Reconnection-Intro

August 3, 2022

1 Motivation

In this module we will discuss current computational methods to modeling magnetic reconnection. This is motivated by the fact that many students studying this field are poorly taught the computational skills needed to succeed in this field. From more broad skills like using Git version control and plotting in Python, to more narrow knowledge of magnetohydrodynamic equations, we will aim to properly introduce these vital skills to students, taking care to cater to various levels of prior knowledge. (**figures demonstrating high performance computing**)

2 Background

2.1 An overview of magnetic reconnection

Magnetic reconnection is a process when the magnetic topology is rearranged in the presence of a plasma. This causes a burst of energy in the form of heat, kinetic energy for electrons and ions, large scale plasma flow. When this occurs, the magnetic field lines snap and connect with the other field lines (*see the gif below*). Magnetic reconnection occurs with a pair of oppositel directed (antiparallel) magnetic field lines in a system that experiences high heat levels. Common examples of this are coronal mass ejections and when the solar wind impacts the Earth's magnetosphere. Zooming out a bit, magnetic field lines have inherent tension, which act to straighten out magnetic field lines. When reconnection occurs, this force acts towards the center of curvature and pushes the plasma out, similar to a slingshot. We will discuss the source of tension later, but first we will find out what, mathematically, causes this process. (**discuss tension more after ohms law.**)

Source: By ChamouJacoN - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=7496329

2.2 A deeper dive into magnetic reconnection

We first start with the ideal Ohms Law equation for ideal non-resistive magnetohydrodynamic models

$$E + u \times B = 0$$

which describe systems that follow the frozen in theorem. The theorem states that without any resistivity in a fluid, the magnetic field is frozen in to said fluid. In a plasma this means that the magnetic field lines move with the plasma. Furthermore, the field will not change when moving along with the field lines, but motions transverse to the field carry the field with the motion. When we introduce resistivity, Ohm's Law transforms into

$$E + u \times B = \eta J$$

where η is the magnetic diffusivity, and the system no longer follows the frozen in theorem. This means that the magnetic field lines are no longer frozen to the plasma and can reconnect.

We can derive the frozen-in theorem by taking Ohm's Law and applying it to Faraday's Law, getting

$$\frac{\partial B}{\partial t} = \nabla \times (V \times B) - c\nabla \times (\eta J)$$

since the current density J is proportional to $\nabla \times B$ through Ampere's Law, we arrive at

$$\frac{\partial B}{\partial t} = \nabla \times (V \times B) + c\eta(\nabla^2 J)$$

after some simplifications, namely using the curl of a curl identity and that $\nabla \cdot B = 0$ since there are no magnetic monopoles.

Since $\eta = \frac{1}{\mu_0 \sigma}$, where μ_0 is the permiability of free space as σ is the electrical conductivity, when we look no resistivity/infinite conductivity, the η term in the above equation (**give reference**) falls out and we are left with the induction equation.

With the induction equation

$$\frac{\partial B}{\partial t} = \nabla \times (V \times B)$$

we use $\Phi = \int_s B \cdot dS$ to get

$$\frac{\mathrm{d}\Phi}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int_s B \cdot dS$$

The two ways the magnetic flux can change is through a change in the magnetic field strength $\frac{\partial B}{\partial t}$ or a movement in the surface we integrate over. We can substitute the induction equation in for the first term and get

$$\int_s \nabla \times (v \times B) \cdot ds + \int_s B \cdot (v \times d\ell)$$

The second term arises from movement on the surface of integration. We can use Stokes' Theorem and the vector identity $(A \times B) \cdot C = -B \cdot (A \times C)$ to get

$$\int_c v \times B \cdot d\ell - \int_c (v \times B) \cdot d\ell = 0$$

Thus, we find that $\frac{d}{dt}\int_s B \cdot ds = 0$, proving the frozen in theorem when we have no resistivity.

2.3 Modeling magnetic reconnection

In order to talk about the models we must find all the physical properties to consider when generating a model and any approximations. First we start with the electron momentum equation for the ith component of the electric field

$$m\frac{\partial n u_i}{\partial t} + \frac{\partial \mathcal{P}_{ij}}{\partial x_j} = nq(E_i + \epsilon_{ijk} u_j B_k)$$

our goal is to end up with the generalized Ohm's Law equation which will play a key role in the future. In order to get there, we rearrange the electron momentum equation to get $E_i = \frac{m}{nq}\frac{\partial n u_i}{\partial t} + \frac{\partial \mathcal{P}_{ij}}{\partial x_j} - \epsilon_{ijk} u_j B_k$ and simplify the ϵ term. Feel free to read up on Einstein notation, but for this demo we will cover the nessessary translations. The u in the term is electron flow, so we can subsitute in

$$u_e = -\frac{J}{q_s n_s} + u_i$$

where J is current density and u_i is ion flow. With this information we get

$$-\epsilon_{ijk} u_j B_k = -u \times B = (\frac{J}{q_s n_s} - u_i) \times B = \frac{J \times B}{q_s n_s} - u_i \times B$$

