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

EnvSeis Short Course, May 27, 2024

Heiner Igel

May 23, 2024

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

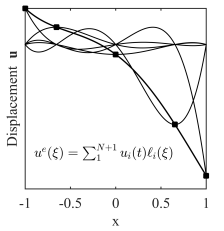
Numerical Methods for Wave Propagation Problems

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 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** $\rightarrow$ trivially inverted
- Explicit extrapolation scheme $\rightarrow$ efficient parallelisation
- Method of choice for global wave propagation (specfem, axisem)
- Meshing required

1-D 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

f external force

ρ mass density

μ shear modulus

Weak Formulation

Multiplication of pde with test function $w(x)$ on both sides.
G is here the complete computational domain defined with $x \in G = [0, L]$.

$$\int_G w\, \rho\, \ddot{u}\, \mathrm{d}x \; - \int_G w\, \partial_x\, (\mu\, \partial_x u)\, \mathrm{d}x \; = \; \int_G w\, f\, \mathrm{d}x$$

Integration by parts

$$\int_G w\,\rho\,\ddot{u}\,\mathrm{d}x + \int_G \mu\,\partial_x w\,\partial_x u\,\mathrm{d}x = \int_G w\,f\,\mathrm{d}x$$

where we made use of the boundary condition

$$\partial_x u(x,t)|_{x=0} = \partial_x u(x,t)|_{x=L} = 0$$

The approximate displacement field

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

- Discretization of space introduced with this step
- Specific basis function not yet specified
- In principle basis functions defined on entire domain (-> PS method)
- Locality of basis functions lead to finite-element type method

Including the function approximation for $u(x,t)$

... leads to an equation for the unknown coefficients $u_i(t)$

$$\sum_{i=1}^{n} \left[\ddot{u}_i(t) \int_{G_e} \rho(x)\, \psi_j(x)\, \psi_i(x)\, \mathrm{d}x \right]$$
$$+ \sum_{i=1}^{n} \left[u_i(t) \int_{G_e} \mu(x)\, \partial_x \psi_j(x)\, \partial_x \psi_i(x)\, \mathrm{d}x \right]$$
$$= \int_{G} \psi_i\, f(x,t)\, \mathrm{d}x$$

for all basis functions ψ_j with $j = 1, ..., n$. This is the classical algebro-differential equation for **finite-element** type problems.

Lagrange polynomials

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

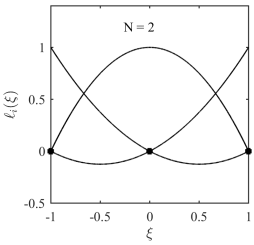
$$u(x,t) \;\approx\; \overline{u}(x,t) \;=\; \sum_{i=1}^{n} u_i(t)\,\psi_i(x)$$

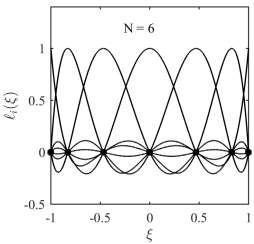
Our final choice: Lagrange polynomials:

$$\psi_i \;\to\; \ell_i^{(N)} \;:=\; \prod_{k=1,\;k\neq i}^{N+1} \frac{\xi-\xi_k}{\xi_i-\xi_k}, \qquad i=1,2,\ldots,N+1$$

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

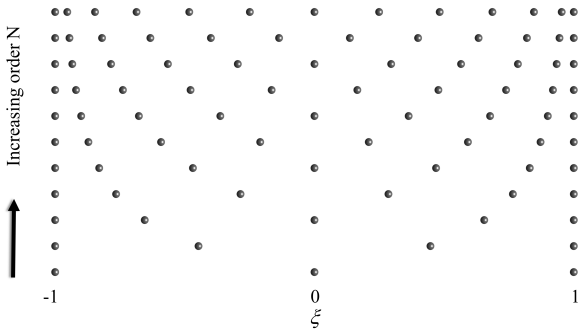


Illustration of the spatial distribution of Gauss-Lobatto-Legendre points in the interval [-1,1] from bottom to top for polynomial order 1 to 12. Note the increasing difference of largest to smallest interval between collocation points! Consequences?

