
UMEÅ
UNIVERSITET

Visual Interactive Simulation 5DV058

Outline

- Brief about broad phase collision detection
- Narrow phase collisions - Intersection tests/find – contact sets
- Separating/resting/colliding contacts
- Single contact (vs multiple simultaneous contacts)
- 1D particle bouncing on a plane – with several integrator examples
- Impulsive collision models
- Pentalty models – damped harmonic oscillator
- Collision impulse (Newton Impulse) and stepping
- Coulomb Friction and Newton-Coulomb impacts

Broad phase collision detection

- Detects collisions at ”typically large distances”.
- E.g. Spatial subdivision, bounding volume hierarchies (BVH), and sweep-and-prune.
- Only briefly covered in this course except particle-particle (and sphere-sphere) interaction where we use spatial subdivsion and hashing. Additional use is optional (bonus).
- In the course, we do explicit ”narrow phase” tests for all other cases, i.e. each and every particle is tested against a plane at every time step. Rigid bodies are also tested this way. Works up to ”tens of rigid bodies”, since cost is $\sim N^2$
- Open source libraries: Opcode (e.g. with ODE), Rapid or Solid can be used.

Narrow-phase collision detection

- Used once the the broad-phase detection reports a potential collision

- Intersection *test* is binary – gives yes/no
- Intersection *find* gives a contact set which can be used for dynamics/physics. Contact set is model dependent.

- Most collision detection libraries do not provide good intersection finds, i.e. they don’t give you a contact set! Often they just give one contact point (e.g. the deepest), while you typically want several from a manifold.

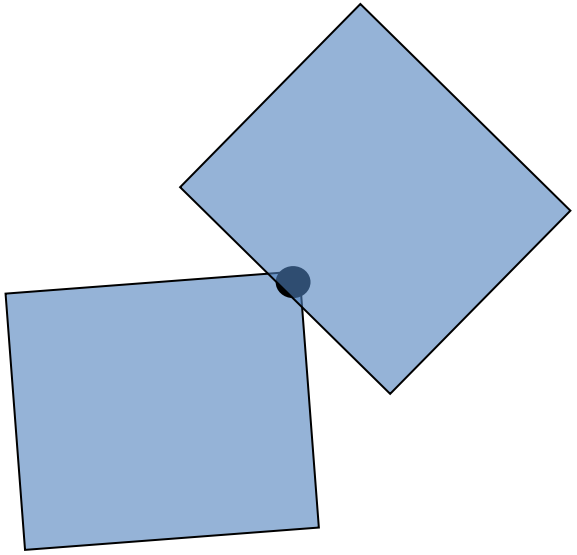
- Not much is published about contact sets. Many unsolved problems.

Intersection of primitives and features

	Particle	Sphere	AABB	OBB	Plane	Trimesh	Convex Hull	Height field	Line	...
Particle	x	x	x	x	x	x	x	x	X	X
Sphere	x	x	x	x	x	x	x	x	X	x
AABB	Etc									
OBB										
Plane										
Trimesh										
Convex Hull										
Height Field										
Line										
...										

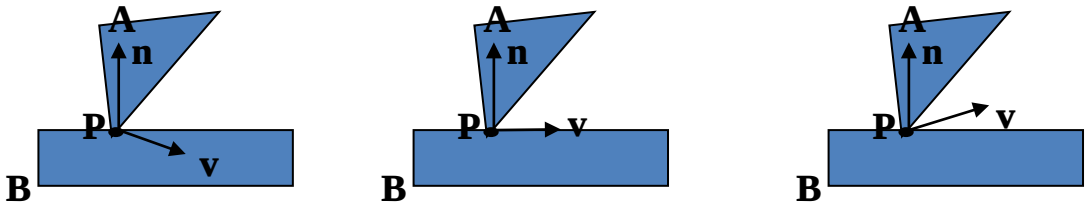
Contact set

- The contact set typically contains:
  - Reference to body/particle/entity A
  - Reference to body/particle/entity B
  - Intersection point
  - Contact normal
  - Penetration depth
    (and/or distance – often for particles)



- In this lecture we will just look at particle/plane and sphere/sphere, which is simple.
- Later we will look at oriented bounding boxes, which is trickier and illustrates the more general problems of relating contact points and manifolds.

Collisions and contacts



- Once we know that A and B are intersecting at P with normal **n,** there are three possible cases, depending on the relative velocity vector **v** of the collision points:

$\boldsymbol{n} \cdot \boldsymbol{v} < 0$	colliding contact
$\boldsymbol{n} \cdot \boldsymbol{v} = 0$	resting contact
$\boldsymbol{n} \cdot \boldsymbol{v} > 0$	separating contact

- Floating point comparisons of identities is a very strong *no-no*, use a small threshold instead.
- N.B. a ”collision” depends on a velocity, and therefore ”geometric collisions” should rather be named ”penetration” or ”interference” to avoid confusion!

Handling collisions and contacts

- Discretize time – definitions of velocity and position at impact non-trivial. Fine grained details of the stepping important!

- A general theory is developed by Claude Lacoursiere et.al. based on
  - variational integration
  - a formalism for going between (infinitely) stiff potentials and algebraic constraints
  - Friction models
  - a regularization and stabilization strategy for the stepping of constrained systems so that they conserves the important laws of physics, and leads to equations that are guaranteed to be solvable

- More details later in the course.

Single vs. multiple contacts

- If there are several simultaneous coupled contacts and collisions, we need to solve a system of equations.
- The general formulation of this is a mixed linear complimentarity problem (LCP).

- The solution strategy can be:
  - Iterative relaxational – solve one equation at a time, and let the information propagate through the system. This is often called ”impulse propagation method”, but this is quite wrong since the propagation of the impulse in general does not follow physics. Error falls linearly with iterations , O(1/n).
  - Other iterative methods e.g. Preconditioned conjugate gradient.
  - Direct sparse methods. Global solve, that dilivers the answer in fixed number of operations with machine precision.
  - Hybrids of these methods.
- More about this later!

Impulsive collision models

- Collision
  - Compression phase
  - Restitution phase.
  - Dissipation and friction can act differently during these phases.

- Three impact models to consider, that encapsulate what is going on during a collision.
  - Newton's impact law (normal velocity)
  - Poisson's hypothesis (normal impulse)
  - Stronge's hypothesis (work by normal impulse)

- Dissipation and friction
  - Coulomb's friction law

Newton's impact law

Real bodies compress and restitute.

