# Fast and smooth simulation of space-time problems

## Day 4


Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
*home.agh.edu.pl/paszynsk*

# Department of Computer Science
# AGH University, Kraków, Poland

# Outline

- Isogeometric finite element method
- Alternating Directions Implicit (ADI) method
- Isogeometric L2 projections
- Explicit dynamics
- Example 1: Heat transfer
- Installation of IGA-ADS solver
- Parallel distributed memory explicit dynamics
- Parallel shared memory explicit dynamics
- Example 2: Non-linear flow in heterogenous media
- **Implicit dynamics**
- **Example 3: Implicit heat transfer**
- **Example 4: Linear elasticity**
- **Example 5: Pollution problem**
- Labs with implict dynamics

# Software

Program Title: IGA-ADS
Code: `git clone https://github.com/marcinlos/iga-ads`
Licensing provisions: MIT license (MIT)
Programming language: C++
Nature of problem: Solving non-stationary problems in 1D, 2D and 3D
Solution method: Alternating direction solver with isogeometric finite element method

If you use this software in your work, please cite

Marcin Łoś, Maciej Woźniak, Maciej Paszyński, Andrew Lenharth, Keshav Pingali *IGA-ADS : Isogeometric Analysis FEM using ADS solver*, **Computer & Physics Communications** 217 (2017) 99-116 (available on researchgate.org)

# Alternating Direction Implicit (ADI) method

The Alternating Direction Implicit (ADI) method

G. Birkhoff, R.S. Varga, D. Young, *Alternating direction implicit methods*, **Advanced Computing** (1962)

$$\frac{du}{dt} - L_x u - L_y u = f$$

$$\frac{du}{dt} - \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} - \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} = f$$

$$\frac{u_{i,j}^{t+0.5} - u_{i,j}^{t}}{dt} - \frac{u_{i-1,j}^{t+0.5} - 2u_{i,j}^{t+0.5} + u_{i+1,j}^{t+0.5}}{h^2} = \frac{u_{i,j-1}^{t} - 2u_{i,j}^{t} + u_{i,j+1}^{t}}{h^2} + f_{i,j}^{t}$$

$$\frac{u_{i,j}^{t+1} - u_{i,j}^{t+0.5}}{dt} - \frac{u_{i,j-1}^{t+1} - 2u_{i,j}^{t+1} + u_{i,j+1}^{t+1}}{h^2} =$$
$$\frac{u_{i-1,j}^{t+0.5} - 2u_{i,j}^{t+0.5} + u_{i+1,j}^{t+0.5}}{h^2} + f_{i,j}^{t+0.5}$$

The Alternating Direction Implicit (ADI) method

G. Birkhoff, R.S. Varga, D. Young, *Alternating direction implicit methods*, **Advanced Computing** (1962)

$$u_{i-1,j}^{t+0.5}[-\frac{2dt}{h^2}] + u_{i,j}^{t+0.5}[1+\frac{2dt}{h^2}] + u_{i+1,j}^{t+0.5}[-\frac{2dt}{h^2}] = dt\frac{u_{i,j-1}^t - 2u_{i,j}^t + u_{i,j+1}^t}{h^2} + dtf_{i,j}^t$$

for $i = 1, ..., N_x$, $j = 1, ..., N_y$.

$$u_{i,j-1}^{t}[-\frac{2dt}{h^2}] + u_{i,j}^{t}[1+\frac{2dt}{h^2}] + u_{i,j+1}^{t}[-\frac{2dt}{h^2}] = dt\frac{u_{i-1,j}^{t+0.5} - 2u_{i,j}^{t+0.5} + u_{i-1,j}^{t+0.5}}{h^2} + dtf_{i,j}^{t+0.5}$$

for $j = 1, ..., N_y$, $i = 1, ..., N_x$.

The Alternating Direction Implicit (ADI) method
G. Birkhoff, R.S. Varga, D. Young, *Alternating direction implicit methods*, **Advanced Computing** (1962)

$$u_{i-1,j}^{t+0.5}[-2dt] + u_{i,j}^{t+0.5}[h^2 + 2dt] + u_{i+1,j}^{t+0.5}[-2dt] = dtu_{i,j-1}^{t} - 2u_{i,j}^{t} + u_{i,j+1}^{t} + h^2 dtf_{i,j}^{t}$$

for $i = 1, ..., N_x$, $j = 1, ..., N_y$.

$$u_{i,j-1}^{t}[-2dt] + u_{i,j}^{t}[h^2 + 2dt] + u_{i,j+1}^{t}[-2dt] = u_{i-1,j}^{t+0.5} - 2u_{i,j}^{t+0.5} + u_{i-1,j}^{t+0.5} + h^2 dtf_{i,j}^{t+0.5}$$

for $j = 1, ..., N_y$, $i = 1, ..., N_x$.

