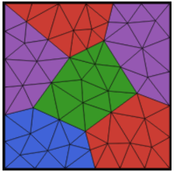
RUHR-UNIVERSITÄT BOCHUM

Using Ferrite.jl for multiscale bone simulations

Mischa Blaszczyk, Klaus Hackl

Ferrite Conference - Braunschweig

26.09.2022



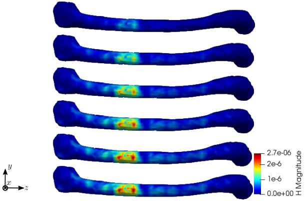
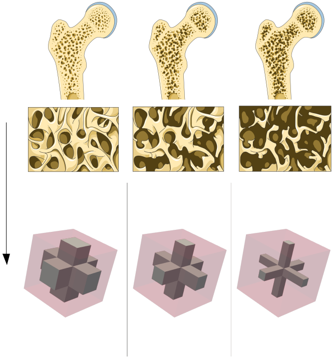


Table of Contents

- Research topic
- Motivation for choosing Ferrite.jl
- Package usage
- Custom functionalities and package synergies
- Wishes for the future

Structure of bone

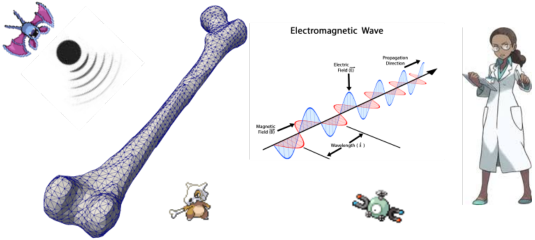
Motivation: simulation of spongy (cancellous) bone



https://commons.wikimedia.org/wiki/File:Osteoporosis_–_Smart-Servier.jpg

- Small beams of bone interconnected with bone marrow in between
- Application: sonography ⇒ early detection of osteoporosis

Effects in bone



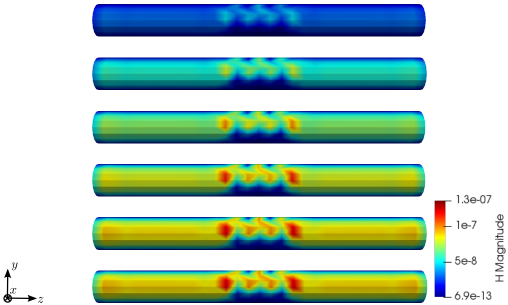
https://pokewiki.de, https://media.istockphoto.com

https://upload.wikimedia.org/wikipedia/commons/2/25/Electromagnetic_waves.png

$$\text{Ultrasound} \rightarrow \mathbf{u}(t) \ \rightarrow \mathbf{E}(t) \ \text{(piezoelectric effect)} \ \rightarrow \frac{\partial \mathbf{E}(t)}{\partial t} \neq 0 \rightarrow \mathbf{H}(t)$$

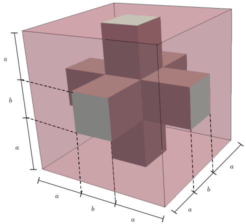
- Apply ultrasound
- Measure resulting magnetic field
- Obtain conclusions about the state of the bone

Example results



Magnetic field strength depending on used RVE (top 1 to bottom 6)

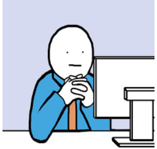
no.	$a[mm]$	$b[mm]$	ρ_{b}
1	0.43	0.14	5.3%
2	0.40	0.20	10.4%
3	0.38	0.24	14.5%
4	0.36	0.28	19.1%
5	0.34	0.32	24.2%
6	0.32	0.36	29.5%



Requirements for our framework

Previous work done in FEAP

- Very complicated framework, basically "black box"
- Oftentimes documentation is not clear
- Difficult to obtain important information in the program
- Needs Pardiso solver for acceptable performance
- Multiscale simulations for my model basically not possible



https://imgflip.com/meme/92682183/Staring-at-computer

Requirements for new framework

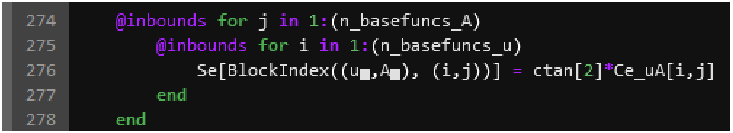
- No black box, element routine accessible
- Multiscale simulations have to be possible
- Fast speed, usage of parallelization and computer cluster

Usage of Ferrite.jl

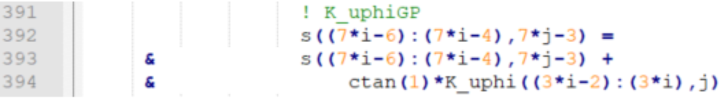
Which parts were helpful for the implementation?

