GPU4GEO

Frontier GPU multi-physics solvers

[1,2] Ludovic Räss

[1,2] Ivan Utkin

[1] Albert De Montserrat

[3] Boris Kaus

[4] Samuel Omlin

[5] Thibault Duretz

1 | ETH Zurich

2 | Swiss Federal Institute for Forest, Snow and Landscape Research (WSL)

3 | Johannes Gutenberg University Mainz

4 | Swiss National Supercomputing Centre, CSCS

5 | Institut für Geowissenschaften, Goethe-Universität Frankfurt, Frankfurt

ETH zürich

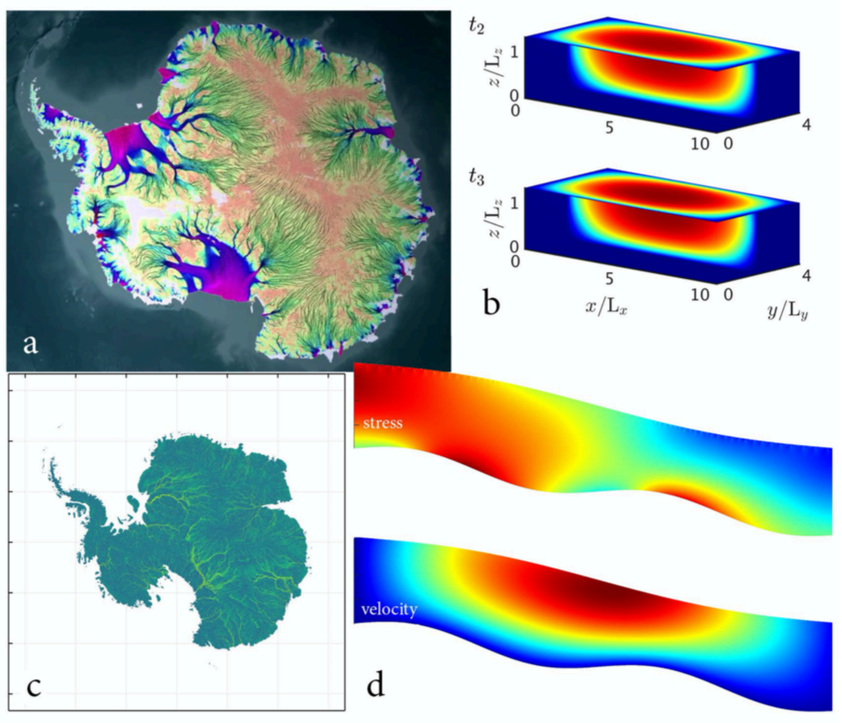
Laboratory of Hydraulics, Hydrology and Glaciology

Platform for Advanced Scientific Computing

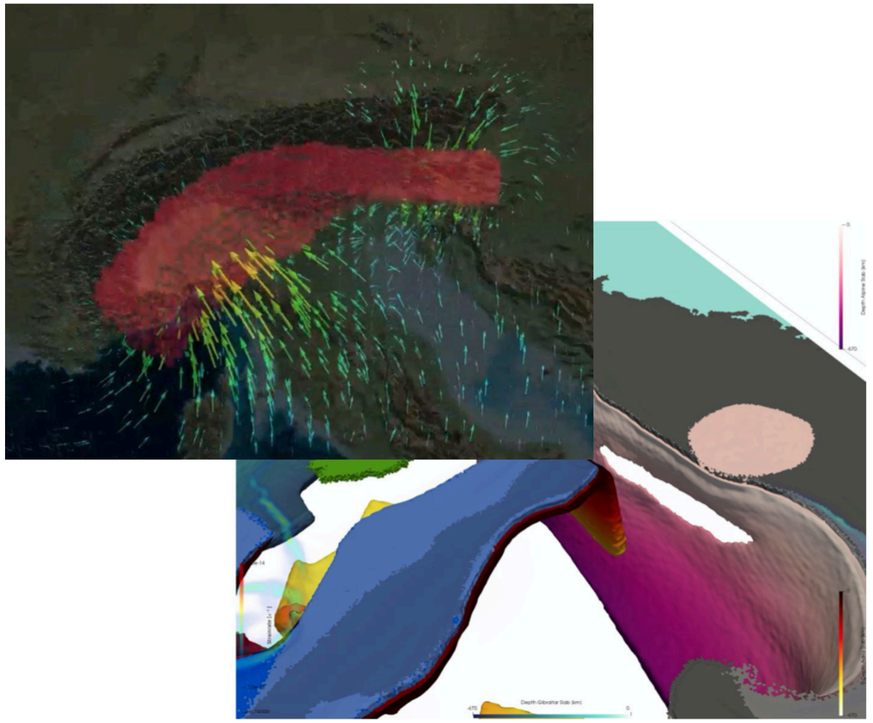
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Application to ice flow dynamics and geodynamics

$$\nabla \cdot \left(\eta \left[\nabla u + (\nabla u)^T \right] \right) - \nabla p = f$$
$$\nabla \cdot u = 0$$

