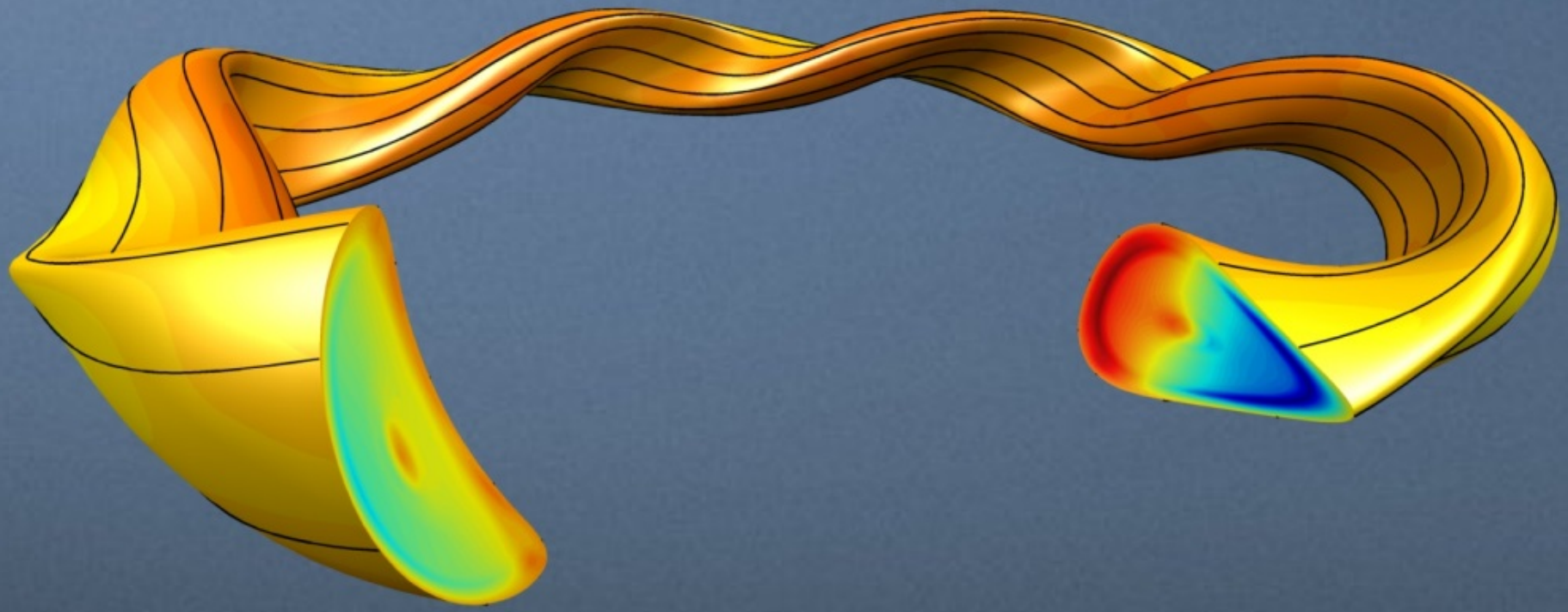
Computational plasma physics



Matt Landreman, University of Maryland
mattland@umd.edu

Thanks to Greg Hammett, Joel Dahlin, Nate Ferraro, GENE group

Computational plasma physics

- The role of computation in science.
- Diffusion equation example.

Break

- Challenges in simulating plasmas.
- Examples of plasma simulations.

Computational plasma physics

- The role of computation in science.
- Diffusion equation example.

Break

- Challenges in simulating plasmas.
- Examples of plasma simulations.

A simplistic view of the scientific method:

1. Theorists think up theories.
2. Generate a hypothesis.
3. Test hypothesis with an experiment.

The practice of science is much more complicated, and computation plays a key role.

Experiments

Mathematical models (equations)

Analytic Theory

Computation

The practice of science is much more complicated, and computation plays a key role.

Experiments

+ Ultimately matter.
+ Grounded in reality.
- Complicated.
- Expensive.
- You cannot measure or control everything.

Mathematical models (equations)

Analytic Theory

Computation

The practice of science is much more complicated, and computation plays a key role.

Experiments

+ Ultimately matter.
+ Grounded in reality.
- Complicated.
- Expensive.
- You cannot measure or control everything.

Mathematical models (equations)

Analytic Theory

+ Determine scalings.
+ Physical insight.
- Often limited to simplified geometry.
- Need to make many approximations.

Computation

The practice of science is much more complicated, and computation plays a key role.

Experiments

+ Ultimately matter.
+ Grounded in reality.
- Complicated.
- Expensive.
- You cannot measure or control everything.

Mathematical models (equations)

Analytic Theory

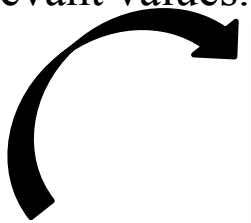
+ Determine scalings.
+ Physical insight.
- Often limited to simplified geometry.
- Need to make many approximations.

Computation

+ Can solve equations with fewer approximations.
+ Everything can be measured.
+ Effects can be turned on & off.
- Worries about bugs, resolution, numerical instabilities.
- Harder to get insight & scalings.

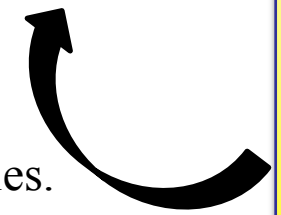
The practice of science is much more complicated, and computation plays a key role.

Validate models.
Experimentally relevant values.
Inspire models.



Experiments

+ Ultimately matter.
+ Grounded in reality.
- Complicated.
- Expensive.
- You cannot measure or control everything.



Design facilities.
Inspire experiments.
Interpret data.

Mathematical models (equations)

Analytic Theory

+ Determine scalings.
+ Physical insight.
- Often limited to simplified geometry.
- Need to make many approximations.

Computation

+ Can solve equations with fewer approximations.
+ Everything can be measured.
+ Effects can be turned on & off.
- Worries about bugs, resolution, numerical instabilities.
- Harder to get insight & scalings.

The practice of science is much more complicated, and computation plays a key role.

Validate models.
Experimentally relevant values.
Inspire models.

Experiments

+ Ultimately matter.
+ Grounded in reality.
- Complicated.
- Expensive.
- You cannot measure or control everything.

Design facilities.
Inspire experiments.
Interpret data.

Mathematical models (equations)

Analytic Theory

+ Determine scalings.
+ Physical insight.
- Often limited to simplified geometry.
- Need to make many approximations.

Need theory to know which equations to simulate.
Test codes by comparing to theory in various limits.
Analysis of equations informs numerical methods.

Simulations can inspire and test theory, indicate good approximations.

Computation

+ Can solve equations with fewer approximations.
+ Everything can be measured.
+ Effects can be turned on & off.
- Worries about bugs, resolution, numerical instabilities.
- Harder to get insight & scalings.

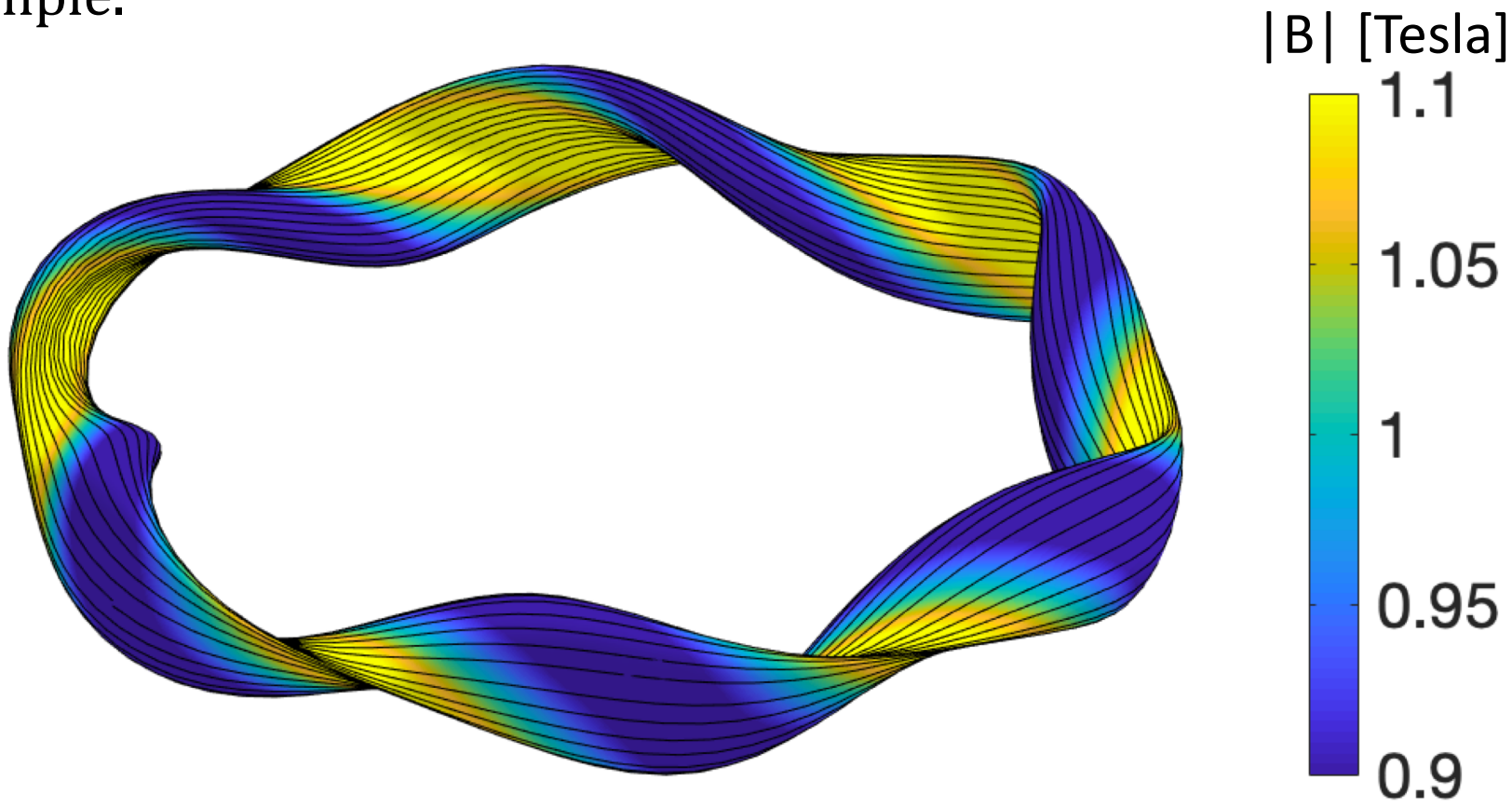
Case study of theory-computation interaction: quasisymmetric stellarators