The Alternating Direction Implicit (ADI) method

G. Birkhoff, R.S. Varga, D. Young, *Alternating direction implicit methods*, **Advanced Computing** (1962)

$$\begin{bmatrix} h^2+2dt & -2dt & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -2dt & h^2+2dt & -2dt & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & -2dt & h^2+2dt & -2dt & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & -2dt & h^2+2dt & -2dt & \\ 0 & \cdots & \cdots & \cdots & 0 & -2dt & h^2+2dt & \end{bmatrix} \begin{bmatrix} u_{1,1}^{t+0.5} \\ u_{1,2}^{t+0.5} \\ u_{1,3}^{t+0.5} \\ \vdots \\ u_{N_x,N_y-1}^{t+0.5} \\ u_{N_x,N_y}^{t+0.5} \end{bmatrix}$$

$$=$$

$$\begin{bmatrix} -2u_{1,1}^t + u_{1,2}^t + h^2 dt f_{1,1}^t \\ u_{1,1}^t - 2u_{1,2}^t + u_{1,3}^t + h^2 dt f_{i,j}^t \\ \vdots \\ u_{N_x,N_y-2}^t - 2u_{N_x,N_y-1}^t + u_{N_x,N_y}^t + h^2 dt f_{N_x,N_y-1}^t \\ u_{N_x,N_y-1}^t - 2u_{N_x,N_y}^t + h^2 dt f_{N_x,N_y}^t \end{bmatrix}$$

# Implicit dynamics

$$\frac{du}{dt} - Lu = f$$

assuming constant coefficients and regular cube shape domain, where $L = L_x + L_y$ is a separable differential operator, e.g. Laplacian, where $L = L_x + L_y = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$. First, we apply the alternating direction method with respect to time. We introduce the intermediate time steps

$$\frac{u_{t+0.5} - u_t}{dt} - L_x u_{t+0.5} - L_y u_t = f_t$$

$$\frac{u_{t+1} - u_{t+0.5}}{dt} - L_x u_{t+0.5} - L_y u_{t+1} = f_{t+0.5}$$

We obtain

$$u_{t+0.5} - dt * L_x u_{t+0.5} = u_t + dt * L_y u_t + dt * f_t$$
$$u_{t+1} - dt * L_y u_{t+0.1} = u_{t+0.5} + dt * L_x u_{t+0.5} + dt * f_{t+0.5}$$

# Implicit dynamics

Now, we transform the problem into a weak form by taking L2-scalar products with test functions

$$(u_{t+0.5} - dt * L_x u_{t+0.5}, v) = (u_t + dt * L_y u_t + dt * f_t, v)$$

$$(u_{t+1} - dt * L_y u_{t+0.1}, v) = (u_{t+0.5} + dt * L_x u_{t+0.5} + dt * f_{t+0.5}, v)$$

# Implicit dynamics

We have $\frac{\partial}{\partial x}B^y_{j;p}(y)=0$ and $\frac{\partial}{\partial x}B^y_{l;p}(y)=0$. Namely,

$$\int B^x_{i;p}(x)B^y_{j;p}(y)B^x_{k;p}(x)B^y_{l;p}(y)dxdy+$$

$$\int(\frac{\partial}{\partial x}B^x_{i;p}(x))B^y_{j;p}(y)(\frac{\partial}{\partial x}B^x_{k;p}(x))B^y_{l;p}(y)$$

and we can separate directions

$$\int B^x_{i;p}(x)B^x_{k;p}(x)B^y_{j;p}(y)B^y_{l;p}(y)dxdy+$$

$$\int(\frac{\partial}{\partial x}B^x_{i;p}(x))(\frac{\partial}{\partial x}B^x_{k;p}(x))B^y_{j;p}(y)B^y_{l;p}(y)$$

so our left-hand-side matrix is the Kronecker product of

$$[\int(B^x_{i;p}(x)B^x_{k;p}(x)+(\frac{\partial}{\partial x}B^x_{i;p}(x))(\frac{\partial}{\partial x}B^x_{k;p}(x)))dx]*$$

$$[\int(B^y_{j;p}(y)*B^y_{l;p}(y))dy]$$

and this can be expressed as multiplication of two multi-diagonal matrices, to be factorized in a linear $O(N)$ cost.

# Implicit dynamics

We can apply the same process for the second equation. Namely, the matrix of the left-hand-side $(u_{t+1}, v) - (L_y u_{t+1}, v)$ of (1) have terms

$$\int B^x_{i;p}(x)B^y_{j;p}(y)B^x_{k;p}(x)B^y_{l;p}(y)dxdy+$$

$$\int \frac{\partial}{\partial y}(B^x_{i;p}(x)B^y_{j;p}(y))\frac{\partial}{\partial y}(B^x_{k;p}(x)B^y_{l;p}(y))dxdy$$

Now, since $\frac{\partial}{\partial y}B^x_{i;p}(x) = 0$ and $\frac{d}{dy}B^x_{k;p}(x) = 0$ we get

$$\int B^x_{i;p}(x)B^y_{j;p}(y)B^x_{k;p}(x)B^y_{l;p}(y)dxdy+$$

$$\int B^x_{i;p}(x)(\frac{\partial}{\partial y}B^y_{j;p}(y))B^x_{k;p}(x)(\frac{\partial}{\partial y}B^y_{l;p}(y))$$

# Implicit dynamics

We can separate directions

$$\int B^x_{i;p}(x)B^x_{k;p}(x)B^y_{j;p}(y)B^y_{l;p}(y)dxdy+$$

$$\int(B^x_{i;p}(x)B^x_{k;p}(x)(\frac{\partial}{\partial y}B^y_{j;p}(y))(\frac{\partial}{\partial y}B^y_{l;p}(y))$$

so our left-hand-side matrix is the Kronecker product of

$$[\int(B^x_{i;p}(x)B^x_{k;p}(x)dx]*$$

$$[\int(B^y_{j;p}(y)*B^y_{l;p}(y))dy+(\frac{\partial}{\partial y}B^x_{j;p}(y))(\frac{\partial}{\partial y}B^y_{l;p}(y)))]$$

and this can be expressed as multiplication of two matrices, and both the first and the second matrix are mutli-diagonal and can be factorized in a linear $O(N)$ cost.

# Unconditional stability

Let us consider heat equation on 2D domain

$$\partial_t u - \Delta u = f \tag{1}$$

Assuming zero boundary conditons (Dirichlet or Neumann) the weak formulation is given by

$$(\partial_t u, v)_{L^2} = -(\nabla u, \nabla v)_{L^2} + (u, f)_{L^2} \tag{2}$$

Let $\mathcal{B}_{ij}(x, y) = \mathcal{B}^x_i(x)\mathcal{B}^y_j(y)$ be the standard tensor product basis. We seek solution of the form

$$u(\mathbf{x}, t) = \sum u^{ij}(t)\, \mathcal{B}_{ij}(\mathbf{x})$$

# Unconditional stability

Let us denote

$$\mathbf{M} = [(\mathcal{B}_{ij}, \mathcal{B}_{kl})_{L^2}] \qquad \mathbf{S} = [(\nabla\mathcal{B}_{ij}, \nabla\mathcal{B}_{kl})_{L^2}] \qquad \mathbf{F} = [(f, \mathcal{B}_{ij})_{L^2}] \tag{3}$$

and

$$\mathbf{M}_x = \left[(\mathcal{B}_i^x, \mathcal{B}_j^y)_{L^2}\right] \qquad \mathbf{S}_x = \left[(\partial_x\mathcal{B}_i^x, \partial_x\mathcal{B}_j^x)_{L^2}\right] \tag{4}$$

and similarly for $\mathbf{M}_y$, $\mathbf{M}_y$.
All these matrices are symmetric and positive definite, as Gram matrices of certain sets of functions.
It is straightforward to prove that

$$\mathbf{M} = \mathbf{M}_x \otimes \mathbf{M}_y \qquad \mathbf{S} = \mathbf{S}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{S}_y \tag{5}$$

# Unconditional stability

In the implicit scheme we have

$$
\begin{aligned}
(u_{t+\frac{1}{2}}, v)_{L^2} + \frac{h}{2}(\partial_x u_{t+\frac{1}{2}}, \partial_x v)_{L^2} &= (u_t, v)_{L^2} - \frac{h}{2}(\partial_y u_t, \partial_y v)_{L^2} \\
(u_{t+1}, v)_{L^2} + \frac{h}{2}(\partial_y u_{t+1}, \partial_y v)_{L^2} &= (u_{t+\frac{1}{2}}, v)_{L^2} - \frac{h}{2}(\partial_x u_{t+\frac{1}{2}}, \partial_x v)_{L^2}
\end{aligned} \tag{6}
$$

Resulting linear systems may be expressed as

$$
\begin{aligned}
\left(\mathbf{M} + \frac{h}{2}\mathbf{S}_x \otimes \mathbf{M}_y\right) \mathbf{u}_{t+\frac{1}{2}} &= \left(\mathbf{M} - \frac{h}{2}\mathbf{M}_x \otimes \mathbf{S}_y\right) \mathbf{u}_t \\
\left(\mathbf{M} + \frac{h}{2}\mathbf{M}_x \otimes \mathbf{S}_y\right) \mathbf{u}_{t+1} &= \left(\mathbf{M} - \frac{h}{2}\mathbf{S}_x \otimes \mathbf{M}_y\right) \mathbf{u}_{t+\frac{1}{2}}
\end{aligned} \tag{7}
$$

# Unconditional stability

Since $\mathbf{M} = \mathbf{M}_x \otimes \mathbf{M}_y$,

$$\begin{aligned}
\left[\left(\mathbf{M}_x + \frac{h}{2}\mathbf{S}_x\right) \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}} &= \left[\mathbf{M}_x \otimes \left(\mathbf{M}_y - \frac{h}{2}\mathbf{S}_y\right)\right] \mathbf{u}_t \\
\left[\mathbf{M}_x \otimes \left(\mathbf{M}_y + \frac{h}{2}\mathbf{S}_y\right)\right] \mathbf{u}_{t+1} &= \left[\left(\mathbf{M}_x - \frac{h}{2}\mathbf{S}_x\right) \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}}
\end{aligned} \tag{8}$$