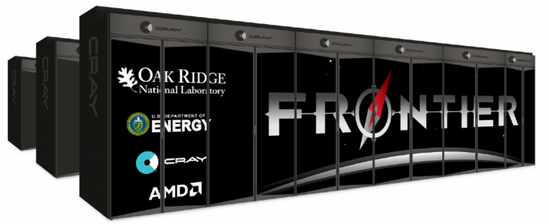


ice flow dynamics



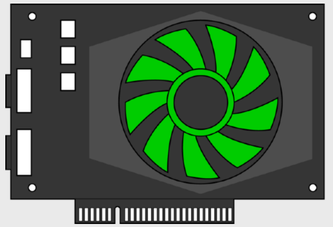
geodynamics

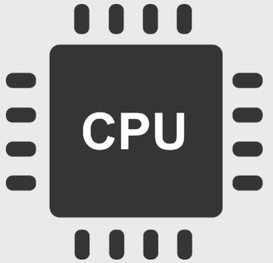
“The goal of the GPU4GEO project is to propose software tools which provide a way forward in these two domains by exploiting two powerful emerging paradigms in HPC: massively parallel iterative solvers, and HPC with Julia on graphical processing units (GPUs).”



NVIDIA

AMD





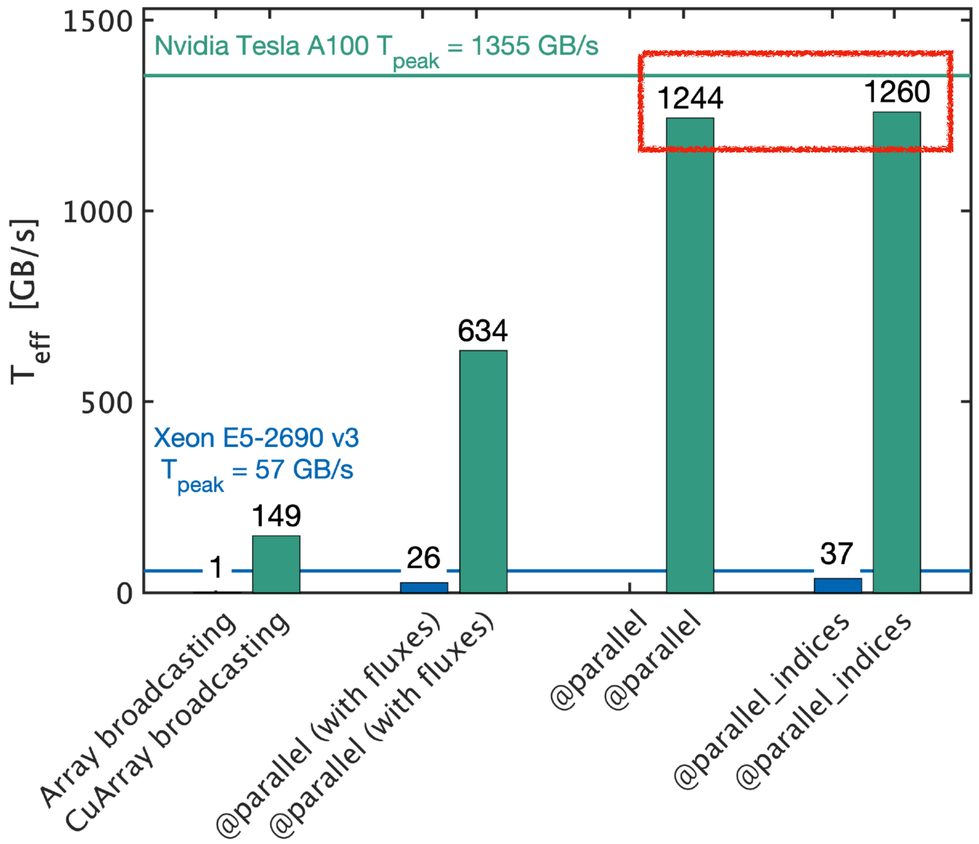
1500 GB/s 100 GB/s

Memory bandwidth

