KATHOLIEKE UNIVERSITEIT
LEUVEN

Nonlinear Real-time Finite Element Analysis using CUDA

Vukasin Strbac

Biomechanics section

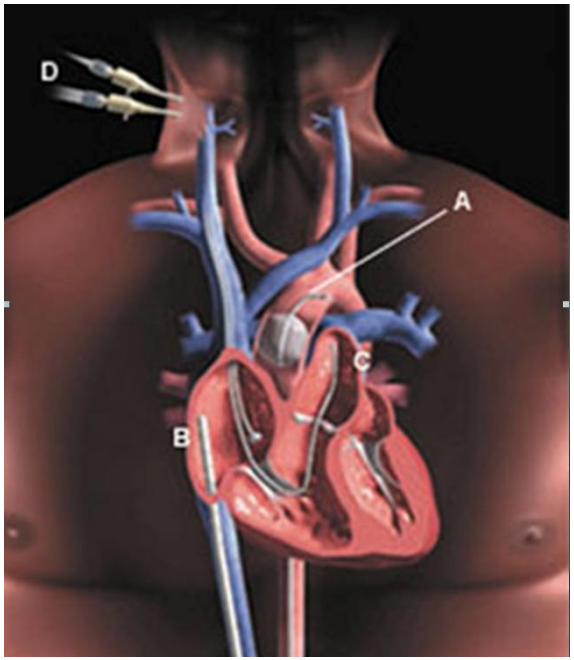
KU Leuven

- Introduction
  - Biomechanics & surgery application
  - What are we trying to do?
  - Why is FE relevant to us?
- FE and parallelization, TLED
  - Which formulation is better and why?
  - Basic Nonlinear Implicit
  - Total Lagrangian Explicit Dynamic
- Results, use case
- Conclusion

Intro

- Biomechanics
  - Help DURING surgery, need fast computation of stresses/damage
    - Can't influence surgery workflow
    - tool: FE, generally very slow– we use CUDA

- Finite Element Analysis
  - Speed and accuracy is what we are going for.
  - Not "embarrassingly parallel"



Heartport Endoarctic clamp, Redwood, USA

Time integration:

Implicit

$$Ku = R$$

Explicit

$$M\ddot{u} + c_d M\dot{u} + F(u) = R$$

+ e.g. central difference method

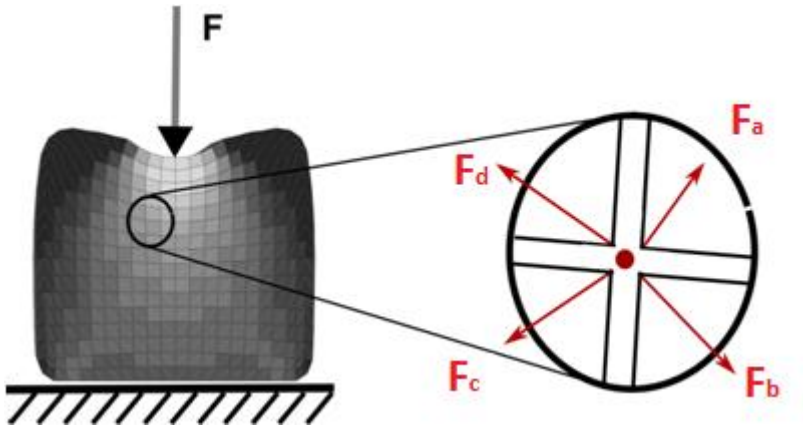
$${}^{t+\Delta t}\boldsymbol{u}_e = a({}^{t}\boldsymbol{R}_e - {}^{t}\boldsymbol{F}_e) + b\,{}^{t}\boldsymbol{u}_e - c\,{}^{t-\Delta t}\boldsymbol{u}_e$$

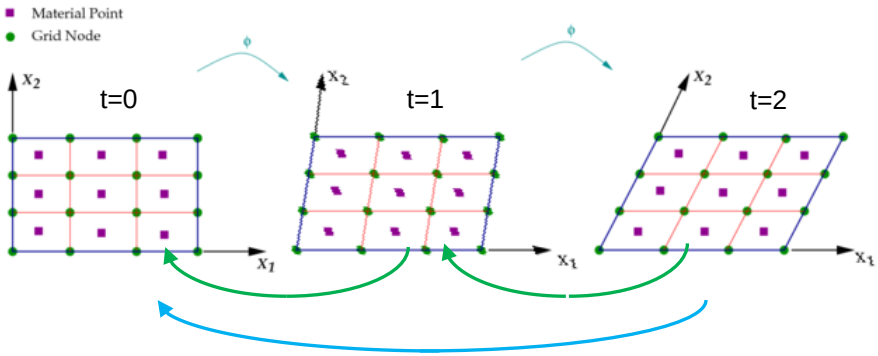
$$a = -\frac{2(\Delta t^2)}{2t\Delta t M_e} \qquad b = \frac{1+(2-q\Delta t)}{2+q\Delta t} \qquad c = \frac{2-q\Delta t}{2+q\Delta t}$$