Collision starts at $t = t_i$ and ends at $t = t_f$

and maximum compression occurs at t_{mc}

Some energy is dissipated and some is returned as kinetic energy. Relative normal velocity changed as:

$$v_n(t_f) = - e\, v_n(t_i)$$

Normal component of relative velocity changes according to a restitution coefficient $0 \leq e \leq 1$. Sets bounciness.

Poisson's hypothesis

Alternative restitution law.

Uses *e* as the ratio between the impulses rather than velocities,

$$I_t(t_f) = (1 + e)\, I_t(t_{mc})$$

States that the normal component of the impulse delivered during restitution is *e* times the normal component delivered during the compression.

Equivalent to Newton's impact law when there's no friction.

Stronge's hypothesis

- Both Newton's impact law and Poisson's hypothesis can cause total energy to increase during collisions when friction is present.

- Stronge proposed (-91) that the coefficient of restitution instead should be the ratio of the *work* done by the normal of the components of impulse,

$$W_n(t_f) - W_n(t_{mc}) = - e^2 W_n(t_{mc})$$

So that the positive work done during the restitution phase is $-e^2$ times the negative work during compression.

Stronge's hypothesis guarantees that effects of the normal force, are always dissipative.

Used in impulse based solvers to determine when collision integration should stop (see e.g. Mirtich).

Newton impact for particle against oriented plane

Relative velocity, before ($\boldsymbol{v}$) and after ($\boldsymbol{v}$') collision.

Normal velocity found by projecting on collision normal, $\boldsymbol{n}$, and this can be used to compute tangential velocity,

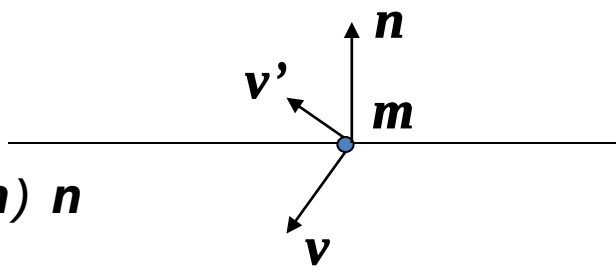
$\boldsymbol{v}_n = (\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n}$ $\qquad\qquad$ $\boldsymbol{v}_t = \boldsymbol{v} - (\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n}$

Normal velocity is damped and reflected, tangential velocity is not changed,

$\boldsymbol{v'}_n = -e\,\boldsymbol{v}_n$ $\qquad\qquad$ $\boldsymbol{v'}_t = \boldsymbol{v}_t$

Thus, post collision velocity is,

$$\boldsymbol{v'} = \boldsymbol{v'}_n + \boldsymbol{v'}_t = -e\,\boldsymbol{v}_n + \boldsymbol{v}_t = -e\,(\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n} + \boldsymbol{v} - (\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n}$$



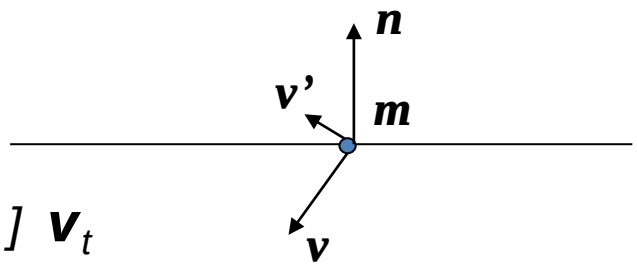
$$\boldsymbol{v'} = \boldsymbol{v} - (1+e)\,(\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n}$$

or $\qquad \boldsymbol{v}' = \boldsymbol{v} + J_n/m\;\boldsymbol{n}$

where $J_n = -m\,(1+e)\,\boldsymbol{v}\cdot\boldsymbol{n}$

Coulomb friction – Newton Coulomb impacts

Model for damping the tangential velocity.



$\boldsymbol{v'}_n = -\, e\, \boldsymbol{v}_n \quad , \qquad \boldsymbol{v'}_t = max[0,1-\mu\,(|\boldsymbol{v}_n|/|\boldsymbol{v}_t|)]\;\boldsymbol{v}_t$

Or,

$\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{J}/m$

$J = -\, m(1+e)\,(\boldsymbol{v}\cdot\boldsymbol{n})\,\boldsymbol{n} \;+\; m\,(max[0,1-\mu\,(|\boldsymbol{v}_n|/|\boldsymbol{v}_t|)] - 1)\;\boldsymbol{v}_t$

Where $0 \le \mu \le 1$ is the Coulomb friction coeffcient.

(crude approximation for static/dynamic friction since static friction is larger then dynamic friction!)

Stepper for particle on a plane

- Timestep
  - Detect an intersection
  - Check if intersection corresponds to collision, separation or resting.
  - If there is a collision
    - Compute the new velocity from the Newton-Coulomb impact law
  - If intersection corresponds to a collision, and velocity is low (details later), this is interpreted as a resting contact!
    - Treat the contact…
  - Integrate velocity and position
  - Visualize
- end

How do we make contacts stable?

1D particle on a plane (GBF)

Plain leapfrog with collision impulse for a particle bouncing on a 1D plane located at x=0. Treated as in Fedkiw, Bridson, Guendelman 2003 paper (simplified version, linearized in time).

```
if (x<0 && v<0)          //check for penetration/collision
        v = - e*v;       //reflect velocity in normal
Endif

v = v – g*dt;            //add gravity

if (x<0 && v<0)          //check penetration and _contact_
        v = 0;           //contact should have zero velocity
endif

x = x + v*dt;            //Position integrate

if (x<0)                 //project back to surface
        x=0;
endif
```

1D particle on a plane (GBF)

Strengths:

Conceptually extremely straight-forward, and easy to implement!

Problems with the algoritm:

Control over energy is awkward. First kill contact velocity, and then lift the particle in the gravitational field! Small error for a single particle, but with multiple contacts this gives way too much jitter, or requires a tiny timestep.

The position projection to x = 0 means that x < 0 is not (never!) true in the next timestep! So, we need a ”shell” to get this stable/consistent, i.e. we identify a penetration as x < delta, or at least with x <= 0.