For the $\mathcal{P}$ (stress tensor) term, we need to find the divergence over the j index, since $\frac{\partial \mathcal{P}_{ij}}{\partial x_j} = \nabla \cdot \mathcal{P}$ (**fix notation if needed**). We can substitute in the equation of the tensor involving reynolds stress

$$\mathcal{P}_{ij} = P_{ij} + \rho u_i u_j$$

where $u_i u_j = \sum_j u_i u_j$ for $i, j = \{x, y, z\}$. Now we are left with

$$E_i + u_i \times B = \frac{J \times B}{nq} + \frac{1}{nq}\frac{\partial nmu_i}{\partial t} + \frac{\nabla \cdot P}{nq} + \frac{\nabla \cdot nmu_i u_j}{nq}$$

We can combine the time derivative of momentum and divergence of Reynolds stress using product rule, so

$$\frac{\partial nmu_i}{\partial t} = nm\frac{\partial u_i}{\partial t} + u_i \frac{\partial nm}{\partial t}$$
$$\nabla \cdot (nmu_i u_j) = u_i \nabla \cdot (nmu_j) + nmu_j \cdot \nabla u_i$$

Resulting in

$$nm(\frac{\partial u_i}{\partial t} + u_i \cdot \nabla u_j) + u_i(\frac{\partial nm}{\partial t} + \nabla \cdot (nmu_j)) = nm(\frac{\partial u_i}{\partial t} + u_i \cdot \nabla u_j)$$

since the second term equals 0 due to mass conservation $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0$. Now we can switch back to vector notation to look at the electric field as a whole and plug in our new expression for the electron inertia contribution, arriving at

$$E + u \times B = \eta J + \frac{J \times B}{nq} - \frac{\nabla \cdot P_e}{nq} + \frac{m}{q}[(\frac{\partial u_e}{\partial t} + u_e \cdot \nabla u_e)]$$

where the left hand side comes from the ideal MHD, the ηJ, $J \times B$, and P_e terms areknown as the resistive dissipation, hall and the pressure tensor terms, respectively(**EXPAND**).

In this computational demo we will start with the 5 moment simulation, which symplifies the pressure tensor down to a scalar. This is Eulers equations where the force is lorentz force (write eulers equations with general force, F=E+(VxB) which is 0 in an ideal plasma. In generalized ohms law, there are other terms, called nonideal terms, that cause reconnection) Then we will expand the tensor out and compute the 10 moment simulation. Normally, only including the resistive dissipation term causes reconnection to occur far too slowly to only be attributed to the physics. But the hall term and the divergence of the pressure tensor causes the simulations to reconnect much more faster, closer to real life phenomenon. (**Talk about timescale**) This can be explored in the Apendix.

2.3.1 5 moment models

As mentioned before, 5 moment models simplify the pressure tensor down to a scalar. The 5 moment model is in fact a generalization of the Hall-MHD model (taking Ohm's law until the hall term) which can be found from the 5 moment model by approximating the speed of light to be infinite. By simplifying the pressure tensor, the 5 moment model is significantly easier to run, but can be an inaccurate model outside of the reconnection zone.

3 Setup Procedure - Windows

3.1 Install WSL 2

- follow directions on https://gkeyll.readthedocs.io/en/latest/install.html#note-on-the-windows-linux-subsystem-wsl (Ubuntu installation preferred, the following instructions may not work on other distributions)
  - If installing Ubuntu, download the "Ubuntu" or "Ubutu 20.04.4 LTS" entry from the Microsoft Store, **NOT** "Ubuntu 22... LTS"
- If installing CUDA, use https://docs.nvidia.com/cuda/wsl-user-guide/index.html#cuda-support-for-wsl2
- Gkeyll works best up to Ubuntu version 20.04.4, check by running `lsb_release -a` in the terminal and making sure that you don't have version 22+ installed. If you do, install "Ubutu 20.04.4 LTS" from the Microsoft store
- Install common libraries with `sudo apt install build-essential`
- Run

```
sudo apt update
sudo apt upgrade
sudo apt autoremove
```

- When using Gkeyll and Jupyter Lab in the future, open the ubuntu application from windows, or type "ubuntu" in another terminal application
  - Using windows terminal to do both of these is recommended for tabs, themes and easy configuration