To solve: Iterative schemes (Newton-Raphson)

Convergence: $(R - F) < tolerance$

Energy has stabilized





Total and Updated Lagrangian

Basic nonlinear implicit FE, example on elasticity

Implicit, nonlinear:

$set\ \nabla \mathbf{N}_{iso}$ ⟶ precomputed

$for\ t = 1 \ldots timesteps$

  $for\ i = 1..maxIterations$

    $for\ e = 1 \ldots elements\{$

      $construct\ {}^{e}\boldsymbol{D}$

      $\boldsymbol{J} = \nabla \mathbf{N}_{iso}({}^{e}\boldsymbol{d} + {}^{e}\boldsymbol{u})$ cheap

      $\nabla \mathrm{N}_{glo} = J^{-1} * \nabla \mathrm{N}_{iso}$ cheap

      $construct\ {}^{e}\boldsymbol{B}$

      $\boldsymbol{I} = {}^{e}\boldsymbol{B}^{T}\,{}^{e}\boldsymbol{D}\,{}^{e}\boldsymbol{B} * |J|$ cheap

      ${}^{e}K$ =(Gauss)*I

    }

    $K = \sum {}^{e}K$ cheap

    Impose B.C.s cheap

    $\Delta \boldsymbol{u}_i = \boldsymbol{K}^{-1}(\boldsymbol{R} - \boldsymbol{F}_{i-1})$ expensive

    $\boldsymbol{u}_i = \boldsymbol{u}_{i-1} + \boldsymbol{\Delta u}_i$ cheap

    $\boldsymbol{F}_i = \boldsymbol{K}_{i-1} * \boldsymbol{u}_i$ cheap

    $Check(\boldsymbol{R} - \boldsymbol{F}_i)$ and continue/exit

  }

}

Newton-Raphson iteration

- Pros:

less timesteps
unconditionally stable
insensitive to incompressbility

- Cons:

Large matrix inversion
More memory intensive
More computationally intensive
Little is precomputed

*D=stress/strain matrix
*B=strain/displacement matrix

Total Lagrangian Explicit Dynamic

set $\nabla \mathbf{N}_{iso}$
Loop over elements{
$^{e}J = \nabla \mathbf{N}_{iso}\, ^{e}\boldsymbol{u}$
$\nabla \mathbf{N}_{glo} = J^{-1} \nabla \mathbf{N}_{iso}$
construct $^{e}\boldsymbol{B}$
}
precompute();

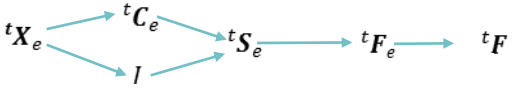
$$a = -\frac{2(\Delta t^2)}{2t\Delta t M_e}$$
$$b = \frac{1+(2-q\Delta t)}{2+q\Delta t}$$
$$c = \frac{2-q\Delta t}{2+q\Delta t}$$

construct (diagonalized) mass matrix M
Loop over timesteps t {
Loop over elements e {
$^{t}\boldsymbol{X}_e = \boldsymbol{I} + \nabla \boldsymbol{N}_e\, ^{t}\boldsymbol{u}_e$
$^{t}\boldsymbol{C}_e = {}^{t}\boldsymbol{X}_e{}^{T}\, {}^{t}\boldsymbol{X}_e$
computeForces();
$J = \det(\boldsymbol{X}_e)$
$$^{t}\boldsymbol{S}_e = (1-d)\left[\mu J^{-1}\left(\mathbf{I} - \frac{\mathrm{tr}(^{t}\boldsymbol{C}_e)}{2}\, {}^{t}\boldsymbol{C}_e{}^{-1}\right) + \kappa J(J-1)\, {}^{t}\boldsymbol{C}_e{}^{-1}\right]$$
$^{t}\boldsymbol{F}_e = \nabla \boldsymbol{N}_e * {}^{t}\boldsymbol{S}_e * {}^{t}\boldsymbol{X}_e *$ weight(e)
end element loop
$^{t}\boldsymbol{F} = \sum {}^{t}\boldsymbol{F}_e$
$^{t+\Delta t}\boldsymbol{u}_e = a(^{t}\boldsymbol{R}_e - {}^{t}\boldsymbol{F}_e) + b\, ^{t}\boldsymbol{u}_e - c\, ^{t-\Delta t}\boldsymbol{u}_e$
updateDisplacements();
$^{t+\Delta t}\boldsymbol{u}_e = \boldsymbol{d}(t+\Delta t)$ $^{t+\Delta t}\boldsymbol{F}_e = \boldsymbol{F}(t+\Delta t)$
doLoading();
}

