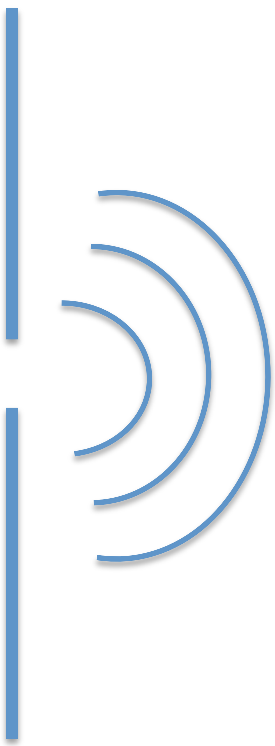
Lecture 12

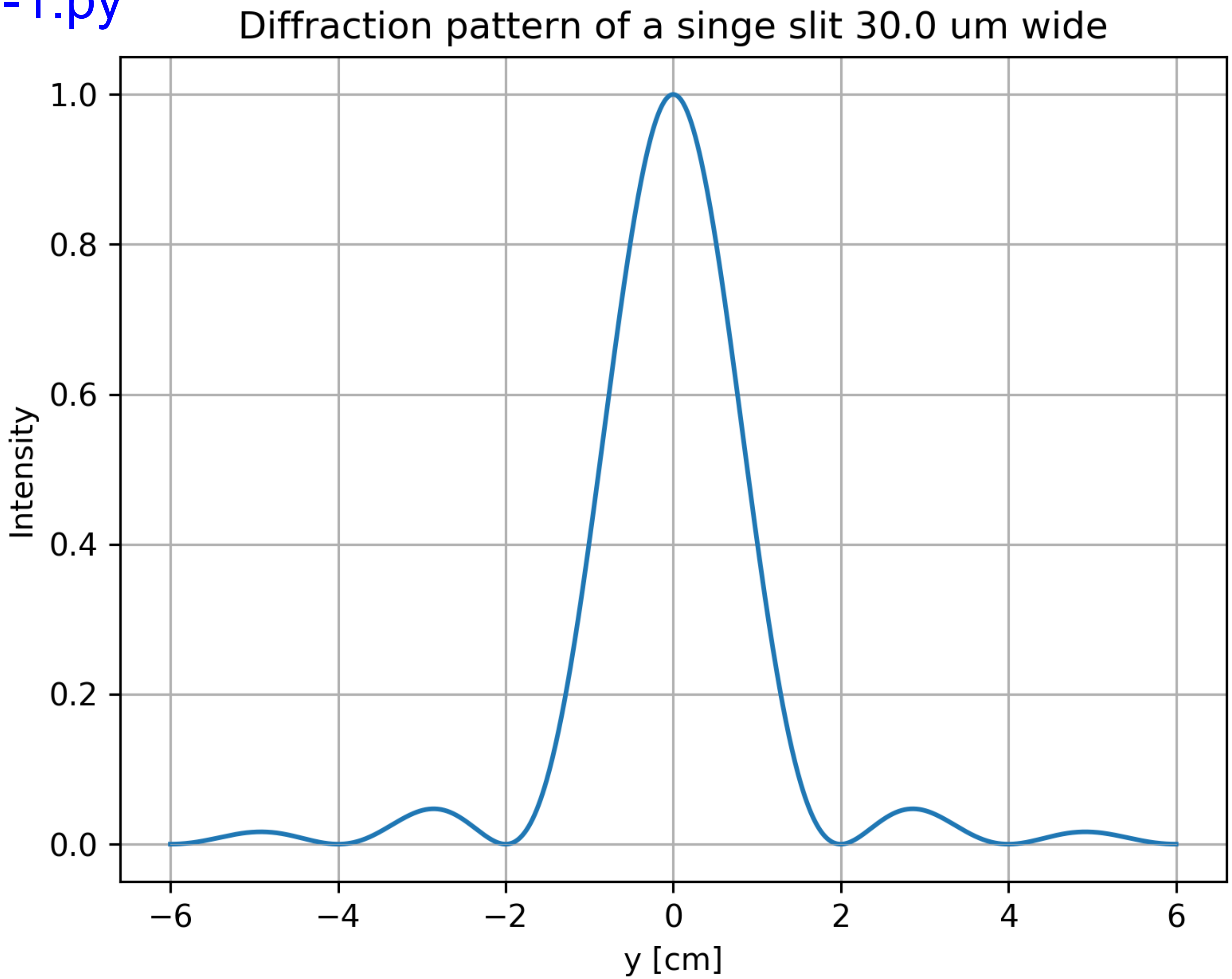
- Solutions of 4th assignment set
- **ODEs. Ordinary diff. eqs. (cont.):**
  - Astrophysical N-body problems:
    2-Body problem in pseudo-Newtonian GR, 3-Body problem, R3B (restricted 3B), circular R3B
  - Symplectic integrators for astrodynamics, 4th order
  - UTSC research on massive N-body calculations
  - Cosmological N-body simulations: Millenium & Bolshoi
- **Some PDEs (Partial differential equations):**
  - Heat or diffusion equation (unsharp masking algorithm)
  - Wave equation in 2 dimensions:
    Pond or swimming pool surface,
  - Young's double slit experiment
  - UTSC research on CFD
- **Overflow topics**
  - FFT

Solution of some assignments #4 - problem 1

Compute diffraction pattern

- diffraction-1.py

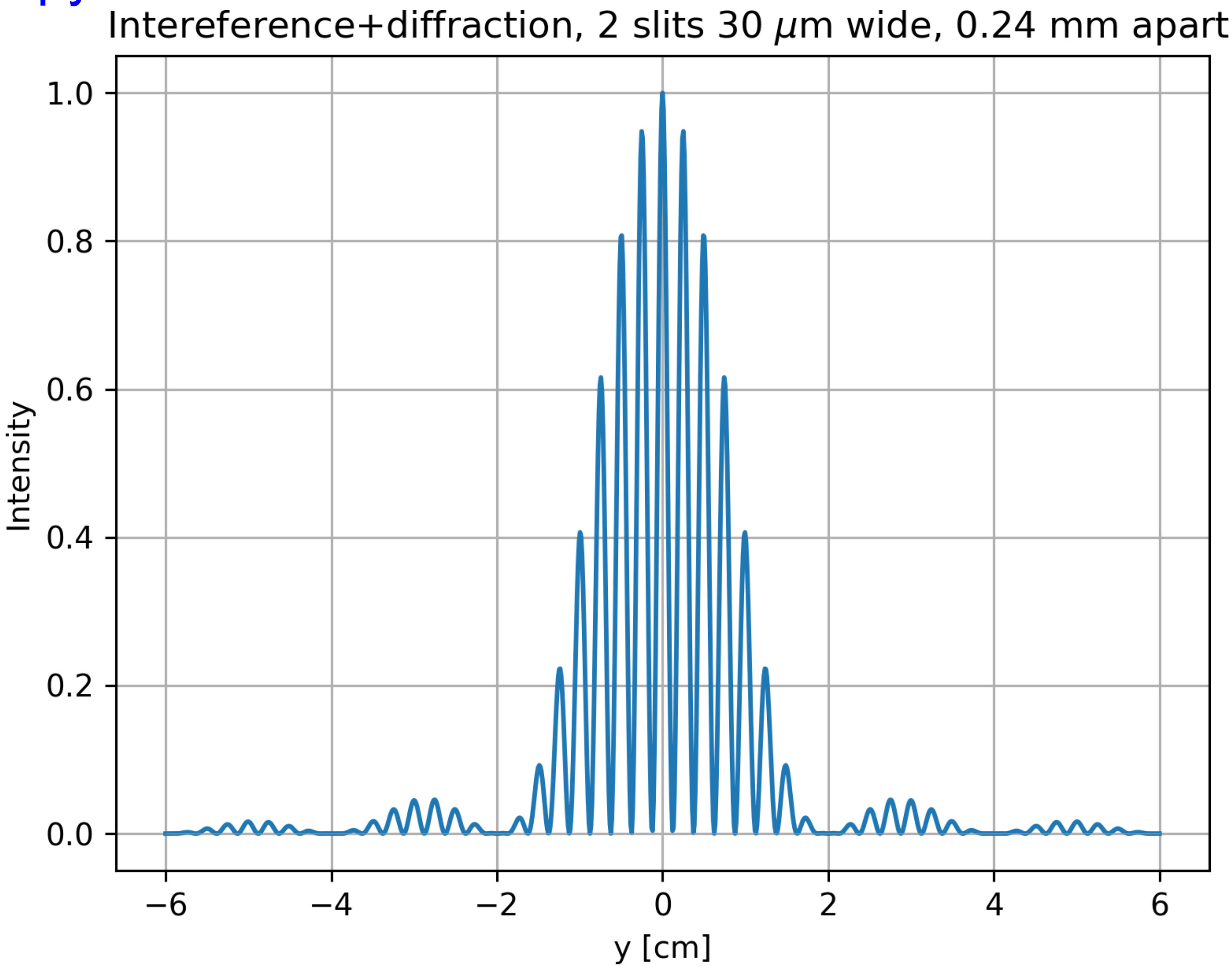




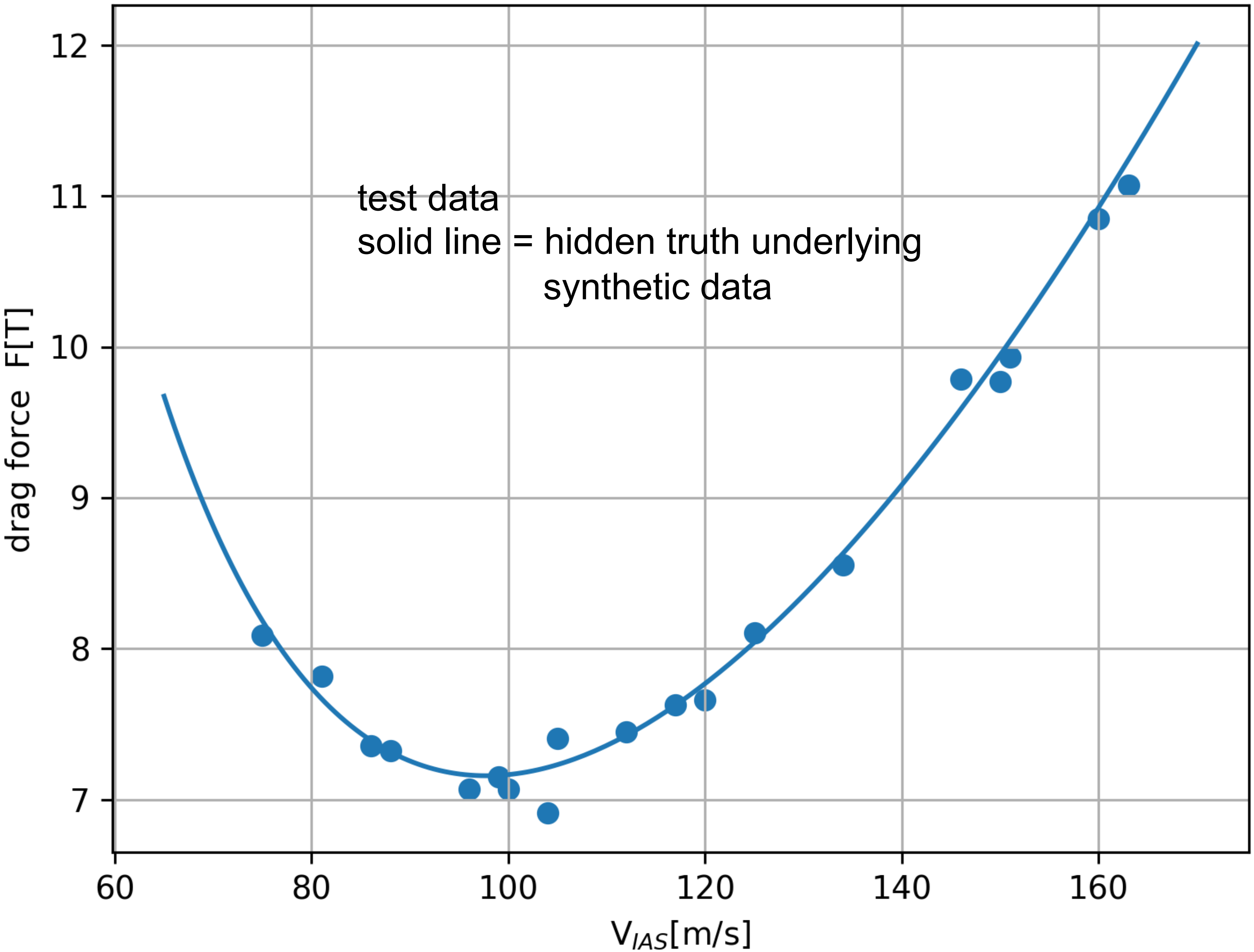
Solution of some assignments #4 – extension

Compute diffraction+interference pattern from 2 slits

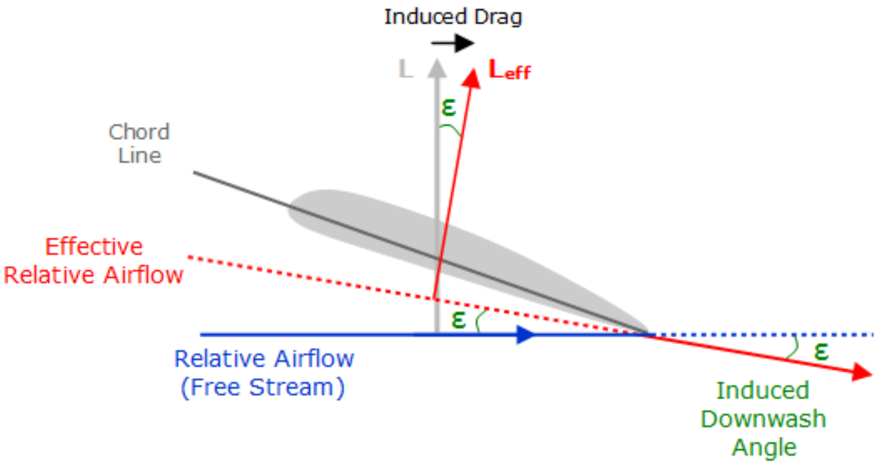
interference-1.py



Solution of assignments #4 – problem 2



WHY IS THERE INDUCED DRAG?



- $C_L \sim AOA$

  lift coeff.

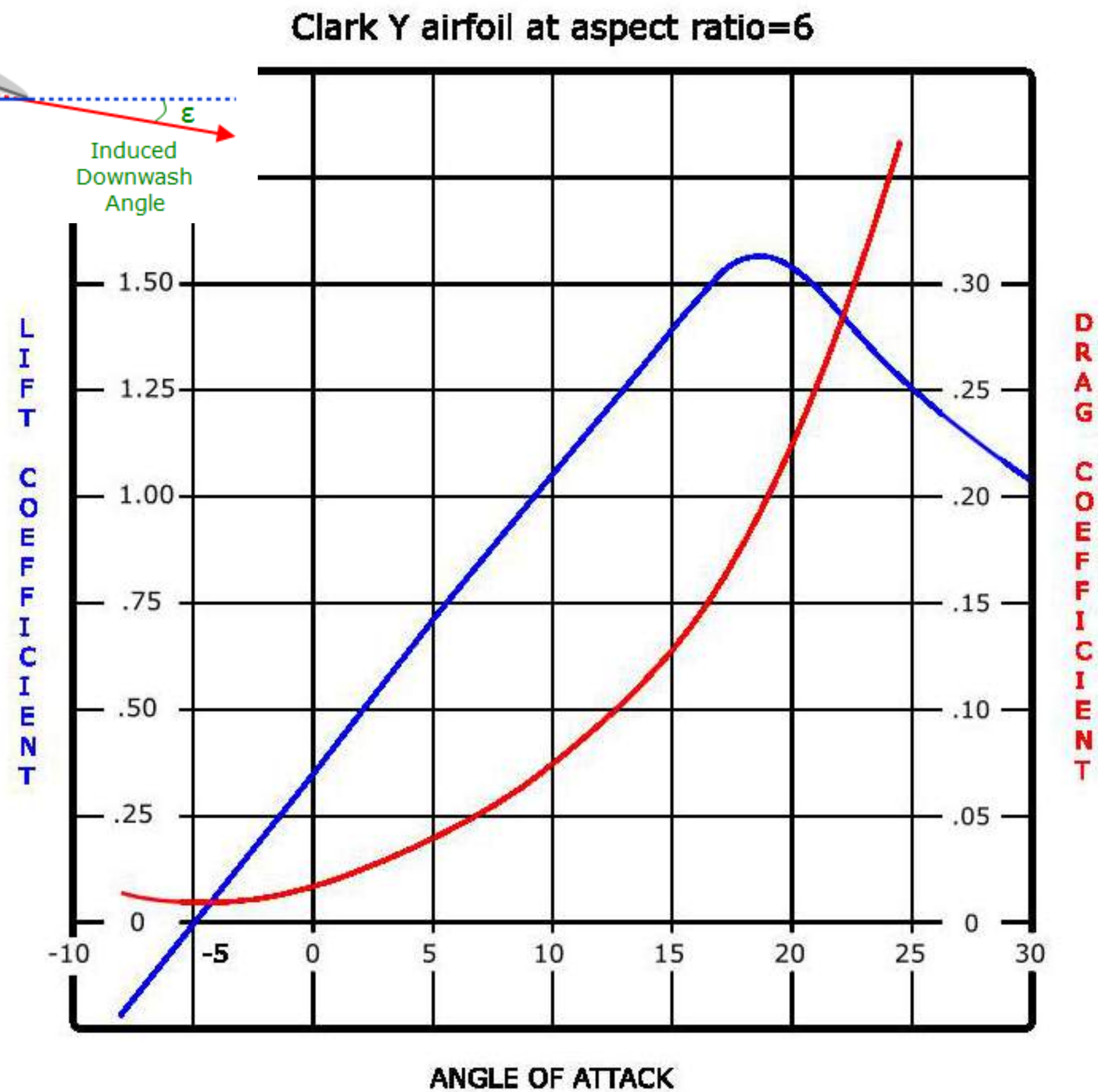
- $C_{id} \sim AOA^2$

  $\sim C_L^2$

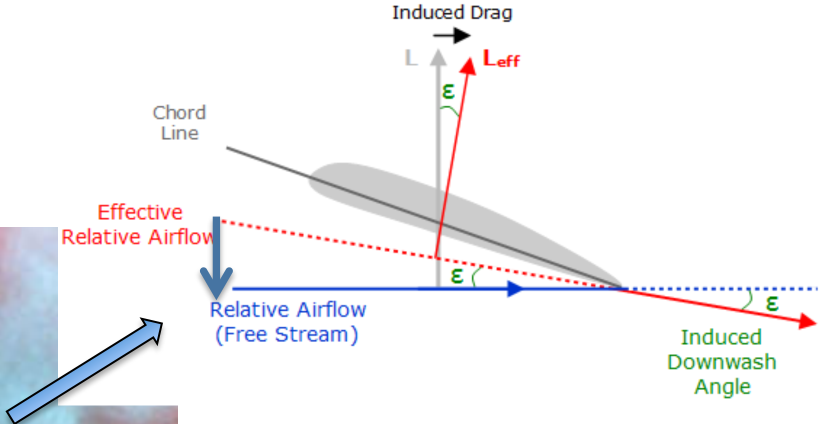
  induced drag coeff.

$C_{pd} \sim const$

parasitic drag coeff.



Why is there induced drag?
Because of what goes on in 3-D!



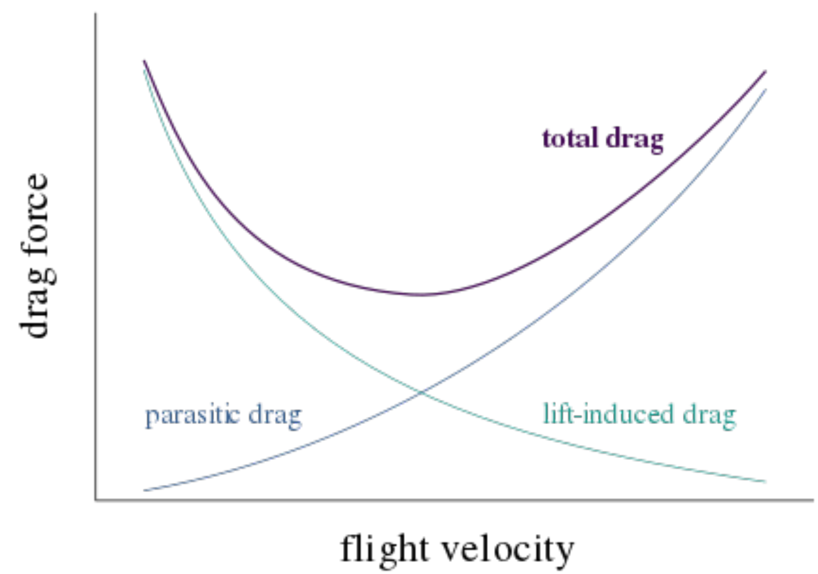
Lift and drag according to airfoil theory

- dynamic pressure $p_{ram} = \frac{1}{2} \rho V^2 \sim V^2$; ρ = density of air, V = airspeed
- force of drag = parasitic drag + induced drag
- $F_d = (C_{pd} + C_{id})\, p_{ram}\, A$ A = area of wing

- lift force = weight (by assumption)
- $F_L = C_L\, p_{ram}\, A = W = \text{const.}$, $C_L \sim W/(A\, p_{ram})$
- $C_{id} = C_L^{\,2} A/(\pi L^2)$ ← a result of airfoil theory (L = wingspan)
- Drag force is then
- $F_d = C_{pd}\, p_{ram}\, A + W^2/(\pi p_{ram} L^2) \sim k\, V^2 + q\, V^{-2}$

Constants k, q can be obtained from testing the aircraft.

From k and q, one derives $V(F_d = \text{min})$, as well as $V(\text{power} = V\, F_d = \text{min})$, and finally $V(F_d\, \text{dist}/V = \text{min})$.



- Drag force
- $F_d = C_{pd}\, p_{ram}\, A + W^2/(\pi p_{ram} L^2) \quad =: \; k\,V^2 + q\,V^{-2}$
- $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad F_{pd} \qquad\quad F_{id}$
- $V(F_d = \min)$ corresponds to longest range of glide:

$F_L = W \cos\theta$

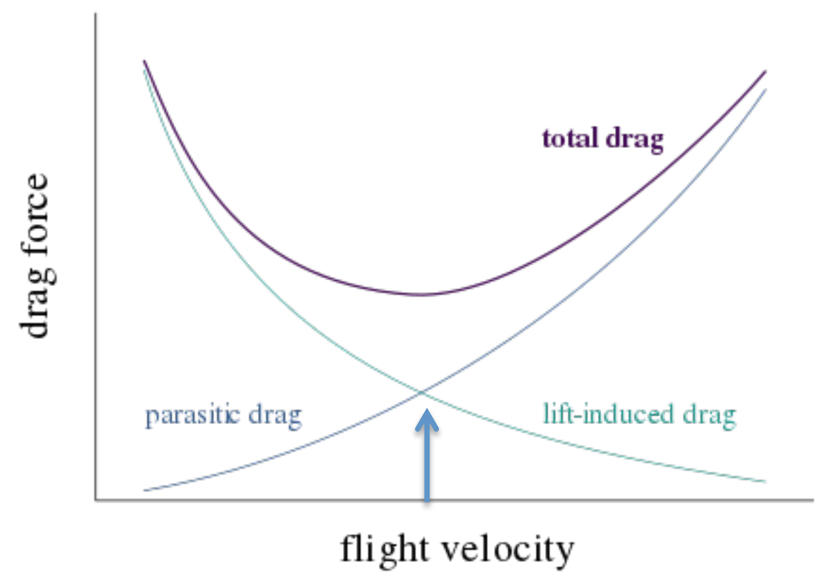
$F_d = W \sin\theta$ min $\tan\theta$ at minim. F_d/F_L i.e. min F_d

Using calculus,

$dF_d/dV = (2/V)(kV^2 - qV^{-2}) = 0$

when $F_{pd} = F_{id}$

$V(\min F_d) = (k/q)^{1/4}$



$F_d = C_{pd}\, p_{ram}\, A + W^2/(\pi p_{ram} L^2) \;=:\; k\,V^2 + q\,V^{-2}$

- $V(VF_d = \min)$ corresponds to longest time of glide without engine power, or minimum engine power in horizontal flight, or minimum fuel burn rate.

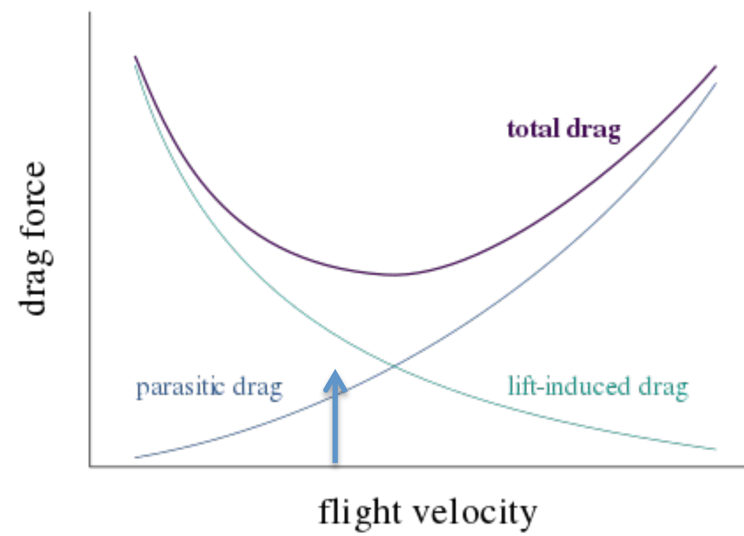
Power $= V\,F_d(V) = k\,V^3 + q/V$

Using calculus,

$dPower/dV = (1/V)(3kV^2 - qV^{-2}) = 0$

when $3F_{pd} = F_{id}$

$V(\min F_d) = (k/q/3)^{1/4} = 0.759\,(k/q)^{1/4}$



$F_d = C_{pd}\, p_{ram}\, A + W^2/(\pi p_{ram} L^2) \quad =: \; k\, V^2 \; + q\, V^{-2}$

- Carson's speed minimizes product of travel time and fuel consumed between points A and B

$V(\, F_d/V = \min)$

time = distance / V,

fuel consumed ~ energy ~ F_d distance

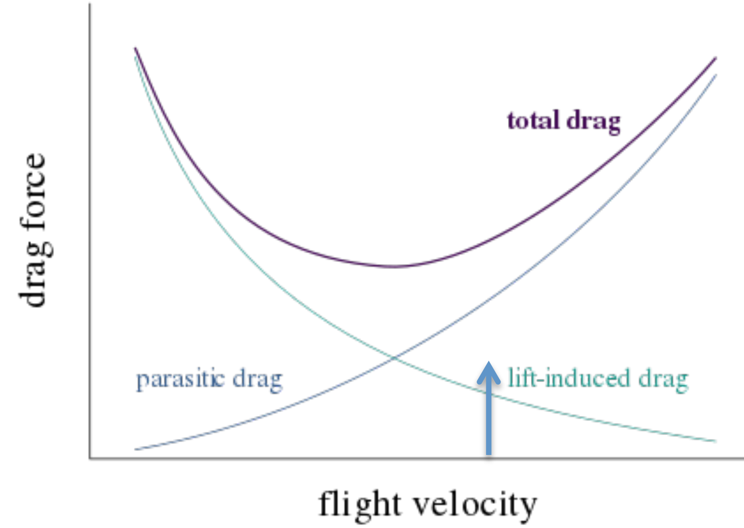
time * fuel ~ $F_d(V)/V = k\,V + q/V^3$

Using calculus,

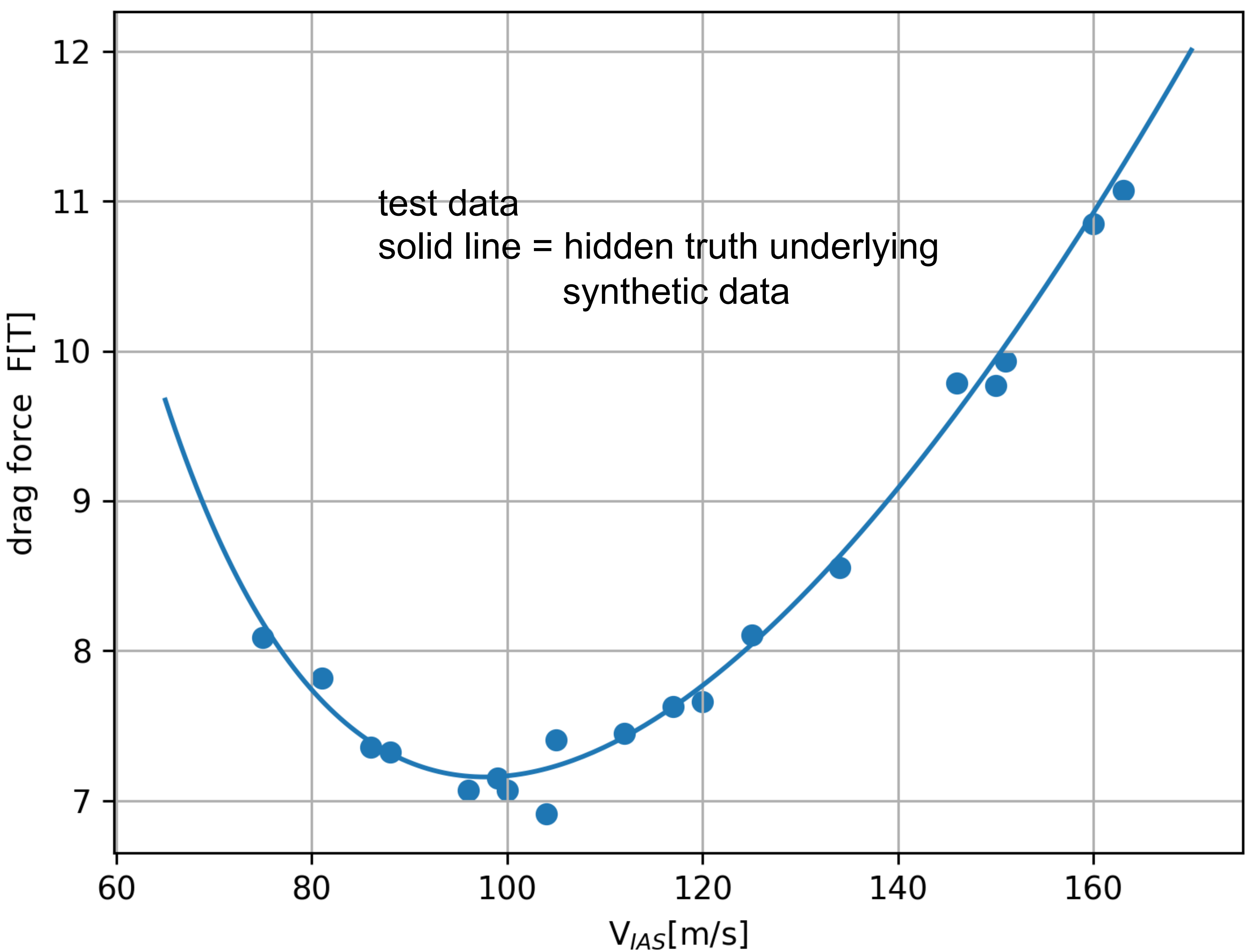
$d\,(F_d/V)/dV = (1/V^2)(kV^2 - 3qV^{-2}) = 0$

when $F_{pd} = 3F_{id}$

$V(\min F_d) = (3k/q)^{1/4} = 1.316\,(k/q)^{1/4}$



linear combination of V^2 and $1/V^2$



Solution of assignments #4 – problem 2

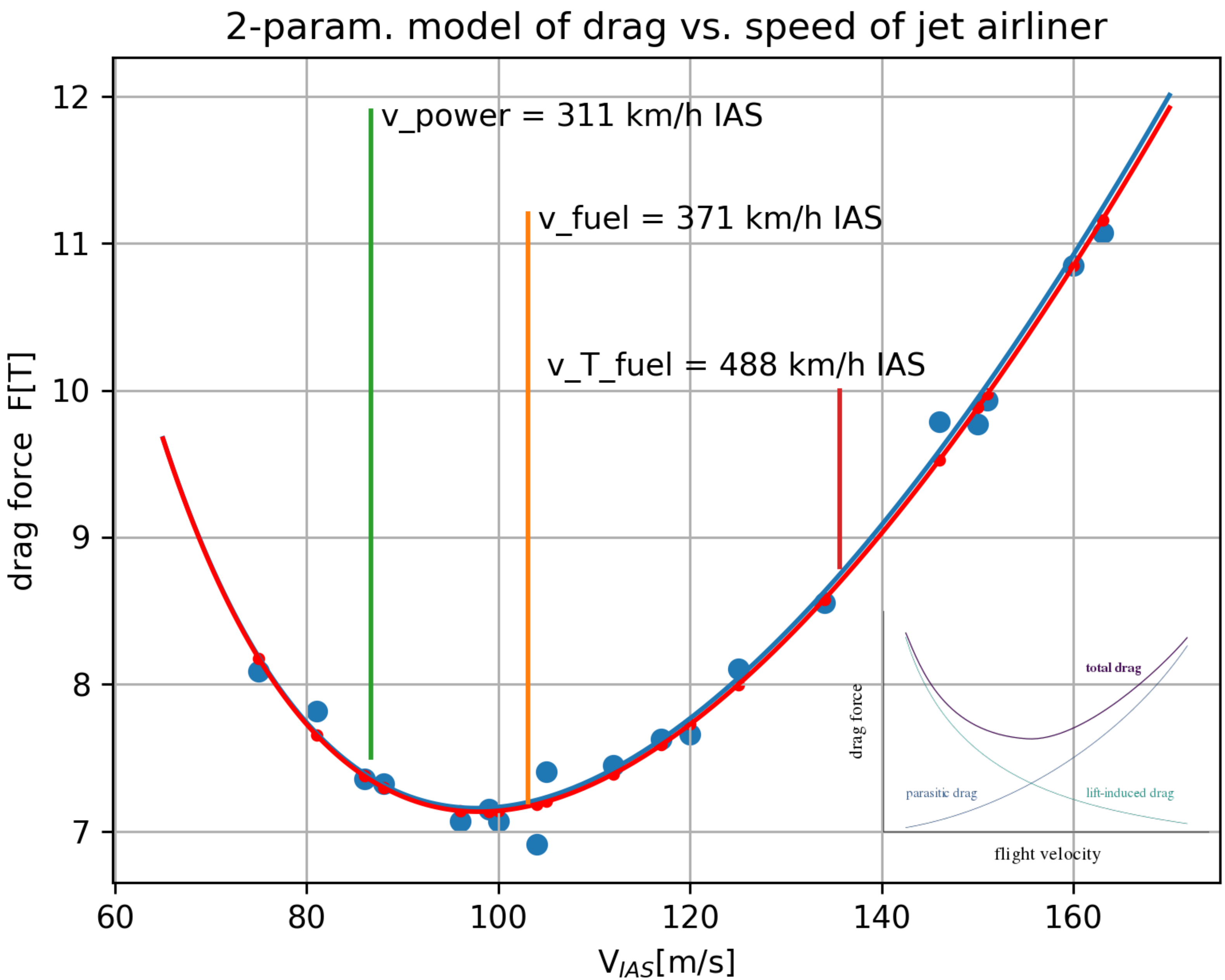
- Least Squared fit to aeronautical test data
- fit-drag-1.py

```
•  Pawels-MacBook-Pro[136]:~/py3% python3 fit-drag-1.py

•   assumed A,M,L, 200 90000.0 37.7
•   Ap, W_L 6.0 23419.09814323607
•   v data [ 75.  81.  86.  88.  96.  99. 100. 104. 105. 112. 117. 120.
•  125. 134. 146. 150. 151. 160. 163.]
•   generated Fdata [T] [ 8.08596424  7.81498155  7.35682356
   7.32279163  7.06857043  7.14673581 7.06621127  6.91026535
   7.40488755  7.44936492  7.62545887  7.6619365  8.10453331
   8.55727423  9.78611397  9.77122489  9.93360421 10.8494722
    11.07041866]
•  fit?
•   obtained parameters:  335740760.  5.95166
•   obtained ratios    :  1.0012410   0.991943
•   best speeds:
•   v_T_glide = v_min_pow =  86.6645386 m/s   311 km/h
•   v_glideslope = v_fuel =  103.06209   ,,   371  ,,
•   v_T_fuel   = v_Carson =  135.63733   ,,   488  ,,
```

Why is Carson's V optimizing time and fuel so different from real airspeeds of airliners? It isn't. The 488 km/h is instrumental speed (speed gauges are calibrated using standard sea-level air density $\rho = 1.225\ kg/m^3$). But ρ at cruise altitude is 2.7 times lower, and true airspeed is $2.7^{1/2}$ times higher (TAS~800 km/h). Real speeds are similar to Carson's V.

Solution of assignments #4 – problem 2



Solution of assignments #4

- problem 3 - RK4 integration of Lorenz chaotic system
- problem 4 – estimation of parameter uncertainty in Least Squares Method

The only tricky point was to avoid the possible misunderstanding of how much to perturb the data. The answer is this:

so much that the particulars of noise change (realization of random perturbation differs)

but the amount of spread around the linear (in this case) trend is remains the same.

I’m curious how you did that, we’ll see what approaches you found.

◆ODEs. Ordinary diff. eqs.:

- N-body problems:
- *From our home page:*

  3-Body problem, R3B (restricted 3B), circular R3B.

- UTSC research on supercomputing N-body systems
- Cosmological simulations: Millenium, Bolshoi

◆PDEs. Partial differential equations:

- Wave equation in 2 dimensions

  $Z\_tt = c^2 \ (Z\_xx + Z\_yy)$ PDE, c = speed of the wave

  _tt = second time deriv., _xx = second deriv after x, etc.

- Pond or swimming pool surface
  - pond1.py, pond3.py, pond4.py
  - pond4-1obj.py,
  - Young's double slit experiment:

    pond4-2slit3.py, pond4-2slit4.py
- CFL condition
- research on CFD

◆ODEs. Ordinary diff. eqs.:

- N-body problems:
- *From our home page:*

  3-Body problem, R3B (restricted 3B), circular R3B.

  see program hill3.pro in IDL language
  and all of its graphical output on our course page

- UTSC research on supercomputing N-body systems

- Cosmological simulations: Millenium, Bolshoi

Massively parallel integration on the newest HPC platforms: CPU, GPU and MIC

15 Aug 2017, UTSC

(...)

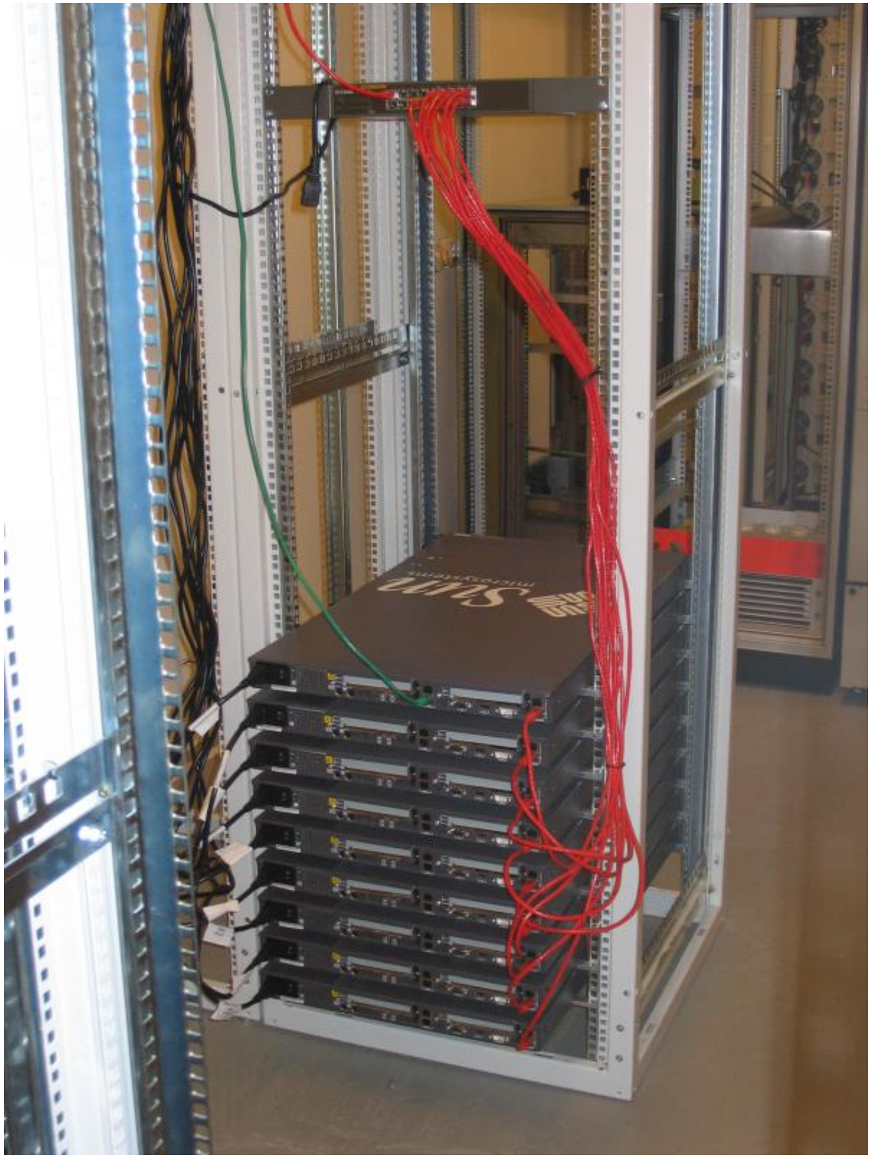
3. Concurrent simulation of 200 or 7000 planetary systems on CPUs or MIC

Conclusions

4. Collisionless gigaparticle disks. Interaction with binary system.
   - Hybrid algorithm (4^{th} order symplectic with collisions)
   - Implementation and optimization in Fortran90 on 1..32 MIC (Φ)
   - Migration problem
   - Tests and preliminary results
   - Fast migration in particle disks as type III CR-driven migration

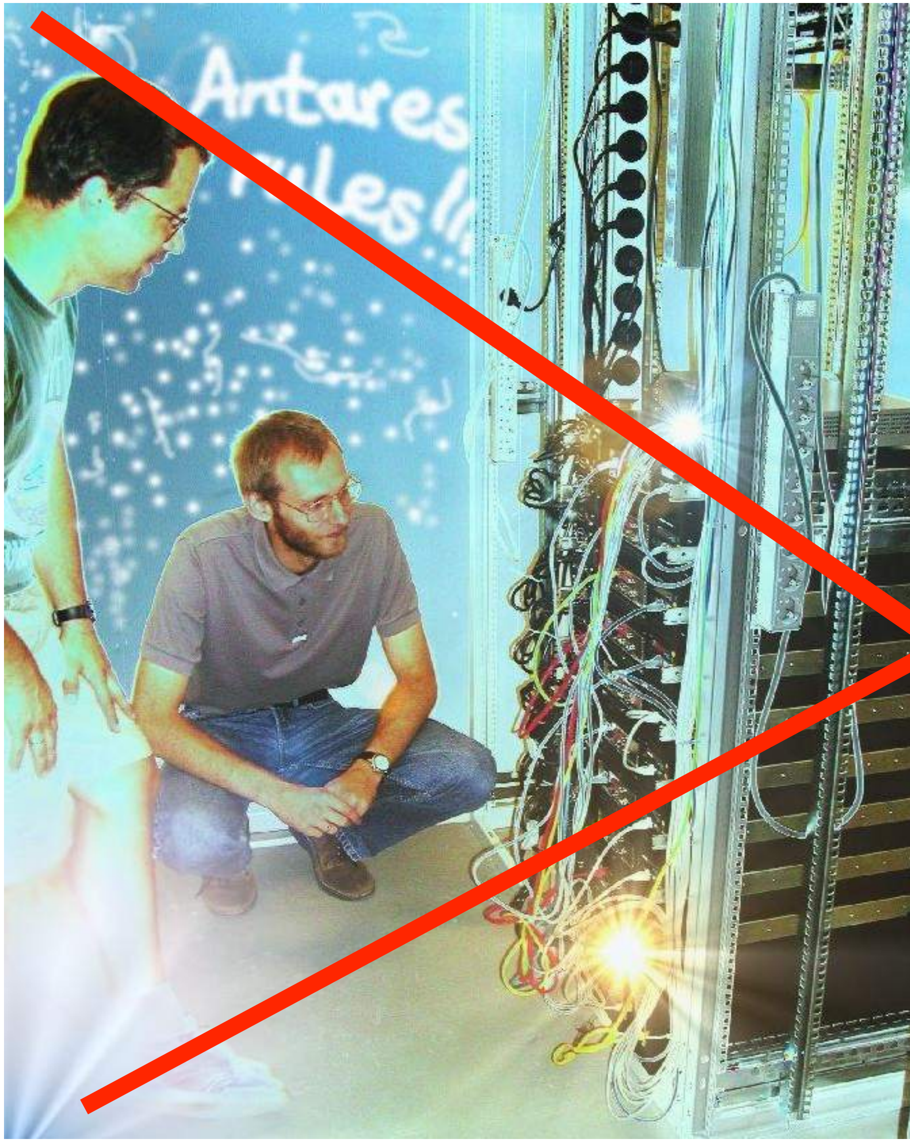
Conclusions

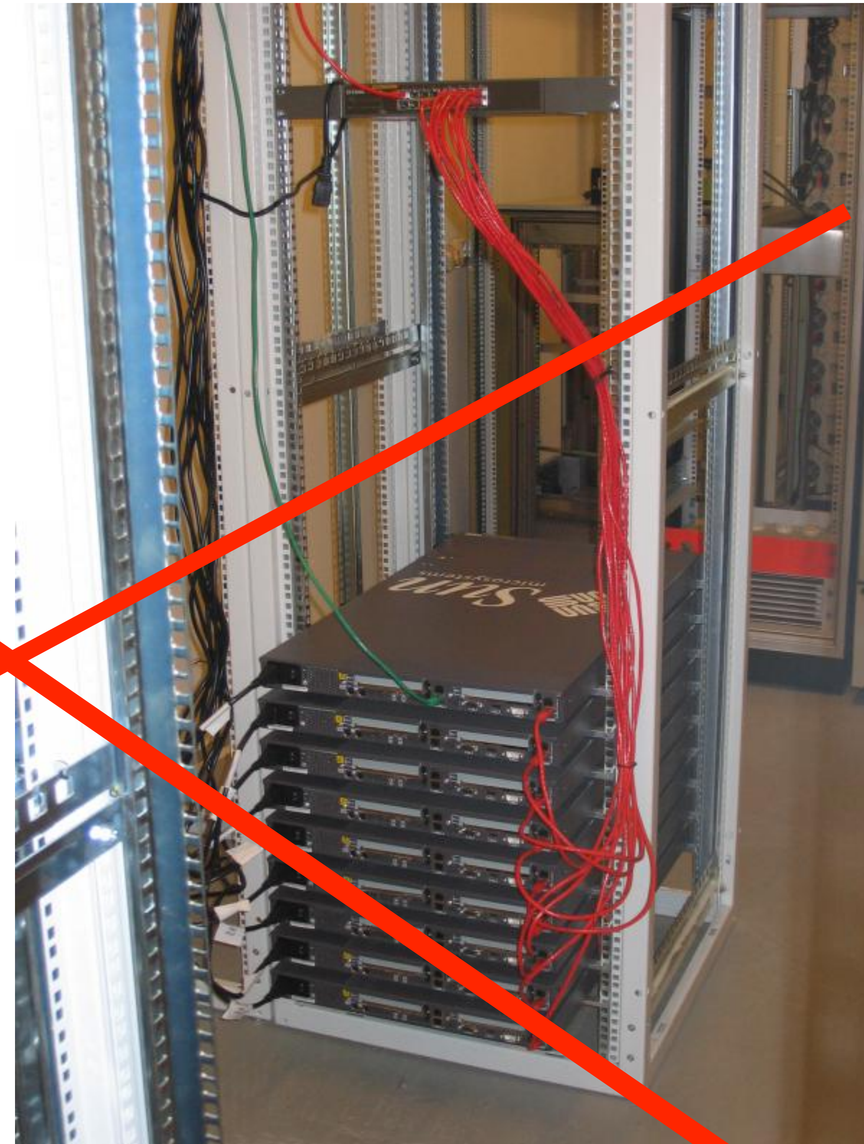
1990s and 2000s was the era of clusters



MPI for parallelization

For many years in 2000s we thought…

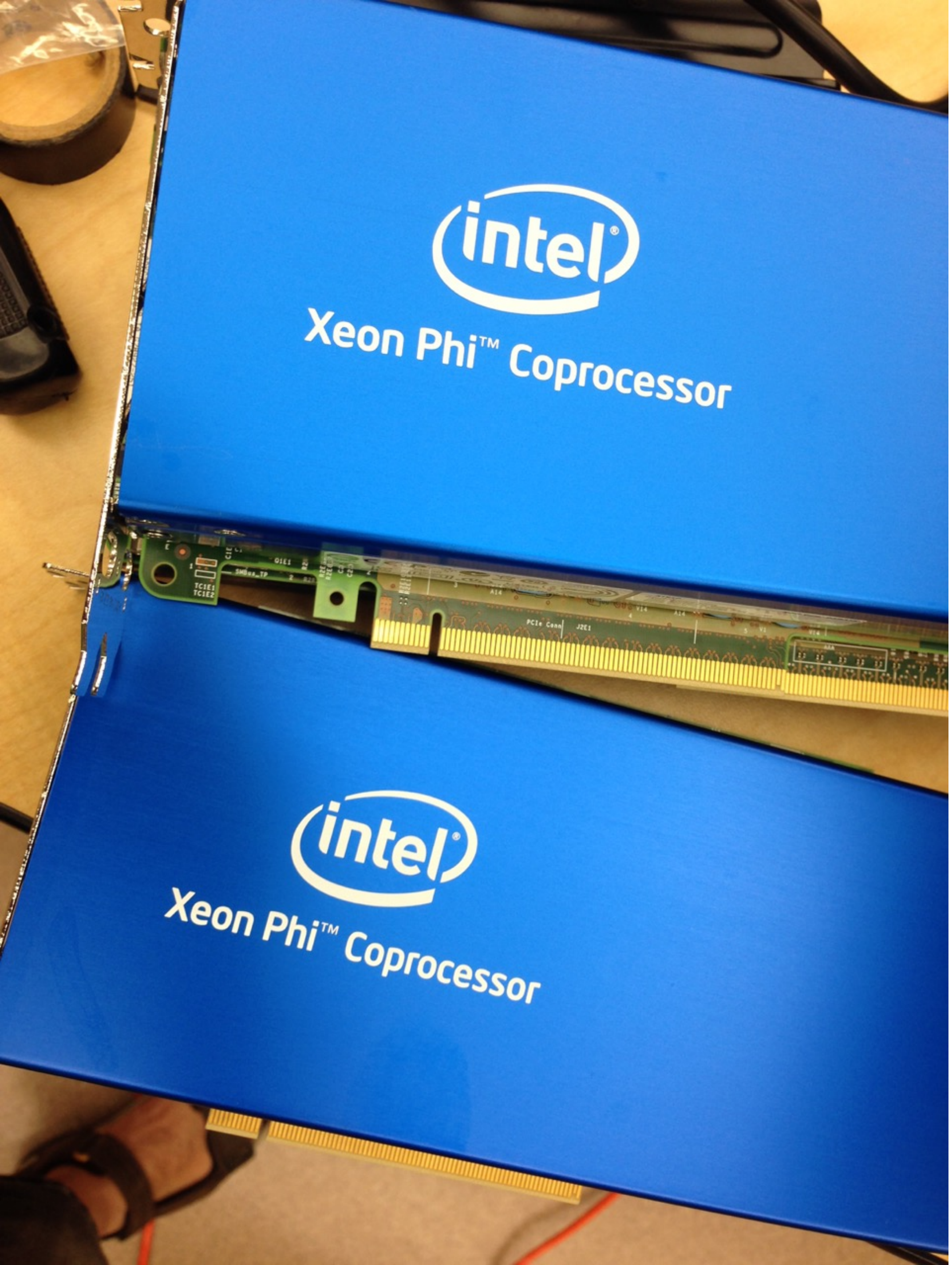




...but were wrong

MIC

many integrated cores
(Intel’s name for massively parallel CPU-like processors)



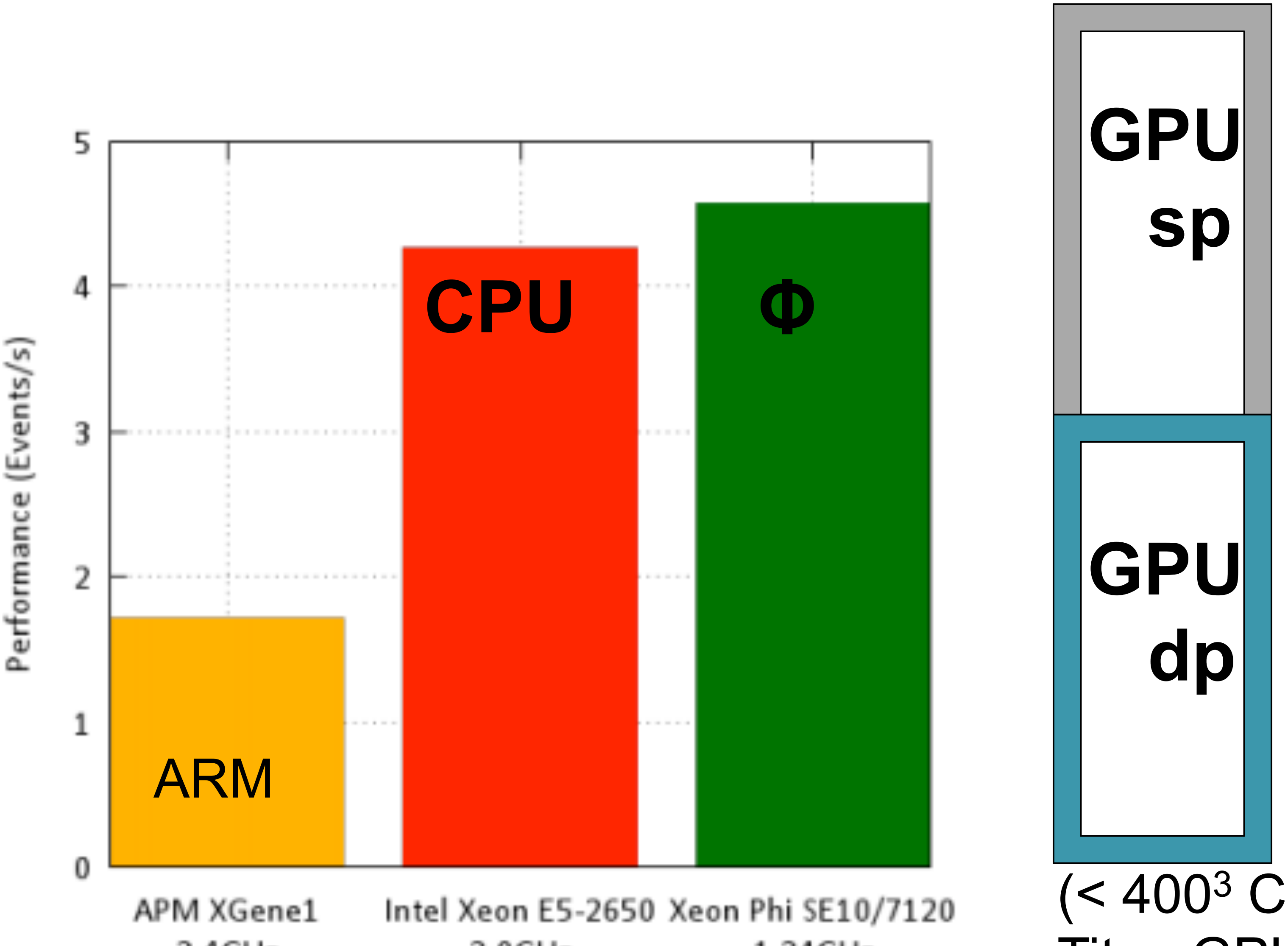
MIC = many integrated CPU-like cores (~60)

Intel Xeon Phi accelerators

Knights Corner:
~1 TFLOP dp
~2 TFLOP sp

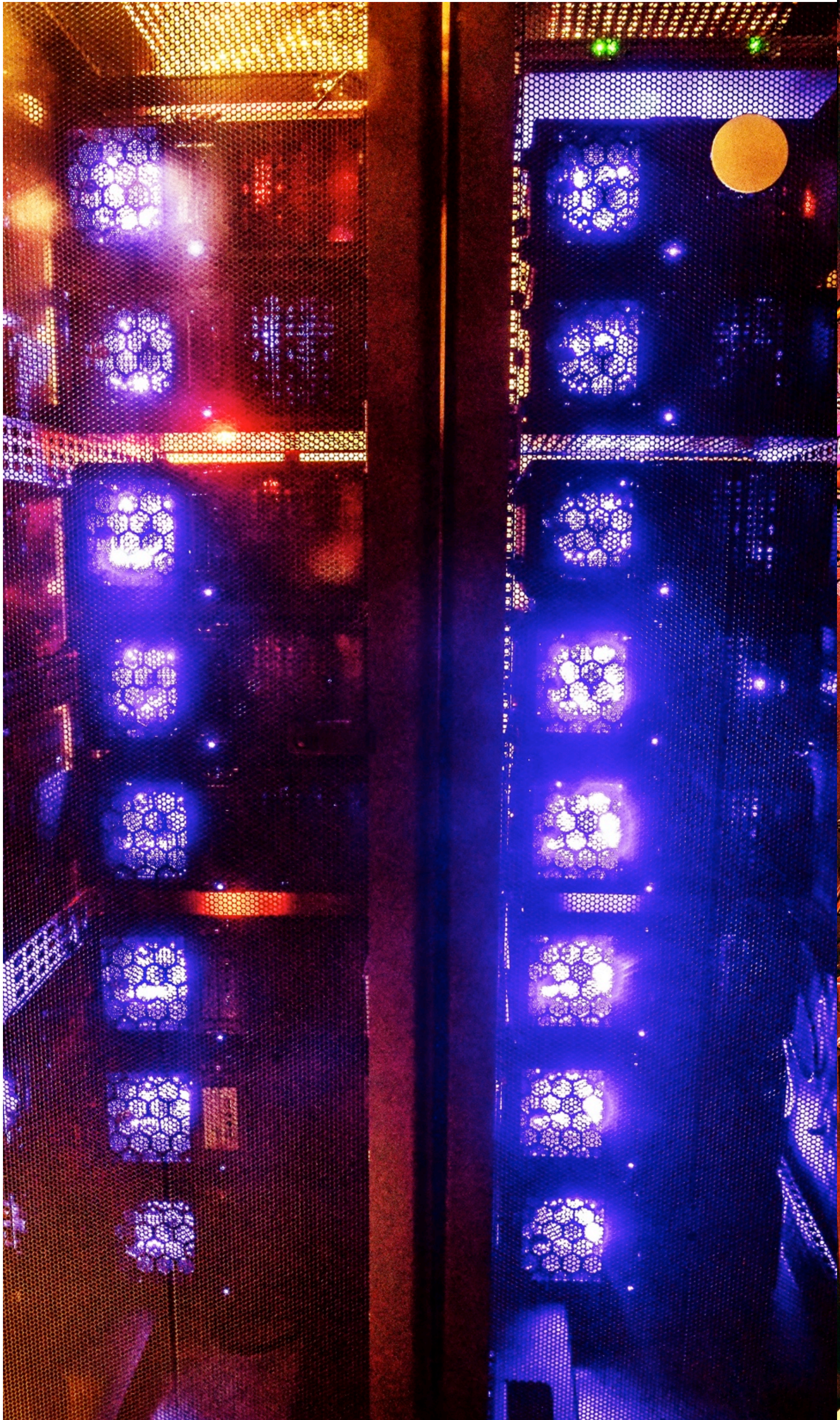
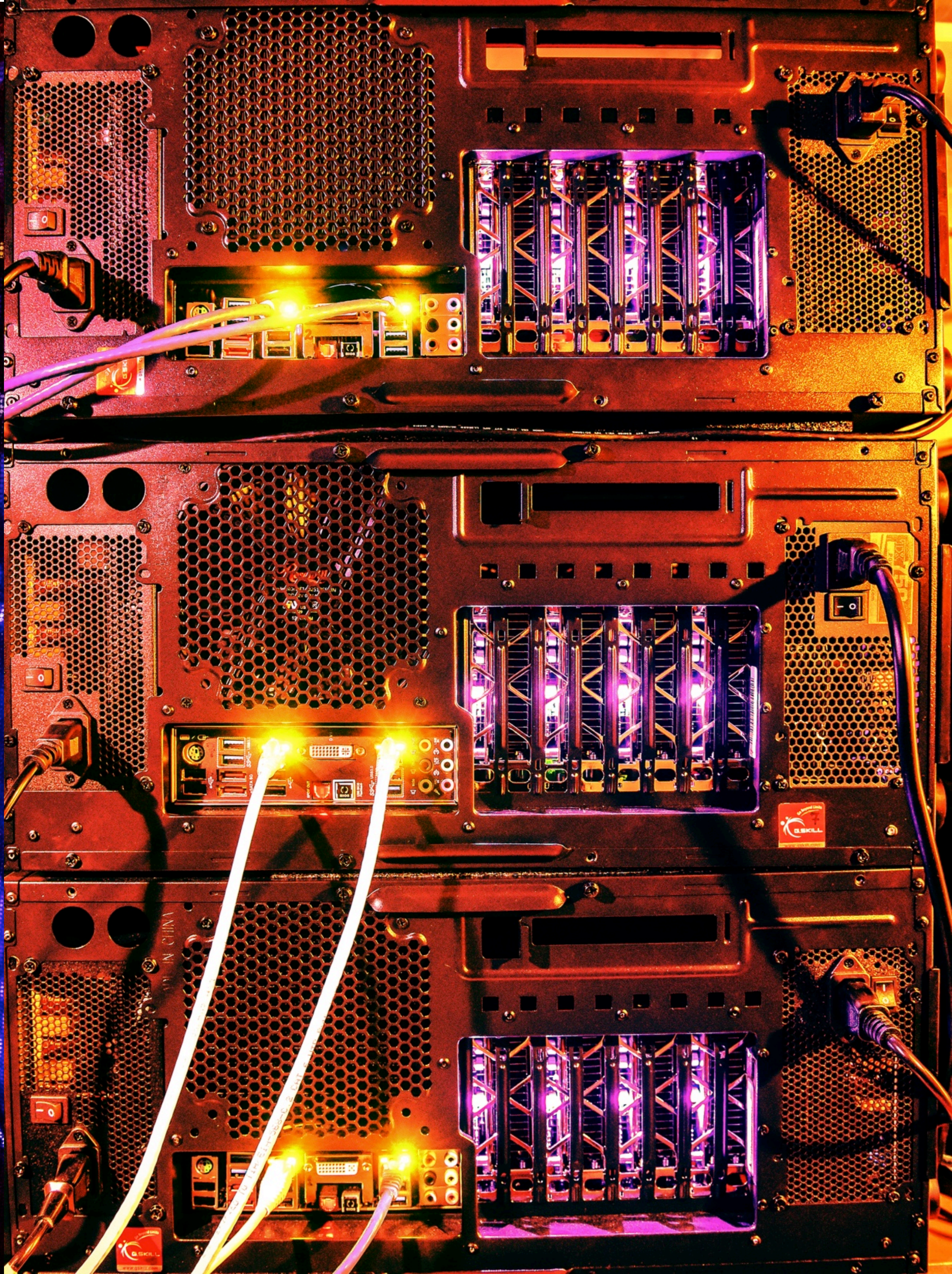
Knights Landing:
~3x more TF

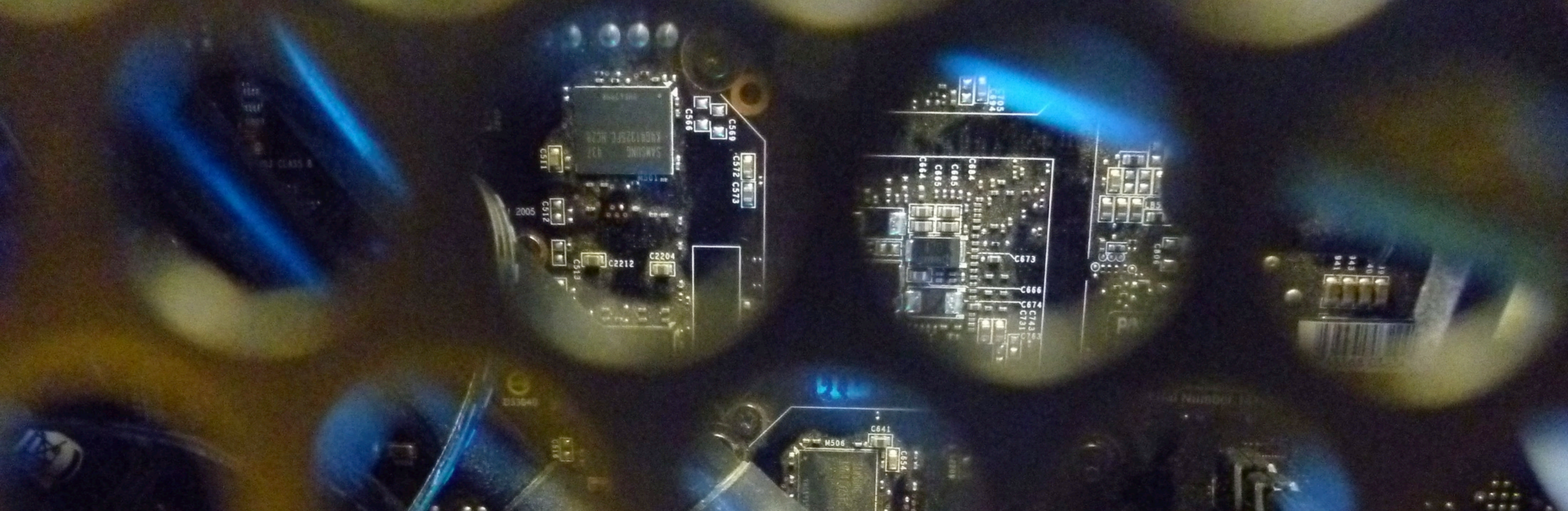
In 2014, CERN Researchers considered which of the platforms makes the most sense for distributed Worldwide LHC Computing Grid, processing data for LHC experiments in 170 computing centers in 40 countries (incl. UofT)



(< 400^3 CFD on Titan GPU)


EVGA GTX 480



In a small cluster, CPU, MIC and GPU are combined. We can simulate a Galaxy’s inner part star by star (~4 G stars), and/or its gas disk at high resolution (2 Gcell)

We can run 200   8-planet simulations fast (1G periods per day at 360+ steps per orbit),
or 7000 simulations 5 times slower

Large N-body systems by direct summation

20 arithmetic operations per one pairwise grav. interaction

leapfrog (Fortran90) **leapfrog (CUDA C)**

N = 10 K ... 1 M

CPU (i7-5820K 4GHz)	**MIC** (KNC)	**GPU** (gtx 980, Titan)
0.28 TFLOP sp 14 G interac/s	1.33 TFLOP sp 67 G interac/s	3.5 TFLOP sp (gtx980) 190 G interac/s
0.09 TFLOP dp 4.5 G interac/s	0.51 TFLOP dp 25 G interac/s	0.81 TFLOP dp (Titan) 40 G interac/s

!
! on MIC the calculation is **2.8 times slower** than on GPU (sp)
! **1.6 times slower** than on GPU (dp)
! CPU (6c.) is **9..13 times slower** than GPU
!

note: this is a rare fully compute-bound calculation!

Concurrent 8-body systems by 4th order symplectic code

n8b-aug14.3.f90 Same double precision program. Compiled with ifort

platform	CPU	MIC
compiler flag	-xhost	-mmic
number of N-body systems per processor	12	224
N [#threads per sys.]	8 [1]	8 [1]
exec. time per step	0.871 μs	4.58 μs
steps per orbit	360	360
exec. time of 1 orbit	0.313 ms	1.65 ms
exec. time (1G orbits)	3.63 days	19.1 days
system clock	4 GHz	1.1 GHz
throughput	13.8 M sys-step/s	49 M sys-step/s
# concurrent systems (SciPhi cluster UTSC)	192	10752

Practical capabilities of processor platforms for dynamical astro-calculations. Single (co)processors
CPU ~ E5 and i7 ser. (Intel), MIC = Knights Corner (Intel 2013),
GPU = Nvidia GTX970..1080 (sp) and Titan (dp) run:

1. Gravit. **N-body problem $\mathcal{O}(\sim N^2)$**. N ~10^6 real-time (~1 fps)
   GPU > MIC ~ CPU (mostly comput. limited, > TFLOP)

2. Disks of **particles** (stars; asteroids, planetesimals, meteoroids and dust).
   ~ 10^9/s, ~10^8 in RAM, (~10 fps)
   GPU ~ MIC **>** CPU (bandwidth-limited to 150 GB/s)

3. Pure CFD = **fluids,** cells: ~10^8/s, ~10^8 in RAM
   GPU ~ MIC **~** CPU (mostly bandwidth limited) , (~1 fps)

GPU – some have decent double precision, most don't.
Somewhat difficult to program and optimize, compared to x86 platforms. Very fast on direct summation.

Collisionless gigaparticle disks can be simulated with 4th order symplectic algorithm

Algorithm: 4^{th} Order Symplectic

Forest and Ruth (1990)

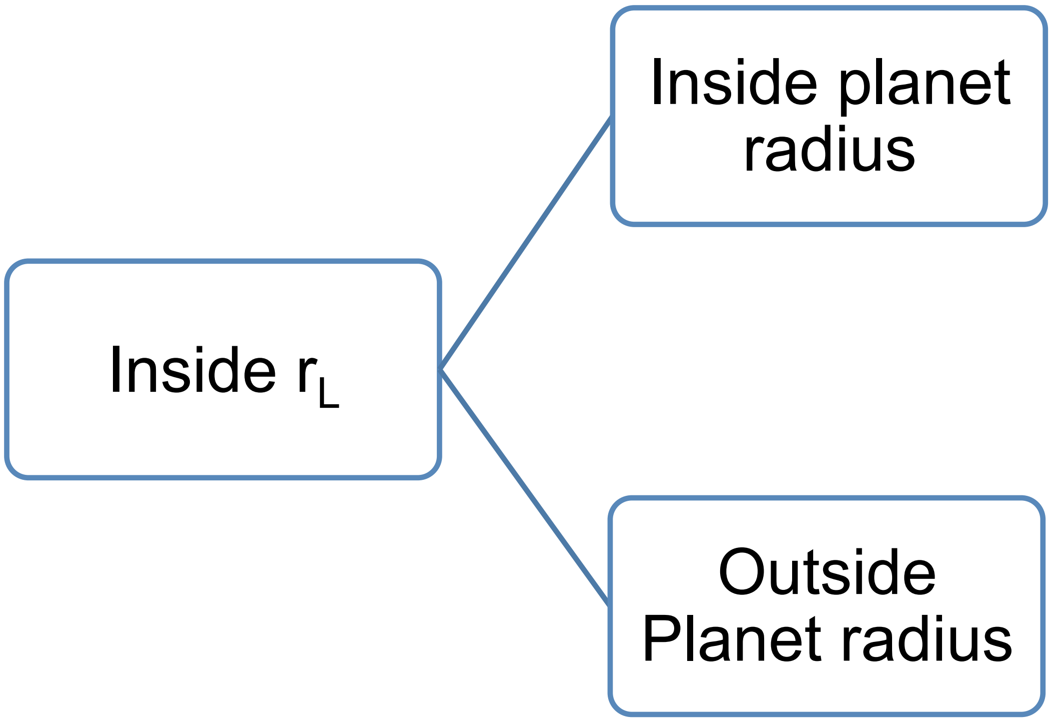
1. Push position: $x_2 = x_1 + c_1*v$
2. Calculate **force** (at updated position)
3. Kick velocity: $v_2 = v_1 + d_1*a$

4. Push position: $x_2 = x_1 + c_2*v$
5. Calculate **force** (at updated position)
6. Kick velocity: $v_2 = v_1 + d_2*a$

7. Push position: $x_2 = x_1 + c_3*v$
8. Calculate **force** (at updated position)
9. Kick velocity: $v_2 = v_1 + d_3*a$

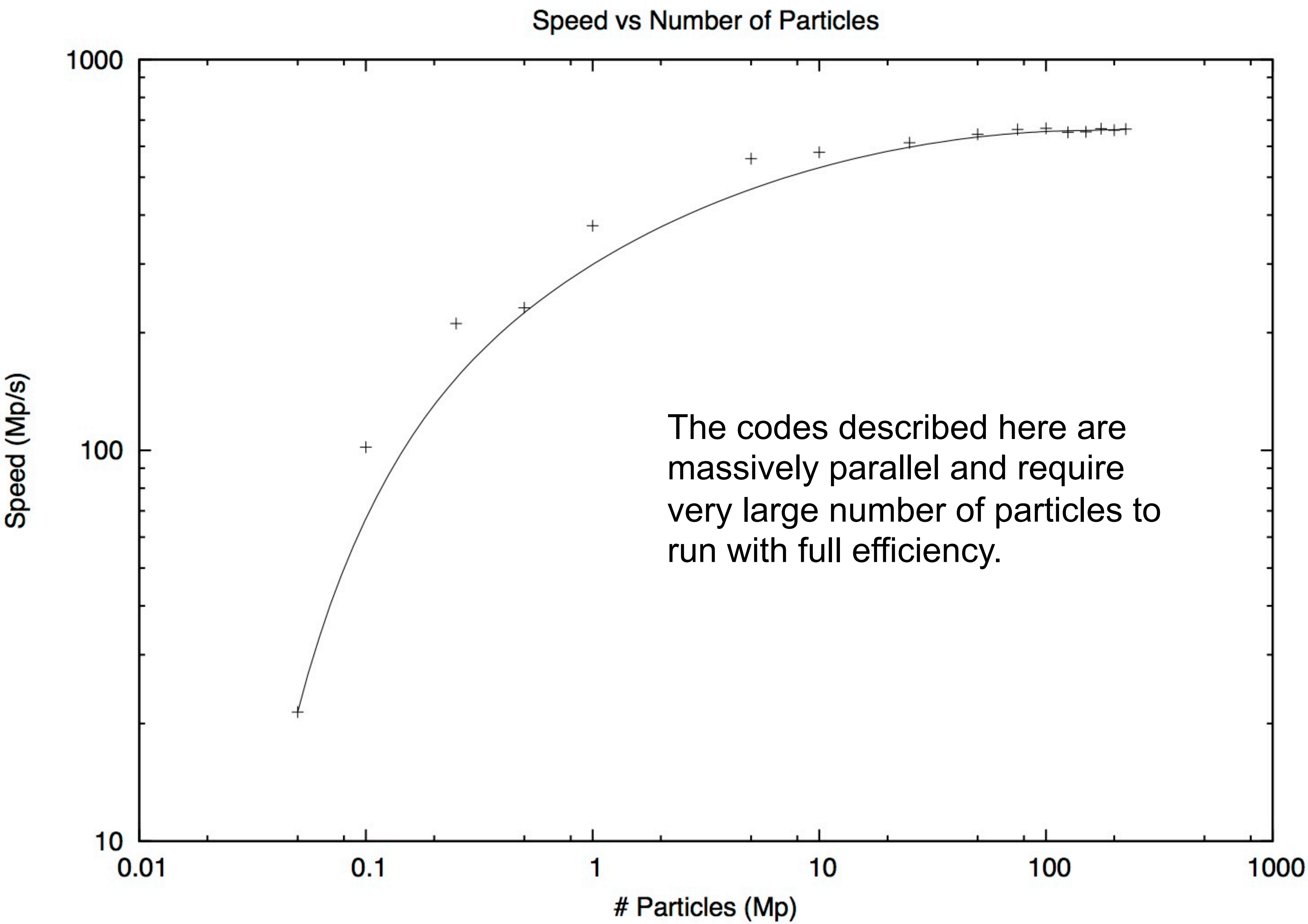
10. Push position: $x_2 = x_1 + c_4*v$

Collision with Binary and Variable dt

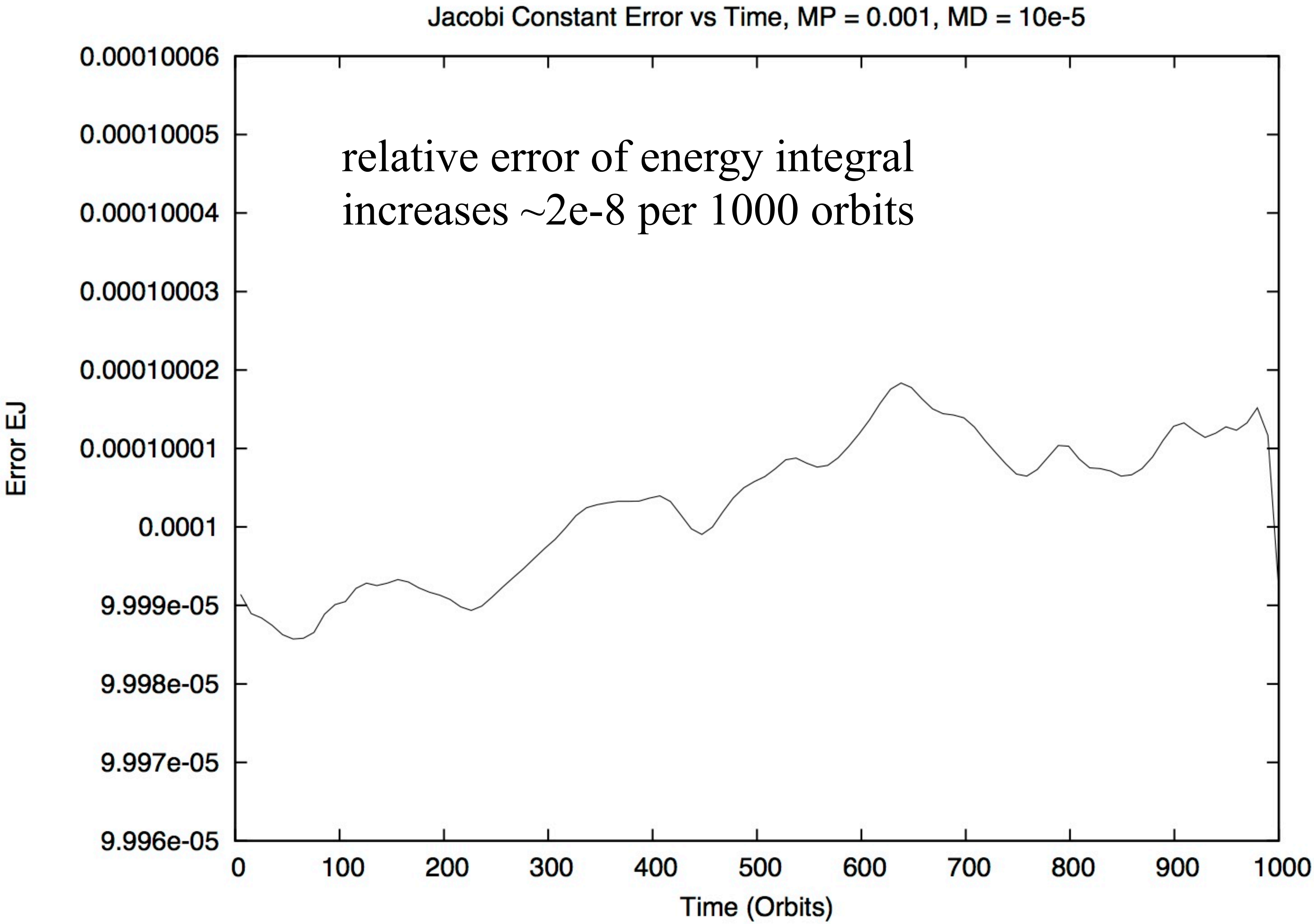


- Store particle and set to large r in main array
- Remove from array
- Transfer momentum and cm position
- Increase mass and spin

- Store particle and set to larger *r* in main array
- Perform same scheme but with variable dt
- Range 1e-8 – dt_1(0.004)



The codes described here are massively parallel and require very large number of particles to run with full efficiency.


Jacobi Constant Error vs Time, MP = 0.001, MD = 10e-5
relative error of energy integral
increases ~2e-8 per 1000 orbits
Error EJ
Time (Orbits)
0.00010006
0.00010005
0.00010004
0.00010003
0.00010002
0.00010001
0.0001
9.999e-05
9.998e-05
9.997e-05
9.996e-05
0
100
200
300
400
500
600
700
800
900
1000

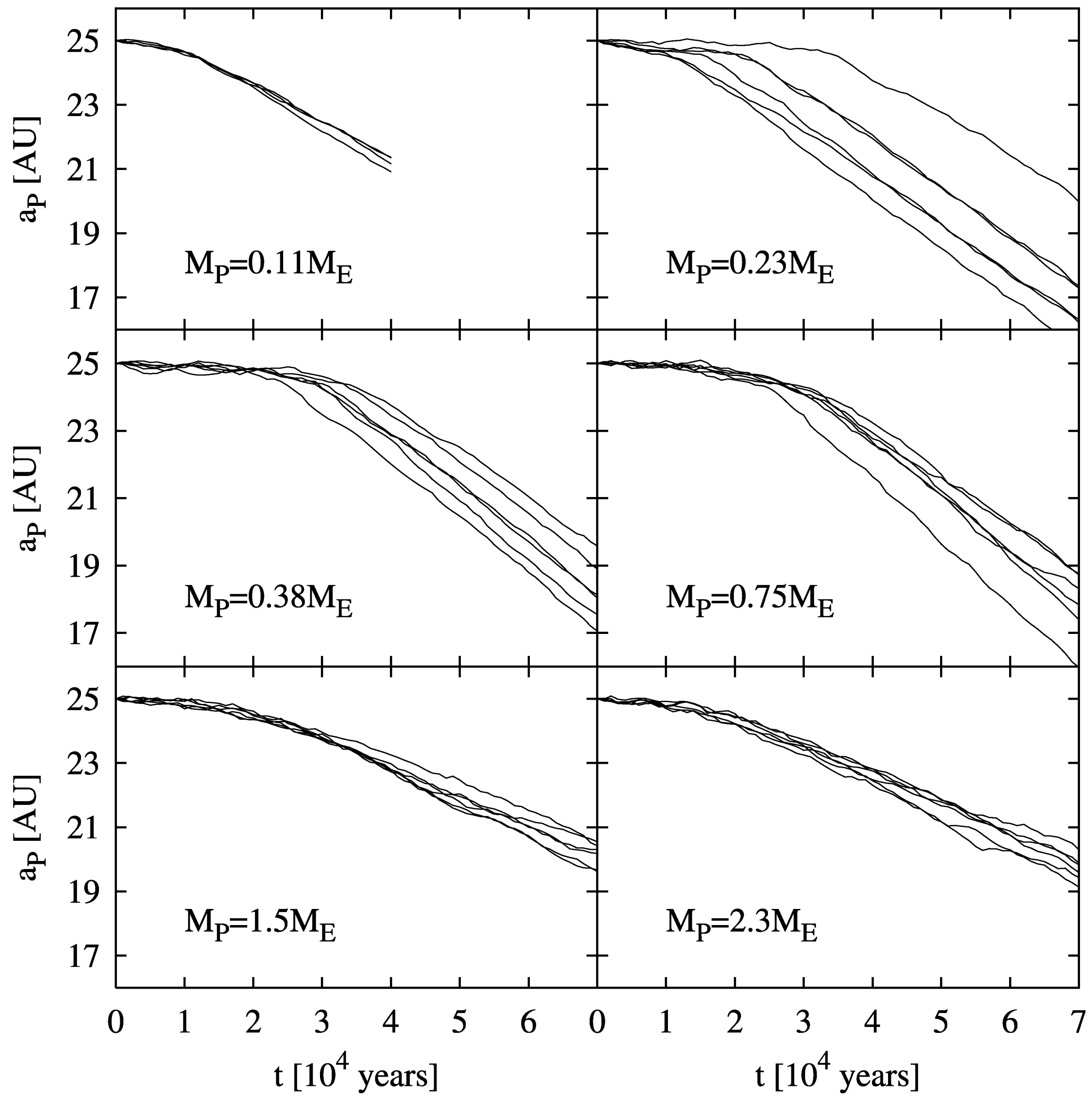
We study type III migration in Disks

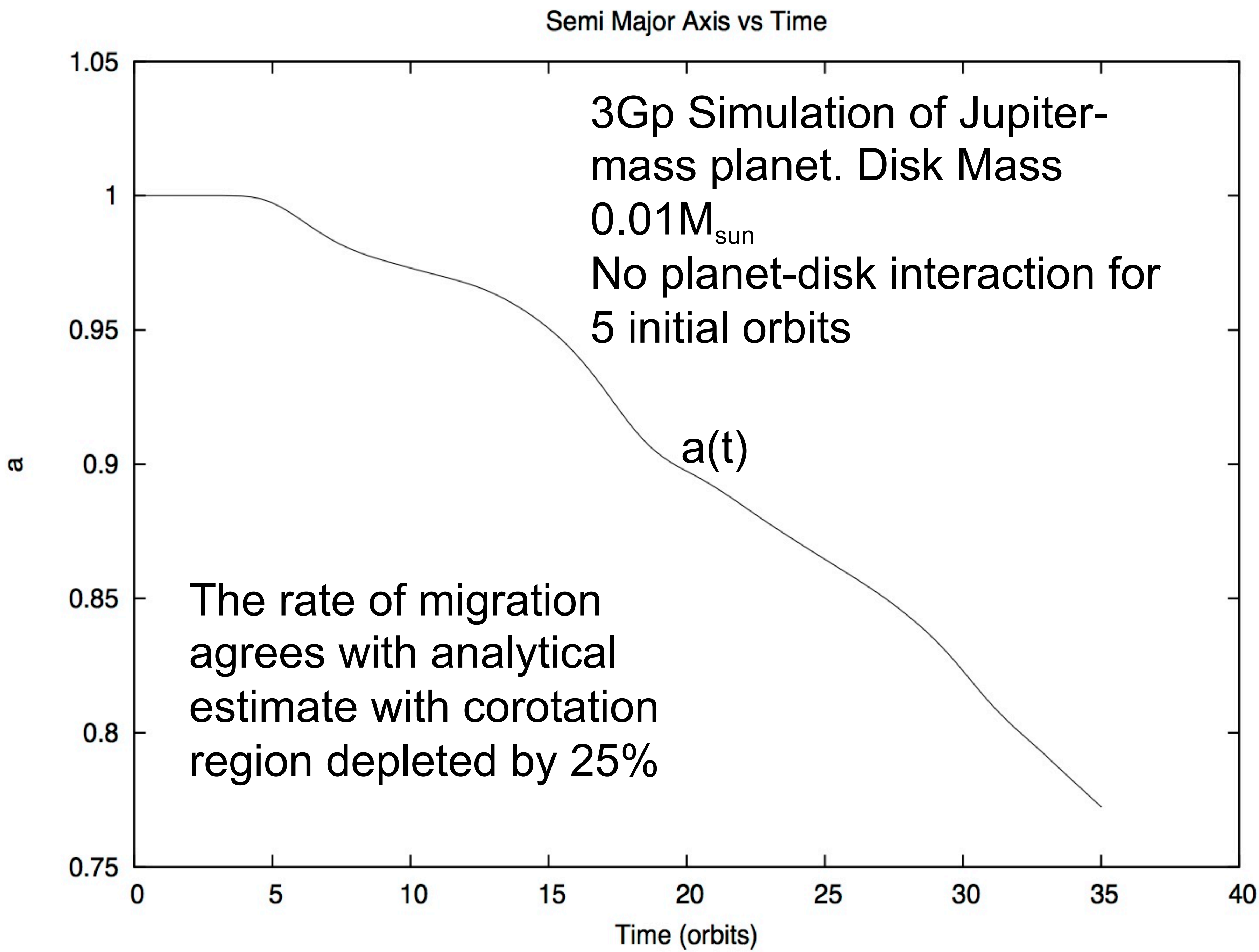
- Very rapid migration in *gas* disks: 40-50 orbits timescale for Jupiter-mass planet in a solar nebula disk

  (Papaloizou et al. in Protostars and Planets V, 2005)
- Rate does not depend on mass of planet
- Criterion compares disk (in CR = corotation region) and planet masses:
  – $M_p < M_{deficit}$. Difficult to satisfy by planetesimals...

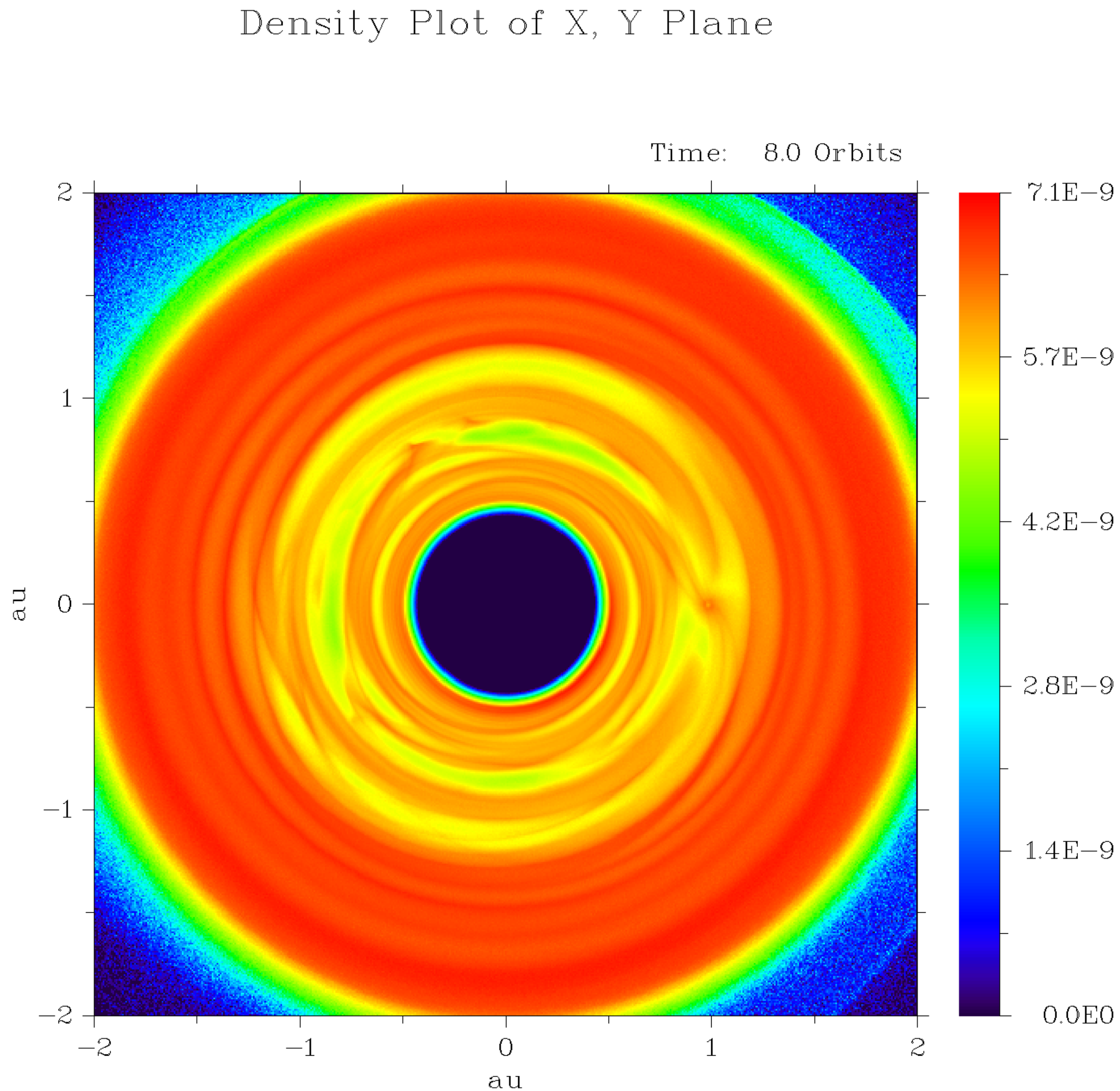
Previous results: Kirsh et al. 2009 identified the fast migration and offered an explanation [without noticing a connection with type III migration, e.g. as reviewed by Papaloizou et al. 2006, PP V]

Much slower migration by mean-motion resonant scattering (w/similarly v. massive disks) proposed by Murray et al (1998).



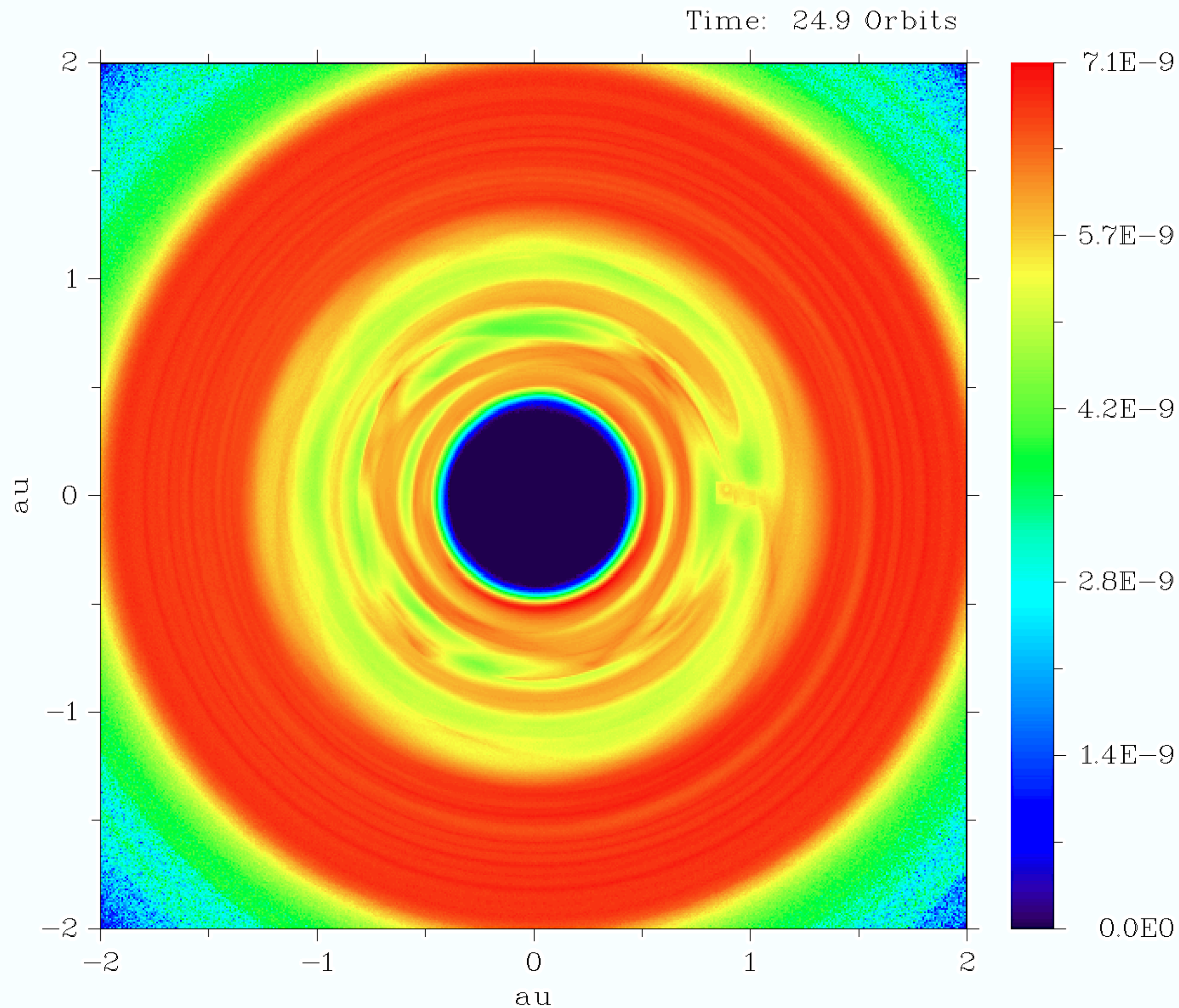


Semi Major Axis vs Time
3Gp Simulation of Jupiter-mass planet. Disk Mass 0.01M_sun
No planet-disk interaction for 5 initial orbits
a(t)
The rate of migration agrees with analytical estimate with corotation region depleted by 25%
a
1.05
1
0.95
0.9
0.85
0.8
0.75
0
5
10
15
20
25
30
35
40
Time (orbits)



Density Plot of X, Y Plane
Time: 8.0 Orbits
2
1
0
−1
−2
au
−2
−1
0
1
2
au
7.1E−9
5.7E−9
4.2E−9
2.8E−9
1.4E−9
0.0E0

Density Plot of X, Y Plane



Conclusions of Fergus Horrobin's summer research in 2017

For large-scale particle integrations in non-collisional disks, codes can run v. fast on MIC cluster (Xeon Phi)

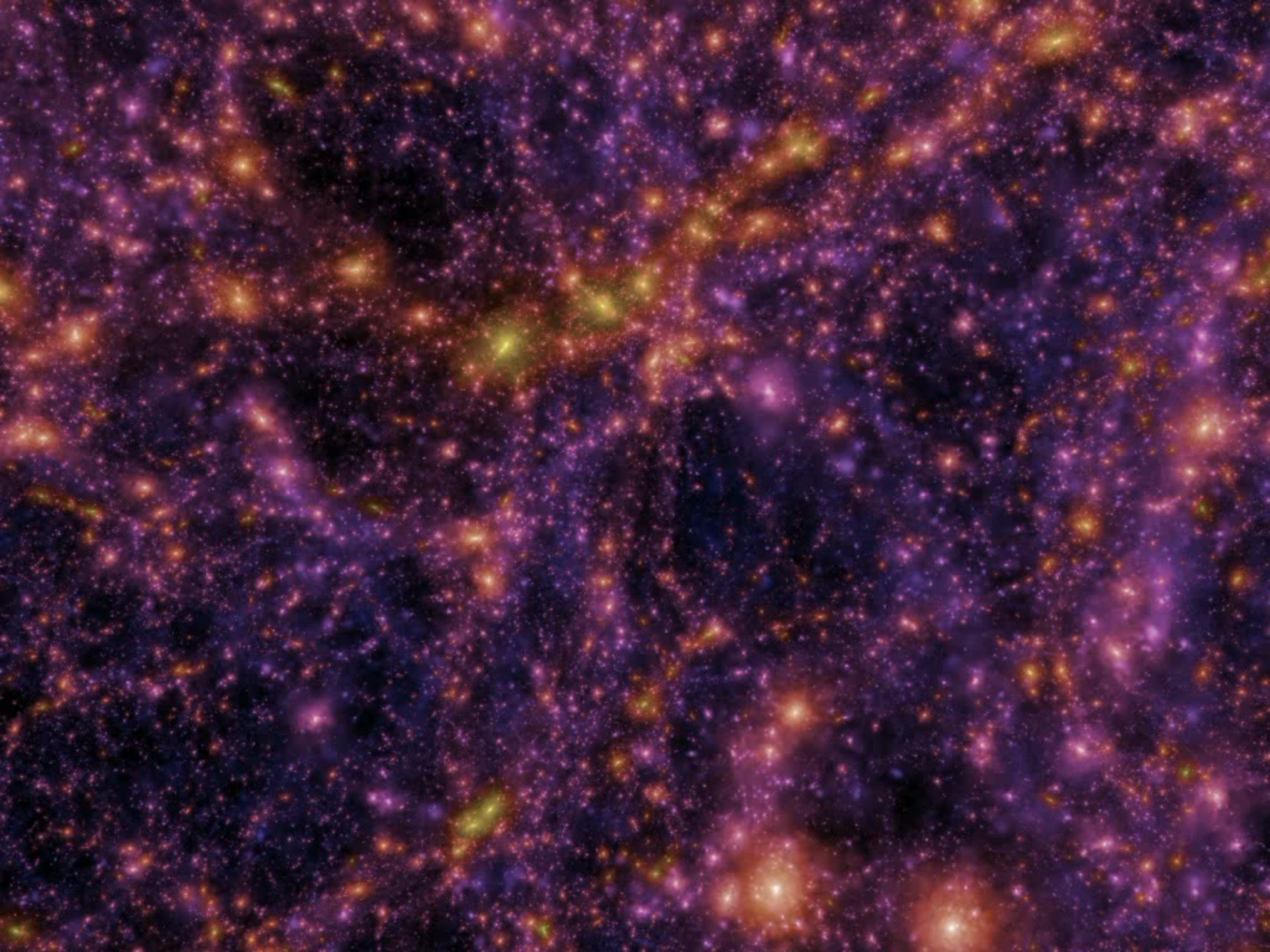
- 3+ billion particles (150M per MIC), timestep ~0.2 s
- Hybrid parallelization method combining OpenMP and MPI seems best for this type of platform
- We've implemented 4^{th} order symplectic integrator.
- Though deeper analysis must be made, we see similarities between gas and particle disks in the context of rapid migrations

N-body simulations of the Universe

- https://www.youtube.com/watch?v=YjUICiYICYE
- Millenium – 10+G particles Gadget code,
- kept the main supercomp at MPI Inst of Astronomy in Garching, Germany, busy for a month in 2004
- (700 MPc)^3

- https://www.youtube.com/watch?v=32qqEzBG9OI

(350Mpc)^3, 5e4 galaxies, 12G particles, 8k CPUs
- Millenium XXL



N-body simulations of the Universe

- https://www.youtube.com/watch?v=YjUICiYICYE
- https://www.youtube.com/watch?v=32qqEzBG9OI

  $(350\text{Mpc})^3$ = $(1 \text{ billion ly})^3$, 50K galaxies, 12G particles
- Simulation name: Bolshoi
- Run on Pleiades cluster (supercomputer) at NASA Ames Research Center in Mountainview, California.

N-body simulations of the Universe

12G particles create 50000 galaxies,   gas: AMR grid

8k CPUs used for Bolshoi-Planck simulation

Pleiades has theor. peak performance 7.3 PFLOPS

- Cosmological simulations: Millenium, Bolshoi are examples of:

PDEs. Partial differential equations:

- Wave equation in 2 dimensions
  Z_tt = c^2 (Z_xx + Z_yy)   PDE, c = speed of the wave
  _tt = second time deriv.,   _xx = second deriv after x, etc.
- Pond or swimming pool surface
  - pond1.py,   pond3.py,   pond4.py
  - pond4-1obj.py,
  - Young's double slit experiment:
    pond4-2slit3.py, pond4-2slit4.py
- CFL condition

- research on astro-CFD at UTSC

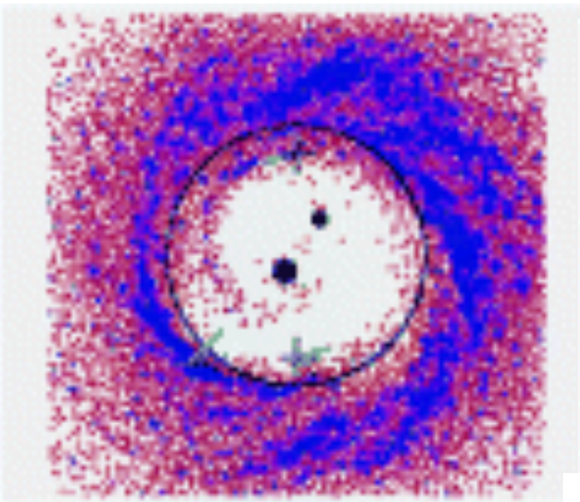
1. Astrophysical problems for CPU and GPU calc's:

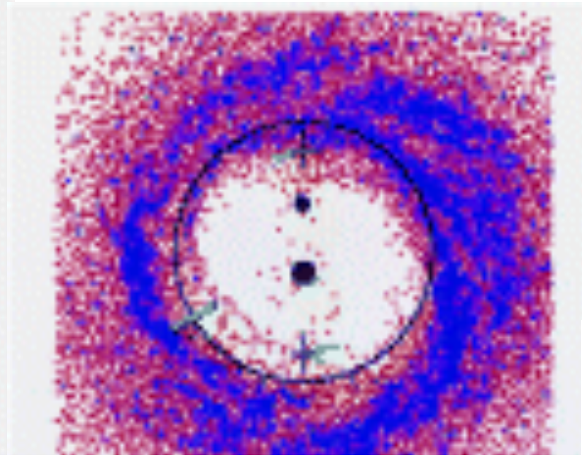
- Disk-planet interaction and migration
- Disks with structure: IRI (irradiation instability in particle and gas disks)
- Flow of gas around Super-Earth (5 M_E)

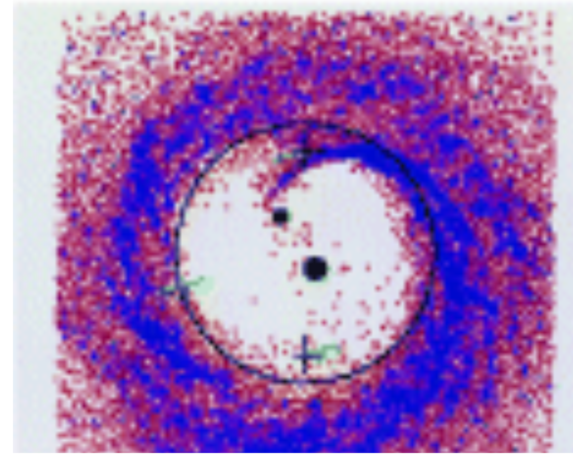
2. Massively parallel numerics on mini-supercomputers:

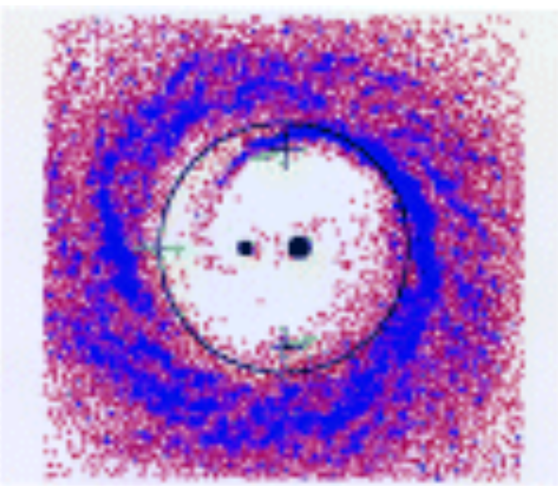
- Comparison of HPC platforms: CPU, GPU, and MIC (Φ)
- UTSC clusters

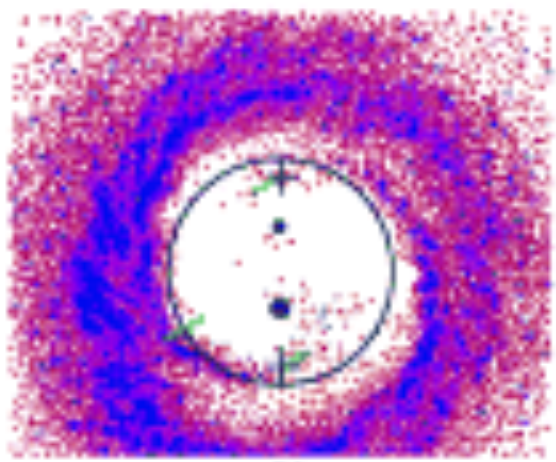
Binary-disk interaction

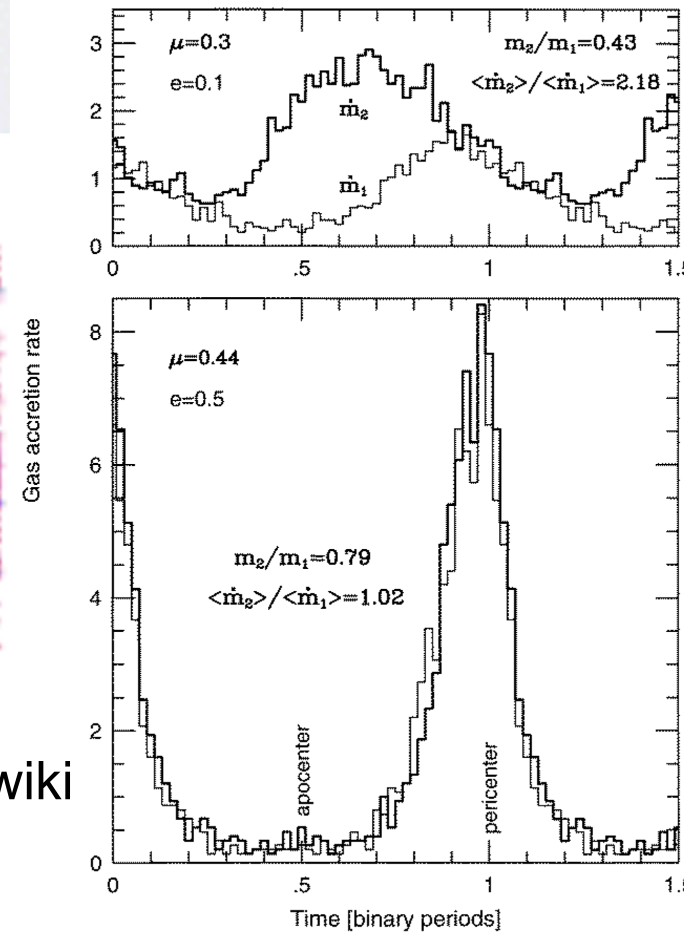










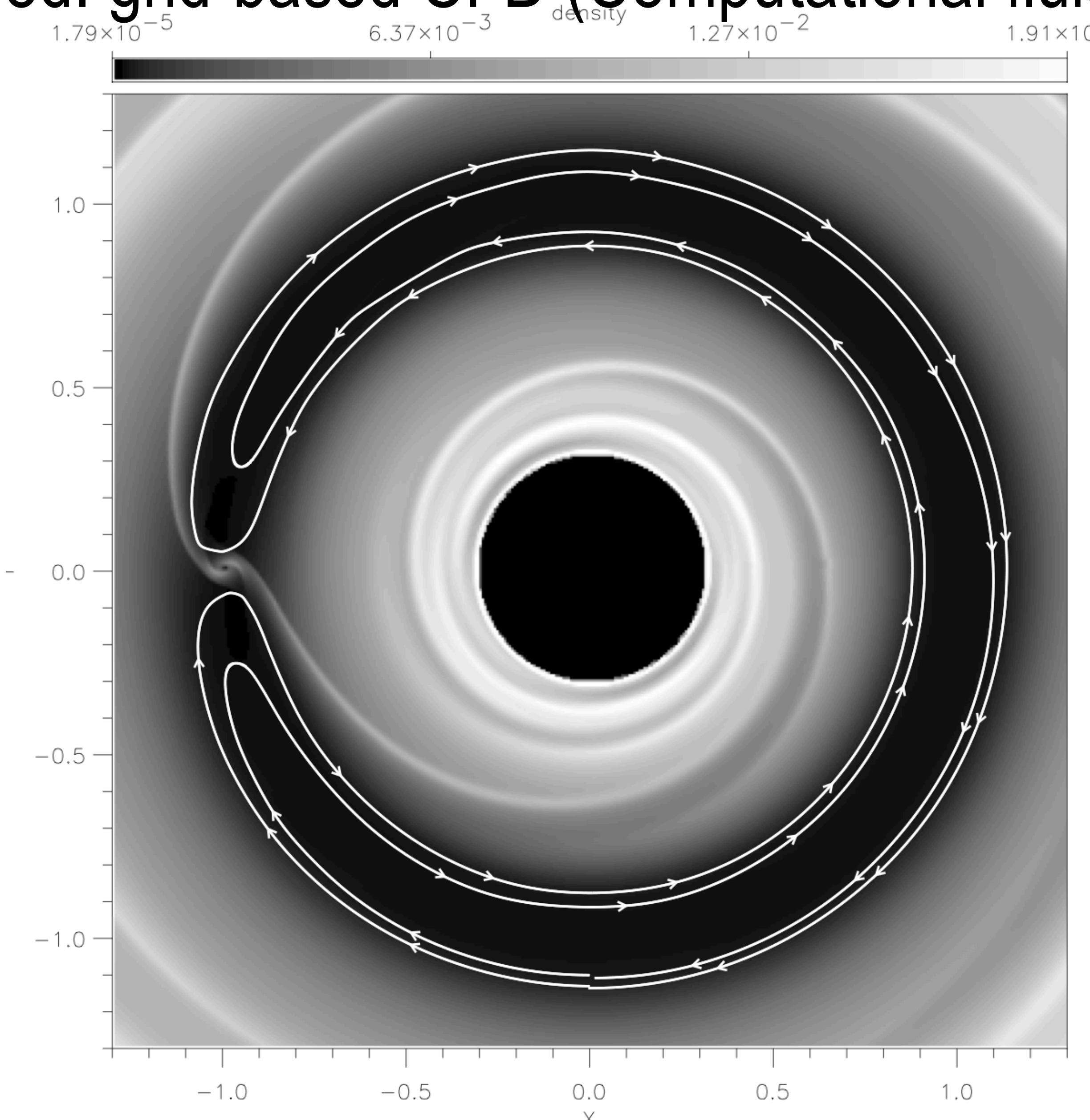


SPH = smoothed hydrodynamics method: see wiki

Artymowicz and Lubow (1996)

Binary-disk interaction

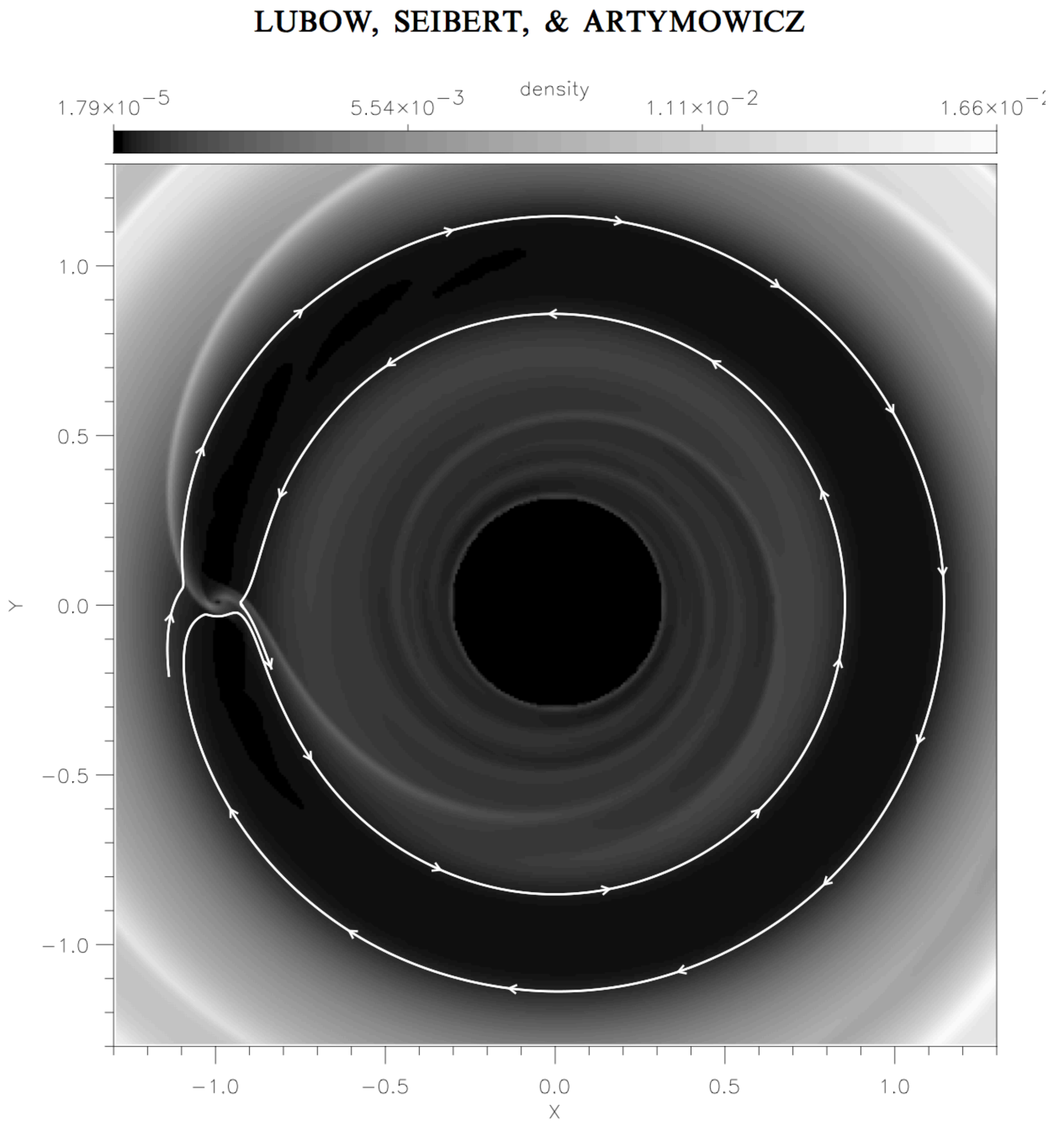
method: grid-based CFD (Computational fluid dynamics



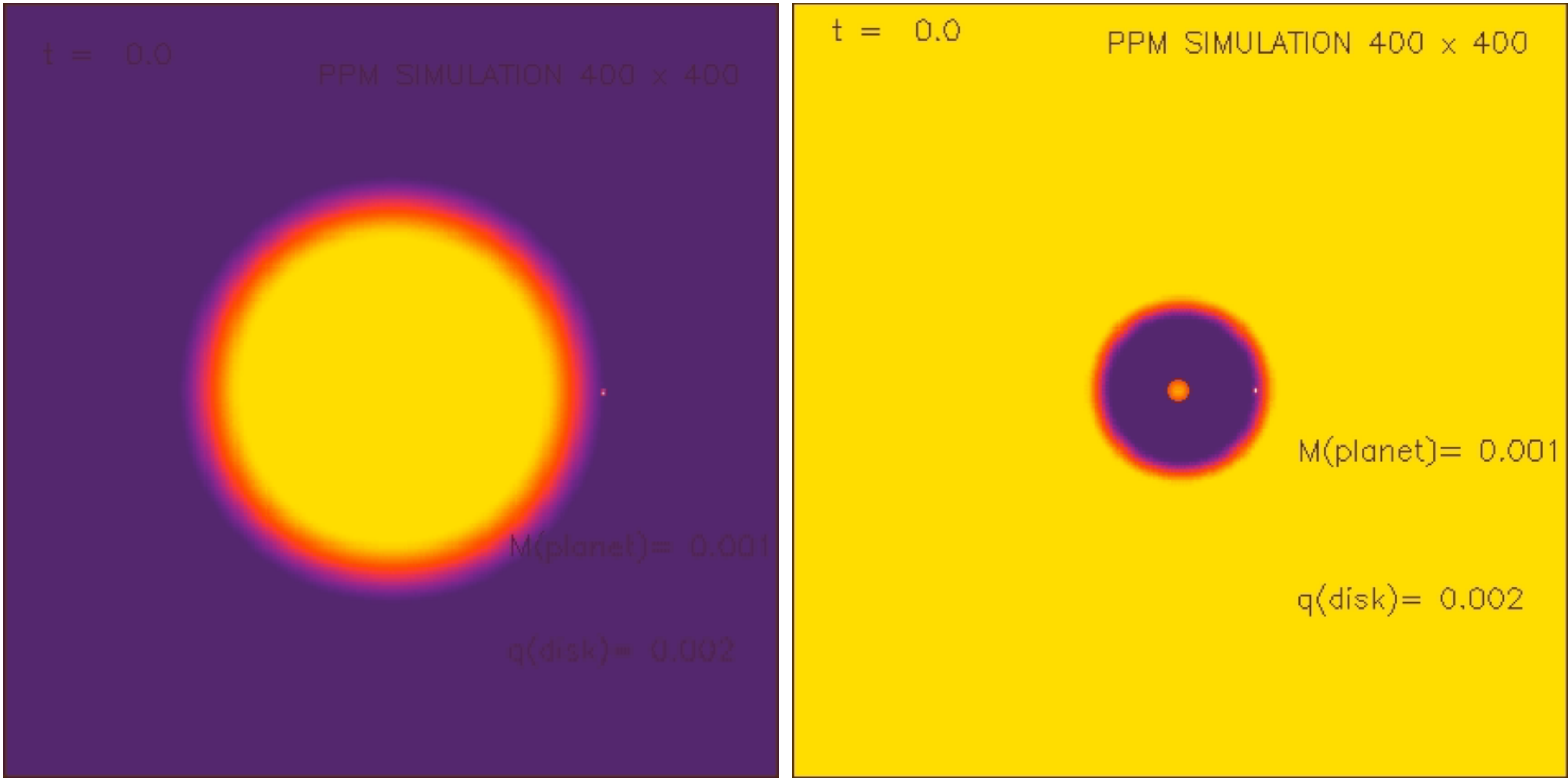
CPU 2-d

2^{nd} order ZEUS hydro

notice mass flow through gap



One-sided disk (**inner/outer disk only**). The rapid inward migration is OPPOSITE to the expectation based on shepherding (Lindblad resonances).

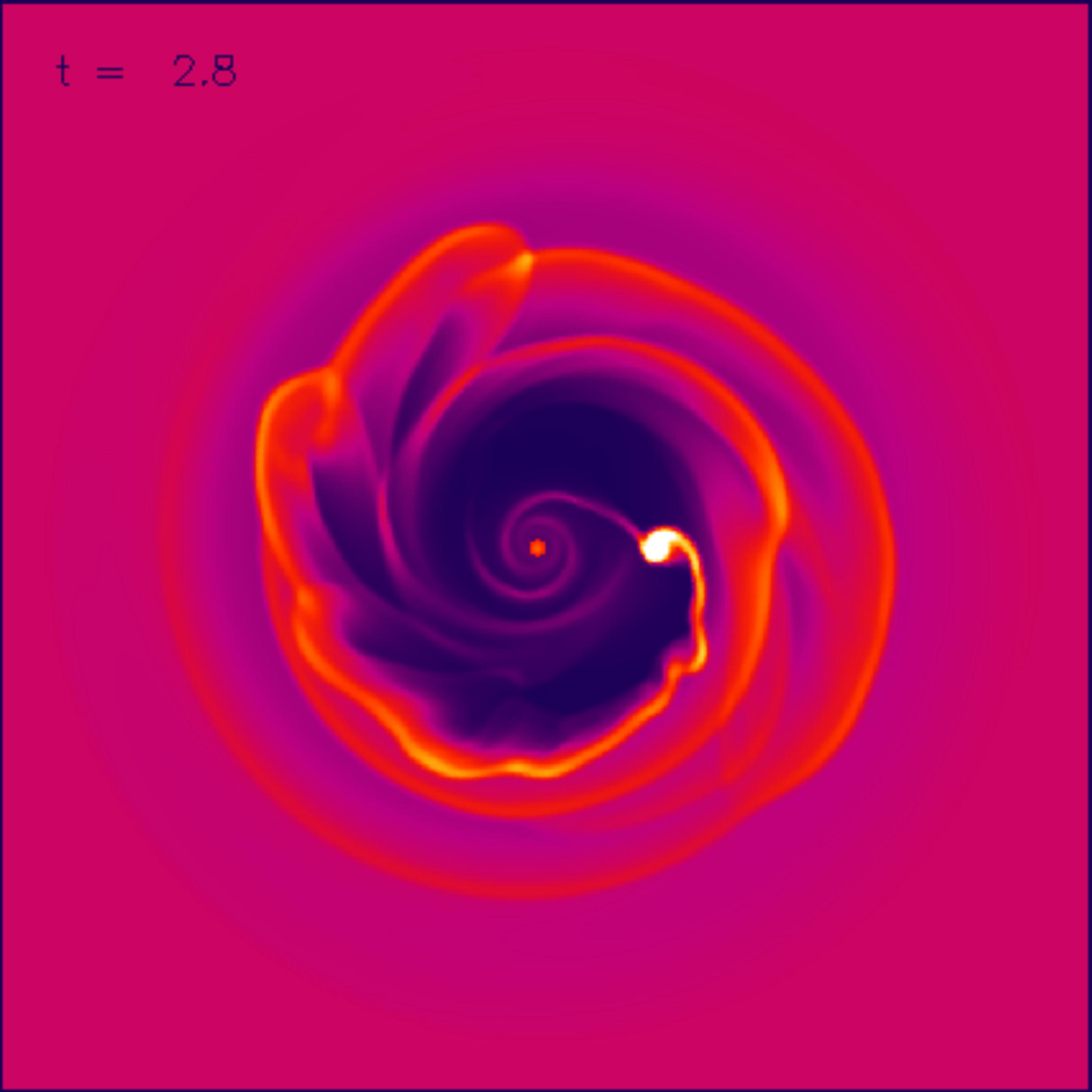


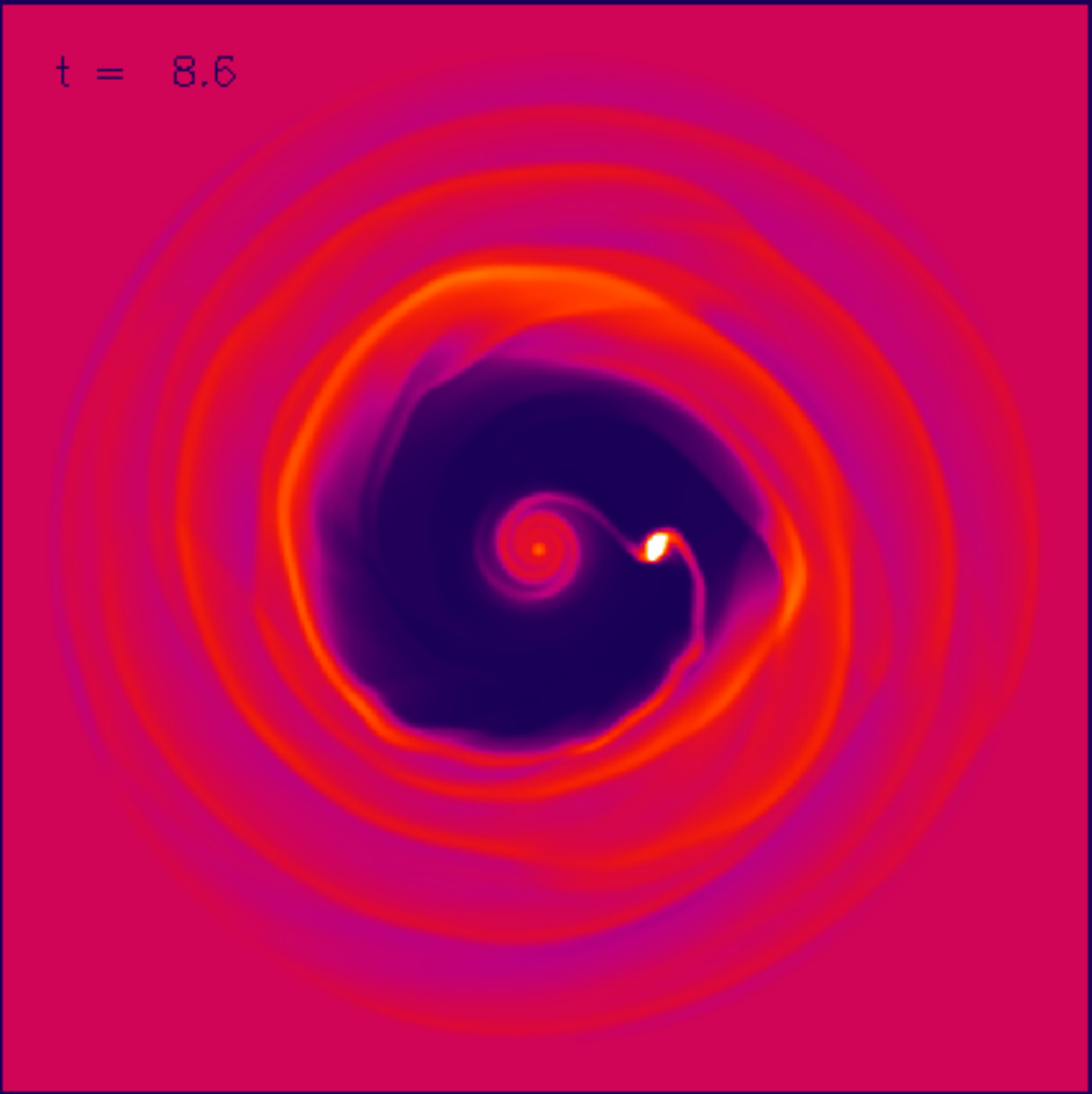
Like in the well-known problem of “sinking satellites” (small satellite galaxies merging with the target disk galaxies),

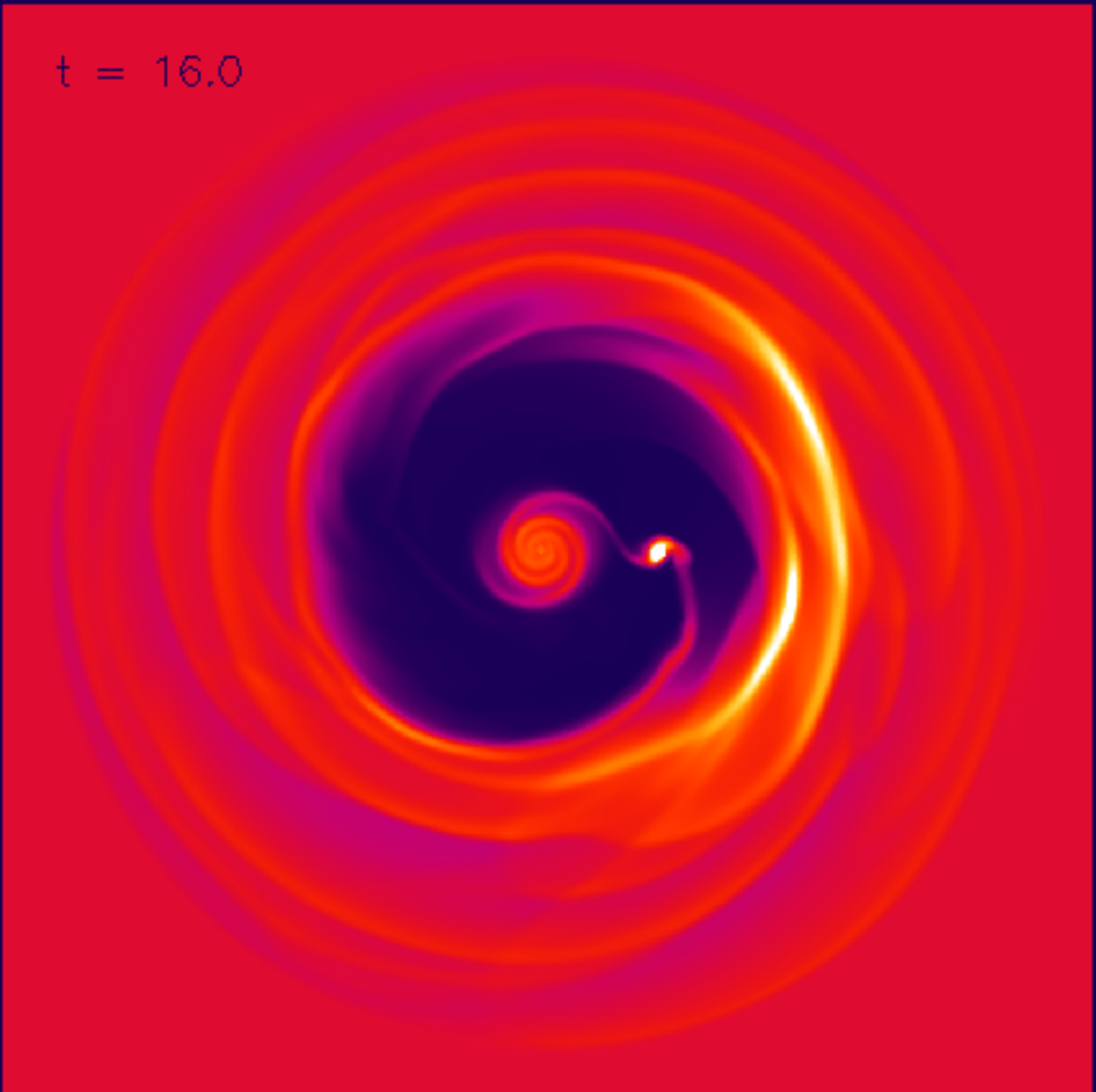
Corotational torques cause rapid inward sinking.

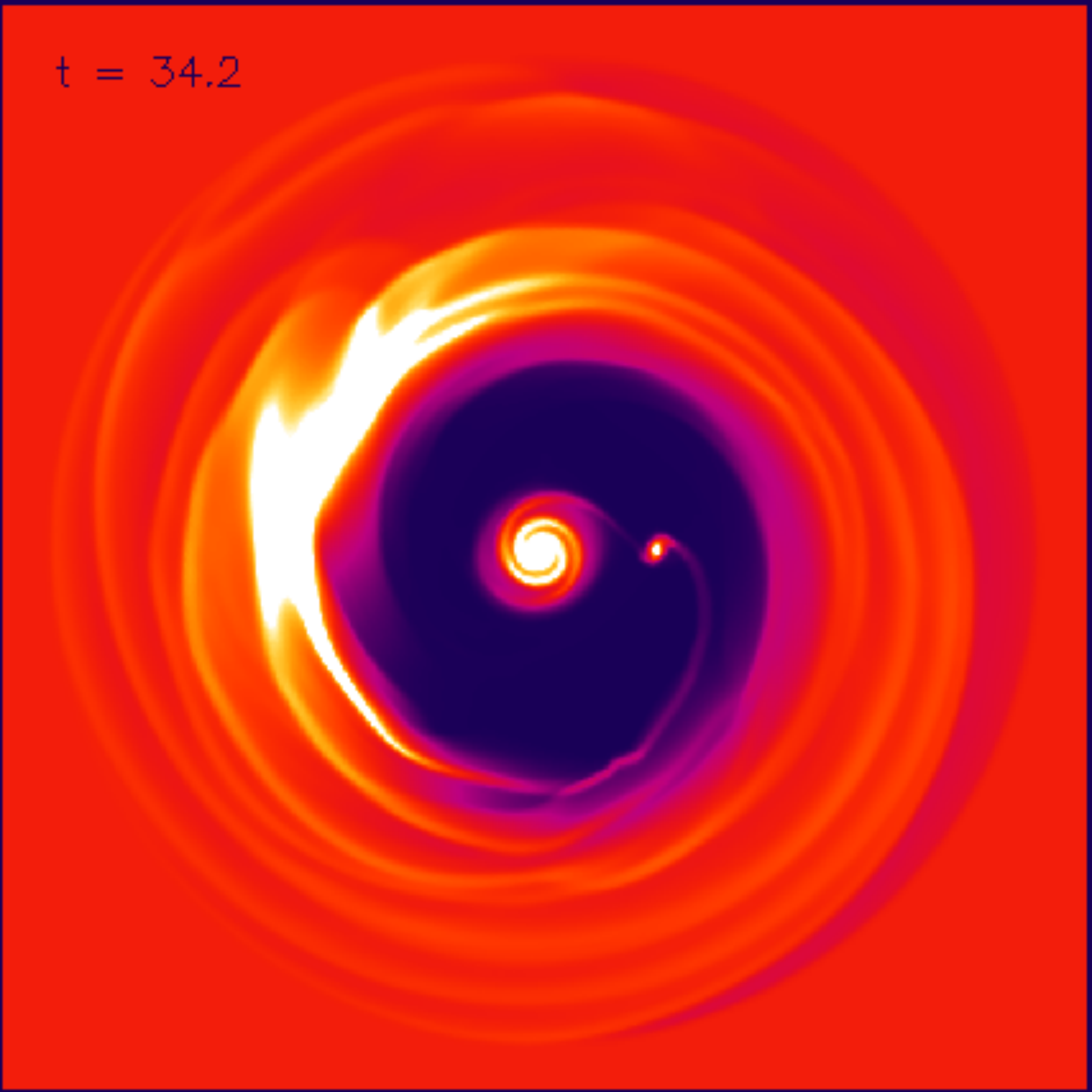
Another timely issue crying out for analytical work is the appearan
of unexpected asymmetr
and blobs:

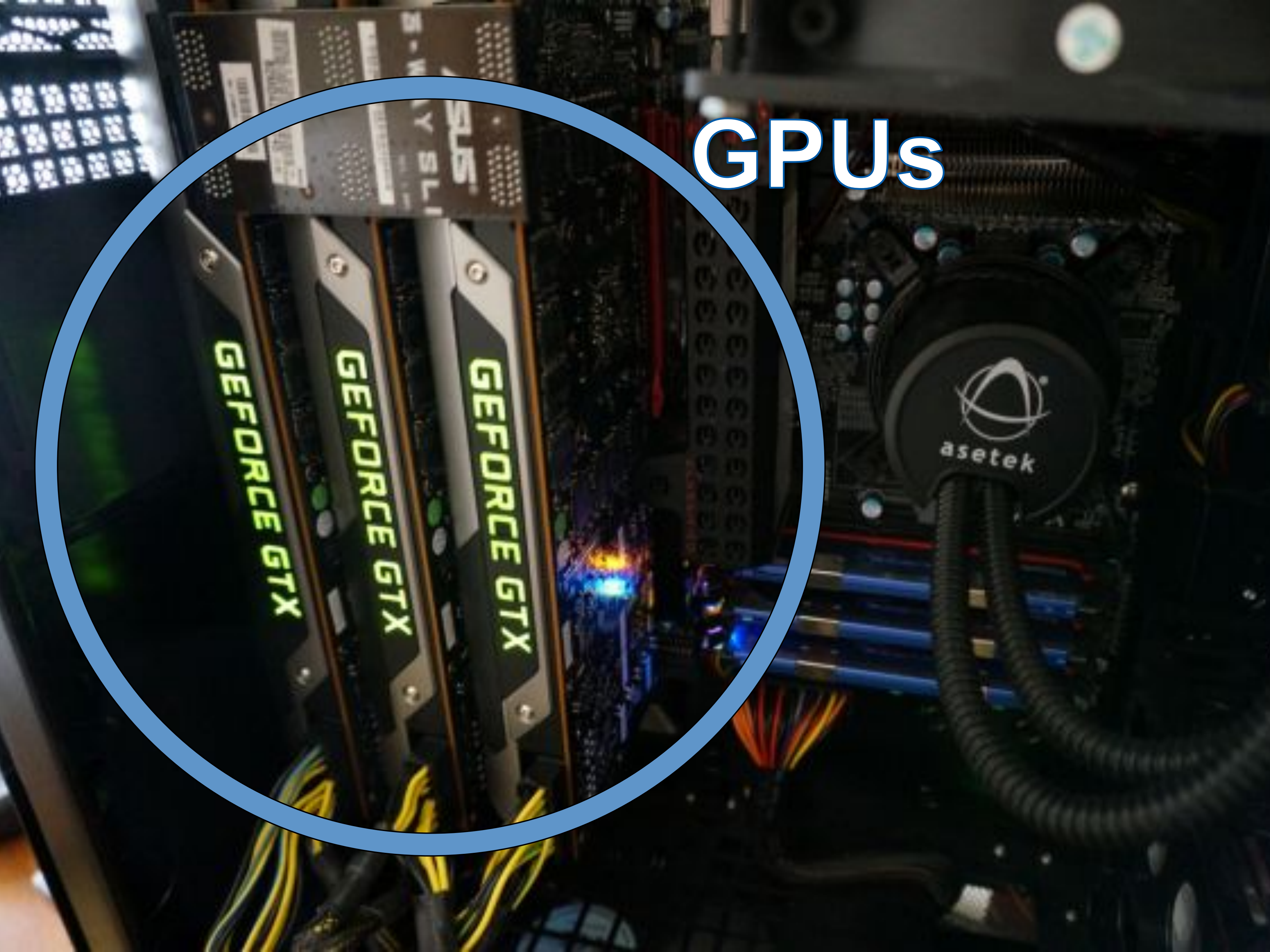
- ❖ are they due streamers?
- ⦿ are they edge modes


t = 2.8


t = 8.6


t = 16.0


t = 34.2


GPUs
ASUS
3-WAY SLI
GEFORCE GTX
GEFORCE GTX
GEFORCE GTX
asetek

The 3-D flow around a small, embedded planet

J. Fung, P. Artymowicz and Y. Wu (ApJ, Nov 2015)

Code: PenGUIn.

CUDA C++. Processes up to ~20 Mcells/s (dp), ~40 Mcell/s (sp)

for comparison, Xeon Phi can run the same size problems at ~30 Mcell/s (sp)

and a modern 6-core CPU does ~28 Mcell/s.

These codes are bandwidth-bound. GPU > MIC ~ CPU

2-D

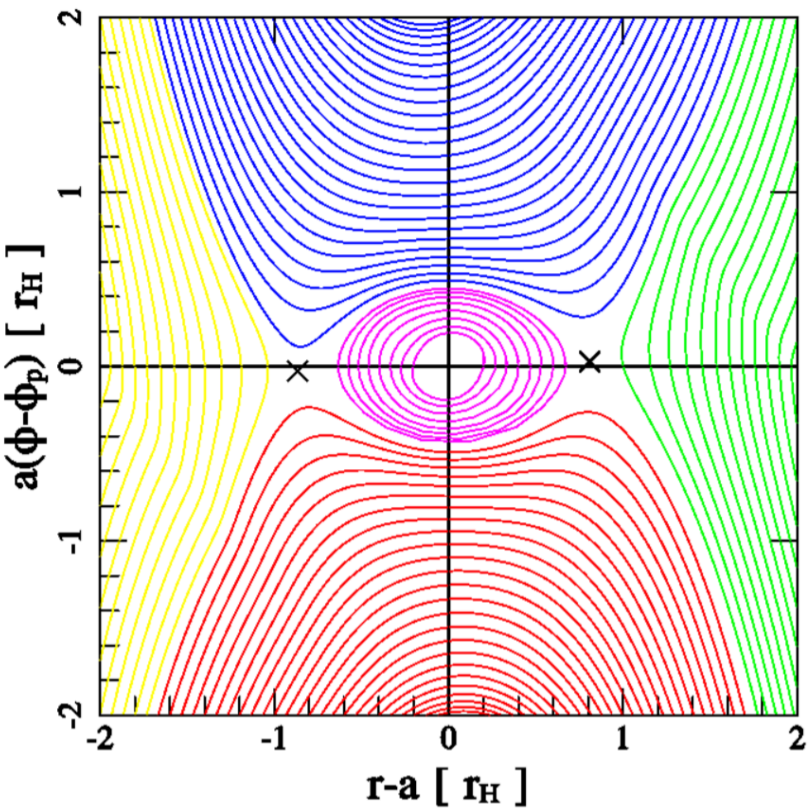


FIG. 1.— Streamlines around a planet in 2D, plotted in the corotating frame

3-D

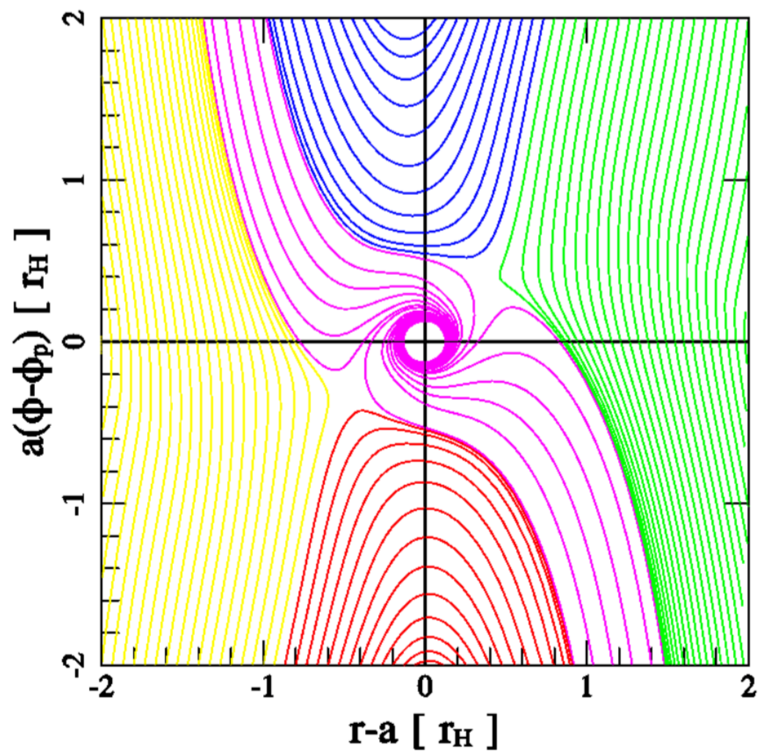
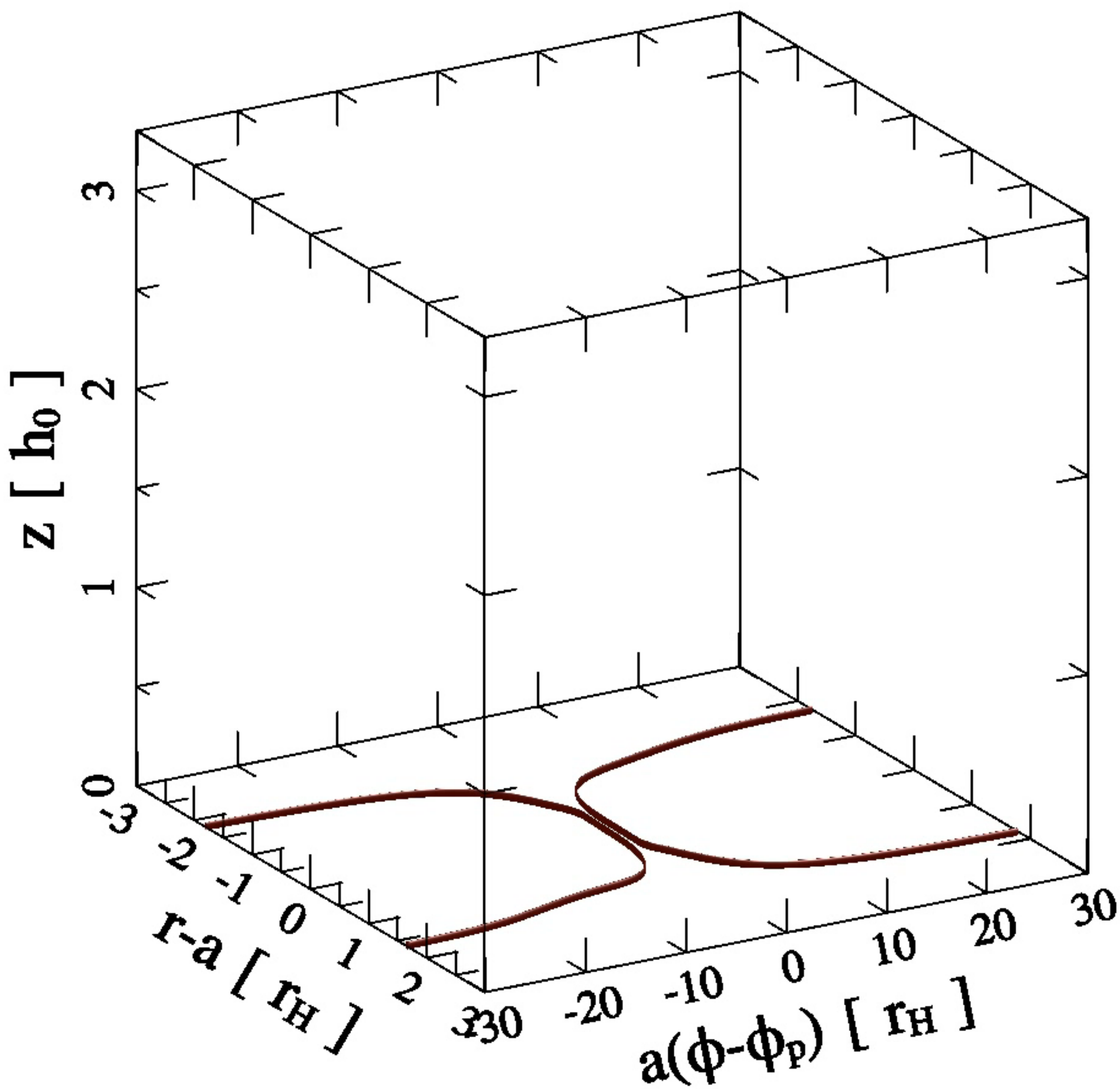
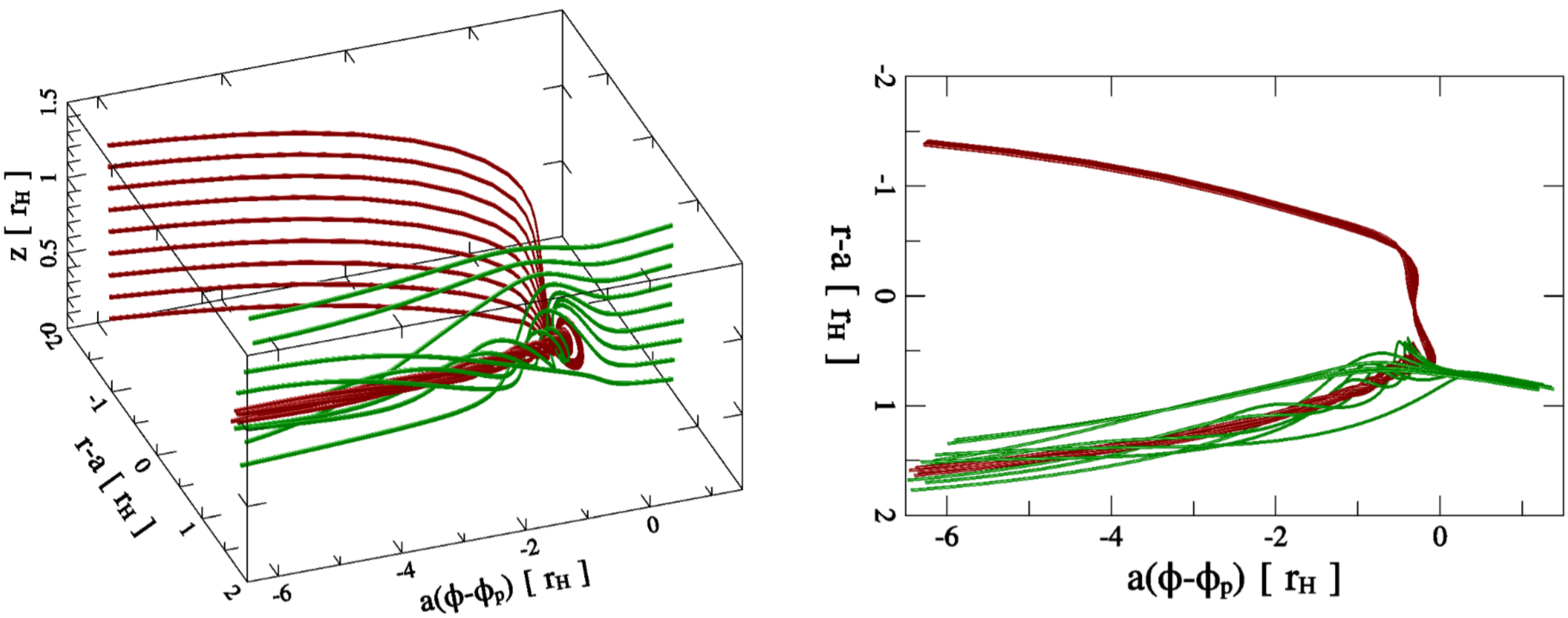
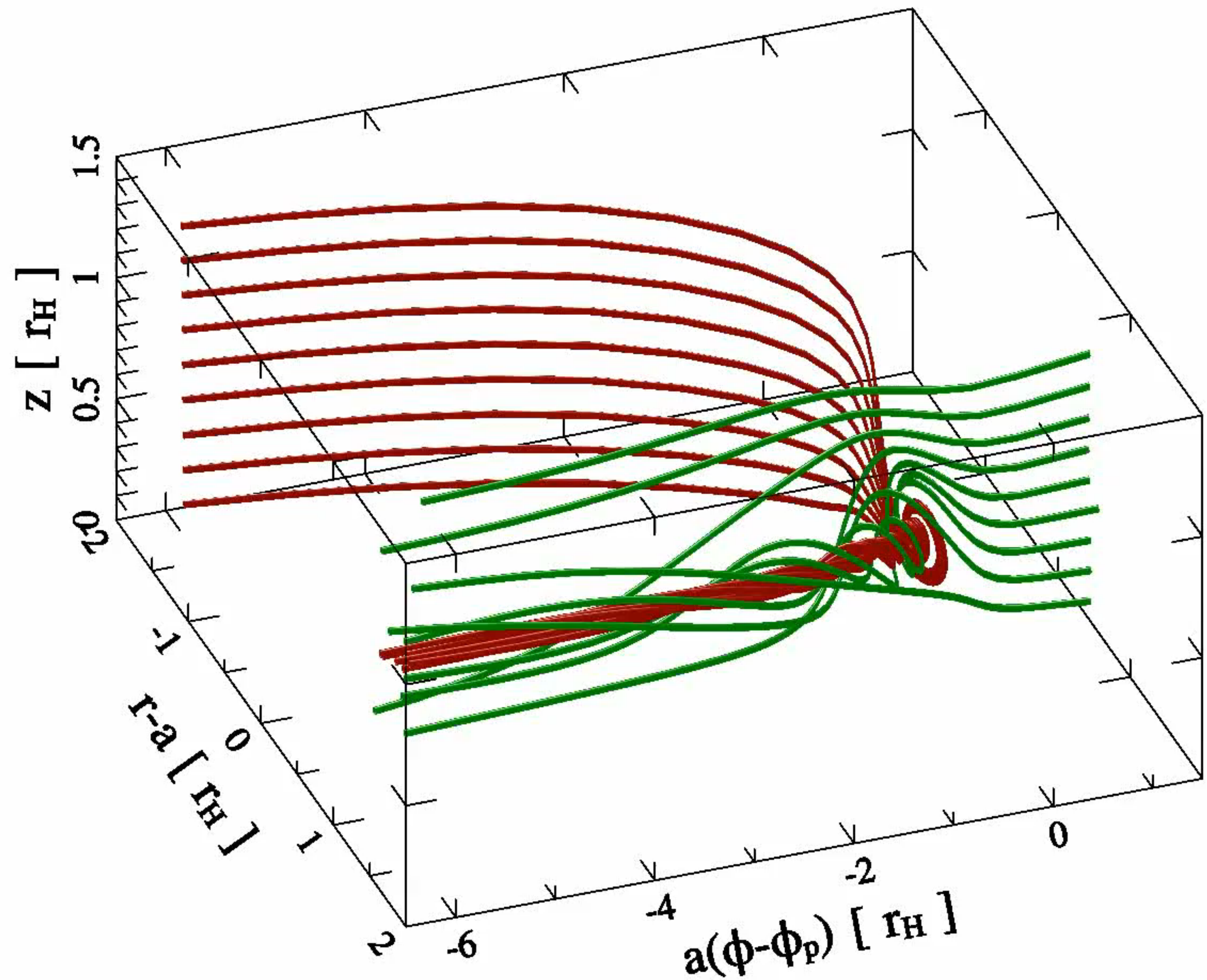


FIG. 8.— Streamlines in the disk midplane. Compare with Figure 1 for differences between 2D and 3D flow. Yellow, red, green, and blue streamlines are assigned in the same manner as Figure 1. Unlike Figure 1, magenta lines are outflows away from the planet, pulled down from initially higher altitudes. They reach as close as $1.5r_s$ from the planet and are unbound.

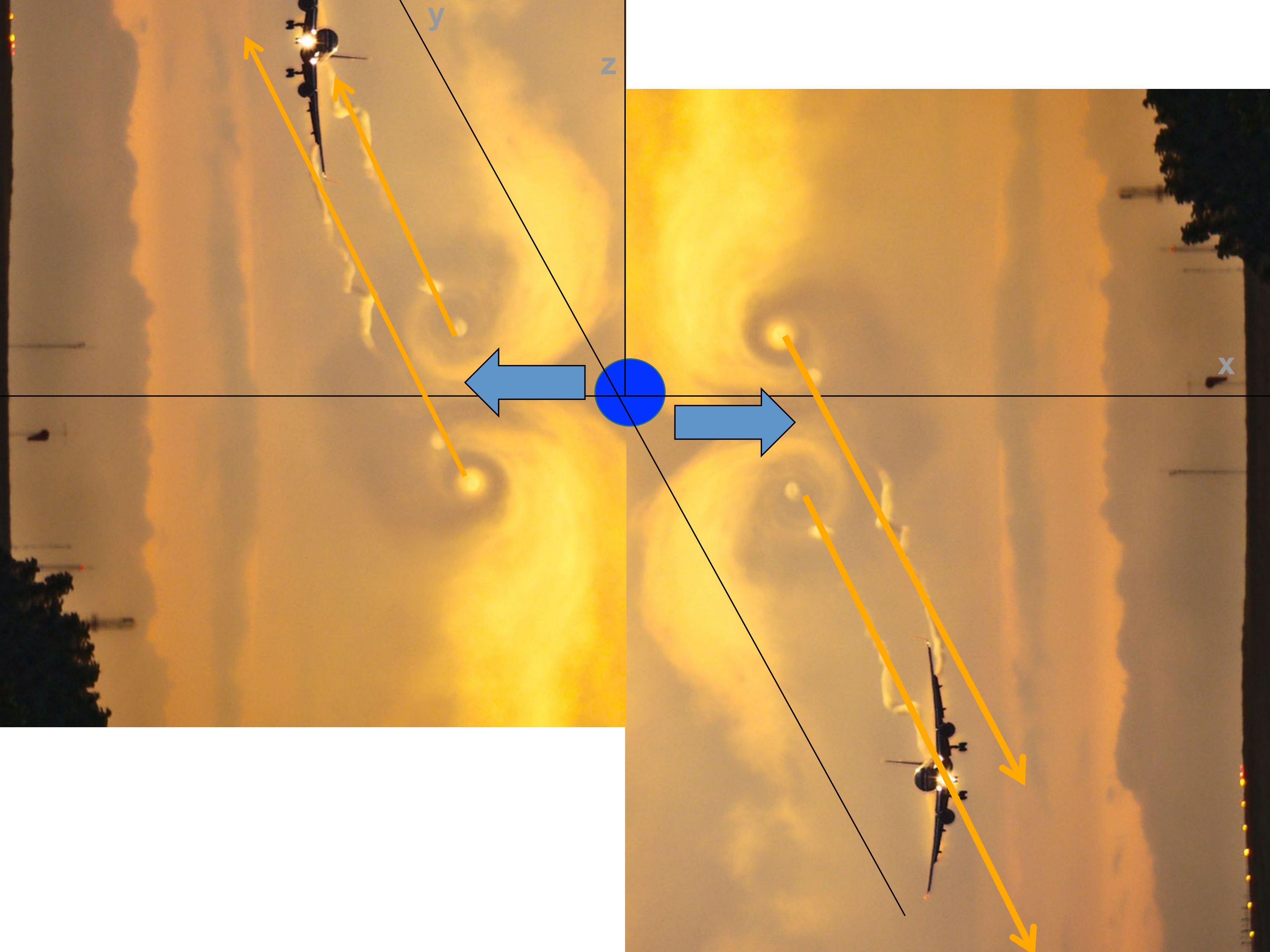


z [h_0]
0
1
2
3
r-a [r_H]
-3
-2
-1
0
1
2
3
a(ϕ-ϕ_p) [r_H]
-30
-20
-10
0
10
20
30


z [r_H]
1.5
1
0.5
0
r-a [r_H]
-2
-1
0
1
2
-6
-4
-2
0
a(ϕ-ϕ$_p$) [r_H]



z [rH]
1.5
1
0.5
0
r-a [rH]
-2
-1
0
1
2
a(φ-φp) [rH]
-6
-4
-2
0


y
z
x

RESULTS in 3D

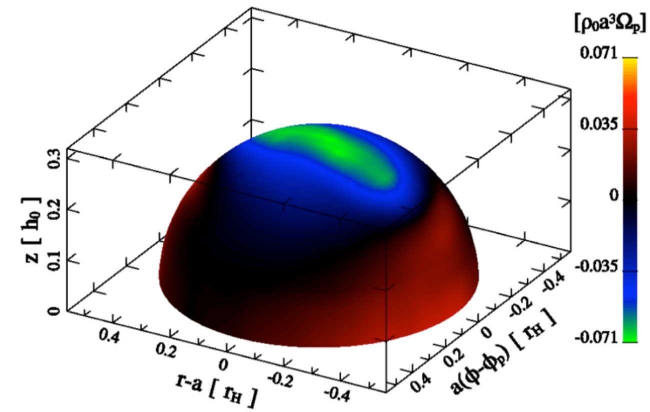


FIG. 9.— Mass flux across the surface of a sphere centered on the planet. The sphere has a radius of $0.5r_B$. Blue and green indicate influx; red and yellow are outflux. The speed of the downward flow is about $0.7c_s$ in this plot, while the two radial outward flows in the midplane (one not visible from this viewing angle) each has a speed of $\sim 0.2c_s$, as is explained in Appendix A. Match this figure with Figure 8 for a more complete view of the flow topology near the midplane.

New 3D phenomena, absent in 2D flows, including new columnar topology

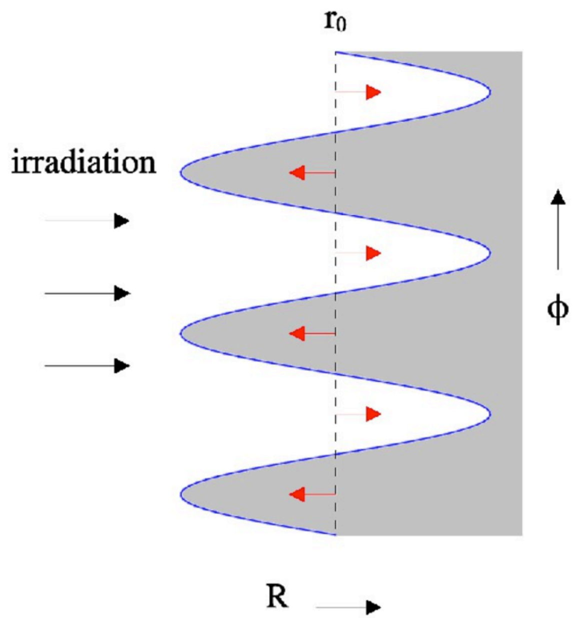
vorticity generation mechanism around a small planet, have a potential to resolve the long-standing problems in planet formation theory:

migration and cooling/contraction of the growing planet, occasional transmutation into a giant gaseous planet.

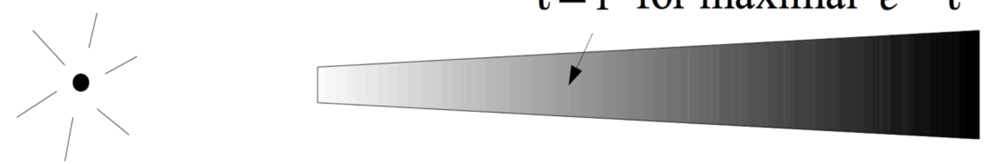
DUST/RADIATION PRESSURE-RELATED INSTABILITIES
including the IRI = IrRadiation Instability

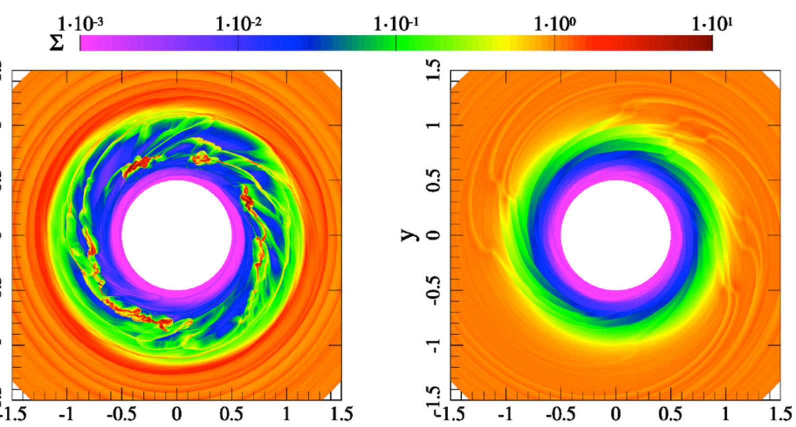
Jeffrey Fung (UC Berkeley)
used workstations at UofT with 3 GPUs
for parallel computations

IRI in 2D



IRI in 2D





Criterion for instability: $\beta e^{-\tau} \tau \dfrac{d \ln[r \Re]}{d \ln r} > 1$

$$\Re \equiv \frac{\Sigma \Omega_k \beta e^{-\tau}}{\kappa^2}$$

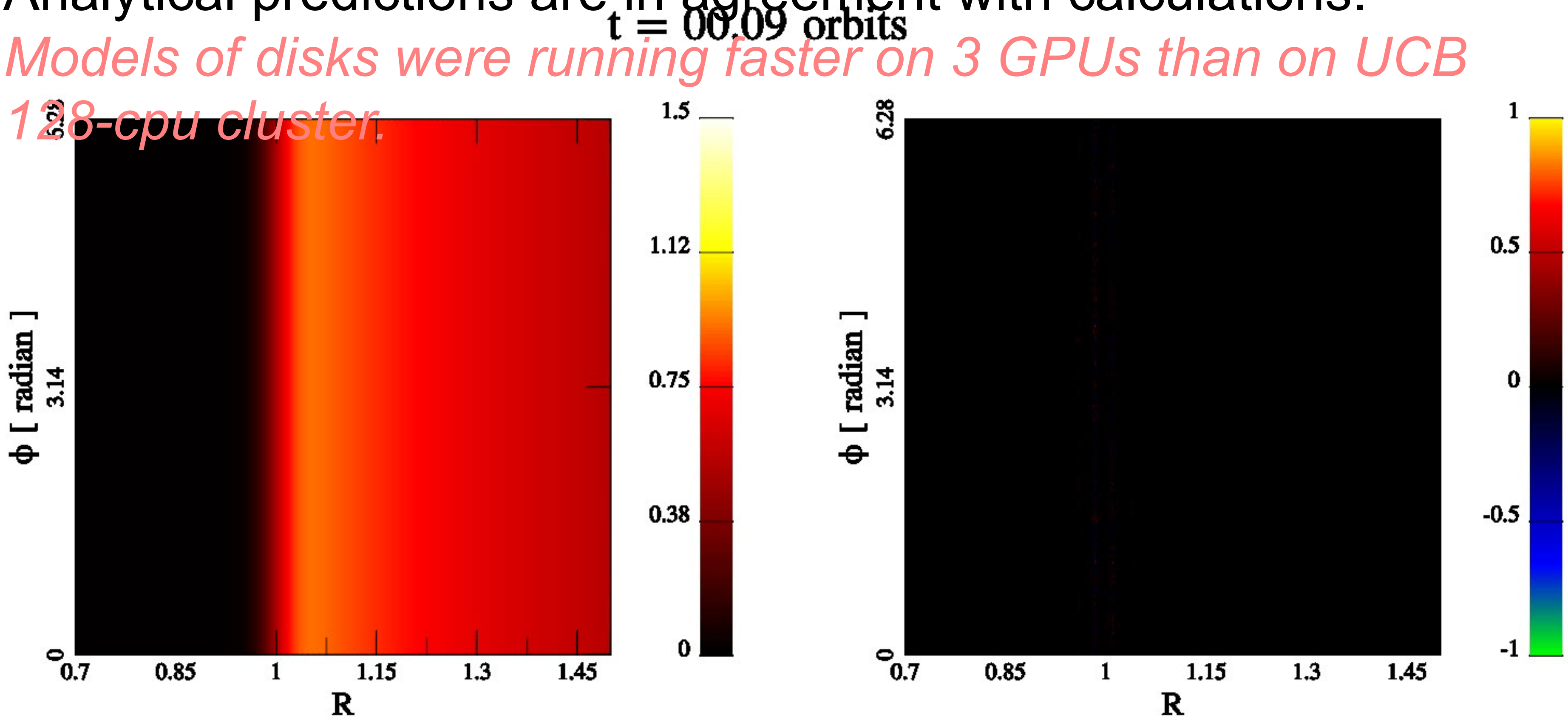
See Fung & Artymowicz (2014) for details

GAS DISK HYDRODYNAMICAL SIMULATION (PPM method, 2-D)

R.h.s. shows a background-removed picture of density of growing modes.

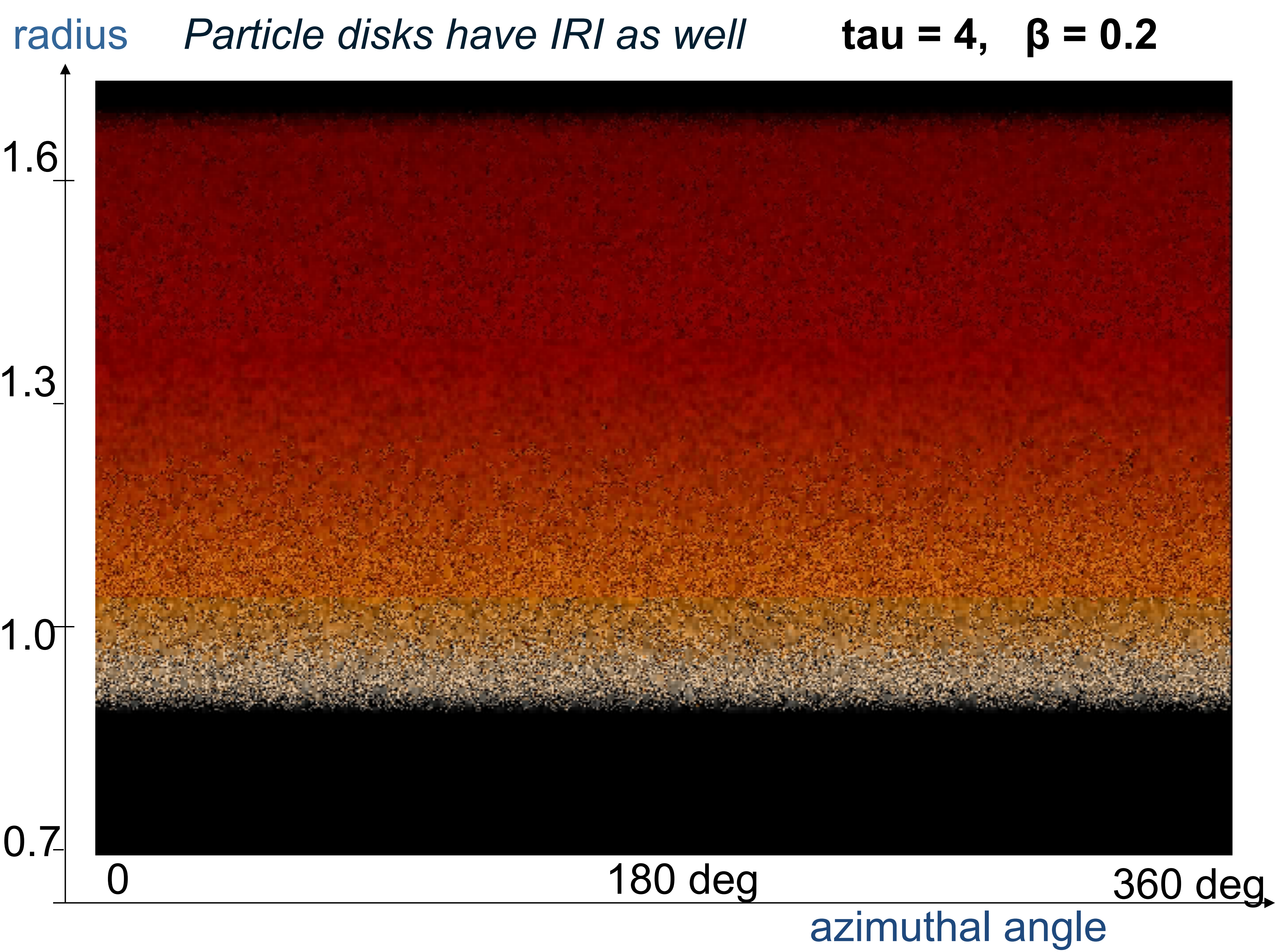
Analytical predictions are in agreement with calculations.

Models of disks were running faster on 3 GPUs than on UCB 128-cpu cluster.



Opaque disks are unstable under illumination by the central object

Particle disks have IRI as well **tau = 4, β = 0.2**



Lecture 12 – overflow topics

- **Introduction to Fast Fourier transform**
- Fourier series and Fourier integral. Convolution theorem.
- Why:
- (f*g)(x) = convolution in real space (or time) is costly $O(N^2)$
- How:
- f(x),g(x)→ f(k),g(k) →f(k)*g(k)→ FFT^{-1}(f*g)
  - Digital FT,
  - Fast Fourier Transform: O(N ln N)
  - Examples