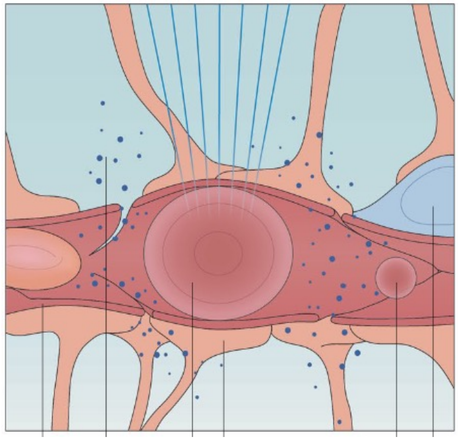
Fast finite element solver for focused ultrasound applications

Adeeb Arif Kor[*†], Igor Baratta[*], Garth Wells[*]

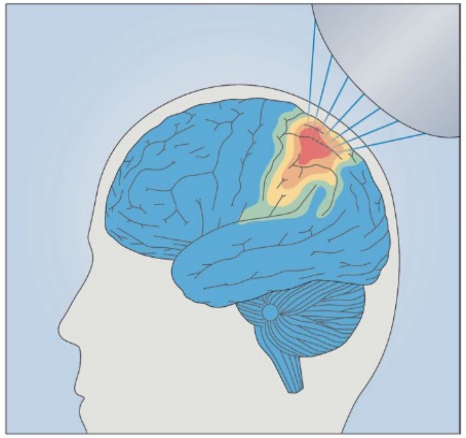
[*]University of Cambridge

[†]University of Malaya

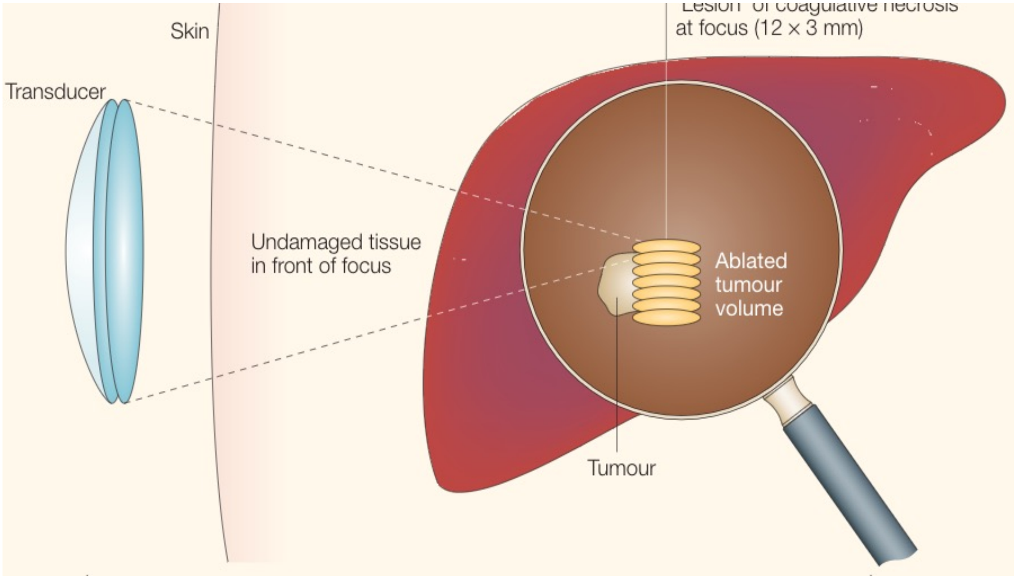
What is focused-ultrasound?



†Drug delivery



†Neuromodulation

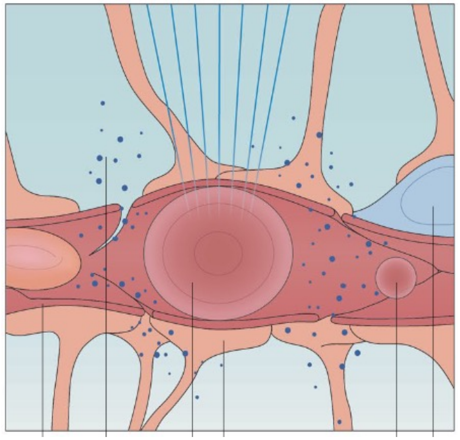


*Thermoablation

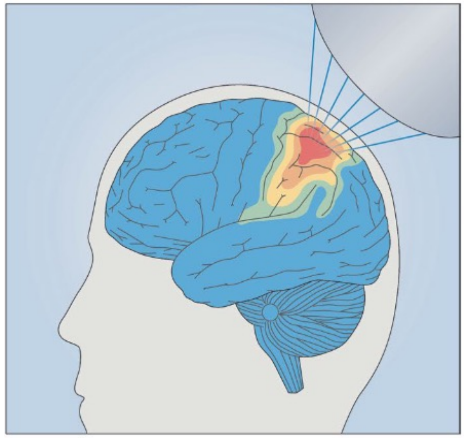
Promising mode of treatment for various medical conditions

- †Meng et. al. (2020)
- *Kennedy (2005)

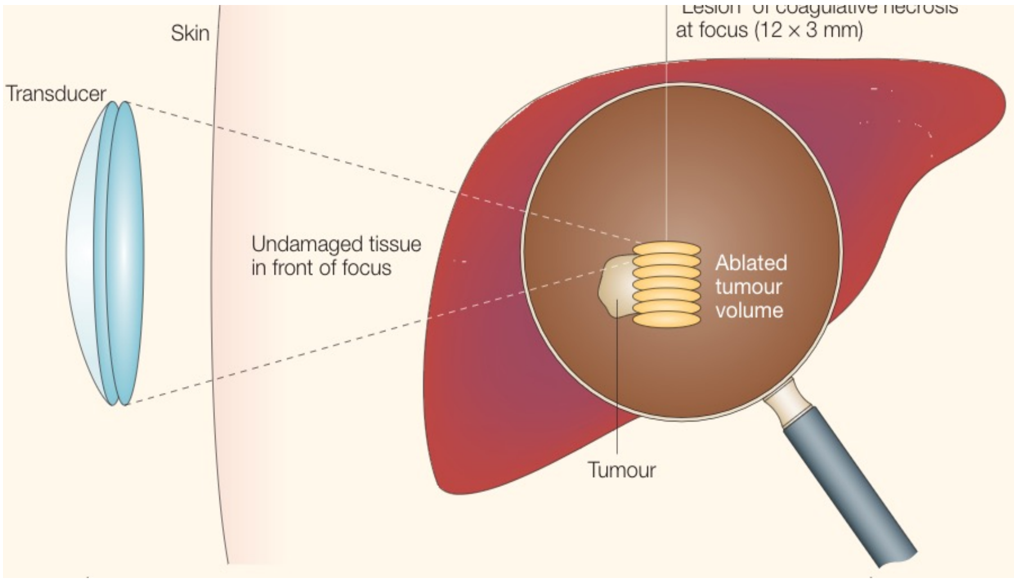
How it works?



†Drug delivery



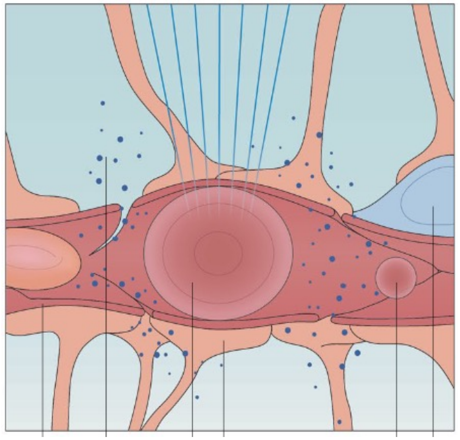
†Neuromodulation



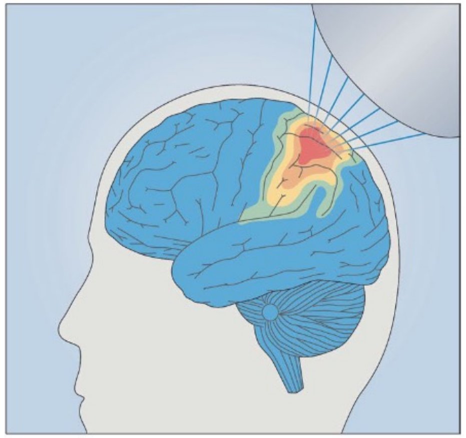
⁎Thermoablation

Focused ultrasound transducer generates and focuses acoustic waves on the targeted region