- Pros:
  lots of precomputation
  no K matrix assembly
  no K inversion
  elements “independent”
  much less memory required
- Cons:
  many more timesteps
  addit. time to converge
  conditionally stable
  sensitive to incompressibility

*neo-hookean material
*d is damage

TLED serial, iteration phase



Get forces()

8

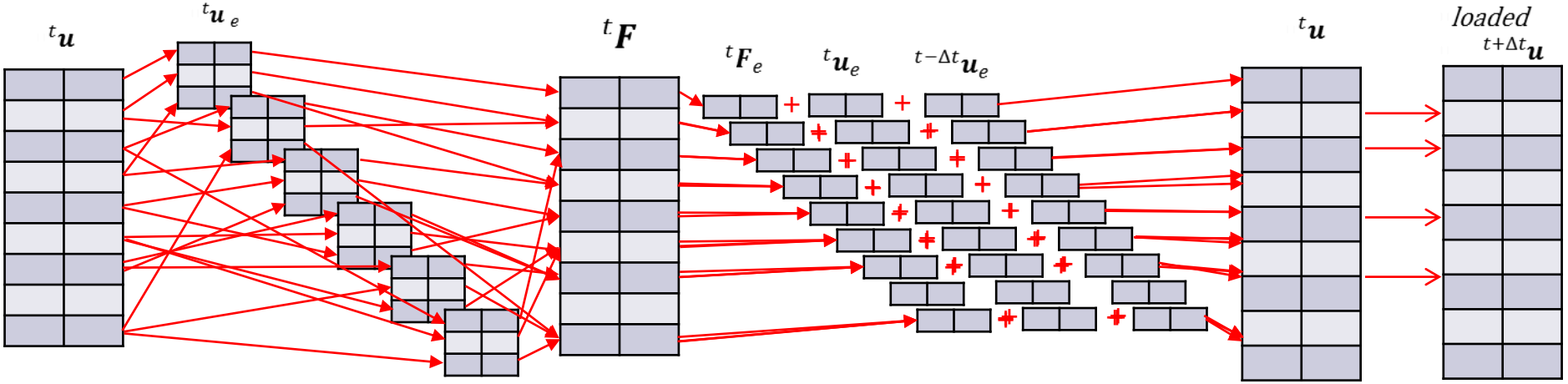
$${}^{t+\Delta t}u_e = a({}^tR_e - {}^tF_e) + b\,{}^tu_e - c\,{}^{t-\Delta t}u_e$$

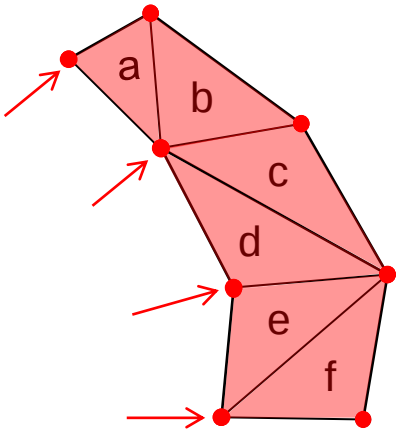
Update Displacements()

9

$${}^{t+\Delta t}u_e = d(t+\Delta t)$$

doDdisplacementLoading()