Theory: Boozer (1983) – If a stellarator could have a symmetry direction for |B| it would confine particles as well as a tokamak.

Case study of theory-computation interaction: quasisymmetric stellarators

Theory: Boozer (1983) – If a stellarator could have a symmetry direction for |B| it would confine particles as well as a tokamak.

Computation: Nuhrenberg & Zille (1988) – Found a numerical example.



Case study of theory-computation interaction: quasisymmetric stellarators

Theory: Boozer (1983) – If a stellarator could have a symmetry direction for |B| it would confine particles as well as a tokamak.

Computation: Nuhrenberg & Zille (1988) – Found a numerical example.

Theory: Garren & Boozer (1991) –
Such a symmetry cannot be
achieved exactly.

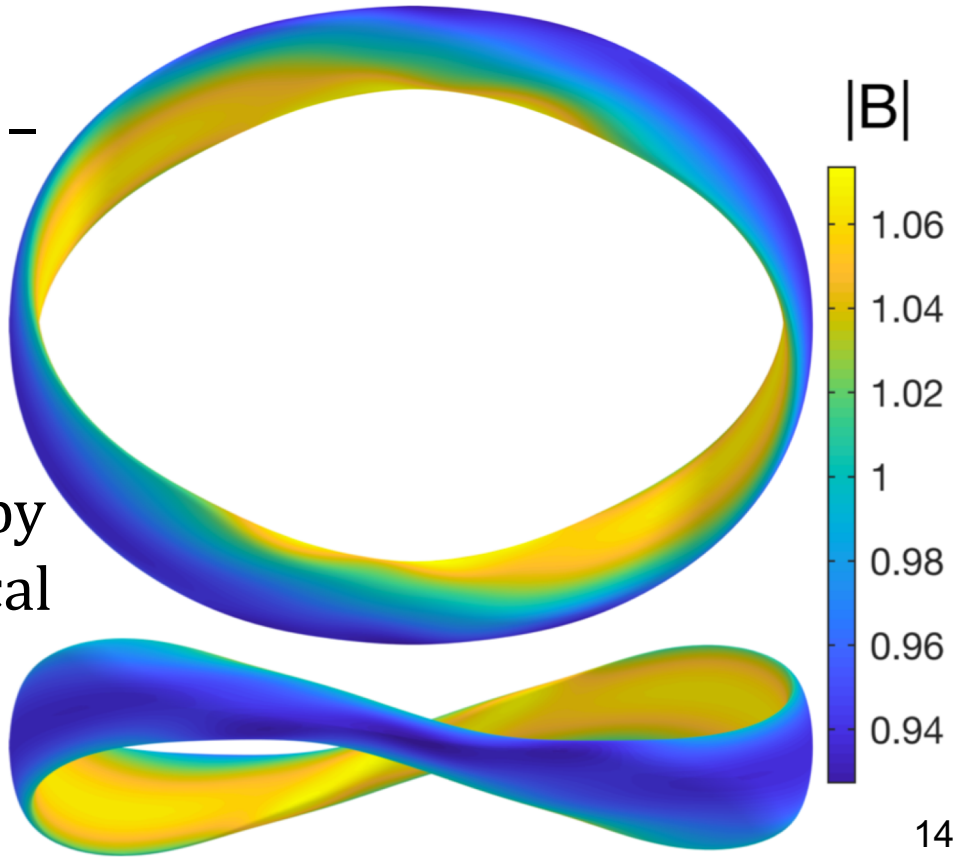
Case study of theory-computation interaction: quasisymmetric stellarators

Theory: Boozer (1983) – If a stellarator could have a symmetry direction for |B| it would confine particles as well as a tokamak.

Computation: Nuhrenberg & Zille (1988) – Found a numerical example.

Theory: Garren & Boozer (1991) – Such a symmetry cannot be achieved exactly.

Computation: Landreman et al (2019) – The equations derived by Garren & Boozer enable a practical algorithm to generate quasisymmetric stellarators.



In many ways, scientific computing is a lot like experiment.

- Keeping an experiment or code running is a game of whac-a-mole. (Alignment of optics drifts, sysadmins update some library, 'bitrot', …)
- A good dataset is precious.
- Must keep lab notebooks.
- Must think about data management.
- Tradeoff between carefulness & getting somewhere:
  - You never have time to really understand all the components.
  - There are always too many mysteries. Need to judge which are worth tracking down.

Computational plasma physics

- The role of computation in science.
- Diffusion equation example
  - Collocation vs modal discretization
  - Explicit vs implicit time advance
  - Numerical (in)stability
- Challenges in simulating plasmas.
- Examples of plasma simulations.

Example: 1D diffusion

$$T(x,t),\ x\in[0,1],\quad t\in[0,\infty)$$

$$T(x,0)=0,\quad T(0,t)=0,\quad T(1,t)=0.$$

$$\frac{\partial T}{\partial t}=\frac{\partial^2 T}{\partial x^2}+\log(x+1)$$

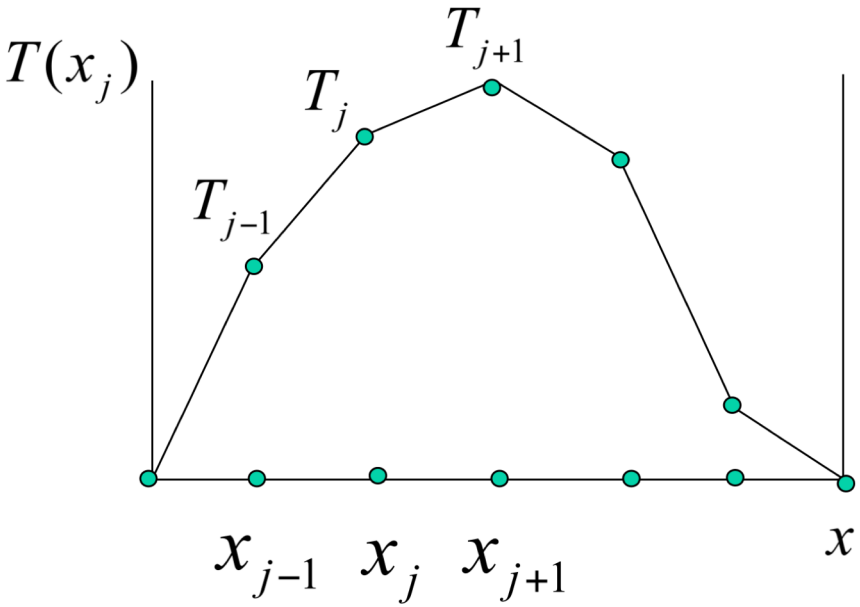
Example: 1D diffusion

$$T(x,t),\ x\in[0,1],\ t\in[0,\infty)$$

$$\frac{\partial T}{\partial t}=\frac{\partial^2 T}{\partial x^2}+\log(x+1)$$

$$T(x,0)=0,\quad T(0,t)=0,\quad T(1,t)=0.$$

Discretize in space with a "collocation" approach:
Store T on gridpoints x_j: $T_j=T(x_j)$



Could also use a "modal" discretization:
Store amplitudes a_j of some basis functions ϕ_j:

$$T(x)=\sum_j a_j\phi_j(x)$$

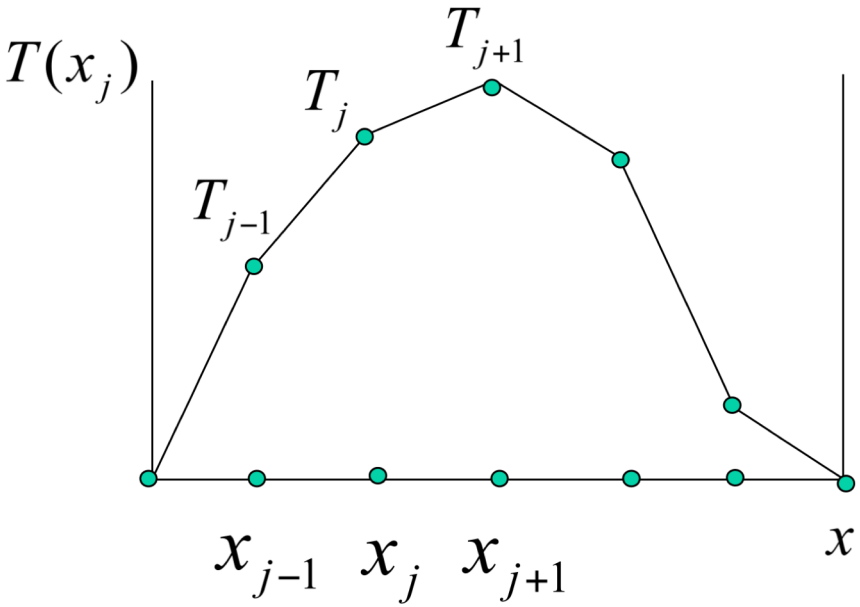
Example: 1D diffusion

$$T(x,t),\ x\in[0,1],\ t\in[0,\infty)$$

$$T(x,0)=0,\quad T(0,t)=0,\quad T(1,t)=0.$$

$$\frac{\partial T}{\partial t}=\frac{\partial^2 T}{\partial x^2}+\log(x+1)$$

Discretize in space with a "collocation" approach:
Store T on gridpoints x_j: $T_j=T(x_j)$



"Finite difference" derivatives:

$$\left(\frac{\partial T}{\partial x}\right)_{x_j}=\lim_{h\to 0}\frac{T(x_j+h)-T(x_j-h)}{2h}$$

$$\approx\frac{T_{j+1/2}-T_{j-1/2}}{\Delta x}.$$

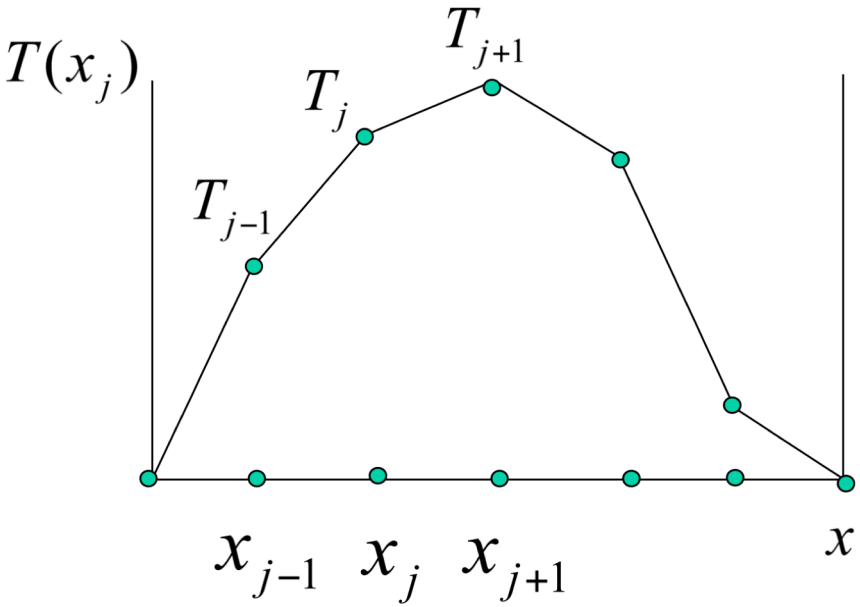
Example: 1D diffusion

$$T(x,t),\ x\in[0,1],\ t\in[0,\infty)$$

$$\frac{\partial T}{\partial t}=\frac{\partial^2 T}{\partial x^2}+\log(x+1)$$

$$T(x,0)=0,\quad T(0,t)=0,\quad T(1,t)=0.$$

Discretize in space with a "collocation" approach:
Store T on gridpoints x_j: $T_j=T(x_j)$



"Finite difference" derivatives:

$$\left(\frac{\partial T}{\partial x}\right)_{x_j}=\lim_{h\to 0}\frac{T(x_j+h)-T(x_j-h)}{2h}$$

$$\approx\frac{T_{j+1/2}-T_{j-1/2}}{\Delta x}.$$

$$\left(\frac{\partial^2 T}{\partial x^2}\right)_{x_j}\approx\frac{[T_{j+1}-T_j]-[T_j-T_{j-1}]}{(\Delta x)^2}$$

We've discretized space; now discretize time.

$$\frac{\partial T_j}{\partial t} = \frac{\left[T_{j+1} - T_j\right] - \left[T_j - T_{j-1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

Introduce a time grid t_m. We store $T_j^m = T\left(x_j, t_m\right)$.

$$\left(\frac{\partial T_j}{\partial t}\right)_{x_j, t_m} \approx \frac{T_j^{m+1} - T_j^m}{\Delta t}.$$

We've discretized space; now discretize time.

$$\frac{\partial T_j}{\partial t} = \frac{\left[T_{j+1} - T_j\right] - \left[T_j - T_{j-1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

Introduce a time grid t_m. We store $T_j^m = T\left(x_j, t_m\right)$.

$$\left(\frac{\partial T_j}{\partial t}\right)_{x_j, t_m} \approx \frac{T_j^{m+1} - T_j^m}{\Delta t}.$$

Should we evaluate right-hand side at t_m or t_{m+1}?

We've discretized space; now discretize time.

$$\frac{\partial T_j}{\partial t} = \frac{\left[T_{j+1} - T_j\right] - \left[T_j - T_{j-1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

Introduce a time grid t_m. We store $T_j^m = T\left(x_j, t_m\right)$.

$$\left(\frac{\partial T_j}{\partial t}\right)_{x_j, t_m} \approx \frac{T_j^{m+1} - T_j^m}{\Delta t}.$$

Should we evaluate right-hand side at t_m or t_{m+1}?

If we choose t_m, we get an **explicit** formula for updating T: "Forward Euler"

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^m - T_j^m\right] - \left[T_j^m - T_{j-1}^m\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

We've discretized space; now discretize time.

$$\frac{\partial T_j}{\partial t} = \frac{\left[T_{j+1} - T_j\right] - \left[T_j - T_{j-1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

Introduce a time grid t_m. We store $T_j^m = T\left(x_j, t_m\right)$.

$$\left(\frac{\partial T_j}{\partial t}\right)_{x_j, t_m} \approx \frac{T_j^{m+1} - T_j^m}{\Delta t}.$$

Should we evaluate right-hand side at t_m or t_{m+1}?

If we choose t_m, we get an **explicit** formula for updating T: "Forward Euler"

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^m - T_j^m\right] - \left[T_j^m - T_{j-1}^m\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

$$T_j^{m+1} = T_j^m + \Delta t \frac{\left[T_{j+1}^m - T_j^m\right] - \left[T_j^m - T_{j-1}^m\right]}{\left(\Delta x\right)^2} + \Delta t \log\left(x_j + 1\right)$$

In an 'implicit' method, terms other than the d/dt term are evaluated at the new time.

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^{m+1} - T_j^{m+1}\right] - \left[T_j^{m+1} - T_{j-1}^{m+1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

"Backward Euler"

In an 'implicit' method, terms other than the d/dt term are evaluated at the new time.

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^{m+1} - T_j^{m+1}\right] - \left[T_j^{m+1} - T_{j-1}^{m+1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

"Backward Euler"

Let T^m be a vector of the T_j^m values:

$$\frac{T^{m+1} - T^m}{\Delta t} = \vec{\vec{D}} T^{m+1} + \log\left(x_j + 1\right),$$

Derivative becomes **differentiation matrix:**

$$\vec{\vec{D}} = \frac{1}{\left(\Delta x\right)^2} \begin{pmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \\ 0 & 1 & -2 & \ddots \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

In an 'implicit' method, terms other than the d/dt term are evaluated at the new time.

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^{m+1} - T_j^{m+1}\right] - \left[T_j^{m+1} - T_{j-1}^{m+1}\right]}{\left(\Delta x\right)^2} + \log\left(x_j + 1\right)$$

"Backward Euler"

Let T^m be a vector of the T_j^m values:

$$\frac{T^{m+1} - T^m}{\Delta t} = \vec{\vec{D}} T^{m+1} + \log\left(x_j + 1\right),$$

Derivative becomes **differentiation matrix:**

$$\vec{\vec{D}} = \frac{1}{\left(\Delta x\right)^2} \begin{pmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \\ 0 & 1 & -2 & \ddots \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

$$\underbrace{T^{m+1} - \Delta t\ \vec{\vec{D}} T^{m+1}}_{= \left(\vec{\vec{I}} - \Delta t\ \vec{\vec{D}}\right) T^{m+1}} = T^m + \Delta t\ \log\left(x_j + 1\right)$$

In an 'implicit' method, terms other than the d/dt term are evaluated at the new time.

$$\frac{T_j^{m+1}-T_j^m}{\Delta t}=\frac{\left[T_{j+1}^{m+1}-T_j^{m+1}\right]-\left[T_j^{m+1}-T_{j-1}^{m+1}\right]}{\left(\Delta x\right)^2}+\log\left(x_j+1\right)$$

"Backward Euler"

Let T^m be a vector of the T_j^m values:

$$\frac{T^{m+1}-T^m}{\Delta t}=\vec{\vec{D}}T^{m+1}+\log\left(x_j+1\right),$$

Derivative becomes **differentiation matrix:**

$$\vec{\vec{D}}=\frac{1}{\left(\Delta x\right)^2}\begin{pmatrix}-2 & 1 & 0 & \cdots\\ 1 & -2 & 1 & \\ 0 & 1 & -2 & \ddots\\ \vdots & & \ddots & \ddots\end{pmatrix}$$

$$\underbrace{T^{m+1}-\Delta t\ \vec{\vec{D}}T^{m+1}}_{=\left(\vec{\vec{I}}-\Delta t\ \vec{\vec{D}}\right)T^{m+1}}=T^m+\Delta t\ \log\left(x_j+1\right)$$

$$T^{m+1}=\left(\vec{\vec{I}}-\Delta t\ \vec{\vec{D}}\right)^{-1}\left[T^m+\Delta t\ \log\left(x_j+1\right)\right]$$

In an 'implicit' method, terms other than the d/dt term are evaluated at the new time.

$$\frac{T_j^{m+1} - T_j^m}{\Delta t} = \frac{\left[T_{j+1}^{m+1} - T_j^{m+1}\right] - \left[T_j^{m+1} - T_{j-1}^{m+1}\right]}{(\Delta x)^2} + \log(x_j + 1)$$

"Backward Euler"

Let T^m be a vector of the T_j^m values:

$$\frac{T^{m+1} - T^m}{\Delta t} = \vec{\vec{D}} T^{m+1} + \log(x_j + 1),$$

Derivative becomes **differentiation matrix:**

$$\vec{\vec{D}} = \frac{1}{(\Delta x)^2} \begin{pmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \\ 0 & 1 & -2 & \ddots \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

$$\underbrace{T^{m+1} - \Delta t\ \vec{\vec{D}} T^{m+1}}_{= \left(\vec{\vec{I}} - \Delta t\ \vec{\vec{D}}\right) T^{m+1}} = T^m + \Delta t\ \log(x_j + 1)$$

$$T^{m+1} = \left(\vec{\vec{I}} - \Delta t\ \vec{\vec{D}}\right)^{-1} \left[T^m + \Delta t\ \log(x_j + 1)\right]$$

- Implicit methods require a matrix inversion, so each time step is slower than in an explicit method.
- In exchange, Δt can be larger without numerical instability.

- MATLAB example

The numerical instability can be understood using Fourier analysis.

Drop inhomogeneous term – amounts to subtracting off long-time solution.

Explicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m}}{\partial x^2}$$

Implicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m+1}}{\partial x^2}$$

Consider Fourier modes $T^m\left(x\right)=\bar{T}^m\exp\left(ikx\right)$:

(Really we should write out the spatial discretization. Here we won't to simplify the presentation.)

The numerical instability can be understood using Fourier analysis.

Drop inhomogeneous term – amounts to subtracting off long-time solution.

Explicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m}}{\partial x^2}$$

Implicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m+1}}{\partial x^2}$$

Consider Fourier modes $T^{m}\left(x\right)=\bar{T}^{m}\exp\left(ikx\right)$:

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2 T^{m}$$

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2 T^{m+1}$$

The numerical instability can be understood using Fourier analysis.

Drop inhomogeneous term – amounts to subtracting off long-time solution.

Explicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m}}{\partial x^2}$$

Implicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m+1}}{\partial x^2}$$

Consider Fourier modes $T^{m}\left(x\right)=\bar{T}^{m}\exp\left(ikx\right)$:

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2 T^{m}$$

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2 T^{m+1}$$

Rearrange:

$$\frac{T^{m+1}}{T^{m}}=\underbrace{1-\Delta t\ k^2}$$

If $\Delta t>2/k^2$ for largest k,

then $\left|T\right|$ increases.

The numerical instability can be understood using Fourier analysis.

Drop inhomogeneous term – amounts to subtracting off long-time solution.

Explicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m}}{\partial x^2}$$

Implicit

$$\frac{T^{m+1}-T^{m}}{\Delta t}=\frac{\partial^2 T^{m+1}}{\partial x^2}$$

Consider Fourier modes $T^m\left(x\right)=\bar{T}^m\exp\left(ikx\right)$:

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2T^{m}$$

$$\frac{T^{m+1}-T^{m}}{\Delta t}=-k^2T^{m+1}$$

Rearrange:

$$\frac{T^{m+1}}{T^{m}}=\underbrace{1-\Delta t\ k^2}$$

$$\frac{T^{m+1}}{T^{m}}=\frac{1}{\underbrace{1+\Delta t\ k^2}}$$

Magnitude always <1

If $\Delta t>2/k^2$ for largest k,

then $\left|T\right|$ increases.

Computational plasma physics

- The role of computation in science.
- Diffusion equation example.

Break

- Challenges in simulating plasmas.
- Examples of plasma simulations.

Computational plasma physics

- The role of computation in science.
- Diffusion equation example.

Break

- Challenges in simulating plasmas.
- Examples of plasma simulations.

Why not just numerically solve Maxwell's equations + Lorentz force?

$$\nabla \cdot \mathbf{E} = \rho / \varepsilon_0 \qquad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}$$

$$m\frac{d\mathbf{v}}{dt} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right)$$

Why not just numerically solve Maxwell's equations + Lorentz force?

$$\nabla \cdot \mathbf{E} = \rho / \varepsilon_0 \qquad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}$$

$$m\frac{d\mathbf{v}}{dt} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right)$$

Typical lab plasma has ~ 10^{20} particles.

Why not just numerically solve Maxwell's equations + Lorentz force?

$$\nabla \cdot \mathbf{E} = \rho / \varepsilon_0 \qquad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}$$

$$m\frac{d\mathbf{v}}{dt} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right)$$

Typical lab plasma has ~ 10^{20} particles.

Storing a floating-point number ("double precision") takes 8 bytes.

Just storing (x, y, z, v_x, v_y, v_z) for 10^{20} particles would take $6 \times 8 \times 10^{20}$ bytes ~ 5×10^9 terabytes.

Why not just numerically solve Maxwell's equations + Boltzmann equation?

Distribution function $f\left(t,x,y,z,v_x,v_y,v_z\right)$

$$\frac{\partial f}{\partial t}+\mathbf{v}\cdot\nabla f+\frac{q}{m}\left(\mathbf{E}+\mathbf{v}\times\mathbf{B}\right)\cdot\nabla_v f=\left(\frac{df}{dt}\right)_{\text{collisions}}$$

High number of dimensions + scale separation are challenges.

Why not just numerically solve Maxwell's equations + Boltzmann equation?

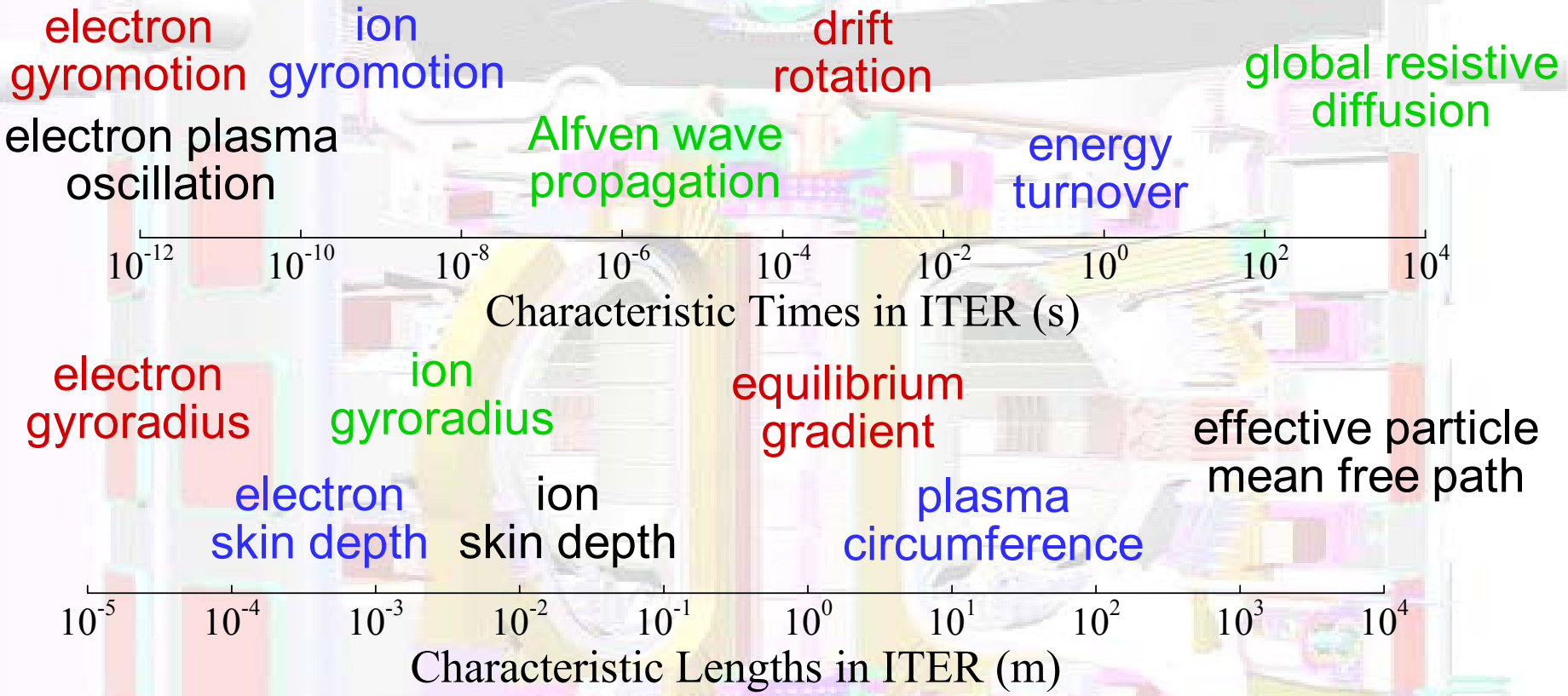
Distribution function $f\left(t,x,y,z,v_x,v_y,v_z\right)$

$$\frac{\partial f}{\partial t}+\mathbf{v}\cdot\nabla f+\frac{q}{m}\left(\mathbf{E}+\mathbf{v}\times\mathbf{B}\right)\cdot\nabla_v f=\left(\frac{df}{dt}\right)_{\text{collisions}}$$

High number of dimensions + scale separation are challenges.

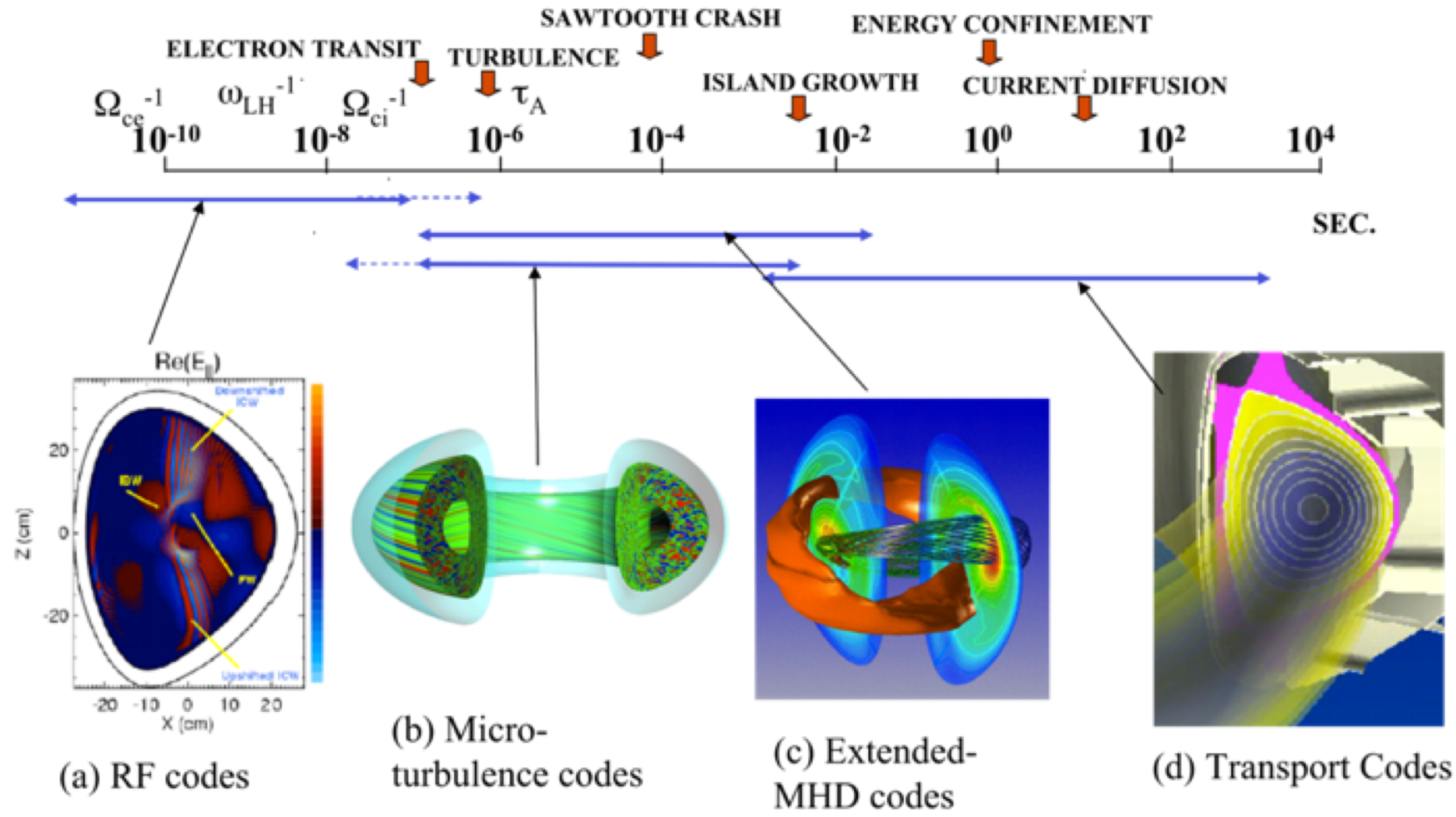
Suppose spatial grid scale is ~ electron gyroradius: ~ 10^{-5} m.
Suppose plasma size is 1 m^3 ➔ $(10^5)^3 = 10^{15}$ spatial grid points.
Suppose velocity grid is 10×10×10.
Storing a double-precision number takes 8 bytes.
So just storing f would take $8\times10^3\times10^{15}$ bytes ~ 10^7 terabytes.

Fusion plasmas exhibit enormous ranges of temporal and spatial scales.



- Using 10^4 time steps in a simulation can be reasonable. 10^{16} is not.
- Using 10^3 grid points/dimension in a simulation can be reasonable. 10^9 is not.

Different classes of code are used to handle different ranges of scales.



(a) RF codes

(b) Micro-turbulence codes

(c) Extended-MHD codes

(d) Transport Codes

Computational plasma physics

- The role of computation in science.
- Diffusion equation example.
- Challenges in simulating plasmas.
- Examples of plasma simulations.

Plasma turbulence can now be simulated using ‘gyrokinetic’ equations.

- Average equations over gyration to eliminate the fastest timescale.
- Only keep gyroradius scale perpendicular to B. Use coarser grid along **B**.

