a standard element to make integration easier. For example,

$$x \in D_i \equiv [x_i, x_{i+1}] \to \xi = \frac{x - x_i}{x_{i+1} - x_i} \in [0, 1]$$

We now derive the so-called **shape functions** used to describe the wavefield at element level in ξ. First let us look at a wavefield that is linear over the element

$$u(\xi) = c_1 + c_2 \xi \tag{36}$$

and satisfies the condition that $u(\xi = 0) = u_1$ and $u(\xi = 1) = u_2$.

linear shape function

Hence $c_1 = u_1$ and $c_2 = -u_1 + u_2$, or $\mathbf{u} = \mathbf{A}\mathbf{c}$, and

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{37}$$

or $\mathbf{c} = \mathbf{A}^{-1}\mathbf{u}$. And

$$u(\xi) = u_1 + (-u_1 + u_2)\xi = u_1(1-\xi) + u_2\xi = u_1 N_1(\xi) + u_2 N_2(\xi) \tag{38}$$

where the **shape functions** are defined as

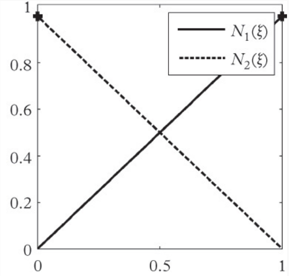
$$N_1(\xi) = 1 - \xi, \quad N_2(\xi) = \xi \tag{39}$$

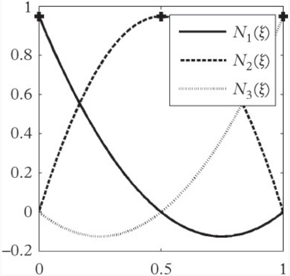
and in general, the shape functions of general order N satisfy

$$u(\xi) = \sum_{i=1}^{N} u_i N_i(\xi) \tag{40}$$

which is the **approximate continuous representation** of the solution field $u(\xi)$ inside the element.

1D shape functions





Extending the concept to higher order ($N > 2$), e.g.,

$$u(\xi) = c_1 + c_2\xi + c_3\xi^2$$

which satisfy the field exactly at three points $\xi = 0, \frac{1}{2}, 1$ as $u_{1,2,3}$ then $\mathbf{u} = \mathbf{Ac}$ and

$$u(\xi) = \sum_{i=1}^{3} u_i N_i(\xi)$$

$$N_1(\xi) = 1 - 3\xi + 2\xi^2,$$

$$N_2(\xi) = 4\xi - 4\xi^2,$$

$$N_3(\xi) = -\xi + 2\xi^2$$

Shape functions in 2D

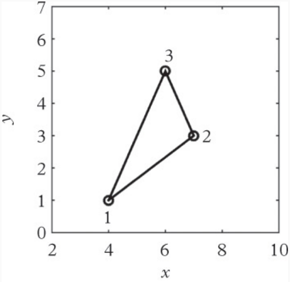
The most frequently used element shapes in 2D are triangles (e.g. after Delauney triangulation of arbitrary point clouds) and rectangles. We limit to only look at linear case. Transformation from (x, y) to (ξ, η)

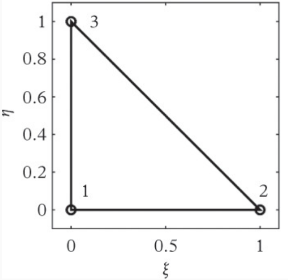
$$x = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta$$
$$y = y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta$$

Assuming $u(\xi, \eta) = c_1 + c_2\xi + c_3\eta$, where c_i's are solved by using $u(\xi = 0, 1, \eta = 0, 1)$

$$N_1(\xi, \eta) = 1 - \xi - \eta$$
$$N_2(\xi, \eta) = \xi,$$
$$N_3(\xi, \eta) = \eta$$

2D shape function



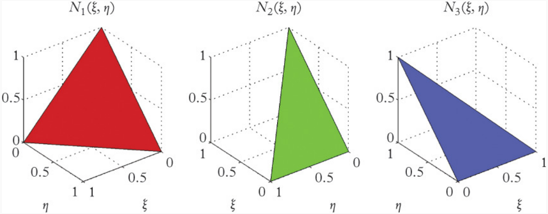


$$
\begin{aligned}
u(\xi,\eta) &= u(0,0)N_1(\xi,\eta) + u(1,0)N_2(\xi,\eta) \\
&\quad + u(0,1)N_3(\xi,\eta)
\end{aligned}
$$

$$N_1(\xi,\eta) = 1 - \xi - \eta$$

$$N_2(\xi,\eta) = \xi$$

$$N_3(\xi,\eta) = \eta$$

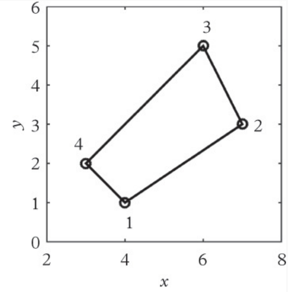


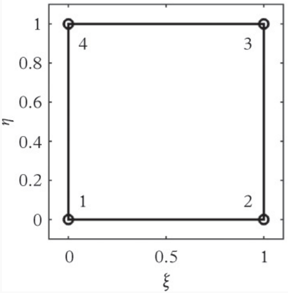
Rectangular shape functions

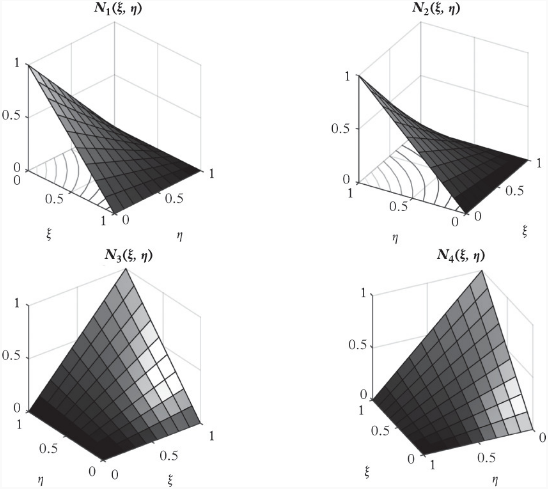
Shape functions for quadrilateral elements can be derived by a mapping to a standard square element

$$
\begin{aligned}
x &= x_1 + (x_2 - x_1)\xi + (x_4 - x_1)\eta + (x_3 - x_2)\xi\eta \\
y &= y_1 + (y_2 - y_1)\xi + (y_4 - y_1)\eta + (y_3 - y_2)\xi\eta \\
u(\xi, \eta) &= u(0,0)N_1(\xi, \eta) + u(1,0)N_2(\xi, \eta) \\
&\quad + u(1,1)N_3(\xi, \eta) + u(0,1)N_4(\xi, \eta) \\
N_1(\xi, \eta) &= (1 - \xi)(1 - \eta) \\
N_2(\xi, \eta) &= \xi(1 - \eta) \\
N_3(\xi, \eta) &= \xi\eta \\
N_4(\xi, \eta) &= (1 - \xi)\eta
\end{aligned}
$$

2D quadrilateral shape functions







3D Shape functions

- 1D to 2D/3D extension is substantially more involved than for 3D FD.
- References: Bao et al. (1996) and Bielak et al. (1998), Bielak et al. (2005) for adaptive mesh using Octree approach.
- Finite-element discontinuous Galerkin method, have recently been introduced to seismic wave propagation, in particular for dynamic rupture problems and wave propagation through media with highly complex geometrical features.