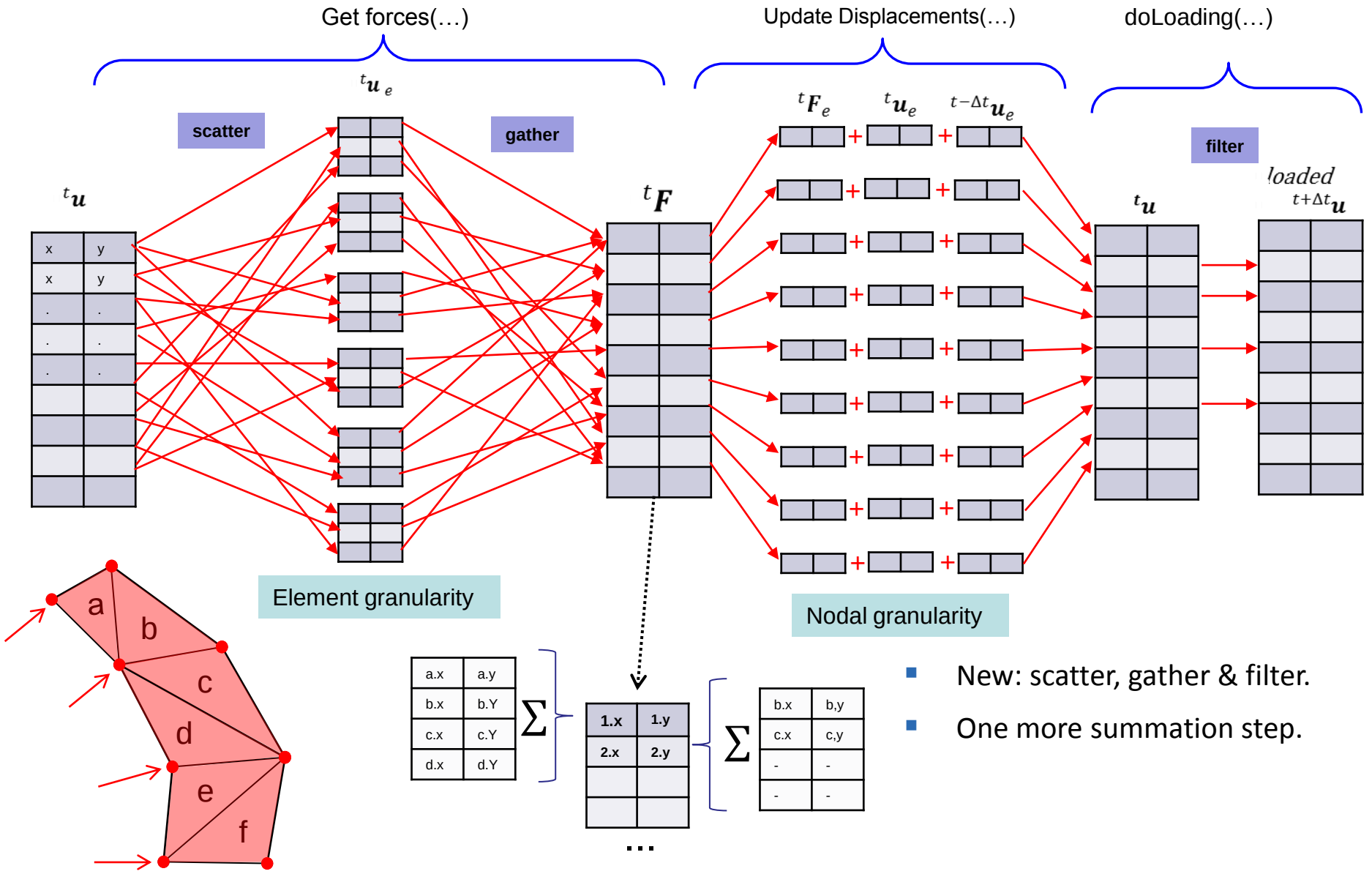




1. ${}^tX_e = I + \nabla N_e\,{}^tu_e$
2. ${}^tC_e = {}^tX_e{}^T\,{}^tX_e$
3. $J = \det(X_e)$
4. $d = \gamma(1 - e^{-\frac{\beta}{m}})$
5. ${}^tS_e = (1-d)\left[\mu J^{-1}\left(I - \frac{\mathrm{tr}({}^tC_e)}{2}{}^tC_e{}^{-1}\right) + \kappa J(J-1){}^tC_e{}^{-1}\right]$
6. ${}^tF_e = \nabla N_e * {}^tS_e * {}^tX_e * \mathrm{weight(e)}$
7. ${}^tF = \sum {}^tF_e$

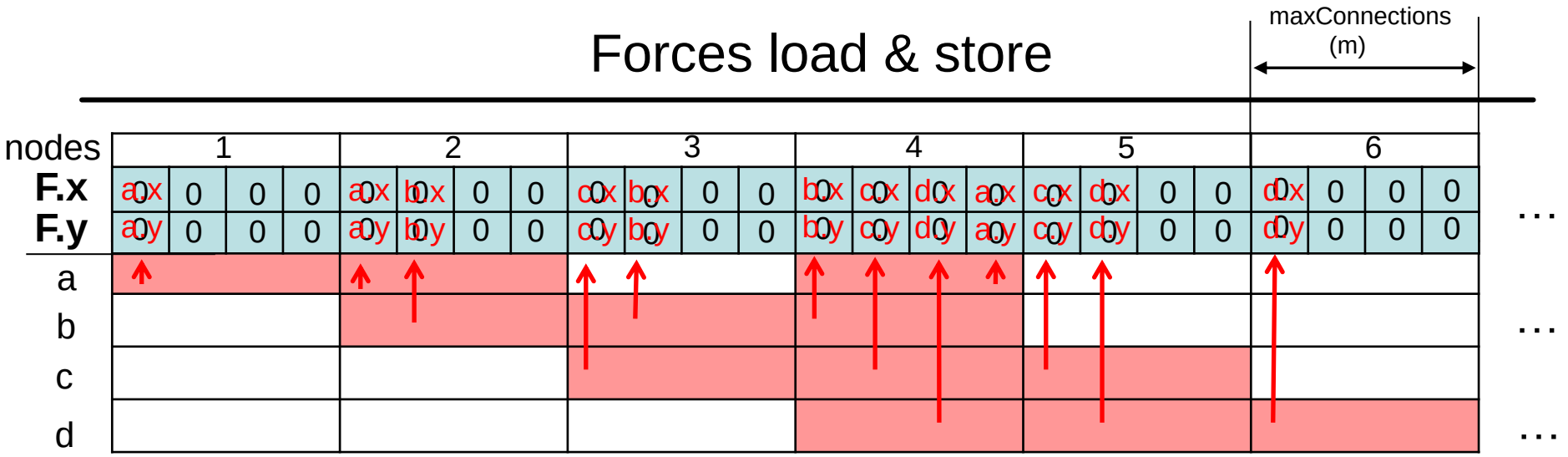
○ = precomputed

TLED parallel, iteration phase



- New: scatter, gather & filter.
- One more summation step.

Forces load & store

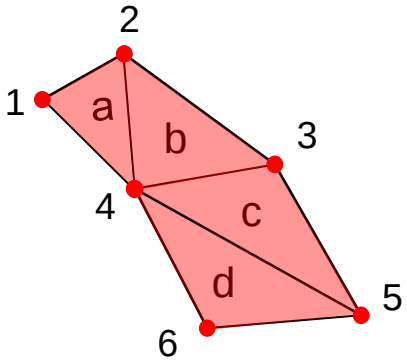


a
…
→ F[4 * 1 + i [me].x] = a;
→ F[4 * 2 + i [me].y] = a;
→ F[4 * 4 + i [me].z] = a;
…

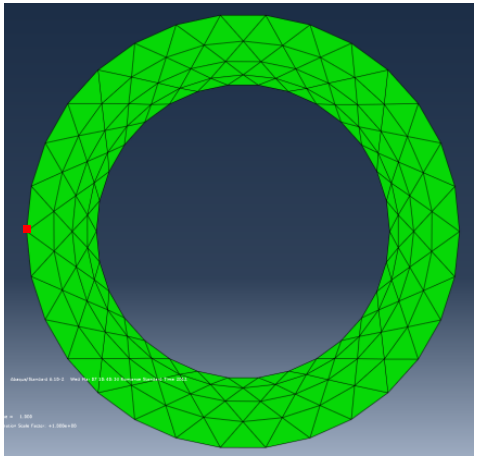
b
…
→ F[4 * 4+ i [me].x] = b;
→ F[4 * 2+ i [me].y] = b;
→ F[4 * 3+ i [me].z] = b;
…

c
…
→ F[4 * 4+ i [me].x] = c;
→ F[4 * 3 + i [me].y] = c;
→ F[4 * 5 + i [me].z] = c;
…

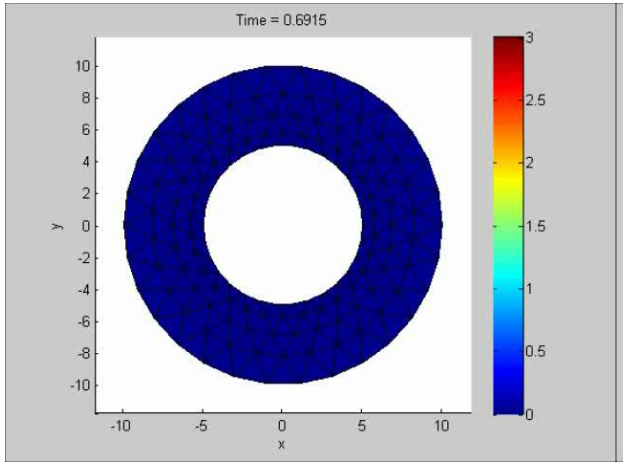
d
…
→ F[4 * 6 + i [me].x] = d;
→ F[4 * 4 + i [me].y] = d;
→ F[4 * 4 + i [me].z] = d;
…



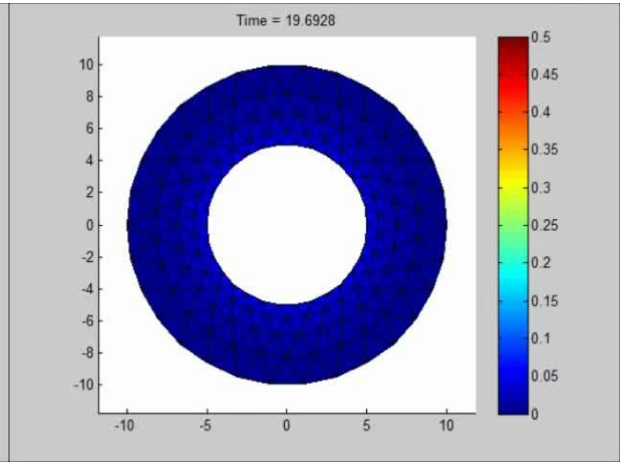

```
Force[m * myNode [1] + index[me].x] = f1;
Force[m * myNode [2] + index[me].y] = f2;
Force[m * myNode [3] + index[me].z] = f3;
```