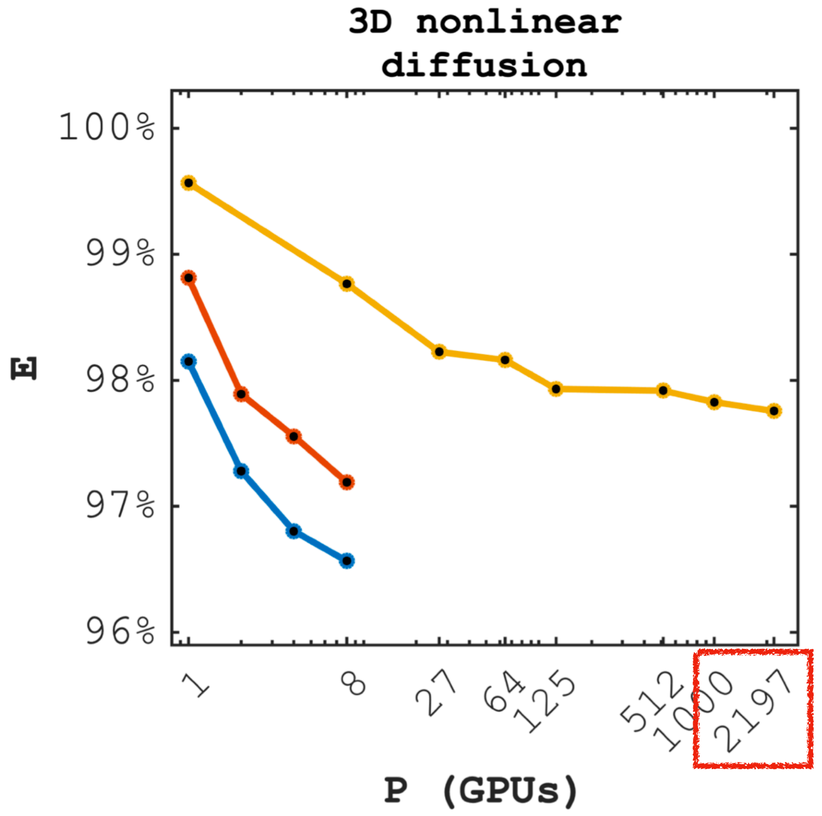
But why julia +  ?

Unleash julia HPC on (multi-)GPUs

`ParallelStencil.jl` enables parallel stencil computations on GPUs (Nvidia & AMD) and CPUs.



`ImplicitGlobalGrid.jl` enables stencil-based distributed parallelisation of GPU and CPU.

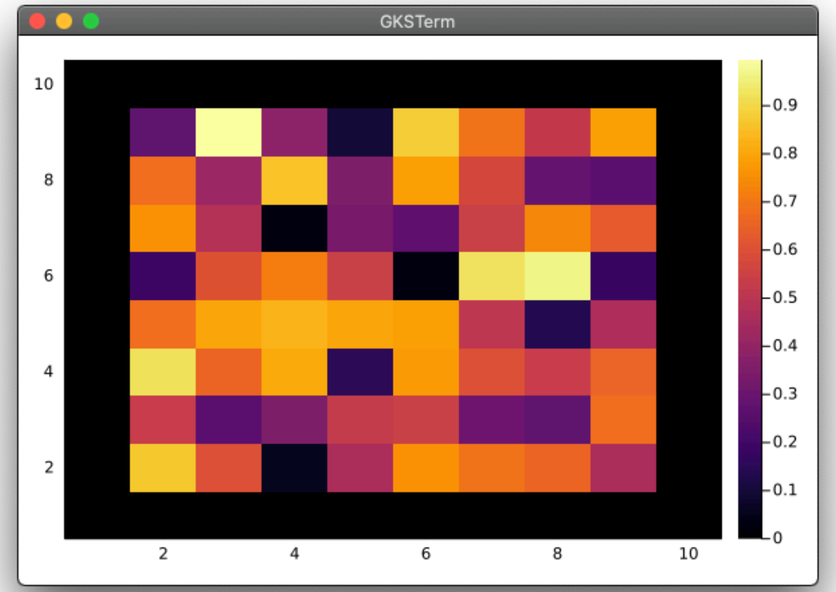


Räss et al. (2022)

Scalable solvers

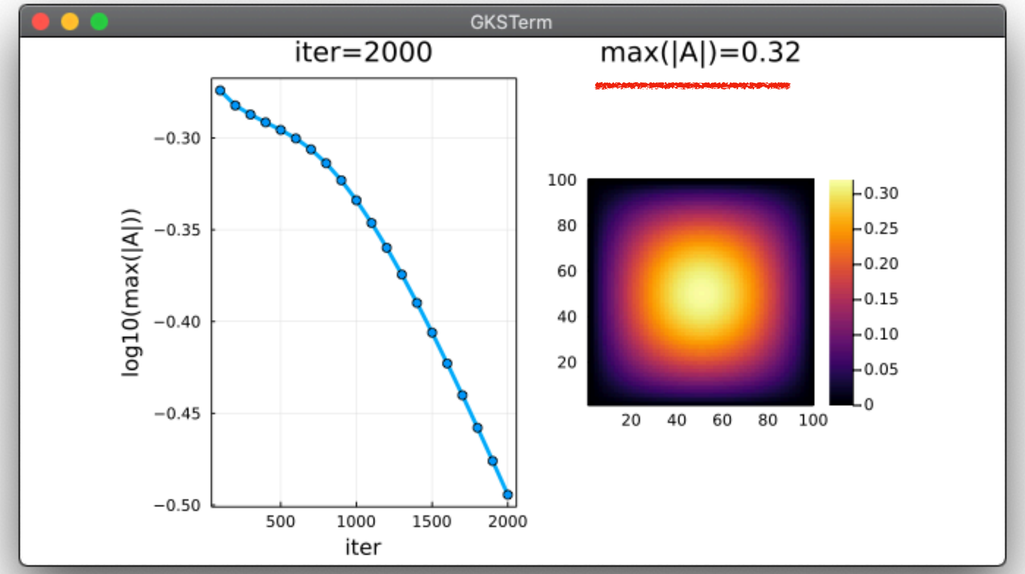
The accelerated pseudo-transient method - using wave-like pseudo-physics to accelerate convergence.