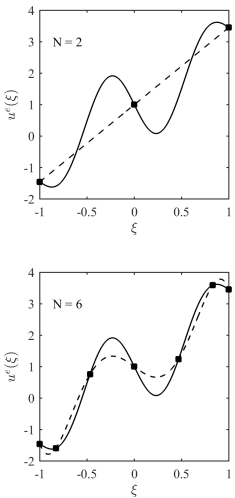
Function approximation

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.

Exercise - Jupyter Notebooks

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

- **se_Lagrange_interpolation**
- Change the sin to a function of your choice (e.g., sum over multiple sin function of different frequencies)
- Change the order N of the Lagrange polynomials and observe the convergence

Note down **questions** during the process!

Matrix notation

This system of equations with the coefficients of the basis functions (meaning?!) as unknowns can be written in matrix notation

$$\mathbf{M} \cdot \ddot{\mathbf{u}}(t) + \mathbf{K} \cdot \mathbf{u}(t) = \mathbf{f}(t)$$

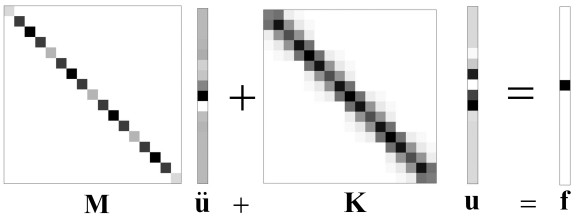
M Mass matrix

K Stiffness matrix

terminology originates from structural engineering problems

Matrix system graphically

The figure gives an actual graphical representation of the matrices for our 1D problem.



The unknown vector of coefficients **u** is found by a simple finite-difference procedure. The solution requires the inversion of mass matrix **M** which is trivial as it is diagonal. That is the key feature of the spectral element method.

Mapping

A simple centred finite-difference approximation of the 2nd derivative and the following mapping

$$\begin{aligned} \mathbf{u}^{new} &\rightarrow \mathbf{u}(t + \mathrm{d}t) \\ \mathbf{u} &\rightarrow \mathbf{u}(t) \\ \mathbf{u}^{old} &\rightarrow \mathbf{u}(t - \mathrm{d}t) \end{aligned}$$

leads us to the solution for the coefficient vector $\mathbf{u}(t + \mathrm{d}t)$ for the next time step as already well known from the other solution schemes in previous schemes.

Solution scheme

$$\mathbf{u}^{new} = \mathrm{d}t^2 \left[\mathbf{M}^{-1} \left(\mathbf{f} - \mathbf{K}\ \mathbf{u} \right) \right] + 2\mathbf{u} - \mathbf{u}^{old}$$

General solution scheme for finite-element method (wave propagation)

Independent of choice of basis functions

Mass matrix needs to be inverted (trivial if diagonal)!

Mass and stiffness matrix

Definition of the - at this point - global mass matrix

$$M_{ji} = \int_G \rho(x)\, \psi_j(x)\, \psi_i(x)\, \mathrm{d}x$$

and the stiffness matrix

$$K_{ji} = \int_G \mu(x)\, \partial_x \psi_j(x)\, \partial_x \psi_i(x)\, \mathrm{d}x$$

and the vector containing the volumetric forces $f(x,t)$

$$f_j(t) = \int_G \psi_i\, f(x,t)\, \mathrm{d}x \quad .$$

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

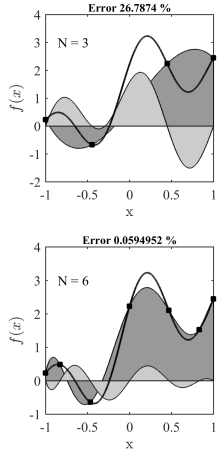
$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \int_{-1}^{1} P_N(x)\mathrm{d}x = \sum_{i=1}^{N+1} w_i f(x_i)$$

