Transient Heat Equation in Cylindrical Coordinates

Justin Miller
APMA2822b

Motivation:

- Background in power transmission
- Wire Overheating: Sag Clearance & Annealing

- Knowing wire temperature under different loading is valuable to optimal grid operation

- Emergency Loading Situations
    - Abrupt high-current power flow to fill demand until the conductor reaches Max. Rated Temp.

Equations:

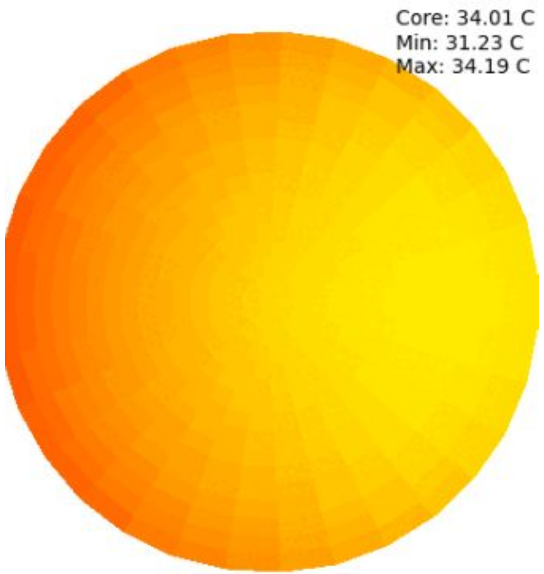
- Fourier-Biot Equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{e}_{\text{gen}}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t}$$

- Cylindrical Version

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial T}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right) + \dot{e}_{\text{gen}} = \rho c\frac{\partial T}{\partial t}$$

- Note: Using a 2D Cross Section of the above equation (dT/dz = 0)



Finite Difference Scheme:

- Spatial Derivatives Appx. Using Central Difference
- Time Derivative Appx. Using Backward Difference

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial T}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right) + \dot{e}_{\text{gen}} = \rho c\frac{\partial T}{\partial t}$$

$O(\Delta x^2)$ centered difference approximations:

$$f'(x): \quad \{f(x+\Delta x) - f(x-\Delta x)\}/(2\Delta x)$$

$$f''(x): \quad \{f(x+\Delta x) - 2f(x) + f(x-\Delta x)\}/\Delta x^2$$

$O(\Delta x^2)$ backward difference approximations:

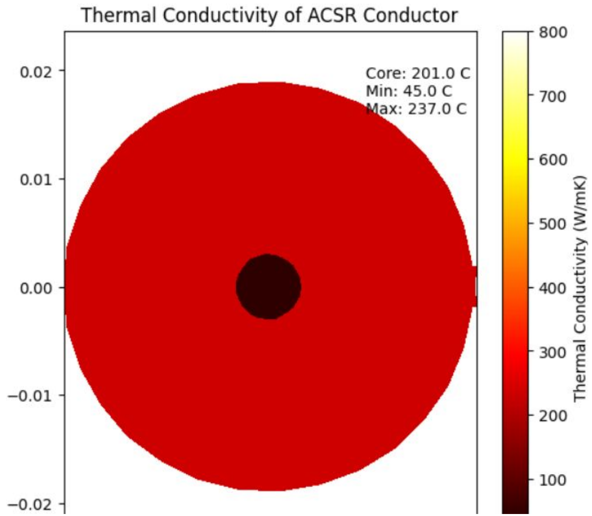
$$f'(x): \quad \{3f(x) - 4f(x-\Delta x) + f(x-2\Delta x)\}/(2\Delta x)$$

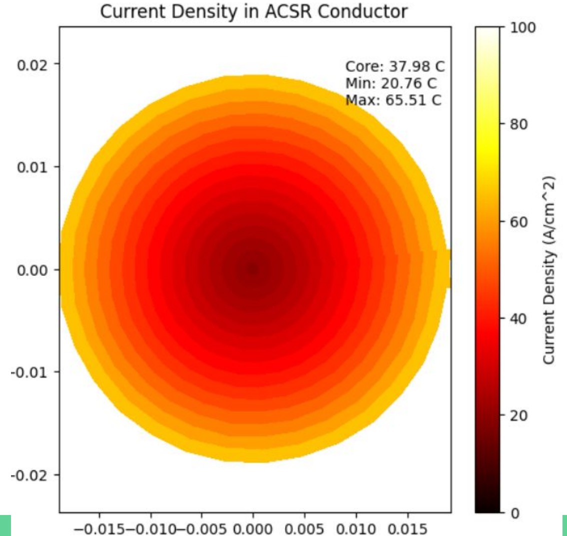
- Challenges:
  - Singularity Handling
  - Uneven Domain (much more time resolution required for this problem of interest)

- Benefits
  - Stability
  - Ease of calculation / parallelization

Model Parameters

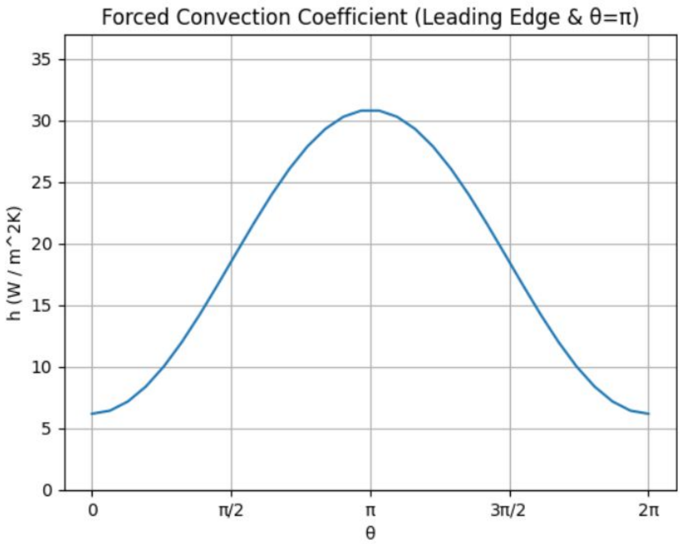
- “ACSR Linnet” Transmission Wire
- 1.84 cm radius; Aluminum w/ Steel Core
- K, alpha material properties
- Joule Heating based off of Current Load
  - Adjusted for skin effect



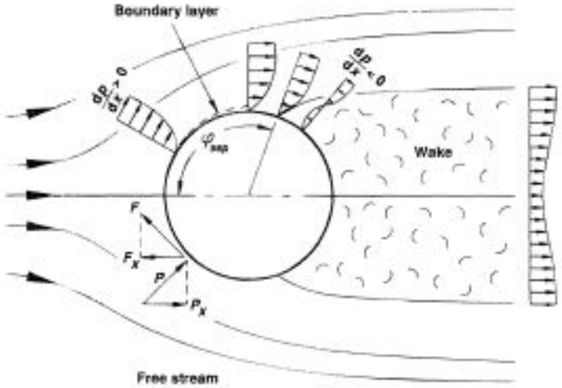


Boundary Conditions

- Forced Convection:
  - 2 m/s perpendicular wind, constant direction
  - 25 °C Ambient Temperature°
  - Results in **asymmetric** convection pattern
    - Key reason for FD model



- Periodic Boundary Condition
  - Ensure wraparound in theta dir



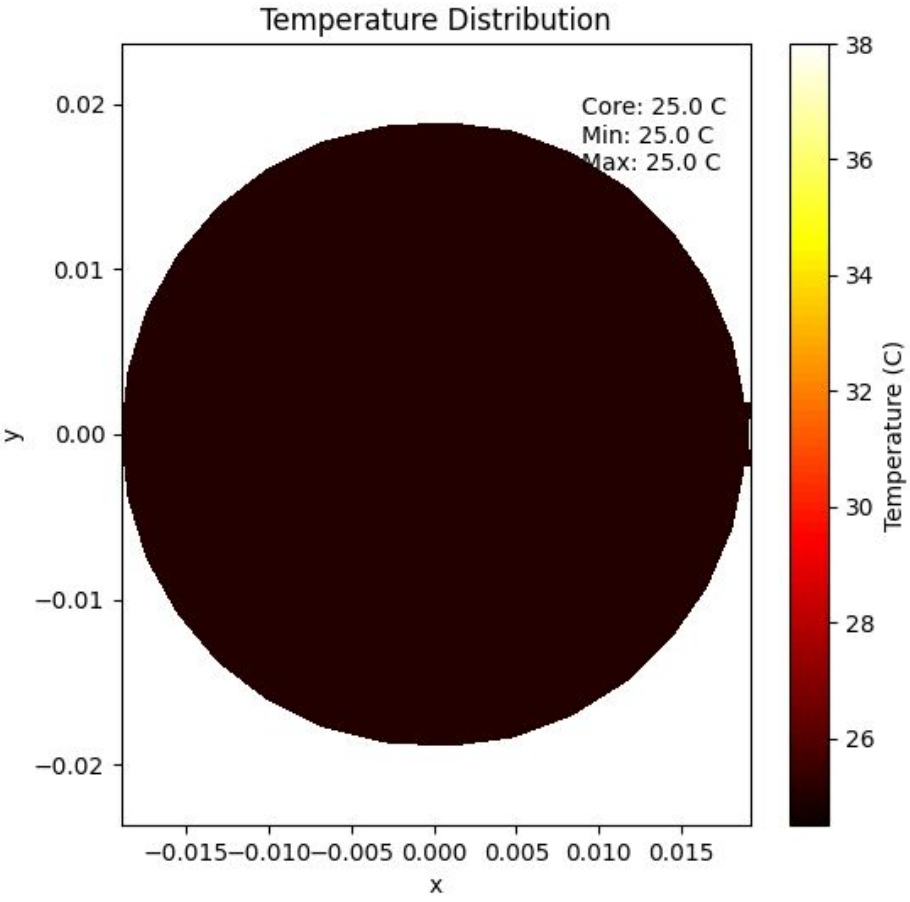
Domain Parameters

- N_r = 32
- N_theta = 64
- N_t = 512
    - delta_t = 10 seconds
    - Majority of resolution is in t direction to simulate ~ an hour and a half of transient heating

- Initial Conditions:
- Power line at ambient temperature (25 C)
- Internal Heat Generation started at first timestep.

Sequential Runs: Visualizing

- Transient heating of wire shown
    - Takes about an hour to reach steady state
- Uneven Convection Apparent
- Overall Success
    - Smooth heat transfer over singularity
    - Convection at surface as expected
    - Symmetric across horizontal place
    - No Halo remnants in parallel code



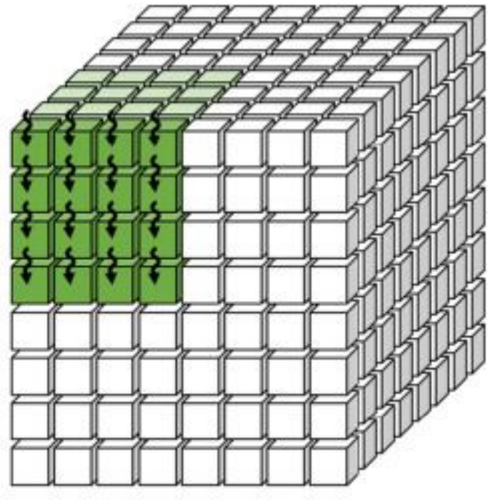
Misc. Notes: Singularity Handling & Convergence:

- Singularity at i=0 (R=0)
- Effectively skip cell, take difference formulas from i+1 on the current theta and the theta across the center
- Periodicity: treat last element of theta as before first element

- L1 Convergence; Tolerance: 1e-6
- 42,000 iterations required, consistent among all methods
- Stable, monotonically decreasing

Parallelizing: Shared Overview

- Create FiniteDifferenceKernel() to run every iteration
- 3D Simulation: Splitting into 3D Threads / Blocks
- Allocating & initializing data to device using Memcpy()
- Execute kernel on each iteration.
  - Sync after run
  - Check convergence if desired
  - Memcpy() back to host when done



Parallelizing: Shared Optimization

- Hipcc built in optimization -O2

- Thread Management
  - Recall domain size is N_R:32, N_theta:64, N_t:512
  - By splitting our threads into (2, 4, 16) we can keep our grid sizes nice and even