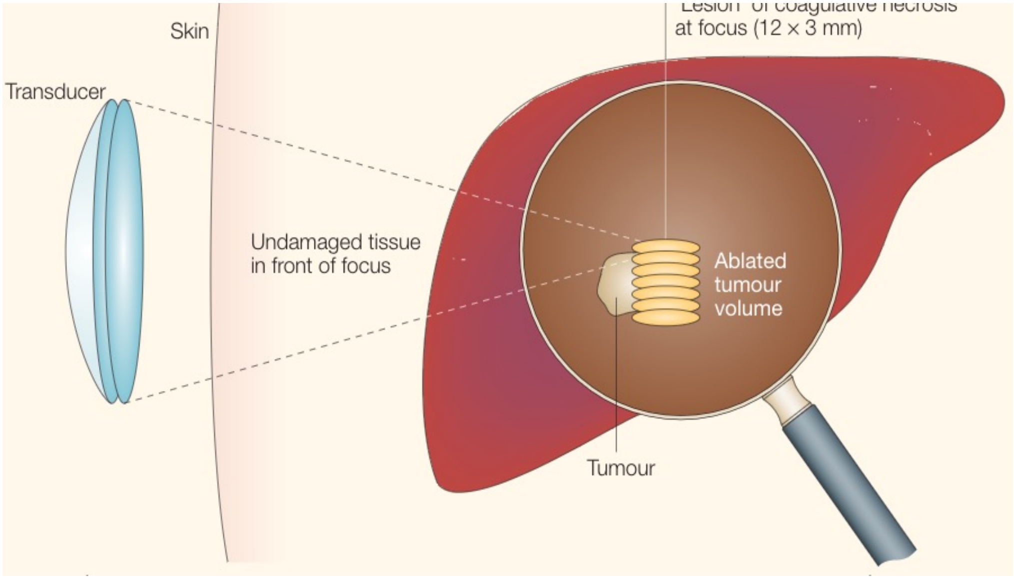
Advantages



†Drug delivery



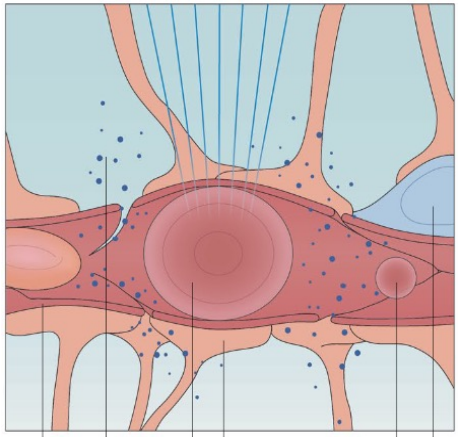
†Neuromodulation



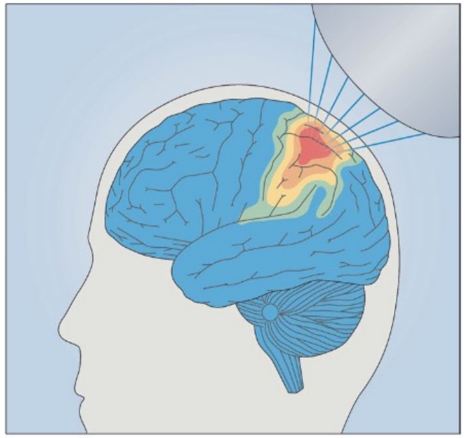
*Thermoablation

- Non-invasive
- The effect is localized
- Possibility of multiple repeated treatments

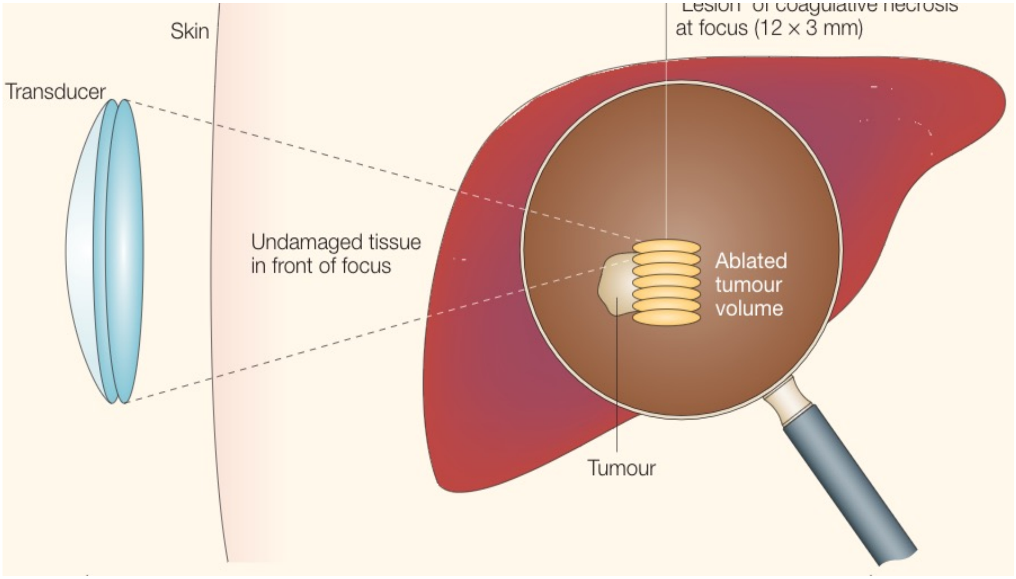
Advantages



†Drug delivery



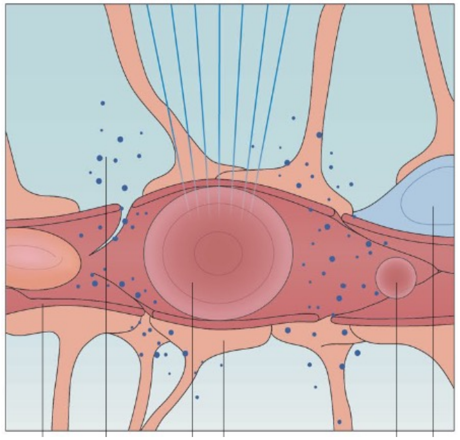
†Neuromodulation



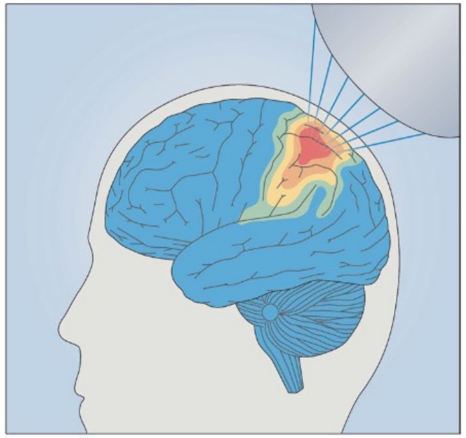
⁕Thermoablation

- Non-invasive
- The effect is localized
- Possibility of multiple repeated treatments

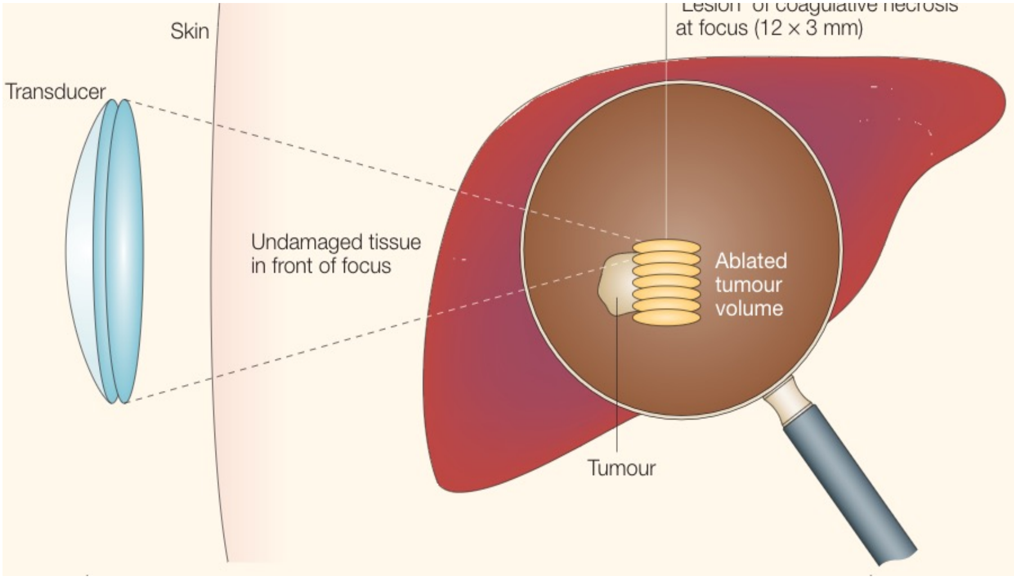
Advantages



†Drug delivery



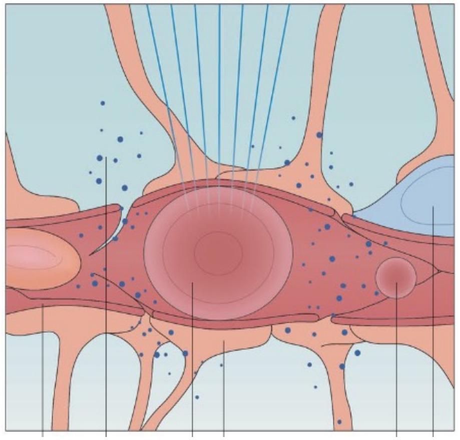
†Neuromodulation



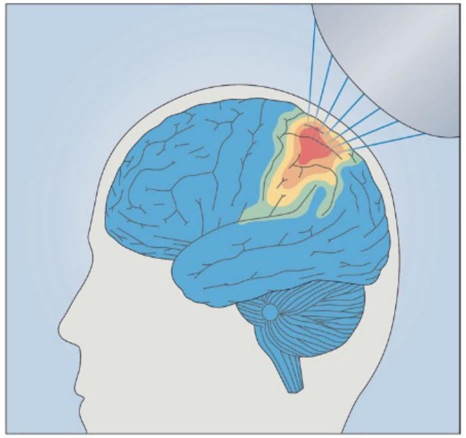
⁎Thermoablation

- Non-invasive
- The effect is localized
- Possibility of multiple repeated treatments

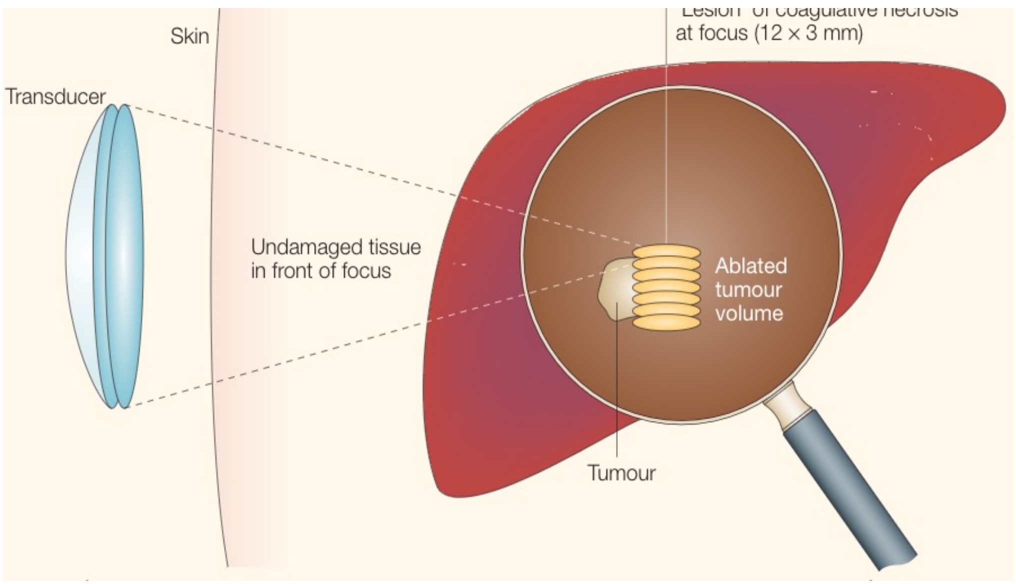
Why simulate?



†Drug delivery



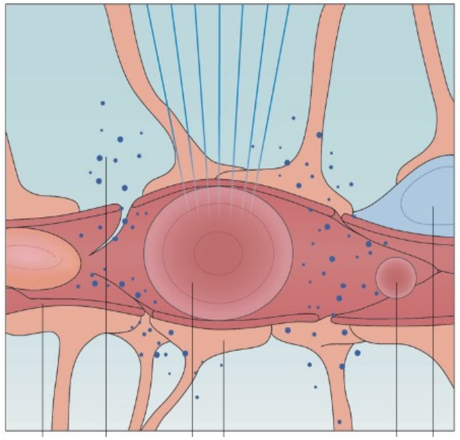
†Neuromodulation



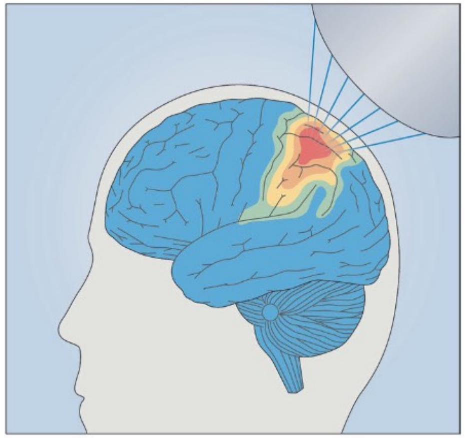
⁎Thermoablation

- Enable patient specific treatment planning
- Prevent unwanted heating at non-target regions

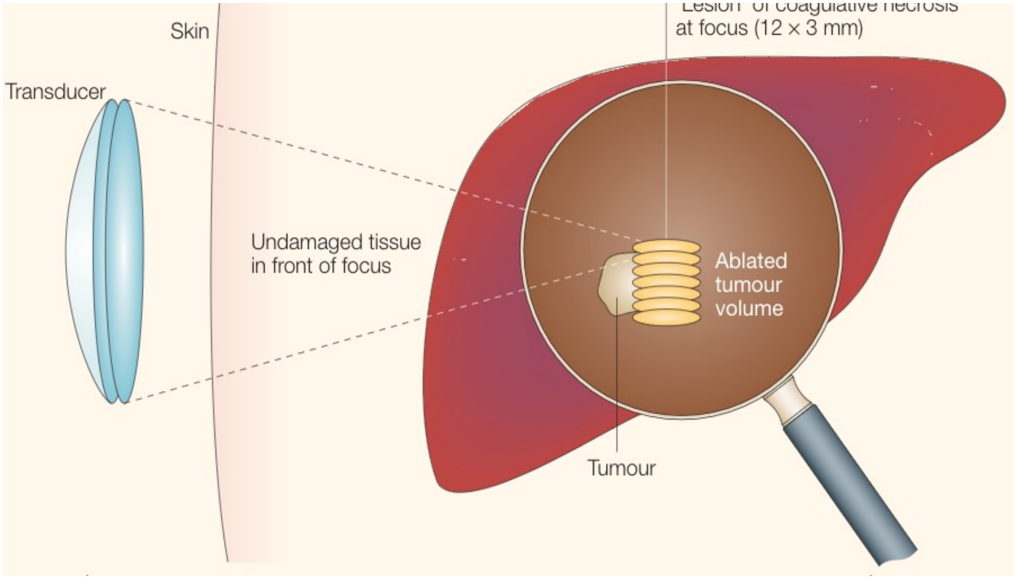
Why simulate?



†Drug delivery



†Neuromodulation



⁕Thermoablation

- Enable patient specific treatment planning
- Prevent unwanted heating at non-target regions

Model equation

$$\frac{1}{\rho_0 c_0^2}\frac{\partial^2 p}{\partial t^2} - \frac{1}{\rho_0}\nabla^2 p = \frac{\delta}{\rho_0 c_0^2}\nabla^2 \frac{\partial p}{\partial t} \quad \text{in } \Omega \times (0,T)$$

- Viscoelastic wave equation
- The model is typically used for low-intensity ultrasound application such as in transcranial focused ultrasound application

Source boundary condition

$$\frac{1}{\rho_0 c_0^2}\frac{\partial^2 p}{\partial t^2} - \frac{1}{\rho_0}\nabla^2 p = \frac{\delta}{\rho_0 c_0^2}\nabla^2 \frac{\partial p}{\partial t} \quad \text{in } \Omega \times (0,T)$$

$$\nabla p \cdot \mathbf{n} + \frac{1}{c_0} p_t = g(t) \quad \text{on } \Gamma_s \times (0,T)$$

- Pressure wave generated by the ultrasound transducer

Absorbing boundary condition

$$\frac{1}{\rho_0 c_0^2}\frac{\partial^2 p}{\partial t^2} - \frac{1}{\rho_0}\nabla^2 p = \frac{\delta}{\rho_0 c_0^2}\nabla^2 \frac{\partial p}{\partial t} \quad \text{in } \Omega \times (0, T)$$

$$\nabla p \cdot \mathbf{n} + \frac{1}{c_0} p_t = g(t) \quad \text{on } \Gamma_s \times (0, T)$$

$$\nabla p \cdot \mathbf{n} + \frac{1}{c_0} p_t = 0 \quad \text{on } \Gamma \times (0, T)$$

- Absorbing boundary condition to absorb outgoing waves

Initial-boundary value problem

$$\frac{1}{\rho_0 c_0^2}\frac{\partial^2 p}{\partial t^2} - \frac{1}{\rho_0}\nabla^2 p = \frac{\delta}{\rho_0 c_0^2}\nabla^2\frac{\partial p}{\partial t} \quad \text{in } \Omega \times (0,T)$$

$$\nabla p \cdot \mathbf{n} + \frac{1}{c_0}p_t = g(t) \quad \text{on } \Gamma_s \times (0,T)$$

$$\nabla p \cdot \mathbf{n} + \frac{1}{c_0}p_t = 0 \quad \text{on } \Gamma \times (0,T)$$

$$p(\boldsymbol{x},0) = 0 \quad \text{for } \boldsymbol{x} \in \Omega$$

$$p_t(\boldsymbol{x},0) = 0 \quad \text{for } \boldsymbol{x} \in \Omega$$

Semi-discrete equation

$$\frac{1}{\rho_0 c_0^2}\mathbf{M}_\Omega\ddot{\mathbf{p}} + \frac{\delta}{\rho_0 c_0^3}\mathbf{M}_\Gamma\ddot{\mathbf{p}} = -\frac{1}{\rho_0}\mathbf{K}_\Omega\mathbf{p} + \mathbf{l}_1 - \frac{1}{\rho_0 c_0}\mathbf{M}_\Gamma\dot{\mathbf{p}} - \frac{\delta}{\rho_0 c_0^2}\mathbf{K}_\Omega\dot{\mathbf{p}} + \mathbf{l}_2$$

Runge-Kutta method

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\Omega + \mathbf{M}_\Gamma \end{bmatrix} \begin{pmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K}_\Omega & -\mathbf{M}_\Gamma - \mathbf{K}_\Omega \end{bmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{l}_1 + \mathbf{l}_2 \end{pmatrix}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \sum_{i=1}^{s} b_i \dot{\mathbf{u}}_i$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \sum_{i=1}^{s} b_i \dot{\mathbf{v}}_i$$

Solver design

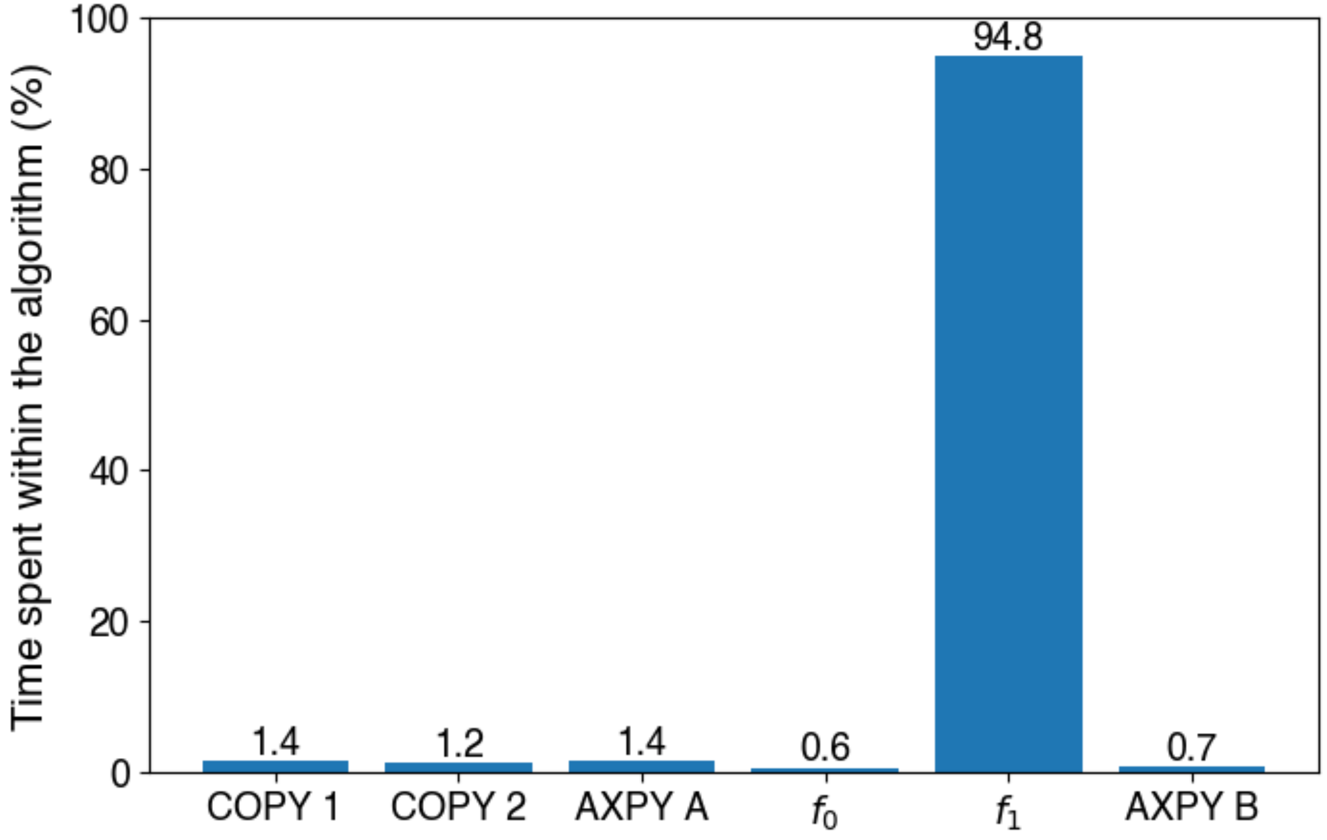
- Fully hexahedral mesh
- High order GLL-based Lagrange finite element basis function
- Numerical integration is performed using GLL quadrature
- Mass lumping – diagonal mass matrix
- Matrix-free method
- 4^{th} order explicit Runge-Kutta method
- Implemented using the FEniCSx open-source software

Solver algorithm

```
while (t < tf) {

    // COPY 1
    copy(*u_, *u0);
    copy(*v_, *v0);

    for (int i = 0; i < n_RK; i++) {

        // COPY 2
        copy(*u0, *un);
        copy(*v0, *vn);

        // AXPY A
        axpy(*un, dt * a_runge[i], *udot, *un);
        axpy(*vn, dt * a_runge[i], *vdot, *vn);

        // RK time evaluation
        tn = t + c_runge[i] * dt;

        // Solve for udot and vdot
        f0(tn, un, vn, udot);
        f1(tn, un, vn, vdot);

        // AXPY B
        axpy(*u_, dt * b_runge[i], *udot, *u_);
        axpy(*v_, dt * b_runge[i], *vdot, *v_);
    }

    // Update time
    t += dt;
}
```

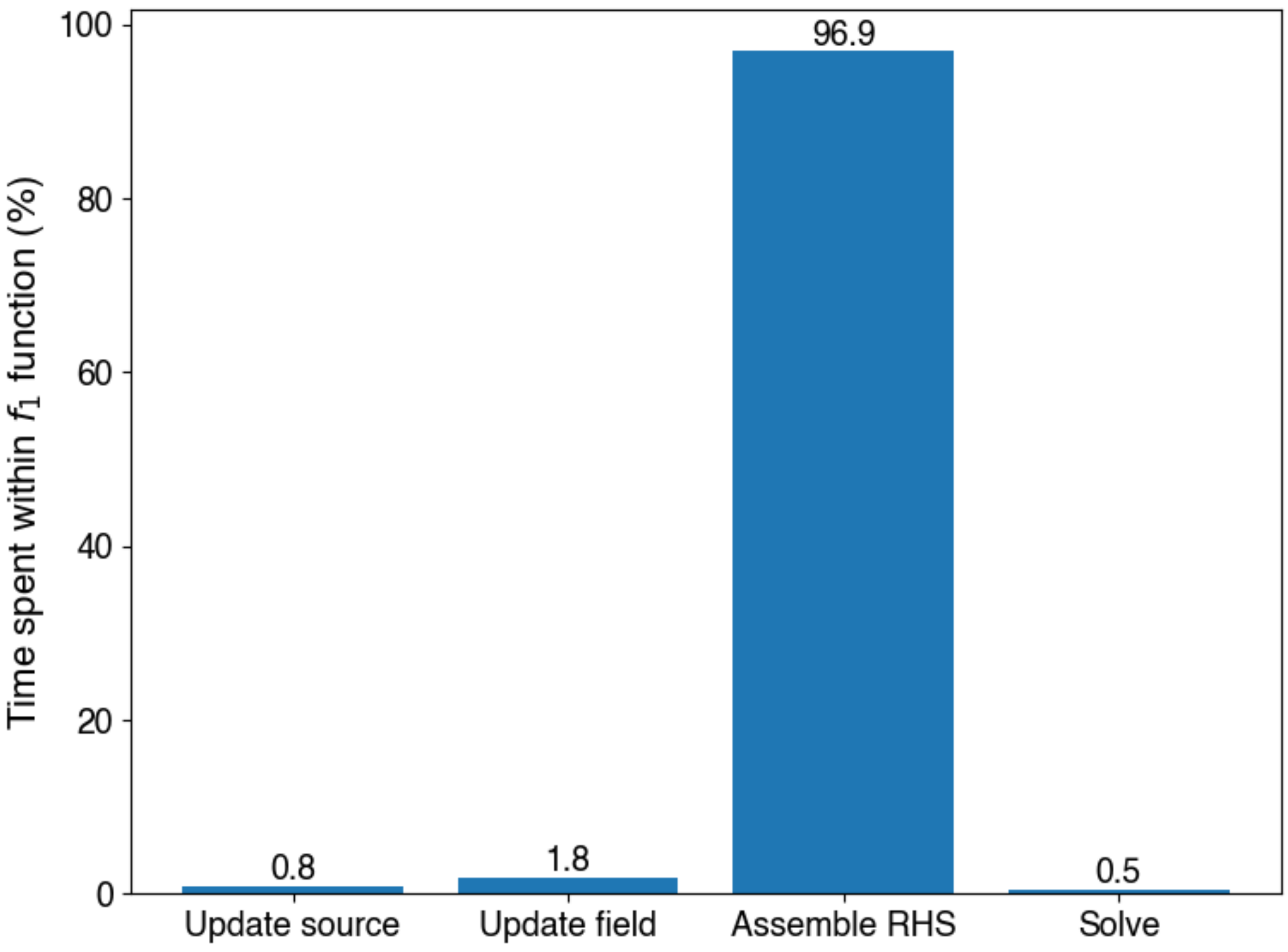
$$f_0 \Rightarrow \dot{\mathbf{u}} = \mathbf{v}$$

$$f_1 \Rightarrow \dot{\mathbf{v}} = (\mathbf{M}_\Omega + \mathbf{M}_\Gamma)^{-1}(-\mathbf{K}_\Omega \mathbf{u} - (\mathbf{M}_\Gamma + \mathbf{K}_\Omega)\mathbf{v} + \mathbf{l}_1 + \mathbf{l}_2)$$

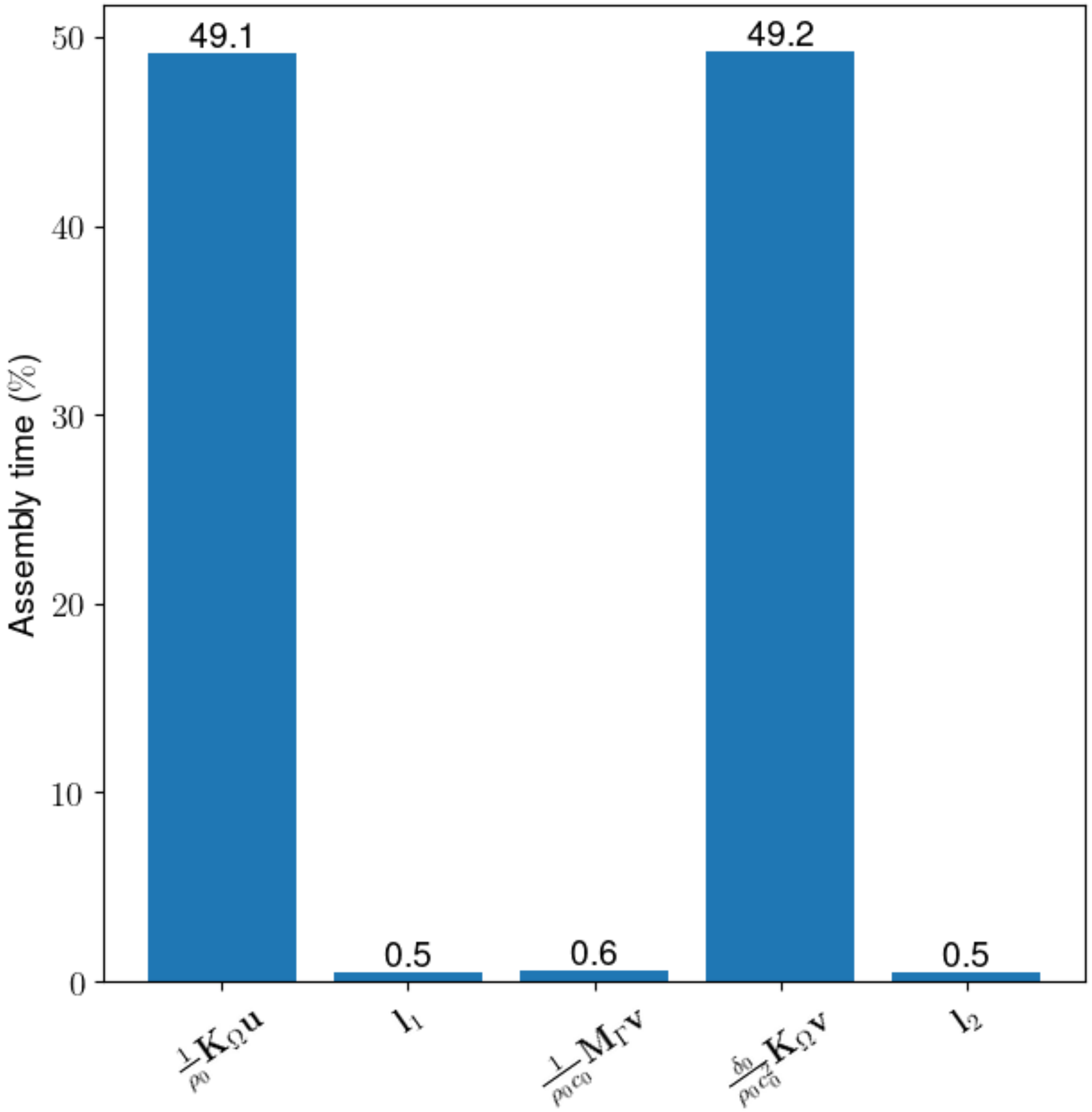


f_1 function

```
// Update source
fill(g_.begin(), g_.end(), window * 2.0 * p0 * w0 / s0 * cos(w0 * t));
fill(dg_.begin(), dg_.end(), dwindow * 2.0 * p0 * w0 / s0 * cos(w0 * t)
                  - window * 2.0 * p0 * w0 * w0 / s0 * sin(w0 * t));

// Update fields
u->scatter_fwd();
copy(*u, *u_n->x());

v->scatter_fwd();
copy(*v, *v_n->x());

// Assemble RHS
fill(b_.begin(), b_.end(), 0.0);
assemble_vector(b_, *L);
b->scatter_rev(plus());

// Solve
{
  // Element wise division
  // out[i] = b[i]/m[i]
  transform(_b.begin(), _b.end(), _m.begin(), out.begin(),
            [](const T& bi, const T& mi) { return bi / mi; });
}
```

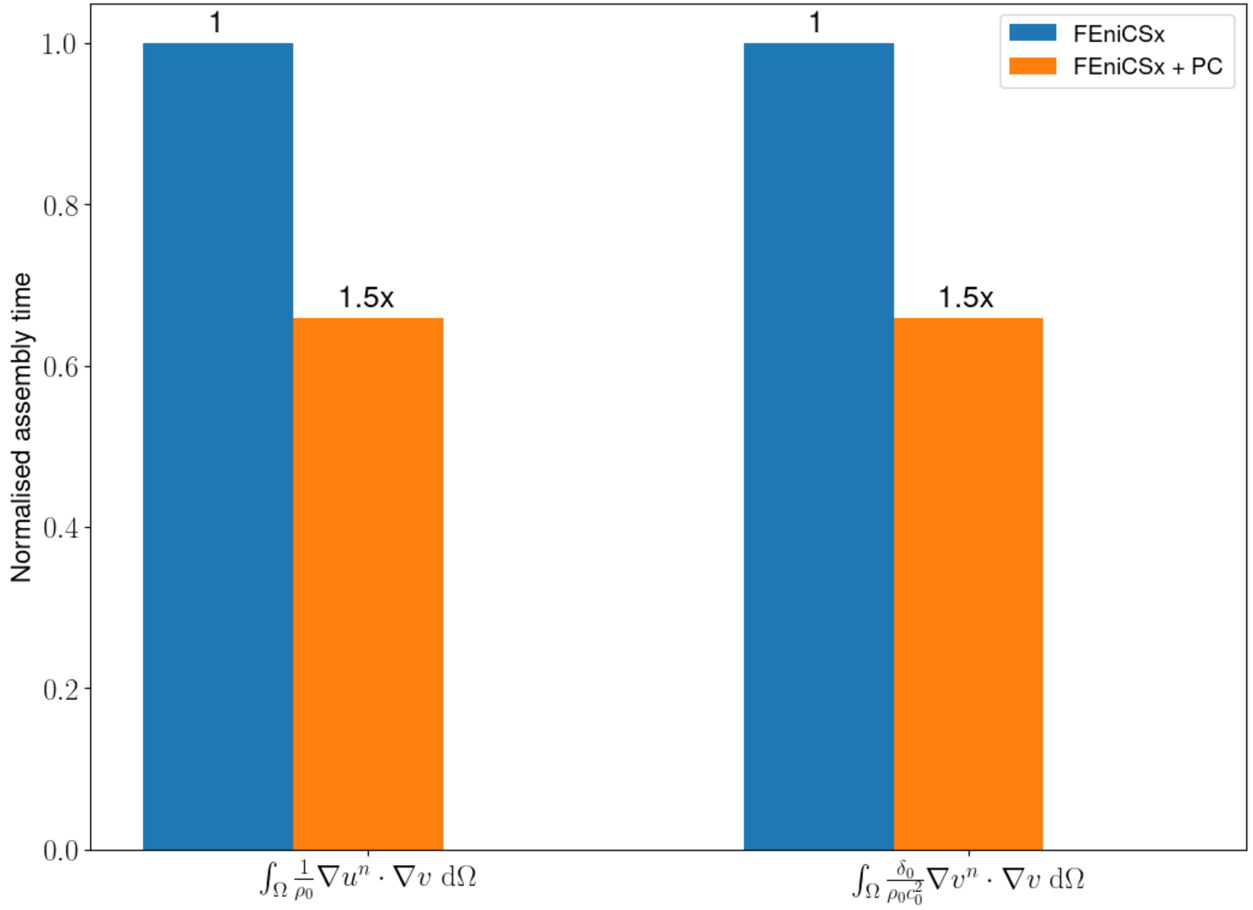


Vector assembly



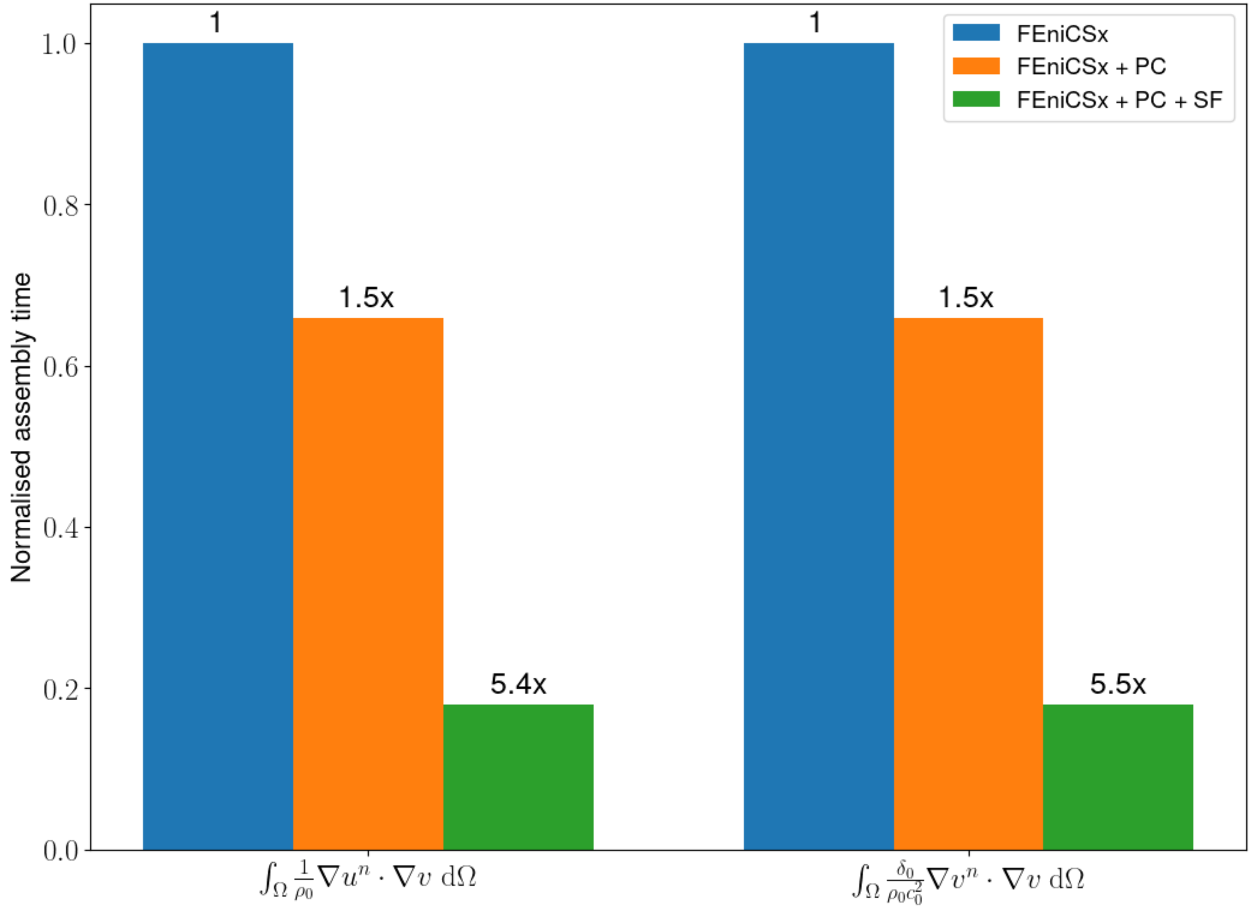
- A high percentage of time is spent on the vector assembly of the cell-wise operators.
- The action of the stiffness operator constitutes the highest percentage of time for vector assembly.

Vector assembly – speed-up



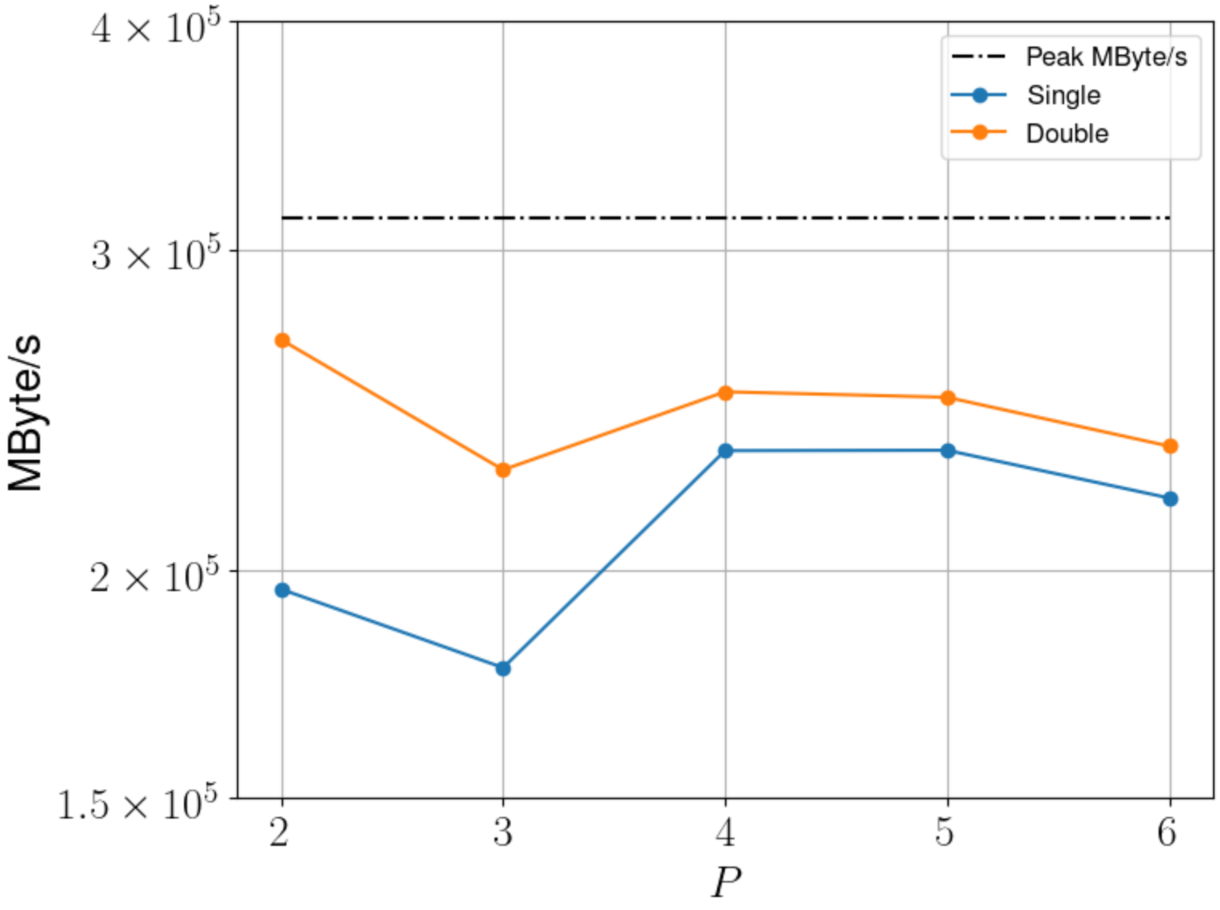
- Precomputed geometric data implementation gives 1.5 times speed-up

Vector assembly – speed-up



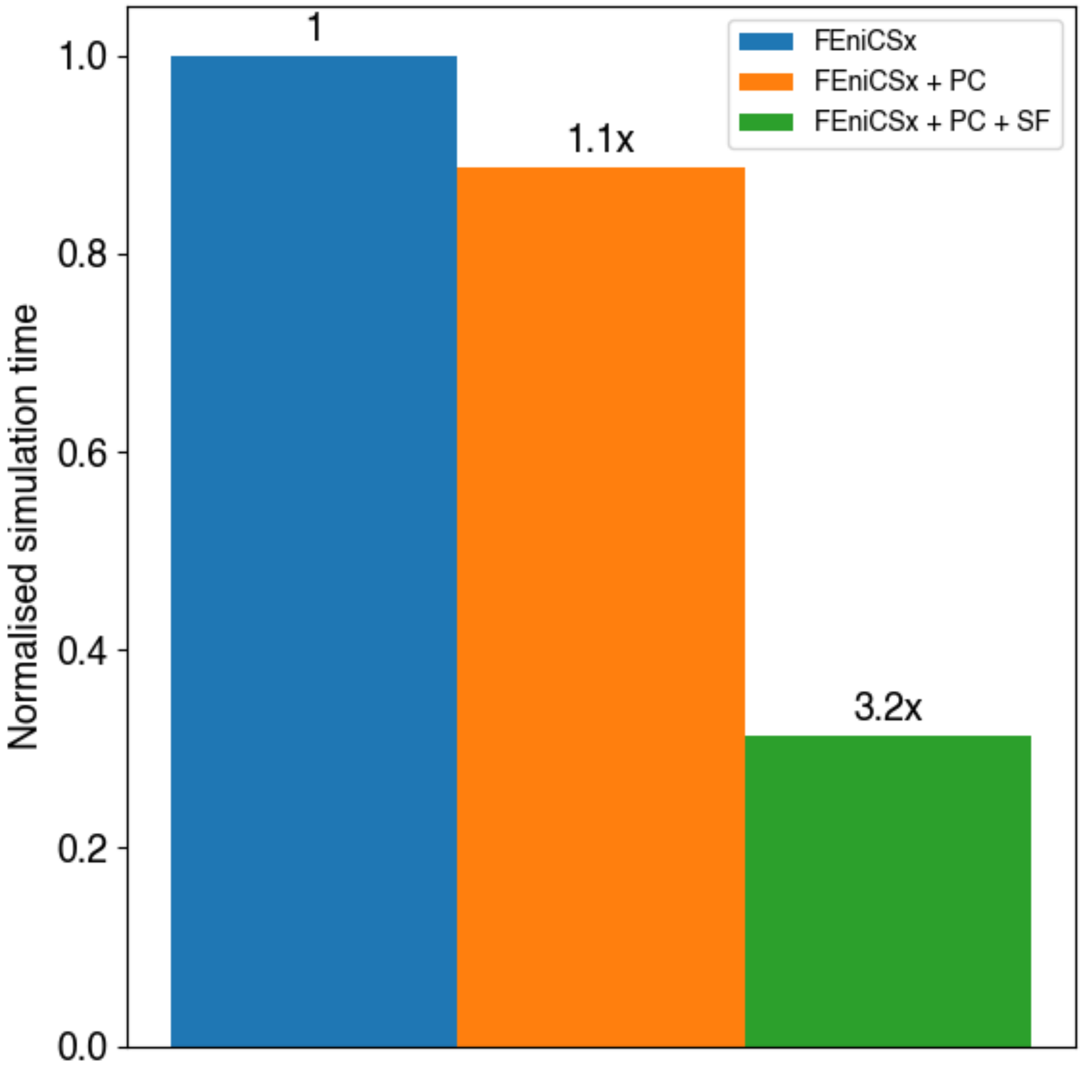
- Precomputed geometric data implementation gives 1.5 times speed-up
- Adding sum-factorization implementation gives approximately 5.5 times speed-up

Fraction of peak performance



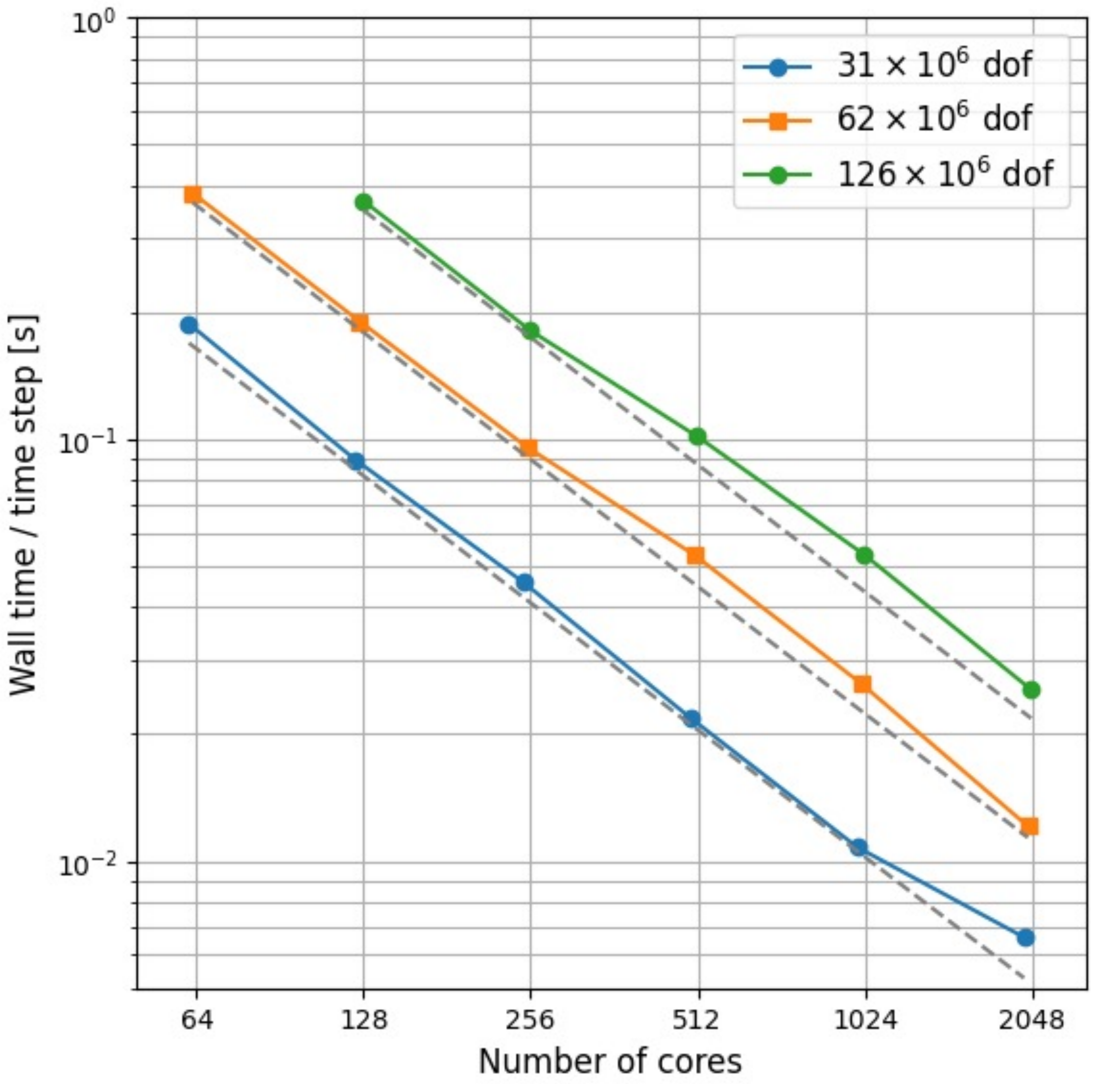
- Experiment was performed on the Intel Icelake CPU.
- The stiffness operator achieve a good fraction of peak performance in terms of the memory bandwidth.
- Between 50% – 90% of peak performance.

Solver simulation time – speed-up



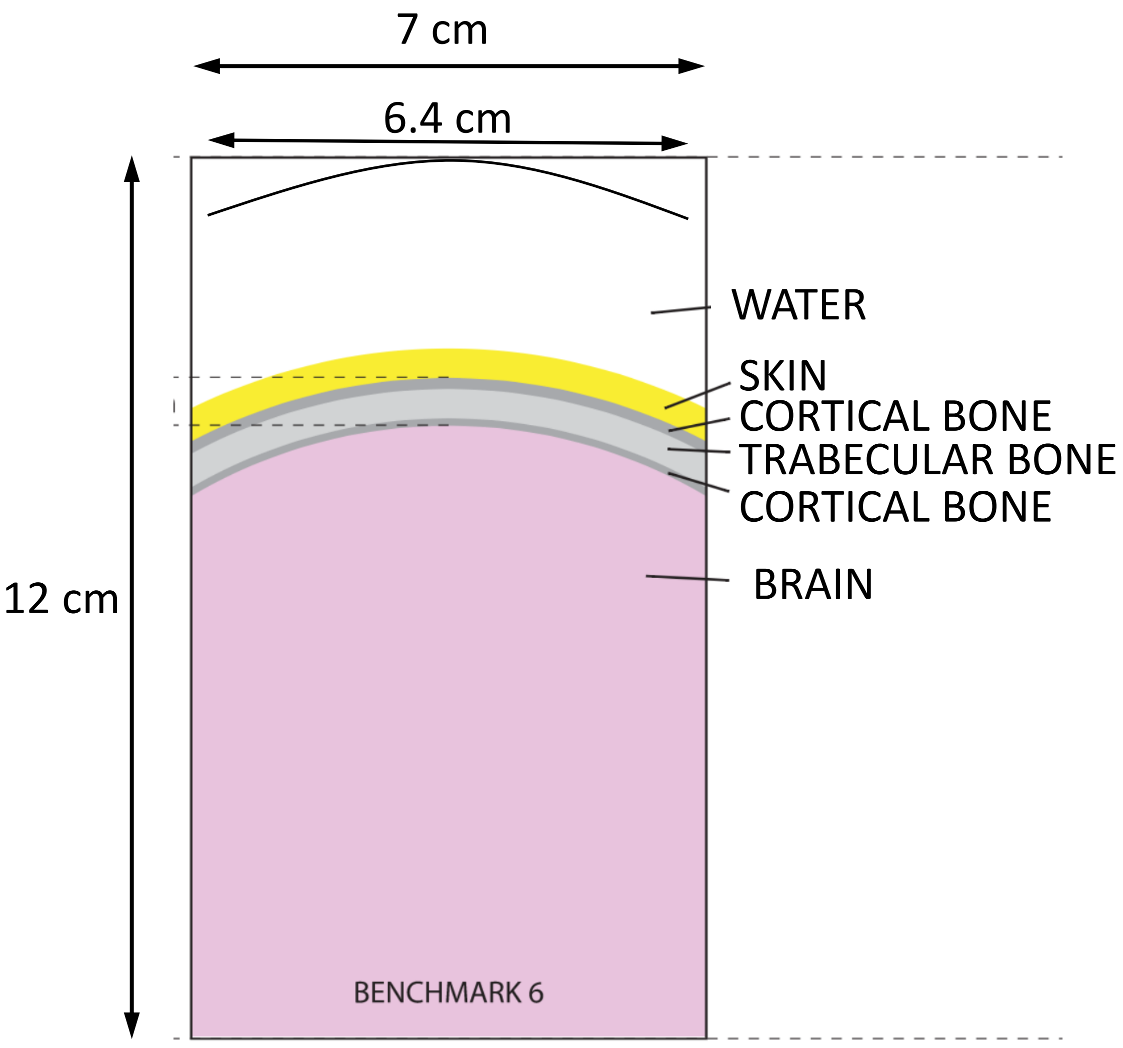
- The simulation time using the precomputed geometric data achieves a 1.1 times speed-up.
- The operator with sum-factorization achieves 3.2 times speed-up of total simulation time.

Strong scaling



- Doubling the number of processes half the time required for simulation
- The solver shows good strong scaling capability

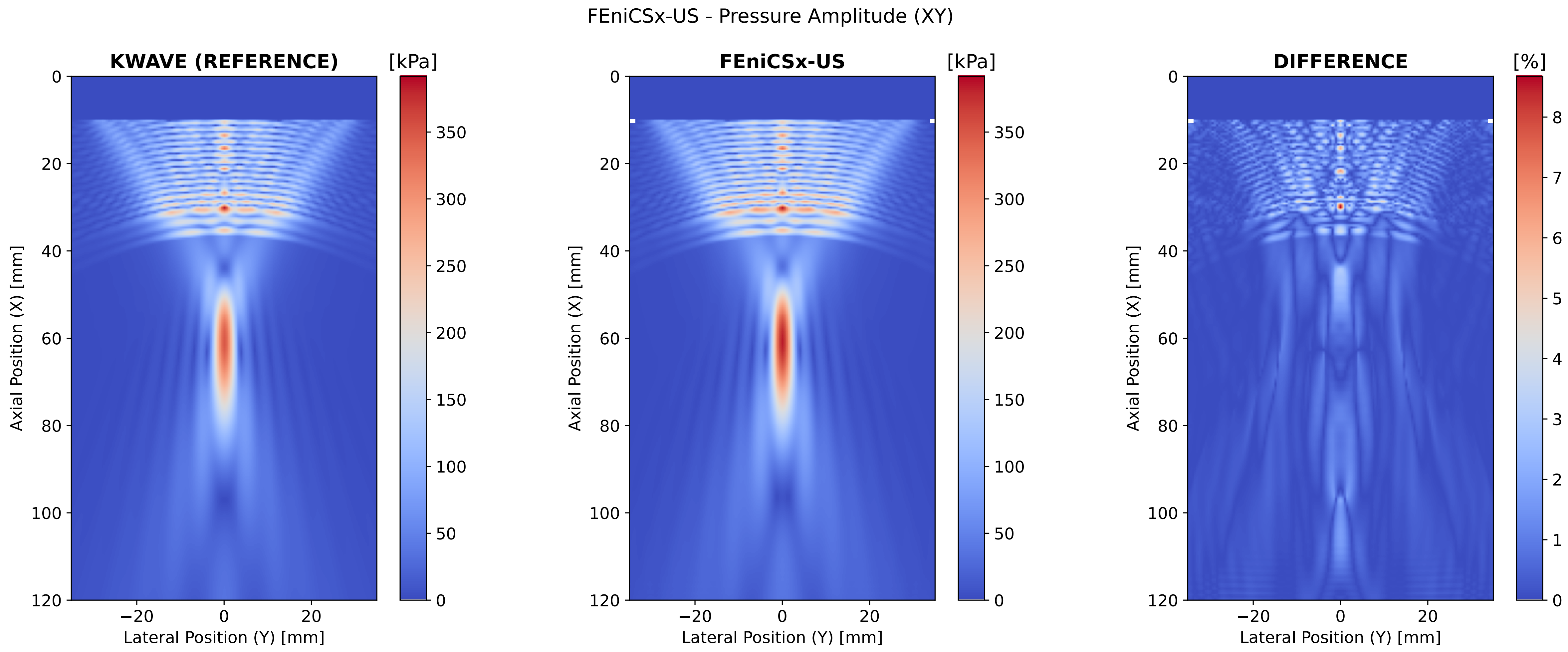
Solver verification



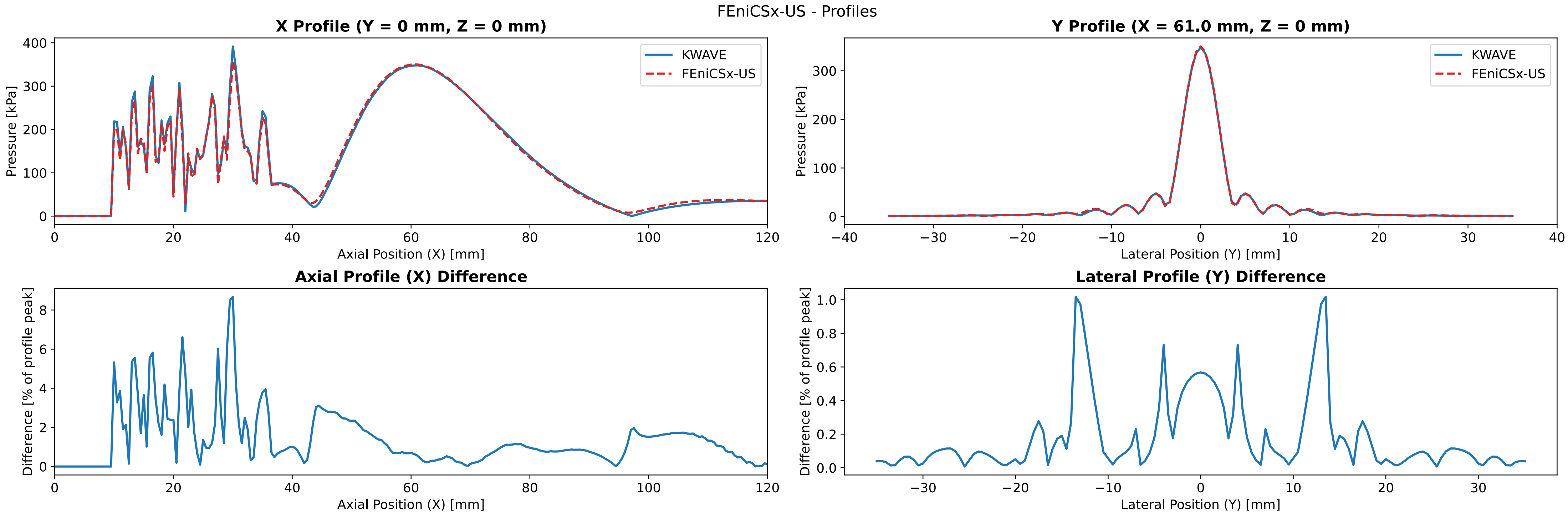
Aubry et. al. (2022)

- A 3D transcranial ultrasound simulation
  - Spherical curved transducer – focal length of 64 mm
  - Transducer amplitude = 60 kPa
  - Transducer frequency = 500 kHz
- Degrees of freedom: 70 x 10^6
- Number of time steps: 3.4 x 10^4
- The simulation took 1.5 hours using 256 processes on Intel Skylake CPU.
- The simulation was run using double-precision floating-point type.

Solution comparison with k-Wave solver



Solution comparison with k-Wave solver



Summary

- We have implemented a high-order matrix-free finite element solver for focused ultrasound application.
- The cell-wise operators achieve an excellent fraction of peak performance.
- The solver shows excellent parallel scalability through strong scaling.
- The solver is capable to handle realistic transducer shape, domain heterogeneity as well as geometrically *complex* scatterer shape.

Outlook

- Nonlinear model equation – Westervelt equation
  - Important for modeling high-intensity ultrasound application
  - The main motivation of solving in the acoustic wave equation in the time-domain
- Heterogenous computing
  - GPU implementation of the solver

Acknowledgement

Registration and travel support for this presentation was provided by the Society for Industrial and Applied Mathematics