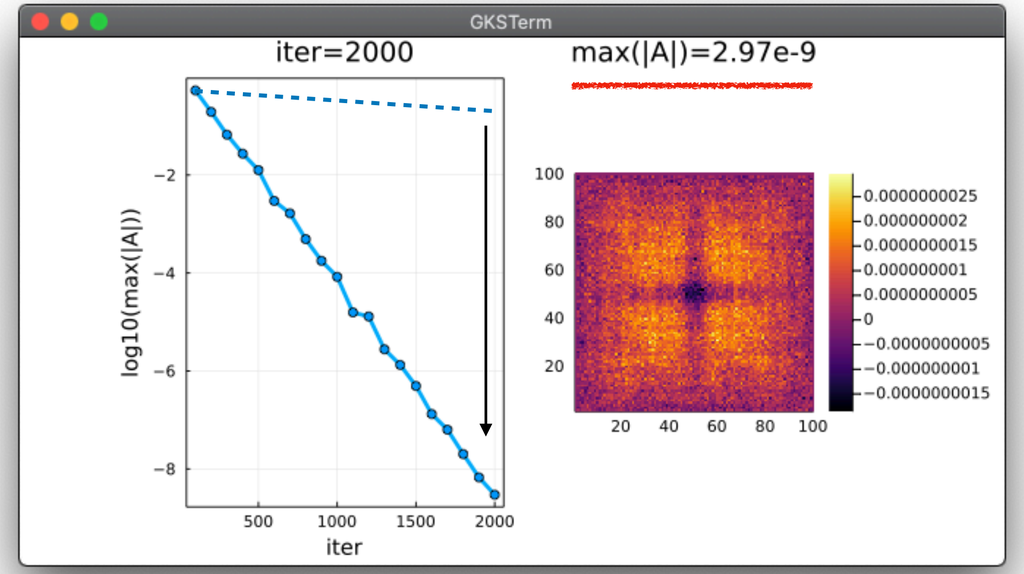
```
@views function laplacian2D()
    order   = 2
    fact    = 2
    # Physics
    lx, ly  = 10, 10
    D       = 1
    # Numerics
    nx, ny  = fact*50, fact*50
    dx, dy  = lx/nx, ly/ny
    niter   = 20*nx
    alpha   = 2.0/nx
    dt      = dx/sqrt(D)/2.1
    # Initial conditions
    A       = zeros(Float64, nx,ny)
    dAdt    = zeros(Float64, nx,ny)
    A[2:end-1,2:end-1] .= rand(nx-2,ny-2)
    # iteration loop
    for it = 1:niter
        dAdt[2:end-1,2:end-1] .= dAdt[2:end-1,2:end-1]*(1-alpha)*(order-1) +
                                 dt*( diff(diff(A[:,2:end-1],dims=1),dims=1)/dx^2 +
                                      diff(diff(A[2:end-1,:],dims=2),dims=2)/dy^2 )
        A                     .= A + dt*dAdt
    end
    return
end
```



Scalable solvers

Acceleration is cool - it keeps the iterations count low and scales linearly with **dx**

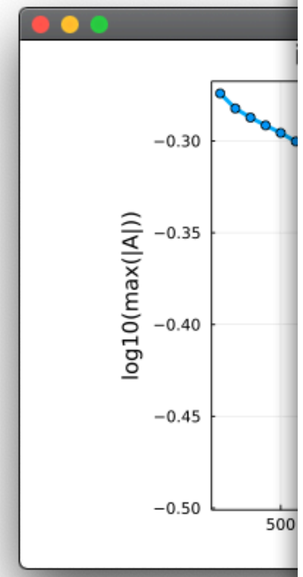




Application	GPU	n_x local	n_x global	$n_{\text{dof}}^{\text{tot}}$	t/iter (s)	$n_{\text{iter}}^{\text{tot}}$	wall time
Nonlinear diffusion 3D	Tesla P100	512	6632	0.292×10^{12}	0.0348	12'932	7.5 min
Visco-elastic Stokes 3D	Tesla P100	383	4955	1.23×10^{12}	0.0644	408'788	7.31 h

Scalable solvers

Acceleration is cool - it keeps the iterations count low and scales linearly with **dx**



Geosci. Model Dev., 15, 5757–5786, 2022
https://doi.org/10.5194/gmd-15-5757-2022

Geoscientific Model Development
Open Access
EGU

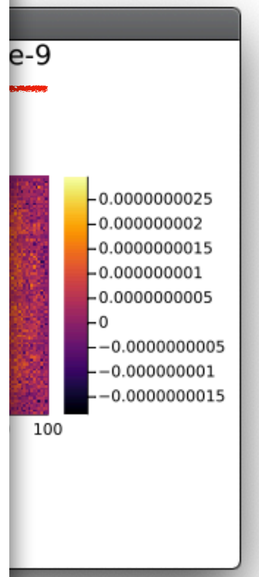
Development and technical paper

Assessing the robustness and scalability of the accelerated pseudo-transient method

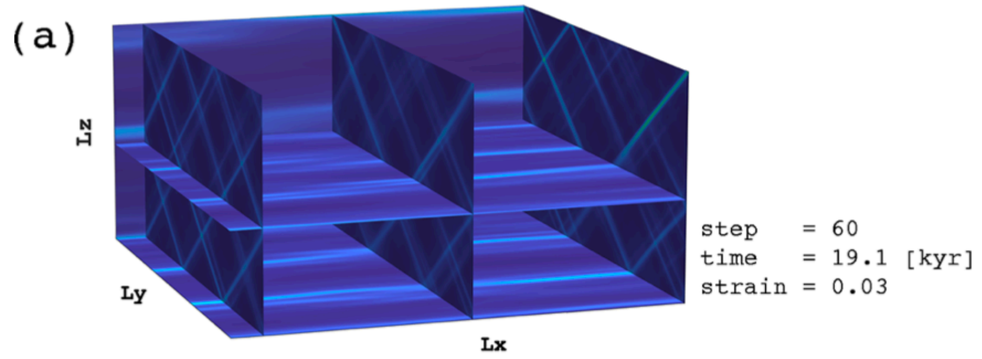
Ludovic Räss[1,2,★], **Ivan Utkin**[1,2,3,★], **Thibault Duretz**[4,5], **Samuel Omlin**[6], **and Yuri Y. Podladchikov**[3,7,8]

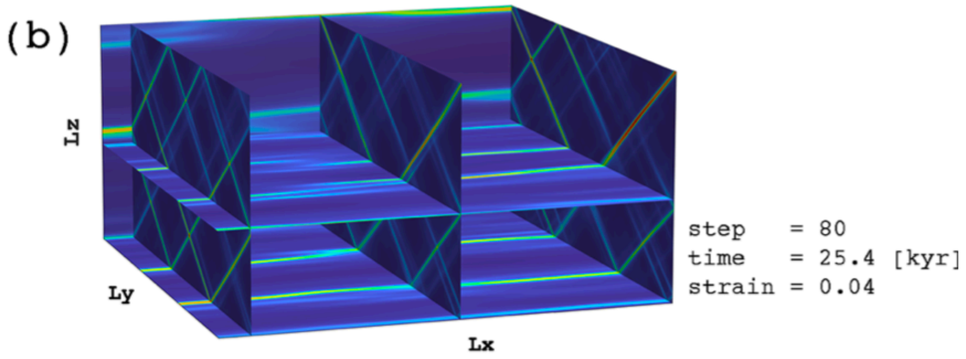
[1]Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Zurich, Switzerland
[2]Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland
[3]Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, Moscow, Russia
[4]Institut für Geowissenschaften, Goethe-Universität Frankfurt, Frankfurt, Germany
[5]Univ. Rennes, CNRS, Géosciences Rennes UMR 6118, 35000 Rennes, France
[6]Swiss National Supercomputing Centre (CSCS), ETH Zurich, Lugano, Switzerland
[7]Institute of Earth Sciences, University of Lausanne, Lausanne, Switzerland
[8]Swiss Geocomputing Centre, University of Lausanne, Lausanne, Switzerland
★These authors contributed equally to this work.

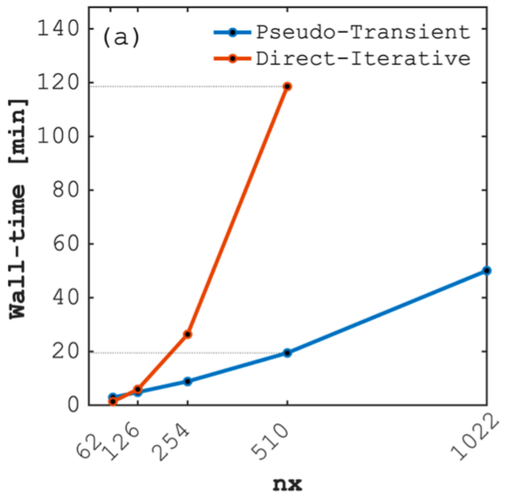
Correspondence: Ludovic Räss (luraess@ethz.ch)