3.2 Install Conda

- (Recommended) If you want to run Jupyter Lab in WSL 2 on your local machine's browser, install a conda distribution if you do not have one
  - Either install Miniconda (lightweight, easier installation) or Anaconda (far more packages, useful if you also need more features for other applications)
    - Miniconda: first download the installer with the command

    ```
    wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
    sudo chmod +x Miniconda3-latest-Linux-x86_64.sh
    ./Miniconda3-latest-Linux-x86_64.sh
    ```

    and follow the prompts. When you have installed Miniconda correctly, jump to Install Jupyter Lab below.
    - Anaconda: follow https://docs.anaconda.com/anaconda/install/linux/ up to step 9 and jump to Install Jupyter Lab below.
- If you want to install the standalone application, follow directions on https://github.com/jupyterlab/jupyterlab-desktop#download, then jump to "Add WSL 2

terminal in JL"

3.3 Install Jupyter Lab

- First, either open a new terminal or run `source ~/.bashrc`, then activate conda with `conda activate`
- Then update conda with,

```
conda update conda
conda update --all
```

- Create a new conda virtual environment with `conda create --name [NAME] python=3.8` with a name of your choice. If you choose to do this, you must also run `conda activate [NAME]` with the same name as before if you want to use anything you will install in the next steps. If you create the virtual environment correctly, you should see **(NAME)** text before the terminal prompt. Avoid using a newer python version than 3.8 due to compatability issues, but an older Python 3 version can work. (**ADD IMAGE**)
- Then install Jupyter Lab and dependencies with

```
conda install matplotlib jupyter jupyterlab pip git
conda install -c conda-forge jupyter_contrib_nbextensions
conda update --all
```

- Then launch Jupyter Lab by entering `jupyter lab --no-browser` in the terminal, and copying one of the URLs listed at the end of the output into a browser on your *local* machine. (**ADD IMAGE OF JL**)

3.4 Installing and compiling GkeyllZero

- In the folder you want the repository to be in (for best performance use something in /home/USER_NAME/..., **NOT** /mnt/...) type `git clone https://github.com/ammarhakim/gkylzero.git`
- Then, install the two main dependencies that G0 needs: cmake and gfortran
  - cmake: follow instructions in https://apt.kitware.com/ and after run `sudo apt install cmake`
  - gfortran: run `sudo apt install gfortran`
- Navigate to the gkylzero directory and follow instructions in the G0 README file at https://github.com/ammarhakim/gkylzero, starting from the "Building on your machine" section

3.5 Installing Postgkyl

- Follow the instructions at https://gkeyll.readthedocs.io/en/latest/install.html#installing-with-conda-preferred-for-non-developers, make sure that you are in the same conda environment you used previously (**add Note section to troubleshooting**)
- When using WSL 2 on Windows 10, by default GUI is not supported. In order to be able to view the pgkyl output plots, install VcXrv (recommended), or Xming if not possible. Follow the instructions at https://gkeyll.readthedocs.io/en/latest/install.html#additional-steps-required-on-windows-10
  - Then, install xterm on WSL by entering `sudo apt install xterm`

  - Whenever you need the GUI in the future, open XLaunch first and disable access control, the XLaunch icon should appear in the icon tray

3.6 Installing Luajit

- Normally, GkeyllZero uses C code to run the simulations, but in this module we will be using lua wrappers for the sake of more efficiently teaching the process. In order to run lua scripts we need to install Luajit.
  - First clone the fully featured Gkeyll version 2 on your machine (WSL if using windows) by entering `git clone https://github.com/ammarhakim/gkyl`
  - Navigate to `gkyl/install-deps` and enter `./mkdeps.sh --build-adios=no --build-luajit=yes --build-eigen=no`
    - This will install only luajit on your machine
  - Once the installation is done, remove the Gkeyll 2 cloned repository either through a file explorer or by typing `rm -r /path/to/directory` (for example `rm -r ~/gkyl` when using WSL). **WARNING:** make sure the path is correct or you may delete another folder unintentionally.
- Now we need to add the Gkeyll lua files to the PATH. For WSL users, can be done on the bash prompt by editing the .bashrc file (type `vim ~/.bashrc` or use another editor), and adding `export LUA_PATH="$HOME/gkylsoft/gkylzero/lib/?.lua;;` towards the end of the file. A similar method can be done on different shells as well.
- (Recommended) create an alias/shortcut to use luajit without typing the full path to the executable.
  - In the .bashrc file, add the line `alias run_lua='~/gkylsoft/luajit/bin/luajit'` if you installed luajit to the default directory.
- Then, restart the terminal with `source ~/.bashrc`. Now, whenever you need to run a lua script, just type `run_lua file.lua`, instead of the full path.