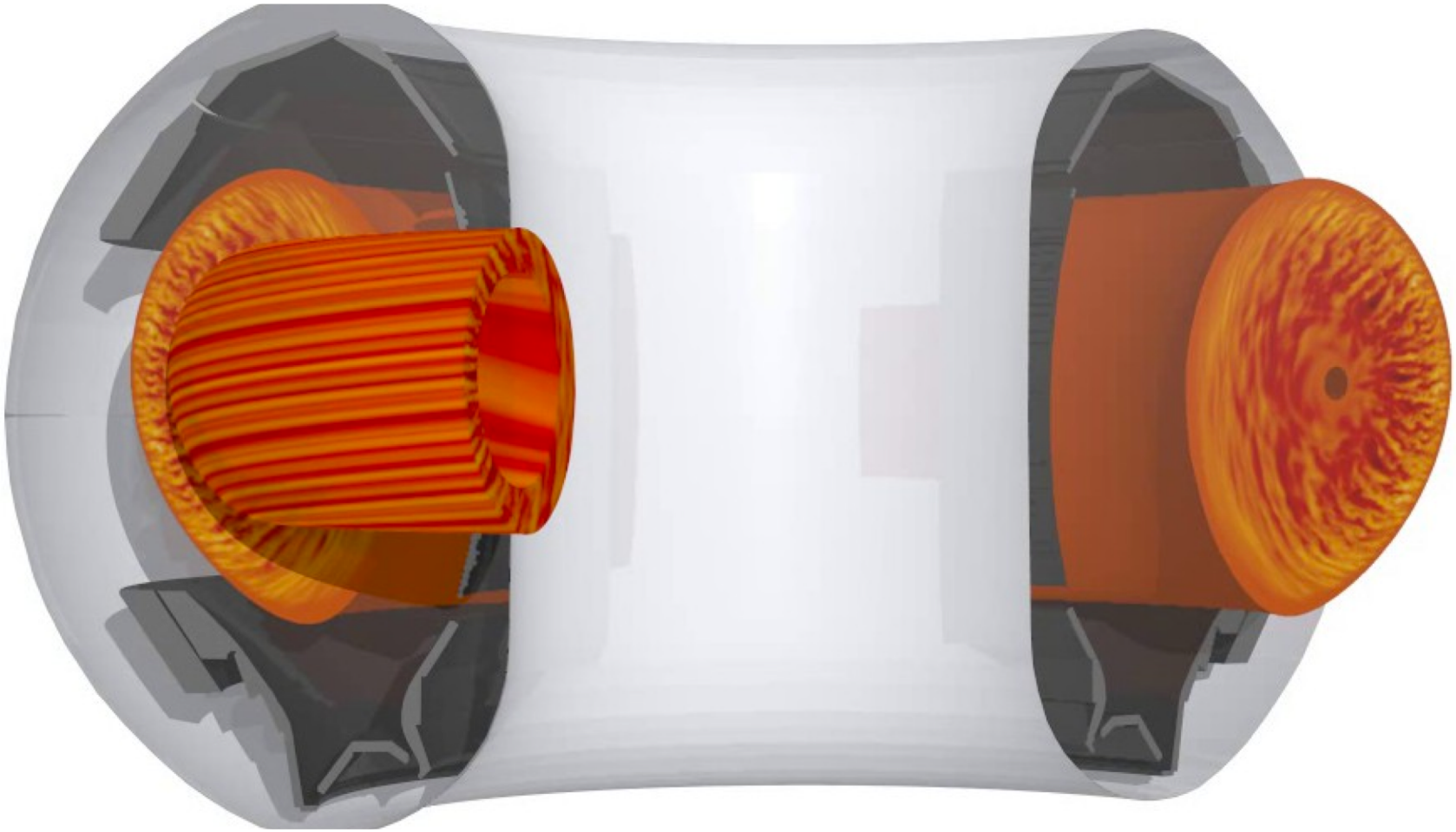


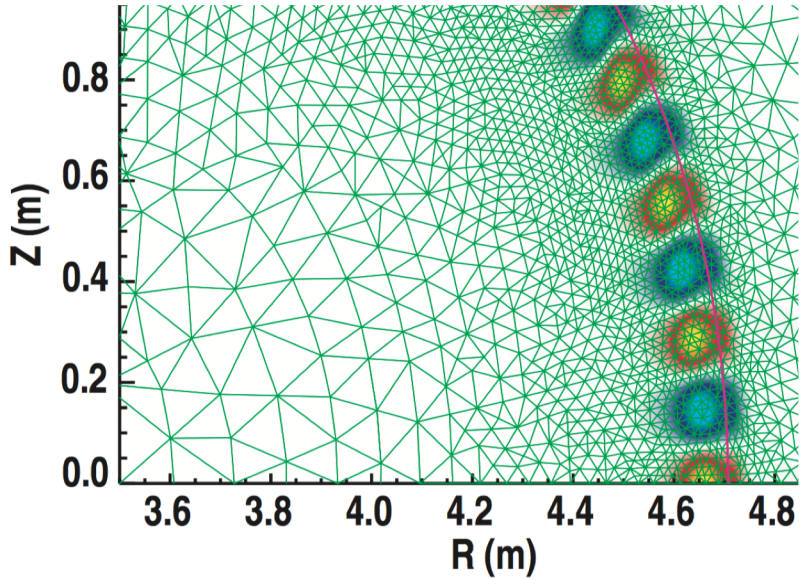
Figure + movie by GENE group: genecode.org

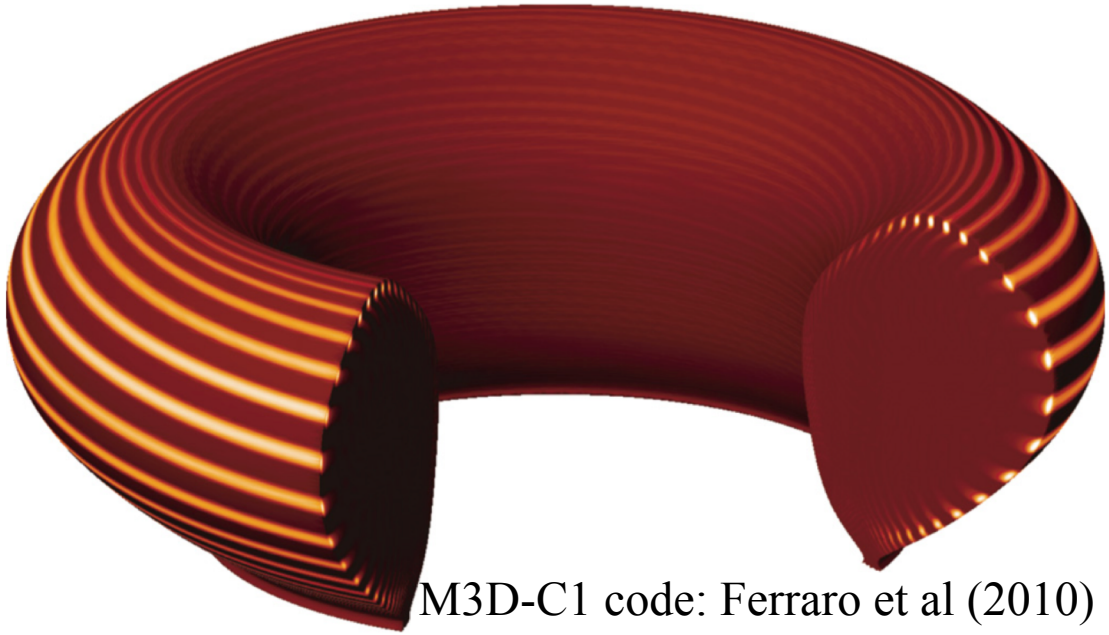
- GENE movie

"Extended MHD" codes are used to simulate large-scale plasma dynamics.

- Fluid equations: reduced dimensionality yields huge computational savings, though some kinetic effects not captured (e.g. calculating bootstrap j).
- More sophisticated equations than *ideal* magnetohydrodynamics.
- Diffusion terms added to mock up the underlying turbulent transport.

HiFi code:
Shaffner et al (2014)



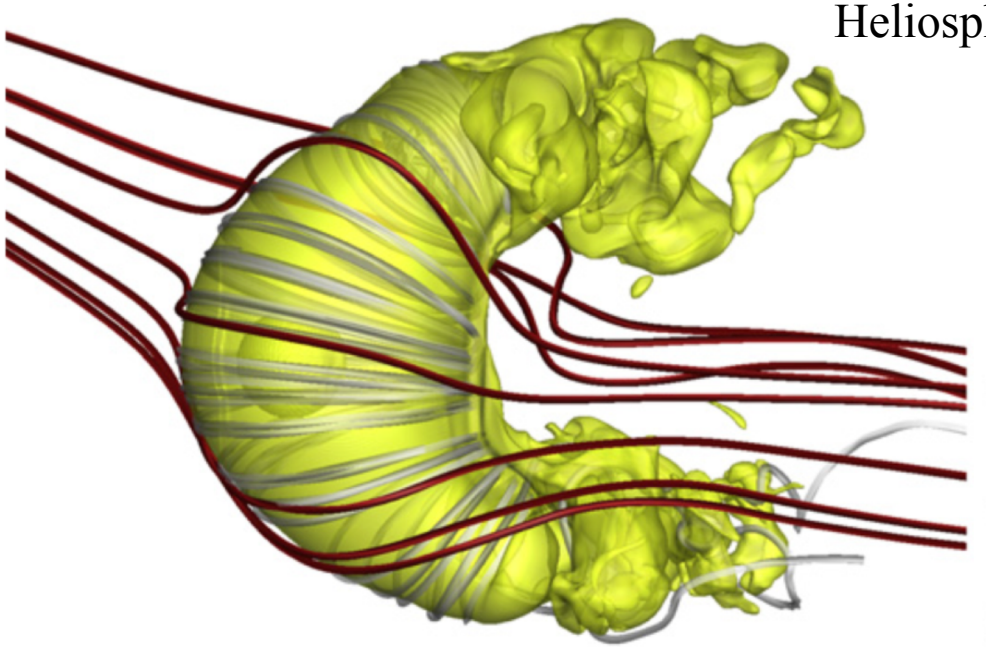


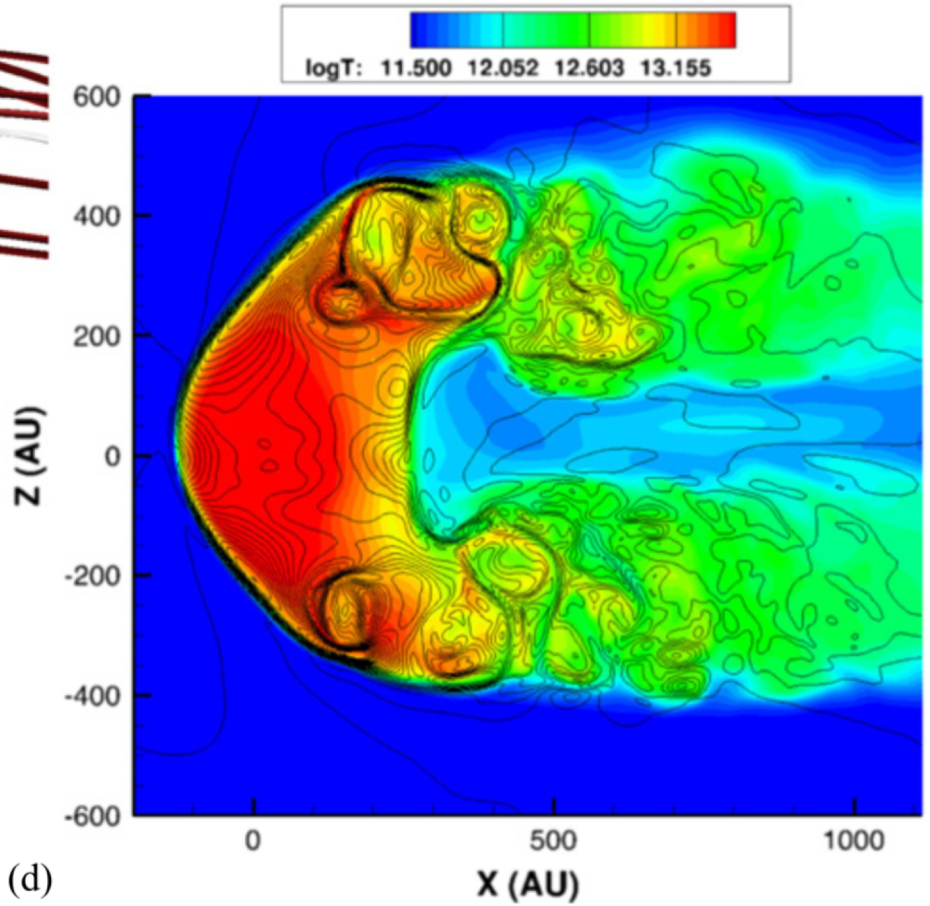
M3D-C1 code: Ferraro et al (2010)

- M3D-C1 movie

Sophisticated fluid & kinetic computations are also used to simulate plasmas in space.

Heliosphere fluid simulation: Opher et al (2015)



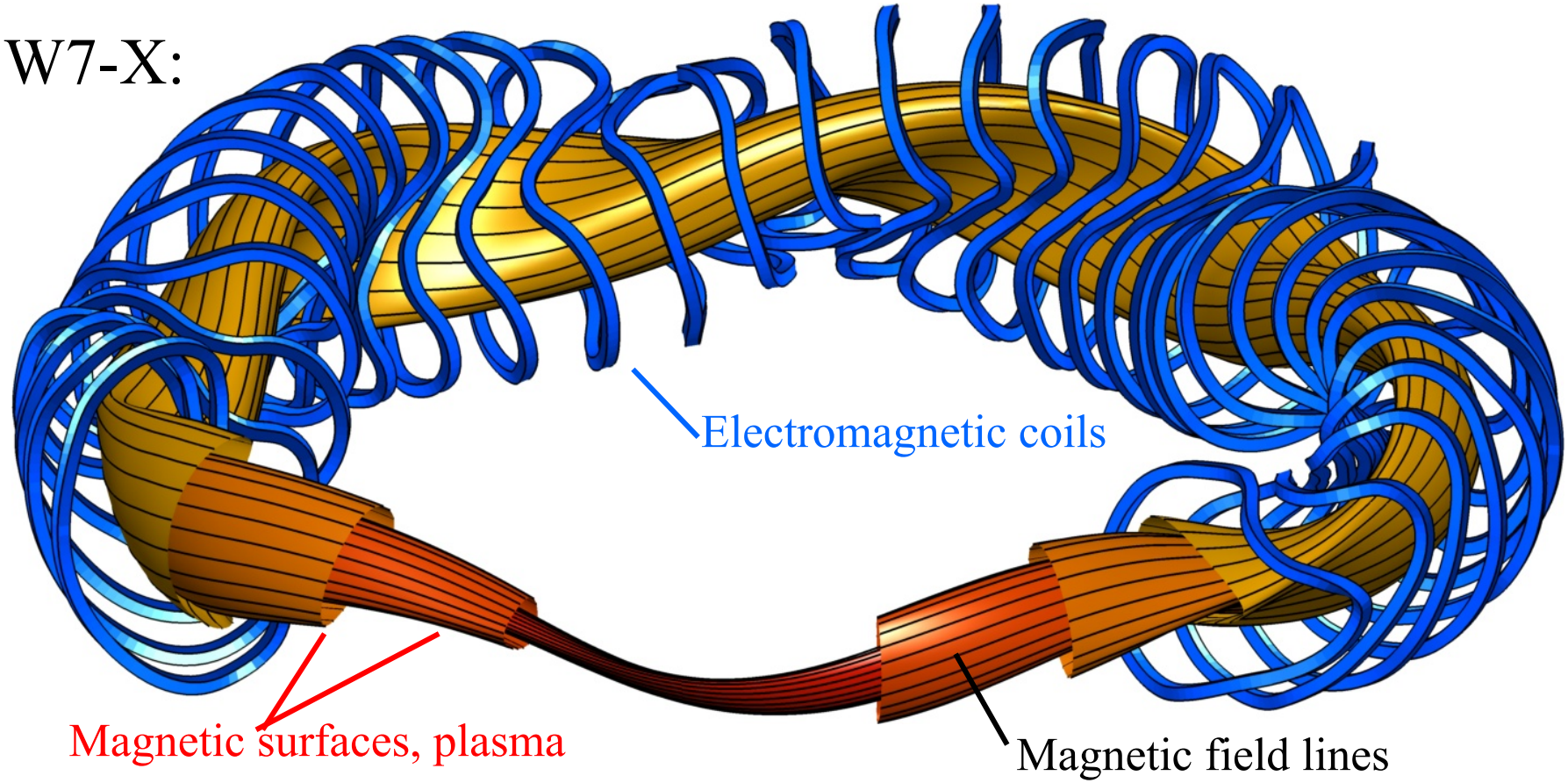


(d)

- Kinetic reconnection movie

Numerical optimization is central to the design of modern stellarators.

W7-X:



Plasma shape varied to extremize

- Confinement of particle orbits,
- MHD stability,
- Low plasma current,
- …

Coil shapes varied to produce the desired plasma shape.

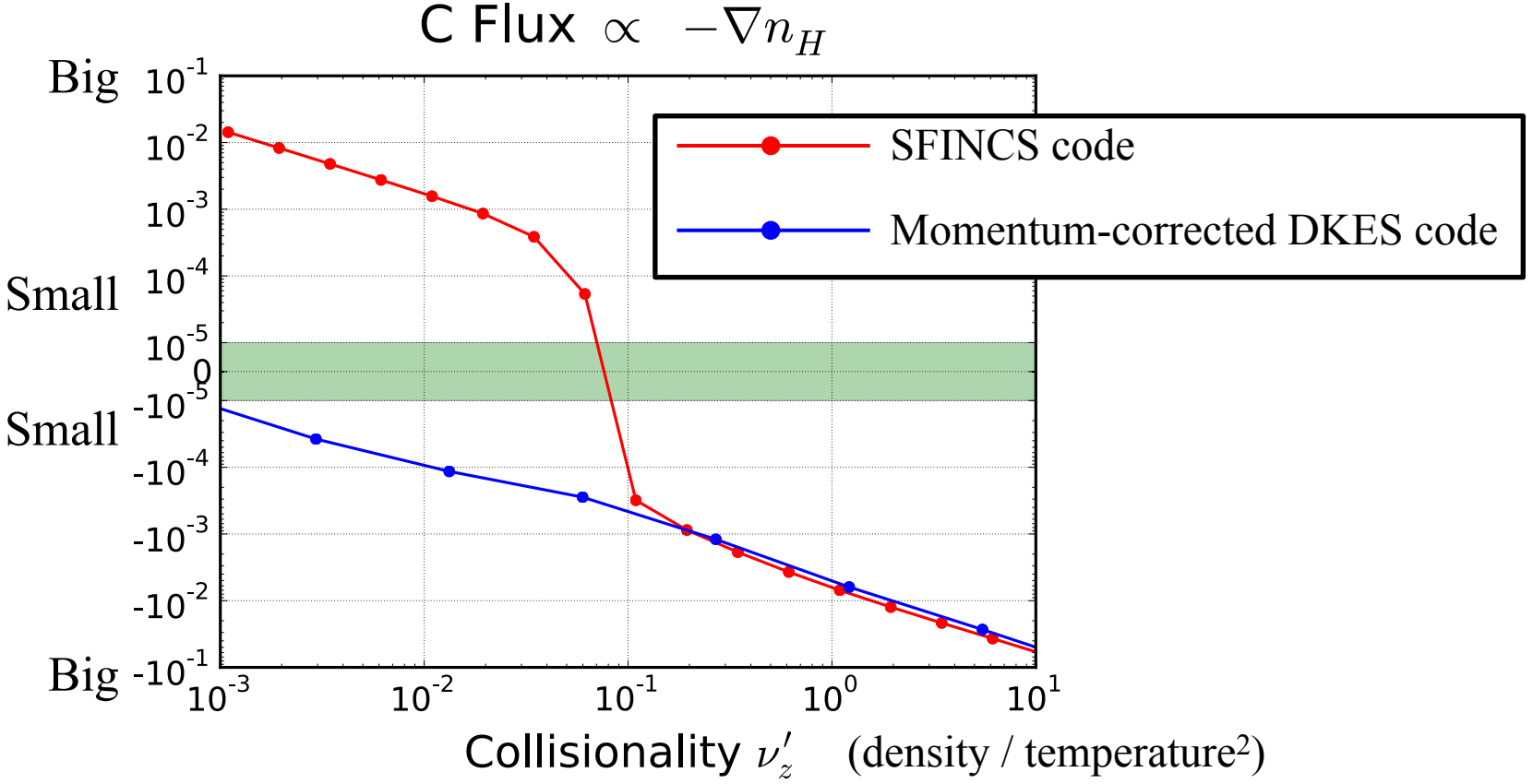
Conclusions:
Is computational plasma physics right for you?

- Need to know analytic plasma theory too, lots of algebra.
- Need to keep up with numerical methods & libraries.
- Need to spend lots of time dealing with e.g. stupid compiler errors.
- Need to be very organized.
- Can work anywhere with internet. (Nice for parents.)
- Funding seems more stable than in experiment?
- Coding & algebra = opportunities for "flow".
- Good if approximations like $3 \gg 1$ make you nervous.
- Good if you like being omnipotent and omniscient.
- You can work on exciting inter-disciplinary problems at the cutting edge of physics, applied math, and computer science.

Extra slides

Example of computation leading theory: impurity transport in a stellarator.

2 codes gave very different predictions for one of the transport coefficients:



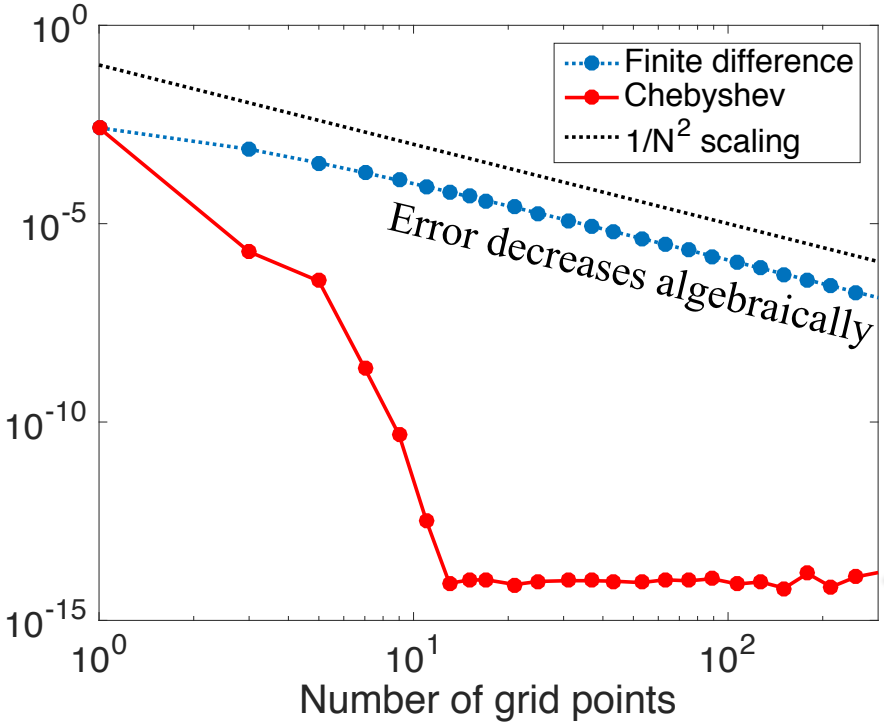
Led to new analytic theory: a difference between the codes that was thought to be unimportant (diffusion in |v| due to collisions) is actually important here.

A Mollen et al, Phys Plasmas (2015)

Using Fourier modes or orthogonal polynomials, you can achieve 'spectral accuracy'

E.g., Chebyshev grid for x_j and associated $\vec{\vec{D}}$:

Error in $T(x=0.5, t=0.1)$, loglog plot

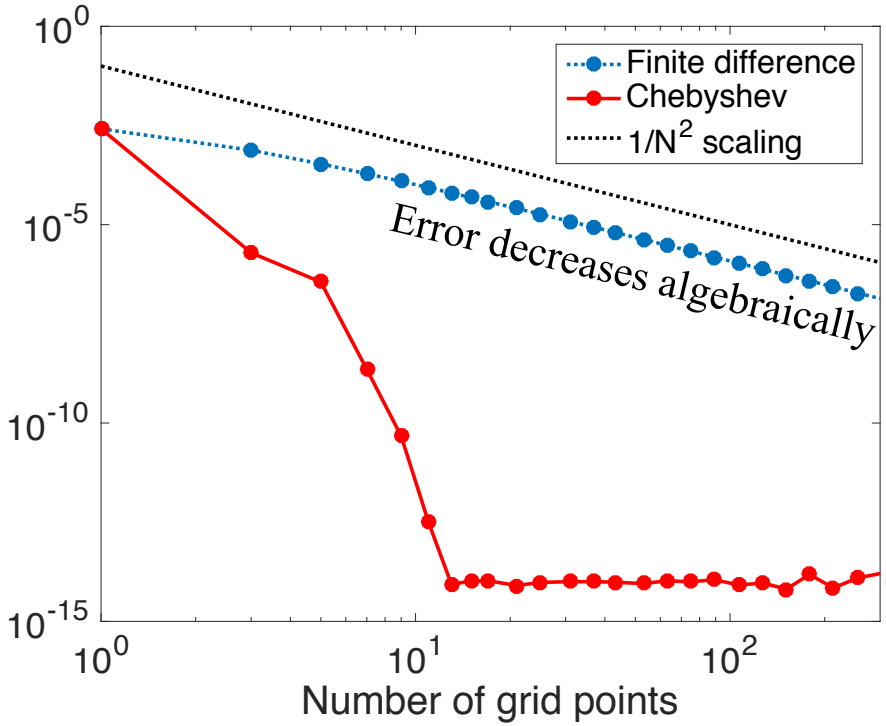


Best conceivable precision due to roundoff error.

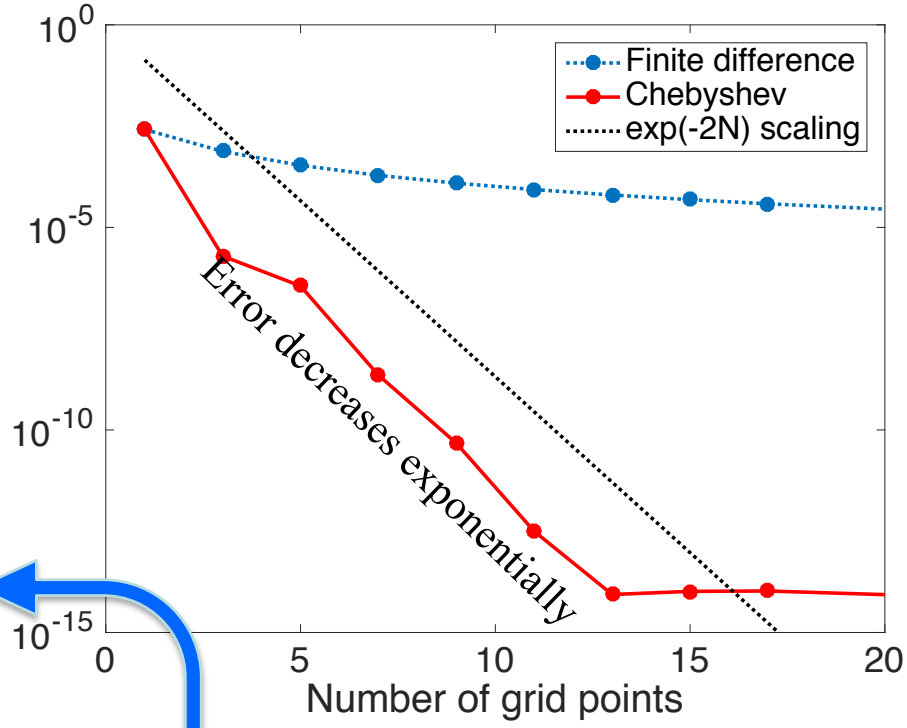
Using Fourier modes or orthogonal polynomials, you can achieve 'spectral accuracy'

E.g., Chebyshev grid for x_j and associated $\vec{\vec{D}}$:

Error in $T(x{=}0.5, t{=}0.1)$, loglog plot



Same data on a semilog plot



Best conceivable precision due to roundoff error.