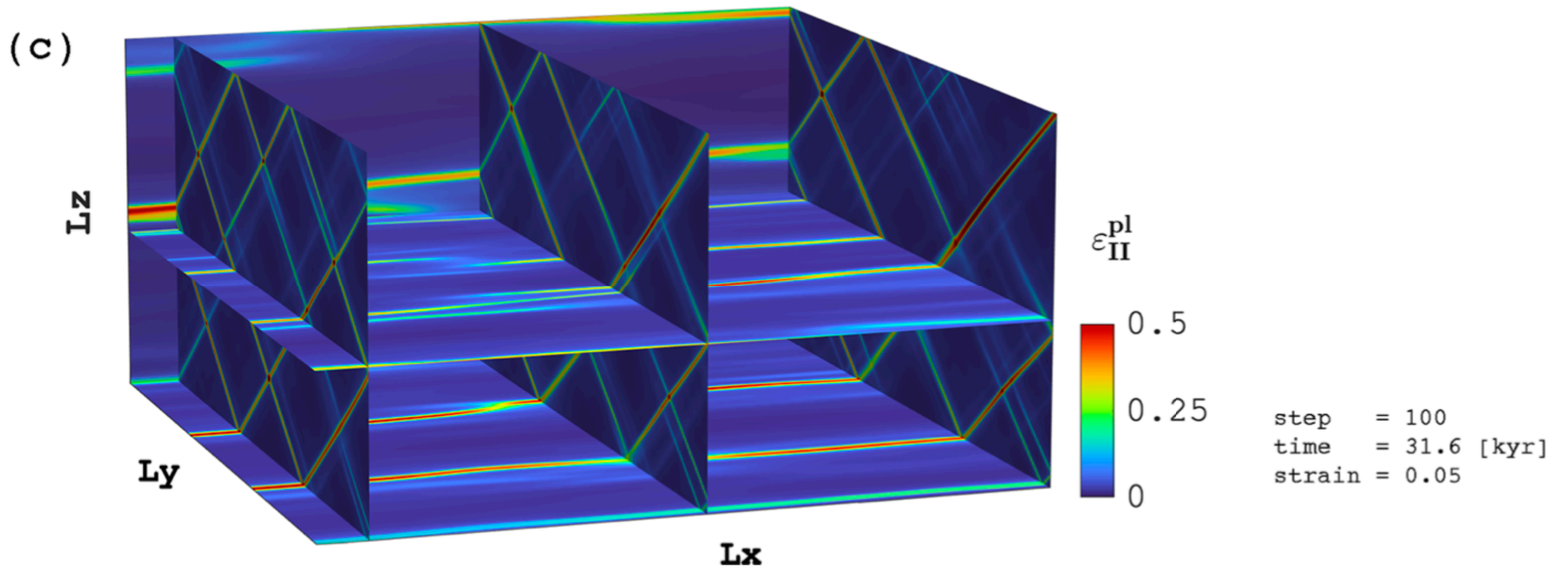


Nonlinear mechanical elasto-viscoplastic solvers

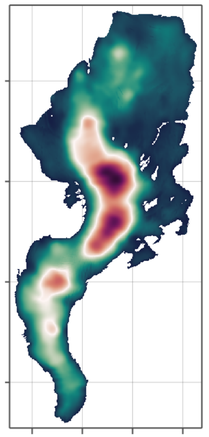




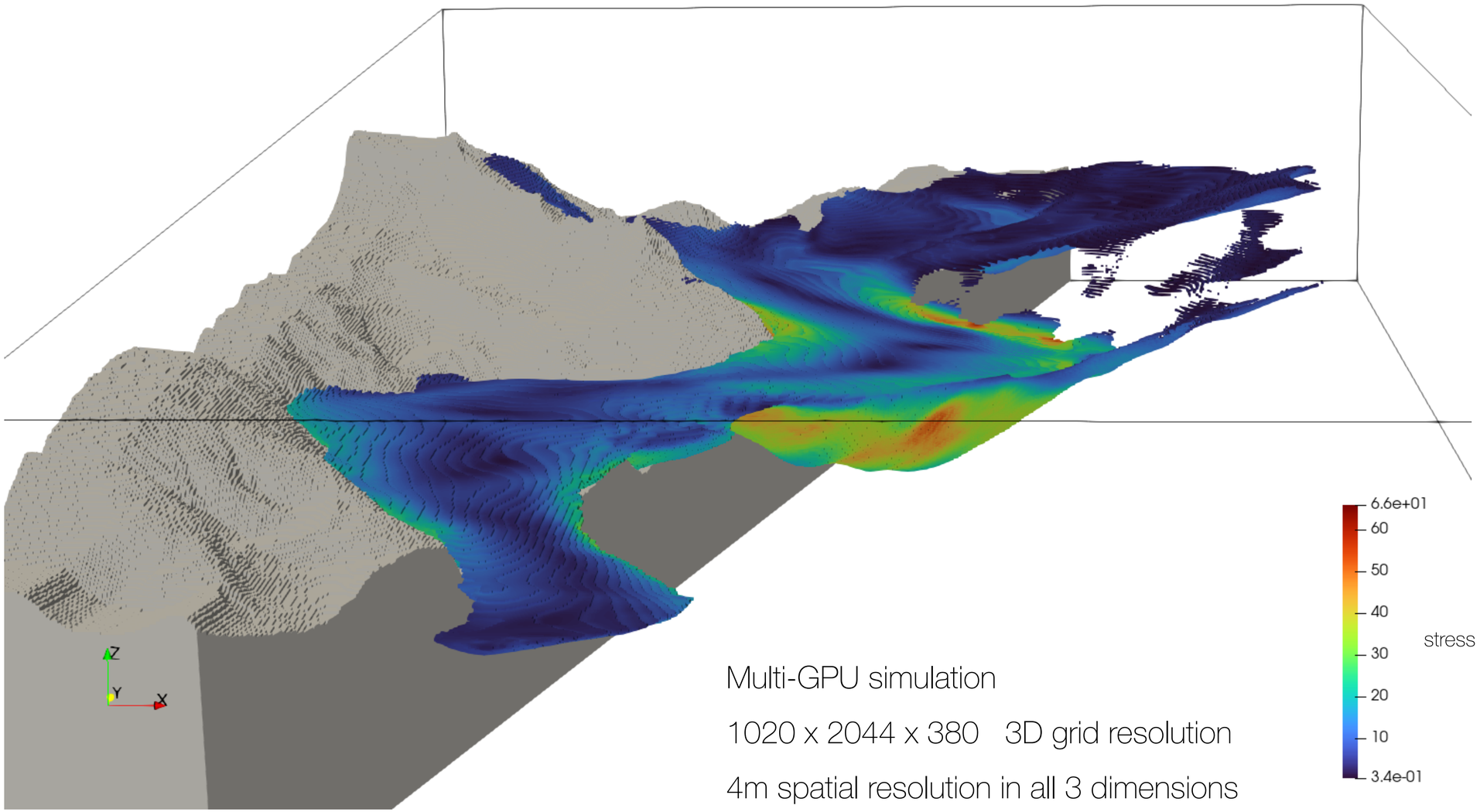




Thermomechanical ice flow solvers - FastIce



Rhone glacier



Multi-GPU simulation

1020 x 2044 x 380 3D grid resolution

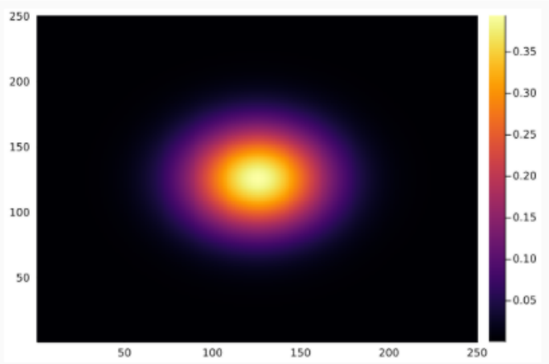
4m spatial resolution in all 3 dimensions

stress invariant

AMD GPUs for exascale

2D diffusion solver running on 8 AMD MI250x GPUs using ROCm-aware MPI on OLCF's Crusher test system, a small replica of Frontier, the first exascale supercomputer.

`ImplicitGlobalGrid.jl`




```
Start diffusion process
Global grid: 32762x16382x1 (nprocs: 8, dims: 4x2x1)
Process 0 selecting device 1
Process 1 selecting device 2
Process 2 selecting device 3
Process 3 selecting device 4
Process 5 selecting device 6
Process 6 selecting device 7
Process 7 selecting device 8
Process 4 selecting device 5
Starting the time loop 🚀...done
Executed 1000 steps in = 2.679e+00 sec (@ T_eff = 595.00 GB/s)
```

- Recently added AMDGPU.jl support to MPI.jl and ImplicitGlobalGrid.jl
- AMDGPU.jl support soon available in ParallelStencil.jl
- Enables ROCm-aware MPI: check out https://github.com/luraess/ROCm-MPI

HPC julia packages we develop

`ParallelStencil.jl` enables parallel stencil computations on GPUs (Nvidia & AMD) and CPUs.

