

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

2021 International Conference on Earthquake Engineering and Seismology

---

Heiner Igel

October 11, 2021

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

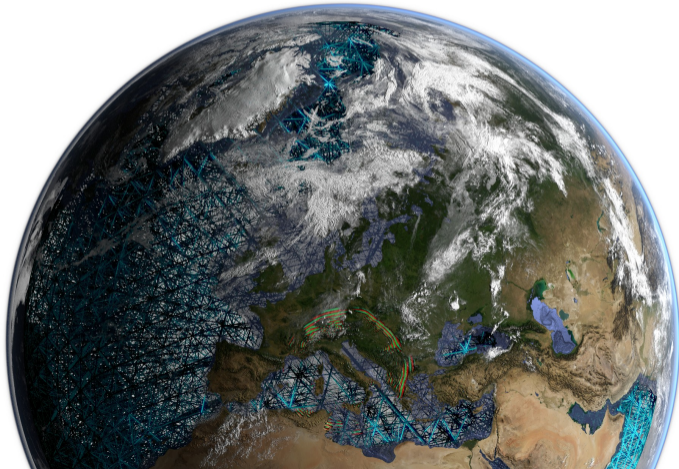
## Format of Short Course (approx.)

- 3 one-hour blocks
- 40 minutes lecture
- 20 minutes
  - Questions by you (also allowed during lecture!)
  - A few comprehensive questions to be discussed in subgroups
  - Checking answers

# Course Content

- Block 1
  - Introduction - Motivation
  - Some fundamentals of wave propagation
  - The finite-difference method
  - The pseudospectral method
- Block 2
  - The finite (spectral) element method
- Block 3
  - The finite-volume method
  - The discontinuous Galerkin method
  - Outlook

# Introduction





## Goals of lecture

- Who needs **computational seismology**?
- What are some fundamental aspects of computational **wave propagation**?
- Is it a tough or an easy problem as far as **computational resources** are concerned?
- Which **numerical methods** are on the market, basic principles, and domains of application?
- What options do you have to get training (**Jupyter notebooks**, **COURSERA**, etc) ...?

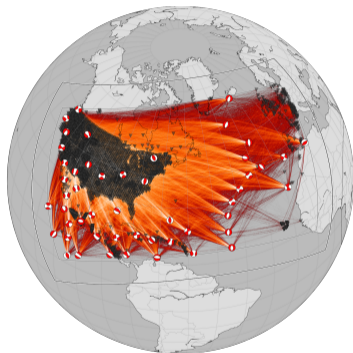
# What is Computational Seismology?

We define **computational seismology** such that it **involves the complete solution of the seismic wave propagation (and rupture) problem for arbitrary 3-D models by numerical means.**

## What is not covered ... but you can do tomography with ...

- **Ray-theoretical** methods
- **Quasi-analytical** methods (e.g., normal modes, reflectivity method)
- **Frequency-domain** solutions
- **Boundary integral** equation methods
- **Discrete particle** methods

These methods are important for **benchmarking** numerical solutions!



# Who needs Computational Seismology

Many problems rely on the analysis of **elastic wavefields**

- **Global seismology** and tomography of the Earth's interior
- The quantification of **strong ground motion - seismic hazard**
- The understanding of the **earthquake source process**
- The monitoring of **volcanic processes** and the forecasting of eruptions
- **Earthquake early warning** systems
- **Tsunami early warning** systems
- Local, regional, and global **earthquake services**
- Global monitoring of **nuclear tests**
- **Laboratory scale analysis** of seismic events

## Who needs Computational Seismology (cont'd)

(...)

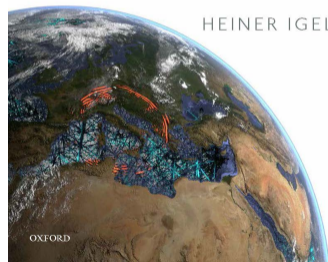
- Ocean generated **noise measurements** and cross-correlation techniques
- **Planetary seismology - Apollo, INSIGHT**
- **Exploration geophysics**, reservoir scale seismics
- **Geotechnical engineering** (non-destructive testing, small scale tomography)
- **Medical applications**, breast cancer detection, reverse acoustics

- Igel, **Computational Seismology: A Practical Introduction** (Oxford University Press, 2016)
- Shearer, **Introduction to Seismology** (3rd edition, 2019)
- Aki and Richards, **Quantitative Seismology** (1st edition, 1980)
- Mozco, **The Finite-Difference Modelling of Earthquake Motions** (Cambridge University Press)
- Fichtner, **Full Seismic Waveform Modelling and Inversion** (Springer Verlag, 2010).

## COMPUTATIONAL SEISMOLOGY

*A Practical Introduction*

HEINER IGEL



# Why numerical methods?



Photo ©Überraschungsbilder via Wikimedia commons

# Computational Seismology, Memory, and Compute Power

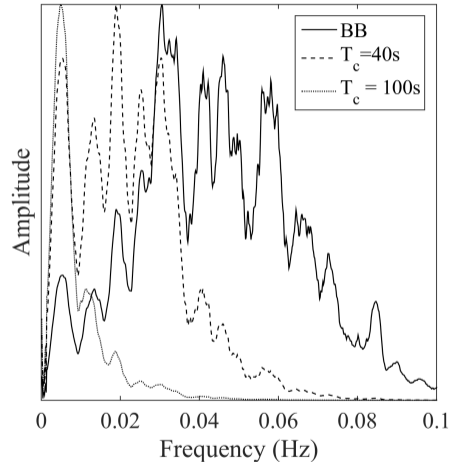
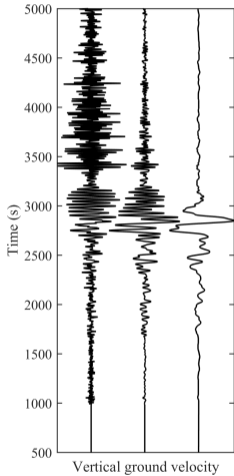
Numerical solutions necessitate the discretization of Earth models. Estimate how much memory is required to store the Earth model and the required displacement fields.

**Are we talking laptop or supercomputer?**



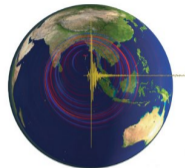


# Seismic Wavefield Observations



## Exercise: Sampling a global seismic wavefield

- The highest frequencies that we observe for global wave fields is 1Hz.
- We assume a homogeneous Earth (radius 6371km).
- P velocity  $v_p = 10\text{km/s}$  and the  $v_p/v_s$  ratio is  $\sqrt{3}$
- We want to use 20 **grid points (cells) per wavelength**
- How many grid cells would you need (assume cubic cells).
- What would be their size?
- How much memory would you need to store one such field (e.g., density in single precision).