Abaqus/Standard solution



Cuda/TLED: displacement (mm)



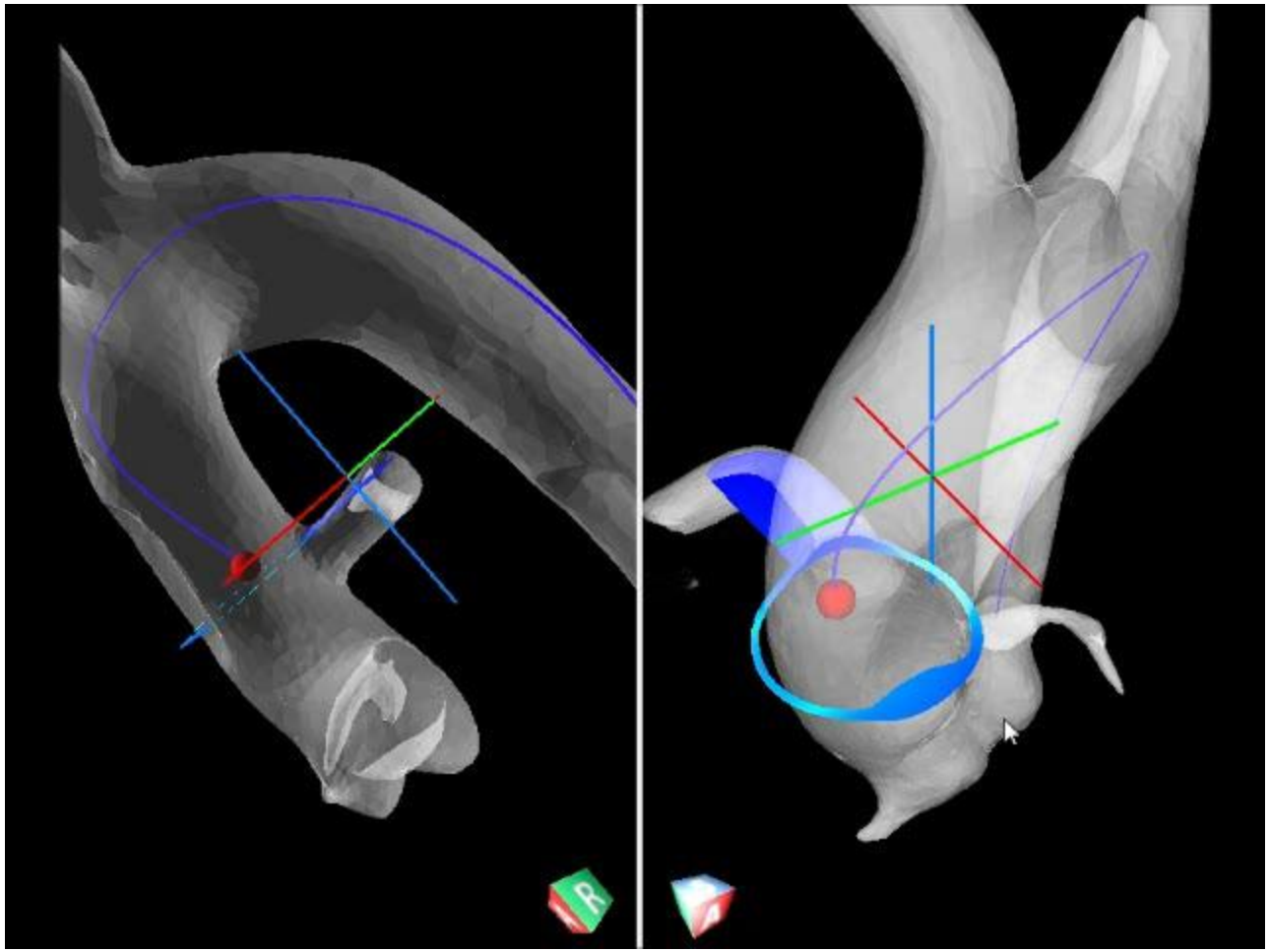
damage(-)

Parameter	Value
μ	1006.7 MPa
κ	50e3 MPa
γ	0.9 [-]
m	1000 MPa

Method	Material Model	Displacement of node x[mm]	% deviation	Calculation time
Abaqus explicit	Neo Hookean Damage	1.8135 1.8119		4min 4.6min
Abaqus implicit	Neo Hookean Damage	1.8125 1.8109		400ms 500ms
CUDA TLED	Neo Hookean Damage	1.8163 1.8147	0.21% 0.21%	81ms 87ms

*xeon e6520 vs Tesla C2075(in single precision)

Code in action: human aorta model + slices



- Mimicking endoclamp balloon expansion.
- Loading is 0-500mmHg sine wave.
- Simulation at 14 fps ~70ms per solution, 1000 timesteps each.
- Material is Neo-Hookean with E=3000Pa, v=0,49.
- Small enough to fit all element/node data into registers.