4 Setup Procedure - Macos

4.1 Install GkeyllZero

5 Running a 5 moment model

We will first start by going through a brief demo of running simulations and viewing outputs. In the 5 moment folder of this repository is a premade lua script to run a 5 moment simulation. - Navigate to this folder and use the luajit alias to run the simulation: `run_lua rt-5m-gem.lua` for example. - After the simulation is done, you will see in the terminal (**add terminal image**) and new output files will appear in the folder. # Plotting Initial Conditions The output of the fluid models will have multiple sets of files based on the lua instructions, each with a different file name ending (and then a number). The file used in this demo outputs the electron fluid moments end in "elc", ion fluid moments end in "ion" (**documentation typo**), and electromagnetic field files end in "field". The number after the suffix corresponds to the frame the file contains. The number of frames is also determined by a variable in the input file called `nFrame`. First we will start by plotting the initial conditions (files ending in "0") using postgkyl. - First make sure you are in the correct conda environment(and if using WSL, that XLaunch is open) - In that folder, type

```
pgkyl rt-5m-gem-elc_0.gkyl rt-5m-gem-field_0.gkyl rt-5m-gem-ion_0.gkyl select -c 3 plot --fix-a
```

```
- The `select -c 3` flag outputs only the z axis plots, while `-f0 --subplots` collates all thi
- Explore the pgkyl documentation at https://gkeyll.readthedocs.io/en/latest/postgkyl/main.html
```

(**Emphasize why these initial conditions are unstable to magnetic reconnection**) Now that we briefly showed how to view output through the command line, lets take a look at creating python scripts to do the postprocessing.

5.0.1 Add WSL 2 terminal in JL

Run `jupyter-lab --generate-config` Open generated file and add `c.NotebookApp.terminado_settings = { 'shell_command': ['C:\\Users\\d3mon\\AppData\\Local\\Microsoft\\WindowsApps\\ubuntu.exe'] }` to the end (change path) and restart JL

5.1 Install JL in WSL2

https://towardsdatascience.com/configuring-jupyter-notebook-in-windows-subsystem-linux-wsl2-c757893e9d69

6 Troubleshooting

- https://stackoverflow.com/questions/66744219/jupyter-lab-networkerror-running-on-wsl2
- If when typing xterm, you get an error similar to `xterm: Xt error: Can't open display: 192.168.128.1:0`, replace the `export display` line in the .bashrc (or equivalent) file with `export DISPLAY=$(route.exe print | grep 0.0.0.0 | head -1 | awk '{print $4}'):0.0`

[48]:
```
import os
os.chdir('/home/kasaiyuki/gkylnotebooks')
!ls
```

```
5-moment                   Reconnection-Intro.pdf  fluxPlotNotebook.ipynb
Reconnection-Intro.ipynb  Reconnection.gif
```

[49]:
```
%%bash
ls
```

```
5-moment
Reconnection-Intro.ipynb
Reconnection-Intro.pdf
Reconnection.gif
fluxPlotNotebook.ipynb
```

[50]:
```
!ipython nbconvert --to html Reconnection-Intro.ipynb
```

```
usage: ipython [-h] [--debug] [--show-config] [--show-config-json] [--quiet]
               [--init] [--autoindent] [--no-autoindent] [--automagic]
               [--no-automagic] [--pdb] [--no-pdb] [--pprint] [--no-pprint]
               [--color-info] [--no-color-info] [--ignore-cwd]
```

```
                  [--no-ignore-cwd] [--nosep] [--autoedit-syntax]
                  [--no-autoedit-syntax] [--simple-prompt] [--no-simple-prompt]
                  [--banner] [--no-banner] [--confirm-exit] [--no-confirm-exit]
                  [--term-title] [--no-term-title] [--classic] [--quick] [-i]
                  [--log-level TerminalIPythonApp.log_level]
                  [--profile-dir ProfileDir.location]
                  [--profile TerminalIPythonApp.profile]
                  [--ipython-dir TerminalIPythonApp.ipython_dir]
                  [--config TerminalIPythonApp.extra_config_file]
                  [--autocall TerminalInteractiveShell.autocall]
                  [--colors TerminalInteractiveShell.colors]
                  [--logfile TerminalInteractiveShell.logfile]
                  [--logappend TerminalInteractiveShell.logappend]
                  [-c TerminalIPythonApp.code_to_run]
                  [-m TerminalIPythonApp.module_to_run]
                  [--ext TerminalIPythonApp.extra_extensions]
                  [--gui TerminalIPythonApp.gui]
                  [--pylab [TerminalIPythonApp.pylab]]
                  [--matplotlib [TerminalIPythonApp.matplotlib]]
                  [--cache-size TerminalInteractiveShell.cache_size]
                  [extra_args [extra_args …]]
ipython: error: unrecognized arguments: Reconnection-Intro.ipynb
```

[ ]: