❑ LECTURE 11

Newest computing trends:
Grace Hopper : CPU+GPU board for supercomputing, Nvidia 2023 (planned)
Software environment.

CUDA practice

Lagrangian hydrodynamics:
about the Smoothed Particle Hydrodynamics (SPH) project
How SPH works to simulate gas

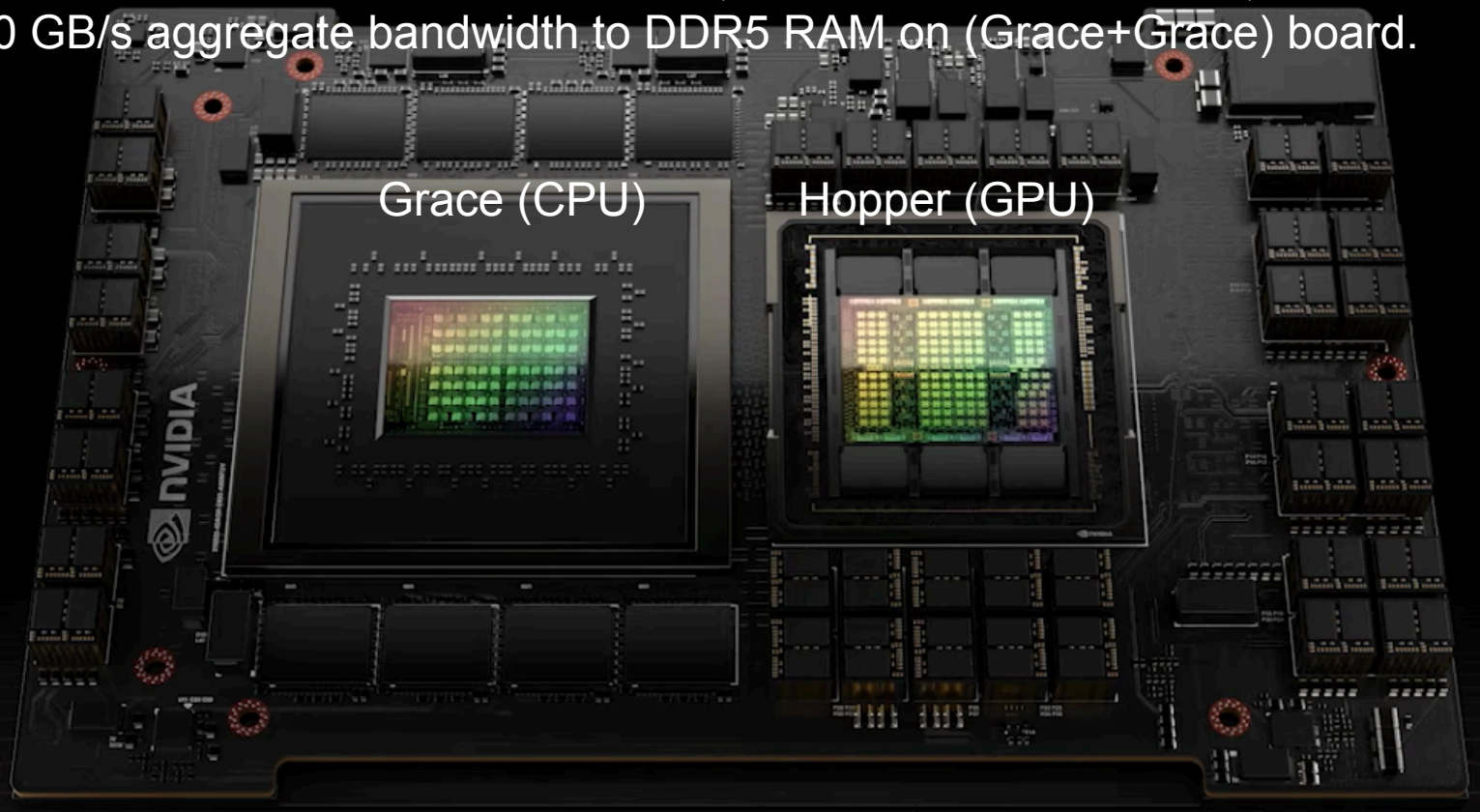
Deterministic chaos, basics. Orbital dynamics and chaos.
Background for project 1 of the 4 final projects to choose from.

(c) Pawel Artymowicz UofT, 2022 Only for use by enrolled UTSC students

Near-term future of HPC

In 2023 NVIDIA plans to sell CPUs and GPUs mounted on one board:

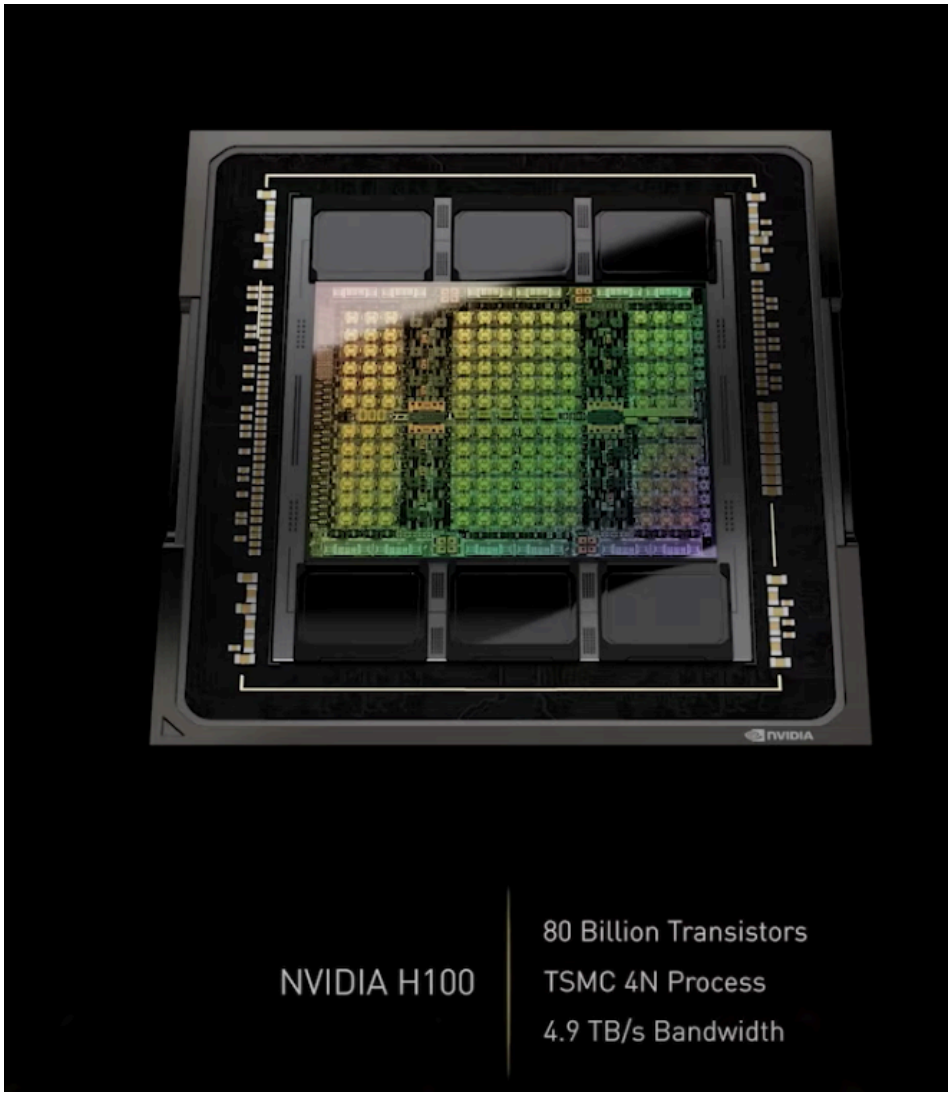
CPU: 1 or 2 ARM-instruction set CPUs, each with max. 72 cores,
1000 GB/s aggregate bandwidth to DDR5 RAM on (Grace+Grace) board.



ANNOUNCING
NVIDIA GRACE HOPPER

Grace Hopper Superchip
Densest NVIDIA Accelerated Computing System
New NVLink Chip-to-Chip Coherent Inference
900 GB/s ← CPU/GPU link called NVlink C2C

NVIDIA GPU named **H100** consumes up to 300W, works with up to 80 GB of RAM

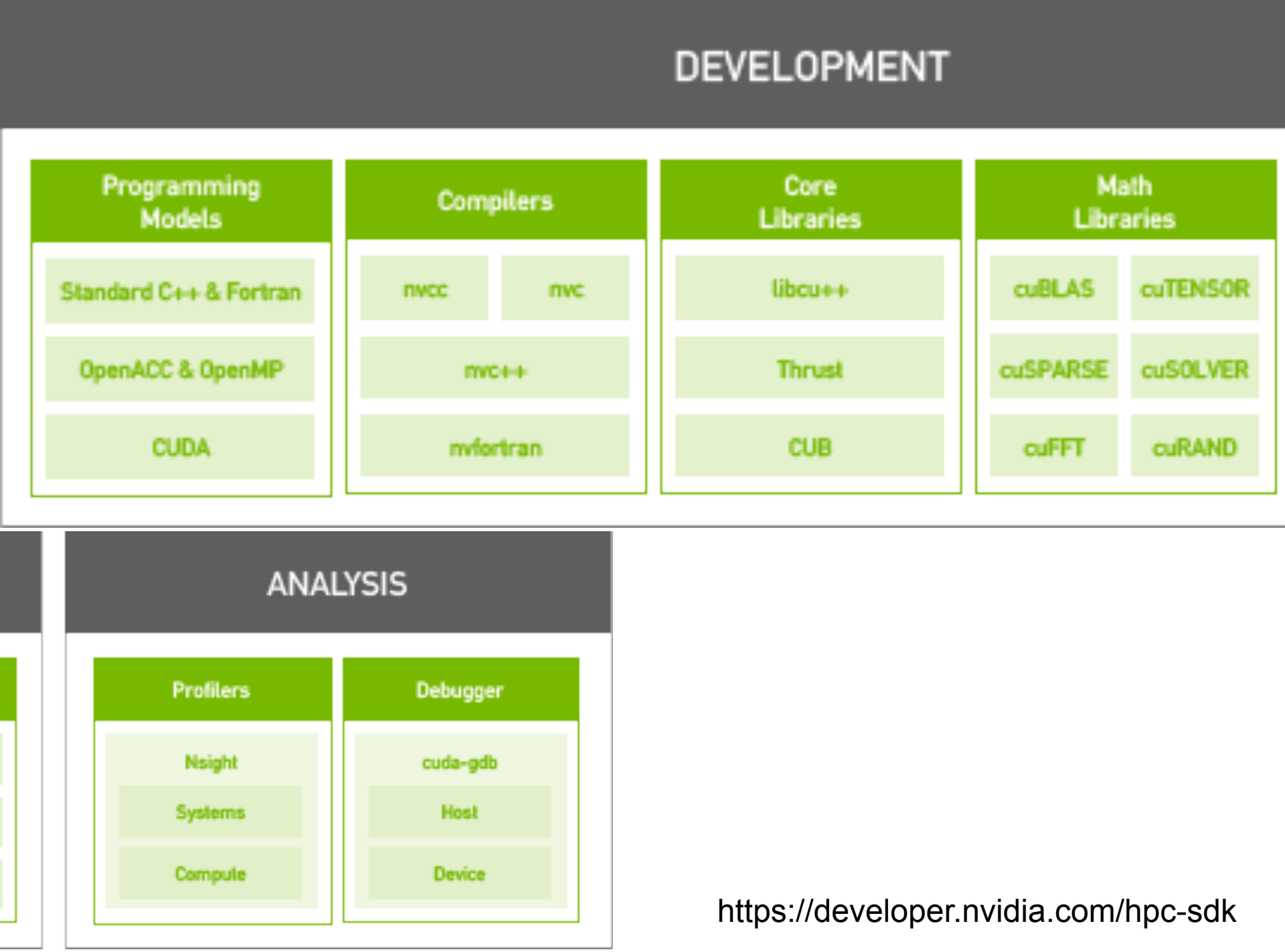


According to press release on 3/23/2022:

- GPU memory bandwidth 2-3 TB/s
  (4.9 TB/s ??, see picture)
- 6x faster 3D FFT tan previous GPUs
  tested on $(4K)^3$ FFT
  &
  a similar speedup in gene sequencing
- 24-30 TFLOPS of arithmetic perform.
  in fp64 and fp32 (2x more in tensor ops)
- claimed ~1000 TFLOP in fp16
  (AI tensor cores)

talks to other computer parts at up to
128 GB/s (new PCIe version)

https://www.nvidia.com/en-us/data-center/h100/

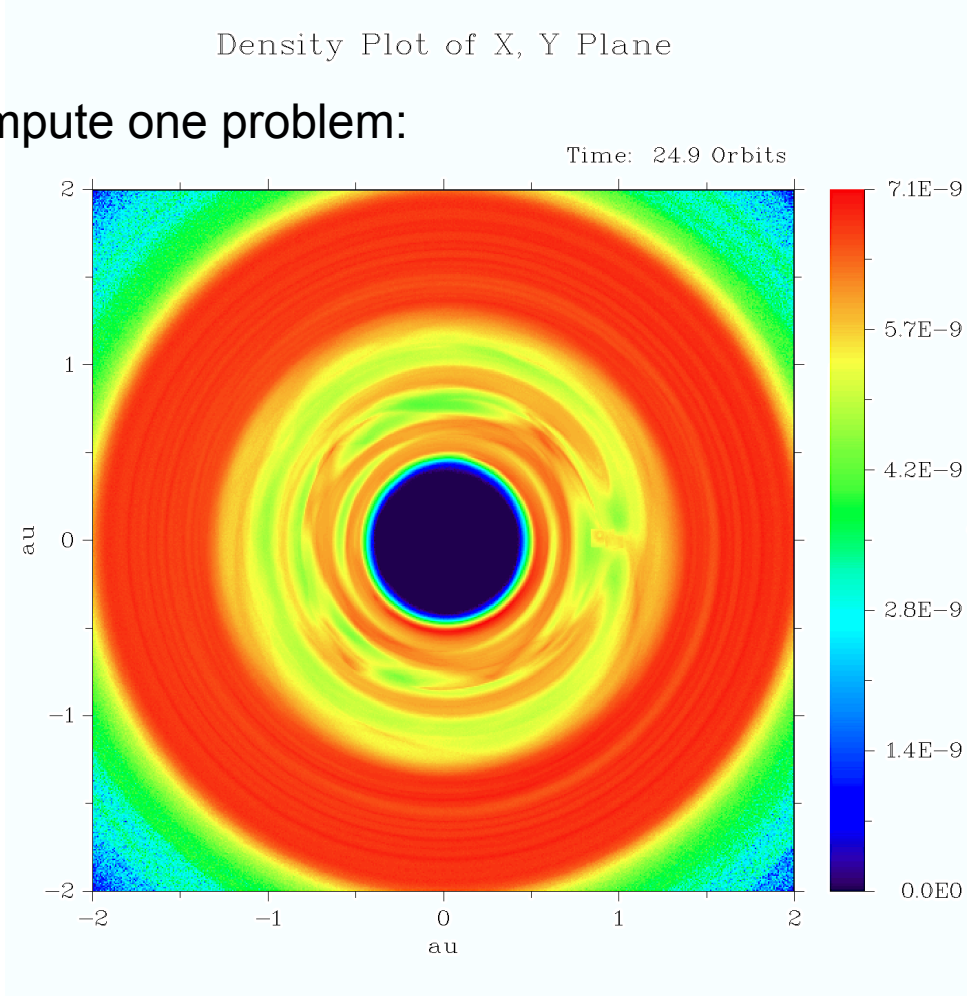


https://developer.nvidia.com/hpc-sdk

CUDA practice: cf. phyd57 account on art-2, directory progD57

simple_saxpy_cuda-.cu	CPU code in C
simple_saxpy_cuda.cu	CUDA C code for compilation with nvcc
cuda-perf.f95	CUDA Fortran code doing the same with more diagnostics
cudafor-laplace3-dp.f95	CUDA Fortran code doing heat equation modeling. Smears a 1024 x 1024 image repeatedly, to measure performance & compare different methods and languages

Many processors (all of their cores) compute one problem:



Another use of parallelism:
same algorithm, many

K x N-body

K > 1000, N <10 (planetary system) (final project)

A crash course.....

Is the Solar System orbitally stable?…

Yes, it appears so in practical sense (no orbit crossings, ejections, collisions of major bodies for billions of years), but we cannot be absolutely sure!

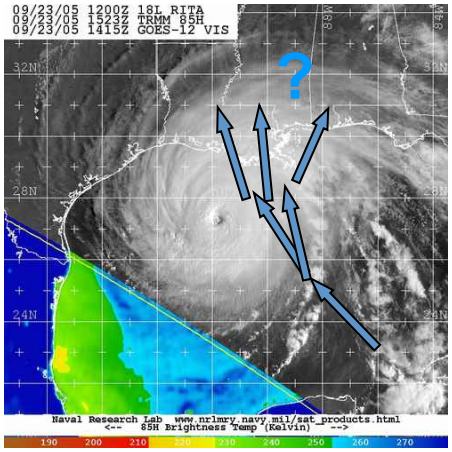
Semi-analytical and numerical simulations of the future of
Solar System show that **chaos** rules the orbits on long enough time scales.
Beyond a certain time (called Lyapunov time), results become a statistics of various possible outcomes rather than a unique prediction.

Chaos does not necessarily mean that *orbits are crossing* or that there must come to a mayhem. The more massive planets are always near their current places on timescale of Hubble time (10 Gyr).
It may mean that we don't know exactly the orientation and eccentricity of an orbit, and the position along that elliptic path.

So is the Solar System stable for sure?

There is no certainty, now or ever.

The reason is that, like the weather on Earth, the detailed configuration of the planets after 1 Gyr, or even 100 mln yr is impossible to predict or compute.



Hurricane Rita, Sept. 23, 2005

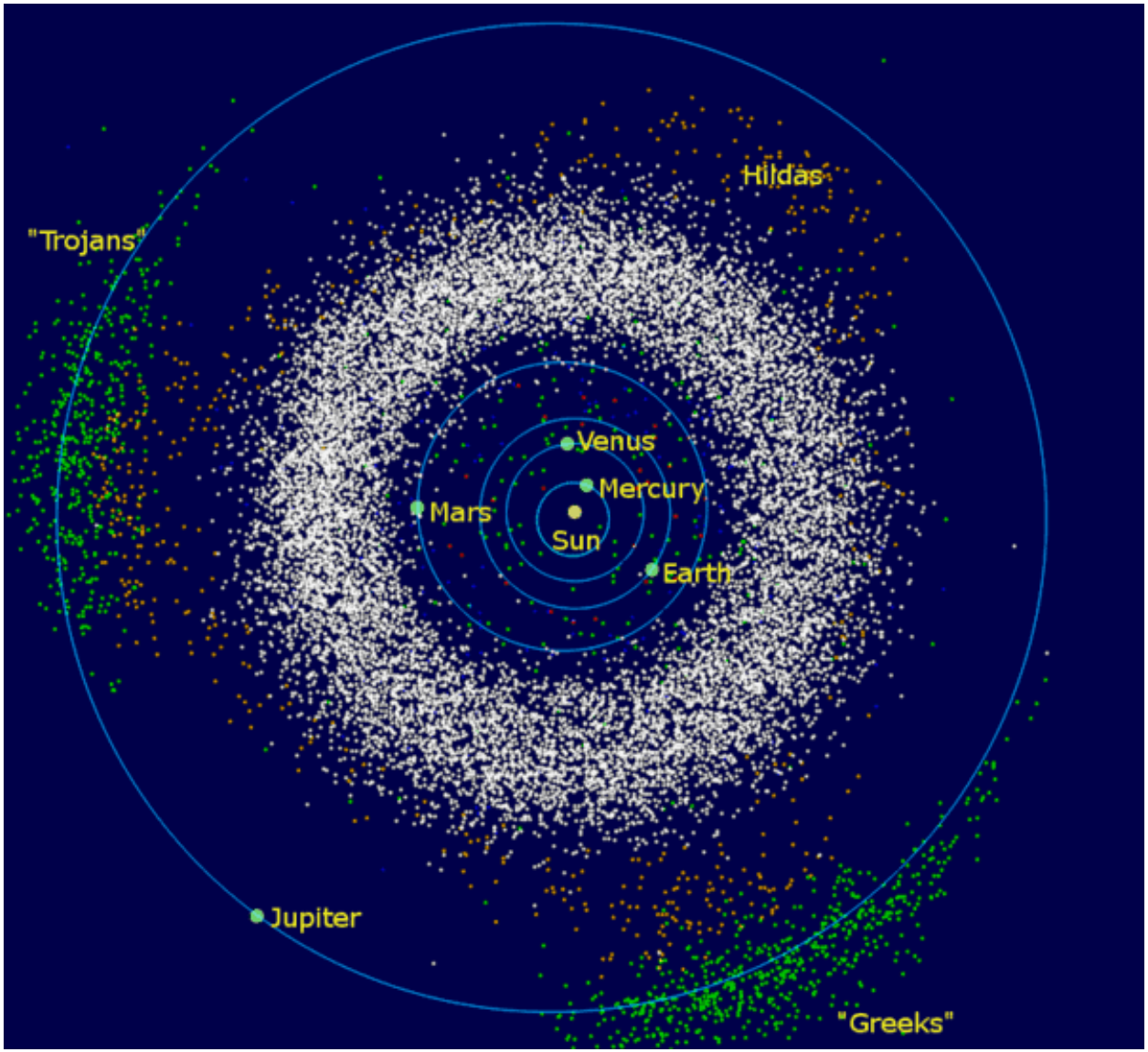
On Earth, this is because of chaos in weather systems (super-sensistivity to initial conditions, too many coupled variables)

In planetary systems, chaos is due to planet-planet gravitational perturbations amplified by resonances.

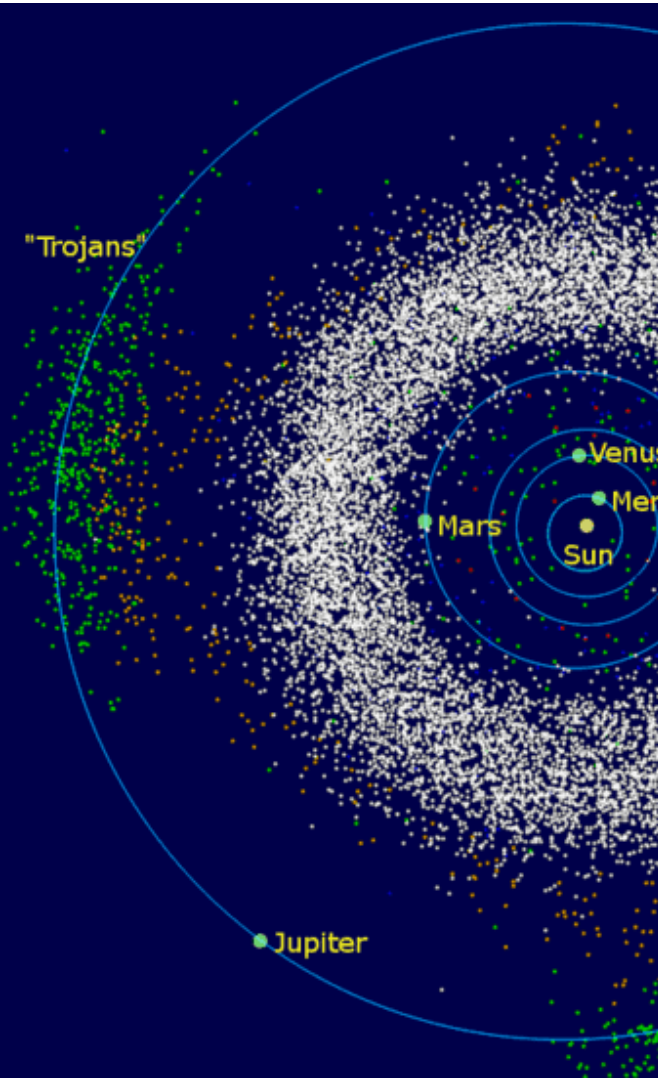
Two or more overlapping resonances can make the precise predictions of the future futile.

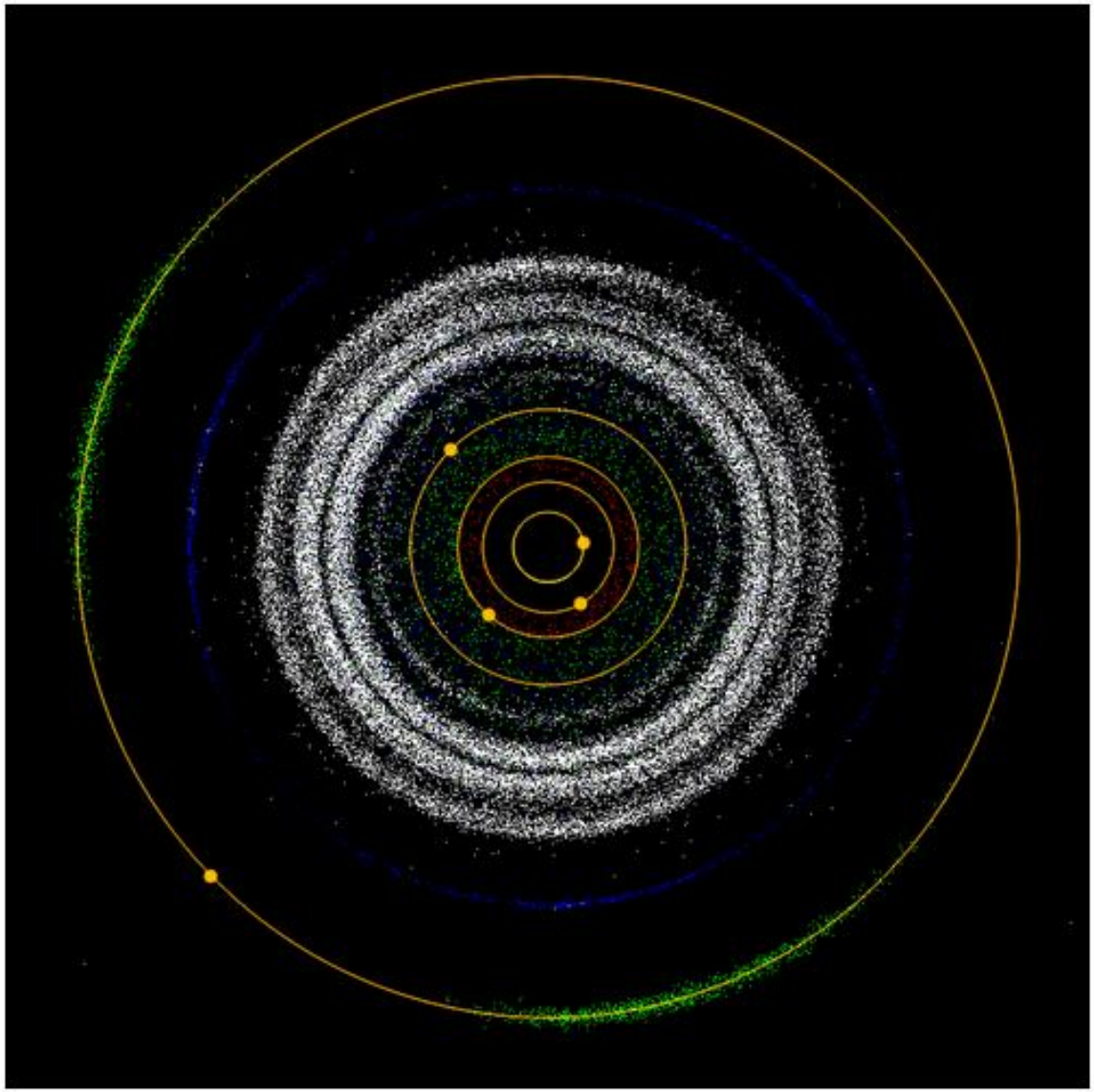
(it weakened unexpectedly fast)

ORBITAL RESONANCES - example: Astroid Belt between Mars and Jupiter. Clearly visible are 1:1 resonant objects (Trojans and Greeks). Other commensurabilities of mean motions (periods) *are* present but smeared out by eccentric motion on elliptical orbits.

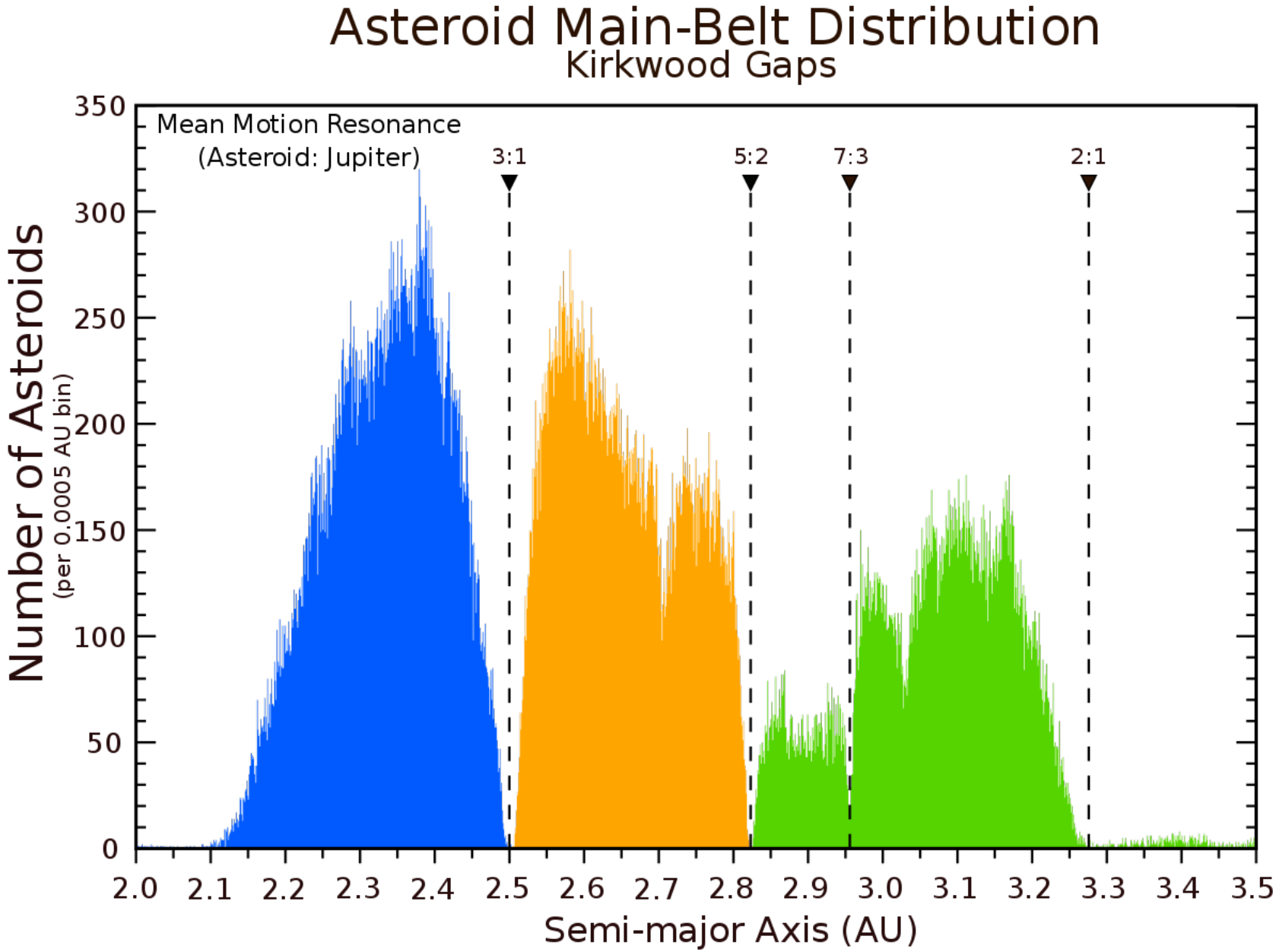


ORBITAL RESONANCES – visualization of a and ω as polar coord. on the right.

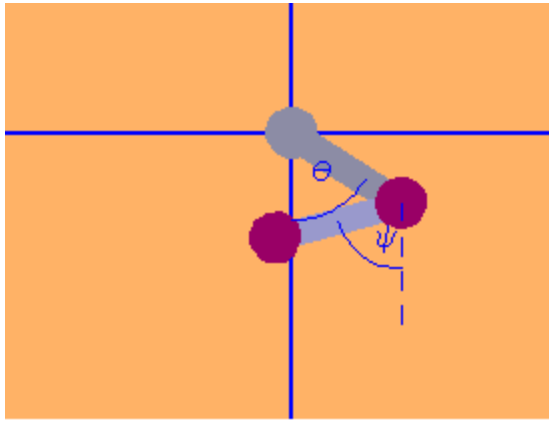




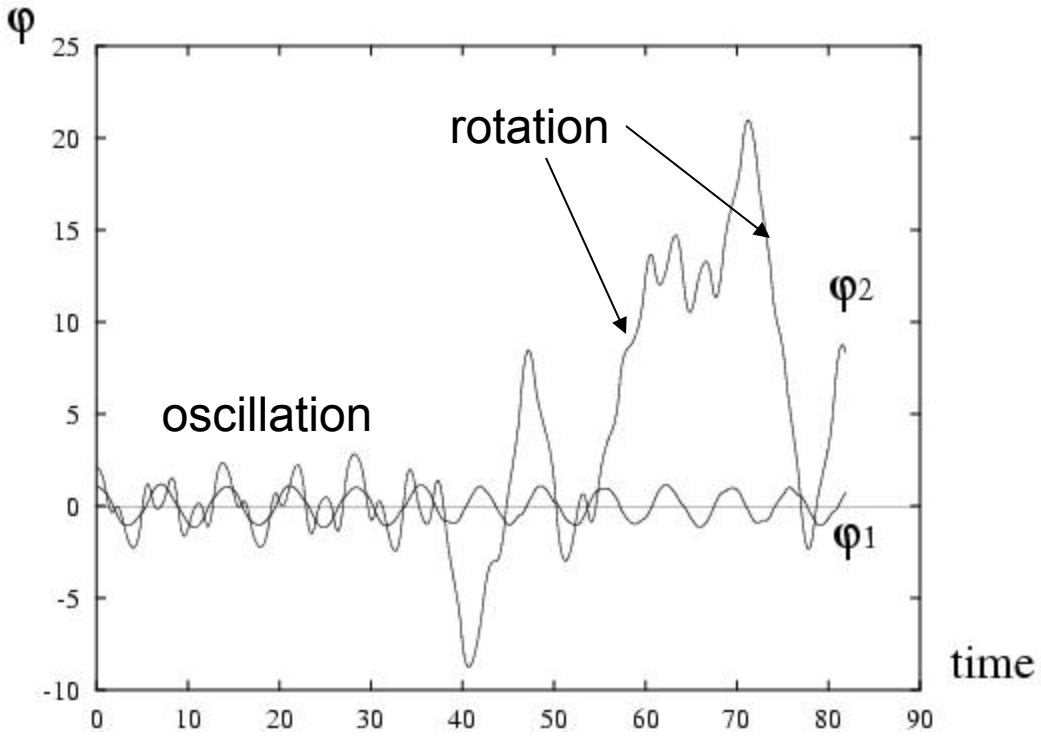
Resonances are of different types, e.g., mean-motion commensurabilities that we find in the so-called Kirkwood gaps:



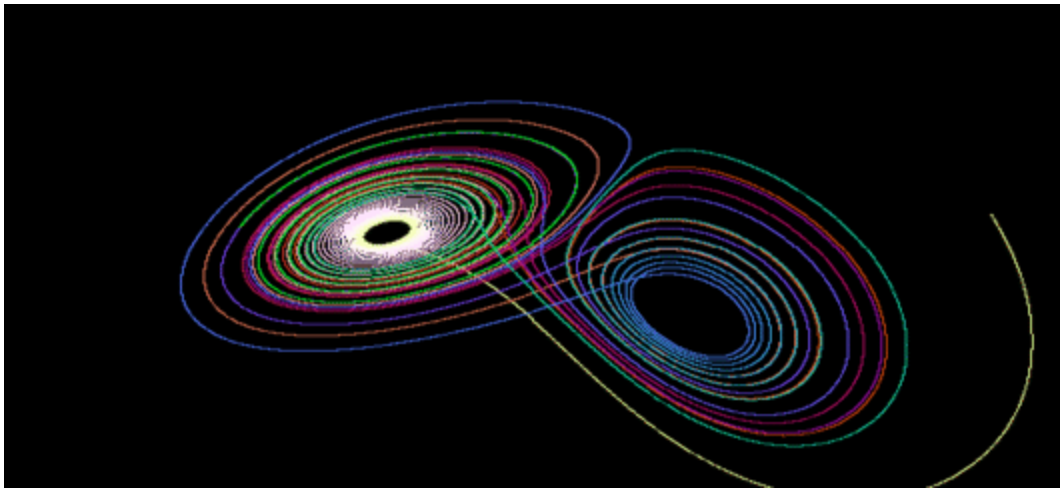
Chaos in:



Double pendulum



$$\frac{dx}{dt} = -10x + 10y, \quad \frac{dy}{dt} = 28x - y - xz, \quad \frac{dz}{dt} = -\frac{8}{3}z + xy$$



Lorentz attractor
(modeled after weather system equations by meteorologist Ed Lorenz)

In the Solar System, in 2-body resonances resonant angles librate (i.e. oscillate)

Table 8.8. Known first- and second-order mean motion resonances involving planets or satellites in the solar system. In each case the unprimed and primed quantities refer to the inner and outer bodies respectively. All known planetary and satellite resonances are included.

System	Resonant Argument	Amplitude	Period (y)
Planets			
Neptune–Pluto	$3\lambda' - 2\lambda - \varpi'$	76°	19,670
Jupiter			
Io–Europa	$2\lambda' - \lambda - \varpi$	1°	—
Io–Europa	$2\lambda' - \lambda - \varpi'$	3°	—
Europa–Ganymede	$2\lambda' - \lambda - \varpi$	3°	—
Saturn			
Mimas–Tethys	$4\lambda' - 2\lambda - \Omega' - \Omega$	43.6°	71.8
Enceladus–Dione	$2\lambda' - \lambda - \varpi$	0.297°	11.1
Titan–Hyperion	$4\lambda' - 3\lambda - \varpi'$	36.0°	1.75

Example of a chaotic orbit due to *overlapping* resonances

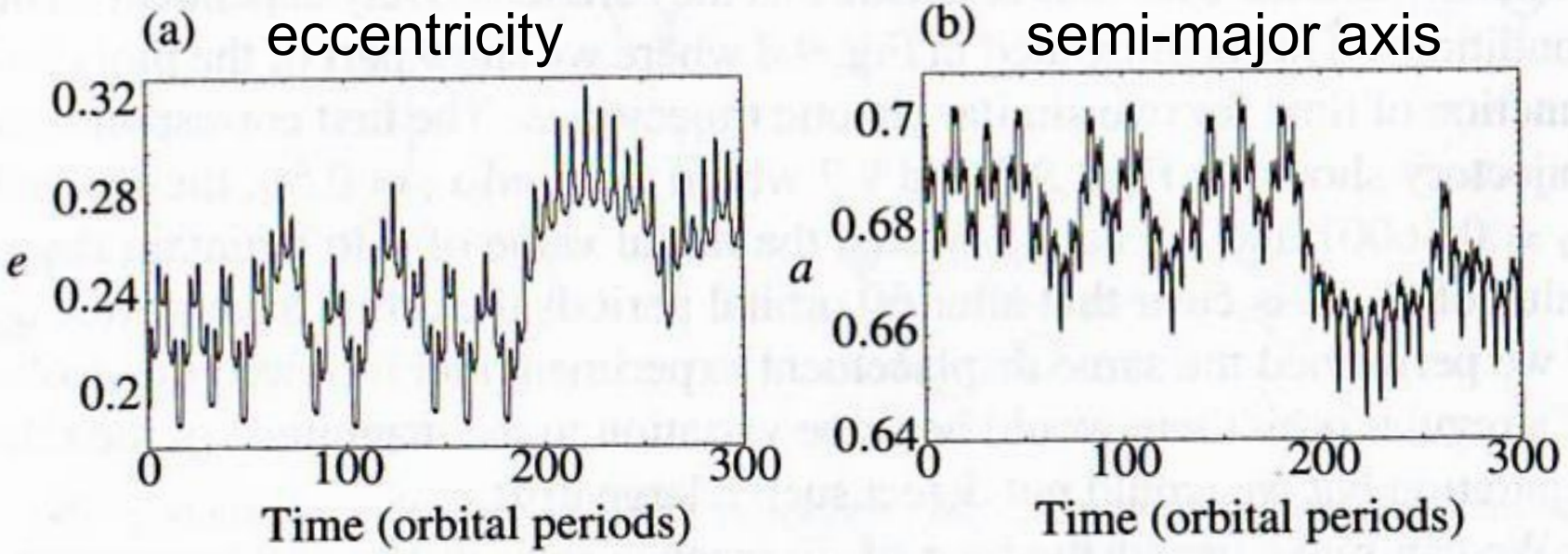


Fig. 9.6. The time variability of (a) eccentricity e and (b) semi-major axis a for initial values $a_0 = 0.6984$ and $e_0 = 0.1967$. The plots show a behaviour characteristic of chaotic orbits. (Adapted from Murray 1998.)

Orbits and planet positions on them are unpredictable on a timescale of 100 mln yr or less (50 mln yr for Earth).

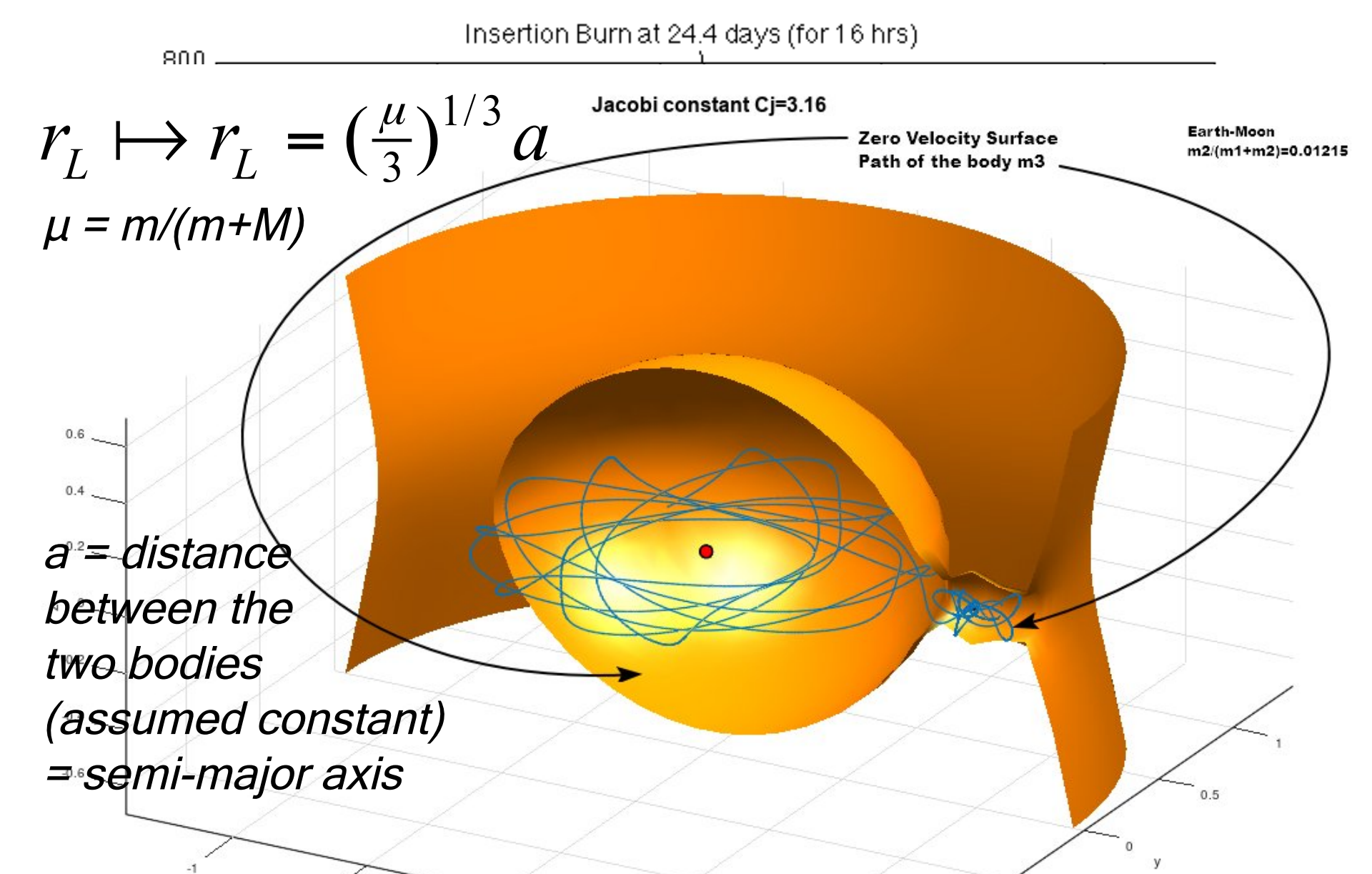
For instance, let the longitudes of perihelia be denoted by ω and the ascending nodes as Ω, then using subscripts *E* and *M* for Earth and Mars, there exists a resonant angle

$$f_{ME} = 2(\omega_M - \omega_E) - (\Omega_M - \Omega_E)$$

that shows the same hesitating behavior between oscillation (libration) and circulation (when resonant lock is broken) as in a double pendulum experiment.

But **chaos** in our system is long-term **stable** for a time of order of its age. Orbits have the numerical, long-term, stability. They don't cross and planets don't exchange places or get ejected into Galaxy.

The only questionable stability case is that of Mercury & Sun. Under the action of more massive planets, Mercury makes such wide excursions in orbital elements that in some simulations it drops onto the Sun in 3-10 Gyr.



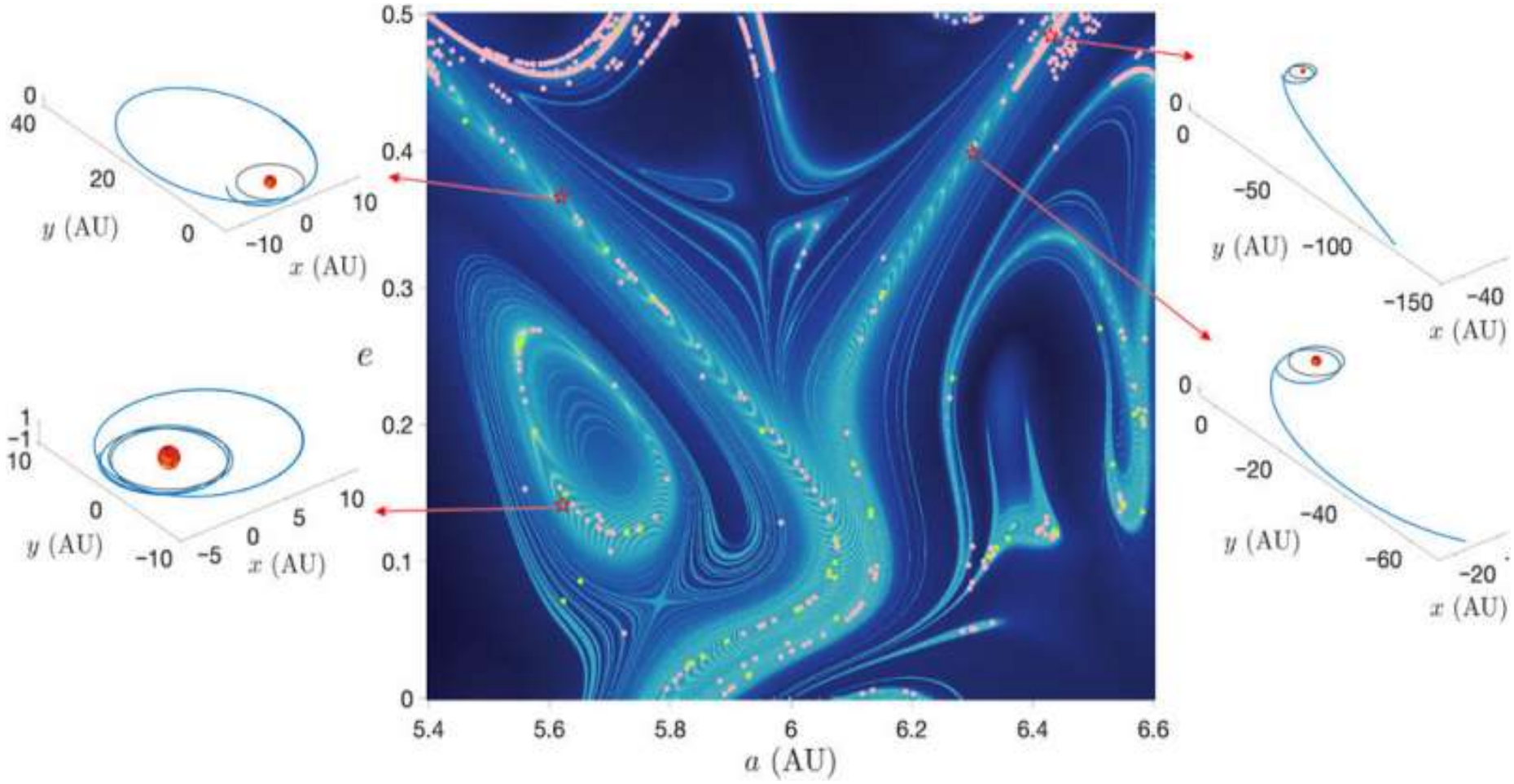
$$r_L \mapsto r_L = \left(\frac{\mu}{3}\right)^{1/3} a$$

μ = m/(m+M)

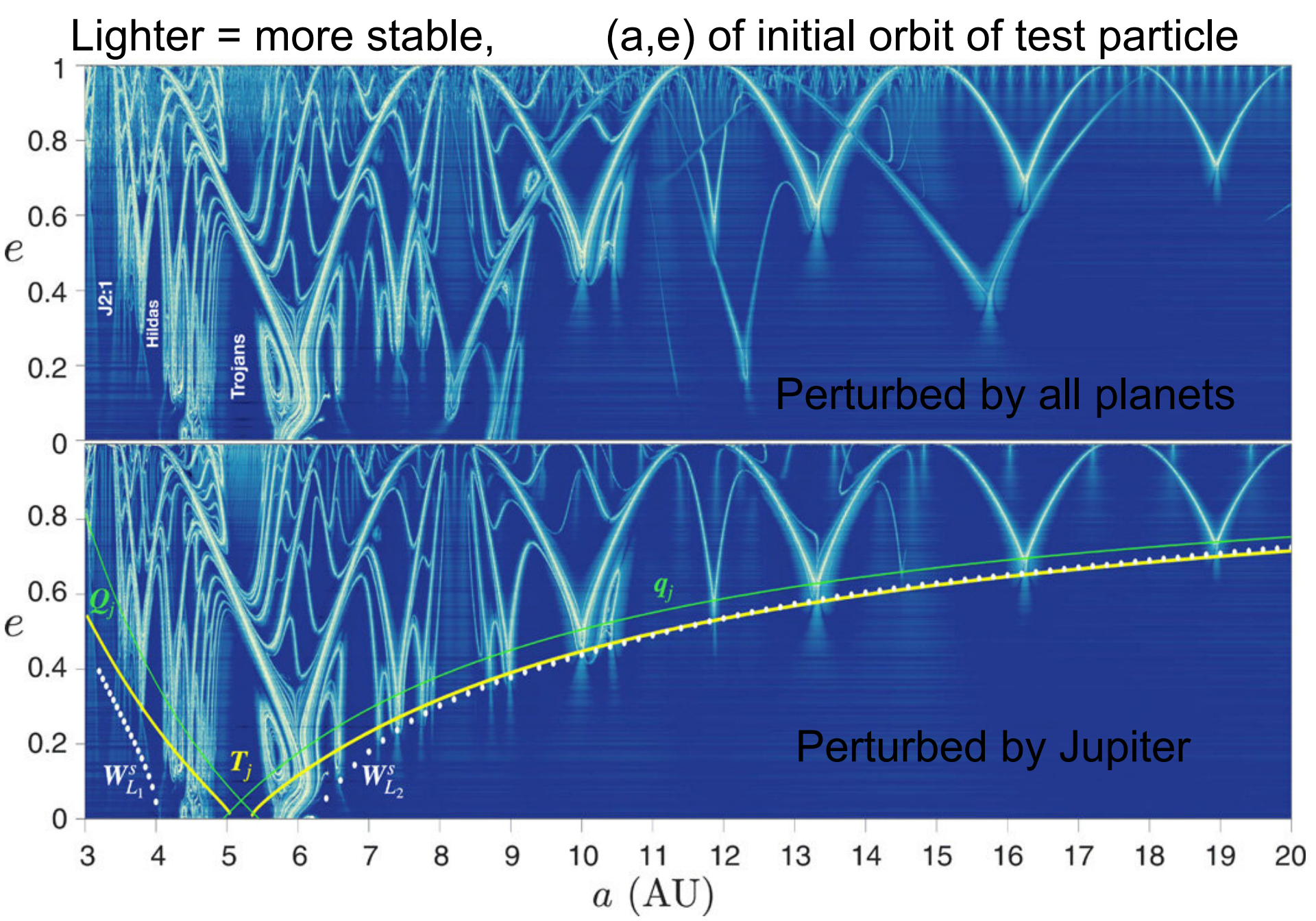
a = distance between the two bodies (assumed constant) = semi-major axis

Trajectories are often computable fairly precisely for small number of orbits only. On long time scales they are chaotic. Re-entry into the Roche lobe of a planet can occur occasionally.

There are indicators of chaos (so-called fast Lyapunov exponents) that can map stable and unstable manifolds in parameter space *(a,e)* for asteroids like Centaurs (in Jupiter-Neptune region). Centaurs use overlapping resonances to travel fast (in a few million yr) from the outer to the inner Solar System.



Nataša Todorović et al. *The arches of chaos in the Solar System*, Science Adv. (2020).



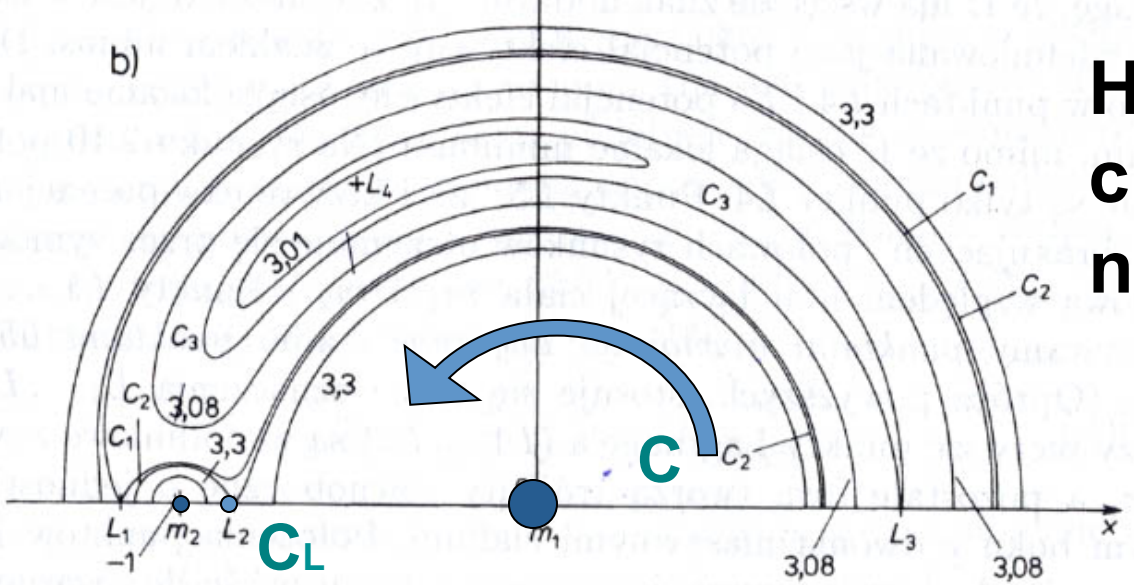
https://phys.org/news/2020-12-accessing-arches-chaos-solar-fast.html

How wide a region is quickly destabilized by a planet?
(We call it **Corotational Region**)

The gravitational influence of a small body (a planet around a star, for instance) dominates the motion inside its Roche lobe, so particle orbits there are circling around the planet, not the star. The circumstellar orbits (usually in a disk), in the vicinity of the planet's orbit are affected, too.

To what radial extent?

Corotational region defines the 'feeding zone' of a growing protoplanet. We will see how it is populated by tadpoles and littered by horseshoes...



Hill stability of circumstellar motion near the planet

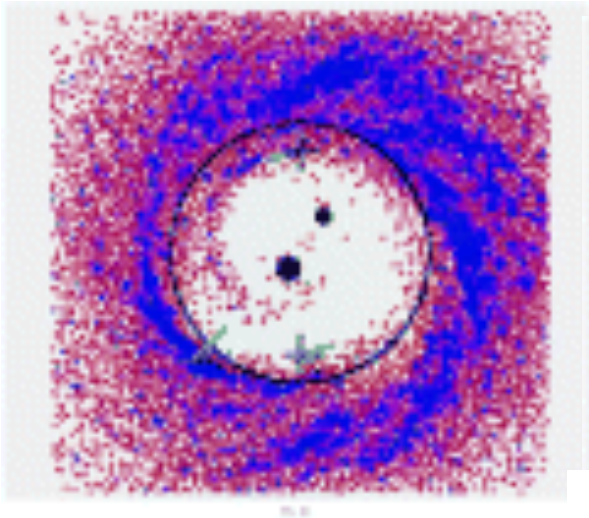
$$r_L \mapsto r_L = \left(\frac{\mu}{3}\right)^{1/3} a$$

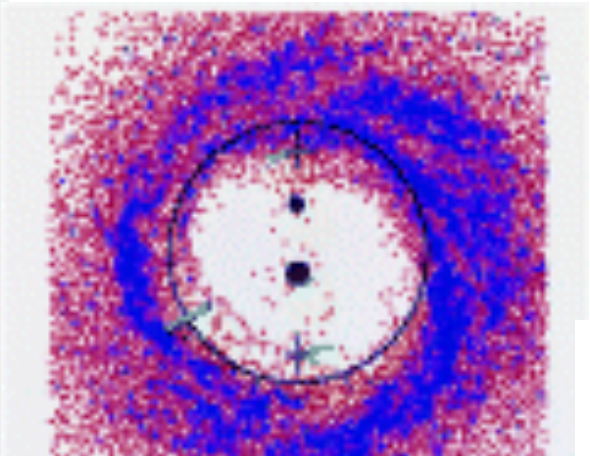
Bodies on "disk orbits" (meaning the disk of bodies circling around the star) have Jacobi constants **C** depending on the orbital separation parameter x = (r-a)/a (r=initial circular orbit radius far from the planet, a = planet's orbital radius). If |x| is large enough, the disk orbits are forbidden from approaching L1 and L2 and entering the Roche lobe by energy constraint. Their effective energy is not enough to pass through the saddle point of the effective potential.

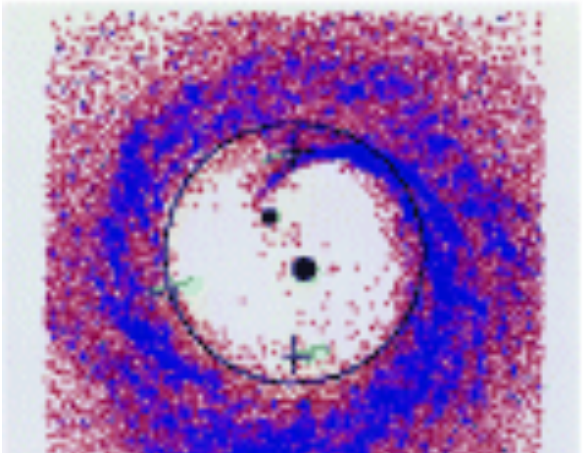
Disk regions farther away than some minimum separation |x| (assuming circular initial orbits) are guaranteed to be Hill-stable, or isolated from the planet by Jacobi energy constrain.

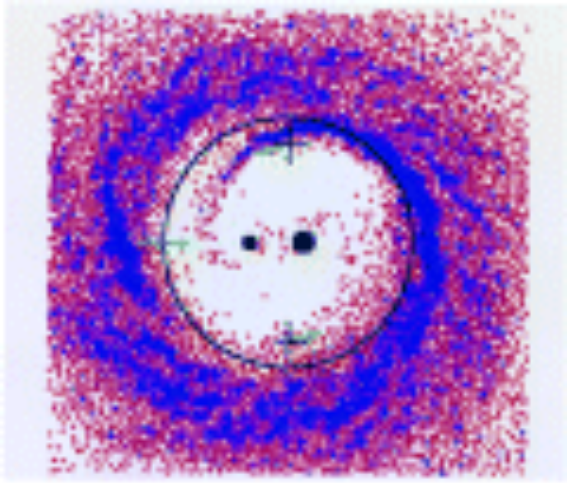
The unstable distance is +-3 sqrt(2) times the Roche lobe of a planet (r_L)

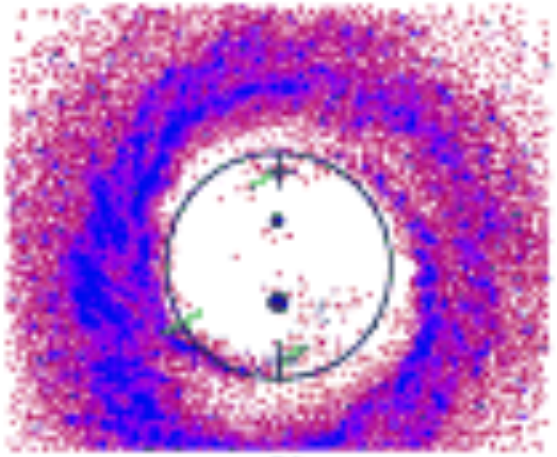
Binary-disk interaction

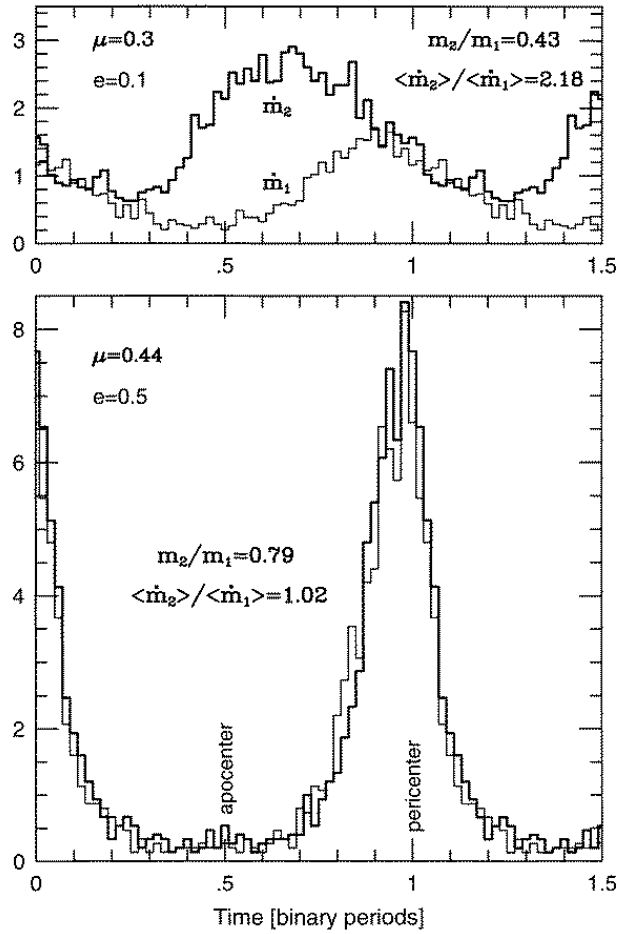












SPH = smoothed hydrodynamics: cf. wiki

Artymowicz and Lubow (1996)

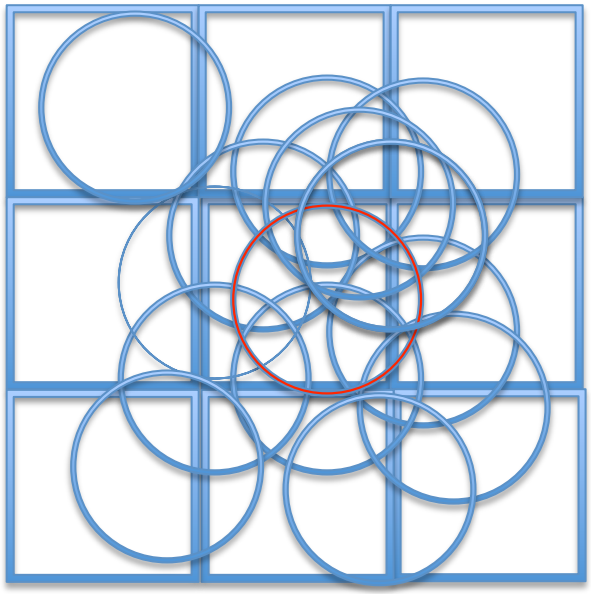
Smoothed Particle Hydrodynamics

A method of representing fluid parcels as fuzzy **clouds** a.k.a. particles

Monaghan(1977) – smoothing kernel $W(|r_i - r_j|, h)$ = *e.g., Gaussian*

Fast finding of nearest neighbors: rough grid with mesh size 2h allows to locate an SPH pa

The trick is that you only need to search 9 cells for all the neighbors



N(neighbors) ~ 20...40

Description of SPH gas dynamics implementation
is found on our main course page, cf. papers by Monaghan.

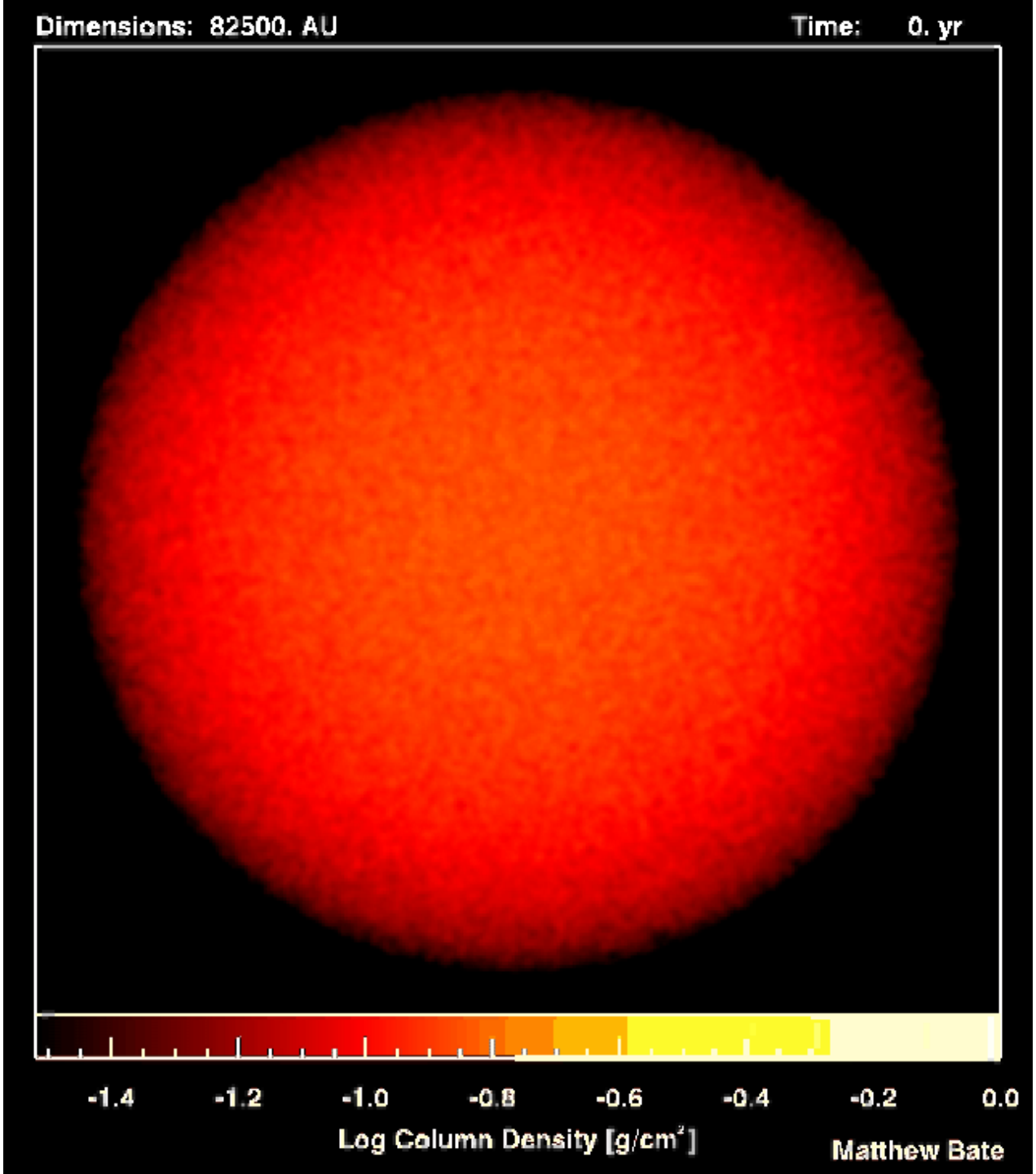
Collapse of a Molecular Cloud Core to Stellar Densities:

Rotational Instability of the First Hydrostatic Core

Matthew R. Bate

MPI für Astronomie, Heidelberg, Germany
Institute of Astronomy, Cambridge, U.K.

October 1998

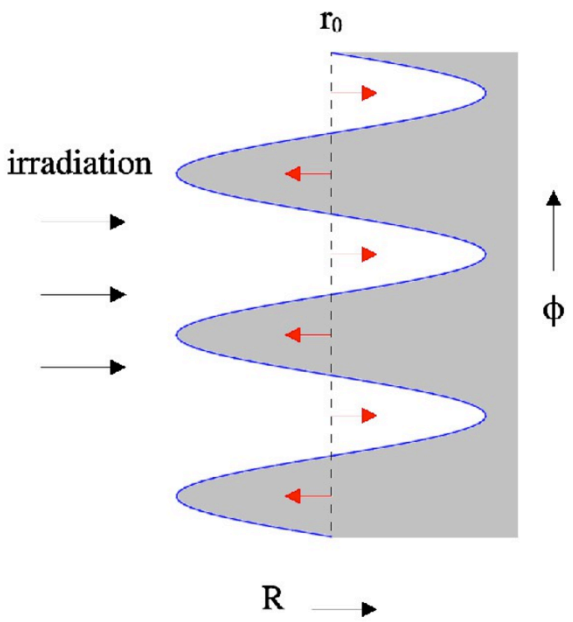


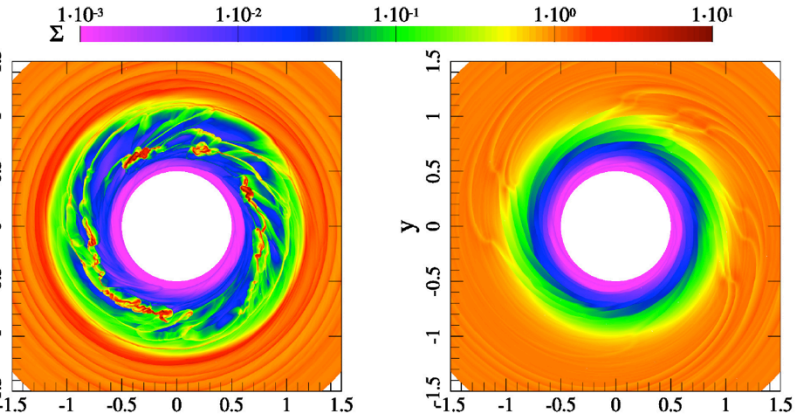
1 Mil-particle SPH simulation
from 1990s.

This was a great technical achievement back then.

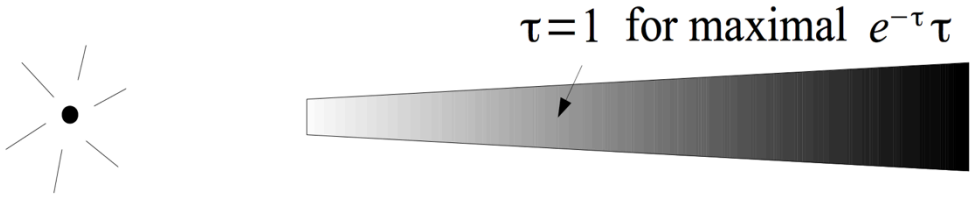
M. Bate(1990s) – star formation calculations

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





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



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

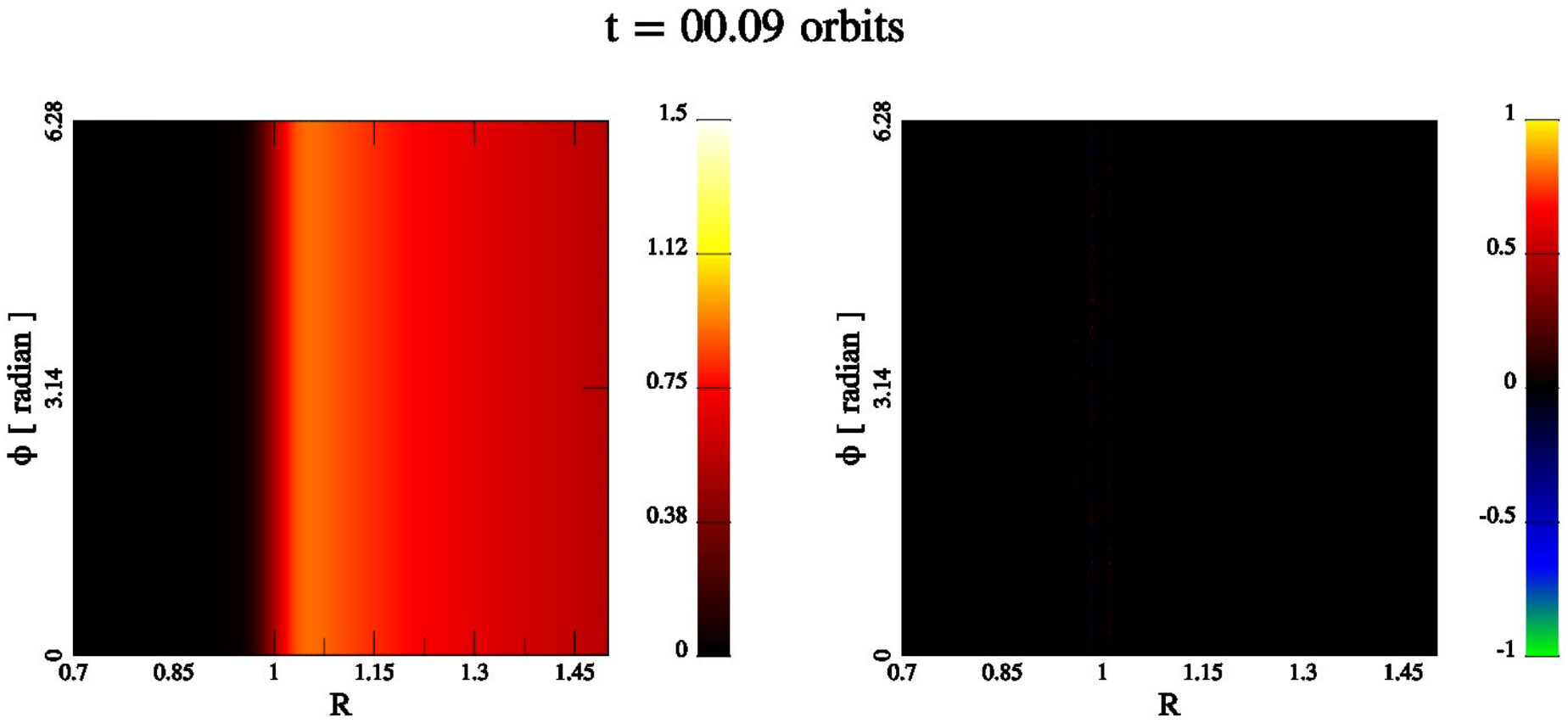
$$\Re\equiv\frac{\Sigma\Omega_k\beta e^{-\tau}}{\kappa^2}$$

See Fung & Artymowicz (2014) for details

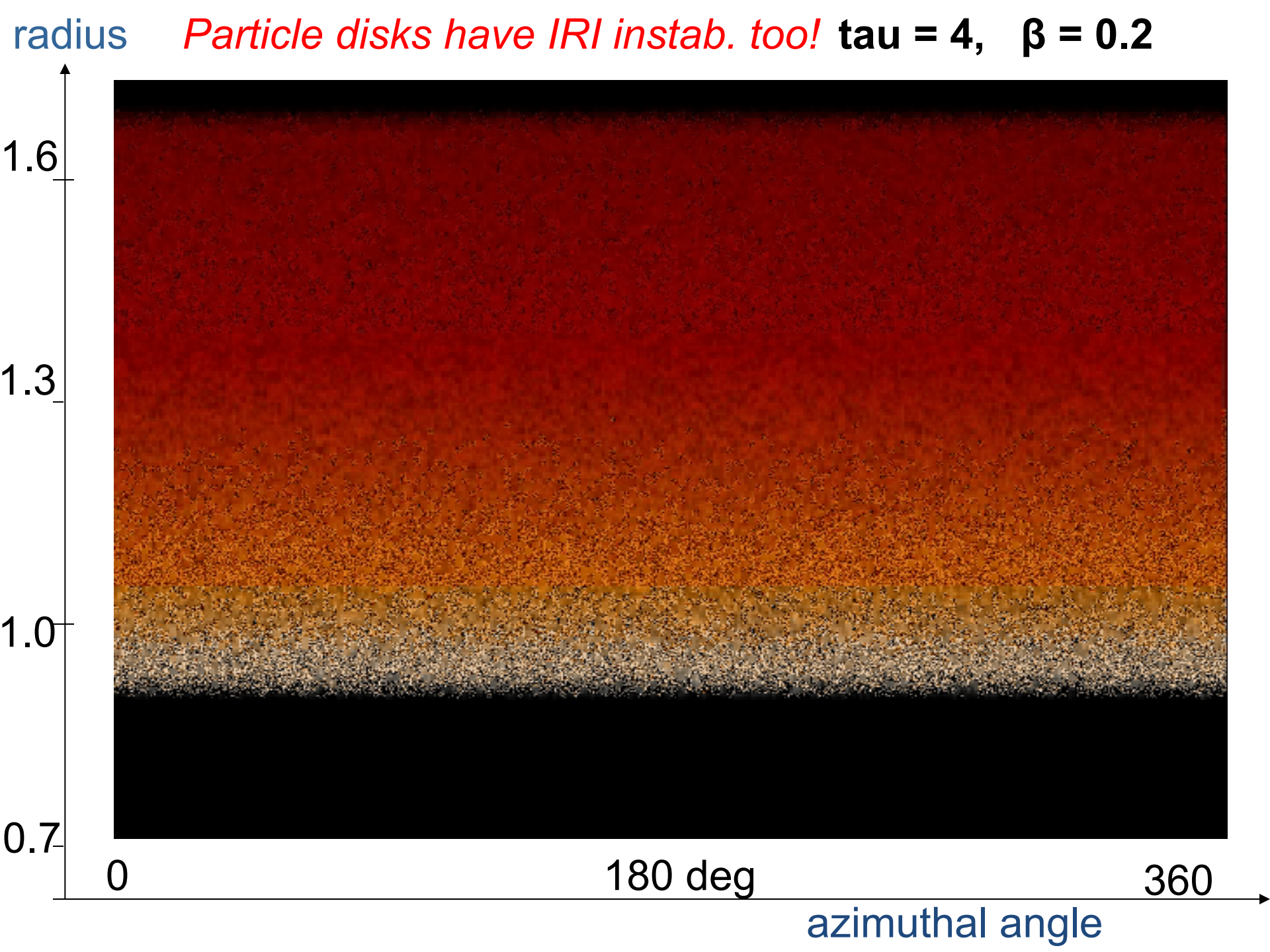
GAS DISK HYDRODYNAMICAL SIMULATION (PPM method, 2-D)

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


radius
Particle disks have IRI instab. too!
tau = 4, β = 0.2
1.6
1.3
1.0
0.7
0
180 deg
360
azimuthal angle

tau-dis-cpu.f90

```
 module dataset
       use omp_lib
       implicit none
! constants
       integer, parameter :: Tau_inf = 5,   Beta = 0.2
       integer, parameter :: Nthreads = 12, Nj = 8, Mil = 1000000
       integer, parameter :: Nr = 800/12*13, Nphi = Nr
       integer, parameter :: Np = 1*Mil /Nthreads*(1+Nthreads)
       real, parameter    :: pi = atan(1.)*4.,   Rout = 2.5
       real, parameter    :: T_max = pi
       real, parameter    :: dr = Rout/Nr
! arrays and variables
       real :: partic(Nj,Np)                  ! particle data: r,phi,vr,L
       real :: dtau(0:Nr-1,0:Nphi-1), tau(0:Nr-1,0:Nphi-1)   ! dtau,tau (r,phi)
       real :: rand_r(2,Np*3)
       real :: rad, c = 1.0
       real :: time = 0., dt = 0.004
       integer :: ix, iy, mode, i, j
```

```
 contains

!-------------------------------------------------------------------------------
 real function Sigma(x)
      real :: arg, x
      arg = max( min(x-0.666,0.666),0.)    ! 0..1 for x = 2/3...4/3
      Sigma = sin(arg*pi/2.) ! sine ramp
      if (x > Rout*0.95) Sigma = 0.
      return
 end
```

[cont. of initialize]

```
!$omp end critical

! compute tau(r) and radiation pressure force
      print*,'r,phi established, calling rad_press.'
      call rad_pressure(0)
! finish initialization of velocity, based on beta
      partic(7,:) = (1.-partic(3,:))/partic(1,:)**2  ! GM(1-beta)/r^2
      partic(4,:) = sqrt(partic(7,:)*partic(1,:))                ! v_phi from smooth force
      partic(4,:) = partic(4,:) *(1.+0.04*(partic(5,:)-0.5))   ! randomize
      partic(5,:) = partic(4,:)*partic(1,:)                      ! store vphi * r = L = const.
      partic(6,:) = 0.02 * partic(1,:)**(-0.5)  ! vr = +-0.01 * vK
  end subroutine initialize
```

partic(K,:) is:

K=1 r for all particl.
K=2 phi
K=3 beta = F_{rad}/F_{gr}
K=4 V_{phi}
K=5 L = r V_{phi}
K=6 V_r
K=7 dV_r/dt

subroutine rad_pressure(mode) when run with mode=0, does the following

(i) takes each particle and converts its coordinates (r,phi) to integers which are indexes in array dtau(r,phi), then dumps the density of particle divided by r^2 onto that grid. The r^2 modifications is because the blocking of light is really done by solid dust that blocks less “sky” if placed further from the star.

(ii) The dtau is a bit like density map in volcano program., and tau = integral from 0 to r over dtau(r) is the optical thickness from the star to the point in space.

(iii) Renormalize the computation correctly so that total tau across the disk is tau=5

```
!_____________________________________________________________________
  subroutine timestep(step)
       integer :: step
       time = time + dt
! evaluate optical thickness, beta = F_rad/F_grav,
! and radial forces (accelerations)
       call rad_pressure (1)
! evolve particles using angular momentum conservation
       partic(4,:) = partic(5,:)/partic(1,:)                    ! L/r = v_phi
       partic(2,:) = mod(partic(2,:)+dt*partic(4,:), 2*pi)      ! phi+=v_phi*dt -1
       partic(6,:) = partic(6,:) + dt*(-partic(7,:) + &          ! r += (dr/dt)*dt
                          +partic(4,:)**2/partic(1,:))
       partic(1,:) = partic(1,:) + dt*partic(6,:)                ! r += (dr/dt)*dt
  end subroutine timestep

 end module dataset
```

```
! MAIN DRIVER
!
 program IRI
       use dataset
       real (kind=8) :: t
       integer :: step

! init
       call initialize()
       print*,'disk initialized'
! main time loop
  do step = 0, int(T_max/dt)
       t = omp_get_wtime()
       call timestep(step)
       t = omp_get_wtime() -t
       if(mod(step,40)==0) print*,'time',time,' t[s]',real(t)
  end do
! sometimes
       do i = 1,Np,Np/40
         print('(7e11.3)'),partic(1:7,i)
       end do
       call display()
```