You may want to make use of

$$c = \frac{\lambda}{T} = \lambda f = \frac{\omega}{k}$$



## Exercise: Solution (Matlab)

```
% Earth volume
v_e = 4/3 * pi * 6371^3;
% smallest velocity (ie, wavelength)
vp=10; vs=vp/sqrt(3);
% Shortest Period
T=10;
% Shortest Wavelength
lam=vs*T;
% Number of points per wavelength and
% required grid spacing
nplambda = 20;
dx = lam / nplambda;
% Required number of grid cells
nc = v_e / (dx^3);
% Memory requirement (TBytes)
mem = nc * 8/1000/1000/1000/1000;
```

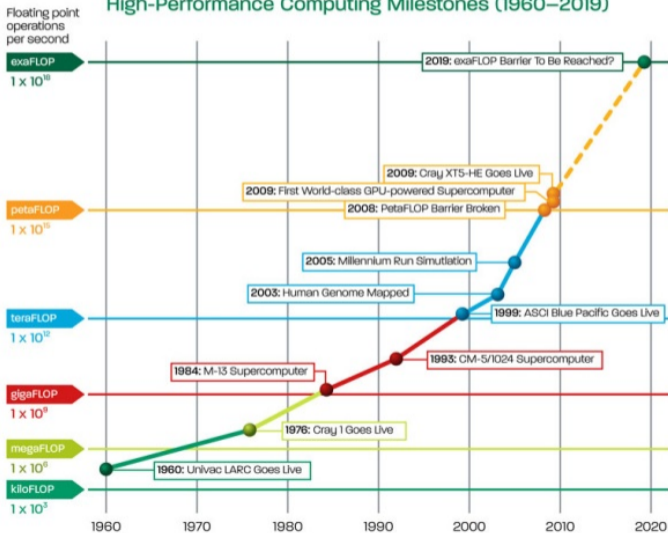
Results (@ $T = 1s$ ) : 360 TBytes  
Results (@ $T = 10s$ ) : 360 GBytes  
Results (@ $T = 100s$ ) : 360 MBytes

# Computational Seismology, Memory, and Compute Power

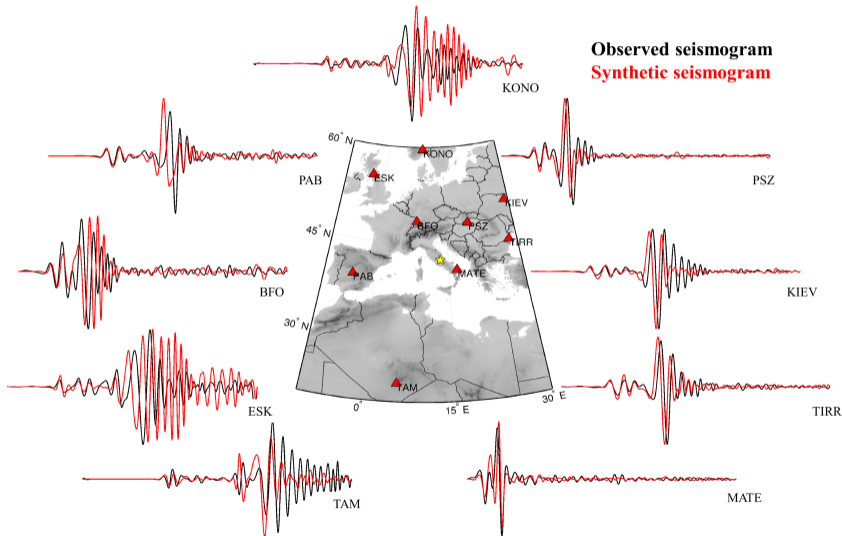


## High-Performance Computing Milestones (1960–2019)

1960: 1 MFlops  
1970: 10MFlops  
1980: 100MFlops  
1990: 1 GFlops  
1998: 1 TFlops  
2008: 1 Pflops  
20??: 1 EFlops



# The Ultimate Goal: Matching Wavefield Observations



## **A Bit of Wave Physics**

---

# Acoustic wave equation: no source

## Acoustic wave equation

$$\partial_t^2 p = c^2 \Delta p + s$$

$p \rightarrow p(\mathbf{x}, t)$ , pressure

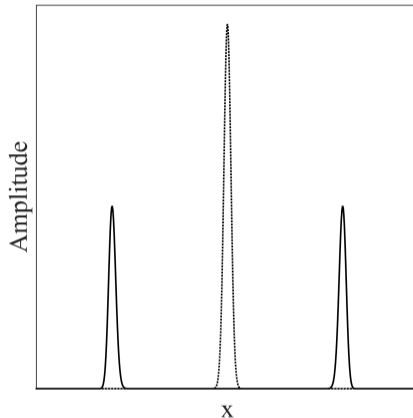
$c \rightarrow c(\mathbf{x})$ , velocity

$s \rightarrow s(\mathbf{x}, t)$ , source term

## Initial conditions

$$p(\mathbf{x}, t = 0) = p_0(\mathbf{x}, t)$$

$$\partial_t p(\mathbf{x}, t = 0) = 0$$



Snapshot of  $p(\mathbf{x}, t)$  (solid line) after some time for initial condition  $p_0(\mathbf{x}, t)$  (Gaussian, dashed line), 1D case.

# Acoustic wave equation: external source

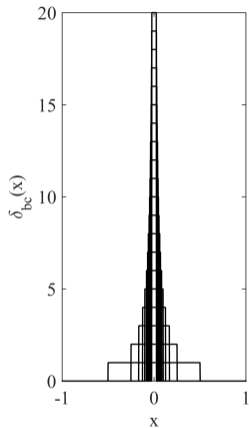
## Green's Function $G$

$$\partial_t^2 G(\mathbf{x}, t; \mathbf{x}_0, t_0) - c^2 \Delta G(\mathbf{x}, t; \mathbf{x}_0, t_0) = \delta(\mathbf{x} - \mathbf{x}_0) \delta(t - t_0)$$

## Delta function $\delta$

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x) dx = 1, \quad \int_{-\infty}^{\infty} f(x) \delta(x) dx = f(0)$$



$\delta$ -generating function using boxcars.

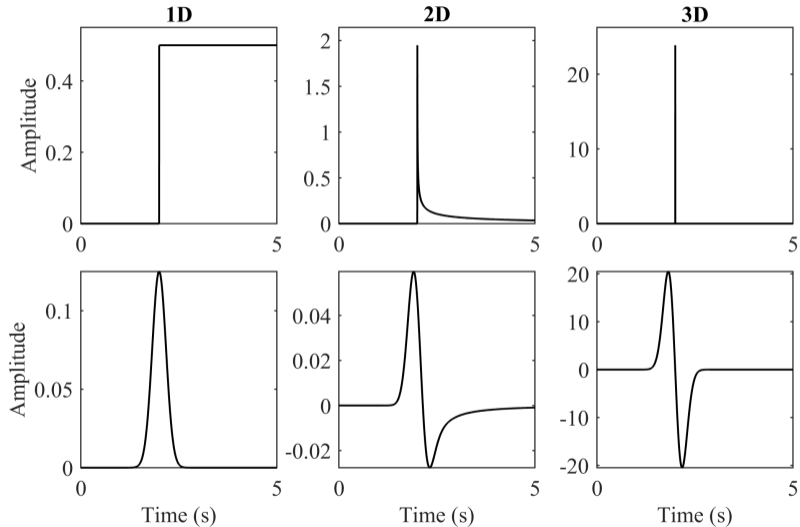


## Acoustic wave equation: analytical solutions

Green's functions for the inhomogeneous acoustic wave equation for all dimensions.  $H(t)$  is the Heaviside function.

1D	2D	3D
$\frac{1}{2c} H\left(t - \frac{ r }{c}\right)$	$\frac{1}{2\pi c^2} \frac{H\left(t - \frac{ r }{c}\right)}{\sqrt{t^2 - \frac{r^2}{c^2}}}$	$\frac{1}{4\pi c^2 r} \delta\left(t - r/c\right)$
$r = x$	$r = \sqrt{x^2 + y^2}$	$r = \sqrt{x^2 + y^2 + z^2}$

# Acoustic wave equation: analytical solutions



# Wave Equation as Linear System

Seismogram for arbitrary source  $s(t)$  as convolution (exact)

$$p(\mathbf{x}, t) = G(\mathbf{x}, t, \mathbf{x}_0) \otimes s(t)$$

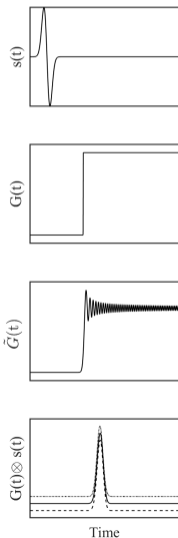
Seismogram for arbitrary source  $s(t)$  as convolution (numerical)

$$\tilde{p}(\mathbf{x}, t) = \tilde{G}(\mathbf{x}, t, \mathbf{x}_0) \otimes s(t)$$

## Important consequence:

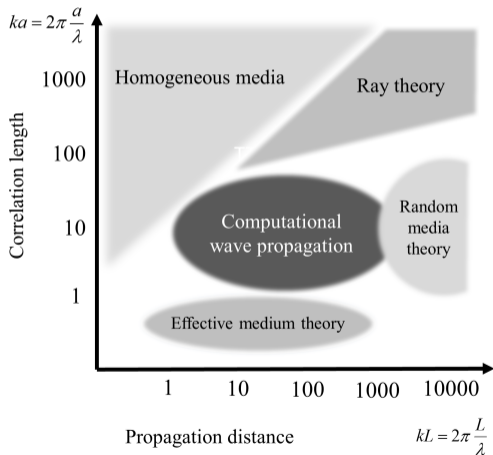
Even if your numerical Green's function  $\tilde{G}(\mathbf{x}, t, \mathbf{x}_0)$  is inaccurate, the numerical solution  $\tilde{p}(\mathbf{x}, t)$  might be very accurate provided the  $s(t)$  is defined in the right frequency band!

# Wave Equation as Linear System



- Accurate Green's functions cannot be calculated numerically
- A numerical solver is a **linear system**
- The convolution theorem applies
- Inaccurate simulations can be filtered afterwards
- Source time functions can be altered afterwards
- ... provided the sampling is good enough ...

# Spatial Scales, Scattering, Solution Strategies



- Recorded seismograms are affected by ...
- ... the ratio of seismic wavelength  $\lambda$  and structural wavelength  $a$  ...
- ... how many wavelengths are propagated ...
- strong scattering when  $a \approx \lambda \rightarrow$  numerical methods
- ray theory works when  $a \gg \lambda$
- random medium theory necessary for strong scattering media (and long distances)

# Numerical Methods for Wave Propagation Problems

---

# What's on the market

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

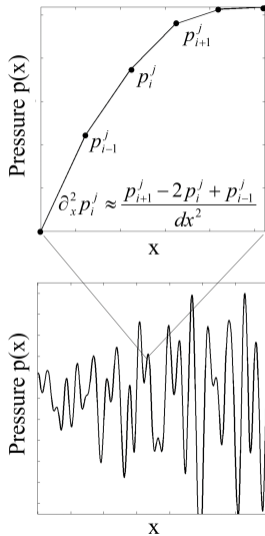


# The Finite-Difference Method

---



# Finite differences in a Nutshell



- Direct numerical approximation of partial derivatives using **finite-differences**
- Local computational scheme → **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!

## Acoustic waves in 1D

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

The constant density acoustic wave equation in 1D

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

imposing pressure-free conditions at the two boundaries  
as

$$p(x) |_{x=0,L} = 0$$

## 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  $dt$ .

$$\begin{aligned}x_j &= jdx, & j &= 0, j_{max} \\t_n &= ndt, & n &= 0, n_{max}\end{aligned}$$

## 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

$$p_j^{n+1} \rightarrow \rho(x_j, t_n + dt)$$

$$p_j^n \rightarrow \rho(x_j, t_n)$$

$$p_j^{n-1} \rightarrow \rho(x_j, t_n - dt)$$

$$p_{j+1}^n \rightarrow \rho(x_j + dx, t_n)$$

$$p_j^n \rightarrow \rho(x_j, t_n)$$

$$p_{j-1}^n \rightarrow \rho(x_j - dx, t_n)$$

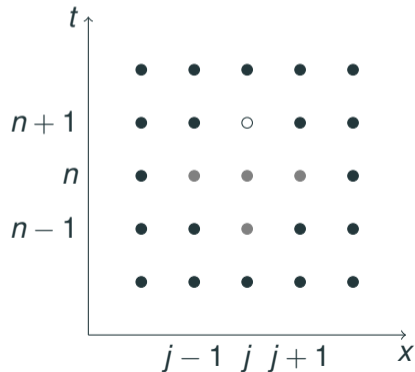
.

## Acoustic waves in 1D

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{dt^2} = c_j^2 \left[ \frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{dx^2} \right] + s_j^n.$$

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  $p_j^{n+1}$ :

$$p_j^{n+1} = c_j^2 \frac{dt^2}{dx^2} [p_{j+1}^n - 2p_j^n + p_{j-1}^n] + 2p_j^n - p_j^{n-1} + dt^2 s_j^n$$

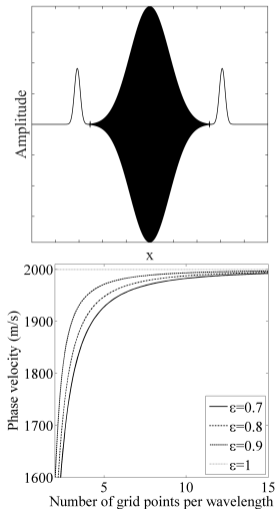
The initial condition of our wave simulation problem is such that everything is at rest at time  $t = 0$ :

