Zlotnik’s 3 Level Finite Elements for the Wave Equation Implemented in Fenics/Python

Sebastian Engel

Introduction - Problem

Consider the following wave equation:

\[
\begin{aligned}
D_{tt}y - \Delta y &= f & \text{in} & \quad (0, T) \times \Omega \\
y &= 0 & \text{on} & \quad (0, T) \times \partial \Omega \\
(y, \partial_t y) &= (y_0, y_1) & \text{in} & \quad \{0\} \times \Omega
\end{aligned}
\]

\[\tag{1}\]

- \(\Omega \subset \mathbb{R}^n \) (n=1,2,3) open bounded, \(\partial \Omega\) smooth enough, \(T \in (0, \infty) = I\)
- \((f, y_0, y_1) \in L^1(0, T; L^2(\Omega)) \times H^1_0(\Omega) \times L^2(\Omega)\)
**Introduction - Weak Solution**

**Definition (Weak Solution of the Wave)**

We call \( y \in C([0, T]; H^1_0(\Omega)) \) a weak solution of the wave equation (1) if

i) \( D_t y \in C([0, T]; L^2(\Omega)) \)

ii) \( y \big|_{t=0} = y_0 \) and

iii) satisfies the integral identity

\[
- (D_t y, D_t \eta)_Q + (\nabla y, \nabla \eta)_Q = (y_1, \eta_0)_\Omega + (f, \eta)_Q
\]  

(2)

for all \( \eta \in L^1(0, T; H^1_0(\Omega)) \) such that \( D_t \eta \in L^1(0, T; L^2(\Omega)) \), \( \eta \big|_{t=T} = 0 \) and \( \eta_0 = \eta \big|_{t=0} \).

Note: \((a, b)_Q := \int_0^T \int_\Omega a(t, x)b(t, x)dxdt \) and \((u, v)_\Omega := \int_\Omega u(x)v(x)dx\).
Wave Equation Numerics

- Let $\Omega \subset \mathbb{R}^n$ be an triangulable set with a "good" mesh.
- Let $S_h \subset H^1_0(\Omega)$ be a finite dimensional space of finite Elements constructed by some triangulation with mesh size $h$.
- We uniform discretized $I = [0, T]$ w.r.t. mesh size $\tau$, i.e. $0 = t_0 < ... < t_M = T$ and $\tau = t_i - t_{i-1}$ for $i = 1, ..., M$.
- We use the space $\hat{S}_\tau$ of piecewise linear and continuous functions w.r.t. uniform time mesh to discretise $H^1(0, T)$.
- Let us denote the following time mesh operators

$$\ddot{w}_m = w_{m-1} \quad \hat{w}_m = w_{m+1}$$

$$\bar{\partial}_t w = \frac{w - \ddot{w}}{\tau} \quad \partial_t w = \frac{\hat{w} - w}{\tau}$$
3 Level Finite Element - Zlotnik

As an approximate solution to the initial-boundary value problem (1) with respect to the weak solution we consider a function \( v \in S = S_h \otimes \hat{S}_r \) satisfying the integral identity

i) 

\[
-(D_t v, D_t \eta)_Q - \left( \sigma - \frac{1}{6} \right) (\nabla (\tau^2 D_t v), \nabla (D_t \eta))_Q + (\nabla v, \nabla \eta)_Q = (y_1, \eta_0)_\Omega + (f, \eta)_Q
\]

for all \( \eta \in S \) with \( \eta \big|_{t=T=0} = 0 \).

ii) with \( v \big|_{t=0} = v^{(0)} \in S_h \) satisfying the equation (\( \approx \) projection, \( \sigma = 0 \))

\[
\left( v^{(0)}, \varphi \right)_\Omega + \sigma \tau^2 \left( \nabla v^{(0)}, \nabla \varphi \right)_\Omega = (y_0, \varphi)_\Omega
\]

for all \( \varphi \in S_h \).
3 Level Finite Element - Equivalent Form

Consider \( \eta(x, t) = \varphi(x)e_m(t) \), then the scheme in (3) is equivalent to

\[
(\partial_t \partial_t v_m, \varphi)_\Omega + (\nabla(\sigma \tau^2 \partial_t \partial_t v_m + v_m), \nabla \varphi)_\Omega = (f^\tau_m, \varphi)_\Omega
\]

(5)

for \( 1 \leq m \leq M - 1 \) and

\[
(\partial_t v_0, \varphi)_\Omega + \left( \nabla \left( \sigma \tau^2 \partial_t v_0 + \frac{T}{2} v_0 \right), \nabla \varphi \right)_\Omega = (y_1, \varphi)_\Omega + \frac{T}{2} (f^\tau_0, \varphi)_\Omega
\]

(6)

for all \( \varphi \in S_h \), time interpolant \( f \), i.e. \( f^\tau(t, x) = \sum_{i=0}^{M} f(t_i, x)e_i(t) \) and

\( f^\tau_m = f^\tau(t_m, x) \).

Remark

This scheme can be used for Fenics!
frame
3 Level Finite Element - Equivalent Form

The equations (5) and (6) can be rewritten into:

\[
(v_{m+1}, \varphi)_{\Omega} + \sigma \tau^2 (\nabla (v_{m+1}), \nabla \varphi)_{\Omega} = (2v_m - v_{m-1}, \varphi)_{\Omega} + \tau^2 (\nabla ([\sigma (2v_m - v_{m-1}) - v_m], \nabla \varphi)_{\Omega} + \tau^2 (f_m^\tau, \varphi)_{\Omega}
\]

(7)

for \(1 \leq m \leq M - 1\) and

\[
(v_1, \varphi)_{\Omega} + \sigma \tau^2 (\nabla v_1, \nabla \varphi)_{\Omega} = (v_0, \varphi)_{\Omega} + (\sigma - 0.5) \tau^2 (\nabla v_0, \nabla \varphi)_{\Omega} + \tau (y_1, \varphi)_{\Omega} + \frac{\tau^2}{2} (f_0^\tau, \varphi)_{\Omega}
\]

(8)

Remark

This scheme can be used for Fenics!
**Fenics**

**FEniCS** = software project with aim to develop an automation for the solution of various problems in the field of mathematical modelling.

- Ability to express equations in a notation similar to written mathematics
- A compiler translates the notation into a valid programming language
- DOLFIN is a C++/Python library used for solving differential equations with the finite element method (part of FEniCS)
Fenics Implementation - Zlotnik Wave Equation

In the following we only show the implementation of (4):

```python
from dolfin import *
y0=Expression('0*x[0]*x[1]')
y1=Expression('0*x[0]*x[1]')
ydirichlet=Constant(0.0)
f=Expression('sin(2t)*
exp(-x[0]*x[0]-x[1]*x[1])',t=0)

mesh = RectangleMesh(Point(-10,10),Point(10, -10),
space_refine,space_refine)
```

- Represent the initial data for the wave equation, i.e.

\[
f(t, x, y) = \sin(2t)e^{-(x^2+y^2)}
\]

for \(t=0\) (can update: \(f.t=t0 \in \mathbb{R}\))

\[y(0) = 0, \quad \partial_t y(0) = 0, \quad y \mid_{\partial \Omega} = 0\]

- Constructing a rectangle (\(= \Omega\)) mesh with "space_refine \times space_refine" squares (each square 2 triangles)

\[x[0] \text{ and } x[1] \text{ are the x and y axis in the Cartesian coordinates.}\]
Fenics Implementation - Zlotnik Wave Equation

\[ V = \text{FunctionSpace}(\text{mesh}, "\text{Lagrange}", 1) \]

\[
\text{def boundary}(x, \text{on\_boundary}):
    \text{return on\_boundary}
\]

\[ bc = \text{DirichletBC}(V, \text{ydirichlet}, \text{boundary}) \]

- "FunctionSpace" produces a finite element family (mesh depended, "Lagrange" = continuous elements, 1 = linear).
- "boundary" defines us the mesh nodes for the boundary via on\_boundary.
- use for example return on\_boundary and \[x[0]<0\] to have different boundaries.
Fenics Implementation - Zlotnik Wave Equation

```
# Fenics Implementation - Zlotnik Wave Equation

u_Movie=Function(V, name="Solution_wave")

# First Zlotnik Step
y0_h=project(y0,V)
u= TrialFunction(V)
v= TestFunction(V)

L_0=y0_h*v*dx
a_0=u*v*dx+sigma*dt*dt*inner(nabla_grad(u),nabla_grad(v))*dx

A=assemble(a_0)
b=assemble(L_0)
```

- "Function" defines a representative in the FE space "V"
- "project" function to "V" (or use "interpolate")
- Recall (4):
  \[
  (v^{(0)}, \varphi)_\Omega + \sigma \tau^2 (\nabla v^{(0)}, \nabla \varphi)_\Omega = (y_0, \varphi)_\Omega
  \]
Fenics Implementation - Zlotnik Wave Equation

- $u_0$ will be used for the 3 level time step (later $u_1$ and $u_0$ gives new $u$)
- "bc.apply(A,b)" is applying the influence of "bc" boundary function to "A" and "b".
- Alternatively, e.g. 
  "solve(a_0==L_0, u,bc,"gmres"), with pre conditioner options.

```python
u=Function(V)
u_0=Function(V)
bc.apply(A,b)
solve(A, u.vector(), b)
u_0.assign(u)
u_Movie.assign(u)
```
Fenics - Start Parallelization

- Using the terminal in Linux, one is able to start the FEniCS code with the command:

  python Zlotnik.py

- If one want to parallelize the code (e.g. for 8 cores), write for example:

  mpirun -np 8 python Zlotnik.py

Each core will solve its assign region in the mesh and the rank 0 core will finally put all together.
Fenics - Partitioning of a Mesh

Fenics uses **METIS** to partition the mesh one uses for computations. METIS is a set of serial programs for partitioning graphs:

http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

"The algorithms implemented in METIS are based on the multilevel recursive-bisection, multilevel k-way, and multi-constraint partitioning schemes developed in our lab."
The **multi level graph partitioning algorithm** used in METIS can be prescribed in the following 3 main steps:

1) Suppose the **mesh** is an undirected **graph** $G = \{V, E\}$ with nodes as **vertices** ($V$) and **edges** ($E$) as the lines which are connecting the nodes in the mesh (e.g. triangle sides). Then start to **coarse the graph** into a smaller graph (aggregating vertices).

2) **Partition** the **coarser graph** and **take** the partition **back** to the original graph (**uncoarsening**).

3) **Refine** the partition **locally** in each uncoarsening step (degrees of freedom grows) to get better results for the original graph.
Fenics - Multi Level Graph Algorithm

The graph has for each $v \in V$ the weight $\eta(v)$ and each $e \in E$ the weight $\theta(e)$ with $e = \{v, u\}$ as the edge between $v, u \in V$.

1) **Aggregating vertices** - Example maximal matching algorithm from Hendrickson and Leland [1]:

   i) **Randomly** taking a vertex which is not already an aggregation of vertices and has at least one neighbour vertex which is not already an aggregation of vertices.

   ii) **Concatenate** the vertex in i) with an uncombined neighbour vertex if the edge weight between them is the smallest w.r.t. all possible uncombined neighbour vertices. If the smallest weight is not unique choose randomly an uncombined neighbour vertex.

   iii) The concatenated vertex has now the weight of adding the vertex weights of vertex in i) and the chosen neighbour vertex weight in ii). Similarly we **sum up** the weights corresponding the those edges which after the aggregation above convert to an edge.

   iv) If i) is unachievable **define all vertices as uncombined**.
Fenics - Multi Level Graph Algorithm

Stopping criteria: E.g. next coarsed graph has less than \( k \) vertices.
2) **Partitioning** of the coarse graph - Example Hendrickson and Leland [1]:
   - **Communication Metric - hop-weight:**
     - This metric respects the wires between processors and the edge weights. Possibly many messages (communication phase) are simultaneously competing for wires.
     - Minimizing only edge weights can cause message congestions.
     - Define $M : V \rightarrow P =$processors, $p_i =$processor which posses vertex $v_i$ and $h_{i,j} =$|wires between$p_i$ and $p_j$|.
     - hop-weight: $\text{hop-weight}(M) := \sum_{e_{i,j} \in E} \theta(e_{i,j})h_{i,j}$
     - Minimize hop-weight s.t. each core has approximately the same number of vertices.
     - Hendrickson and Leland [1] are approximating the minimal hop-weight and thus the partitioning of the vertices with a continuous optimal control problem, i.e. finding the minimum of a quadratic function in a finite dimensional euclidean vector space.
Fenics - Multi Level Graph Algorithm

3) **Refine** of the uncoarsed graph - Example LaSalle and Karypis [2]:
   i) **Refinement** is justified due to the **increasing of freedom** w.r.t. each uncoarsening phase.
   ii) METIS uses for example the **Greedy algorithm**:
       - Iterations over the vertices located on partition boundaries
       - In each iteration, **vertices are moved individually (reduction in edgecut weights)** with the restriction that each vertex can move only once per iteration.
       - Stopping criteria: No improvement in an iteration or maximum number of iterations reached.
Fenics - Ghost Nodes

After partitioning each **processor** posses its own

- **local nodes** (processors assigned nodes) and
- its **ghost nodes** (communication nodes), which are the neighbours of the local nodes and itself not local nodes.
Zlotnik algorithm with initial data \((y_0, y_1) = (0, 0)\) and forcing function 
\[ f(t, x, y) = \sin(2t) e^{-(x^2+y^2)}, \] 
\[ 3 \cdot 2^{2N+1} \] space nodes and \( T2^{N+1} \) time nodes:
Experiments

Refinement level $N=6$ with end time $T=10$:


Thank you for your attention