How to install Julia and run a model of JustRelax on Eejit

Lukas van de Wiel

December 1, 2025

1 Acknowledgements

This manual was created with the valuable help of Ludovic Raas and Albert de Montserrat

2 Prerequisites

You can use the existing installation of Julia on Eejit, which is nice and stable, or install your own version, if you want to try the latest git versions. This section explains the installation of Julia and the required modules (called 'packages' in Julia), and uses the existing Julia installation as example.

2.1 Directory structure

To install Julia, first select a directory in which you want to install it.
On Eejit, this is:

```
/scratch/tectonics/GPU4GEO/julia_local
```

In this directory, create three subdirectories:

1. JuliaUp
2. julia_depot
3. julia_prefs

2.2 bashrc

1. In your /.bashrc file, link to these directories, by adding:

   ```
   export JULIA_DEPOT_PATH=/scratch/tectonics/GPU4GEO/julia_local/julia_depot
   export JULIA_LOAD_PATH="$JULIA_LOAD_PATH:/scratch/tectonics/GPU4GEO/julia_local/julia_prefs/"
   ```

   Note that these paths cannot be relative, and cannot contain variables
   (We tried:

   ```
   SCRATCH=/scratch/tectonics/GPU4GEO
   export JULIA_DEPOT_PATH=$SCRATCH/julia_local/julia_depot
   export JULIA_LOAD_PATH="$JULIA_LOAD_PATH:$SCRATCH/julia_local/julia_prefs/"
   ```

   but that failed because it could not find the julia_depot directory and did not apply the preferences; more about that later.)

2. Source the bashrc:

   ```
   $> source ~/.bashrc
   ```

2.3 toml files

In the *julia_prefs* directory, create files

1. Project.toml

   ```
   [extras]
   CUDA_Runtime_jll = "76a88914-d11a-5bdc-97e0-2f5a05c973a2"
   MPIPreferences = "3da0fdf6-3ccc-4f1b-acd9-58baa6c99267"
   ```

2. localPreferences.toml

   ```
   [CUDA_Runtime_jll]
   local = "true"

   [MPIPreferences]
   _format = "1.0"
   abi = "OpenMPI"
   binary = "system"
   libmpi = "/trinity/opt/apps/development/dev2024/install/lib"
   mpiexec = "mpirun"
   ```

   (and possible hdf5 preferences if it is MPI-aware. Still look at that. Not urgent)

2.4 juliaUp

1. In the directory JuliaUp, run:

   ```
   curl -fsSL https://install.julialang.org | sh
   ```

   (stolen from: `https://github.com/JuliaLang/juliaup`)

   Thanks to the proper setting of *JULIA_DEPOT_PATH* and *JULIA_LOAD_PATH*, this will install the desired Julia in this desired directory.

   JuliaUp will also make changes to your /.bashrc, to ensure that the path to the Julia binary is in $PATH. This addition is marked by

   ```
   # >>> juliaup initialize >>>
   # !! Contents within this block are managed by juliaup !!
   [Julia environment variables are set here.]
   # <<< juliaup initialize <<<
   ```

2. Source the updated bashrc:

   ```
   $> source ~/.bashrc
   ```

   The correctness of the procedure can be validated by testing:

   ```
   $> which julia
   julia is /scratch/tectonics/GPU4GEO/julia_local/JuliaUp/bin/julia
   ```

2.5 Install the Julia packages

There are two ways to install the packages.

1. Using the Julia package manager
   - Pro: Usually working
   - Con: Last release version, so can be not state of the art
2. From Github
   - Pro: State of the art
   - Con: Can be broken every now and then, when one of the packages has changed something, but a related package has not yet adapted.

As we aim for a stable working installation, we use the Julia package manager

This can be done from the REPL (The interactive Julia shell; *Read-Evaluate-Print Loop*). Can be entered by typing 'julia' without any arguments, that should now be possible if JuliaUp has functioned properly.

The Julia REPL has 4 modes, a bit like the vi editor on steroids.

1. The default is the *Julian mode* (green prompt),
2. The *package manager mode* (blue prompt, can be entered with ']' and left to Julia mode with [backspace])
3. The *help mode* (yellow prompt, can be entered with '?' and left to Julia mode with [backspace])
4. The *shell mode* (red prompt, can be entered with ';' and left to Julia mode with [backspace]. From here you have a bash shell... and can even run Julia again!)

From any mode, Julia can be exited with ctrl-d. From the Julian mode also with 'exit()'

Go into Julia with the option *–project:*

```
$> julia --project
```

and check the status.

```
pkg> st
```

If you have used the existing install on Eejit, this should now give a list of the installed packages:

```
Status `/scratch/tectonics/GPU4GEO/testSubduct/Project.toml`
  [052768ef] CUDA v5.9.5
  [13f3f980] CairoMakie v0.15.7
  [d35fcfd7] CellArrays v0.3.2
  [e018b62d] GeoParams v0.7.8
  [3700c31b] GeophysicalModelGenerator v0.7.14
  [10dc771f] JustPIC v0.5.11
  [34418575] JustRelax v0.4.2
  [da04e1cc] MPI v0.20.23
  [94395366] ParallelStencil v0.14.3
```

And if you make a fresh installation, you should get:

```
Status `[your favourite Julia path here]` (empty project)
```

In that case, those packages still have to be installed. For running the 2D subduction model, that is all the packages that have been included with the *using foo*:

```
(@v1.12) pkg> add GeoParams, GeoParams, CairoMakie, JustRelax, ParallelStencil, JustPIC
```

If you want to run this model in parallel, add:

```
(@v1.12) pkg> add MPI, CellArrays
```

If you want to run on GPUs, load Eejit modules:

add:

```
(@v1.12) pkg> add CUDA
```

Similar to your */.bash_history*, the REPL also logs your commands for later reference, in

```
$HOME/.julia/logs/repl_history.jl
```

3 The subduction2D model

3.1 Getting the model

Get the subduction2D model. They can be obtained in two ways, each leading to a vastly different model:

1. Via Github, as part of the JustRelax package, hidden in directory `JustRelax.jl/miniapps/subduction/2D`. This version works!
2. Via the online documentation at: `https://ptsolvers.github.io/JustRelax.jl/dev/man/subduction2D/subduction2D`. This version does not work!

3.2 Fixing the model

A few changes need to be made for the model to run somewhat properly:

1. On line 1, replace 'GLMakie' by 'CairoMakie'. GLMakie generates windows with images in them, which is not allowed from the compute nodes. CairoMakie writes them to file instead, which is fine.
2. by default, CUDA is enabled, with line four:

   ```
   const isCUDA = true
   ```

   Change to 'false' if you want to run on CPU only.
3. Near line 86, change the arguments of the call to *init_particles* from

   ```
   particles = init_particles(backend_JP, nxcell, max_xcell, min_xcell, xvi, di, ni)
   ```

   to

   ```
   particles = init_particles(backend_JP, nxcell, max_xcell, min_xcell, xvi... )
   ```
4. Near line 260, change the call

   ```
   advection!(particles
   ```

   to

```
advection_MQS!(particles
```

5. The number of timesteps make the model run about 15 hours in serial. To reduce, this, adapt the number of time steps near line 180, for example from:

```
while it < 1000 # run only for 5 Myrs
```

to

```
while it < 100 # run only for 5 Myrs
```

3.3 Run the model in serial

The model can be run from the command line from the directory *miniapps/subduction/2D/Subduction2D* with a simple

```
$> julia Subduction2D.jl
```

(but do this only from an interactive session on a compute node) or from the REPL in the Julian mode with

While running, the model will create a directory and add output data it: images and vti files. The vti files can be viewed in Paraview.

```
julia> include("Subduction2D.jl")
```

A directory called 'Subduction2D' will de created containing:

