Week 7 - 8* Report

Sam Frederick

* Fall Break (Mon. - Tues.)

Problem Conditioning

Haskell_B-Field_Condition_Numbers.pdf:

- Investigates:
  - Structure of B-field
    - B_r, B_θ, B_φ from $(0 < r < 2)$
  - Relative condition numbers for B_r, B_θ, B_φ over the radial domain
- Verifies:
  - Boundary conditions Kuhn defined for the B-Field, but comes about these results in a more qualitative, mathematically verifiable way.
  - Well-behaved nature of B-field components

The relative condition number: κ

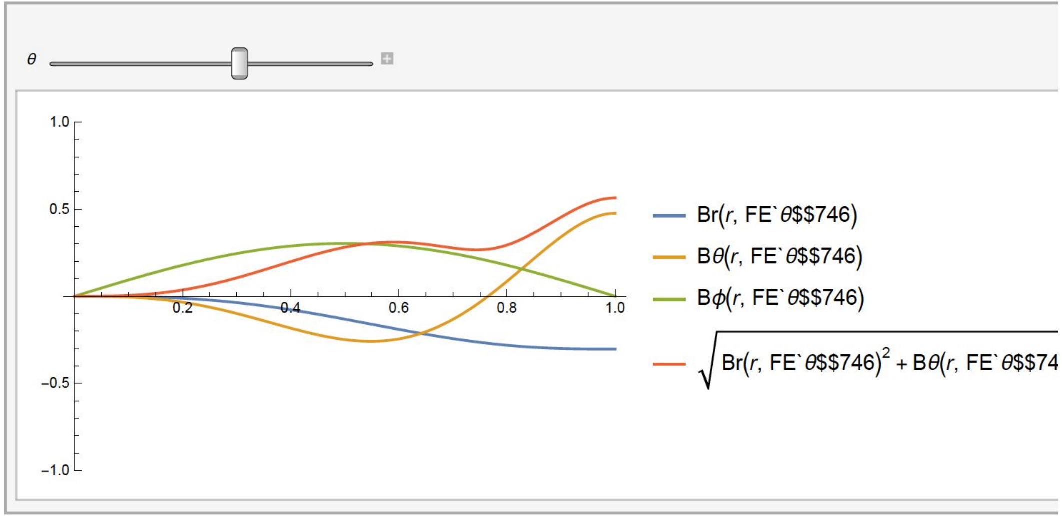
- Quantifies the relative change in the output of some function f(x) and the arbitrary variable x:
    - $\kappa(x) = |x * f'(x) / f(x)|$
    - Asymptotic behavior or large values of κ(x) signify large relative error in a certain region.
        - For example, if $\kappa(x) = 10^8$ and some computed value based on f(x) reports 16 digits such that machine error is $O(10^{-16})$, we should only trust digits out to (machine epsilon*κ(x)) ~ 10^{-8}.

$$\texttt{Br[r\_, }\theta\texttt{\_] :=} \frac{\texttt{Bmax * Cos[}\theta\texttt{]}}{\pi\left(\pi^2 - 6\right)}\left(\pi^3 r^3 + 3\left(\pi^2 r^2 - 2\right)\texttt{Sin[}\pi * r\texttt{]} + 6\,\pi * r * \texttt{Cos[}\pi * r\texttt{]}\right)$$

$$\texttt{B}\theta\texttt{[r\_, }\theta\texttt{\_] :=} \frac{\texttt{Bmax * Sin[}\theta\texttt{]}}{2\,\pi\left(\pi^2 - 6\right)}\left(-2\,\pi^3 r^3 + 3\left(\pi^2 r^2 - 2\right)\left(\texttt{Sin[}\pi * r\texttt{]} - \pi * r * \texttt{Cos[}\pi * r\texttt{]}\right)\right)$$

$$\texttt{B}\phi\texttt{[r\_, }\theta\texttt{\_] :=} \frac{\texttt{Bmax * Sin[Pi * r] * Sin[}\theta\texttt{]}}{\texttt{Pi}}$$

```
Manipulate[Plot[{Br[r, θ], Bθ[r, θ], Bϕ[r, θ], Sqrt[Br[r, θ]^2 + Bθ[r, θ]^2]},
  {r, 0, 1}, PlotRange → {-1, 1}, PlotLegends → "Expressions"], {θ, 0, π}]
```



