SPEAR Files Description

List of directories and files in the repository:

1. data:

This folder will contain the output from the simulations within a subfolder. You can set the name of your output folder in `run.jl` Line 22. The output files are simple binary text files that should be readable with any text editor. You can choose what you want to output here. Go to `src/main.jl` Lines 133-153 to look at the list of outputs stored here. You can additionally look at each variable in those lines and look it up in main to see what exactly it is storing.

2. plots:

This folder stores the plots analyzed from each simulation for publication quality figures. After running the simulation, you can open `analyze_results.jl` and specify which simulation you want to look at, and create a new folder with the same simulation name to store the plots (Lines 7-10, variables `path` and `out_path`). Running `analyze_results.jl` will unpack the stored simulation data into variables that you can then use for post-processing and plotting.

3. post:

This is the post-processing folder to look at plots from the output simulation. Most of the plots are created as functions so you can run them from the julia terminal, it will automatically save the plots into the above folder as png image.

The following files are there for now (you can always add more later):

- `plotting_script.jl`: This script does the most basic plotting to look at simulation output. The `plot_params()` function sets up the axes and ticks to make the plot publication quality. Other functions are just for plotting specific things. I usually add more functions here if I want to plot anything new, and I try to have separate files for separate projects.

- `event_details.jl`: This script is for calculating some rupture parameters like moment magnitude, rupture length, and start and stop of rupture. It is already included in the `analyze_results.jl` script so use this to add any new parameters you want to create from the results output (e.g., stress drop, surface rupture, or anything else you don't want to calculate during the main time loop of the simulation).

4. tests:

I just used this folder to show a basic test demo on the github page. I don't use this for anything else.

5. xfer:

Some one line bash executables to transfer files to (`xfer_up`) and from (`xfer_down`) remote machines.

6. src:

The source directory for setting up and running the simulations. Few files listed below are important here. You can learn more about the other files as you get more comfortable with the code, but they should not require much tweaking.

- `initial_conditions/`: This directory has the file `defaultInitialConditions.jl` where I set up the normal and shear stresses, and the friction parameters a-b. All these parameters are depth dependent thererfore I have separate file for them. You can look at what exists already, or add your own.

- `Assemble.jl`: This files sets up the mass matrix assembly. You don't need to change anything here for a fault-parallel damage zone simulation. If you want a different fault zone geometry, check out the folder `faultZoneGeometry/` for some examples and modify as needed. The stiffness is setup by `Kassemble.jl` and `MaterialProperties.jl`.

- `BoundaryMatrix.jl, MeshBox.jl, GetGLL.jl`: These files setup the Spectral Element matricies for the entire mesh, including the boundaries, the bulk. `GetGLL` is used for getting the polynomial interpolated nodes. This should not be changed for a majority of the cases.

- `NRsearch.jl, PCG.jl, dtevol.jl`: These are numerical procedures for the seismic part, the aseismic part, and the timestep evolution respectively. I do not use PCG anymore so it is mostly useless. Again, only change these parameters once you are more comfortable with the code.

- `main.jl`: Everything happens here sequentially. This is the file where you call all the other functions sequentially to run the simulation timeloop and store output. It is a good idea to get familiarized with this part the most. I set up the output files in here, as well as the time-dependent tweaking like precursor setup or rigidity evolution through time.

7. other files in the working directory:

- `install_dependencies.jl`: This is just loading external julia packages required for my simulations. You should run this once on each new computer you are running the code on.

- `output.jl`: I'm not using this anymore. This was used to create data structures for simulation output, needed when storing data as hdf5 native julia format.
- `par.jl`: This sets up the simulation parameters. Most of the variables are explained within the script, and the first 90 lines will change for most simulations. The other lines are using functions within the `src/` directory to set up mesh, timestep, and boundary conditions.
- `run.jl`: Calling this script runs the simulation. Here I set up the simulation resolution and the output directory to store files.