Let us denote

$$\begin{aligned}
\mathbf{K}_x^+ &= \mathbf{M}_x + \frac{h}{2}\mathbf{S}_x \qquad & \mathbf{K}_x^- &= \mathbf{M}_x - \frac{h}{2}\mathbf{S}_x \\
\mathbf{K}_y^+ &= \mathbf{M}_y + \frac{h}{2}\mathbf{S}_y \qquad & \mathbf{K}_y^- &= \mathbf{M}_y - \frac{h}{2}\mathbf{S}_y
\end{aligned} \tag{9}$$

Then we can write

$$\begin{aligned}
\left[\mathbf{K}_x^+ \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}} &= \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] \mathbf{u}_t \\
\left[\mathbf{M}_x \otimes \mathbf{K}_y^+\right] \mathbf{u}_{t+1} &= \left[\mathbf{K}_x^- \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}}
\end{aligned} \tag{10}$$

$$\begin{aligned}\left[\mathbf{K}_x^+ \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}} &= \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] \mathbf{u}_t \\ \left[\mathbf{M}_x \otimes \mathbf{K}_y^+\right] \mathbf{u}_{t+1} &= \left[\mathbf{K}_x^- \otimes \mathbf{M}_y\right] \mathbf{u}_{t+\frac{1}{2}}\end{aligned} \tag{11}$$

and so finally, combining two steps

$$\mathbf{u}_{t+1} = \left[\mathbf{M}_x \otimes \mathbf{K}_y^+\right]^{-1} \left[\mathbf{K}_x^- \otimes \mathbf{M}_y\right] \left[\mathbf{K}_x^+ \otimes \mathbf{M}_y\right]^{-1} \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] \mathbf{u}_t \tag{12}$$

This can be simplified using properties of tensor product:

- $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$
- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$ whenever the products make sense

# Unconditional stability

Using these, we conclude that

$$
\begin{aligned}
&\left[\mathbf{M}_x \otimes \mathbf{K}_y^+\right] \left[\mathbf{K}_x^- \otimes \mathbf{M}_y\right] \left[\mathbf{K}_x^+ \otimes \mathbf{M}_y\right]^{-1} \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] = \\
&\left[\mathbf{M}_x^{-1} \otimes \left(\mathbf{K}_y^+\right)^{-1}\right] \left[\mathbf{K}_x^- \otimes \mathbf{M}_y\right] \left[\left(\mathbf{K}_x^+\right)^{-1} \otimes \mathbf{M}_y^{-1}\right] \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] = \\
&\left[\mathbf{M}_x^{-1} \otimes \left(\mathbf{K}_y^+\right)^{-1}\right] \left[\mathbf{K}_x^- \left(\mathbf{K}_x^+\right)^{-1} \otimes \mathbf{I}\right] \left[\mathbf{M}_x \otimes \mathbf{K}_y^-\right] = \\
&\left[\mathbf{M}_x^{-1} \otimes \left(\mathbf{K}_y^+\right)^{-1}\right] \left[\mathbf{K}_x^- \left(\mathbf{K}_x^+\right)^{-1} \mathbf{M}_x \otimes \mathbf{K}_y^-\right] = \\
&\left[\mathbf{M}_x^{-1}\mathbf{K}_x^- \left(\mathbf{K}_x^+\right)^{-1} \mathbf{M}_x\right] \otimes \left[\left(\mathbf{K}_y^+\right)^{-1} \mathbf{K}_y^-\right]
\end{aligned}
\tag{13}
$$

Using these, we conclude that

$$\mathbf{u}_{t+1} = \left[\mathbf{M}_x^{-1}\mathbf{K}_x^{-}\left(\mathbf{K}_x^{+}\right)^{-1}\mathbf{M}_x\right] \otimes \left[\left(\mathbf{K}_y^{+}\right)^{-1}\mathbf{K}_y^{-}\right]\mathbf{u}_t \tag{14}$$

Eigenvalues of $\mathbf{A} \otimes \mathbf{B}$ are products of eigenvalues of $\mathbf{A}$ and $\mathbf{B}$,
so it is enough to determine eigenvalues of the above two matrices.
The matrix $\left[\mathbf{M}_x^{-1}\mathbf{K}_x^{-}\left(\mathbf{K}_x^{+}\right)^{-1}\mathbf{M}_x\right]$ is similar to $\mathbf{K}_x^{-}\left(\mathbf{K}_x^{+}\right)^{-1}$
and so it has the same eigenvalues.
Furthermore, $\mathbf{AB}$ and $\mathbf{BA}$ always have the same spectrum,
so $\mathbf{K}_x^{-}\left(\mathbf{K}_x^{+}\right)^{-1}$ has the same eigenvalues as $\left(\mathbf{K}_x^{+}\right)^{-1}\mathbf{K}_x^{-}$.
So we check the eigenvalues of $\left(\mathbf{K}_x^{+}\right)^{-1}\mathbf{K}_x^{-}$.
By **Lemma** eigenvalues $\lambda$ of $\left(\mathbf{K}_x^{+}\right)^{-1}\mathbf{K}_x^{-}$ and $\left(\mathbf{K}_y^{+}\right)^{-1}\mathbf{K}_y^{-}$
satisfy $|\lambda| < 1$, so the single step full matrix has spectral radius <1.
Thus, the implicit ADS scheme is unconditiionally stable.

# Unconditional stability

## Lemma

*Let* $\mathbf{A}$, $\mathbf{B}$ *be symmetric and positive-definite. For each eigenvalue* $\lambda$ *of* $(\mathbf{A}+\mathbf{B})^{-1}(\mathbf{A}-\mathbf{B})$ *we have* $|\lambda| < 1$.

## Proof.

Let $\lambda \neq 0$ be such eigenvalue and let $\mathbf{x}$ be the corresponding eigenvector. Then $(\mathbf{A}+\mathbf{B})^{-1}(\mathbf{A}-\mathbf{B})\,\mathbf{x} = \lambda\mathbf{x}$ and so
$(\mathbf{A}-\mathbf{B})\,\mathbf{x} = \lambda\,(\mathbf{A}+\mathbf{B})\,\mathbf{x}$ and $(1-\lambda)\mathbf{A}\mathbf{x} = (1+\lambda)\mathbf{B}\mathbf{x}$.
Since $\mathbf{B}$ is positive-definite it is nonsingular, and so $\mathbf{B}\mathbf{x} \neq 0$.
Thus $\lambda \neq 1$. Multiplying by $\mathbf{x}^{\mathrm{T}}$ on the left gives $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \frac{1+\lambda}{1-\lambda}\,\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x}$.
Since $\mathbf{A}$ and $\mathbf{B}$ are positive definite, both products are positive,
thus $\lambda \in \mathbb{R}$ and $\frac{1+\lambda}{1-\lambda} > 0 \implies \lambda^2 < 1 \iff |\lambda| < 1$ $\square$

# Example 3: Implicit heat transfer

Click in the middle
1000 less time steps

Unit cube deformed by short impulse applied at the corner

$$\begin{cases} \rho \dfrac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{F} & \text{on } \Omega \times [0, T] \\ \boldsymbol{u}(x, 0) = 0 & \text{for } x \in \Omega \\ \boldsymbol{\sigma} \cdot \widehat{\mathbf{n}} = 0 & \text{on } \partial\Omega \end{cases} \tag{15}$$

where $\Omega = [0, 1]^3$ is an unit cube,
$\mathbf{u}$ is an unknown 3-dimensional displacement vector
$\rho$ is material density,
$\mathbf{f}$ is the applied external force,
$\sigma_{ij} = c_{ijkl}\epsilon_{lk}$ is the stress tensor
where $\epsilon_{ij} = \frac{1}{2}\left(\partial_j u_i + \partial_i u_j\right)$
and $\mathbf{c}$ is rank-4 elasticity tensor (prescribed for a given material)

## Example 4: Propagation of elastic waves

$\rho = 1$,
$\mathbf{c} = 0$ except for $c_{ijij} = c_{ijji} = 1$ for $i, j = 1, 2, 3$,
so that $\mathbf{c}$ is positive definite and satisfies symmetry constraints stemming from its physical meaning.
The force applied is given by

$$\mathbf{F}(\mathbf{x}, t) = -\phi(t/t_0) r(\mathbf{x})\, \mathbf{p} \tag{16}$$

$$\mathbf{p} = (1, 1, 1) \tag{17}$$

$$t_0 = 0.02 \tag{18}$$

$$\phi(t) = \begin{cases} t^2(1-t)^2 & \text{if } t \in (0,1) \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$r(\mathbf{x}) = 10 \exp\left(-10 \left\|\mathbf{x} - \mathbf{p}\right\|^2\right) \tag{20}$$

i.e. a short impulse directed towards the origin, applied at the opposite corner of the cube.

# Example 4: Propagation of elastic waves

In order to utilize forward Euler integration scheme, the above system of 3 equations is converted to system of 6 equations by introducing additional variables corresponding to components of the displacement's velocity:

$$\mathbf{v_i} = \frac{\partial \mathbf{u_i}}{\partial t} \tag{21}$$

Corresponding weak formulation discretized with the Euler scheme is given by

$$\begin{cases} \langle u_i^{(t+1)}, w \rangle = \langle u_i^{(t)} + \Delta t \, v_i^{(t)}, w \rangle \\ \langle v_i^{(t+1)}, w \rangle = \langle v_i^{(t)} + \dfrac{\Delta t}{\rho} \left( \sigma_{ij,j} + F_i \right), w \rangle \end{cases} \tag{22}$$

for all $w \in H^1(\Omega)$, where $\langle \cdot, \cdot \rangle$ denotes standard scalar product in $L^2(\Omega)$, and $u_i, v_i \in H^1(\Omega)$.

# Code for Example 4 (Linear elasticity)

"problems/elasticity/main.cpp"

We are going to use multiple cores

```
#include "ads/executor/galois.hpp"
#include "problems/elasticity/elasticity.hpp"
```

pilot for the simulation

```
int main() {
```

quadratic B-splnes, 12 elements along axis

```
  dim_config dim{ 2, 12 };
```

40000 time steps, time step size $10^{-4}$

```
  timesteps_config steps{ 40000, 1e-4 };
```

we will need to compute first derivatives during the computations

```
  int ders = 1;
```

some auxiliary objects for configuration and simulation

```
  config_3d c{dim, dim, dim, steps, ders};
  problems::linear_elasticity sim{c};
```

run the simulation

```
  sim.run();
}
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

```
#include <cmath>
#include "ads/simulation.hpp"   Parallel loop processing
#include "ads/executor/galois.hpp" Generation of graphics
#include "ads/output_manager.hpp"
namespace problems {
class linear_elasticity :  public ads::simulation_3d {
struct state {
vector_type ux, uy, uz;
vector_type vx, vy, vz;
state(std::array<std::size_t, 3> shape)
   :  ux{ shape }, uy{ shape }, uz{ shape }
   :  vx{ shape }, vy{ shape }, vz{ shape }
   { }
};
state now, prev;
```

The class to write down output for graphics

```
ads::output_manager<3> output;
```

The class to parallel loop processing

```
ads::galois_executor executor{8};
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

```
class linear_elasticity : public ads::simulation_3d
{
...
```

implementation of the initial state

```
double init_state(double x, double y, double z)
```

executed once before the simulation starts

```
void before() override
```

executed before every simulation step

```
void before_step() override
```

implementation of the simulation step

```
void step() override
```

executed after every simulation step

```
void after_step() override
```

implementation of generation of RHS

```
void compute_rhs() override
```

executed once after the simulation ends

```
void after() override
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

this function is called before every time step

```
void before_step(int /*iter*/, double /*t*/) override
{
    using std::swap;
```

swap $u_t$ and $u_{t-1}$

```
    swap(u, u_prev);
}
```

this function implements every time step

```
void step(int /*iter*/, double /*t*/) override {
```

generate new RHS using u_prev

```
    compute_rhs();
```

forward and backward substitutions with multiple RHS

```
    for_all(now, [this](vector_type& a) { solve(a); });
}
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

this function is called once before the simulation starts

```
void before() override {
```

performs LU factorization of three 1D systems, representing B-splines along $x$, $y$ and $z$ axes

```
    prepare_matrices();
    }

void compute_rhs(double t) {
for_all(now, [](vector_type& a) { zero(a); });
  executor.for_each(elements(), [&](index_type e) {
  auto local = local_contribution(e, t);
  executor.synchronized([&] {
  apply_local_contribution(local, e);
  });
});
}
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

```
void apply_local_contribution(const state& loc,
index_type e) { //    update_global_rhs(now.ux,
loc.ux, e);
   update_global_rhs(now.uy, loc.uy, e);
   update_global_rhs(now.uz, loc.uz, e);
   update_global_rhs(now.vx, loc.vx, e);
   update_global_rhs(now.vy, loc.vy, e);
   update_global_rhs(now.vz, loc.vz, e);
  }
```

# Code for Example 4 (Linear elasticity)

```
state local_contribution(index_type e, double t) const
{
auto local=state{ local_shape()};doubleJ=jacobian(e);
for (auto q :  quad_points()) {
  auto x = point(e, q); double w = weigth(q);
  value_type ux = eval_fun(prev.ux, e, q);
  value_type uy = eval_fun(prev.uy, e, q);
  value_type uz = eval_fun(prev.uz, e, q);
  value_type vx = eval_fun(prev.vx, e, q);
  value_type vy = eval_fun(prev.vy, e, q);
  value_type vz = eval_fun(prev.vz, e, q);
  tensor eps = {
   {ux.dx,0.5*(ux.dy+uy.dx),0.5*(ux.dz+uz.dx) },
   {0.5*(ux.dy+uy.dx),uy.dy,0.5*(uy.dz+uz.dy) },
   {0.5*(ux.dz+uz.dx),0.5*(uy.dz+uz.dy),uz.dz }
  };
  tensor s; stress_tensor(s, eps);
  auto F = force(x, t);
```

# Code for Example 4 (Linear elasticity)

```
for (auto a :  dofs_on_element(e)) {
  value_type b = eval_basis(e, q, a);
  double rho = 1;
  double axb = (-s[0][0]*b.dx - s[0][1]*b.dy - s[0][2]*b.dz +
   F[0]*b.val) / rho;
  double ayb = (-s[1][0]*b.dx - s[1][1]*b.dy - s[1][2]*b.dz +
   F[1]*b.val) / rho;
  double azb = (-s[2][0]*b.dx - s[2][1]*b.dy - s[2][2]*b.dz +
   F[2]*b.val) / rho;
  double dt = steps.dt; double t2 = dt * dt / 2;
  auto aa = dof_global_to_local(e, a);
  ref(local.ux, aa) += ((ux.val+dt*vx.val)*b.val + t2*axb) *w*J;
  ref(local.uy, aa) += ((uy.val+dt*vy.val)*b.val + t2*ayb) *w*J;
  ref(local.uz, aa) += ((uz.val+dt*vz.val)*b.val + t2*azb) *w*J;
  ref(local.vx, aa) += (vx.val * b.val + dt * axb) * w*J;
  ref(local.vy, aa) += (vy.val * b.val + dt * ayb) * w*J;
  ref(local.vz, aa) += (vz.val * b.val + dt * azb) * w*J;
```

# Code for Example 4 (Linear elasticity)

"problems/elasticity/elasticity.hpp"

```
void after_step(int iter, double t) override {
if (iter % 100 == 0) {
  double Ek = kinetic_energy();
  double Ep = potential_energy();
  compute_potential_energy();
  output.to_file("out_%d.vti", iter,
    output.evaluate(now.ux),
    output.evaluate(now.uy),
    output.evaluate(now.uz),
    output.evaluate(energy));
  }
```

Click in the middle

# Direction splitting for linear elasticity

We solve linear elasticity problem given by

$$\begin{cases} \rho\,\partial_{tt}\mathbf{u} = \nabla\cdot\boldsymbol{\sigma} + \mathbf{F} & \text{on } \Omega\times[0,\,T] \\ \boldsymbol{u}(x,0) = u_0 & \text{for } x\in\Omega \\ \boldsymbol{\sigma}\cdot\widehat{\mathbf{n}} = 0 & \text{on } \partial\,\Omega \end{cases} \tag{23}$$

where $\Omega = [0,1]^3$ is an unit cube, $\mathbf{u}$ is a 3-dimensional displacement vector to be calculated, $\rho$ is material density, $\mathbf{F}$ is the applied external force, and $\boldsymbol{\sigma}$ is the Cauchy stress tensor, given by

$$\sigma_{ij} = c_{ijkl}\epsilon_{lk}, \qquad \epsilon_{ij} = \frac{1}{2}\left(\partial_j u_i + \partial_i u_j\right) \tag{24}$$

and $\mathbf{c}$ is the elasticity tensor.

# Direction splitting for linear elasticity

The above second-order system can be converted to system of 6 first-order equations by introducing additional variable $\mathbf{v} = \partial_t \mathbf{u}$:

$$\begin{cases} \partial_t \mathbf{u} = \mathbf{v} \\ \rho \partial_t \mathbf{v} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{F} \end{cases} \tag{25}$$

We assume an isotropic elastic material and thus the Lame parameters,

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda \operatorname{tr} \boldsymbol{\epsilon}\, I \tag{26}$$

thus

$$\nabla \cdot \boldsymbol{\sigma} = 2\mu(\nabla \cdot \boldsymbol{\epsilon}) + \lambda \nabla \operatorname{tr} \boldsymbol{\epsilon} \tag{27}$$

Furthermore,

$$\nabla \cdot \epsilon = \frac{1}{2}\begin{pmatrix} 2\partial_{xx} u_x & + \partial_{yy} u_x + \partial_{yx} u_y & + \partial_{zz} u_x + \partial_{zx} u_z \\ \partial_{xx} u_y + \partial_{xy} u_x & + 2\partial_{yy} u_y & + \partial_{zz} u_y + \partial_{zy} u_z \\ \partial_{xx} u_z + \partial_{xz} u_x & + \partial_{yy} u_z + \partial_{yz} u_y & + 2\partial_{zz} u_z \end{pmatrix} \tag{28}$$

and

$$\nabla \operatorname{tr} \epsilon = \begin{pmatrix} \partial_{xx} u_x + \partial_{xy} u_y + \partial_{xz} u_z \\ \partial_{yx} u_x + \partial_{yy} u_y + \partial_{yz} u_z \\ \partial_{zx} u_x + \partial_{zy} u_y + \partial_{zz} u_z \end{pmatrix} \tag{29}$$

# Direction splitting for linear elasticity

We split every timestep into three substeps

$$\begin{cases} \mathbf{u}^{(t+\frac{1}{3})} = \mathbf{u}^{(t)} + \dfrac{dt}{3}\,\mathbf{v}^{(t)} \\ \rho\mathbf{v}^{(t+\frac{1}{3})} = \rho\mathbf{v}^{(t)} + \dfrac{dt}{3}\left(\nabla\cdot\boldsymbol{\sigma}^{(t+\frac{1}{3})} + \mathbf{F}\right) \end{cases} \tag{30}$$

$$\begin{cases} \mathbf{u}^{(t+\frac{2}{3})} = \mathbf{u}^{(t+\frac{1}{3})} + \dfrac{dt}{3}\,\mathbf{v}^{(t+\frac{1}{3})} \\ \rho\mathbf{v}^{(t+\frac{2}{3})} = \rho\mathbf{v}^{(t+\frac{1}{3})} + \dfrac{dt}{3}\left(\nabla\cdot\boldsymbol{\sigma}^{(t+\frac{2}{3})} + \mathbf{F}\right) \end{cases} \tag{31}$$

# Direction splitting for linear elasticity

$$
\begin{cases} \mathbf{u}^{(t+1)} = \mathbf{u}^{(t+\frac{2}{3})} + \dfrac{dt}{3}\mathbf{v}^{(t+\frac{2}{3})} \\ \rho\mathbf{v}^{(t+1)} = \rho\mathbf{v}^{(t+\frac{2}{3})} + \dfrac{dt}{3}\left(\nabla\cdot\boldsymbol{\sigma}^{(t+1)} + \mathbf{F}\right) \end{cases} \tag{32}
$$

and $\boldsymbol{\sigma}^{(t+\frac{k}{3})}$ are constructed using

$$
\mathbf{u} = \mathbf{u}^{(t+\frac{k-1}{3})} + \frac{dt}{3}\,\mathbf{v}^{(t+\frac{k-1}{3})}
$$

in most places, except when $u_i$ appears inside $i$-th component of $\nabla\cdot\boldsymbol{\sigma}$ under a double derivative with respect to $x$ ($k=1$), $y$ ($k=2$) or $z$ ($k=3$).

# Direction splitting for linear elasticity

These cases are marked in equations (28) and (29) with red, brown, and blue color, respectively. In other words, we separate the operator into its diagonal part and the off-diagonal part, the diagonal part we treat implicitly, while the remainder we treat explicitly. After moving all the terms with values to be computed to the left-hand side, substep equations have the following form:

$$\begin{cases} \rho\, v_x^{(t+\frac{1}{3})} - \dfrac{dt}{3}(\lambda + 2\mu)\, \partial_{xx} v_x^{(t+\frac{1}{3})} & = \dots \\ \rho\, v_y^{(t+\frac{1}{3})} - \dfrac{dt}{3}\mu\, \partial_{xx} v_y^{(t+\frac{1}{3})} & = \dots \\ \rho\, v_z^{(t+\frac{1}{3})} - \dfrac{dt}{3}\mu\, \partial_{xx} v_z^{(t+\frac{1}{3})} & = \dots \end{cases} \tag{33}$$

$$\begin{cases} \rho\, v_x^{(t+\frac{2}{3})} - \dfrac{dt}{3}\mu\, \partial_{yy} v_x^{(t+\frac{2}{3})} & = \dots \\ \rho\, v_y^{(t+\frac{2}{3})} - \dfrac{dt}{3}(\lambda + 2\mu)\, \partial_{yy} v_y^{(t+\frac{2}{3})} & = \dots \\ \rho\, v_z^{(t+\frac{2}{3})} - \dfrac{dt}{3}\mu\, \partial_{yy} v_z^{(t+\frac{2}{3})} & = \dots \end{cases} \tag{34}$$

# Direction splitting for linear elasticity

$$\begin{cases} \rho\, v_x^{(t+1)} - \dfrac{dt}{3}\mu\, \partial_{zz} v_x^{(t+1)} & = \dots \\ \rho\, v_y^{(t+1)} - \dfrac{dt}{3}\mu\, \partial_{zz} v_y^{(t+1)} & = \dots \\ \rho\, v_z^{(t+1)} - \dfrac{dt}{3}(\lambda + 2\mu)\, \partial_{zz} v_z^{(t+1)} & = \dots \end{cases} \tag{35}$$

This formulation is then transformed into a sequence of isogeometric projections in a standard way.

Click in the middle 10 times less time steps

## Example 5: Pollution from a chimney with a wind

We seek the pollution density scalar field $c \colon \Omega \to \mathbb{R}$ such as:

$$\begin{cases} \dfrac{\partial c}{\partial t} + u \cdot \nabla c - \nabla \cdot (K \nabla c) = e & \text{on } \Omega \times [0, T] \\ \nabla c \cdot \hat{\mathbf{n}} = 0 & \text{on } \partial\Omega \times [0, T] \\ c(\mathbf{x}, 0) = c_0(\mathbf{x}) & \text{on } \Omega \end{cases} \tag{36}$$

where $\Omega = [0, 1]^3$,
$\hat{\mathbf{n}}$ is a normal vector of the domain boundary,
$T$ is a length of the time interval for the simulation,
$u$ is the prescribed wind,
$e$ is the prescribed emission from the chimney,
$K$ is the diffusion,
and $c_0$ is an initial state.

# Example 5: Pollution from a chimney with a wind

$\Omega = 5km \times 5km \times 5km$
Mesh size $= 100 \times 100 \times 100$
Wind $= F * (cosa(t), sina(t), v(t))$ where
$a(t) = \pi/3(sin(s) + 0.5sin(2.3s)) + 3/8\pi$
$v(t) = 1/3sin(s)$
$s = t/150$
chimney $e(p) = (r-1)^2(r+1)^2$ where $r = min(1, (|p - p_0|/25)^2)$
$p_0 = (3000, 2000, 2000)$
Diffusion $K = (50, 50, 0.5)$
We run 300 time steps of the implicit method

# Example 5: Pollution from a chimney with a wind

Click in the middle