1. Images of the reasult
2. A directory called *vtk* containing VTK files of every timesteps

The model will run a thousand time steps of 55-ish seconds each to complete in about 15 hours. Such a long model should be run via the scheduler, via a Slurm script:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --threads-per-core=1
#SBATCH --time=3-12:00:00
#SBATCH --partition=allq
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH -o job.%N.%j.out  # STDOUT
#SBATCH -e job.%N.%j.err  # STDERR
#SBATCH --job-name subduction2D

module purge
module load development
module load dev2024

julia Subduction2D.jl
```

3.4 Run the model in parallel

Julia kan run in parallel, but it comes with its own MPI version. This can bite the system-MPI.

To isolate it, in the slurm script, do not load the Eejit development environment.

This is installed in *mpiexecjl*.

Syntax to run takes the form of:

```
/scratch/tectonics/GPU4GEO/julia_local/julia_depot/bin/mpiexecjl -np 8 julia --project Subduction2D_
```

3.5 Run the model on GPUs

To run on CUDA, the CUDA module on Eejit is needed. To prevent requiring the entire dev202x module (causing conflicts with the Julia in it), there is a separate CUDA module that can be loaded:

```
module purge
module load development
module load CUDA/12.5
```

Be sure to set in Subduction2D.jl:

```
const isCUDA = true
```

after which a quick test on the login node can be run with:

```
julia --project Subduction2D.jl
```

and if everything has gone well, the job can be seen in the GPU processes:

```
$> nvidia-smi
```

Similarly this model can be run on a compute node with GPUs, with the slurm script:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --threads-per-core=1
#SBATCH --time=3-12:00:00
#SBATCH --partition=gpu
#SBATCH --gres=gpu
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH -o job.%N.%j.out  # STDOUT
#SBATCH -e job.%N.%j.err  # STDERR
#SBATCH --job-name IIsNotAnEejit

module purge
module load development
module load CUDA/12.5

julia --project Subduction2D.jl
```

Timesteps are run an order of magnitude faster on the GPU than on the CPU. :-)

4 The shearband2D model

The shearband model is also located in the justRelax package, at: *julia_depot/packages/JustRelax/3qFRD/miniapps/bench*

4.1 Run the model in serial

```
$> julia --project ShearBand2D.jl
```

It takes a few minbutes to run

4.2 Run the model in parallel

This model can be run interactively, or via slurm:

4.2.1 Interactively

An interactive session to a compute node can be opened with the shell command:

```
srun -p allq --ntasks-per-node=48 --exclusive --pty bash
```

or for a node with a GPU:

```
srun -p gpu --gres=gpu --ntasks-per-node=48 --exclusive --pty bash
```

The prompt will change from

```
(16:25 lukas@login01 ~) >
```

(or similar to:)

```
(16:25 lukas@gpu004 ~) >
```

Note that even though this node is called a gpu004, it is not a node with GPUs. It used to have in the past. The only nodes with actual GPUs are gpu041-045.

The model can then be run interactively from this single compute node:

```
$> /scratch/tectonics/GPU4GEO/julia_local/julia_depot/bin/mpiexecjl -np 8 julia --project ShearBand2
```

Apply the option '–project' to ensure that the packages defined in */scratch/tectonics/GPU4GEO/julia_local/julia_prefs* are available. Julia finds this Project.toml because of environment variable: *JULIA_LOAD_PATH=:/scratch/tectonics/GI*

4.2.2 slurm

The same can also be run in a Slurm script, on multiple nodes, for example on 4, using this script. Notice that no modules are loaded.

```
#!/bin/bash
#SBATCH --ntasks=192
#SBATCH --ntasks-per-node=48
#SBATCH --threads-per-core=1
#SBATCH --time=3-12:00:00
#SBATCH --partition=allq
#SBATCH --nodes=4
#SBATCH -o job.%N.%j.out  # STDOUT
#SBATCH -e job.%N.%j.err  # STDERR
#SBATCH --job-name shearBand

module purge

/scratch/tectonics/GPU4GEO/julia_local/julia_depot/bin/mpiexecjl \
-np 192 julia --project ShearBand2D_MPI.jl
```

Not that many of the SBATCH flags are the same to arguments of the *srun*-command above.