Ad hoc ”prescription algorithm”- common in computer graphics. Gives ”plausible” simulations, but isn’t derived from underlying principles (physics in terms of math), and therefore we barely understand the method, nor its limitations nor strengths. It is certainly innovative though, and the splitting strategy inspires improved methods!

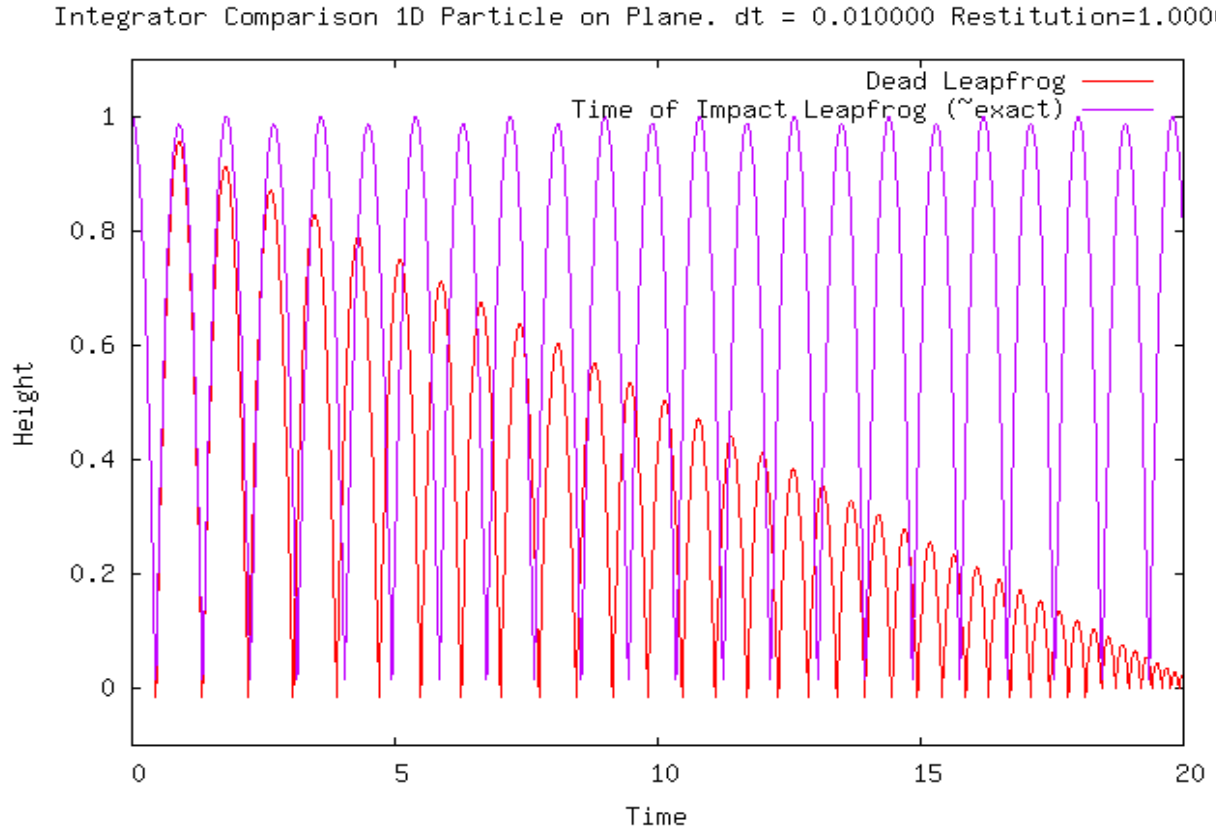
The collision impulse is computed using the previous Leapfrog velocity, which actually is v(t-dt/2), while Leapfrog is derived for forces F(t), so gravity is not balanced in the collision, which causes drift (relatively small error in h^2 but very obvious in elastic collisions).

Extension to many body case:

Each of the ”if...endif” can be done iteratively for each pair, until convergence is reached (hopefully). Errors in collisions (i.e. non-resolved collisions) are damped to death in the contact iteration.

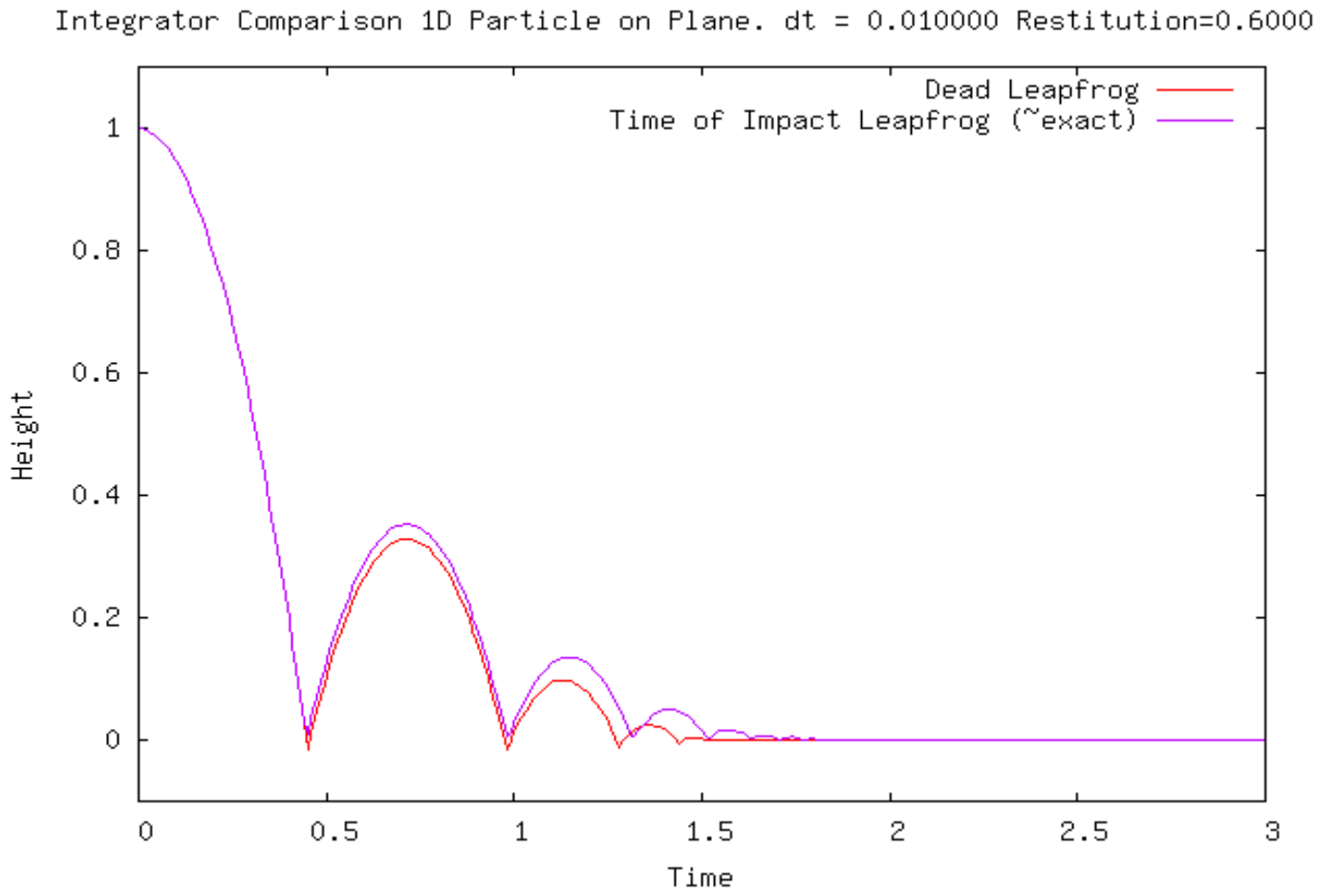
1D particle on a plane (GBF)

Shows drift/creep problem in elastic case due to imbalanced force computation.
Notice that this depends very much on size of timestep (smaller timestep = smaller problem) and typically isn't a critical problem in the inelastic case.



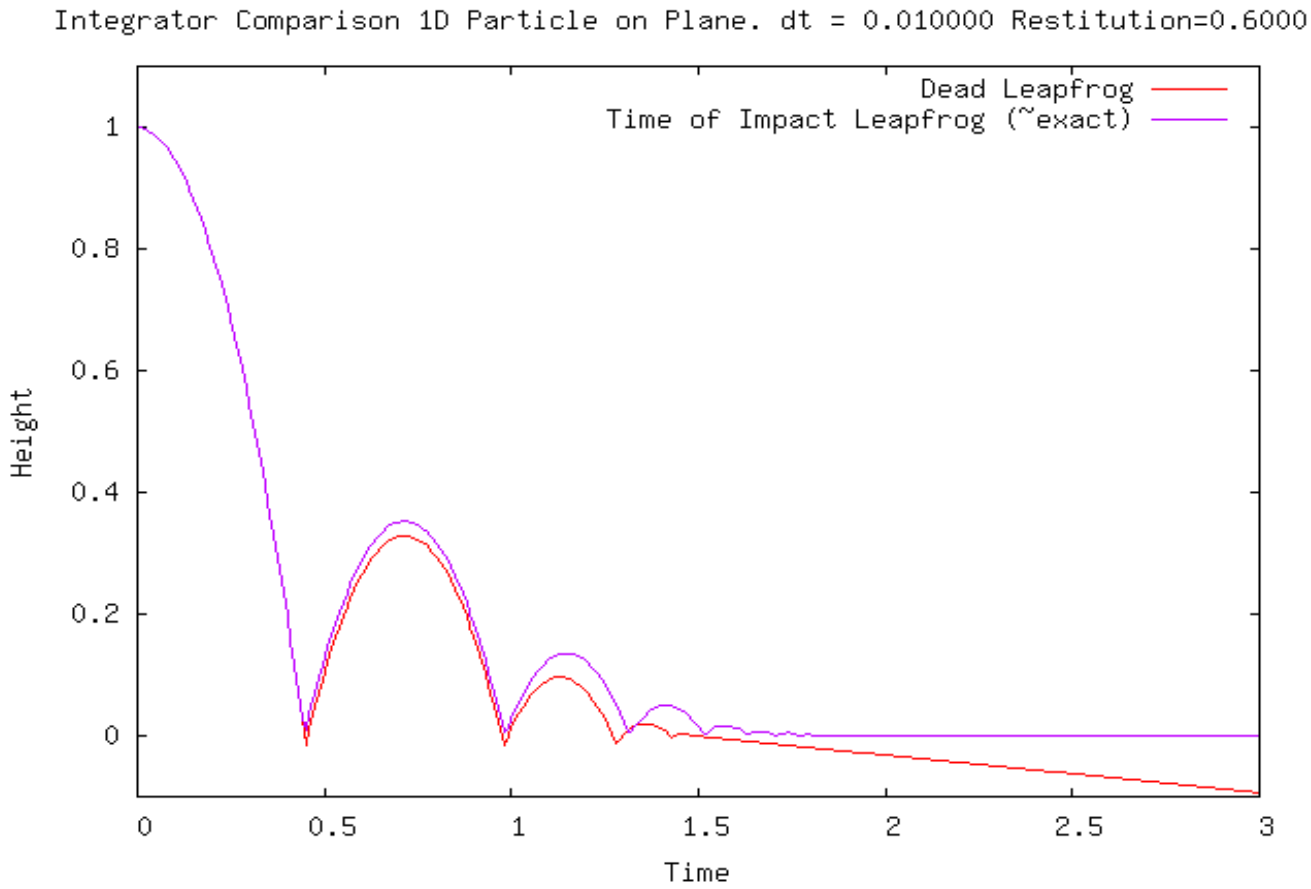
1D particle on a plane (GBF)

Same as before, but with inelastic collisions. Drift is not so noticable now.



1D particle on a plane (GBF)

Same as before, but without the contact step. This shows that impact impulses alone cannot produce stable contacts.



1D particle on a plane – ”Balanced GBF”

In Leapfrog, forces should be computed as *F(t).* However, we only know *v(t-dt/2)* and since an impulse entirely depends on the velocity, this becomes wrong. One important force we can integrate explicitly is gravity!

v(t) ~ *v(t-dt/2)* + *g dt/2* (g is gravitational acceleration)

```
v = v – g*dt/2;          //half step gravity
if (x<0 && v < 0)        //check for penetration/collision
   v = - e*v;            //reflect velocity in normal
endif
v = v – g*dt/2;          //another half step of gravity

if (x<0 && v<0)          //check penetration and _contact_
   v = 0;                //contact should have zero velocity
endif

if (x<0)                 //project back to surface
   x=0;
endif
```

1D particle on a plane – ”Balanced GBF”

Strengths:

- Balances gravity in collisions and avoids drift/dissipation of a bouncing particle.
  Costs nothing!
- The collision impulse takes care of a portion of the contact force, and thus setting $v=0$ and later $x= 0$ becomes a slightly better approximation.

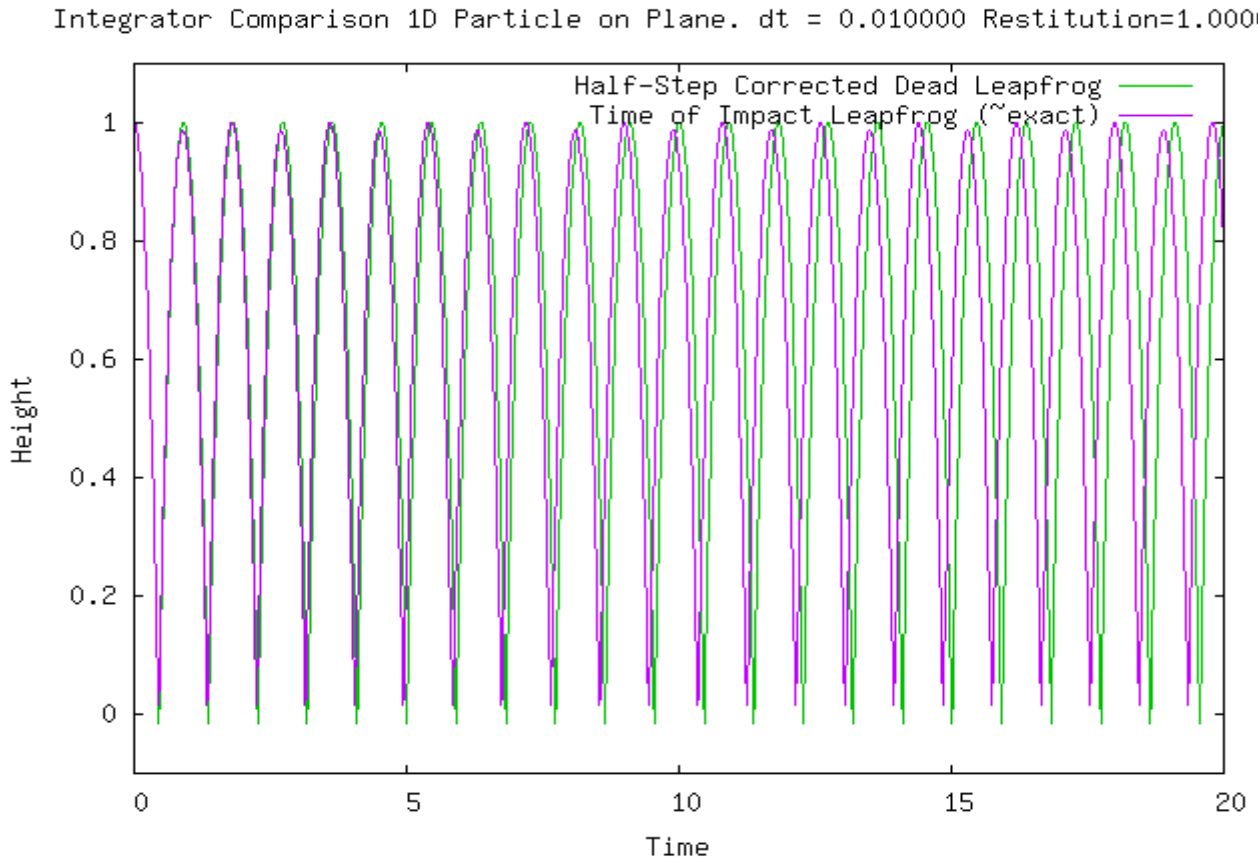
Problems:

- Like before. Patching of v and x for contacts is not a well controlled approximation.
  Only works for body-forces, i.e. gravity.

Extends well to many-body case, and gives much less jitter and is stable for larger timesteps.

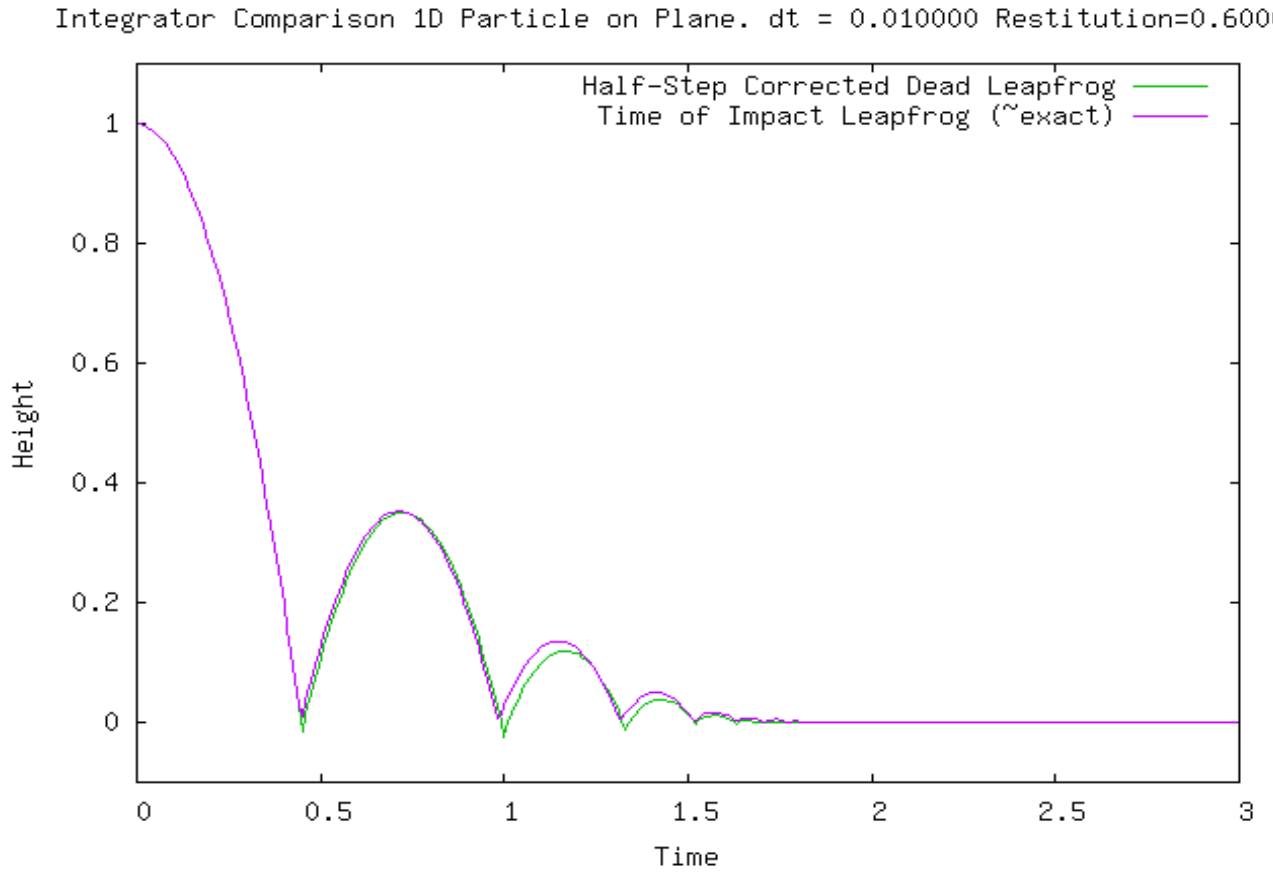
1D particle on a plane – ”Balanced GBF”

Shows how the drift is reduced when collision impulses are half-step corrected.



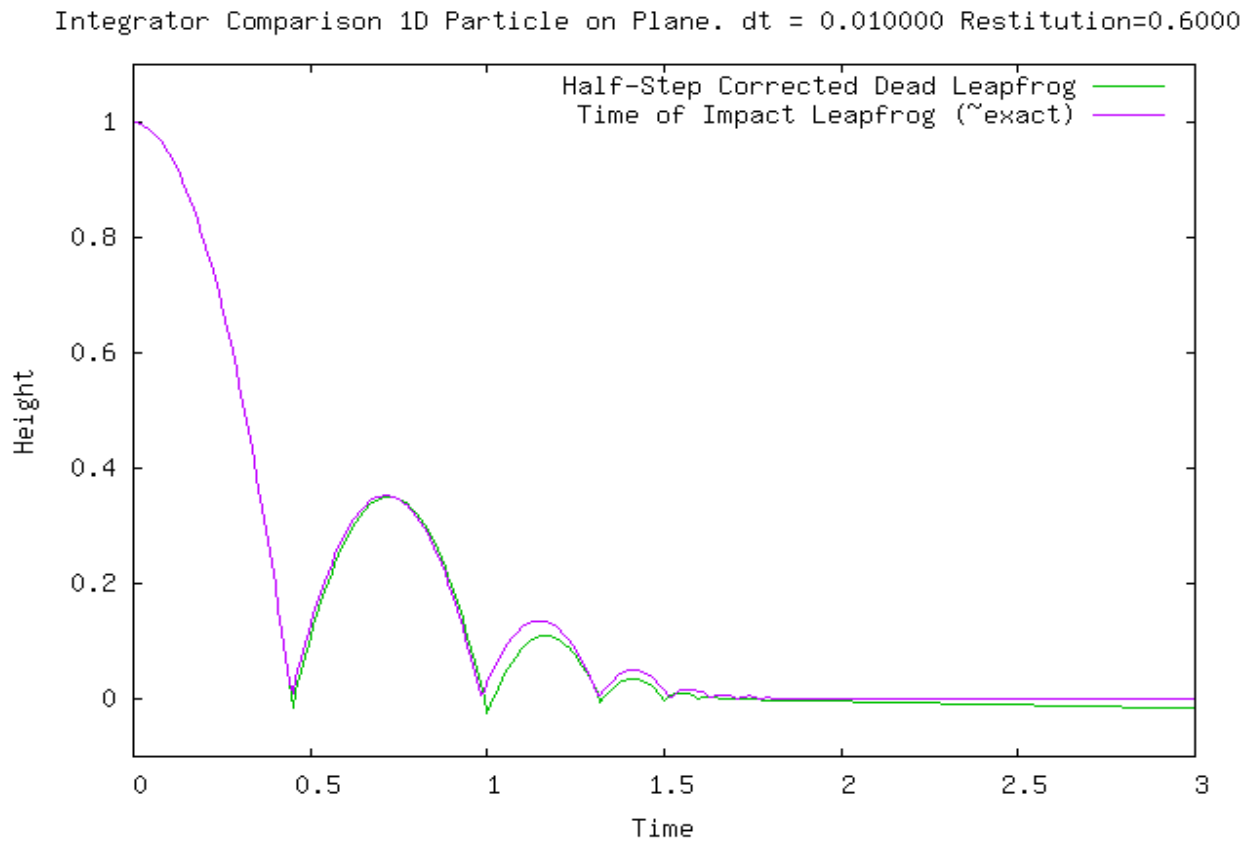
1D particle on a plane – ”Balanced GBF”

Inelastic case. The half-step correction is not so noticable here, but in fact, it makes quite a difference in the transition from bouncing to contacts, where it allows for a larger timestep.



1D particle on a plane – ”Balanced GBF”

Inelastic case, where contacts are not treated. This shows that the creep is now smaller, which means that the v=0, x=0 procedure becomes a better approximation.



1D particle on a plane – ”Spring model”

We can attach a damped spring att the penetration point, which penalizes the penetration. The stepping equation then becomes:

$v = v + F/m\ dt$ where $F = -k\,x - b\,v - mg + F_{ext}$

$x = x + v\ dt$

```
F = - m*g;              //add gravity to the force
if (x<0)                //check for penetration
   F += -k*x – b*v;  //damped spring force if we penetrate
endif

v += F*dt/m;            //integrate velocity

x = x + v*dt;           //integrate position
```

1D particle on a plane – ”Spring model”

Strengths:

- Conceptually very simple. Collisions and contacts do not differ.
- Maps very well to the microscopic case.