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

## 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
```

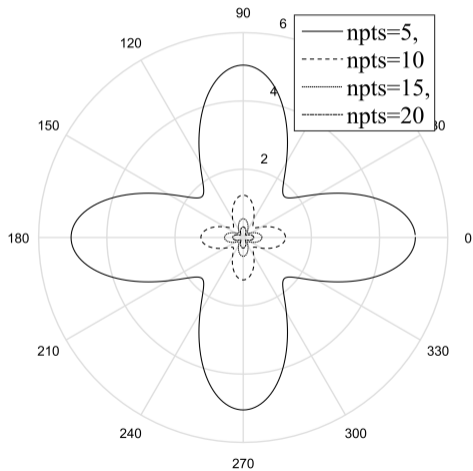
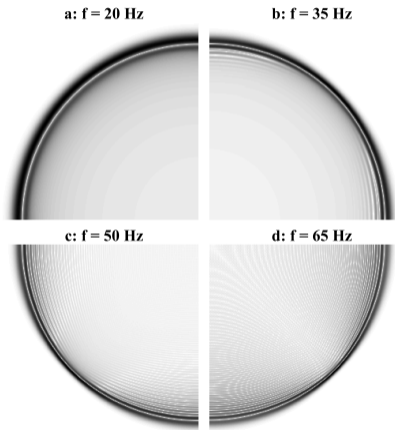
# von Neumann Analysis, Stability, Dispersion



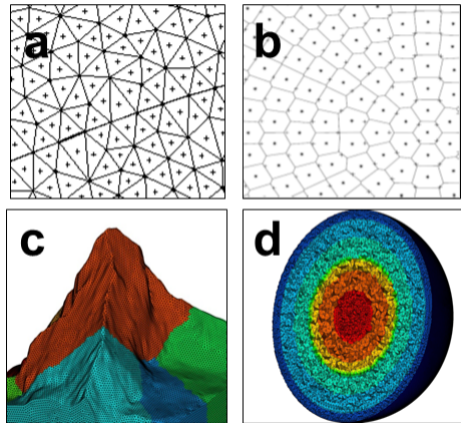
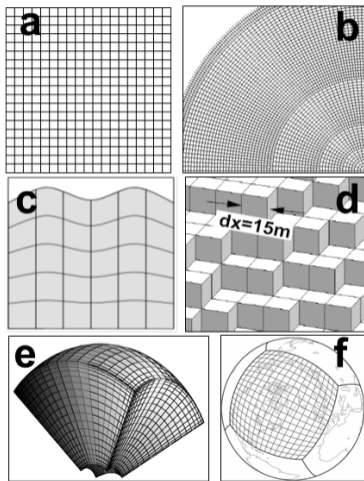
- Plane waves in a discrete world
- $p(x, t) = e^{kj dx - \omega n dt}$
- Simulations are conditionally stable
- $c \frac{dt}{dx} \leq \epsilon \approx 1$  CFL - criterion
- Simulated phase velocity becomes numerically dispersive!
- The more points per wavelength the more accurate
- How to check?



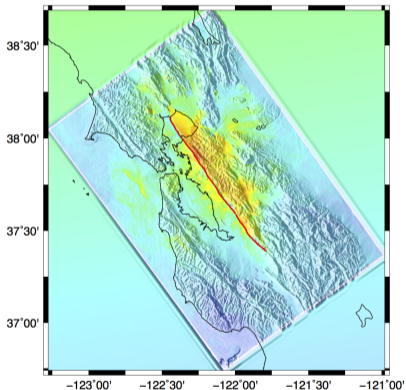
# Numerical Anisotropy



# Meshes, grids, structured, unstructured



# Applications, recent , community codes



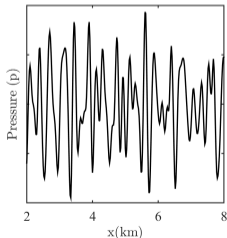
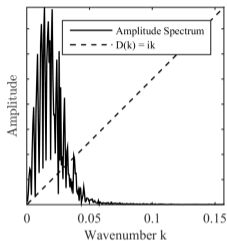
Source: geodynamics.org (SW4)

- 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

## Forward Transform

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

## Inverse Transform

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

## Fourier Series and Transforms

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  $\mathcal{F}$  we can obtain an exact  $n$  –  $th$  derivative using

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



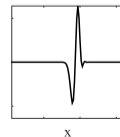
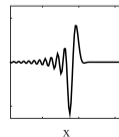
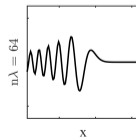
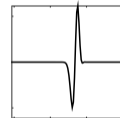
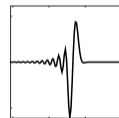
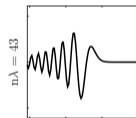
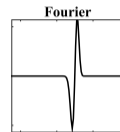
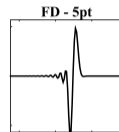
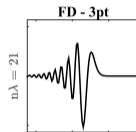
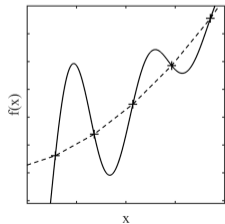
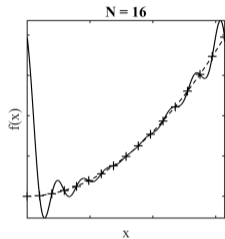
Calculating the 2<sup>nd</sup> derivatives using the Fourier transform

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

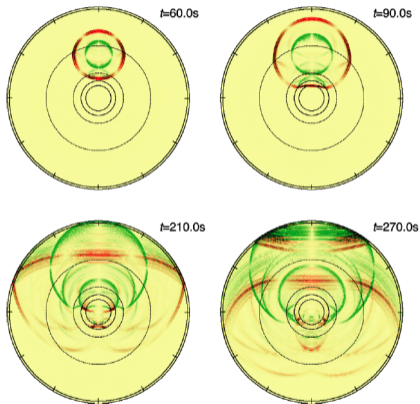
where  $P_\nu^n$  is the discrete complex wavenumber spectrum at time  $n$  leading to an exact derivative with only numerical rounding errors.

```
# [...]  
# 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



# 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  $\theta$  direction
- Finite-difference approach in radial direction
- Used in combination with axisem ( $\rightarrow$  axisem3d)

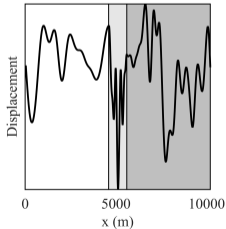
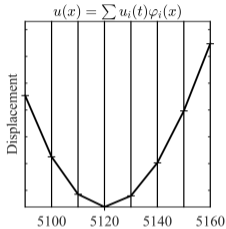
# I: Comprehensive Questions

- 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?

# 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 fulfilled**
- 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  $\implies$  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  $\bar{u}$ , a sum over some basis functions  $\varphi_i$  that we do not yet specify

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

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

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

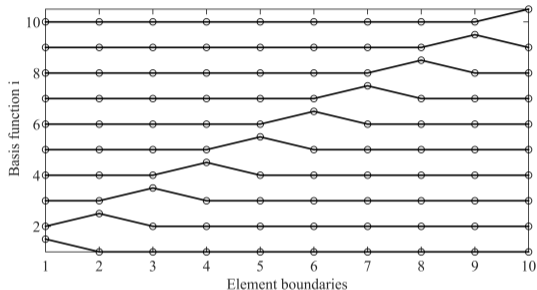
## Basis functions

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

**What is the simplest choice for our basis functions  $\varphi_i$ ?** Denoting  $x_i, i = 1, 2, \dots, N$  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  $x_j$ .

## 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  $\varphi_j$  with  $j = 1, \dots, N$ . In the second equation we switched the sequence of integration and sum.

## Matrix-Vector Notation

The discrete system thus obtained can be written using matrix-vector notation. Defining the solution vector  $\mathbf{u}$  as

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

the source vector  $\mathbf{f}$  as

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

and the matrix containing the integral over the basis function derivatives as  $\mathbf{K}$

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



## 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

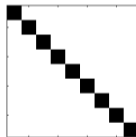
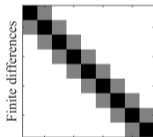
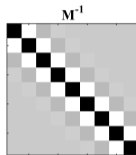
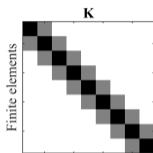
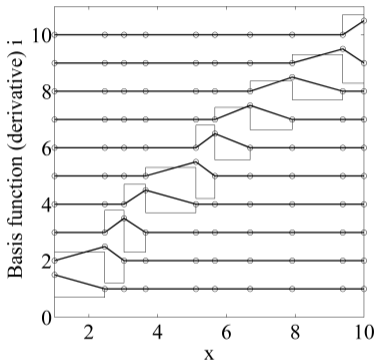
$$\mathbf{K}^T \mathbf{u} = \mathbf{f}$$

Note: Matrix  $\mathbf{K}$  is called the 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

## 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  $\mathbf{f}$ :

$$\mathbf{u}(t) \rightarrow u_i(t)$$

$$\mathbf{M} \rightarrow M_{ij} = \int_D \rho \varphi_i \varphi_j dx$$

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

$$\mathbf{f} \rightarrow f_j = \int_D f \varphi_j dx .$$

## 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

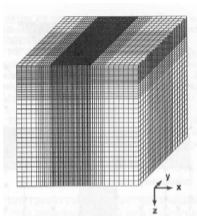
Replacing the original partial derivative with respect to time to obtain

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

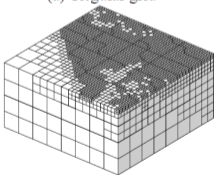
Starting from an initial state  $\mathbf{u}(t = 0) = 0$  we can determine the displacement field at time  $t + dt$  by

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

# Applications, Recent Developments



(a) Regular grid



(c) Octree

Finite element mesh, octree approach

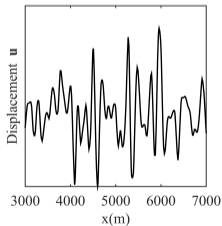
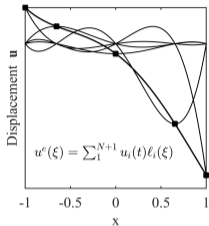
(Bielak et al.)

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

# The Spectral-Element Method

---

# The Spectral-Element Method in a Nutshell



- Same mathematical derivation as the finite-element method
- Lagrange polynomial representation of wave field
- Gauss-Lobatto-Legendre collocation points (stability!)
- **Diagonal mass matrix** → trivially inverted
- Explicit extrapolation scheme → efficient parallelisation
- Method of choice for global wave propagation (specfem, axisem)
- Meshing required



## Lagrange polynomials

Remember we seek to approximate  $u(x, t)$  by a sum over space-dependent basis functions  $\psi_i$  weighted by time-dependent coefficients  $u_i(t)$ .

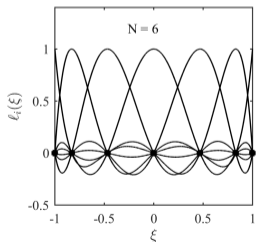
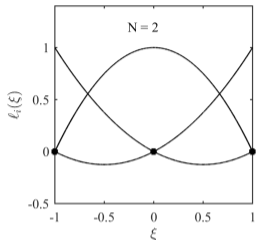
$$u(x, t) \approx \bar{u}(x, t) = \sum_{i=1}^n u_i(t) \psi_i(x)$$

Our final choice: Lagrange polynomials:

$$\psi_i \rightarrow \ell_i^{(N)} := \prod_{k=1, k \neq i}^{N+1} \frac{\xi - \xi_k}{\xi_i - \xi_k}, \quad i = 1, 2, \dots, N+1$$

where  $x_i$  are fixed points in the interval  $[-1, 1]$ .

# Lagrange polynomials graphically



**Top:** Family of  $N + 1$  Lagrange polynomials for  $N = 2$  defined in the interval  $\xi \in [-1, 1]$ . Note their maximum value in the whole interval does not exceed unity.

**Bottom:** Same for  $N = 6$ . The domain is divided into  $N$  intervals of uneven length. When using Lagrange polynomials for function interpolation the values are exactly recovered at the collocation points (squares).

# Gauss-Lobatto-Legendre points

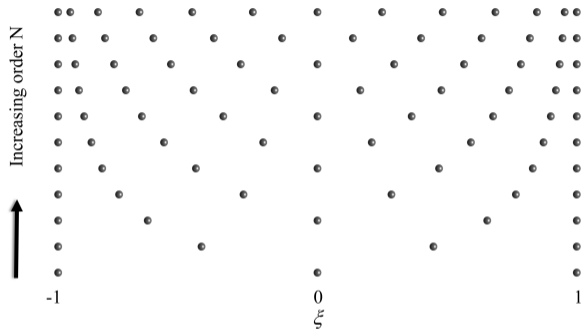


Illustration of the spatial distribution of Gauss-Lobatto-Legendre points in the interval  $[-1, 1]$  from bottom to top for polynomial order 1 to 12. Note the increasing difference of largest to smallest interval between collocation points! Consequences?

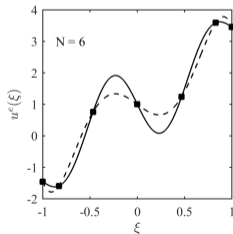
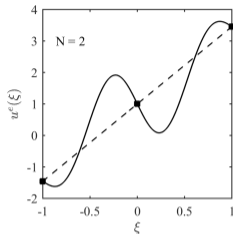
## Function approximation

This is the final mathematical description of the unknown field  $u(x, t)$  for the spectral-element method based on Lagrange polynomials.

$$u^e(\xi) = \sum_{i=1}^{N+1} u^e(\xi_i) \ell_i(\xi)$$

Other options at this point are the Chebyhev polynomials. They have equally good approximation properties (but ...)

# Interpolation with Lagrange Polynomials



The function to be approximated is given by the solid lines. The approximation is given by the dashed line exactly interpolating the function at the GLL points (squares). **Top:** Order  $N = 2$  with three grid points. **Bottom:** Order  $N = 6$  with seven grid points.

## Integration scheme for an arbitrary function $f(x)$

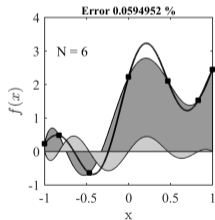
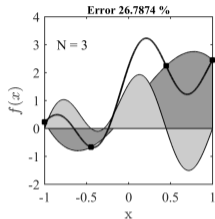
$$\int_{-1}^1 f(x)dx \approx \int_{-1}^1 P_N(x)dx = \sum_{i=1}^{N+1} w_i f(x_i)$$

defined in the interval  $x \in [-1, 1]$  with

$$P_N(x) = \sum_{i=1}^{N+1} f(x_i) \ell_i^{(N)}(x)$$

$$w_i = \int_{-1}^1 \ell_i^{(N)}(x)dx .$$

# Numerical integration scheme



- Exact function (thick solid line)
- Approximation by Lagrange polynomials (thin solid line)
- Difference between true and approximate function (light gray)

## Extrapolation for time-dependent coefficients $\mathbf{u}_g$

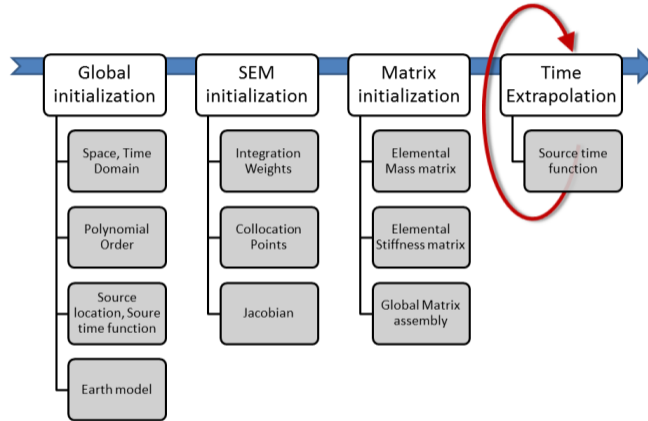
This is our final algorithm as it is implemented using Matlab or Python

$$\begin{aligned}\mathbf{u}_g(t + dt) = dt^2 & \left[ \mathbf{M}_g^{-1} (\mathbf{f}_g(t) - \mathbf{K}_g \mathbf{u}_g(t)) \right] \\ & + 2\mathbf{u}_g(t) - \mathbf{u}_g(t - dt)\end{aligned}$$

Looks fairly simple, no?



# Spectral elements: work flow

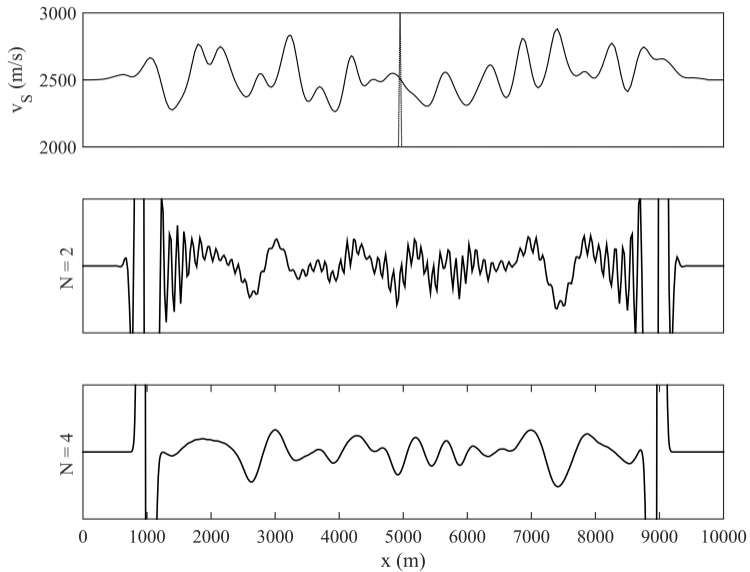


A substantial part consists of preparing the interpolation and integration procedures required to initialize the global mass- and stiffness matrices. The final time-extrapolation is extremely compact and does not require the inversion of a global matrix as is the case in classic finite-element methods.

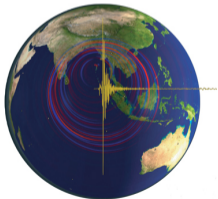
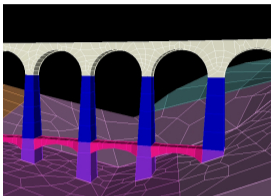
## Python Code: Time Extrapolation

```
# -----  
# Time extrapolation  
# -----  
x_t = []  
for it in range(nt):  
    # Source initialization  
    f = np.zeros(ng)  
    if it < len(src):  
        f[isrc-1] = src[it-1]  
  
    # Time extrapolation  
    unew = dt**2 * Minv @ (f - K @ u) + 2 * u - uold  
    uold, u = u, unew  
  
    # Solution in space-time  
    x_t.append(u)
```

# Example, convergence



# Applications, Recent Developments



- Many applications in **regional** and **global seismology**
- Method of choice whenever **surface waves** are involved
- Spherical geometry with **cubed sphere** approach
- Applications to soil-structure interaction
- Works for hexahedral and tetrahedral meshes (→ *salvus*)
- ***specfem*** or ***salvus*** are widely used community software for seismology (3D)

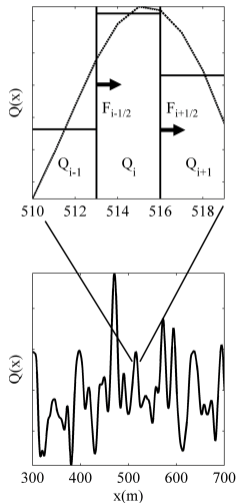
## II: Comprehensive Questions

- What are some major points that speak for a finite- (spectral-) element type solution to your problem?
- Why are spectral-element methods so efficient on parallel computers despite the large system of equations to solve?
- Why do you think is the spectral element method called *spectral*?

# The Finite-Volume Method

---

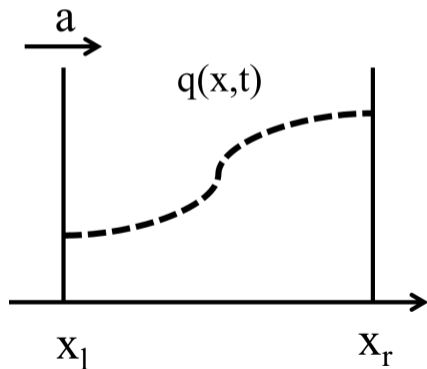
# The Finite-Volume Method in a Nutshell



- Mathematically derived from energy and mass conservation law
- Spatial discretization with **arbitrary volumes**
- Extreme geometric flexibility (e.g., shock waves)
- **Voronoi cells**, tetrahedra, polygons
- Entirely local formulation (cell based)
- Communication with neighbours through flux scheme
- Hardly used in seismology

## The Finite-Volume Method via Conservation Laws

To describe what is happening we put ourselves into a finite volume cell that we denote as  $\mathcal{C}$  and denote the boundaries as  $x \in x_1, x_2$ . We further assume a positive advection speed  $a$ .





## The Finite-Volume Method via Conservation Laws

The total mass of any quantity inside the cell is

$$\int_{x_1}^{x_2} q(x, t) dx$$

and a change in time can only be due to fluxes across the left and/or right cell boundaries. Thus

$$\partial_t \int_{x_1}^{x_2} q(x, t) dx = F_1(t) - F_2(t)$$

where  $F_i(t)$  are rates (e.g., in  $g/s$ ) at which the quantity flows through the boundaries.

## The Finite-Volume Method via Conservation Laws

If we assume advection with a constant transport velocity  $a$  this flux is given as a function of the values of  $q(x, t)$  as

$$F \rightarrow f(q(x, t)) = aq(x, t)$$

in other words

$$\partial_t \int_{x_1}^{x_2} q(x, t) dx = f(q(x_1, t)) - f(q(x_2, t))$$

This is called the integral form of a hyperbolic conservation law.

## The Finite-Volume Method via Conservation Laws

Using the definition of integration and antiderivates to obtain

$$\begin{aligned}\partial_t \int_{x_1}^{x_2} q(x, t) dx &= - \int_{x_1}^{x_2} \partial_x f(q(x, t)) dx \\ \int_{x_1}^{x_2} [\partial_t q(x, t) + \partial_x f(q(x, t))] dx &= 0\end{aligned}$$

which leads to the well-known **partial-differential equation of linear advection**

$$\partial_t q(x, t) + \partial_x f(q(x, t)) = 0$$

or

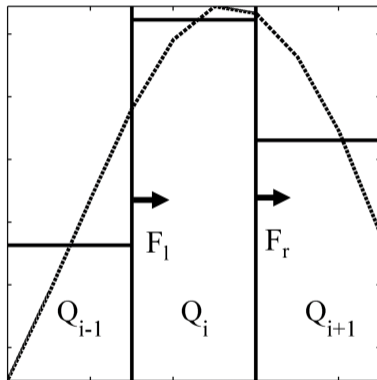
$$\partial_t q(x, t) + a \partial_x q(x, t) = 0$$

## Upwind scheme

Instead of working on the field  $q(x,t)$  itself we approximate the integral of  $q(x,t)$  over the cell  $\mathcal{C}$  by

$$Q_i^n \approx \frac{1}{dx} \int_{\mathcal{C}} q(x, t^n) dx$$

This is the average value of  $q(x, t)$  inside the cell.



## Upwind scheme

Using the following terms for the fluxes at the boundaries

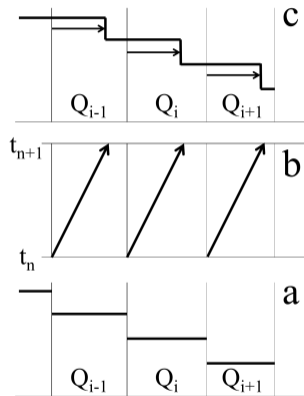
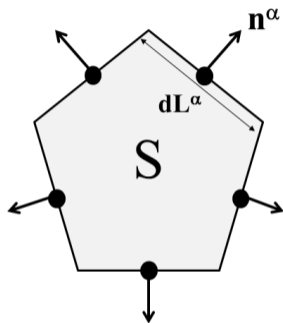
$$F_{L,R}^n = \int_t^{t_{n+1}} f(q(x_{L,R}, t)) dt$$

we obtain a time-discrete scheme for the average values of our solution field  $q(x,t)$

$$Q_i^{n+1} = Q_i^n - \frac{dt}{dx} (F_R^n - F_L^n)$$

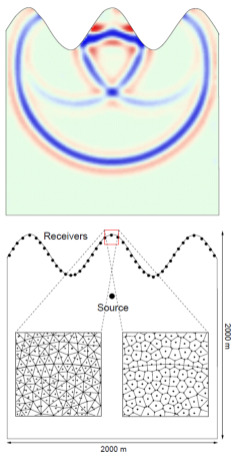
where the upper index  $n$  denotes time level  $t_n = n * dt$  and the lower index  $i$  denotes cell  $\mathcal{C}_i$  of size  $dx$ .

# Fluxes, Riemann problem



The finite-volume approach allows derivation of acoustic wave equation from first principles (i.e., mass conservation)

# Applications, Recent Developments



Kaesler et al., 2000

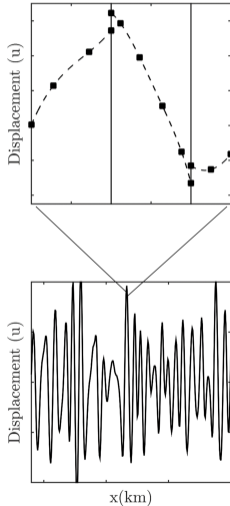
- Method of choice for conservation problems with strongly discontinuous solutions
- Many applications in geophysical fluid dynamics
- Relatively simple, finite-difference style algorithms
- Linear extrapolation schemes strongly diffusive
- Recent general extensions to higher order
- Potential for seismology not fully explored

# The Discontinuous Galerkin Method

---



# The Discontinuous Galerkin Method in a Nutshell



- Numerical solution of **first-order systems**
- Developed for **hyperbolic** problems (e.g., neutron diffusion)
- Local formulation for each element
- Solution of *weak form* of wave equation
- Communication between elements through **fluxes** → FV
- Explicit time extrapolation → efficient parallelisation
- Nodal and modal approaches for hexahedral and tetrahedral meshes

# Wave Equation

## 1st order wave equation

$$\rho \partial_t v = \partial_x \sigma + f$$

$$\partial_t \sigma = \mu \partial_x v$$

## Matrix-vector form

$$\partial_t Q + A \partial_x Q = f$$

## The Discontinuous Galerkin Method Put To Action

In matrix notation we yielded for one element

$$M\partial_t q(t) - K^T q(t) = -F(a, q(t))$$

requiring an extrapolation scheme of the form

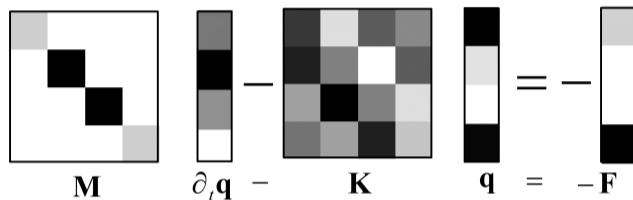
$$\partial_t q(t) = M^{-1}(K^T q(t) - F(a, q(t)))$$

where  $F(a, q(t))$  is the flux vector as defined above. We seek to extrapolate the system from some initial conditions and obtain using the simple Euler method for each element

$$q(t_{n+1}) \approx q(t_n) + dt \left[ M^{-1}(K^T q(t_n) - F(a, q(t))) \right]$$

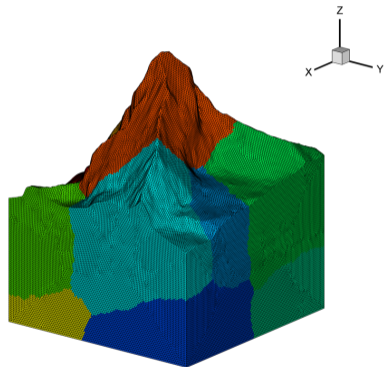
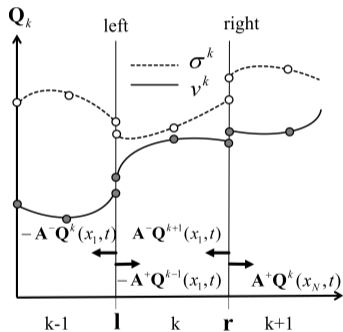
where for the flux scheme  $F()$  we use the upwind approach.

## The Discontinuous Galerkin Method Put To Action



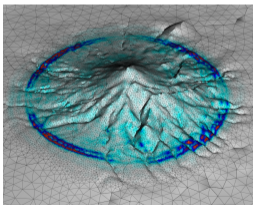
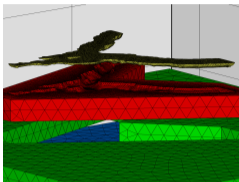
The matrix-vector form of the discontinuous Galerkin method. The system of questions at an elemental level is illustrated by plotting the absolute matrix/vector values.  $N = 3$ ,  $N_p = N + 1 = 4$

# Fluxes, tetrahedral meshes



The first competitive scheme for tetrahedral meshes in seismology, but ...

# Applications, Recent Developments



- Applications in exploration, volcanology, **shaking hazard**, **earthquake physics**
- Inefficient with **tetrahedral** meshes (for smooth problems)
- Method of choice for dynamic rupture simulations
- Extremely well scalable (Gordon Bell finalist 2015)
- New modal, octree approach developed in ExaHype project ([exahype.eu](http://exahype.eu))
- Community codes: *seissol* (Munich), *nex3d* (Bochum)

### III: Comprehensive Questions

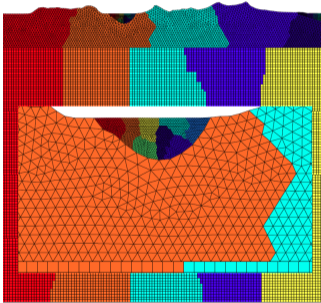
- What do you think is better to discretize complex geometries, tetraedral or hexahedral elements?
- What does the *discontinuous* mean in the DG method??
- What could be the reason why the DG method works so well for earthquake rupture simulations?

**"Meet the future ..." (From: Butch Cassidy and the Sundance Kid)**





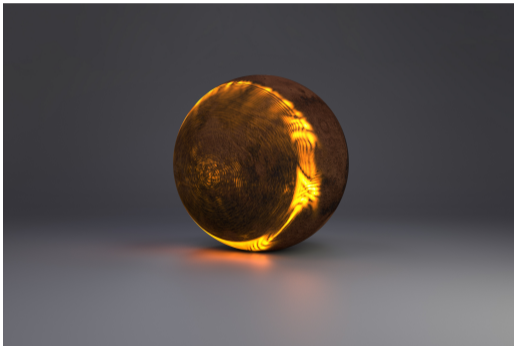
# Challenges - Meshing



Human time	Simulation workflow	cpu time
15%	Design	0%
80% (weeks)	Geometry creation, meshing	10%
5%	Solver	90%

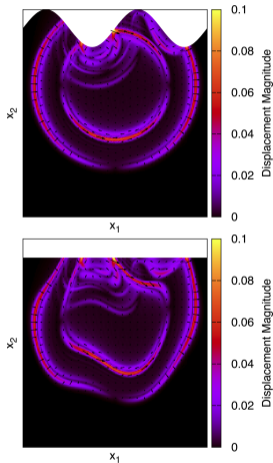
- Meshing work flow not well defined
- Still major bottleneck for simulation tasks with complex geometries
- Tetrahedral meshes easier, but ...
- Salvus?

## Spectral element method - *Salvus*



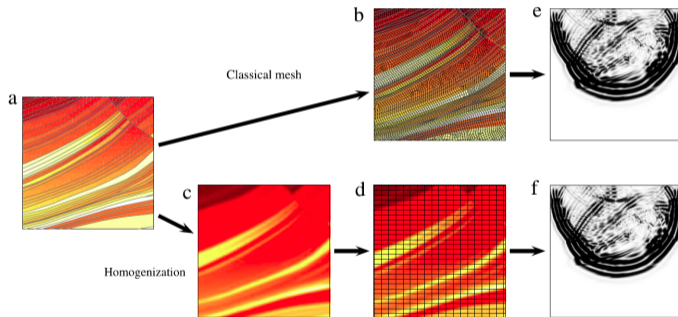
- Developed at ETH, now commercial (!) [mondaic.com](http://mondaic.com)
- Spectral-element implementation
- (tetrahedral and) **hexahedral** meshes
- Built on top of community libraries (e.g., PetSc)
- Meshing routines for some model classes
- Work flow with **Jupyter notebooks**

# Future Strategies - Alternative Formulations



- Particle relabelling, grid stretching
- Mapping geometrical complexity onto regular grids
- Smart pre-processing rather than meshing?
- Similar concept used in summation-by-parts (SBP) algorithms (SW4)

# Future Strategies - Homogenization



- We only see low-pass filtered Earth
- So why simulate models with infinite frequencies?
- Homogenisation of discontinuous model
- Renaissance of regular grid methods?

The screenshot shows the website for SPECIFEM 3D GLOBE. The header includes the logo for 'COMPUTATIONAL INFRASTRUCTURE for GEODYNAMICS' and a navigation menu with links for Home, About, Working Groups, Software, Developers, Projects, Community, News & Publications, and Forum. The main content area is titled 'SPECIFEM 3D GLOBE' and contains several paragraphs of text describing the software's capabilities, such as simulating global and regional seismic wave propagation and offering GPU graphics card support. A sidebar on the left lists 'Software Package List' with links for 'SPECIFEM 3D GLOBE', 'Current Release', 'User Resources', 'Developer Resources', and 'User Map'. A right sidebar features a globe image, a 'Status' section indicating active development, a 'How to Cite' section, and a 'Code changes' section showing 0 commits in the past month and 0 in the past year.

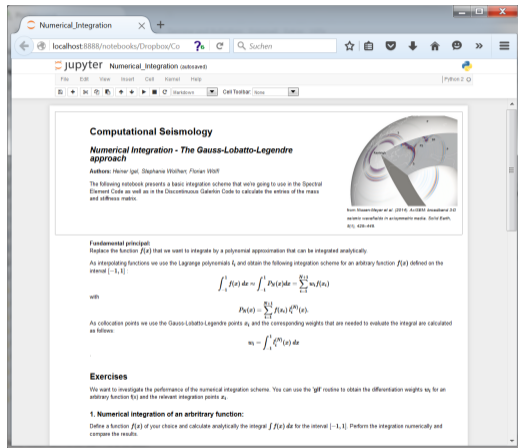
www.geodynamics.org

We would need ...

- Science gateways for basic simulation tasks
- High level model initialization
- Large scale simulations - hidden supercomputers
- Complex admission protocols
- Black boxes
- Great idea, but ...

# Computational Seismology, Practical Exercises, Jupyter Notebooks

- *Jupyter notebooks* are interactive documents that work in any browser
- Simple text editing
- Inclusion of graphics
- Equations with Latex
- Executable code cells with Python (or else)
- The coolest thing since ...
- Many examples on: [www.seismo-live.org](http://www.seismo-live.org)
- *Computational Seismology: A Practical Introduction* (Oxford University Press)

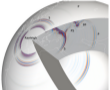


**Computational Seismology**

**Numerical Integration - The Gauss-Lobatto-Legendre approach**

Authors: Homer Igel, Stephanie Walther, Florian Woff

The following notebook presents a basic integration scheme that we're going to use in the Spectral Element Code as well as in the Discontinuous Galerkin Code to calculate the entries of the mass and stiffness matrix.



from Steven Blitzer et al. (2016), *AccESM: Anisotropic 2D seismic tomography in anisotropic media*. *Earth Earth*, 6(1), 428-448.

**Fundamental principle:**  
Replace the function  $f(x)$  that we want to integrate by a polynomial approximation that can be integrated analytically:  
As interpolating functions we use the Lagrange polynomials  $l_i$  and obtain the following integration scheme for an arbitrary function  $f(x)$  defined on the interval  $[-1, 1]$ :

$$\int_{-1}^1 f(x) dx \approx \int_{-1}^1 P_N(x) dx = \sum_{i=1}^{N/2+1} w_i f(x_i)$$

with

$$P_N(x) = \sum_{i=1}^{N/2+1} f(x_i) l_i^{(N)}(x).$$

As collocation points we use the Gauss-Lobatto-Legendre points  $x_i$  and the corresponding weights that are needed to evaluate the integral are calculated as follows:

$$w_i = \int_{-1}^1 l_i^{(N)}(x) dx$$

**Exercises**

We want to investigate the performance of the numerical integration scheme. You can use the 'ipf' routine to obtain the differentiation weights  $w_i$  for an arbitrary function  $f(x)$  and the relevant integration points  $x_i$ .

**1. Numerical integration of an arbitrary function:**  
Define a function  $f(x)$  of your choice and calculate analytically the integral  $\int f(x) dx$  for the interval  $[-1, 1]$ . Perform the integration numerically and compare the results.

Try it out!


# 9-week course in Computational wave propagation on COURSERA (free!)

Browse > Physical Science and Engineering > Research Methods

Offered By

**Computers, Waves, Simulations:  
A Practical Introduction to  
Numerical Methods using  
Python**



★★★★★ 4.7 263 ratings

 Heiner Igel

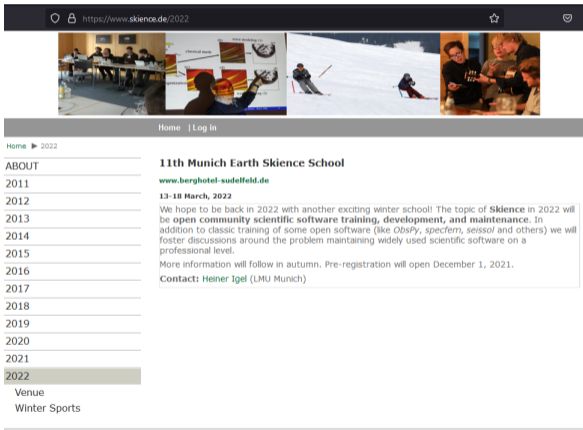
[Go To Course](#)

Already enrolled  
Financial aid available

14,521 already enrolled



Covers the  
finite-difference,  
pseudospectral,  
finite-element, and  
spectral-element  
method.



The screenshot shows the website <https://www.skience.de/2022>. At the top, there are four images: a group of people in a meeting, a person pointing at a presentation slide, two people skiing on a snowy slope, and two people looking at a laptop. Below the images is a navigation bar with "Home" and "Log in". A sidebar on the left contains a list of years from 2011 to 2022, with 2022 highlighted. The main content area features the title "11th Munich Earth Science School" and the URL [www.berghotel-sudelfeld.de](http://www.berghotel-sudelfeld.de). The text below the title states: "13-18 March, 2022. We hope to be back in 2022 with another exciting winter school! The topic of Skience in 2022 will be **open community scientific software training, development, and maintenance**. In addition to classic training of some open software (like *ObsPy*, *specfem*, *seissol* and others) we will foster discussions around the problem maintaining widely used scientific software on a professional level. More information will follow in autumn. Pre-registration will open December 1, 2021. **Contact: Heiner Igel** (LMU Munich)".

Software training in  
seismology: ObsPy,  
seissol, salvus,  
specfem, etc ...



# Conclusions

The forward problem is solved, but ...