Parallelizing: Shared Performance

- Run on MI2104x Partition, Single GPU

- MI210: Stated 22.6 TF & up to a 1.6 TB/s BW

- 53.9% Theoretical Gf/s
- 54.0% Theoretical Gb/s

Kernel Computation:
for each i * j * k: // **Iteration**
3 store;
8 loads
60 FLOP

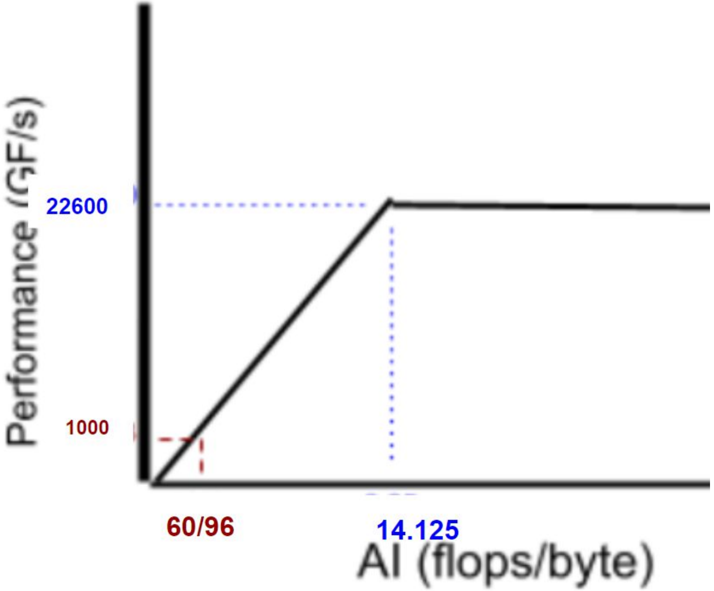
Per Iteration: ijk x 8 x 11 = 88 ijk bytes,
ijk x 60 FLOP = 60 ijk FLOP,
AI = 60 / 88
Performance = 1091 Gf/s

Gf / s = 587.986
Gb / s = 862.380

Tolerance	Runtime (s)	Iterations	Time / Iteration (µs)	Bytes / Iteration	Gb/s	FLOP / Iteration	Gf/s
1e-6	3.946	37,000	107	92,274,688	862.380	62,914,560	587.986

Parallelizing: Shared Performance (cont.)

- Performance roofline plot provided
- % of theoretical efficiency is a bit worse than expected
- Practically acceptable



Tolerance	Runtime (s)	Iterations	Time / Iteration (µs)	Bytes / Iteration	Gb/s	FLOP / Iteration	Gf/s
1e-6	3.946	37,000	107	92,274,688	862.380	62,914,560	587.986

Parallelizing: Distributed (offloading onto GPUs)

- Executing same kernel distributed on 4 MI210 GPUs (using 4 MPI Tasks)

- Using the rankID of each task, set a corresponding GPU device to the CPU

- Leveraging uneven domain to simplify and speed up halo exchange
    - **36 x 64 x 512 simulation: splitting this into 4 GPUs the lowest data transfer is in the t plane**
    - **Backwards difference in time plane simplifies it further, as buffer is only needed on one side**

Parallelizing: Distributed Optimization

- Convergence Checks
  - Limiting convergence checks to every N iterations (ended up with 1000) to limit reductions
- Halo Exchange
  - Also limiting halo exchange to every N iterations as this is expensive. Slightly increases iterations needed for convergence but results in overall runtime improvements.
- Singularity
  - There are no extra considerations required to handle the singularity in the distributed model as we are splitting only on the time plane

Overall Performance:

- Python Implementation, C++ Sequential Implementation, Shared, Distributed

	Time / Iteration (µs)	**Evaluated Runtime (s)**
Python Prototype	94,391	65 Min.
C++ Sequential	40,918	28 Min.
C++ GPU Shared	107	3.95 Sec.
C++ GPU Offloaded (4 GPUs)	89	3.30 Sec

Issues

- Performance Issues: Particularly on shared memory
- Unsolved: Joule Heating Stability. When input current is too high the simulation diverges. Likely a bug in the current distribution.
- Time Prioritization

Questions?