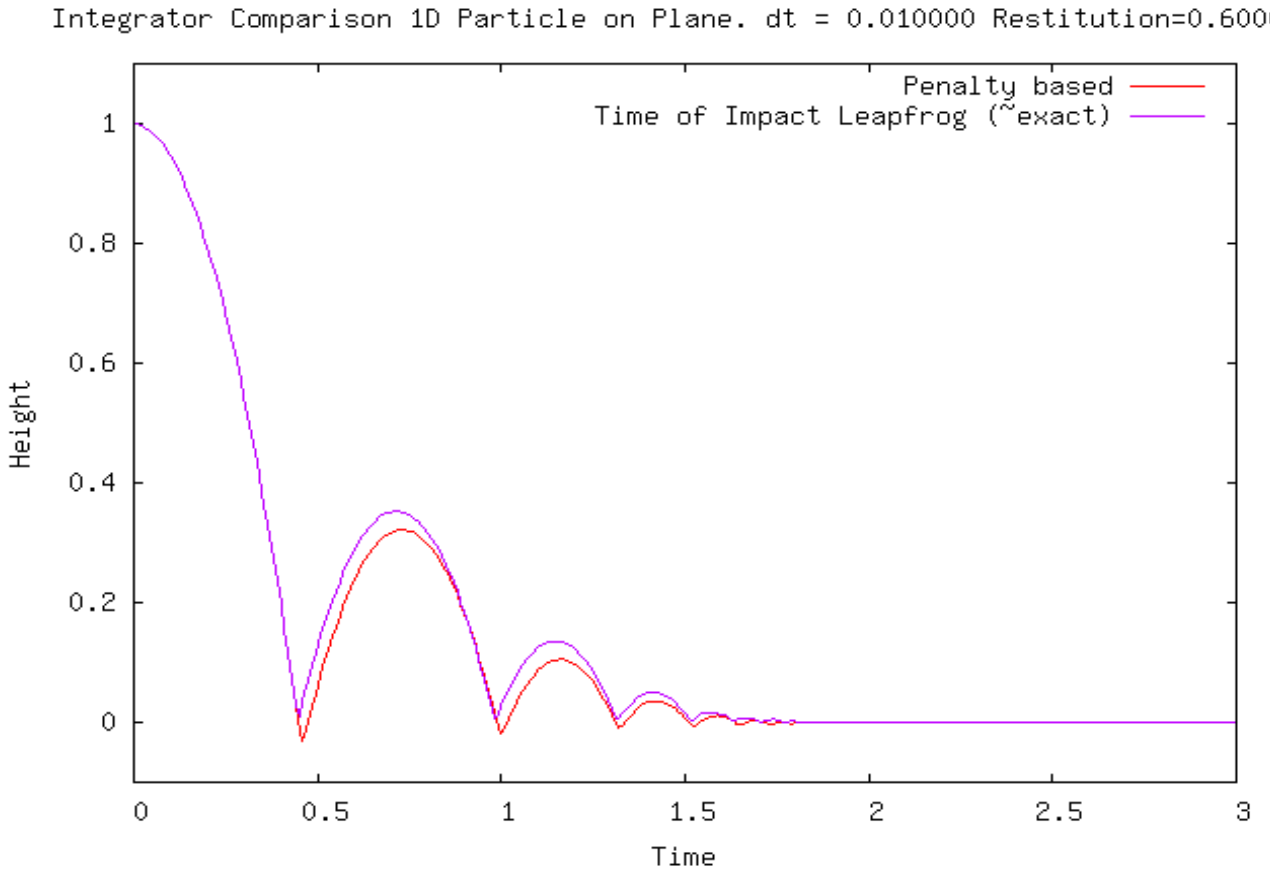
Problems:

- Extremely difficult to treat collisions and contacts the same way since velocities are so different. Large velocities require tiny timesteps.
- Always gives an overlap, unless spring constant is large – and then tiny timesteps are required.
- Damping parameter is pretty much arbitrary and usually not set to give a certain restitution, but rather just to make things stable...
- Velocities in the damping term are one halfstep off (wrong), but this is usually a marginal problem.

Generalizes to the many-body case, but only as a propagational method, and typically becomes very stiff, and also suffers from secondary oscillations and large overlaps. Notoriously difficult to choose the parameters!

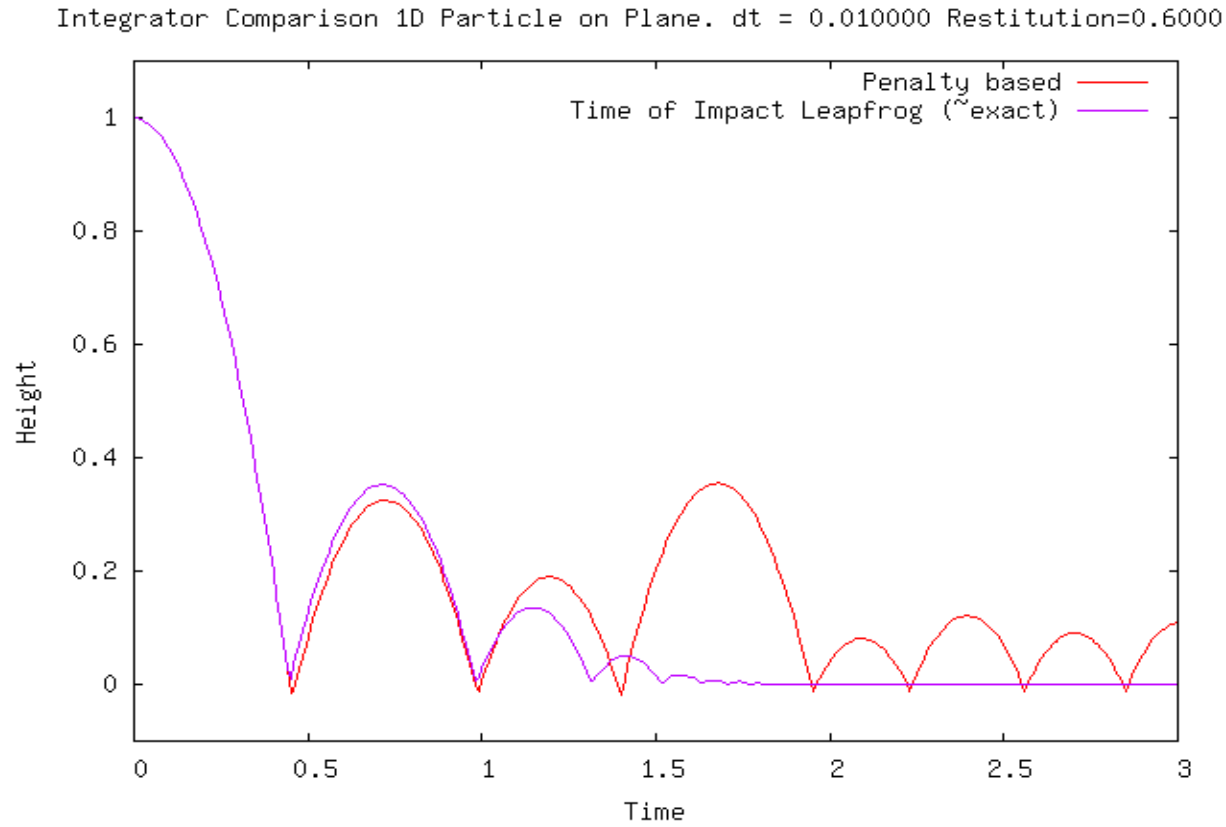
1D particle on a plane – ”Spring model”

Collisions and contacts treated with a penalizing spring model.



1D particle on a plane – ”Spring model”

With a slightly larger spring constant, we introduce large errors in the integration – that adds energy to the system. This will often cause a many-body system with connected springs to blow up!



1D particle on a plane – ”Stabilized model”

Treat collisions with impulses.

Treat contacts with local springs and dampers, AND exaggerate the mass a bit so that the resulting spring force becomes slightly weaker.

This corresponds to a regularized and stabilized method – SPOOK.

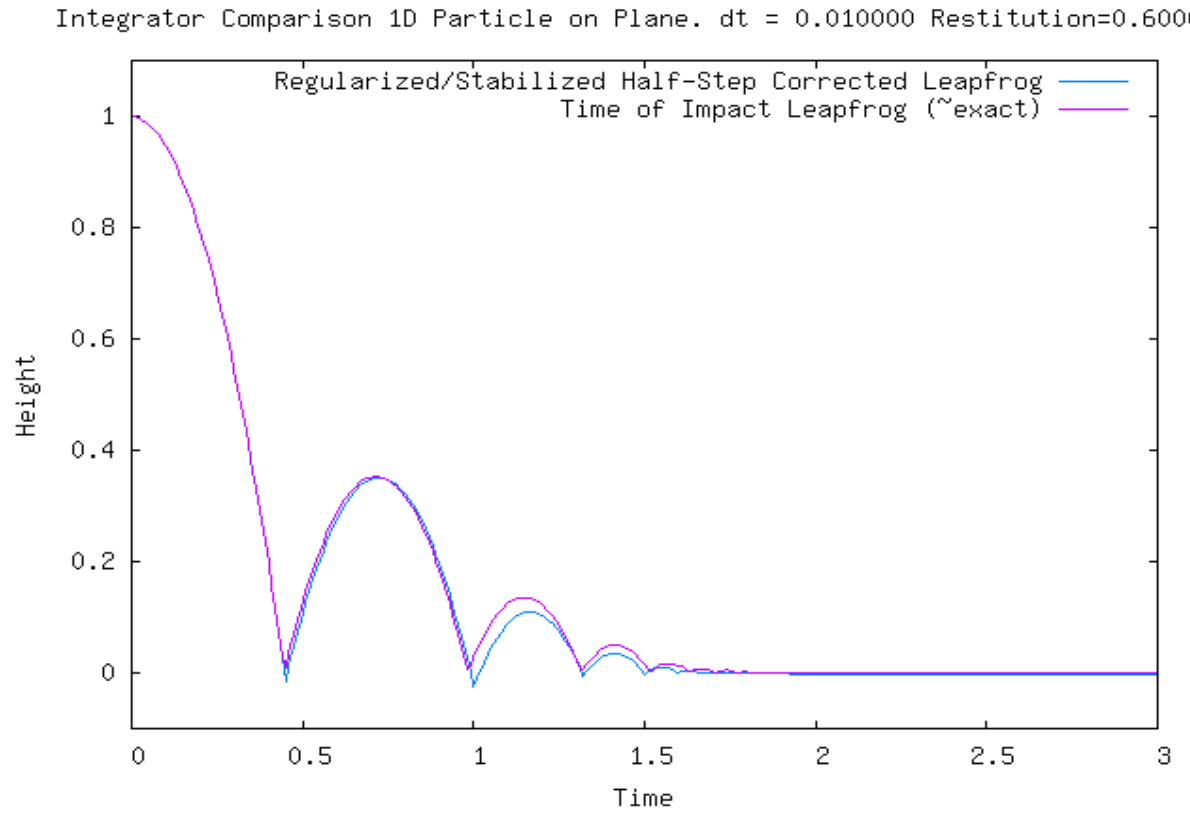
Spring-damper-effective mass parameters all have physical values!

More next lecture.

1D particle on a plane – ”Stabilized model”

Shows the stabilized/regularized model (SPOOK), using collision impulses for collisions, and the stabilization/regularization scheme for contacts.

This results in tiny overlaps, but also a very smooth and stable procedure.



1D particle on a plane – ”Time of impact model”

What about the ”exact” reference model then?

Whenever we detect a penetration, we compute the collision point in time and space, step back to this point, apply an impulse and, then step forward again to the time we were at, where we have a new velocity and can take another position step.

This means that we _move_ the particle inside the timestep.

```
v = v -g*dt;
x =x + v*dt;
if (x < 0)
  vprim = -e*v - 0.5*(1-e)*g*dt*(1-2*x/(v*dt));
  x = -e*x*(1 - g*dt/(2*v) + g*x/(v*v));
  v = vprim;
endif
```

1D particle on a plane – ”Time of impact model”

Strength:

- Unconditionally stable since it always conserves energy.
- Always balances the contact force (normal to gravity) perfectly since the time/position of impact is computed.
- Collisions and contacts are treated equivalently.

Problems:

- ...

Does NOT generalize well to the many-body case since moving around many contact points becomes a non-linear matrix problem in this case.

- In addition, the number of collision times/points grow exponentially with system size, and it doesn’t make sense to track the exact points!
- Even infinitecimal changes in initial conditions can change the trajectories dramatically.