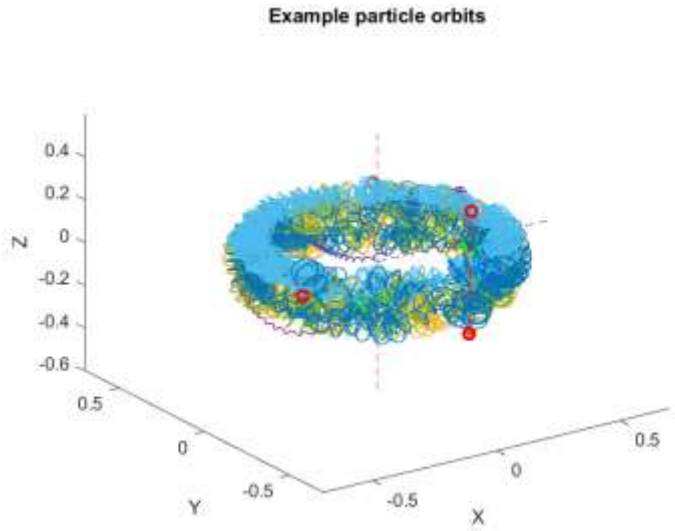
Fast ion orbit modeling using a Chebyshev representation for the magnetic potential.

Maxwell Rosen, Leonid Zakharov

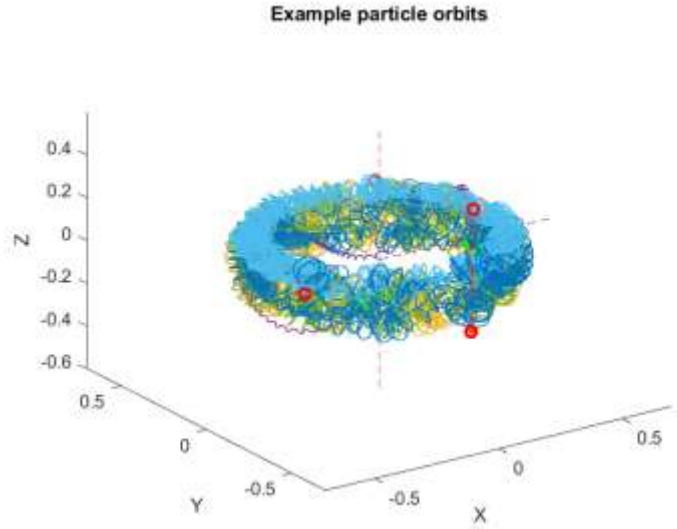
SULI Summer 2020



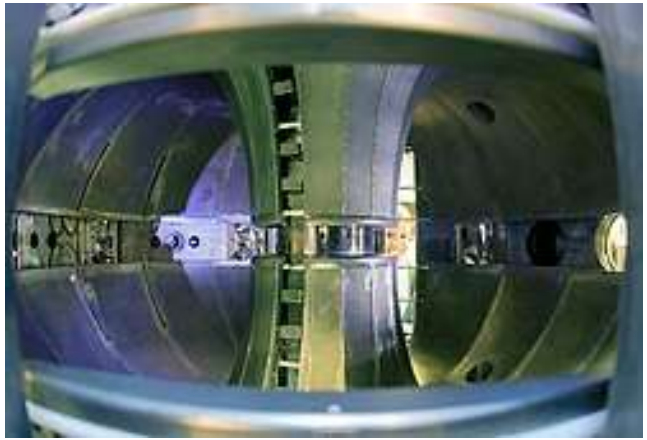
This work was made possible by funding from the Department of Energy for the Summer Undergraduate Laboratory Internship (SULI) program. This work is supported by the US DOE Contract No. DE-AC02-09CH11466

Outline

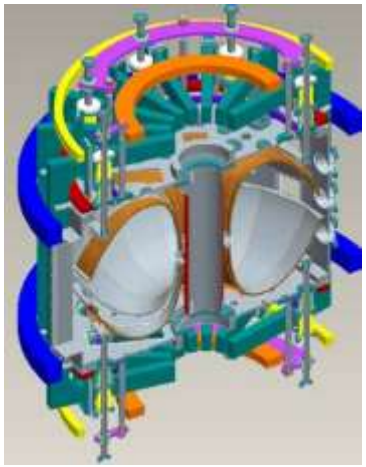
- Information about the Lithium Tokamak Experiment.
- Chebyshev polynomials.
- The Chebyshev approximation in 1D and 2D.
- Computational methods.
- Simulation results.