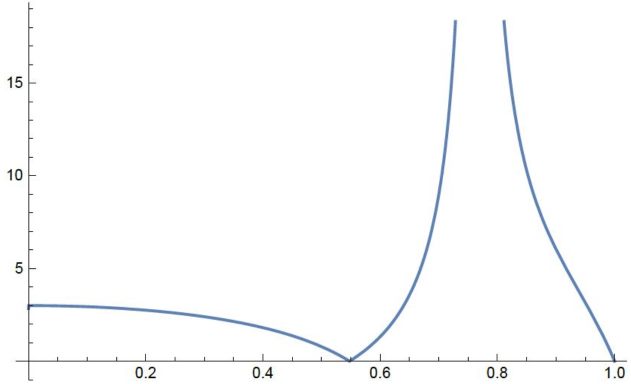
B-Field components for r < 1.0

In[∘]:=
```
KBθ = Simplify[Abs[r * DBθ / Bθ[r, θ]]]
```

Out[∘]=
$$3\,\pi^3 \left| \frac{r^3 (-2\cos(\pi r) + \pi r \sin(\pi r) - 2)}{2\,\pi^3 r^3 + 3\,\pi \left(\pi^2 r^2 - 2\right) \cos(\pi r)\, r + \left(6 - 3\,\pi^2 r^2\right) \sin(\pi r)} \right|$$

In[∘]:=
```
Plot[KBθ, {r, 0, 1}]
```

Out[∘]=



In[∘]:=
```
For[r = .7, r < .9, r += 1 * 10^-7, If[KBθ > 1 * 10^7, Print[Row[{r}]]]]
```

0.765052

0.765052

We see that when we scan over our $\kappa_{B\theta}$ relative condition value function with a δ step of 1E-7, the domain over which the function is poorly behaved, i.e when the condition value $\kappa \geq 10^7$, occurs about r = 0.765052 ± δ. This shows quantitatively that despite this discontinuity, our function B_θ **is well behaved over our computational domain**, as our grid spacing is much larger than 2δ.

Condition_Grid_Analysis.py

- Analyzes radial grid points surrounding a specified asymptote resulting from the condition number for a particular function with respect to radius in the computational domain.

- A tolerance is specified to see whether there exist any grid points significantly close to the asymptote which may result in poorly behaved computation around such points. The tolerance is determined by the threshold magnitude of the condition number below which we deem "well behaved" and vice versa. In numerical analysis, this threshold magnitude is of the order 10^8 or equivalent sufficiently large values.

- Here, we intentionally set the tolerance fairly low (1e-2) to show that even for such low tolerance, we do not expect poorly behaved points within our computational domain.