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

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

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

Numerical integration scheme



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

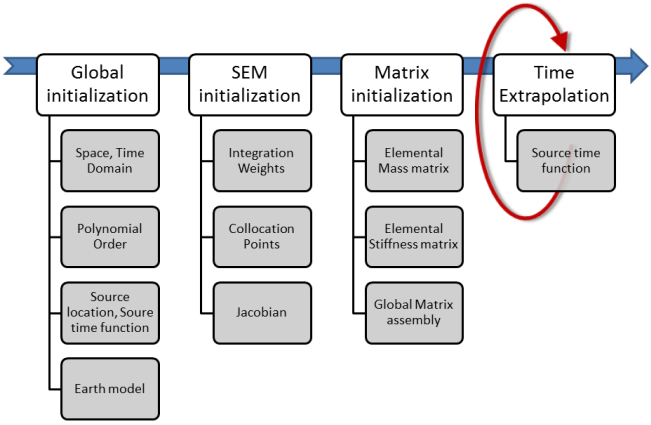
Extrapolation for time-dependent coefficients $\mathbf{u}_g$

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

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

Looks fairly simple, no?

Spectral elements: work flow

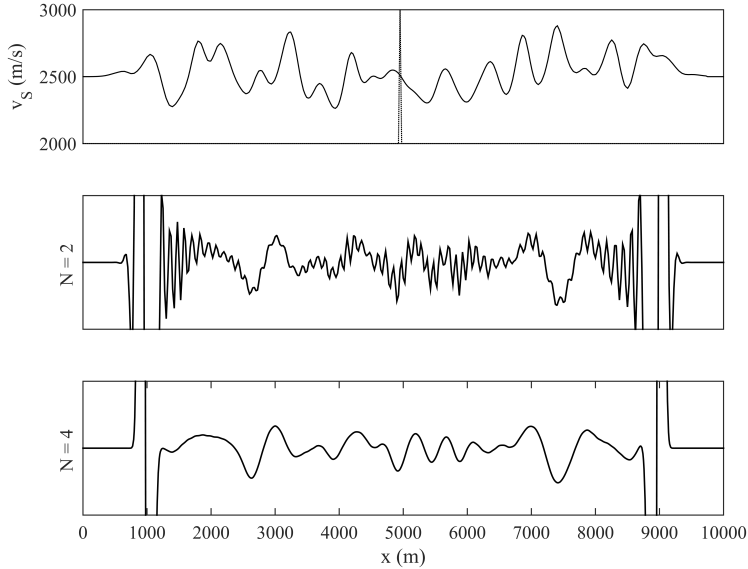


A substantial part consists of preparing the interpolation and integration procedures required to initialize the global mass- and stiffness matrices. The final time-extrapolation is extremely compact and does not require the inversion of a global matrix as is the case in classic finite-element methods.