Lithium Tokamak Experiment (LTX)



- Studies the plasma-material interaction of liquid Lithium (Li).
- B = 0.3 T.
- Neutral beam injection heating.
- Fast ion means energies > 10 keV.

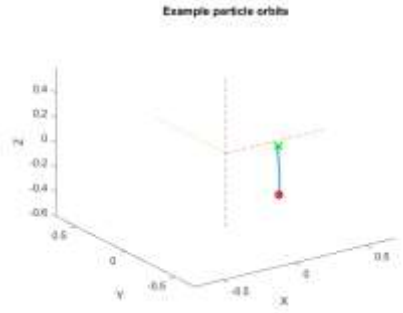


Wikipedia

- Why LTX is important?
  - ITER has a tungsten wall, which is high-Z (74) material.
  - LTX walls are coated by low-Z (3) Li to reduce recycling.
- **Recycling** - when cool particles are re-introduced back to main plasma.

Why are fast ions important?

- Fast ions heat the plasma and supply particles to its core.
- They may escape either at first orbits or due to collisions with the plasma.
- Simulating the ion losses at collision time scales (> 10 ms) is a challenge.
- Past simulations used a bi-cubic spline approximation for calculations of magnetic field
  - Issues at large times because of discontinuous first derivative at mesh boundary.
- A Chebyshev approximation is developed here to resolve this issue.
- **Goal: Simulate particle losses in a fast way**



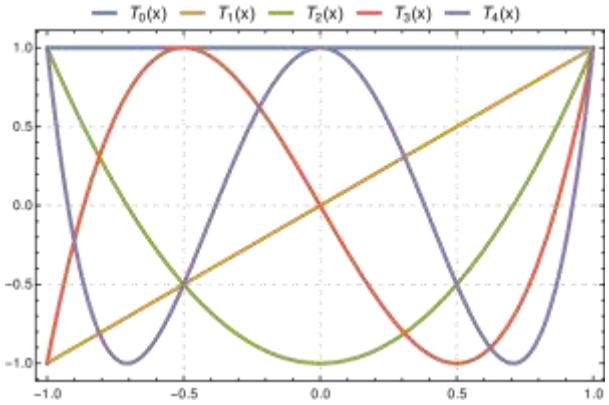
1D Chebyshev Polynomials

Definition of polynomials

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$
$$T_5(x) = 16x^5 - 20x^3 + 5x$$
$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1.$$

$$T_n(x) = \cos(n \arccos x).$$

$$T_{n+1} = 2xT_n - T_{n-1}, \quad n \geq 1.$$



- Values range from -1 to 1 on the domain [-1,1].
- Complete orthonormal system.

Orthogonality identity

$$\sum_{k=1}^{n} T_i(\bar{x}_k) T_j(\bar{x}_k) = \begin{cases} 0 & i \neq j \\ n/2 & i = j \neq 0 \\ n & i = j = 0 \end{cases}.$$

Chebyshev nodes

$$\bar{x}_k = \cos \frac{\pi(2k-1)}{2n}, \quad k = 1, 2, 3, \ldots, n.$$

1D Chebyshev Approximation of a Function

$$f(x) \approx \sum_{i=0}^{N-1} c_i T_i(x),$$

$$c_0 = \frac{1}{N} \sum_{k=1}^{N} f(\bar{x}_k),$$

$$c_i = \frac{2}{N} \sum_{k=1}^{N} f(\bar{x}_k) T_i(\bar{x}_k),$$

$$\bar{x}_k = \cos \frac{\pi(2k-1)}{2n}, \quad k = 1, 2, 3, \ldots, n.$$

- Approximate a function as a sum of the first N Chebyshev polynomials.
- The coefficients c_i are determined through summing the overlap at the Chebyshev nodes.

2D Chebyshev Approximation

$$f(x,y) \approx \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} c_{ij}T_i(x)T_j(y).$$

$$c_{00} = \frac{1}{MN}\sum_{k=1}^{M}\sum_{\ell=1}^{N} f(\bar{x}_k,\bar{y}_\ell),$$

$$c_{i0} = \frac{2}{MN}\sum_{k=1}^{M}\sum_{\ell=1}^{N} f(\bar{x}_k,\bar{y}_\ell)T_i(\bar{x}_k),$$

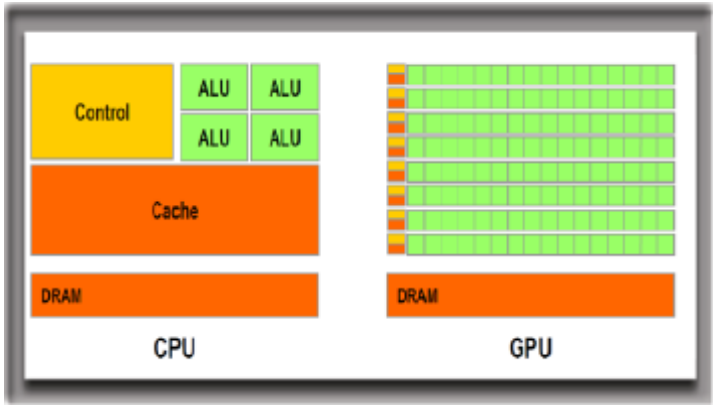
$$c_{0j} = \frac{2}{MN}\sum_{k=1}^{M}\sum_{\ell=1}^{N} f(\bar{x}_k,\bar{y}_\ell)T_j(\bar{y}_\ell),$$

$$c_{ij} = \frac{4}{MN}\sum_{k=1}^{M}\sum_{\ell=1}^{N} f(\bar{x}_k,\bar{y}_\ell)T_i(\bar{x}_k)T_j(\bar{y}_\ell).$$

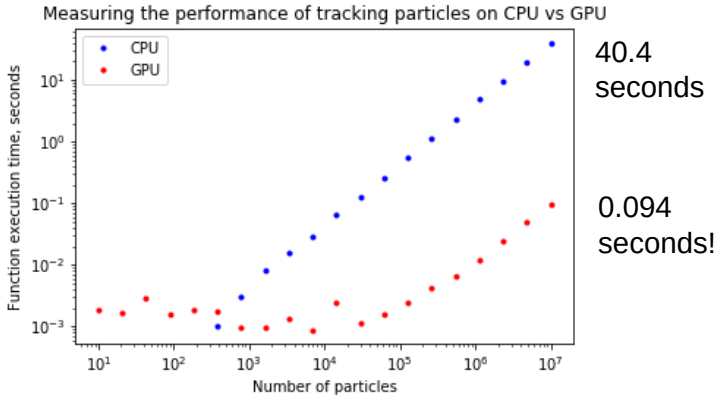
- This extends to 2D, where the functions are multiples of the Chebyshev polynomials.
- The same orthogonality identities hold, so the coefficients c_{ij} are determined the same way.
- Our code evaluates a 2D approximation of the magnetic potential.
  - First partial derivatives are used to calculate the magnetic field.

Computational Methods

- GPU is much faster through **massive parallelism**.
- Useful for calculating the Chebyshev representation for > 100,000 particles.



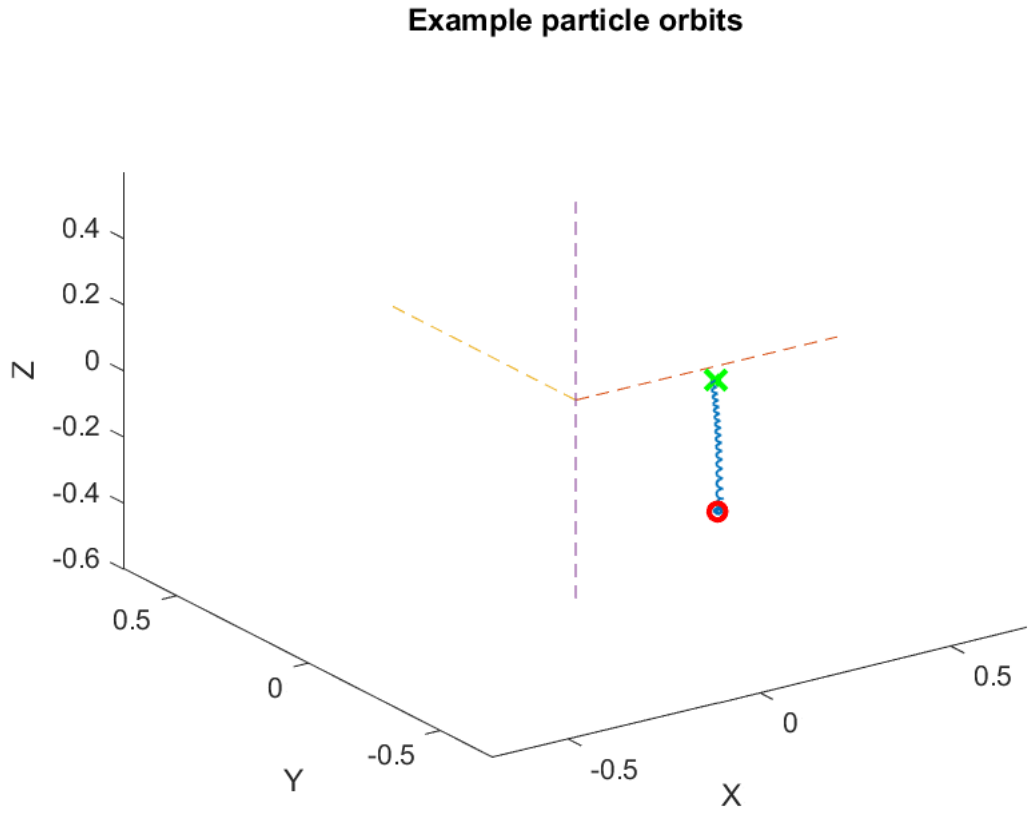
- Varying particle number and measuring time to evaluate the magnetic field.
- Initial offset is due to loading variables onto GPU.



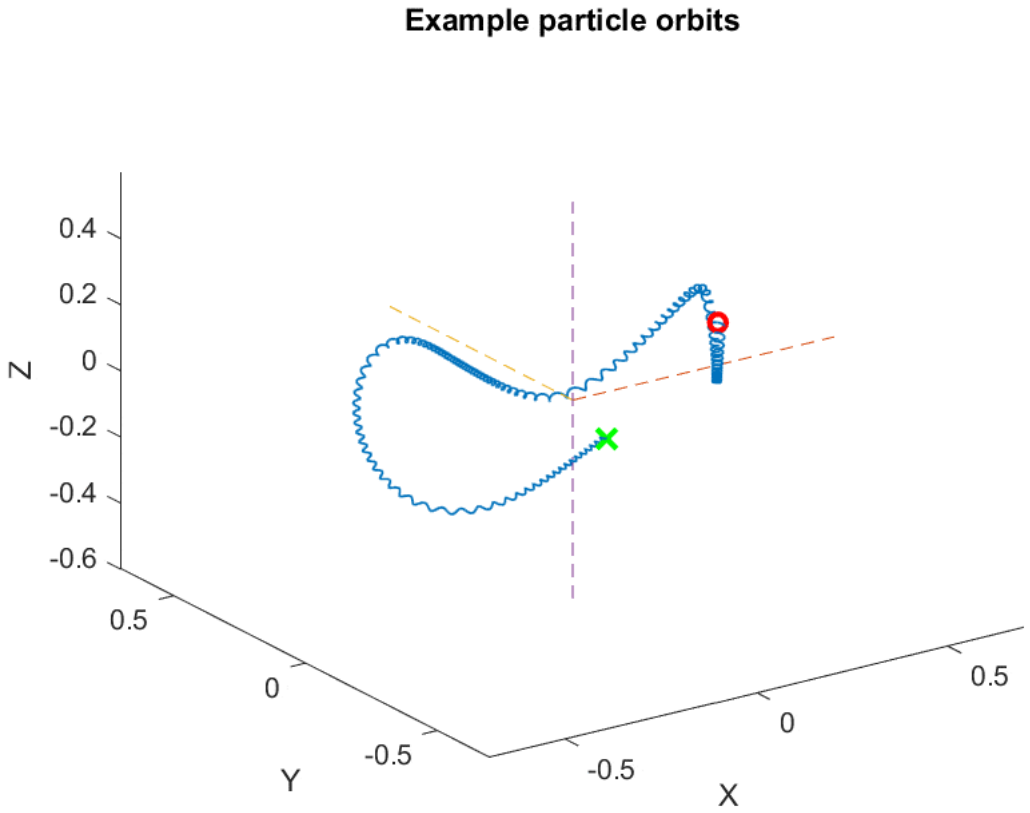
Algorithm structure

- Codes written in C and CUDA C.
- Use the Soloviev solution to the Grad-Shafranov equation to find the magnetic potential.
- Approximate the magnetic potential using the Chebyshev representation and find the first partial derivatives with respect to r and z.
- For each particle,
  - Calculate the magnetic fields using the Chebyshev representation.
  - Generate the time derivatives of particle position and velocities.
  - Use Runge-Kutta-4 algorithm for full orbit calculations.

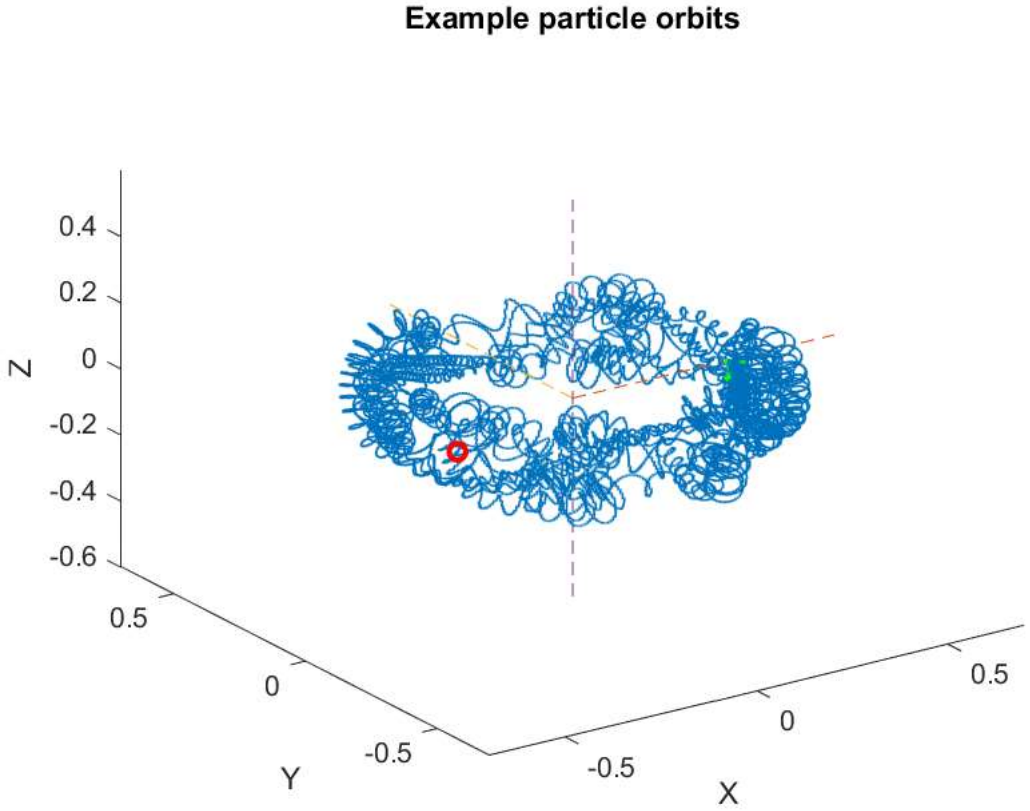
Results: Single particle orbits 1/6 velocity



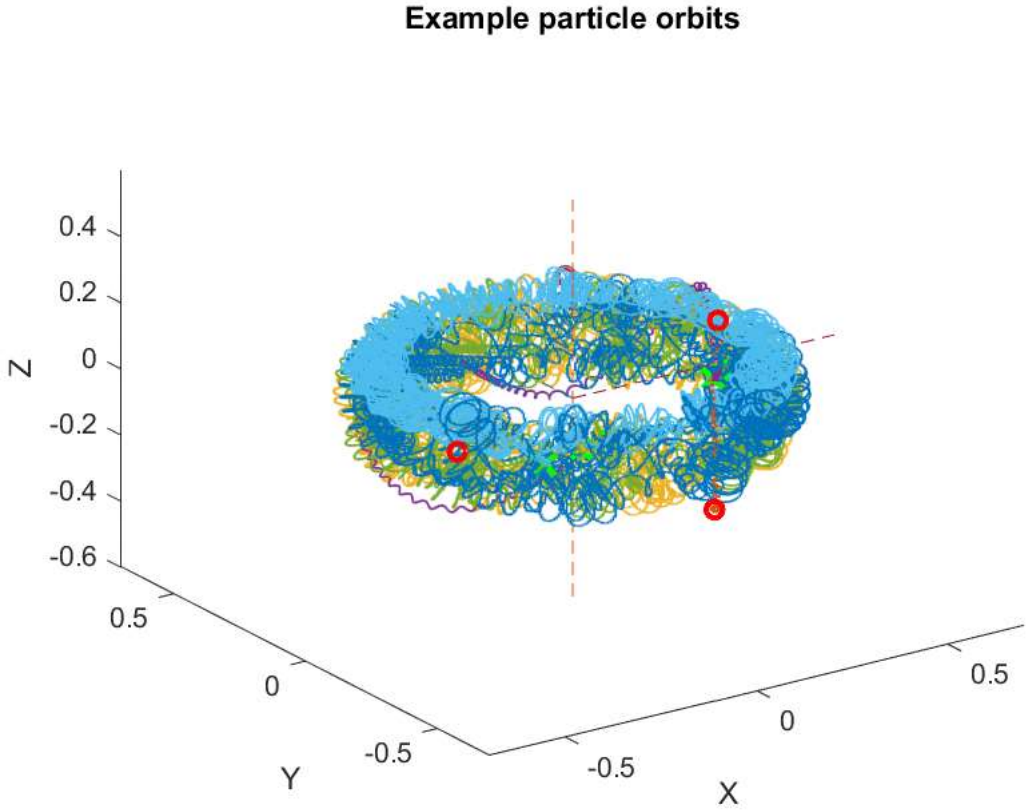
Results: Single particle orbits 1/6 velocity