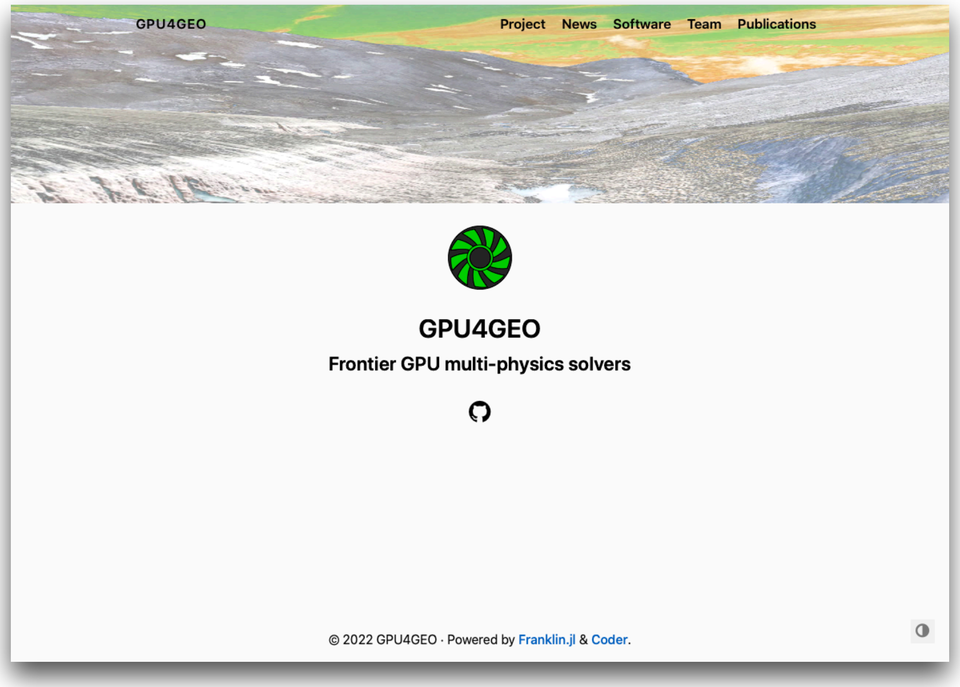
`ImplicitGlobalGrid.jl` renders stencil-based distributed parallelisation of GPU and CPU applications trivial.

`CellArrays.jl` arrays containing logical cells of small arrays or structs.

`GeoParams.jl` provides computational routines to define material parameters and perform non-dimensionalisation.

… and more, see:

https://ptsolvers.github.io/GPU4GEO/software/



HPC julia GPU teaching

https://pde-on-gpu.vaw.ethz.ch

Laboratory of Hydraulics, Hydrology and Glaciology

Fall 2021 | ETHZ 101-0250-00

Solving partial differential equations in parallel on GPUs

by Ludovic Räss, Mauro Werder & Samuel Omlin

Solving PDEs in parallel on GPUs with Julia

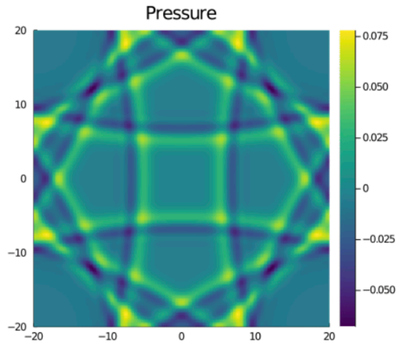
🎉 Welcome to ETH's **course 101-0250-00L** on solving partial differential equations (PDEs) in parallel on graphical processing units (GPUs) with the Julia programming language.

> ***Announce:*** *lectures start at 12h45 every Tuesday (more)*

> ⚠️ *The requirement for a Covid certificate applies from Monday, 20 September 2021 for teachers and students. For further questions, please see the Coronavirus FAQs for Students.*

Course informations

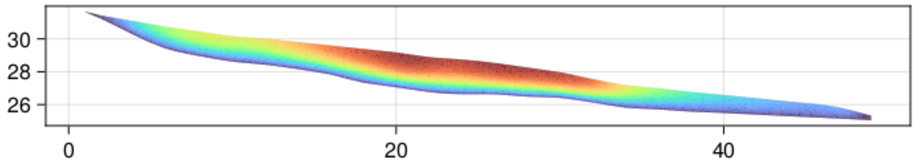
This course aims to cover state-of-the-art methods in modern parallel GPU computing, supercomputing and code development with applications to natural sciences and engineering.



Outlook

GPU4GEO project runs from 2021 to 2024 and we'll further work towards flagship ice dynamics and geodynamic applications, and enhance julia support for AMD GPUs, ARM CPUs, and more:

- Further develop ParallelStencil and ImplicitGlobalGrid
- Work on scalable GPU HPC optimisation using AD with Enzyme.jl
- Explore (hybrid) FD - FEM matrix-free iterative GPU solvers



Stay tuned with related contributions at JuliaCon22:

https://ptsolvers.github.io/GPU4GEO/posts/julia-juliacon22/

contact luraess@ethz.ch

ETH zürich

Laboratory of Hydraulics, Hydrology and Glaciology

Platform for Advanced Scientific Computing

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre