Computational Seismology: What is the best strategy for my problem? Part II

EnvSeis Short Course, May 27, 2024

Heiner Igel

May 23, 2024

Department of Earth and Environmental Sciences
Ludwig-Maximilians-University Munich

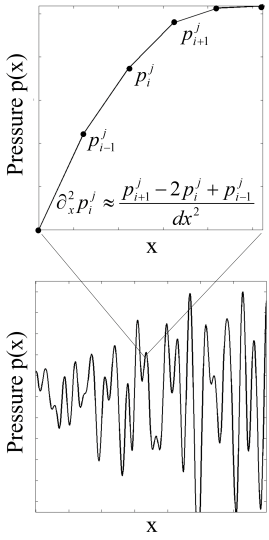
Numerical Methods for Wave Propagation Problems

What’s on the market

- **The finite-difference method**
- **The pseudospectal method**
- **The finite-element method**
- The spectral-element method
- The finite-volume method
- The discontinuous Galerkin method

The Finite-Difference Method

Finite differences in a Nutshell



- Direct numerical approximation of partial derivatives using **finite-differences**
- Local computational scheme $\rightarrow$ **efficient parallelisation**
- Very efficient on **regular grids**, cumbersome for strongly heterogeneous models
- **Boundary conditions** difficult to implement with high-order accuracy
- The method of choice for models with **flat topo and moderate velocity perturbations**
- Highly efficient extensions possible, but rarely used!

Finite Differences

Forward derivative

$$d_x f(x) = \lim_{dx \to 0} \frac{f(x + dx) - f(x)}{dx}$$

Centered derivative

$$d_x f(x) = \lim_{dx \to 0} \frac{f(x + dx) - f(x - dx)}{2dx}$$

Backward derivative

$$d_x f(x) = \lim_{dx \to 0} \frac{f(x) - f(x - dx)}{dx}$$

Finite Differences

Forward derivative

$$d_x f^+ \approx \frac{f(x+dx) - f(x)}{dx}$$

Centered derivative

$$d_x f^c \approx \frac{f(x+dx) - f(x-dx)}{2dx}$$

Backward derivative

$$d_x f^- \approx \frac{f(x) - f(x-dx)}{dx}$$

Finite Differences and Taylor Series

The approximate sign is important here as the derivatives at point x are not exact. Understanding the accuracy by looking at the definition of Taylor Series:

$$f(x + dx) = f(x) + f'(x)\ dx + \frac{1}{2!}\ f''(x)\ dx^2 + O(dx^3)$$

Subtraction with f(x) and division by dx leads to the definition of the forward derivative:

$$\frac{f(x+dx)-f(x)}{dx} = f'(x)\ + \frac{1}{2!}\ f''(x)\ dx + O(dx^2)$$

Finite Differences and Taylor Series

Using the same approach - adding the Taylor Series for $f(x + dx)$ and $f(x - dx)$ and dividing by $2dx$ leads to:

$$\frac{f(x+dx)-f(x-dx)}{2dx} = f'(x) + O(dx^2)$$

This implies a centered finite-difference scheme more rapidly converges to the correct derivative on a regular grid

$\Longrightarrow$ It matters which of the approximate formula one chooses

$\Longrightarrow$ It does not imply that one or the other finite-difference approximation is always the better one

Higher Derivatives

The partial differential equations have often 2nd (seldom higher) derivatives

Developing from first derivatives by mixing a forward and a backward definition yields

$$\partial_x^2 f \approx \frac{\frac{f(x+dx)-f(x)}{dx} - \frac{f(x)-f(x-dx)}{dx}}{dx} = \frac{f(x+dx) - 2f(x) + f(x-dx)}{dx^2}$$

Acoustic waves in 1D

To solve the wave equation, we start with the simplest wave equation:

The constant density acoustic wave equation in 1D

$$\ddot{p} = c^2 \, \partial_x^2 p + s$$

impossing pressure-free conditions at the two boundaries as

$$p(x) \mid_{x=0,L} = 0$$

Acoustic waves in 1D

The following dependencies apply:

$p \rightarrow p(\mathbf{x}, \mathbf{t})$	pressure
$c \rightarrow c(\mathbf{x})$	P-velocity
$s \rightarrow s(\mathbf{x}, \mathbf{t})$	source term

As a first step we need to discretize space and time and we do that with a constant increment that we denote dx and dt.

$$x_j = j\mathrm{d}x, \qquad j = 0,\ j_{max}$$
$$t_n = n\mathrm{d}t, \qquad n = 0,\ n_{max}$$

Acoustic waves in 1D

Starting from the continuous description of the partial differential equation to a discrete description. The upper index will correspond to the time discretization, the lower index will correspond to the spatial discretization

$$
\begin{aligned}
p_j^{n+1} &\rightarrow p(x_j, t_n + \mathrm{d}t) \\
p_j^{n} &\rightarrow p(x_j, t_n) \\
p_j^{n-1} &\rightarrow p(x_j, t_n - \mathrm{d}t) \\
p_{j+1}^{n} &\rightarrow p(x_j + \mathrm{d}x, t_n) \\
p_j^{n} &\rightarrow p(x_j, t_n) \\
p_{j-1}^{n} &\rightarrow p(x_j - \mathrm{d}x, t_n)
\end{aligned}
$$

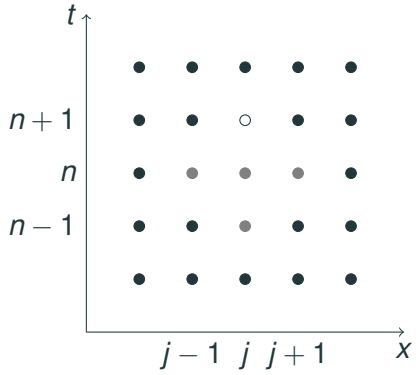
.

Acoustic waves in 1D

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{\mathrm{d}t^2} = c_j^2 \left[\frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{\mathrm{d}x^2} \right] + s_j^n \, .$$

the r.h.s. is defined at same time level n

the l.h.s. requires information from three different time levels



Acoustic waves in 1D

Assuming that information at time level n (the presence) and $n-1$ (the past) is known, we can solve for the unknown field p_j^{n+1}:

$$p_j^{n+1} = c_j^2 \frac{\mathrm{d}t^2}{\mathrm{d}x^2} \left[p_{j+1}^n - 2p_j^n + p_{j-1}^n \right] + 2p_j^n - p_j^{n-1} + \mathrm{d}t^2 s_j^n$$

The initial condition of our wave simulation problem is such that everything is at rest at time t = 0:

$$p(x,t)|_{t=0} = 0\ ,\ \dot{p}(x,t)|_{t=0} = 0.$$

Exercise (pencil and paper)

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{\mathrm{d}t^2} = c_j^2 \left[\frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{\mathrm{d}x^2} \right] + s_j^n \, .$$

... is the FD algorithm for the wave equation ...

$$\partial_t^2 p(x,t) = c(x)^2 \, \partial_x^2 p(x,t) + s(x,t)$$

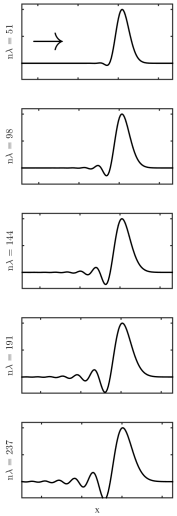
Can you come up with an equivalent FD algorithm for the **advection equation**?

$$\partial_t u(x,t) = c(x) \partial_x u(x,t)$$

Acoustic wave equation: numerical solutions

```
# Time extrapolation
for it in range(nt):
    # calculate partial derivatives (omit boundaries)
    for i in range(1, nx - 1):
        d2p[i] = (p[i + 1] - 2 * p[i] + p[i - 1]) / dx ** 2
    # Time extrapolation
    pnew = 2 * p - pold + dt ** 2 * c ** 2 * d2p
    # Add source term at isrc
    pnew[isrc] = pnew[isrc] +  dt ** 2 * src[it] / dx
    # Remap time levels
    pold, p = p, pnew
```

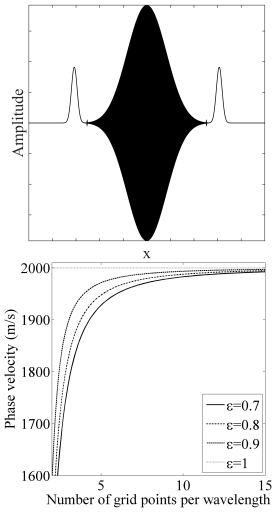
Result



Choosing a grid increment of $dx = 0.5m \longrightarrow$ about 24 points per spatial wavelength for the dominant frequency

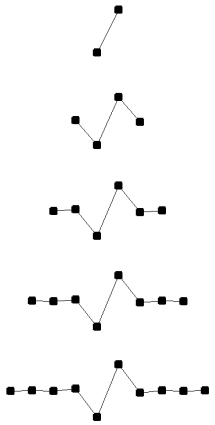
Setting time increment $dt = 0.0012 \longrightarrow$ around 40 points per dominant period

von Neumann Analysis, Stability, Dispersion



- Plane waves in a discrete world
- $p(x,t) = e^{k\mathbf{j\,dx} - \omega\mathbf{n\,dt}}$
- Simulations are conditionally stable
- $c\,\frac{dt}{dx} \leq \epsilon \approx 1$ *CFL - criterion*
- Simulated phase velocity becomes numerically dispersive!
- The more points per wavelength the more accurate
- How to check?

High-Order Operators (first derivative)



Graphical illustration of the Taylor Operators for the first derivative for higher orders The weights rapidly become small for increasing distance to central point of evaluation

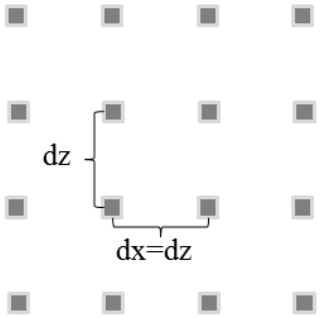
Check out the Jupyter Notebook *"'Taylor Operators"'*

Acoustic Wave Propagation in 2D

Acoustic Wave Propagation in 2D

In 2D the constant-density acoustic wave equation is given as

$$\ddot{p}(x,z,t) \;=\; c(x,z)^2(\partial_x^2 p(x,z,t) + \partial_z^2 p(x,z,t)) \;+ s(x,z,t)$$



Acoustic Wave Propagation in 2D

Discretizing space-time using

$$p(x,z,t) \;\rightarrow\; p^n_{j,k} \;=\; p(ndt, jdx, kdz)\;.$$

and the 3-point operator for the 2nd derivatives in time leads to the extrapolation scheme

$$\frac{p^{n+1}_{j,k} - 2p^n_{j,k} + p^{n-1}_{j,k}}{\mathrm{d}t^2} \;=\; c_j^2(\partial_x^2 p + \partial_z^2 p) \;+ s^n_{j,k}$$

where

$$\partial_x^2 p \;=\; \frac{p^n_{j+1,k} - 2p^n_{j,k} + p^n_{j-1,k}}{\mathrm{d}x^2}$$

$$\partial_z^2 p \;=\; \frac{p^n_{j,k+1} - 2p^n_{j,k} + p^n_{j,k-1}}{\mathrm{d}z^2}$$

Fault zone waves - Landers, CA

Choosing the Right Simulation Parameters

The simulation of **fault-zone trapped waves**

Note: Scalar acoustic wave equation is mathematically identical to the SH-wave propagation problem

Investigate the effects of a narrow 200m wide fault zone with a 25% velocity decrease

	Description	Value
f_0, f_{max}	Dominant, maximum frequency	10 Hz, 30 Hz
c_{min}, c_{max}	Min., max. velocity	2250 m/s, 3000 m/s
x_{max}, z_{max}	Min., max. extension	10 km, 10 km
t_{max}	Seismogram length	3.5 s

Choosing the Right Simulation Parameters

- What is the **smallest wavelength** propagating in the medium?

  Answer: With the $f_{max} = 30Hz$ and the smallest velocity $c_{min} = 2250m/s$ the shortest wavelength is given by $\lambda_{min} = c_{min}/f_{max} = 75m$.

- How many **numbers per smallest wavelength** is required by the numerical method (given the wave propagation distance that needs to be covered)?

  Answer: The dominant wavelength in the low-velocity medium is $\lambda_{dom} = c_{min}/f_{dom} = 225m$. Thus we expect to propagate more than 20 wavelengths to the surface. As we use a 5-point operator we choose 20 points per dominant wavelength, resulting in about 7.5 points per smallest wavelength.

Choosing the Right Simulation Parameters

- What is the **smallest wavelength** propagating in the medium?

  Answer: With the $f_{max} = 30Hz$ and the smallest velocity $c_{min} = 2250m/s$ the shortest wavelength is given by $\lambda_{min} = c_{min}/f_{max} = 75m$.

- How many **numbers per smallest wavelength** is required by the numerical method (given the wave propagation distance that needs to be covered)?

  Answer: The dominant wavelength in the low-velocity medium is $\lambda_{dom} = c_{min}/f_{dom} = 225m$. Thus we expect to propagate more than 20 wavelengths to the surface. As we use a 5-point operator we choose 20 points per dominant wavelength, resulting in about 7.5 points per smallest wavelength.

Choosing the Right Simulation Parameters

- What is the **smallest wavelength** propagating in the medium?

  Answer: With the $f_{max} = 30Hz$ and the smallest velocity $c_{min} = 2250m/s$ the shortest wavelength is given by $\lambda_{min} = c_{min}/f_{max} = 75m$.

- How many **numbers per smallest wavelength** is required by the numerical method (given the wave propagation distance that needs to be covered)?

  Answer: The dominant wavelength in the low-velocity medium is $\lambda_{dom} = c_{min}/f_{dom} = 225m$. Thus we expect to propagate more than 20 wavelengths to the surface. As we use a 5-point operator we choose 20 points per dominant wavelength, resulting in about 7.5 points per smallest wavelength.

Choosing the Right Simulation Parameters

- What is the **smallest wavelength** propagating in the medium?

  Answer: With the $f_{max} = 30Hz$ and the smallest velocity $c_{min} = 2250m/s$ the shortest wavelength is given by $\lambda_{min} = c_{min}/f_{max} = 75m$.

- How many **numbers per smallest wavelength** is required by the numerical method (given the wave propagation distance that needs to be covered)?

  Answer: The dominant wavelength in the low-velocity medium is $\lambda_{dom} = c_{min}/f_{dom} = 225m$. Thus we expect to propagate more than 20 wavelengths to the surface. As we use a 5-point operator we choose 20 points per dominant wavelength, resulting in about 7.5 points per smallest wavelength.

Choosing the Right Simulation Parameters

- What is the (in our case constant) **grid spacing** that needs to be implemented?

  Answer: With 20 points per dominante wavelength we obtain $dx = \lambda_{dom}/20 = 11.25m$ grid spacing.

- What is the **size of the physical domain** and how many overal **grid points** are required?

  Answer: The spatial extent of the 2D model is $10km \times 10km$. In each dimension we thus need $10000m/dx \approx 900$ grid points leading to 900^2 overall grid points.

Choosing the Right Simulation Parameters

- What is the (in our case constant) **grid spacing** that needs to be implemented?

  Answer: With 20 points per dominante wavelength we obtain $dx = \lambda_{dom}/20 = 11.25m$ grid spacing.

- What is the **size of the physical domain** and how many overal **grid points** are required?

  Answer: The spatial extent of the 2D model is $10km \times 10km$. In each dimension we thus need $10000m/dx \approx 900$ grid points leading to 900^2 overall grid points.

Choosing the Right Simulation Parameters

- What is the (in our case constant) **grid spacing** that needs to be implemented?

  Answer: With 20 points per dominante wavelength we obtain $dx = \lambda_{dom}/20 = 11.25m$ grid spacing.

- What is the **size of the physical domain** and how many overal **grid points** are required?

  Answer: The spatial extent of the 2D model is $10km \times 10km$. In each dimension we thus need $10000m/dx \approx 900$ grid points leading to 900^2 overall grid points.

Choosing the Right Simulation Parameters

- What is the (in our case constant) **grid spacing** that needs to be implemented?

  Answer: With 20 points per dominante wavelength we obtain $dx = \lambda_{dom}/20 = 11.25m$ grid spacing.

- What is the **size of the physical domain** and how many overal **grid points** are required?

  Answer: The spatial extent of the 2D model is $10km \times 10km$. In each dimension we thus need $10000m/dx \approx 900$ grid points leading to 900^2 overall grid points.

Choosing the Right Simulation Parameters

- Will the seismogram(s) be influenced by artificial boundary reflections (i.e., is it necessary to implement **absorbing boundaries** or increase the model size)?

  Answer: With the model setup as chosen - the fault zone in the middle of the model - we will not expect problems with reflections from the boundaries. Thus no need to implement absorbing boundaries.

- What is the maximum velocity in the model, and the resulting **time step** (given the grid increment and the **CFL criterion**)?

  Answer: The maximum velocity in the model is $c_{max} = 3000 m/s$. Assuming a CFL value of $\epsilon = 0.7$ we can determine the time step required for a stable simulation as $dt = \epsilon dx / c_{max} = 0.0026 s$.

Choosing the Right Simulation Parameters

- Will the seismogram(s) be influenced by artificial boundary reflections (i.e., is it necessary to implement **absorbing boundaries** or increase the model size)?

  Answer: With the model setup as chosen - the fault zone in the middle of the model - we will not expect problems with reflections from the boundaries. Thus no need to implement absorbing boundaries.

- What is the maximum velocity in the model, and the resulting **time step** (given the grid increment and the **CFL criterion**)?

  Answer: The maximum velocity in the model is $c_{max} = 3000 m/s$. Assuming a CFL value of $\epsilon = 0.7$ we can determine the time step required for a stable simulation as $dt = \epsilon dx / c_{max} = 0.0026 s$.

Choosing the Right Simulation Parameters

- Will the seismogram(s) be influenced by artificial boundary reflections (i.e., is it necessary to implement **absorbing boundaries** or increase the model size)?

  Answer: With the model setup as chosen - the fault zone in the middle of the model - we will not expect problems with reflections from the boundaries. Thus no need to implement absorbing boundaries.

- What is the maximum velocity in the model, and the resulting **time step** (given the grid increment and the **CFL criterion**)?

  Answer: The maximum velocity in the model is $c_{max} = 3000 m/s$. Assuming a CFL value of $\epsilon = 0.7$ we can determine the time step required for a stable simulation as $dt = \epsilon dx / c_{max} = 0.0026 s$.

Choosing the Right Simulation Parameters

- Will the seismogram(s) be influenced by artificial boundary reflections (i.e., is it necessary to implement **absorbing boundaries** or increase the model size)?

  Answer: With the model setup as chosen - the fault zone in the middle of the model - we will not expect problems with reflections from the boundaries. Thus no need to implement absorbing boundaries.

- What is the maximum velocity in the model, and the resulting **time step** (given the grid increment and the **CFL criterion**)?

  Answer: The maximum velocity in the model is $c_{max} = 3000 m/s$. Assuming a CFL value of $\epsilon = 0.7$ we can determine the time step required for a stable simulation as $dt = \epsilon dx / c_{max} = 0.0026 s$.

Choosing the Right Simulation Parameters

- What is the overall **number of time steps** to be propagated?

  Answer: For a desired simulation time of $t_{max} = 3.5s$ the number of time steps required is $t_{max}/dt \approx 1300$.

- How much **core memory** (RAM) will the simulation approximately require?

  Answer: For a simple estimate we focus on the space-dependent fields that will constitute the largest part of the memory allocation. Those fields are 1) the velocity model, and 2) the pressure field at three different time levels, and 3) two temporary fields containing the 2nd space derivatives. Assuming double precision floating point numbers (8 bytes per number) this will require approximately $6 \times 900^2 bytes \approx 40 MBytes$.

Choosing the Right Simulation Parameters

- What is the overall **number of time steps** to be propagated?

  Answer: For a desired simulation time of $t_{max} = 3.5s$ the number of time steps required is $t_{max}/dt \approx 1300$.

- How much **core memory** (RAM) will the simulation approximately require?

  Answer: For a simple estimate we focus on the space-dependent fields that will constitute the largest part of the memory allocation. Those fields are 1) the velocity model, and 2) the pressure field at three different time levels, and 3) two temporary fields containing the 2nd space derivatives. Assuming double precision floating point numbers (8 bytes per number) this will require approximately $6 \times 900^2 bytes \approx 40 MBytes$.

Choosing the Right Simulation Parameters

- What is the overall **number of time steps** to be propagated?

  Answer: For a desired simulation time of $t_{max} = 3.5s$ the number of time steps required is $t_{max}/dt \approx 1300$.

- How much **core memory** (RAM) will the simulation approximately require?

  Answer: For a simple estimate we focus on the space-dependent fields that will constitute the largest part of the memory allocation. Those fields are 1) the velocity model, and 2) the pressure field at three different time levels, and 3) two temporary fields containing the 2nd space derivatives. Assuming double precision floating point numbers (8 bytes per number) this will require approximately $6 \times 900^2 bytes \approx 40 MBytes$.

Choosing the Right Simulation Parameters

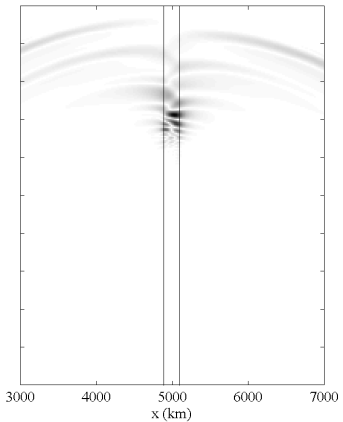
- What is the overall **number of time steps** to be propagated?

  Answer: For a desired simulation time of $t_{max} = 3.5s$ the number of time steps required is $t_{max}/dt \approx 1300$.

- How much **core memory** (RAM) will the simulation approximately require?

  Answer: For a simple estimate we focus on the space-dependent fields that will constitute the largest part of the memory allocation. Those fields are 1) the velocity model, and 2) the pressure field at three different time levels, and 3) two temporary fields containing the 2nd space derivatives. Assuming double precision floating point numbers (8 bytes per number) this will require approximately $6 \times 900^2 bytes \approx 40 MBytes$.

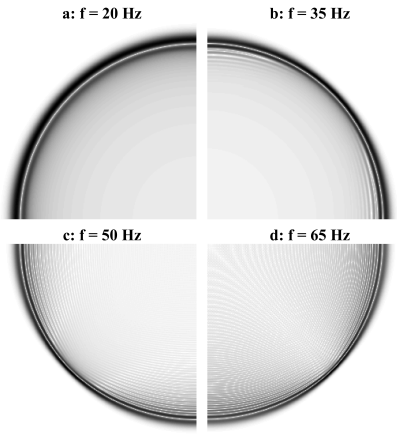
Choosing the Right Simulation Parameters

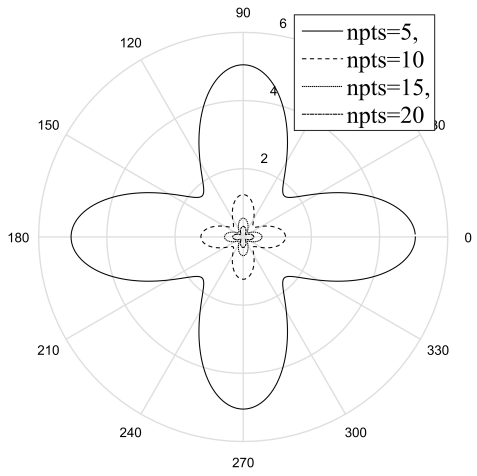


Snapshot of pressure amplitude for a source located at the left edge of a fault zone model

It indicates the occurrence of head waves and the development of a dispersive wavefield trapped inside the low-velocity zone

Numerical Anisotropy





Exercise - Jupyter Notebooks

Play around with the following Jupyter notebooks (Finite difference folder, github, seismo-live, or COURSERA)

- **ac1d** (change dt and central frequency)
- **ac2d_heterogeneou**s (play with the various models, invent your own model)
- **advection** (note the wrong diffusive behavior for a first-order scheme)
- **Taylor operators** (note the quickly decreasing amplitude away from the center, consequence?)

Note down **questions** during the process!

Elastic Wave Propagation in 1D

Velocity - Stress Formulation

Defining velocity v and stress component σ as

$$\begin{aligned} \partial_t u &= v \\ \sigma &= \mu \partial_x u \end{aligned}$$

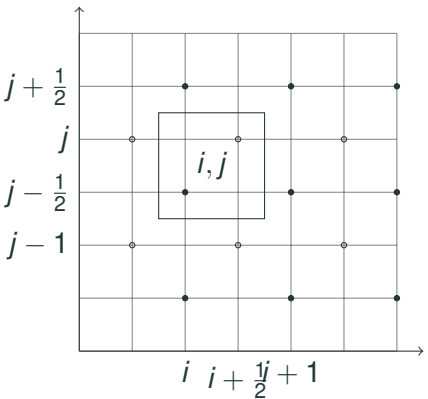
and assuming space-time dependencies leads to the wave equation

$$\begin{aligned} \rho \partial_t v &= \partial_x \sigma + f \\ \dot{\sigma} &= \mu \partial_x v \end{aligned}$$

Our unknowns are

$$\mathbf{q}(x, t) = (v, \sigma)$$

Velocity - Stress: Staggered grid formulation



$\bullet\ v, \rho$
$\circ\ \mu, \sigma$

$$\frac{v_i^{j+\frac{1}{2}} - v_i^{j-\frac{1}{2}}}{\mathrm{d}t} = \frac{1}{\rho_i} \frac{\sigma^j_{i+\frac{1}{2}} - \sigma^j_{i-\frac{1}{2}}}{\mathrm{d}x} + \frac{f_i^j}{\rho_i}$$

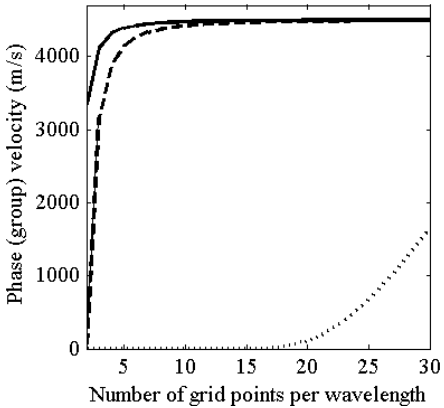
$$\frac{\sigma^{j+1}_{i+\frac{1}{2}} - \sigma^j_{i+\frac{1}{2}}}{\mathrm{d}t} = \mu_{i+\frac{1}{2}} \frac{v_{i+1}^{j+\frac{1}{2}} - v_i^{j+\frac{1}{2}}}{\mathrm{d}x}$$

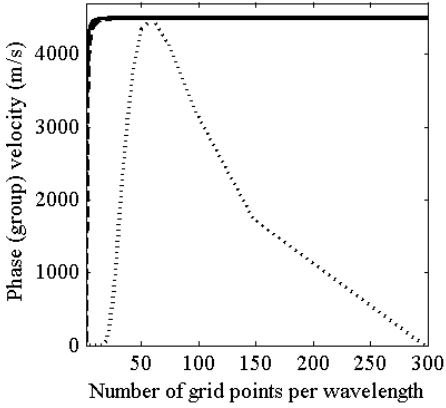
leading to the extrapolation scheme

$$v_i^{j+\frac{1}{2}} = \frac{dt}{\rho_i} \frac{\sigma^j_{i+\frac{1}{2}} - \sigma^j_{i-\frac{1}{2}}}{\mathrm{d}x} + v_i^{j-\frac{1}{2}} + \frac{dt}{\rho_i} f_i^j$$

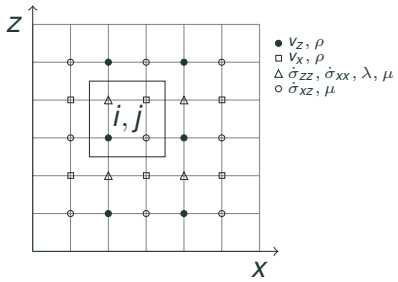
$$\sigma^{j+1}_{i+\frac{1}{2}} = \mathrm{d}t\, \mu_{i+\frac{1}{2}} \frac{v_{i+1}^{j+\frac{1}{2}} - v_i^{j+\frac{1}{2}}}{\mathrm{d}x} + \sigma^j_{i+\frac{1}{2}}$$

Velocity - Stress: Dispersion





Elastic 2D - Staggered Grids



- Contains fundamental aspects of grid staggering for the stress-strain relation for higher dimensions
- The extension to 3D is straight forward
- Scheme in seismology most widely used

Elastic 2D - Staggered Grids

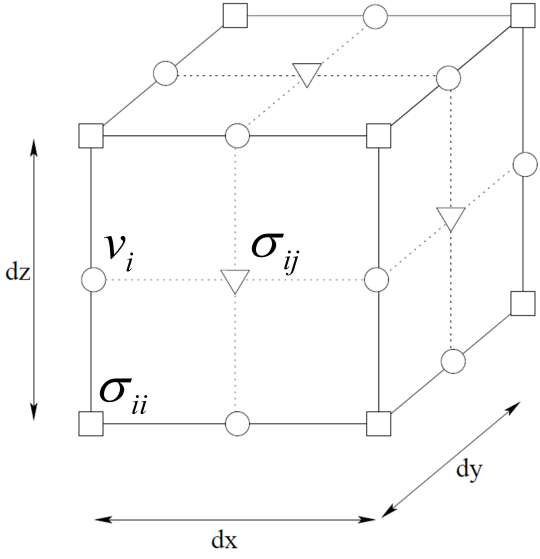
The stress-strain relation in 1D

$$\dot{\sigma}_{ij} = \lambda \dot{\epsilon}_{kk} \delta_{ij} + 2\mu \dot{\epsilon}_{ij}$$

and rewriting it in 2D using the definition of the strain tensor to obtain for each component

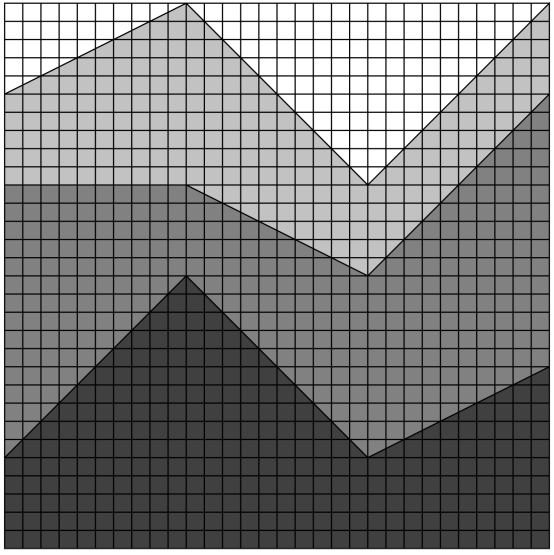
$$\begin{aligned}
\dot{\sigma}_{xx} &= (\lambda + 2\mu)\partial_x v_x + \lambda \partial_z v_z \\
\dot{\sigma}_{zz} &= (\lambda + 2\mu)\partial_z v_z + \lambda \partial_x v_x \\
\dot{\sigma}_{xz} &= \mu(\partial_x v_z + \lambda \partial_z v_x)
\end{aligned}$$

Elastic 3D - Staggered Grids



- Most common FD scheme today
- Usually 4-point operators
- Time extrapolation with Runge-Kutta

Layered models



- Layers might not coincide with grid lines
- Blocky representation of interfaces (errors)
- Grid stretching, curvi-linear grids, homogenization

Optimal Operators (Geller et al.)

Conventional ($1/dt^2$)

t+dt		1	
t		-2	
t-dt		1	
	x-dx	x	x+dx

Optimal ($1/dt^2$)

t+dt	1/12	10/12	1/12
t	-2/12	-20/12	-2/12
t-dt	1/12	10/12	1/12
	x-dx	x	x+dx

Conventional ($1/dx^2$)

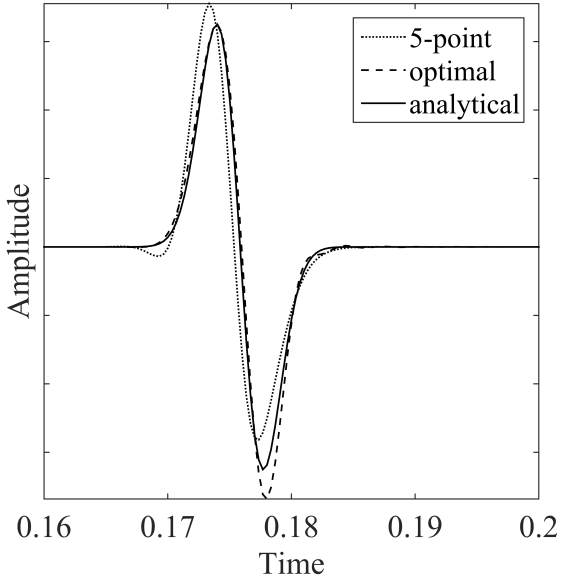
t+dt			
t	1	-2	1
t-dt			
	x-dx	x	x+dx

Optimal ($1/dx^2$)

t+dt	1/12	-2/12	1/12
t	10/12	-20/12	10/12
t-dt	1/12	-2/12	1/12
	x-dx	x	x+dx

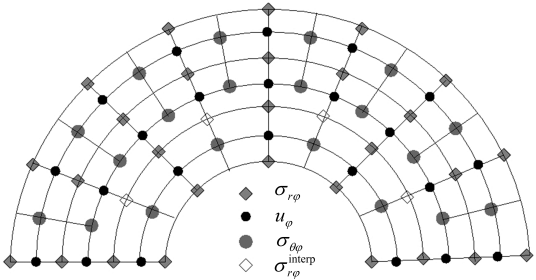
- FD operators spread in time and space
- Implicit scheme turned explicit using Born approximation

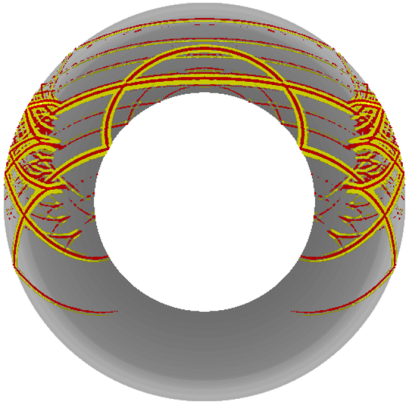
Optimal Operators (Geller et al.)



- Substantial accuracy improvement
- Negligible extra costs

Spherical coordinates

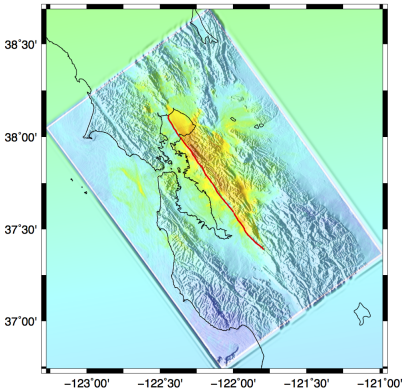




Summary

- Replacing the **partial derivatives** by finite differences allows partial differential equations such as the wave equation to be **solved directly** for (in principle) **arbitrarily heterogeneous media**.
- The resulting space-time discretization leads to unphysical phenomena such as **numerical dispersion** that can only be avoided by sampling with enough **number of grid points per wavelength**.
- The accuracy of finite-difference operators can be improved by using information from more grid points (i.e., **longer operators**). The weights for the grid points can be obtained using **Taylor series**.
- Classic plane-wave analysis of the approximative scheme leads to the famous **Courant-Friedrich-Levy (CFL)** criterion that restricts the choice of the space-time discretization.
- In higher dimensions the error of the wave propagation becomes **anisotropic**. In regular-spaced grids the most accurate direction is at 45° w.r.t. the grid axes.
- The implementation of **boundary conditions** in the case of finite-differences needs special care.

Applications, community codes

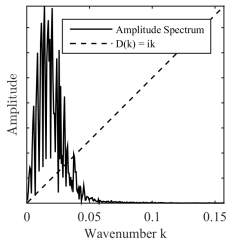


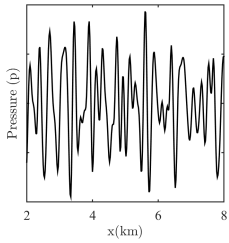
Source: geodynamics.org (SW4)

- Method of choice for **flat surfaces and body wave problems** (exploration)
- Very accurate **(optimal) operators** possible, but ...
- Summation-by-parts approach (better for **topography**)
- Combination with **homogenisation** (regular grid revival)
- Community codes: SW4 (CIG), SOFI3D (Karlsruhe)

The Pseudospectral Method: the road to spectral elements

The Pseudospectral Method in a Nutshell





- Calculation of **exact derivatives** in spectral domain
- **Less dispersive** than the finite-difference method (isotropic errors)
- **Boundary conditions hard** to implement (except with Chebyshev)
- Global communication scheme $\rightarrow$ **inefficient parallelisation**
- Combinations with FD possible
- **Hardly in use** today, but concepts used in the spectral-element method

Fourier Series and Transforms

Forward Transform

$$F(k) = \mathscr{F}[f(x)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx$$

Inverse Transform

$$f(x) = \mathscr{F}^{-1}[F(k)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(k) e^{ikx} dk$$

Taking the formulation of the inverse transform to obtain the derivative of function f(x)

$$\begin{aligned}\frac{d}{dx}f(x) &= \frac{d}{dx}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}F(k)e^{ikx}dk \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}ik\ F(k)e^{ikx}dk \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}D(k)\ F(k)e^{ikx}dk\end{aligned}$$

with $D(k) = ik$

We can extend this formulation to the calculation of the $n-th$ derivative of $f(x)$ to achieve

$$F^{(n)}(k) = D(k)^n\ F(k) = (ik)^n\ F(k)$$

Thus using the condensed Fourier transform operator $\mathscr{F}$ we can obtain an exact $n-th$ derivative using

$$\begin{aligned} f^{(n)}(x) &= \mathscr{F}^{-1}[(ik)^n F(k)] \\ &= \mathscr{F}^{-1}[(ik)^n \mathscr{F}[f(x)]] \, . \end{aligned}$$

Constant-density acoustic wave equation in 1D

$$\ddot{p} = c^2 \partial_x^2 p + s$$

The time-dependent part is solved using a standard 3-point finite-difference operator leading to

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{\mathrm{d}t^2} = c_j^2 \partial_x^2 p_j^n + s_j^n$$

where upper indices represent time and lower indices space.

Acoustic 1D with Python

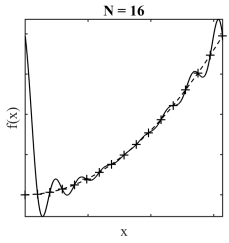
Calculating the 2^{nd} derivatives using the Fourier transform

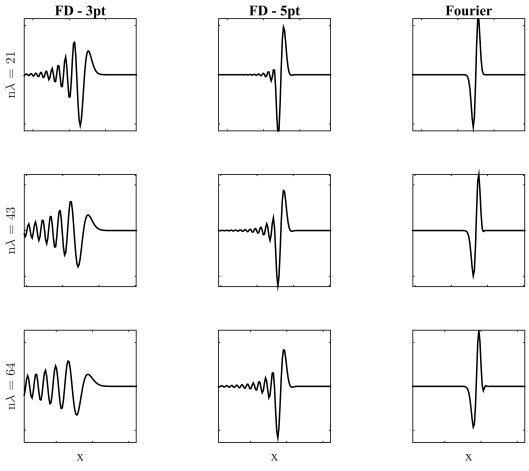
$$\begin{aligned} \partial_x^2 p_j^n &= \mathscr{F}^{-1}[(ik)^2 \, P_\nu^n] \\ &= \mathscr{F}^{-1}[-k^2 \, P_\nu^n] \end{aligned}$$

where P_ν^n is the discrete complex wavenumber spectrum at time n
leading to an exact derivative with only numerical rounding errors.

```
# [...]
# Fourier derivative
def fourier_derivative_2nd(f, dx):
    # [...]
    # Fourier derivative
    ff = (1j * k)**2 * fft(f)
    df = ifft(ff).real
    return df
# [...]
# Time extrapolation
for it in range(nt):
    # 2nd space derivative
    d2p = fourier_derivative_2nd(p, dx)
    # Extrapolation
    pnew = 2 * p - pold + c**2 * dt**2 * d2p
    # Add sources
    pnew = pnew + sg * src[it] * dt**2
    # Remap pressure field
    pold, p = p, pnew
# [...]
```

Exact interpolation/derivative: Fourier Series

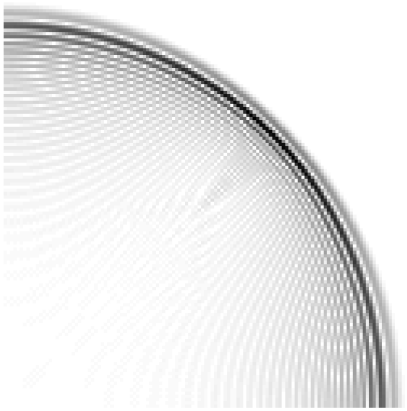




Acoustic 2D

Fourier Method

Finite-Difference Method



1D Elastic wave equation

$$\rho(x)\ddot{u}(x,t) \;=\; \partial_x\left[\mu(x)\partial_x u(x,t)\right] \;+\; f(x,t)$$

u displacement field

μ space-dependent shear modulus

The finite-difference approximation of the extrapolation part leads to

$$\rho_i \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\mathrm{d}t^2} \;=\; \left(\partial_x\left[\mu(x)\partial_x u(x,t)\right]\right)_i^j + f_i^j$$

with space derivatives to be calculated using the Fourier method.

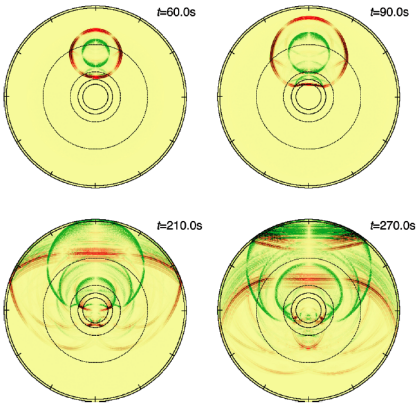
Elastic 1D

The sequence of operations required to obtain the r.h.s. reads

$$u_i^j \rightarrow \mathscr{F}[u_i^j] \rightarrow U_\nu^j \rightarrow -ikU_\nu^j \rightarrow \mathscr{F}^{-1}[-ikU_\nu^j] \rightarrow \partial_x u_i^j$$
$$\partial_x u_i^j \rightarrow \mathscr{F}[\mu_i \partial_x u_i^j] \rightarrow \tilde{U}_\nu^j \rightarrow \mathscr{F}^{-1}[-ik\tilde{U}_\nu^j] \rightarrow \partial_x [\mu(x)\partial_x u(x,t)]$$

where capital letters denote fields in the spectral domain, lower indices with Greek letters indicate discrete frquencies, and $\tilde{U}_\nu^j = \mu_i \partial_x u_i^j$ was introduced as an intermediate result to facilitate notation.

Applications, recent developments



Seismic wave simulation in the Moon (Wang et al., GJI, 2013)

- Axisymmetric wave propagation (Group Prof. Furumura)
- Implementation in spherical coordinates
- Pseudospectral approach in θ direction
- Finite-difference approach in radial direction
- Used in combination with axisem ($\rightarrow$ axisem3d)

Comprehension Questions

- **How much longer** would a simulation take if you want to double the dominant frequency (e.g., 2Hz instead of 1Hz)?
- Can you imagine a strategy to check whether your simulation of a strongly heterogeneous medium is **accurate**?
- What do you think is the reason why the finite-difference method is so popular in the **seismic exploration** domain?
- Can the **Green's function** of the wave equation be simulated?

Exercise - Jupyter Notebooks

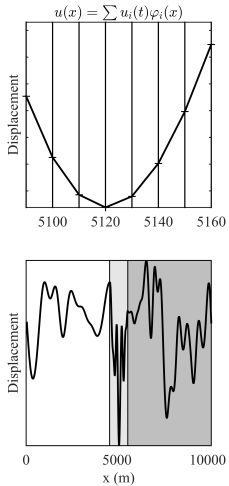
Play around with the following Jupyter notebooks (pseudospectral folder, github, seismo-live, or COURSERA)

- **fourier_acoustic_1d** (compare with fd 3pt and fd 5pt methods, cool!)
- **fourier_acoustic_2d** (note the lack of numerical anisotropy)
- **fourier_derivative** (note the error is only numerical precision!)

Note down **questions** during the process!

The Finite-Element Method

The Finite-Element Method in a Nutshell



- Solution of the *weak form* of the wave equation
- Wavefield is interpolated with (linear) orthogonal basis functions
- Global linear system of equations has to be solved (matrix inversion)
- **Free surface condition implicitly fullfilled**
- Works on hexahedral or tetrahedral meshes

Static Elasticity

Discretization

Departing from the 1D elastic wave equation

$$\rho \partial_t^2 u(x,t) = \partial_x(\mu(x)\partial_x u(x,t)) + f(x,t)$$

we assume the following:

Independent in time: $\partial_t^2 u(x,t) = 0$

Elastic properties of our 1D medium are independent of space: $\mu(x) = \text{const.}$

that leads to the equation

$$-\mu \partial_x^2 u = f\ .$$

Illustration

Static elasticity. A string with homogeneous properties (density and shear modulus) is pulled with a certain force. The Poisson equation determines the displacement of the string given appropriate boundary conditions. Don’t overdo this experiment, in particular if you have old strings.

Weak form

Transform *strong* form into *weak* form by multiplying the equation with an arbitrary space-dependent test function that we denote as $v \rightarrow v(x)$. Then we integrate the equation on both sides over the entire physical domain D with $x \in D$

$$-\int_D \mu \partial_x^2 \, u \, v \, dx = \int_D f \, v \, dx \; .$$

Performing an integration by parts of the left side:

$$-\int_D \mu \partial_x^2 \, u \, v \, dx = \int_D \mu \, \partial_x \, u \, \partial_x \, v \, dx - [\mu \partial_x u \, v]_{x_{min}}^{x_{max}} \; .$$

where the last term is an anti-derivative.

Free surface

Free-surface condition $\Longrightarrow$ No stress at the boundaries.
As the anti-derivative is evaluated at the domain boundaries this implies that this term vanishes.

$$\mu \int_D \partial_x \, u \, \partial_x \, v \, dx = \int f \, v \, dx$$

which is still a description in the continuous world. To enter the discrete world we replace our exact solution $u(x)$ by a $\overline{u}$, a sum over some basis functions φ_i that we do not yet specify

$$u \approx \overline{u}(x) = \sum_{i=1}^{N} u_i \varphi_i \ .$$

Replacing u by $\overline{u}$, we obtain

$$\mu \int_D \partial_x \, \overline{u} \, \partial_x \, v \, dx = \int f \, v \, dx$$

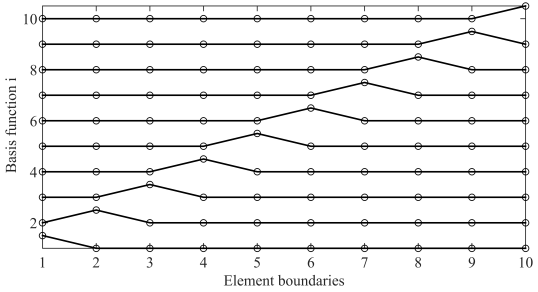
Basis functions

As a choice for our test function $v(x)$ we use the same set of basis functions. Thus $v(x) \to \varphi_j(x)$.

What is the simplemost choice for our basis functions φ_i? Denoting $x_i, i = 1, 2, ..., N$ as the boundaries of our elements we define our basis functions such that $\varphi_i = 1, x = x_i$ and zero elsewhere. Inside the elements our solution field is described by a linear function:

$$\varphi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } x_{i-1} < x \leq x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } x_i < x < x_{i+1} \\ 0 & \text{elsewhere} \end{cases}$$

Basis functions



Linear basis functions for the finite-element method. A 1D domain is discretized with $n-1$ elements having $n=10$ element boundaries (open circles). The basis functions $\varphi_i = 1$ at $x = x_i$. With this basis an arbitrary function can be exactly interpolated at the element boundary points x_i.

Galerkin Principle

We are ready to assemble our discretized version of the weak form by replacing the continuous displacement field by its approximation and applying the Galerkin principle. We obtain

$$\mu \int_D \partial_x \left(\sum_{i=1}^{N} u_i \varphi_i \right) \partial_x \, \varphi_j \, dx = \int f \, \varphi_j \, dx$$

$$\sum_{i=1}^{N} u_i \int_D \mu \, \partial_x \, \varphi_i \, \partial_x \, \varphi_j \, dx = \int f \, \varphi_j \, dx$$

which is a system of N equations as we project the solution on the basis functions φ_j with $j = 1, ..., N$. In the second equation we switched the sequence of integration and sum.

Reference Element, Mapping

Mapping the physical domain to a reference element. In our case we center the local coordinate system denoted as ξ at point x_i and obtain

$$\xi = x - x_i$$
$$h_i = x_i - x_{i-1}$$

where h_i denotes the size of element i defined in the interval $x \in [x_i, x_{i+1}]$. The local basis functions becomes

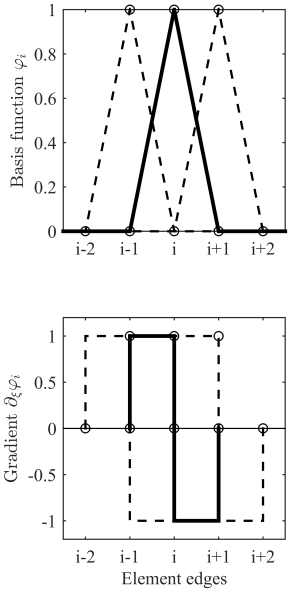
$$\varphi_i(\xi) = \begin{cases} \frac{\xi}{h} + 1 & \text{for } -h < \xi \leq 0 \\ 1 - \frac{\xi}{h} & \text{for } 0 < \xi < h \\ 0 & \text{elsewhere} \end{cases}$$

Reference Element, Mapping

and their derivatives

$$\partial_\xi \varphi_i(\xi) = \begin{cases} \frac{1}{h} & \text{for } -h < \xi \leq 0 \\ -\frac{1}{h} & \text{for } 0 < \xi < h \\ 0 & \text{elsewhere} \end{cases}$$

Reference Element, Mapping



Basis functions and their derivatives. **Top:** The basis function φ_i (thick solid line) is shown along with the neighboring functions $\varphi_{i\pm1}$ (thin dotted lines). **Bottom:** The same for their derivatives with respect to the space coordinate ξ.

Matrix-Vector Notation

The discrete system thus obtained can be written using matrix-vector notation. Defining the solution vector **u** as

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}$$

the source vector **f** as

$$\mathbf{f} = \begin{pmatrix} \int_D f\, \varphi_1\, dx \\ \int_D f\, \varphi_2\, dx \\ \vdots \\ \int_D f\, \varphi_N\, dx \end{pmatrix}$$

Matrix-Vector Notation

and the matrix containing the integral over the basis function derivatives as **K**

$$\mathbf{K} \rightarrow K_{ij} = \mu \int_D \partial_x \, \varphi_i \, \partial_x \, \varphi_j dx$$

Stiffness Matrix

We can now proceed with calculating the elements of the stiffness matrix K defined as

$$K_{ij} = \mu \int_D \partial_x \varphi_i \, \partial_x \, \varphi_j \, dx$$

with the corresponding expression in local coordinates ξ

$$K_{ij} = \mu \int_{D_\xi} \partial_\xi \varphi_i \, \partial_\xi \varphi_j \, d\xi \, .$$

Stiffness Matrix

Let us calculate some of the elements of matrix K_{ij} starting with the diagonal elements. For example, for K_{11} we obtain

$$\begin{aligned} K_{11} &= \mu \int_D \partial_x \varphi_1 \, \partial_x \varphi_1 \, dx \\ &= \mu \int_0^h \frac{-1}{h} \frac{-1}{h} d\xi \; = \frac{\mu}{h^2} \int_0^h d\xi = \frac{\mu}{h} \end{aligned}$$

Stiffness Matrix

For diagonal element A_{22} the derivatives overlap in element 1 and 2 implying the integration has to be performed for the interval $\xi \in [-h, h]$.

$$
\begin{aligned}
K_{22} &= \mu \int_D \partial_x \varphi_2 \, \partial_x \varphi_2 \, dx \\
&= \mu \int_{-h}^{0} \partial_\xi \varphi_2 \, \partial_\xi \varphi_2 \, d\xi + \mu \int_0^h \partial_\xi \varphi_2 \, \partial_\xi \varphi_2 \, d\xi \\
&= \frac{\mu}{h^2} \int_{-h}^{0} d\xi + \frac{\mu}{h^2} \int_0^h = \frac{2\mu}{h}
\end{aligned}
$$

Stiffness Matrix

Equivalently, the off-diagonal terms overlap only in one element, for example

$$
\begin{aligned}
K_{12} &= \mu \int_D \partial_x \varphi_1 \, \partial_x \varphi_2 \, dx \\
&= \mu \int_0^h \partial_\xi \varphi_1 \, \partial_\xi \varphi_2 \, d\xi = \mu \int_0^h \frac{-1}{h} \frac{1}{h} d\xi \\
&= \frac{-\mu}{h^2} \int_0^h d\xi = \frac{-\mu}{h} \\
K_{21} &= K_{12}
\end{aligned}
$$

Stiffness Matrix

Finally, the stiffness matrix for an elastic physical system with constant shear modulus μ and element size h reads

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

Note: Space-dependent terms in our linear system are proportional to the 3-point operator matrix for a 2nd finite-difference derivative

Solution

System of equations can be written in component form as

$$u_i \, K_{ij} = f_j$$

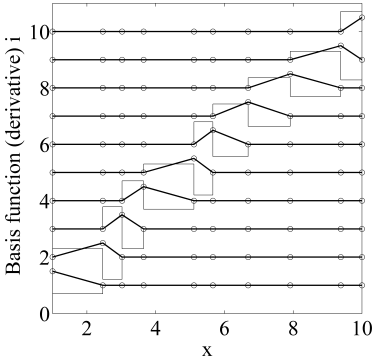
where we use the Einstein summation convention and in matrix-vector notation

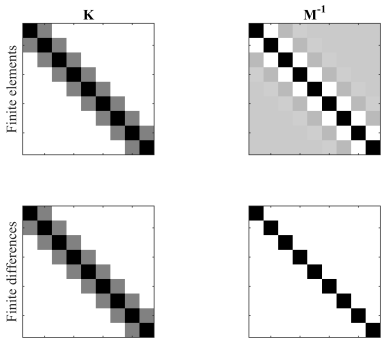
$$\mathbf{K}^T \, \mathbf{u} = \mathbf{f}$$

Note: Matrix **K** is called the it stiffness matrix.
This system of equations has as many unknowns as equations. Provided that the matrix is positive definite we can determine its inverse:

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

Linear basis functions, system matrices





Linear finite-element method and low-order finite-difference method are basically identical

The Finite-Element Method: 1D Elastic Wave Equation

1D Elastic Wave Equation

Apply the Galerkin principle to the 1D elastic wave equation

$$\rho \partial_t^2 u = \partial_x \, \mu \, \partial_x u + f \, .$$

where again we omit space and time-dependencies. From now on we assume that the properties of the medium, density ρ, and shear modulus μ are both space-dependent. We obtain the weak form as

$$\int_D \rho \, \partial_t^2 \, u \, \varphi_j \, dx = \int_D \partial_x \, \mu \, \partial_x \, u \, \varphi_j \, dx + \int_D f \, \varphi_j \, dx \, .$$

Integration by parts of the term containing the space derivatives leads to

$$\int_D \partial_x \, \mu \, \partial_x \, u \, \varphi_j \, dx = \left[\mu \partial_x \, u \, \varphi_j \right] - \int_D \mu \, \partial_x \, u \, \partial_x \, \varphi_j \, dx$$

1D Elastic Wave Equation

Assuming a stress-free boundary condition leads to

$$\int_D \rho\, \partial_t^2\, u\, \varphi_j\, dx + \int_D \mu\, \partial_x\, u\, \partial_x\, \varphi_j\, dx = \int_D f\, \varphi_j\, dx$$

where u is the continuous unknown displacement field. We replace the exact displacement field by an approximation $\overline{u}$ of the form

$$u(x,t) \to \overline{u}(x,t) = \sum_{i=1}^{N} u_i(t)\, \varphi_i(x)$$

where the coefficients u_i are expected to correspond to a discrete representation of the solution field. The wave equation bevomes

$$\int_D \rho\, \partial_t^2\, \overline{u}\, \varphi_j\, dx + \int_D \mu\, \partial_x\, \overline{u}\, \partial_x\, \varphi_j\, dx = \int_D f\, \varphi_j\, dx$$

1D Elastic Wave Equation

Turning the continuous weak form into a system of linear equations

$$\int_D \rho\, \partial_t^2 \left(\sum_{i=1}^{N} u_i(t)\, \varphi_i(x) \right) \varphi_j \, dx$$
$$+ \int_D \mu\, \partial_x \left(\sum_{i=1}^{N} u_i(t)\, \varphi_i(x) \right) \partial_x \varphi_j \, dx$$
$$= \int_D f\, \varphi_j \, dx \, .$$

Changing the order of integration and summation we obtain

$$\sum_{i=1}^{N} \partial_t^2 u_i \int_D \rho\, \varphi_i\, \varphi_j \, dx + \sum_{i=1}^{N} u_i \int_D \mu\, \partial_x \varphi_i\, \partial_x \varphi_j \, dx = \int_D f\, \varphi_j \, dx$$

using the fact that the unknown coefficients u_i only depend on time.

1D Elastic Wave Equation

Using matrix-vector notation with the following definitions for the time-dependent solution vector of displacement values $\mathbf{u}(t)$, mass matrix $\mathbf{M}$, the already well-known stiffness matrix $\mathbf{K}$, and the source vector $\mathbf{f}$:

$$\begin{aligned}
\mathbf{u}(t) &\rightarrow u_i(t) \\
\mathbf{M} &\rightarrow M_{ij} = \int_D \rho\, \varphi_i\, \varphi_j\, dx \\
\mathbf{K} &\rightarrow K_{ij} = \int_D \mu\, \partial_x\, \varphi_i\, \partial_x\, \varphi_j dx \\
\mathbf{f} &\rightarrow f_j = \int_D f\, \varphi_j\, dx\ .
\end{aligned}$$

1D Elastic Wave Equation

Thus we can write the system of equations as

$$\ddot{\mathbf{u}}\mathbf{M} + \mathbf{u}\mathbf{K} = \mathbf{f}$$

or with transposed system matrices as

$$\mathbf{M}^T\ddot{\mathbf{u}} + \mathbf{K}^T\mathbf{u} = \mathbf{f}\,.$$

For the second time-derivative we use a standard finite-difference approximation

$$\ddot{\mathbf{u}} = \partial_t^2\mathbf{u} \approx \frac{\mathbf{u}(t+dt) - 2\mathbf{u}(t) + \mathbf{u}(t-dt)}{dt^2}$$

1D Elastic Wave Equation

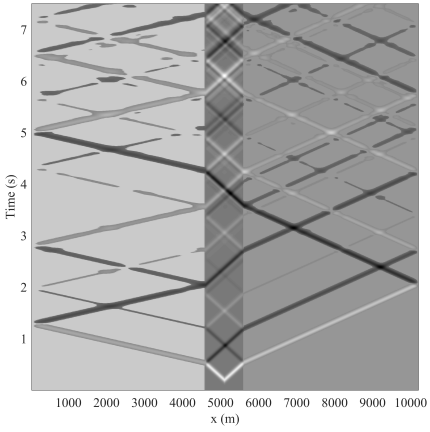
Replacing the original partial derivative with respect to time to obtain

$$\mathbf{M}^T \left[\frac{\mathbf{u}(t+dt) - 2\mathbf{u}(t) + \mathbf{u}(t-dt)}{dt^2} \right] = \mathbf{f} - \mathbf{K}^T \mathbf{u} \, .$$

Starting from an initial state $\mathbf{u}(t=0) = 0$ we can determine the displacement field at time $t + dt$ by

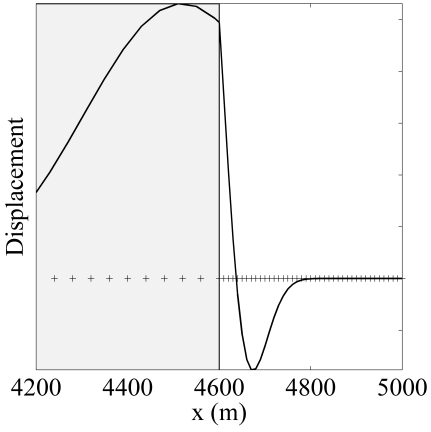
$$\mathbf{u}(t+dt) = dt^2 (\mathbf{M}^T)^{-1} \left[\mathbf{f} - \mathbf{K}^T \mathbf{u} \right] + 2\mathbf{u}(t) - \mathbf{u}(t-dt) \, .$$

h - adaptivity



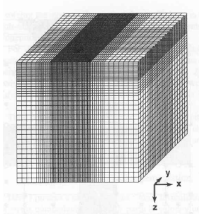
Snapshots of displacement values are shown as a function time. Where displacement amplitudes are below a threshold, the velocity models is shown in gray scale. Note the polarity change of the reflections at the boundaries and the slope of the signals in the $x - t$ plane indicated their velocities.

h - adaptivity



Detail of the finite-element simulation with varying element size at one of the domain boundaries. The crosses indicate the element boundaries and the changing element size. Note the continuous but non-differentiable behaviour of the displacement field at the interface.

Applications, Recent Developments



(a) Regular grid

(c) Octree

Finite element mesh, octree approach

(Bielak et al.)

- Requires linear algebra libraries for matrix inversion
- Suboptimal for parallelization
- Allows arbitrary geometric complexity
- Curbed element boundaries possible
- Standard in engineering applications
- Hardly used in seismology (why?)

Questions?

Exercise - Jupyter Notebooks

Play around with the following Jupyter notebooks (finite element folder, github, seismo-live, or COURSERA)

- **fe_elastic_1d** (note the dense solution scheme!)
- **fe_static_elasticity** (comparison with FD realaxation method)

Note down **questions** during the process!