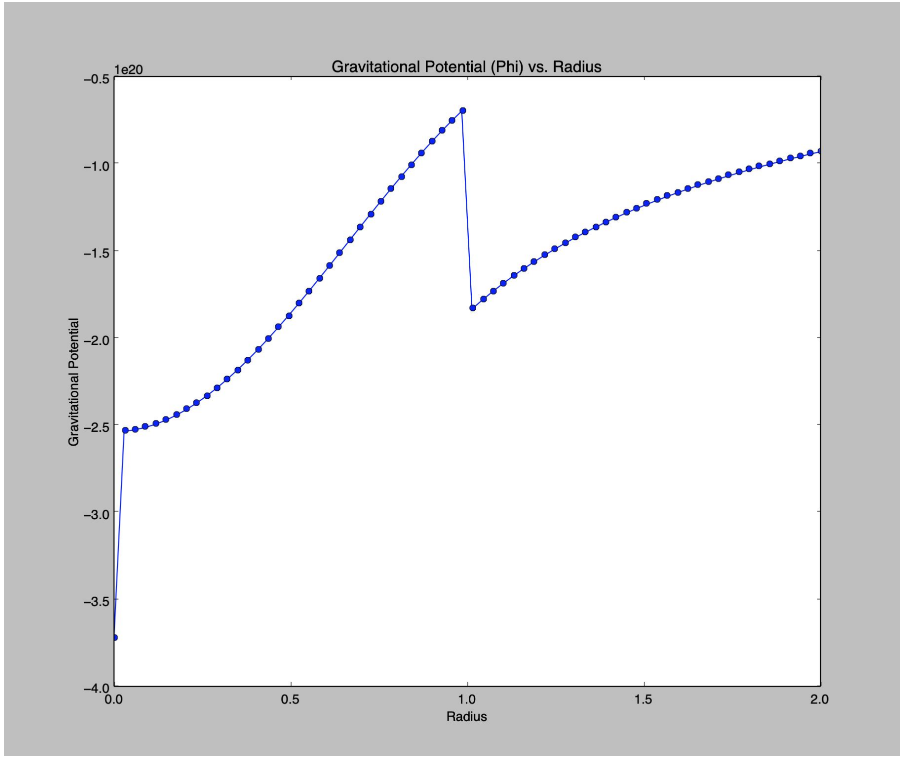
Condition_Grid_Analysis.py

```
rgridmin = 0
rgridmax = 2
divisions = 70
asymp = 0.765052
tolerance = 1e-2
count = 1


vals = np.linspace(rgridmin,rgridmax,divisions)
# print (vals)

for point in vals:
    if abs(point-asymp) < tolerance:
        if count == True:
            print ("The following values for r may blow up:")
            count = False
        print ("r = %(point)f" % {"point": point })

if count == True:
    print ("No points are sufficiently close to the asymptote; " \
    "the computational domain is well behaved.")
```

```
>>  No points are sufficiently close to the asymptote; the computational domain is well behaved.
```

Grav_Pot.py

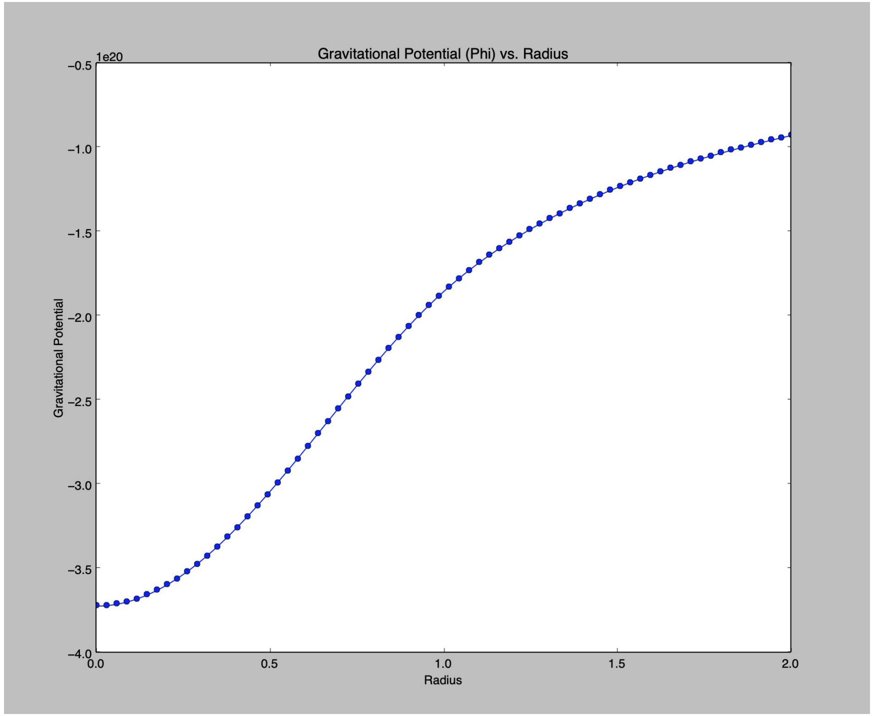
- Graphs the gravitational potential for the computational domain given equations specifying the potential in three main radial regions: r = 0, 0 < r < 1, and r > 1.
- Initially copied and pasted constants and equations as they were from C code to see whether the potential was constant and smooth (nope!).



Gravitational potential φ prior to correction (0 < r < 2)

Grav_Pot.py: The Culprit

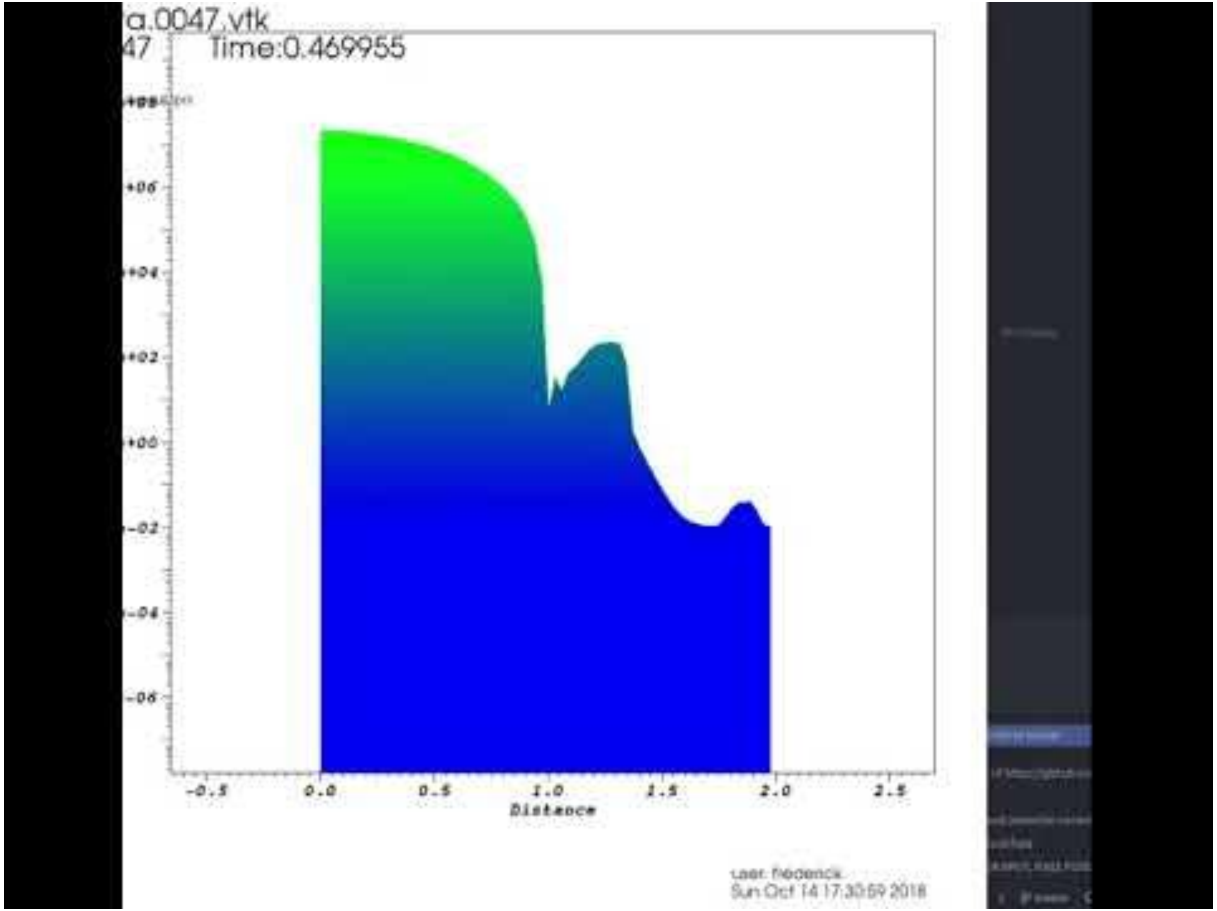
- A while ago, I set the stellar mass to be 1.0E33 which was eventually updated to 2.785*10^33 (1.4 $M_\odot$). However, I didn't update the variable RPOT (gravitational potential at r = R: GM/R) so that φ interior to the star was off by a scalar value in that region. After updating the value of RPOT with the new mass, graphing the gravitational potential shows a smooth, continuous plot.

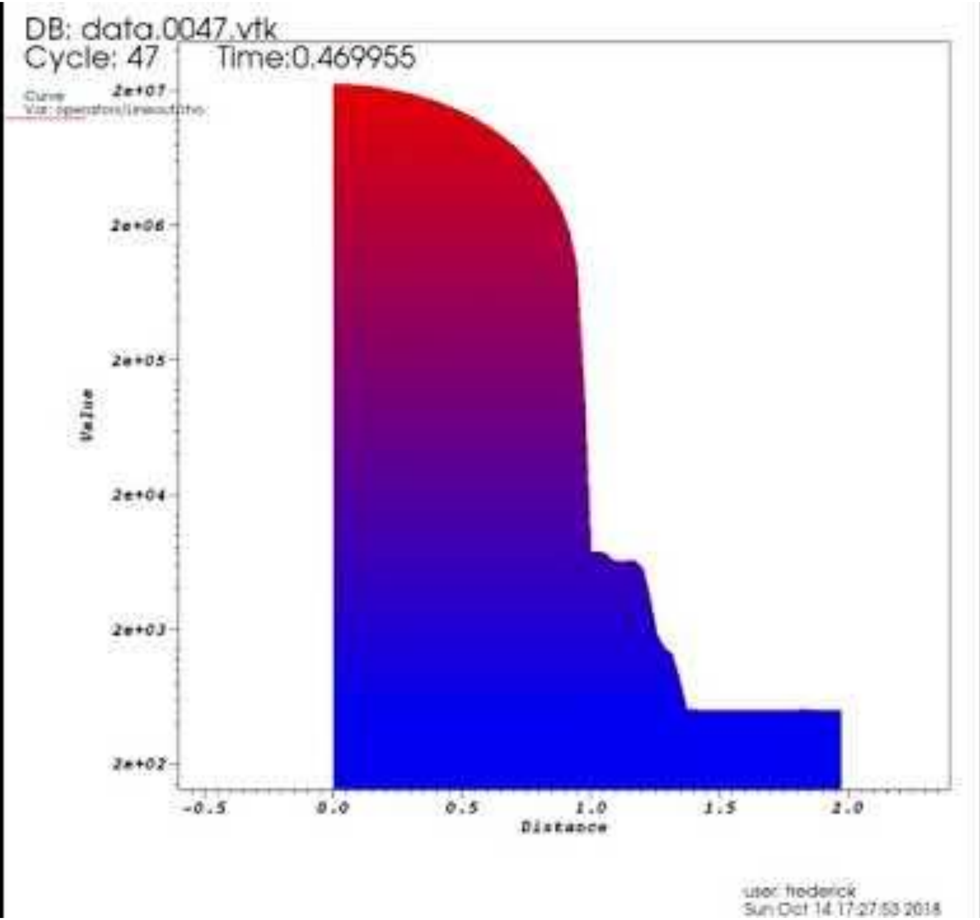


Gravitational potential φ post correction (0 < r < 2)

Results after fixing φ issue

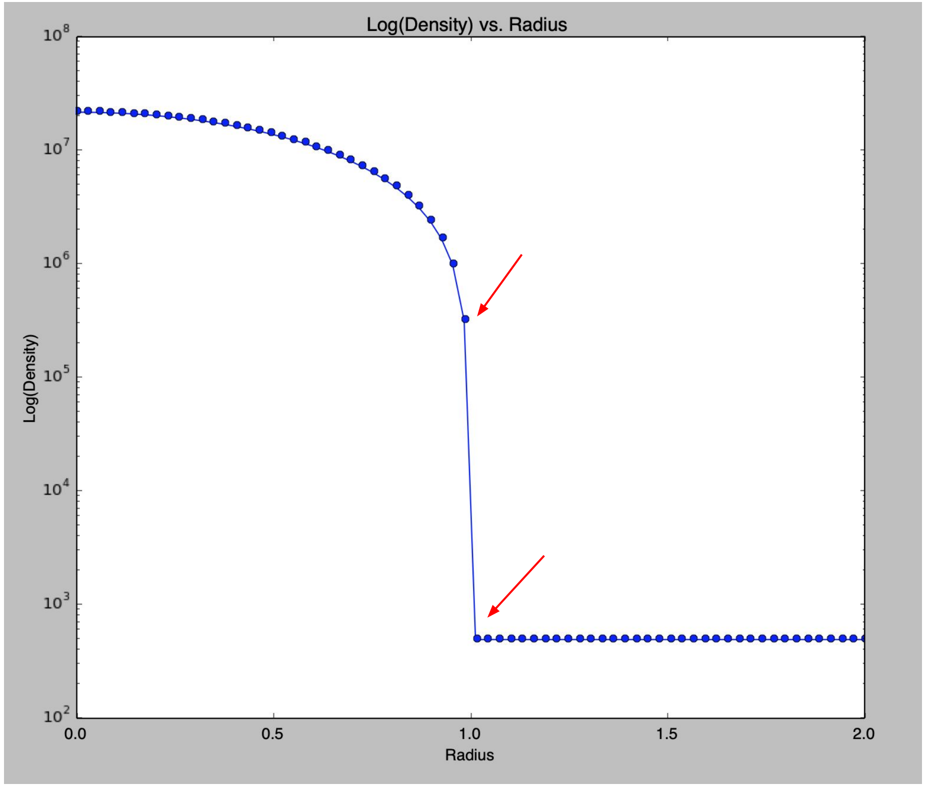
- Allowed me to clean up some of the restrictions I had put in place in the userdefboundary function.
  - Remove arbitrary $r > 1.9$ condition
  - Kept density/pressure floor averaging
  - Time independent conditions at $r = 0$ and $r = 2.0$ remain
- Wasn’t entirely happy with results..
  - Concluded that averaging for boundary floors was poorly modeling pressure/density.


a.0047.vtk
47 Time:0.469955
+06
+04
+02
+00
-02
-04
-06
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
Distance
Sun Oct 14 17:30:59 2018


DB: data.0047.vtk
Cycle: 47
Time:0.469955
Curve
2e+07
2e+06
2e+05
2e+04
2e+03
2e+02
Value
-0.5
0.0
0.5
1.0
1.5
2.0
Distance
user: frederick
Sun Oct 14 17:27:53 2018

Introducing Non-uniform Grid Spacing

- So far, the computational domain has been modeled radially by 70 equally spaced points from r = 0 to r = 2.0.
- *Density_And_Pressure_Analysis.py*
  - Creates log plots for density and pressure over the computational domain.
    - Noticed that the jump between adjacent grid-points is nearly 300,000 for density
      - Results in considerable diffusion and computational inaccuracy at abrupt cutoff

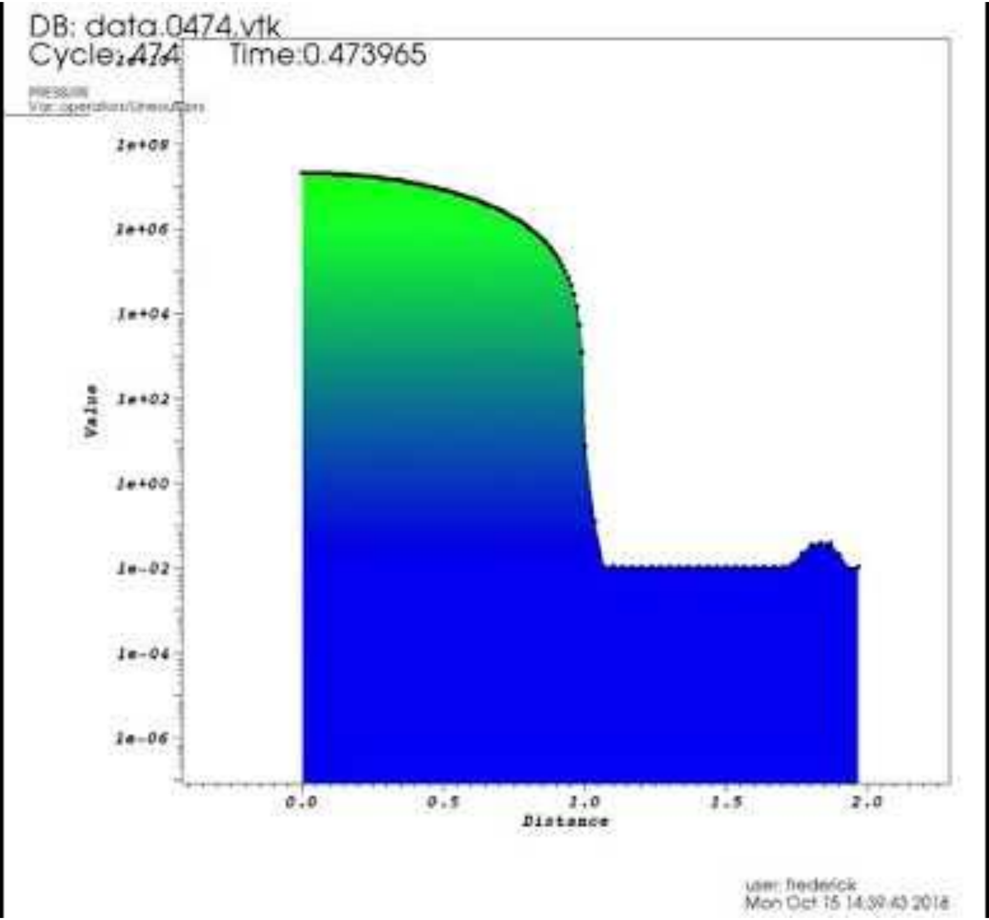


Introducing Non-uniform Grid Spacing

- This can be fixed by imposing a non-uniform radial grid in the computational domain.
    - From `pluto.ini:`

Two consecutive grids
Start at 0.0
100 Grid points
Uniform grid spacing
End / Start at 1.0
30 Grid points
Uniform grid spacing
End at 2.0

```
[Grid]

X1-grid    2    0.0   100    u    1.0 30 u 2.0
X2-grid    1    0     20    u     3.14159265359
X3-grid    1    0     20     u    3.14159265359
```


DB: data.0474.vtk
Cycle: 474 Time:0.473965
Value
1e+08
1e+06
1e+04
1e+02
1e+00
1e-02
1e-04
1e-06
0.0
0.5
1.0
1.5
2.0
Distance
user: frederick

Goals for Next Week:

- Optional logarithmic grid spacing; test whether using log spacing prevents pressure propagation.
- With these results, I’m not getting any negative density/pressure/energy errors, so I feel even more confident to move on to including the B-field.
    - Impose proper boundary conditions
    - Test how density impacts behavior of simulation (motivation for Kuhn implementing vacuum density)
        - Test simulation over only $r < 1$ to avoid density issues?
    - Stability with B-field