Thanks!

TLED – Precomputation phase

- Initialize and compute
  - Mass matrix
  - Shape function derivatives
  - **CDM coefficients**

$$ {}^{t}\dot{\boldsymbol{u}} = \frac{1}{2\Delta t}({}^{t+\Delta t}\boldsymbol{u} - {}^{t-\Delta t}\boldsymbol{u}) $$

$$ {}^{t}\ddot{\boldsymbol{u}} = \frac{1}{2\Delta t^2}({}^{t-\Delta t}\boldsymbol{u} - 2{}^{t}\boldsymbol{u} + {}^{t+\Delta t}\boldsymbol{u}) $$

$$ \boldsymbol{M}\,{}^{t}\ddot{\boldsymbol{u}} + \boldsymbol{K}({}^{t}\boldsymbol{u})\,{}^{t}\boldsymbol{u} = {}^{t}\boldsymbol{R} $$

$$ {}^{t}\boldsymbol{F} $$

- Explicit integration scheme
  - Central Difference Method
  - No need for optimization/iterative methods
  - Covers nonlinear behavior in an intuitive way – given small enough timestep is used.

$$ {}^{t+\Delta t}\boldsymbol{u} = \frac{\Delta t^2}{\boldsymbol{M}}({}^{t}\boldsymbol{R} - {}^{t}\boldsymbol{F}) + 2{}^{t}\boldsymbol{u} - {}^{t-\Delta t}\boldsymbol{u} $$

diagonalized

$$ {}^{t+\Delta t}\boldsymbol{u}_e = \frac{\Delta t^2}{\boldsymbol{m}_e}({}^{t}\boldsymbol{R}_e - {}^{t}\boldsymbol{F}_e) + 2{}^{t}\boldsymbol{u}_e - {}^{t-\Delta t}\boldsymbol{u}_e $$

$$ a = -\frac{2(\Delta t^2)}{2t\Delta t M_e}; $$

$$ b = \frac{1 + (2 - q\Delta t)}{2 + q\Delta t} $$

$$ c = -\frac{2 - q\Delta t}{2 + q\Delta t}; $$

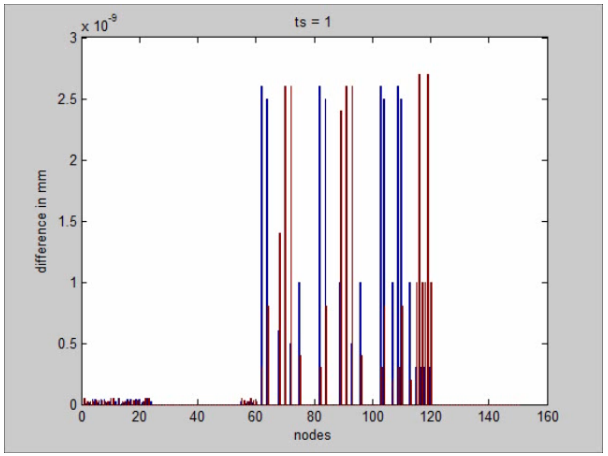
$$ q = \frac{2(1 - C_r{}^2)}{\Delta t(1 + C_r)} $$

- Add mass proportional damping according to The Dynamic Relaxation algorithm

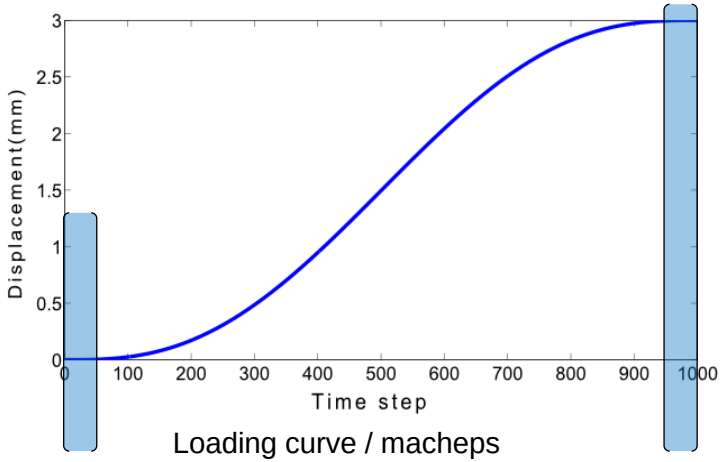
*CDM = central difference method

Accuracy considerations

- Single precision
- Some increments < macheps
- Total error after 1000ts <1mm = single precision warranted