- Grid framework excluding mesh generation
- Shape functions (fe_value, shape_value, shape_divergence, etc.)
- Assembly, block arrays are very nice for coupled problems compared to FEAP
- DoF-handler, Dirichlet boundary conditions
- WriteVTK for output/postprocessing in Paraview

Note: tutorials were very useful for learning how everything works!



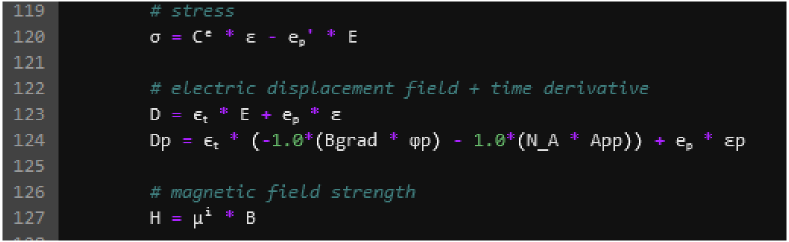
```julia
    @inbounds for j in 1:(n_basefuncs_A)
        @inbounds for i in 1:(n_basefuncs_u)
            Se[BlockIndex((u■,A■), (i,j))] = ctan[2]*Ce_uA[i,j]
        end
    end
```



```fortran
                        ! K_uphiGP
                        s((7*i-6):(7*i-4),7*j-3) =
     &                  s((7*i-6):(7*i-4),7*j-3) +
     &                      ctan(1)*K_uphi((3*i-2):(3*i),j)
```

FE Square

Microscale: use material models for both bone phases

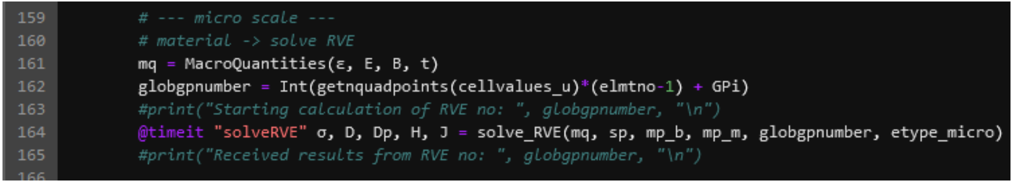


```
        # stress
        σ = Cᵉ * ε - eₚ' * E

        # electric displacement field + time derivative
        D = ϵₜ * E + eₚ * ε
        Dp = ϵₜ * (-1.0*(Bgrad * φp) - 1.0*(N_A * App)) + eₚ * εp

        # magnetic field strength
        H = μⁱ * B
```

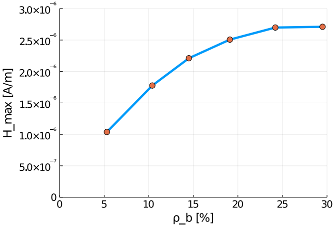
Macroscale: start micro calculation instead, very easy to implement (standard function call)



```
        # --- micro scale ---
        # material -> solve RVE
        mq = MacroQuantities(ε, E, B, t)
        globgpnumber = Int(getnquadpoints(cellvalues_u)*(elmtno-1) + GPi)
        #print("Starting calculation of RVE no: ", globgpnumber, "\n")
        @timeit "solveRVE" σ, D, Dp, H, J = solve_RVE(mq, sp, mp_b, mp_m, globgpnumber, etype_micro)
        #print("Received results from RVE no: ", globgpnumber, "\n")
```

Useful addtional packages and features

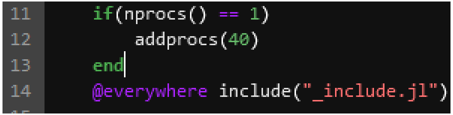
- IterativeSolvers/Krylovmethods: Solver for system of linear equations (We use bicgstab(l) as our system as it is non-symmeetric and not positive definite)
- Time integration: self-implemented from paper, maybe add simple cases to Ferrite.jl? - Problem here is the combination of e.g. Newton Raphson method and the different time integration schemes (in FEAP already implemented but limited number of algorithms, Newmark method is default)
- HDF5.jl: Process HDF5 files which we use to store viscoelastic data (could also be done differently)
- Plots.jl/Makie.jl: Postprocessing, creating images, etc.



Parellelization / Cluster usage

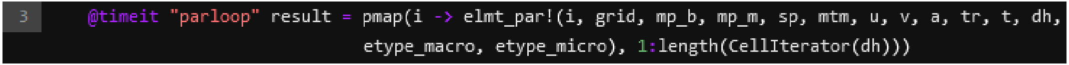
Package: Distributed.jl (enables parallelization e.g. on computer cluster)

First step: create processes and use **@everywhere** macro



```
    if(nprocs() == 1)
        addprocs(40)
    end
    @everywhere include("_include.jl")
```

Second step: split element calculations on processes by using **pmap** function



```
    @timeit "parloop" result = pmap(i -> elmt_par!(i, grid, mp_b, mp_m, sp, mtm, u, v, a, tr, t, dh,
                                   etype_macro, etype_micro), 1:length(CellIterator(dh)))
```

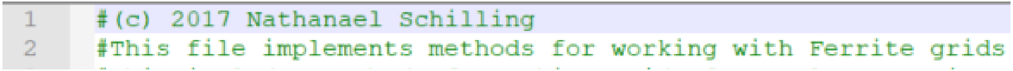
- Microscale calculations are independent of each other, therefore split is easily possible
- Speed up is huge (total time can be divided by number of processes)
- Further speed up might be possible e.g. by using more nodes, getting rid of HDF5 files etc.
- The pmap function automatically estimates the duration of each task and does the split accordingly

Perioduc boundary conditions

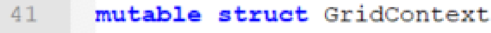
In FEAP: already implemented and easy to use

We required the general case in Ferrite.jl

Packages: CoherentStructures.jl and Distances.jl (Big thanks to O.Junge's group at TU Munich!)

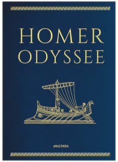


```
#(c) 2017 Nathanael Schilling
#This file implements methods for working with Ferrite grids
```



```
mutable struct GridContext
```

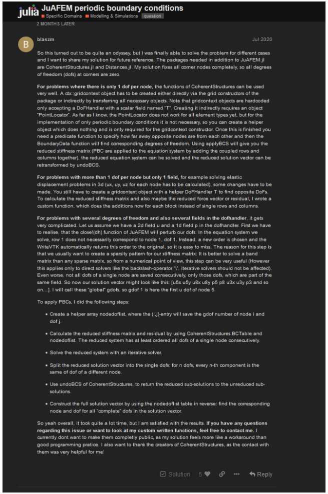
Expansion of Ferrite.jl grid with additional features, allows to find "cohesive" faces and nodes (e.g. left and right, front and back, top and bottom), stored in **BCTable**



Perioduc boundary conditions

Problem: CoherentStructures.jl only supports PBCs for one degree of freedom called "T".

- Find out order in which DoFs are perturbed (done by close!(dh) for numerical reasons)
- Store order in helper array **nodedoflist**
- Calculate reduced stiffness matrix and residual by using **BCTable**
- Solve the reduced system e.g. with an iterative solver
- Split reduced solution vector into the single DoFs
- Use **CoherentStructures.undoBCS** to return unreduced "subsolutions"
- Construct full solution vector by using **nodedoflist** "in reverse"



JuAFEM periodic boundary conditions

blaszm Jul 2020

So this turned out to be quite an odyssey, but I was finally able to solve the problem for different cases and I want to share my solution for future reference. The packages needed in addition to JuAFEM.jl are CoherentStructures.jl and Distances.jl. My solution fixes all corner nodes completely, so all degrees of freedom (dofs) at corners are zero.

For problems where there is only 1 dof per node, the functions of CoherentStructures can be used very well. A ctx::gridcontext object has to be created either directly via the grid constructors of the package or indirectly by transferring all necessary objects. Note that gridcontext objects are hardcoded only accepting a DoFHandler with a scalar field named "T". Creating it indirectly requires an object "PointLocator". As far as I know, the PointLocator does not work for all element types yet, but for the implementation of only periodic boundary conditions it is not necessary, so you can create a helper object which does nothing and is only required for the gridcontext constructor. Once this is finished you need a predicate function to specify how far away opposite nodes are from each other and then the BoundaryData function will find corresponding degrees of freedom. Using applyBCS will give you the reduced stiffness matrix (PBC are applied to the equation system by adding the coupled rows and columns together), the reduced equation system can be solved and the reduced solution vector can be retransformed by undoBCS.

For problems with more than 1 dof per node but only 1 field, for example solving elastic displacement problems in 3d (ux, uy, uz for each node has to be calculated), some changes have to be made. You still have to create a gridcontext object with a helper DoFHandler T to find opposite DoFs. To calculate the reduced stiffness matrix and also maybe the reduced force vector or residual, I wrote a custom function, which does the additions now for each block instead of single rows and columns.

For problems with several degrees of freedom and also several fields in the dofhandler, it gets very complicated. Let us assume we have a 2d field u and a 1d field p in the dofhandler. First we have to realise, that the close!(dh) function of JuAFEM will perturb our dofs: In the equation system we solve, row 1 does not necessarily correspond to node 1, dof 1. Instead, a new order is chosen and the WriteVTK automatically returns this order to the original, so it is easy to miss. The reason for this step is that we usually want to create a sparsity pattern for our stiffness matrix: It is better to solve a band matrix than any sparse matrix, so from a numerical point of view, this step can be very useful (However this applies only to direct solvers like the backslash-operator "\", iterative solvers should not be affected). Even worse, not all dofs of a single node are saved consecutively, only those dofs, which are part of the same field. So now our solution vector might look like this: [u5x u5y u8x u8y p5 p8 u3x u3y p3 and so on…]. I will call these "global" gdofs, so gdof 1 is here the first u dof of node 5.

To apply PBCs, I did the following steps:

- Create a helper array nodedoflist, where the (i,j)-entry will save the gdof number of node i and dof j.
- Calculate the reduced stiffness matrix and residual by using CoherentStructures.BCTable and nodedoflist. The reduced system has at least ordered all dofs of a single node consecutively.
- Solve the reduced system with an iterative solver.
- Split the reduced solution vector into the single dofs: for n dofs, every n-th component is the same of dof of a different node.
- Use undoBCS of CoherentStructures, to return the reduced sub-solutions to the unreduced sub-solutions.
- Construct the full solution vector by using the nodedoflist table in reverse: find the corresponding node and dof for all "complete" dofs in the solution vector.

So yeah overall, it took quite a lot time, but I am satisfied with the results. **If you have any questions regarding this issue or want to look at my custom written functions, feel free to contact me.** I currently dont want to make them completly public, as my solution feels more like a workaround than good programming pratice. I also want to thank the creators of CoherentStructures, as the contact with them was very helpful for me!

Mesh reader

Motivation: mesh generation with Ferrite.jl very limited

- Found old sketch online somewhere
- Improved scetch to Gmsh .msh-file reader for most important cases
- .inp-reader (Abaqus mesh file, also supported by Gmsh) - from D.R.Jantos
- Probably a couple of different mesh readers at our chair alone

Problems and possible solution

- Self-written mesh readers can only be used for specific cases (certain element types, no mixed meshes, i.e. e.g. Quad + Tet elements in a single mesh, etc.)
- A uniformed and "official" mesh reader for the most important file types and mesh types would be nice to have (Ferrite would be more complete as it then support Preprocessing, Solver and Postprocessing)
- Not easy to write it for general cases!

Gmsh

ABAQUS

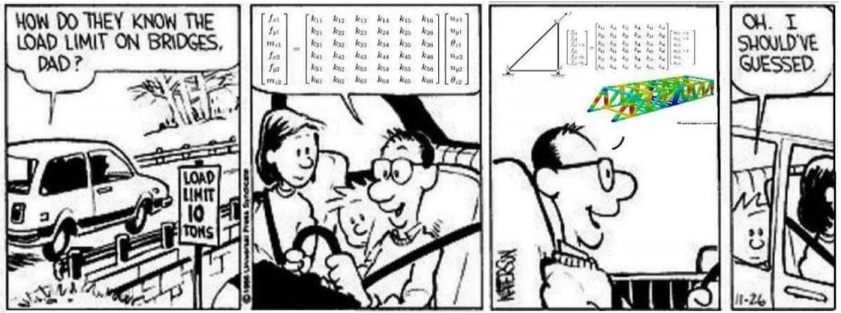
Conclusion and outlook

Summary

- Fully coupled multiscale and multiphase material model of cancellous bone has been implemented using mainly Ferrite.jl
- Ferrite toolbox was very accessible (at least for advanced users of FEM)
- Speed of the simulations is very good, still some optimization potential
- We started also using Ferrite.jl / Julia in general for teaching

Wishlist and outlook

- Fairly recent framework compared to other long term projects ⇒ some features are still missing
- Unified "official" mesh reader for many different cases (element types, mixed meshes etc.)
- For my research specifically: Nédélec (edge) elements (work in progress)


HOW DO THEY KNOW THE LOAD LIMIT ON BRIDGES, DAD?
LOAD LIMIT 10 TONS
© 1986 Universal Press Syndicate
$\begin{bmatrix} f_{x1} \\ f_{y1} \\ m_{z1} \\ f_{x2} \\ f_{y2} \\ m_{z2} \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_{z1} \\ u_{x2} \\ u_{y2} \\ \theta_{z2} \end{bmatrix}$
WATTERSON
OH. I SHOULD'VE GUESSED.
11-26

https://www.reddit.com/r/surrealmemes

Thank you for

your kind attention!