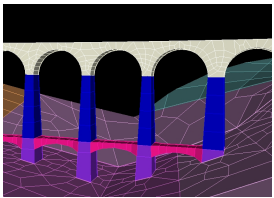
Python Code:Time Extrapolation

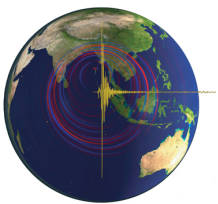
```
# -----------------------------------------------------------
# Time extrapolation
# -----------------------------------------------------------
x_t = []
for it in range(nt):
    # Source initialization
    f = np.zeros(ng)
    if it < len(src):
        f[isrc-1] = src[it-1]

    # 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 ($\rightarrow$ salvus)
- ***specfem*** or ***salvus*** are widely used community software for seismology (3D)

Comprehension 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*?

Exercise - Jupyter Notebooks

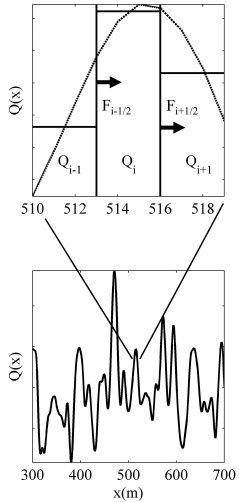
Play around with the following Jupyter notebooks (spectral element folder, github, seismo-live, or COURSERA)

- **se_homo_1d_solution**: change ne to 50, observe the dispersion, increase polynomial order to get better solution
- **se_hetero_1d_solution**: change ne to 50, observe the dispersion, increase polynomial order to get better solution

Note down **questions** during the process!

The Finite-Volume Method

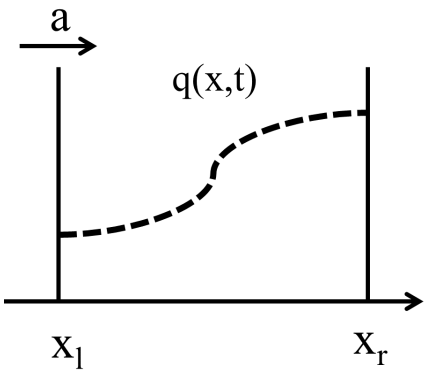
The Finite-Volume Method in a Nutshell



- 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 ourselve into a finite volume cell that we denote as $\mathscr{C}$ 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

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

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

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

$$\int_{x_1}^{x_2} \left[\partial_t q(x,t) + \partial_x f(q(x,t))\right] dx = 0$$

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

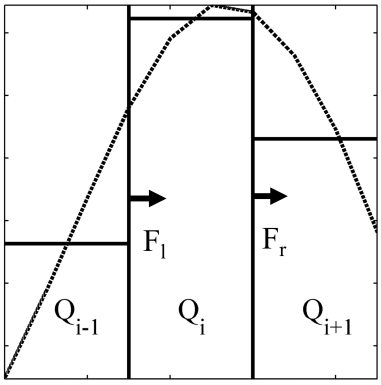
$$\partial_t q(x,t) + a\partial_x q(x,t) = 0$$

Upwind scheme

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

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

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



Upwind scheme

Using the following terms for the fluxes at the boundaries

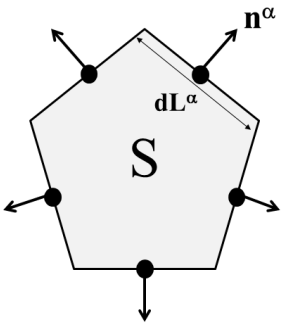
$$F^n_{L,R} = \int_t^{t_{n+1}} f(q(x_{L,R}, t))dt$$

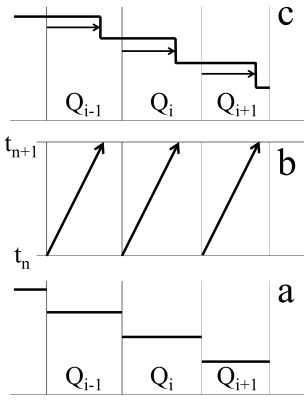
we obtain a time-discrete scheme for the average values of our solution field q(x,t)

$$Q^{n+1}_i = Q^n_i - \frac{dt}{dx}(F^n_R - F^n_L)$$

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

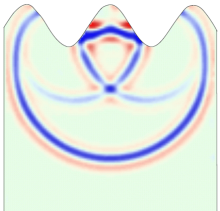
Fluxes, Riemann problem





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

Applications, Recent Developments

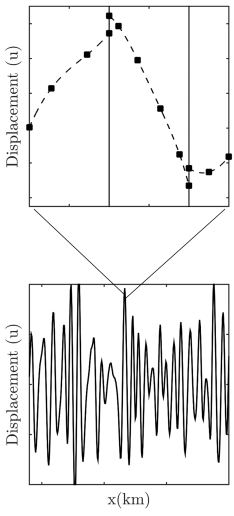


Kaeser et al., 2000

- 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** $\rightarrow$ FV
- Explicit time extrapolation $\rightarrow$ efficient parallelisation
- Nodal and modal approaches for hexahedral and tetrahedral meshes

Wave Equation

1st order wave equation

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

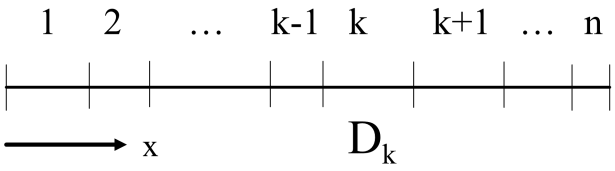
Matrix-vector form

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

Solution to Scalar Advection Equation

Denoting $q(x,t)$ for the unknown *scalar* solution and a for the given (possibly space dependent) advection (wave) velocity to obtain the source-free wave equation

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



The physical domain of each element is denoted by D_k with left and right boundaries x_l^k and x_r^k, respectively. Note the varying element sizes (h-adaptivity).

Solution to Scalar Advection Equation

The k elements are **non-overlapping** which implies a higher number of degrees of freedom. The solution vector in 1D is

$$q_{N_p} = \begin{pmatrix} q_1^1 & q_1^2 & \dots & q_1^n \\ q_2^1 & q_2^2 & \dots & q_2^n \\ \vdots & \vdots & \ddots & \vdots \\ q_{N_p}^1 & q_{N_p}^2 & \dots & q_{N_p}^n \end{pmatrix}$$

where N_p is the number of points per element and n the overall number of elements. The collocation points are Gauss-Lobatto-Legendre stored as

$$x_{N_p} = \begin{pmatrix} x_1^1 & x_1^2 = x_{N_p}^1 & \dots & x_1^n \\ x_2^1 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_{N_p}^1 = x_1^2 & x_{N_p}^2 & \dots & x_{N_p}^n \end{pmatrix}.$$

Weak formulation

To obtain the weak formulation of the scalar advection equation we multiply it

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

with a general test function $\phi_j(x)$, integrate over the k-th element domain D_k to obtain

$$\int_{D_k} \partial_t q(x,t)\phi_j(x)dx + \int_{D_k} a\partial_x q(x,t)\phi_j(x)dx = 0$$

Weak formulation

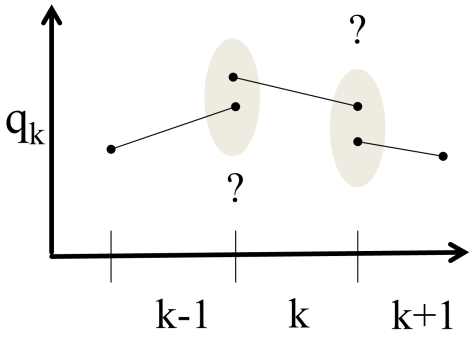
... we obtain for element k

$$\int_{D_k} \partial_t q(x,t)\phi_j(x)dx - \int_{D_k} aq(x,t)\partial_x \phi_j(x)dx = -\left[aq(x,t)\phi_j(x)\right]_{x_l}^{x_r}$$

Replacing the unknown field $q(x,t)$ by a finite polynomial representation in terms of a weighted sum over Lagrange polynomials inside each element k of order N denoted as $\ell_i(x), i = 1,\ldots,N+1$ defined in the interval $x \in [-1,1]$.

The Flux Scheme

The key task in the discontinuous Galerkin scheme concerns the question what values to allocate to the points at the element boundaries. This involves the use of flux schemes.



The Flux Scheme

Starting with the r.h.s. of this equation

$$\int_{D_k} \partial_t q(x,t)\phi_j(x)dx - \int_{D_k} aq(x,t)\partial_x\phi_j(x)dx = -\left[aq(x,t)\phi_j(x)\right]_{x_l}^{x_r}$$

The originial form holds also for higher dimensions

$$\int_{\partial D_k} a\, q(x,t)\phi_j(x)\ \mathbf{n}\ dx$$

where $\mathbf{n} = \pm 1$ denotes the vector normal to the boundary, in 1D taking the values $n = -1$ and $n = 1$ at the left and right boundaries.

The Flux Scheme

Therefore, we introduce the general flux vector with elements $F_j(a, q^k(x,t)), j = 1, .., N_p$ as

$$F_j(a, q^k(x,t)) \;=\; [\ell_j(x)\ (a\ q(x,t))^*]_{x_l^k}^{x_r^k}$$

In vector form

$$\mathbf{F} = \begin{pmatrix} F_1 \\ 0 \\ \vdots \\ 0 \\ F_{N_p} \end{pmatrix} = \begin{pmatrix} -(a\ q(x,t))^*(x_l^k) \\ 0 \\ \vdots \\ 0 \\ (a\ q(x,t))^*(x_r^k) \end{pmatrix}.$$

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.

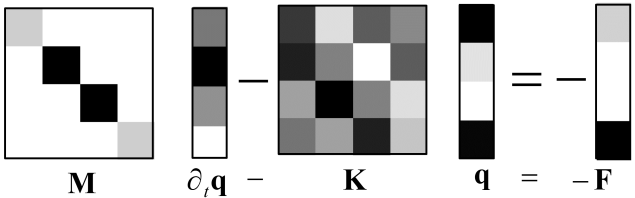
The Discontinuous Galerkin Method Put To Action

The element-wise system of equations can be extrapolated by the Euler scheme as

```
% Time extrapolation
for it = 1:nt,
(...)
% Initialize flux vectors F (...)
for k=1:ne,
q(:,k) = q(:,k) + dt .* (Minv(:,k) * (K(:,k)'*q(:,k)-F(:,k)));
end (...)
end
```

where *nt* is the overall number of time steps, *dt* is the global time increment, and *ne* is the number of elements. Note that can directly update the solution vector *q* without intermediate storage at different time level(s).

The Discontinuous Galerkin Method Put To Action



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

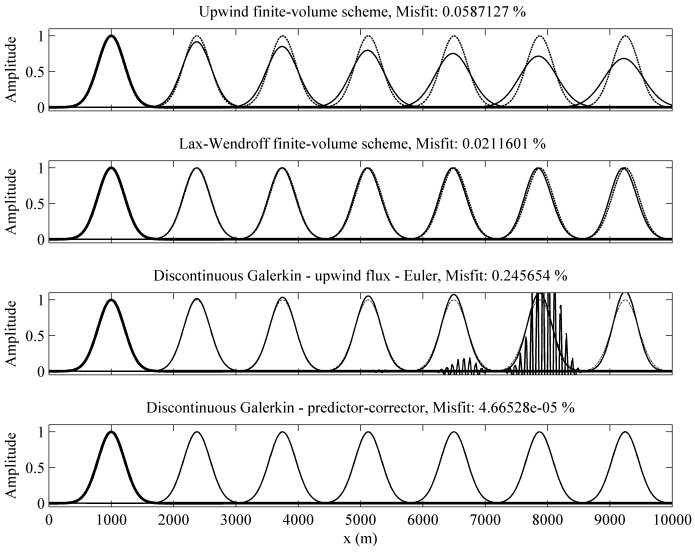
Example

Table 1: Simulation parameters for 1D discontinuous Galerkin advection

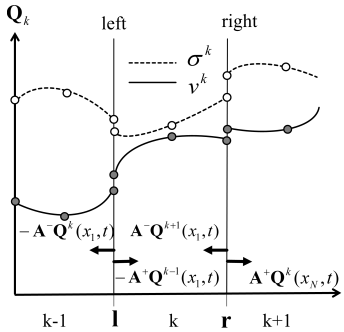
Parameter	Value	Meaning
ne	200	Elements
N	3	Order
a	2500 m/s	Velocity
x_{max}	10000 m	x-domain
dx_{min}	13.82 m	Increment
dt	4.4e-4 s	Time step
eps	0.08	Courant
σ	300 m	Gauss width
x_0	1000 m	Source x
t_{max}	3 s	Duration

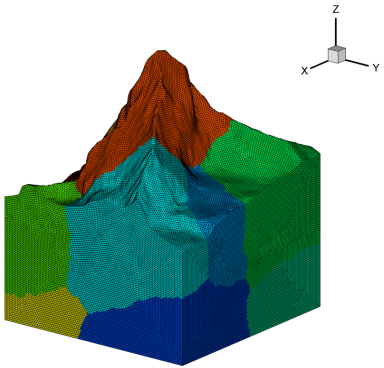
Initiating the simulation with a spatial initial condition using a Gaussian function $e^{-1/\sigma^2(x-x_0)^2}$. A source term can be added to the system of equations in a straight forward way.

Example


Upwind finite-volume scheme, Misfit: 0.0587127 %
Lax-Wendroff finite-volume scheme, Misfit: 0.0211601 %
Discontinuous Galerkin - upwind flux - Euler, Misfit: 0.245654 %
Discontinuous Galerkin - predictor-corrector, Misfit: 4.66528e-05 %
Amplitude
1
0.5
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
x (m)

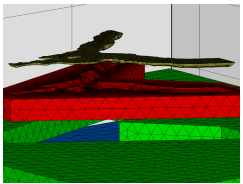
Fluxes, tetrahedral meshes

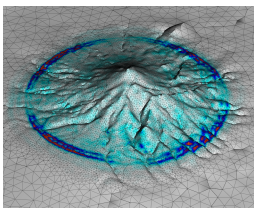




The first competitive scheme for tetrahedral meshes in seismology, but ...

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)
- Community codes: *seissol* (Munich), *nex3d* (Bochum)

Summary

- The discontinuous Galerkin method is a finite-element type method. The main difference is that the solution fields are **not continuous** at the element boundaries.
- The elemental mass and stiffness matrices are formulated very similar to the classic finite-element schemes. However, they are never assembled to a global system of equations. Therefore **no large system matrix needs to be inverted**.
- Elements are linked by a **flux scheme**, similar to the finite volume method. This scheme leads to an entirely **local algorithm** in the sense that all calculations are carried out at an elemental level. Communication happens only to direct neighbours.
- The discontinuous Galerkin scheme can easily be extended to **higher orders**, keeping the local nature of the solution scheme. This leads to high efficiency when parallelizing the scheme.

Summary

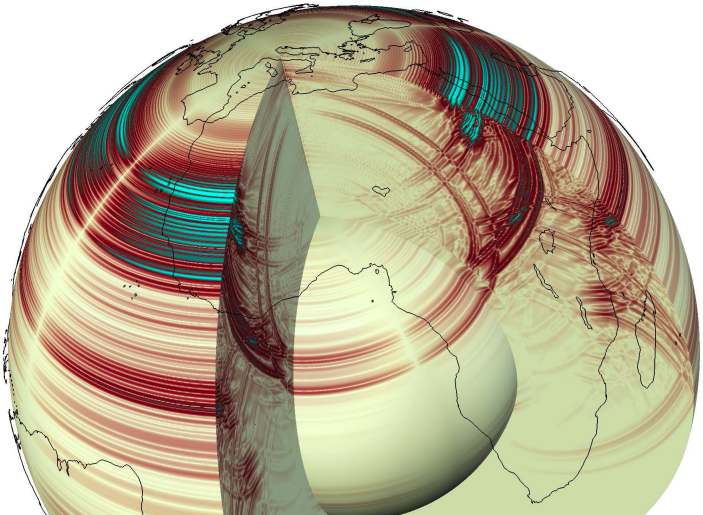
- The solution fields can be extended using **nodal** and **modal** approaches. Modal approaches are preferred for tetrahedral schemes. Nodal solutions are preferred for hexahedral meshes.
- The discontinuous behaviour at the element boundaries and the associated discretization of the element boundaries **increases the number of degrees of freedom** compared to other methods.
- The flexibility with polynomial order, element size, local time stepping leads to a formidable problem when parallelizing a discontinuous Galerkin method: **load balancing** the computational task is not easy! Solution to this problem requires tight cooperation with computational scientists.
- In seismology, the discontinuous Galerkin method is useful for problem with **highly complex geometries** (by using tetrahedral meshes) and for problem with non-linear internal boundary conditions (e.g., **dynamic ruputure** problems).

Comprehension 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?

Conclusions

The forward problem is solved, but ...



Questions?