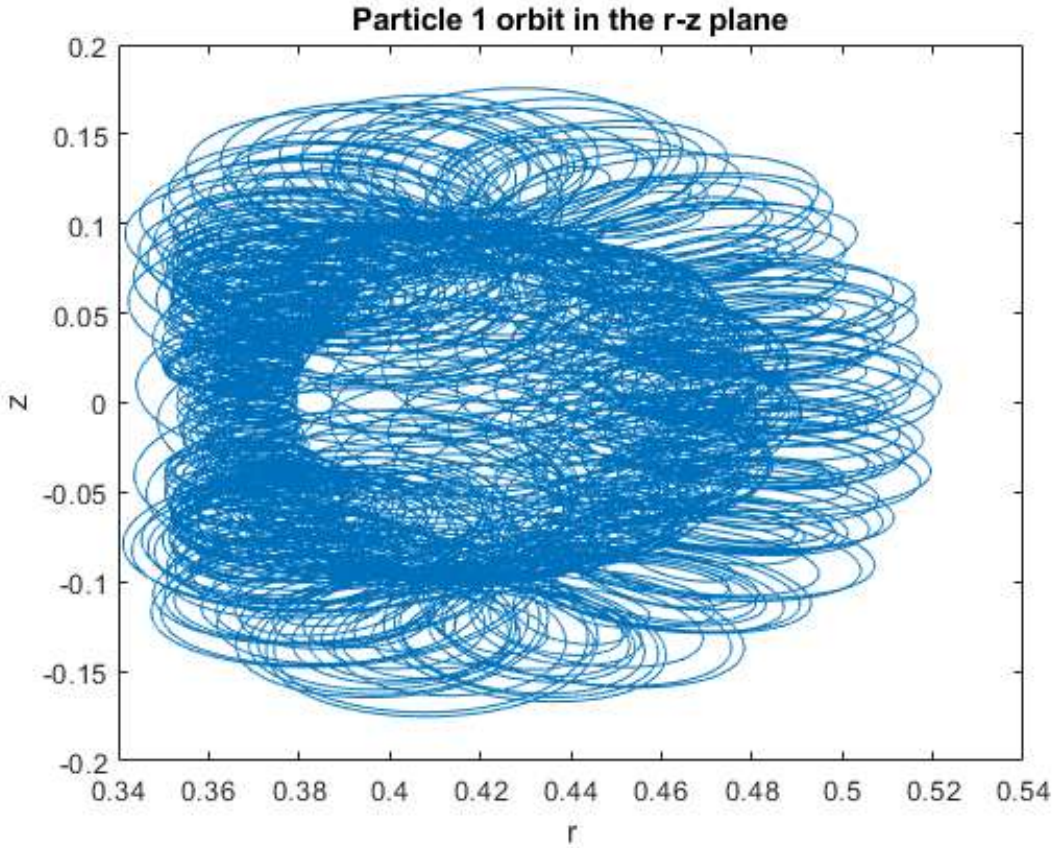
Results: Single particle orbits 1/6 velocity



Results: Single particle orbits 1/6 velocity



Example particle orbit in the r-z plane.



Conclusion

- I learned how to use C and CUDA to successfully build a partial kinetic code!
- GPU speeds up calculations.
- Observed orbits are consistent with theory.
- Issues at large time steps. Limited accuracy of RK4
- Fully kinetic codes are computationally intensive (1 million steps for each particle).
  - It is not feasible to simulate all particles, just the few important ones.
- The Chebyshev approximation, being compact and smooth, is suitable for GPU.
- We need more testing with realistic magnetic configurations.

Thank you!