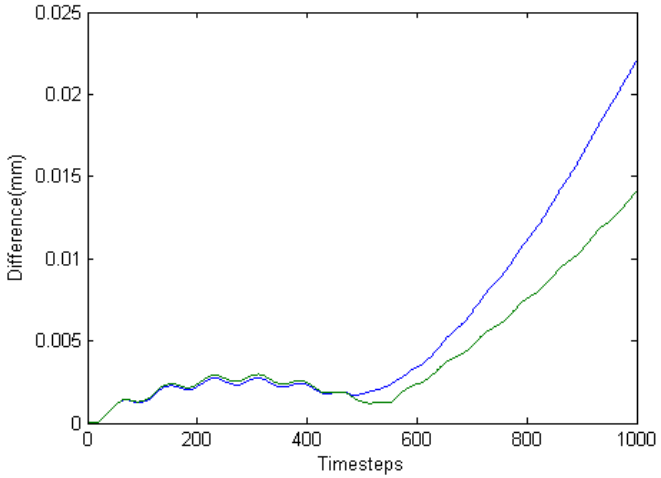


Accumulation of error(CUDA vs Matlab)



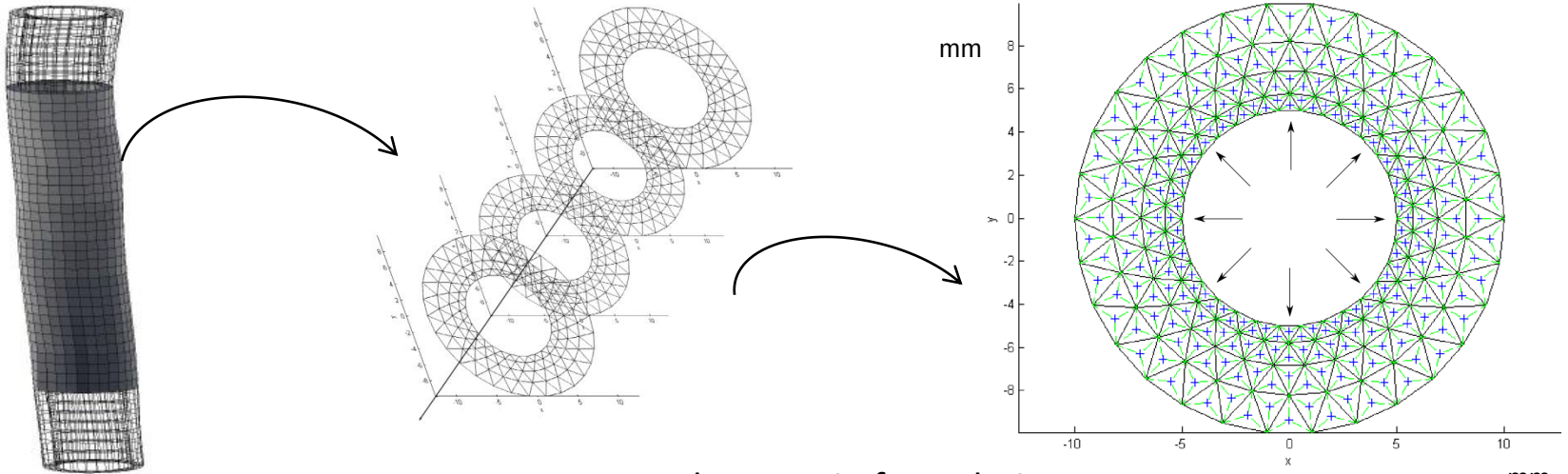
Loading curve / macheps

d=0 d=3



Evolution of error(single vs double)

Endoclamp case study



- Hyperelastic constitutive model
  - Neo-hookean matrix material of artery
  - + damage
- Plane strain formulation
- 3-noded linear triangular element
- single point Gaussian integration

$$\boldsymbol{S} = 2\frac{\partial \Psi}{\partial \boldsymbol{C}}$$

$$\Psi = \frac{\mu}{2}(\boldsymbol{I}_1 - 3) + \frac{\kappa}{2}(J-1)^2 \xrightarrow{\text{+damage}} \Psi^{dev} = (1-d)\Psi^{dev}$$

$$\boldsymbol{S} = \mu J^{-1}\left(\boldsymbol{I} - \frac{tr(\boldsymbol{C})}{2}\boldsymbol{C}^{-1}\right) + \kappa J(J-1)\boldsymbol{C}^{-1} \longrightarrow \boldsymbol{S}^{dam} = (1-d)\boldsymbol{S}$$

$$d = \gamma\left(1 - e^{-\frac{\beta}{m}}\right)$$

$$\beta = \max_{0<t<T}(\Psi(t) - \Psi_0)$$

$$\gamma \in]0,1]$$