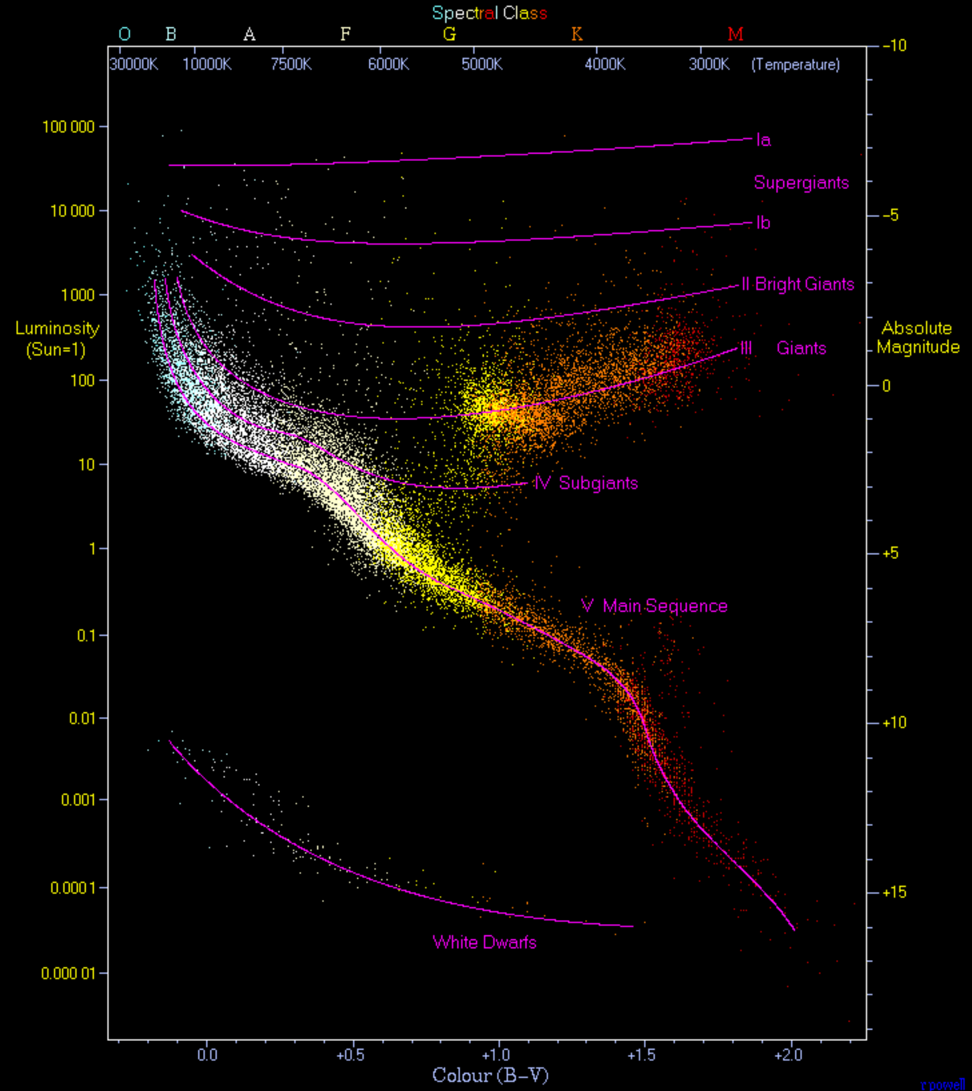
Challenges of Predictive 3D Astrophysical Simulations of Accreting Systems

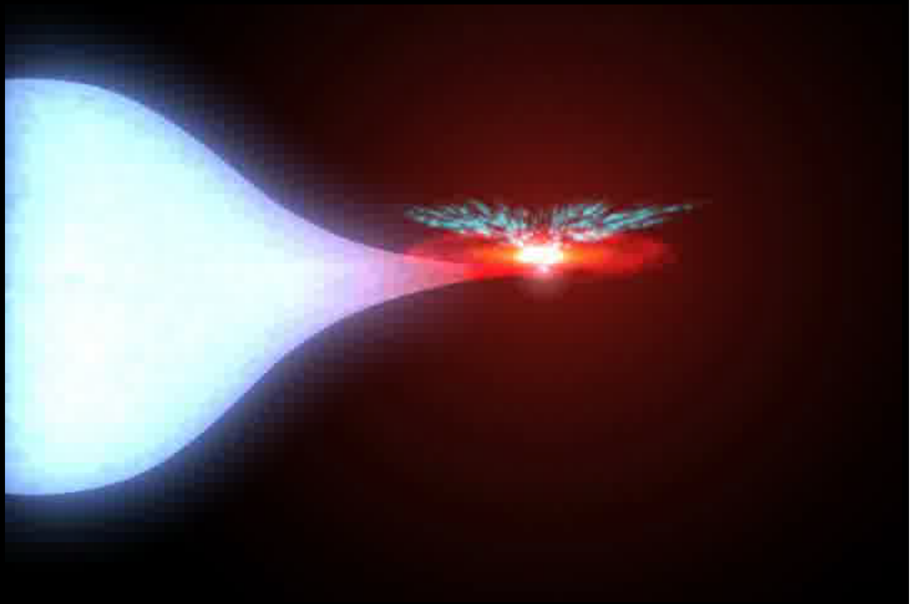
John F. Hawley
Department of Astronomy, University of Virginia

Stellar Evolution: Astro Computing's Early Triumph



Observed Properties of Accreting Systems

- Range of phenomena: black hole binaries, quasars, AGNs
- Different spectral states: thermal, nonthermal, soft-high, hard-low, Eddington accretion, Sub-Eddington
- Transitions between states
- Cataclysmic variables, dwarf novae
- Winds, collimated jets
- Quasi-Periodic Oscillations
- Variability, both local and global, on dynamical timescales

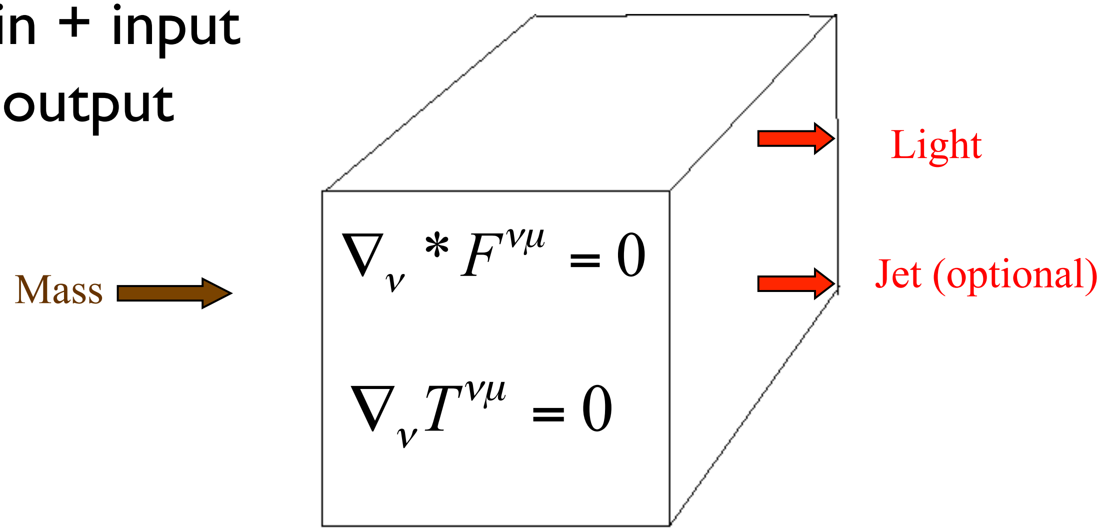


Questions about Accretion

- How are winds and/or jets produced and under what circumstances?
- What is the stress level and the accretion rate?
- What disk structures arise naturally?
- What are the properties of disk turbulence?
- What is the disk luminosity and how is that a function of black hole mass and spin (efficiency)?
- Is there a magnetic dynamo in disks?
- Can we account for different spectral states?
- Origin of *Quasi-Periodic Oscillations* and the Fe Ka line seen in X-ray observations
- What are the properties of the inner disk where it plunges into the hole?
- How does black hole spin affect accretion?
- How does accretion affect the black hole spin?

The Goal: Predictive, First Principle Simulations

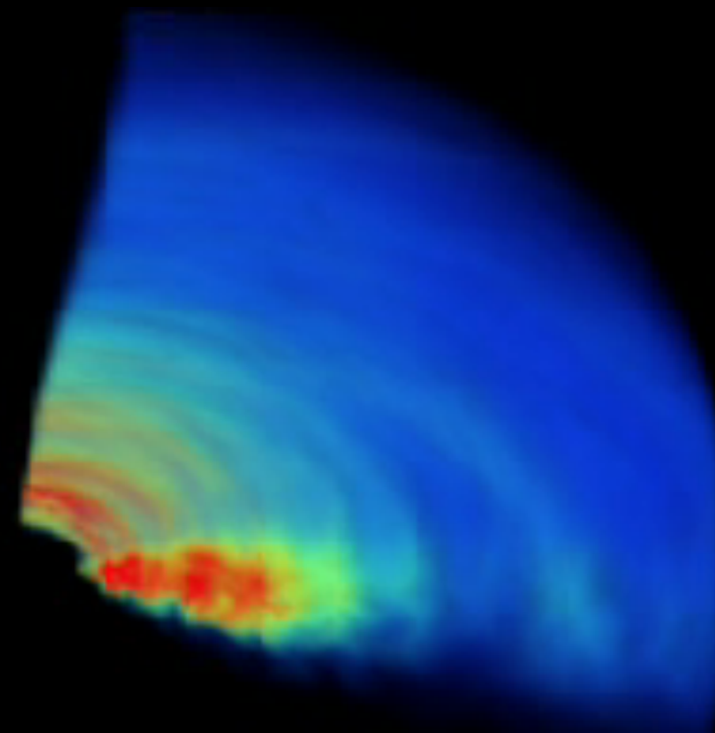
- Let the equations determine the properties of accreting systems
- Black hole mass, spin + input fuel and field yields output



Challenges

- The physics is comprehensive and complex
- Improved, more complex and accurate algorithms
- More complex software: efficiency, scalability, flexibility
- Increasingly large and complex datasets: storage, maintenance, access, analysis
- Collaboration, Education, Training

Accretion Simulations: Local and Global



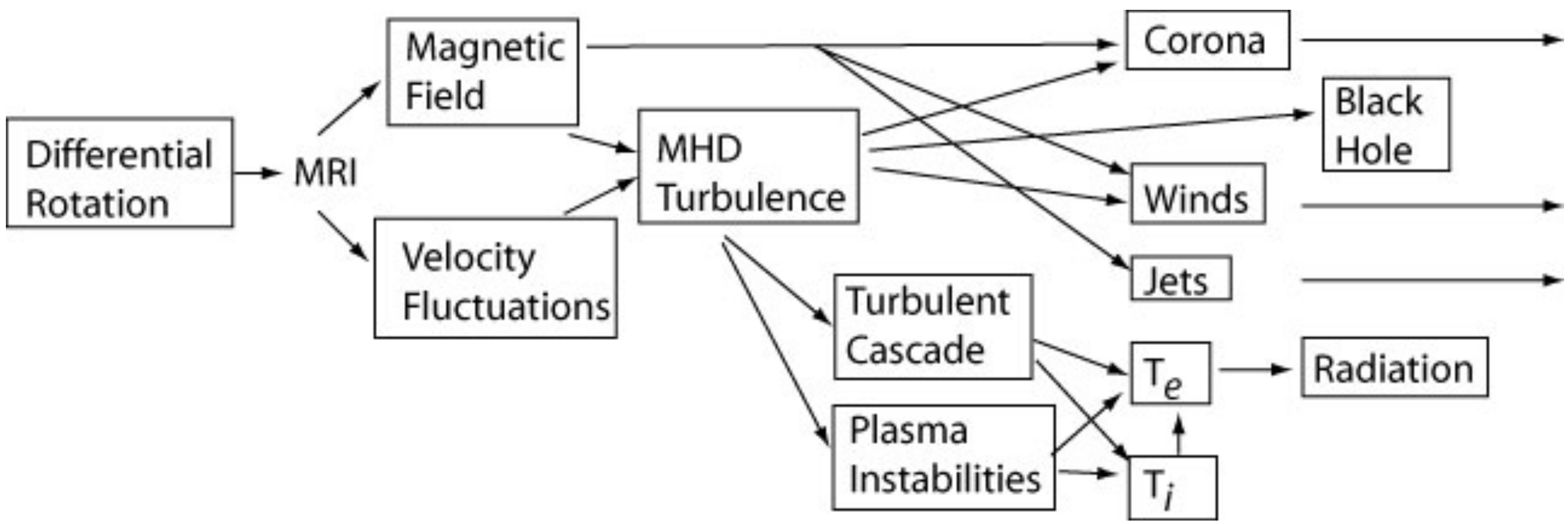
The Importance of Magnetic Fields

Magnetic fields make the ionized gas in an accretion disk spiral inward. The *magneto-rotational instability* (MRI) is important in accretion disks because it converts stable orbits into *unstable* motion.

Magnetic fields can create stresses inside the marginally stable orbit around a black hole, significantly increasing total efficiency.

Magnetic fields can extract energy and angular momentum from spinning holes and drive jets.

Energy flow in Accreting systems



Physics: Ideal MHD, relativistic gravity, resistivity, Hall effect, ambipolar diffusion, plasma physics, pair plasmas, emission/absorption/scattering, self-gravity, relativistic optics

Computational Challenges: Space, Time, Velocity

- Disks are three-dimensional – turbulence and magnetic dynamo essential
- Disks are huge, from black hole horizon to parsec
- Disks are thin: Vertical thickness H much less than R
- Disks are supersonic: sound speeds much less than orbital speed; net accretion inflow velocity much less than sound speed
- Disks can be relativistic – orbital speed ~ c, temperatures ~ mc^2
- Orbital periods vary as $R^{3/2}$ – dynamical processes at each radius
- Stress and dissipation due to MHD (radiation) turbulence – scales much less than H
- *Local* disk simulations ~ adequately resolved with 32-64 zones per H
- Simulating whole system impractical – work on sub-problems and develop hierarchy of models, including subgrid models

Codes and Algorithms For Accretion

- Minimum requirement – 3d MHD plus external gravity
- Numerical approaches: Finite difference, SPH, spectral – ongoing algorithm development
- Codes: ZEUS, Athena, PLUTO, Flash, NIRVANA, GRMHD, HARM3d, COSMOS++ - often several versions of each type of algorithm
- Additional physics in some codes: special and general relativity, non-ideal MHD, collisionless plasma, self-gravity, ambipolar diffusion, Hall terms, flux-limited diffusion, ionization, chemistry
- Most more complex physics simulations are local rather than global

The Need for Speed

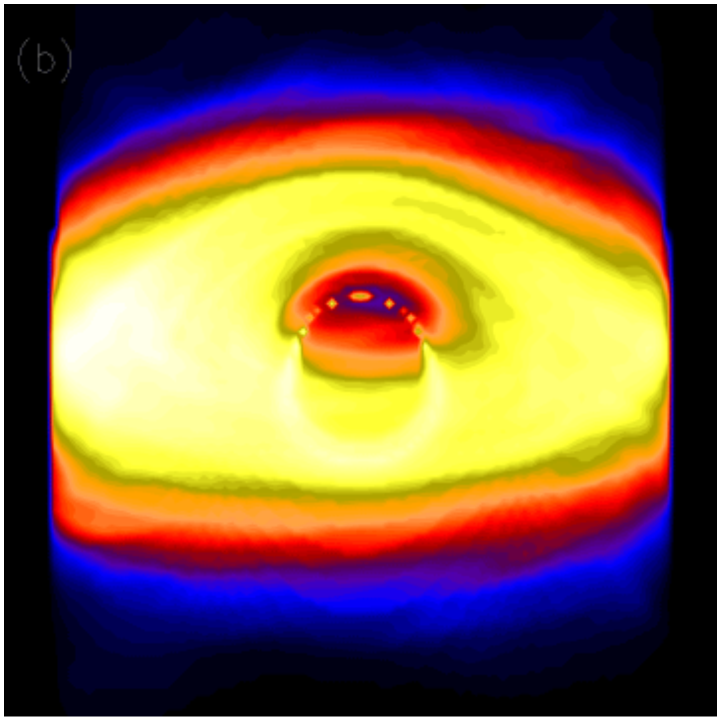
Floats required = $(\text{Zones/dim})^N$ x timesteps x flops/zone

In log: 3 x 3 + 6 + 4 = 19 = 10 E floats

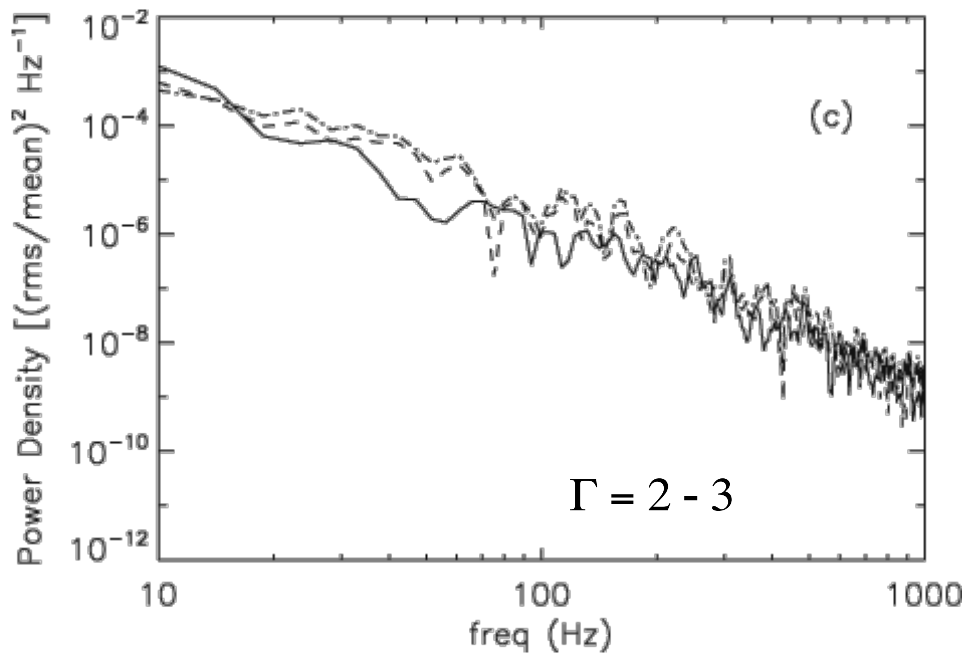
Code Development Challenges

- Application developers focus is on the *algorithm*, not necessarily good code design
- Typical code design does not take advantage of new paradigms and practices; legacy thinking as well as legacy coding
- Scaling distinct from performance – need to address both
- Inadequate attention paid to data management and appropriate data structures
- But: code must be clear, self-documenting, maintainable – can be at odds with performance

Comparison of simulation with observation: Simulated emission



Optically thin line emission
Inclination angle 70 degrees



Power spectrum from simulated light curve

From Schnittman, Krolik & Hawley 2006, ApJ, 651, 1031

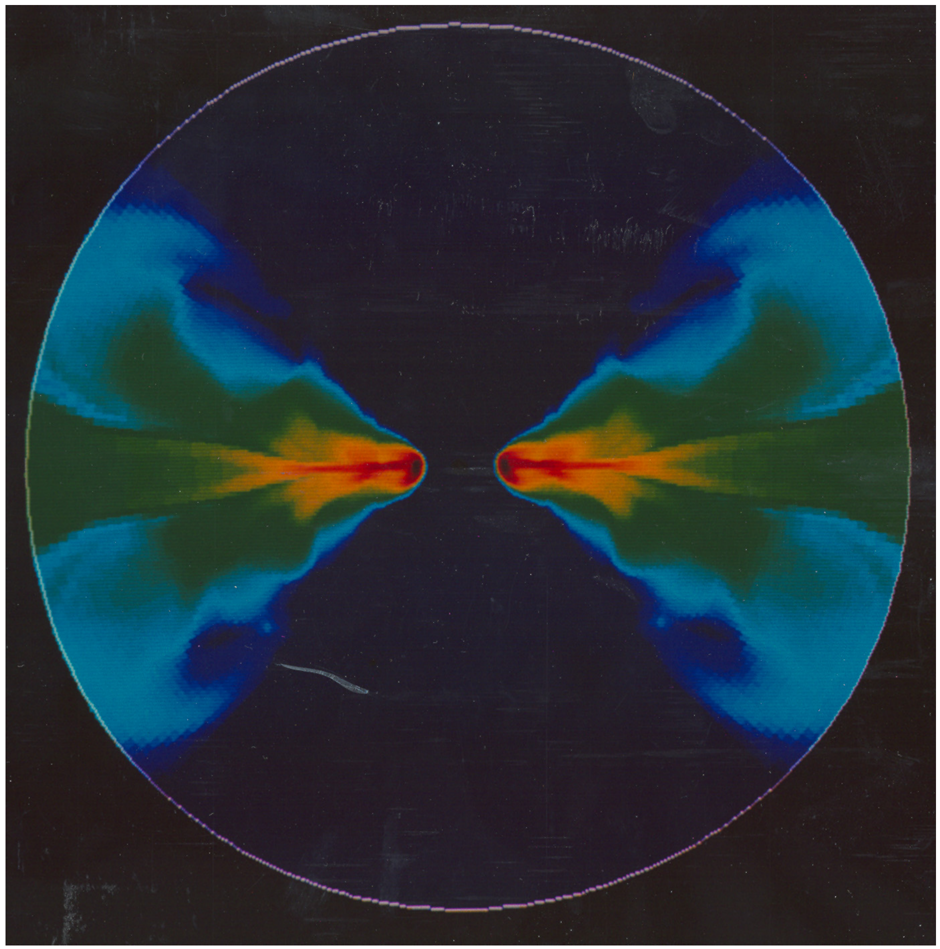
Data and Analysis

- Large 3D datasets – many time slices – complex physics
- Comparison with observation will require a well-developed data pipeline
- No agreed upon standards for data files, diagnostics
- No standards for data interfaces or interoperability of analysis routines
- What is worthwhile for sharing with the community? How should that be done?
- How should data be archived? What data should be archived?

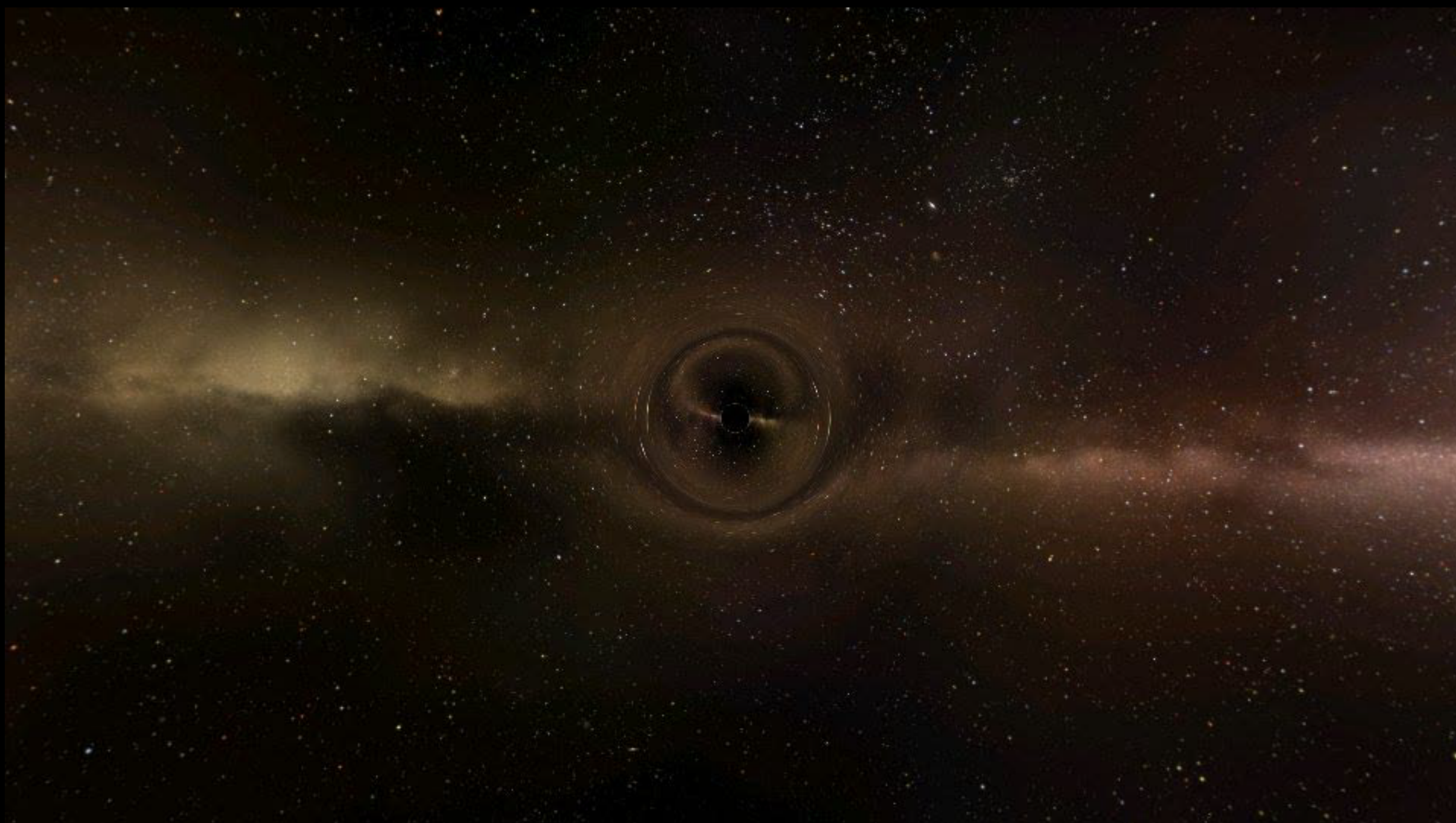
Visualization

- Visualization of large-scale time-dependent 3D simulations difficult
- Open source and funded-project products not maintained, difficult to use, often buggy
- Commercial solutions (e.g. IDL) are useable and well maintained, but scaling is a problem

Visualization circa 1984



Visualization 2010



Collaboration, Education and Training

- Community grew up with "single warrior" mode
- Historically, only occasional funding opportunities for building collaborations (new support recommended by Decadal Survey)
- Graduate curriculum often doesn't include computational science - departments usually don't have additional capacity
- What graduate training there is: Pick it up from the advisor who learned programming 25 years ago, Physics course teaches bad C; CS courses teach Java
- Few CS faculty or CS departments are interested in applications, *per se* – the responsibility is ours
- But it is inherently multi-disciplinary
- UVa CS 6501 graduate course – Matlab, Python, R, F95

Collaborators and Acknowledgements

- Julian Krolik, Johns Hopkins University
- Xiaoyue Guan, Virginia
- Scott Noble, RIT
- Jeremy Schnittman, JHU
- Kris Beckwith, Jake Simon, Colorado
- Jean-Pierre De Villiers
- Andrew Hamilton, Colorado
- Texas Advanced Computing Center, NSF TeraGrid
- National Institute for Computational Sciences, NSF TeraGrid
- NSF Grant AST-0908869
- NASA Grant NNX09AD14G