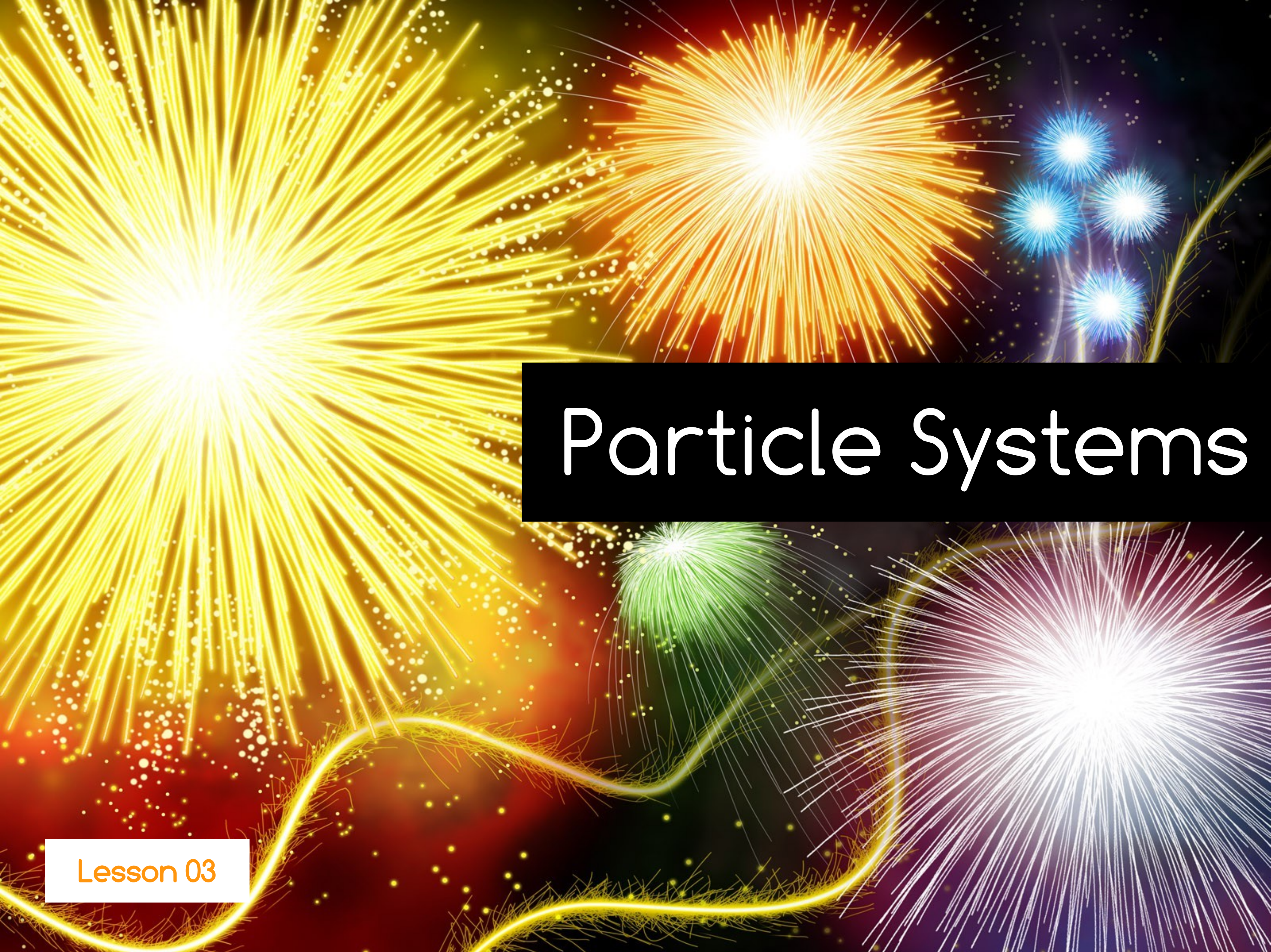


Particle Systems

Lesson 03

Lesson 03 Outline

- Newton dynamics of particles
- Ordinary differential equation (ODE) solver
- Particle - obstacle collision detection
- Practical design of particle system
- Demos / tools / libs

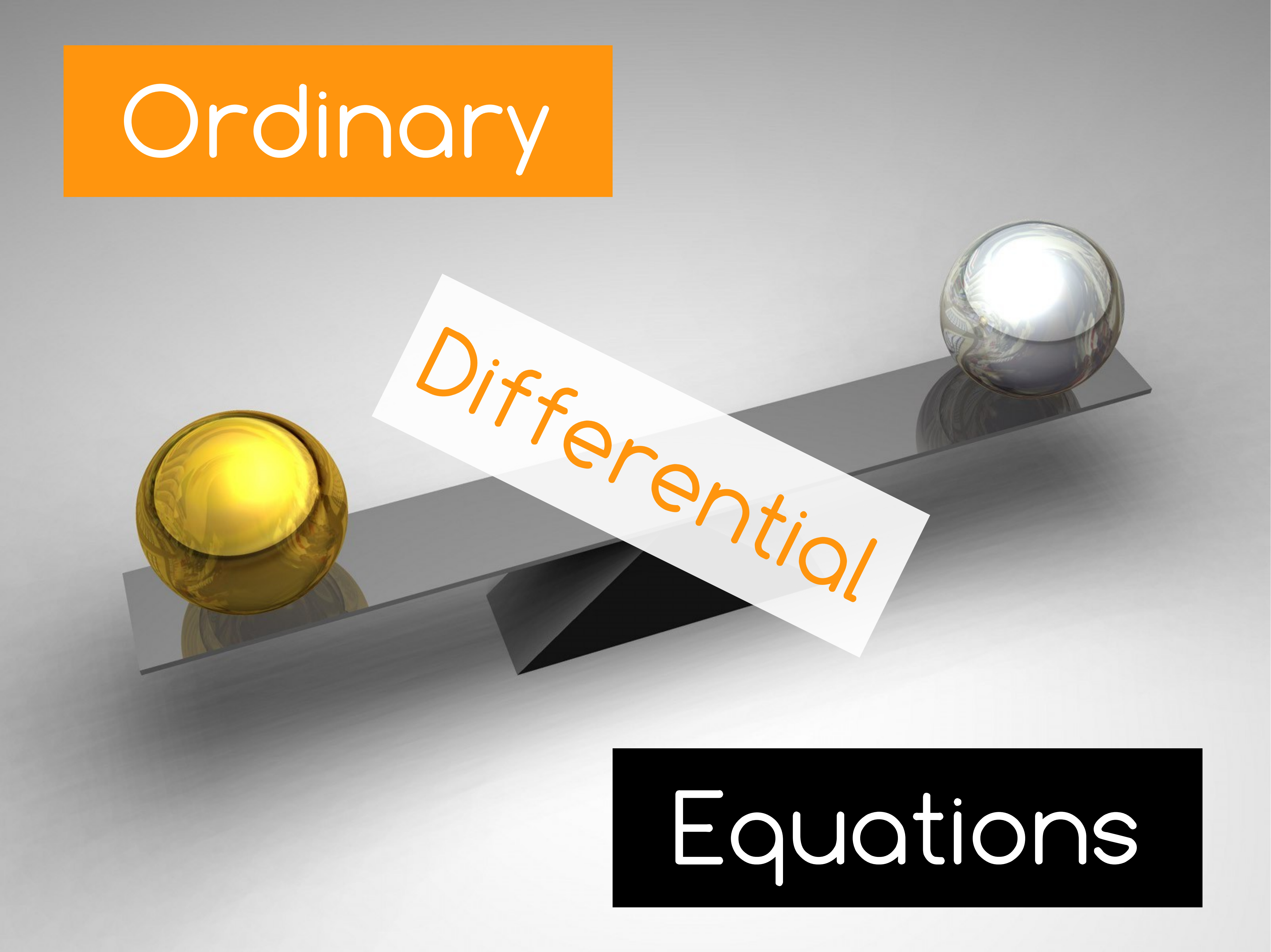
Newton Dynamics

Newton's Dynamics

- Three fundamental Newton's laws of motion
  - (1) Every body remains in a state of rest or uniform motion (constant velocity) unless it is acted upon by an external unbalanced force.
  - (2) A body of mass **m** subject to a force **f** undergoes an acceleration a that has the same direction as the force and a magnitude that is directly proportional to the force and inversely proportional to the mass: **f = ma.**
  - (3) The mutual forces of action and reaction between two bodies are equal, opposite and collinear.

Particle Dynamics

- Dynamical proprties of Particles
  - Mass (m) in [kg]: parameter
  - Position (p) in [m]: dp = v
  - Velocity (v) in [m/s]: dv = a
  - Momentum (L) in [kgm/s]: L=mv
  - Acceleration (a) in (m/s^2): $a = m^{-1}F$; gravity, wind, user...
  - Force (F) in $[kgm/s^2]$: F = ma = dL
- The equation of unconstrained motion (ODE)
  - d(p,v) = (v,a)

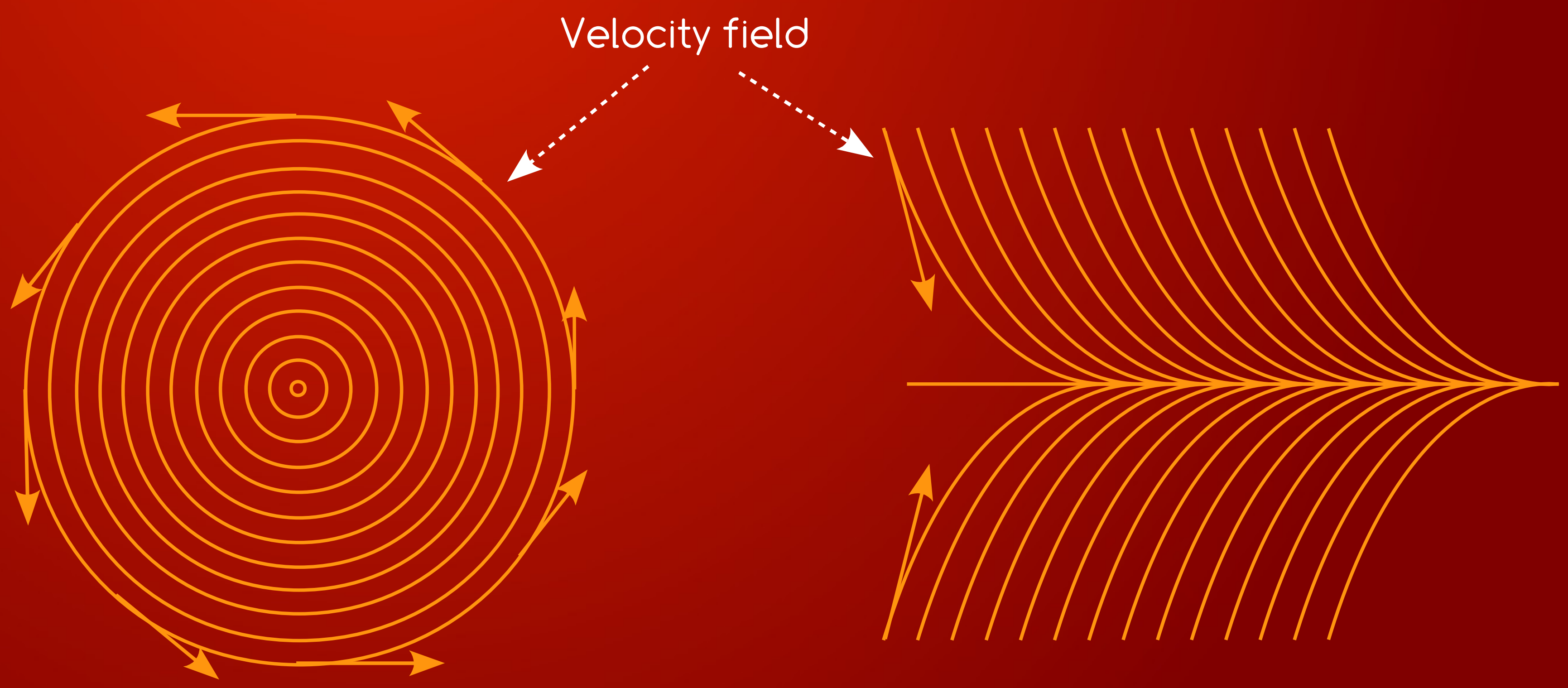

Ordinary
Differential
Equations

Ordinary Differential Equations

- **Definition:** An ordinary differential equation (ODE) is a relation that contains functions of only one independent variable, and one or more of their derivatives with respect to that variable.
- Problem: How to evaluate (in time) position $\boldsymbol{\rho(t)}$ of a particle, when we only know its change in time $\boldsymbol{\rho'(t)}$ is a function of position and time: $\rho'(t)=F(\rho(t),t)$
- Examples
  - $\rho'(t) = -10\rho(t)$
  - $\rho'(t) = t^2\rho(t) - 3\rho^2(t) + 7$
- Objective: Given function $F(\rho,t)$ and the value $\rho(t)$ at some time t, we can compute $\rho'(t) = F(\rho(t),t)$

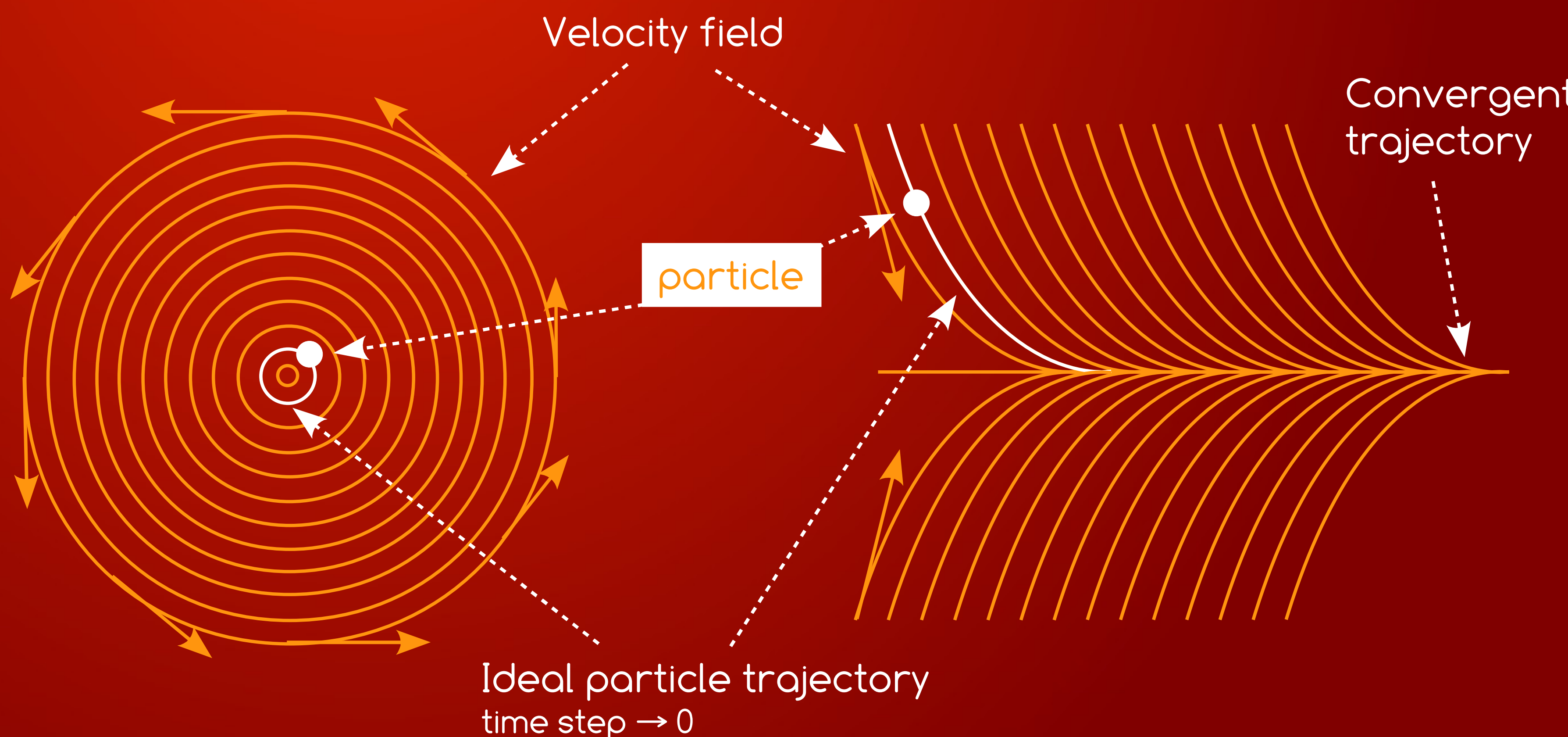
ODE - Numerical problems

- Inaccuracy Problem
- Instability Problem



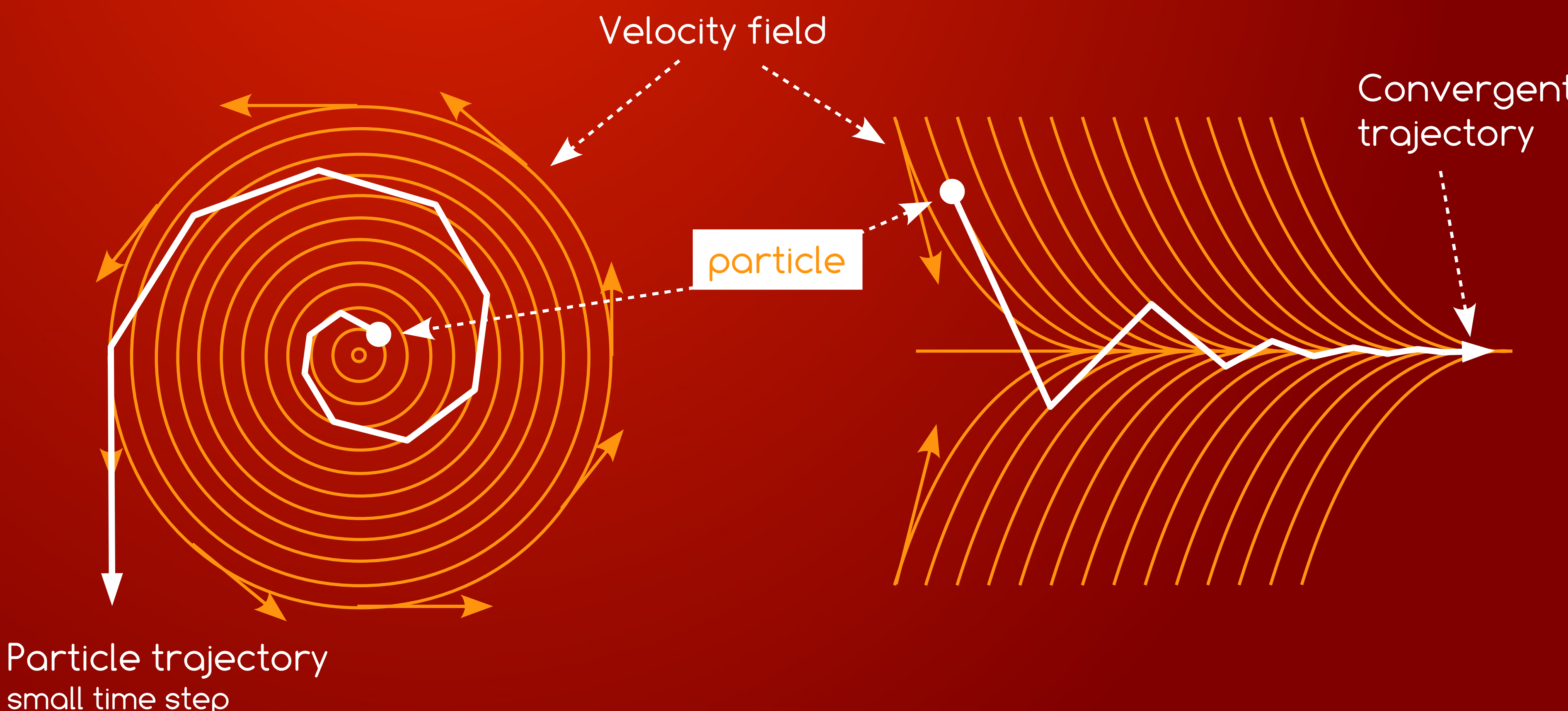
ODE - Numerical problems

- Inaccuracy Problem
- Instability Problem



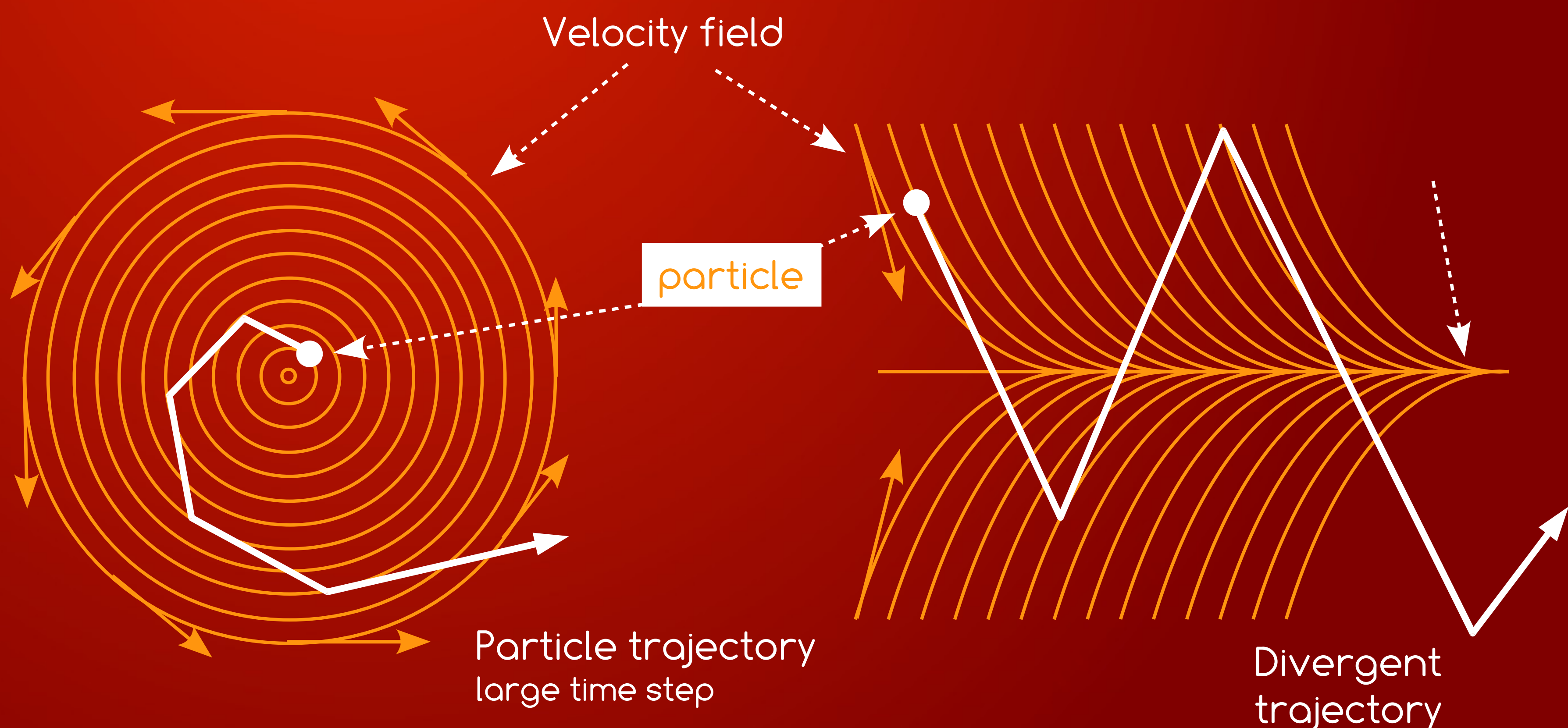
ODE - Numerical problems

- Inaccuracy Problem
- Instability Problem



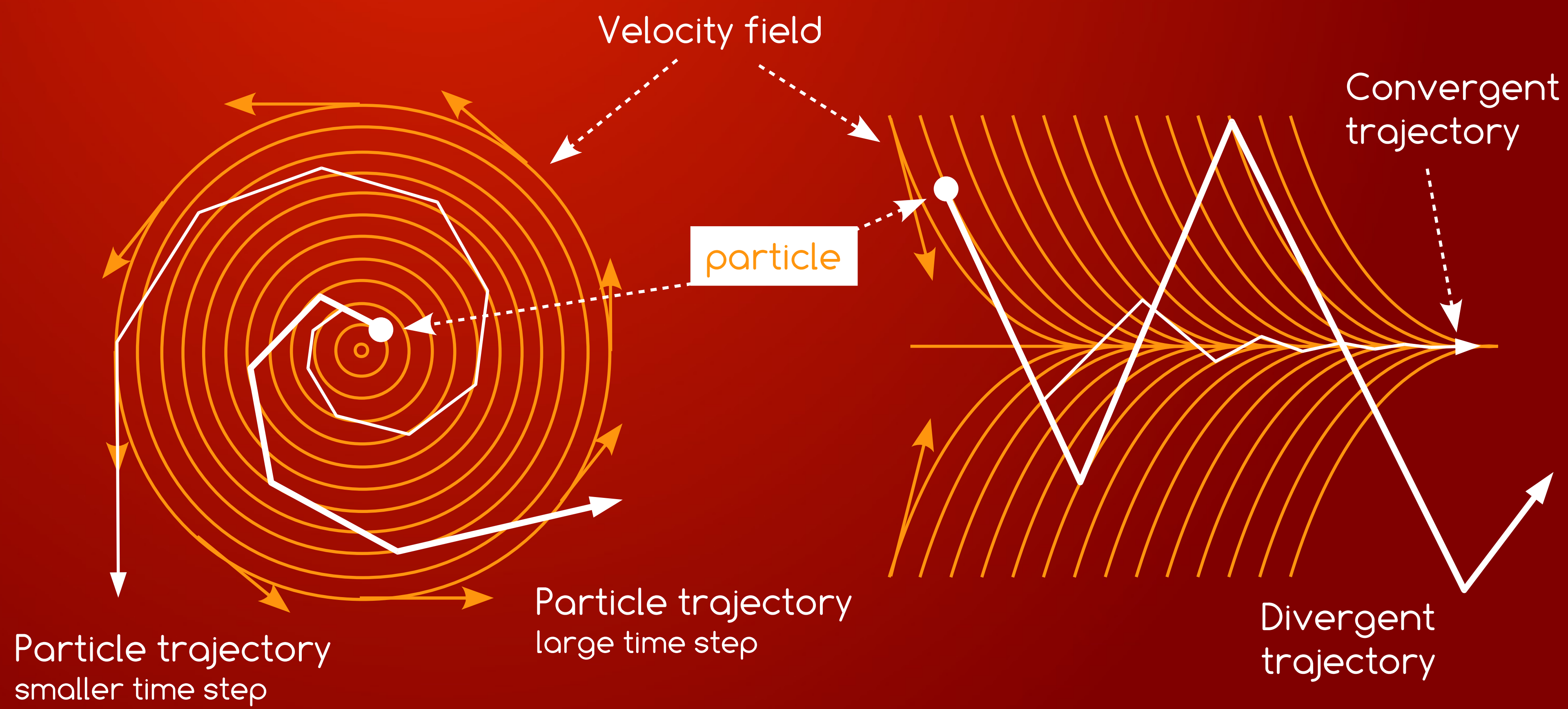
ODE - Numerical problems

- Inaccuracy Problem
- Instability Problem



ODE - Numerical problems

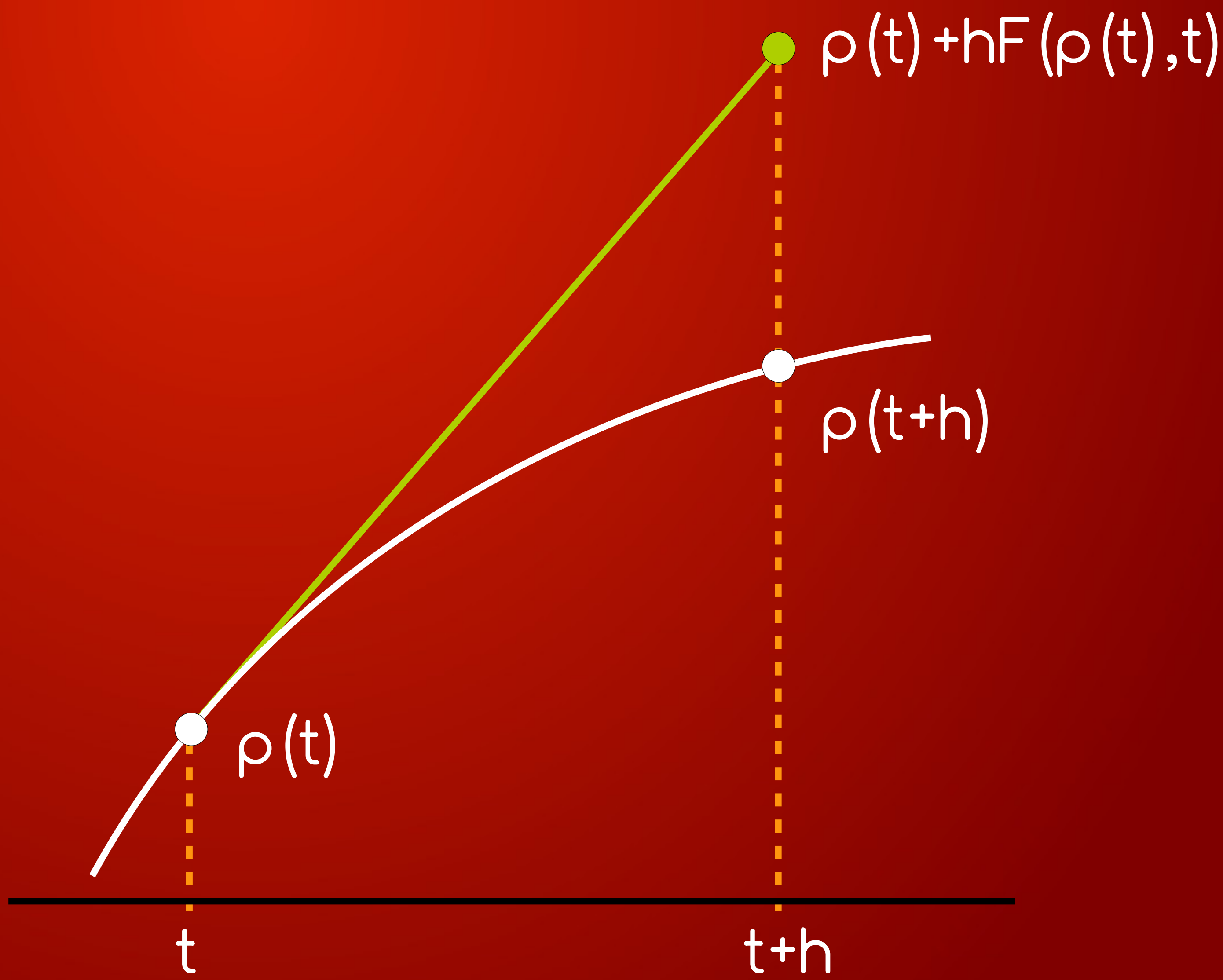
- Inaccuracy Problem
- Instability Problem



ODE Solvers

- Explicit Schemes
  - Euler
  - Mid Point
  - Runge Kutta 4
  - Verlet
- Implicit Schemes
  - Implicit Euler

Explicit Euler Scheme



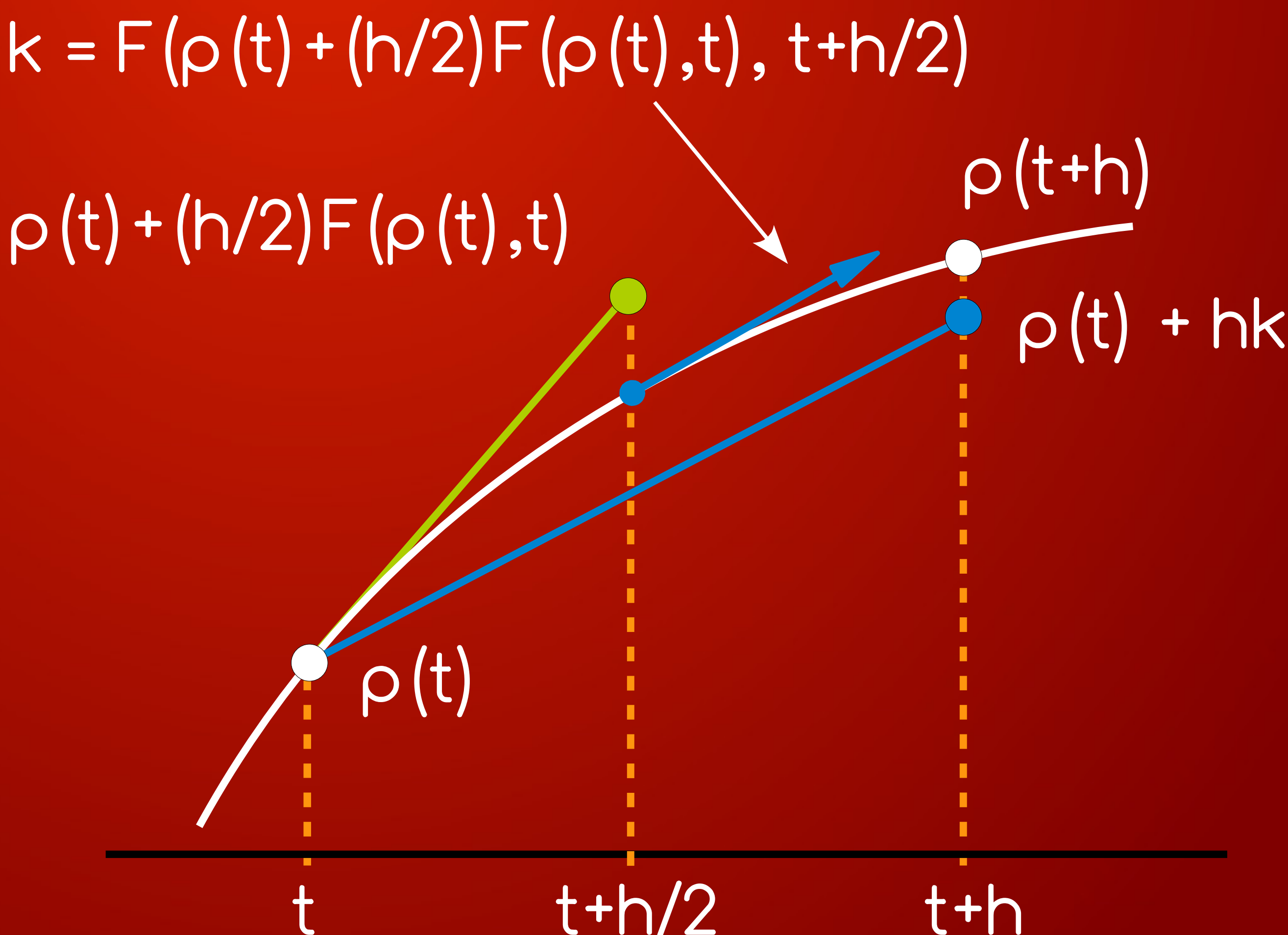
Explicit Euler Scheme

- **Idea:** Given initial value $\rho(t_0)$ of function ρ at time t_0 we can find $\rho(t_0+h)$ using Taylor expansion as
- $\rho(t_0+h) = \rho(t_0) + h\rho'(t_0) + O(h^2)$; $\rho'(t_0) = F(\rho(t_0), t_0)$
- **Numerical algorithm:**
- $\rho_{n+1} = \rho_n + h^*F(\rho_n, t_n)$ where ρ_0 = some initial value
- Pros / Cons:
  - Very simple, fast and easy to implement
  - Huge error = $O(h^2)$
  - Can be unstable – cumulated error increases to infinity

Explicit Midpoint Scheme

- **Idea:** Use approximate derivative $\rho'(t+h/2)$ of $\rho(t)$ at time $t+h/2$ instead of the the simple $\rho'(t)$
- $\rho(t+h) = \rho(t) + hF(\rho(t+h/2), t+h/2) + O(h^3)$
- **Problem:** We do not know function $\rho(t)$ or its derivative at time $t+h/2$.
- Knowing $\rho'(t+h/2) = F(\rho(t+h/2), t+h/2)$ we need to estimate only $\rho(t+h/2)$
- **Solution:** Estimate it using Taylor expansion
- $\rho(t+h/2) = \rho(t) + (h/2)\rho'(t) + O(h^2)$
- **Finally:**
- $\rho(t+h) = \rho(t) + hF(\rho(t)+(h/2)\rho'(t), t+h/2) + O(h^3)$

Explicit Midpoint Scheme



Explicit Midpoint Scheme

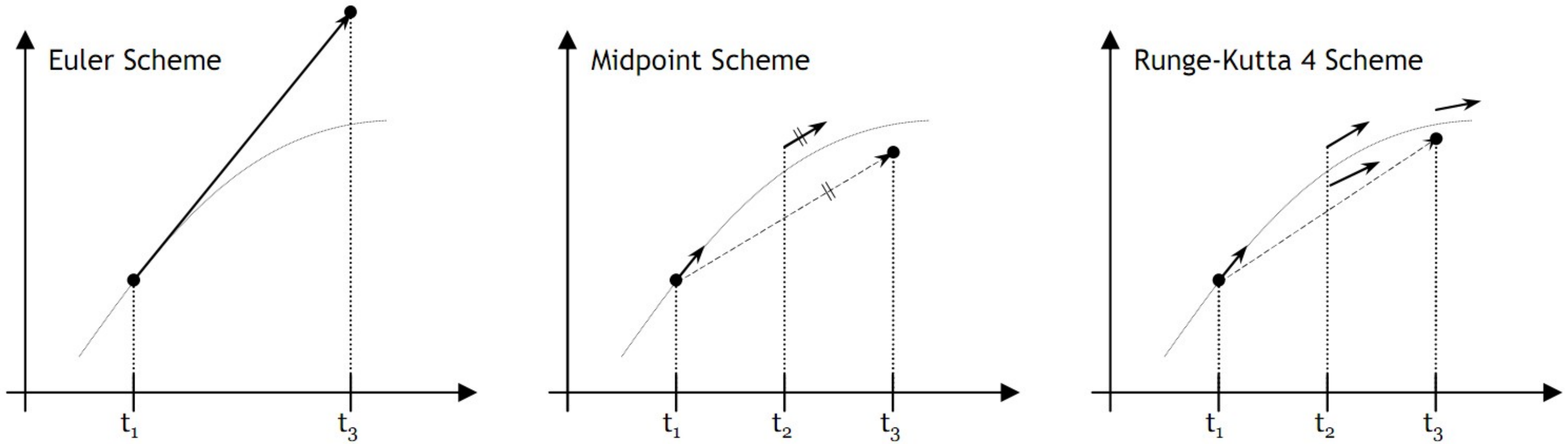
- Numerical algorithm:
- $\rho_{n+1} = \rho_n + hF(\rho_n + (h/2)F(\rho_n, t_n), t_n + h/2)$
- Pros / cons
  - Very simple, fast and easy to implement
  - Smaller error = $O(h^3)$
  - Need to evaluate F two times - more computation

Runge-Kutta Scheme

- Numerical algorithm

$$
\begin{aligned}
k_1 &= hF(\mathbf{p}(t_0), t_0) \\
k_2 &= hF(\mathbf{p}(t_0) + \frac{k_1}{2}, t_0 + \frac{h}{2}) \\
k_3 &= hF(\mathbf{p}(t_0) + \frac{k_2}{2}, t_0 + \frac{h}{2}) \\
k_4 &= hF(\mathbf{p}(t_0) + k_{3,} t_0 + h) \\
\mathbf{p}(t_0 + h) &= \mathbf{p}(t_0) + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)
\end{aligned}
$$

Explicit Integration schemes



Verlet Scheme

- **Preconditions:** Equations are pure 2-order ODEs.
- **Idea:** Taylor expand p(t) at p(t+h) and p(t-h) and subtract / add equations

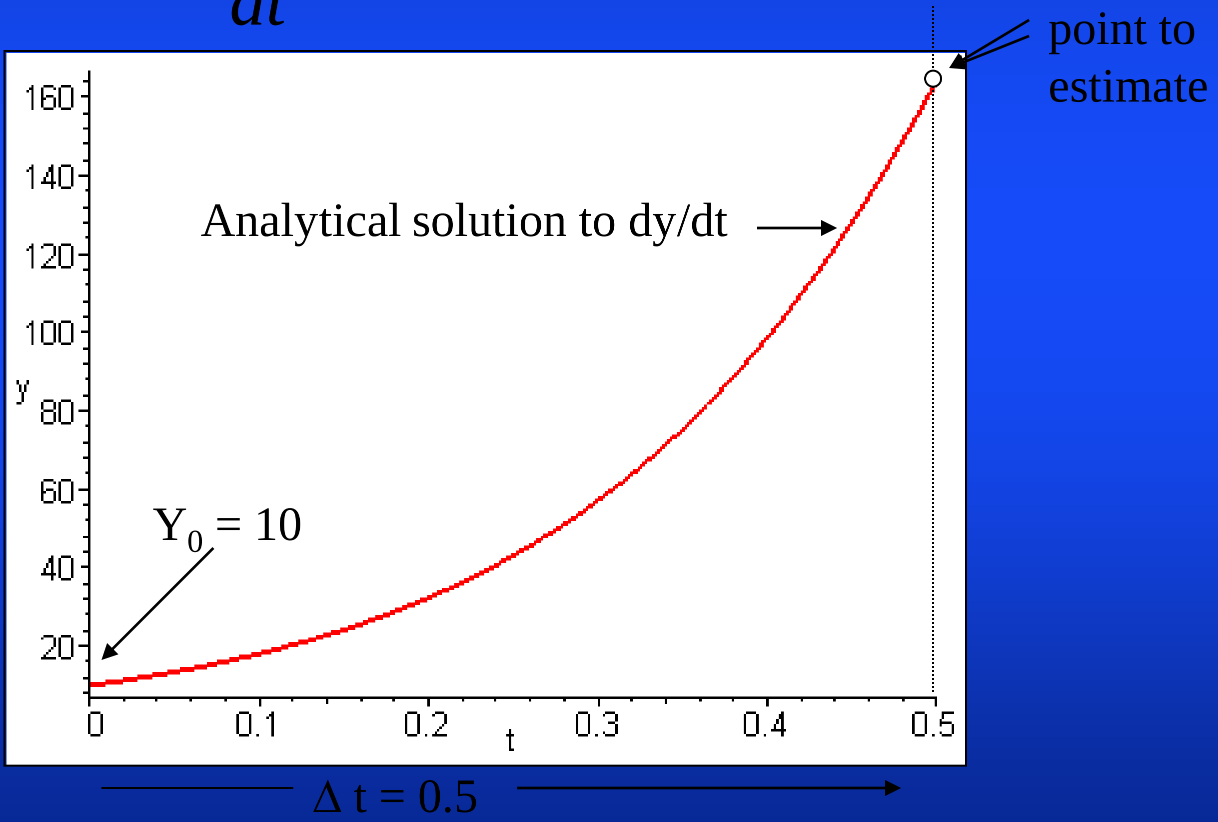
$$
\begin{aligned}
\mathbf{p}(t+h) &= \mathbf{p}(t) + h\,\dot{\mathbf{p}}(t) + \frac{h^2}{2}\ddot{\mathbf{p}}(t) + \frac{h^3}{6}\dddot{\mathbf{p}}(t) + O(h^4) \\
\mathbf{p}(t-h) &= \mathbf{p}(t) - h\,\dot{\mathbf{p}}(t) + \frac{h^2}{2}\ddot{\mathbf{p}}(t) - \frac{h^3}{6}\dddot{\mathbf{p}}(t) + O(h^4) \\
\mathbf{p}(t+h) &= 2\,\mathbf{p}(t) - \cancel{h}\,\mathbf{p}(t-h) + h^2\ddot{\mathbf{p}}(t) + O(h^4) \\
\dot{\mathbf{p}}(t+h) &= \frac{1}{2\mathrm{h}}\mathbf{p}(t+h) - \frac{1}{2\mathrm{h}}\mathbf{p}(t-h) + O(h^2)
\end{aligned}
$$

Implicit Euler

- **Explicit Euler:** $\rho(t+h) = \rho(t) + hF(\rho(t),t) + O(h^2)$
- **Implicit Euler:** $\rho(t+h) = \rho(t) + hF(\rho(t+h),t+h) + O(h^2)$
- **Problem:** We need to solve for $\rho(t+h)$
- **Solution:** Taylor expand $F(\rho,t)$ in ρ
- $F(\rho+\Delta\rho,t) = F(\rho,t) + \Delta\rho F'(\rho,t) + O(\Delta\rho^2)$
- **Set:** $\Delta\rho$ as $hF(\rho+\Delta\rho,t)$
- $F(\rho+\Delta\rho,t) = F(\rho,t) + hF(\rho+\Delta\rho,t)F'(\rho,t) + O(\Delta\rho^2)$
- $F(\rho+\Delta\rho,t) = (1 - hF'(\rho,t))^{-1}F(\rho,t) + O(\Delta\rho^2)$
- **Problem:** $F'(\rho,t)$ (Jacobian) must be known
- More on cloth modeling

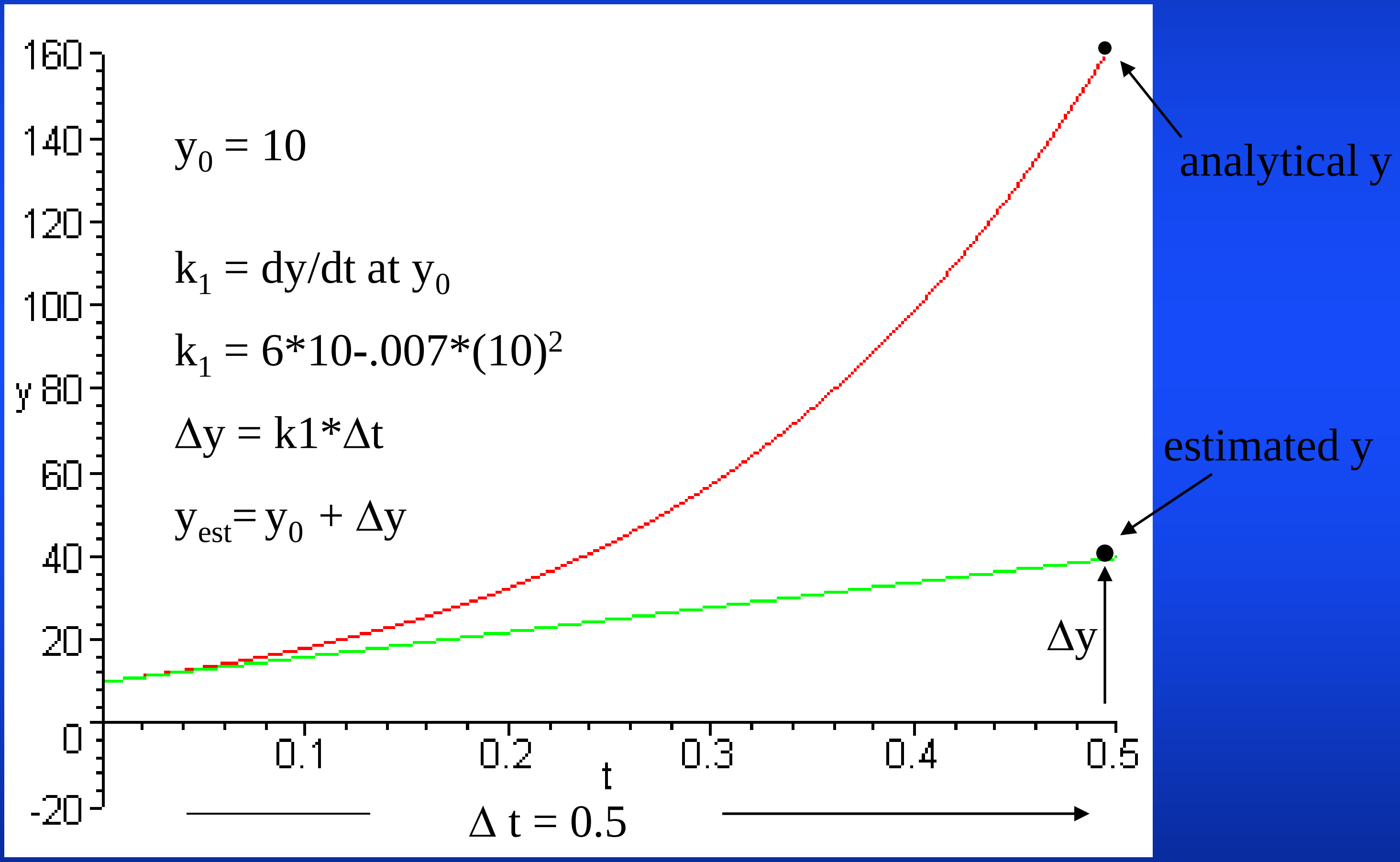
Example

$$\frac{dy}{dt} = 6y - .007y^2$$

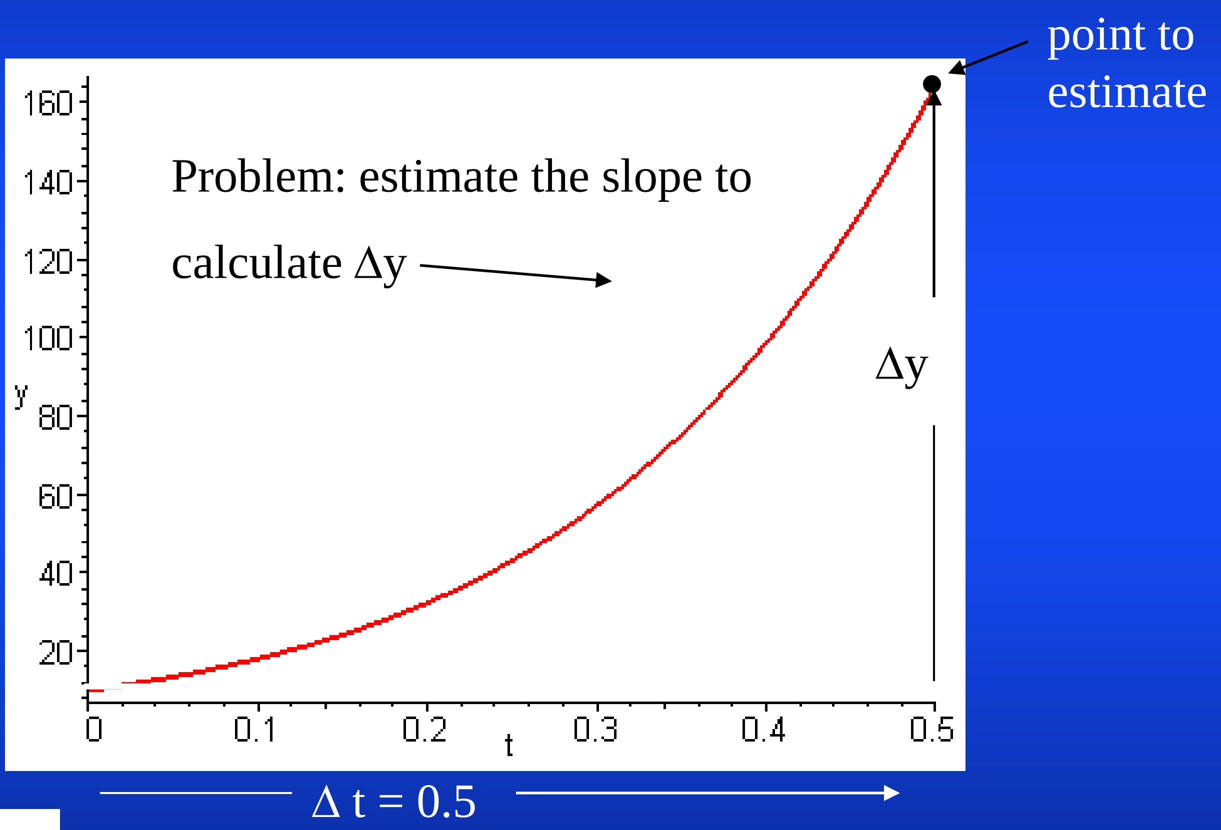


Euler (pronounced "oiler")

y



Runge-Kutta (pronounced Run-gah Kut-tah)



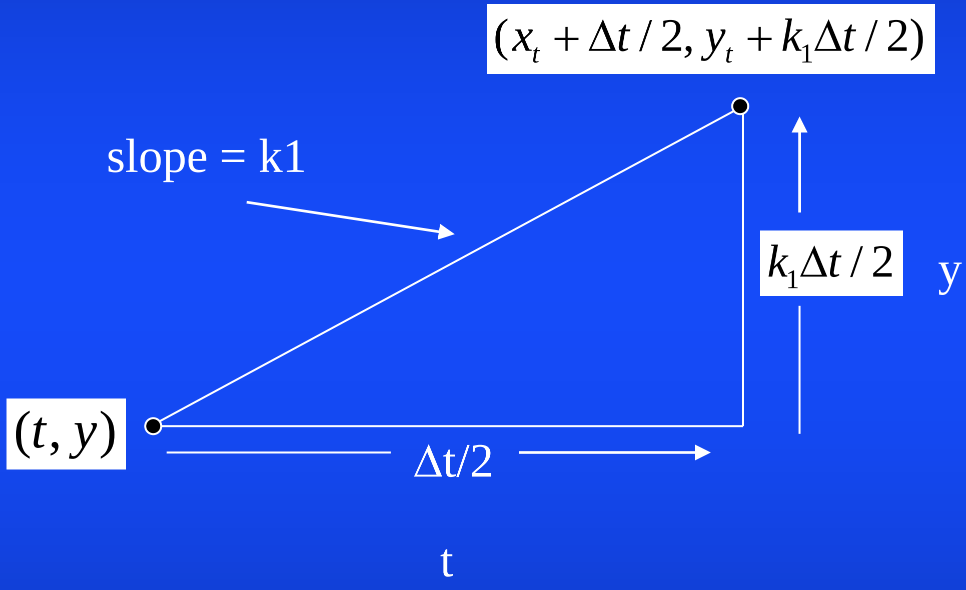
$$\frac{dy}{dt} = 6y - .007y^2$$

Runge-Kutta (4th order)

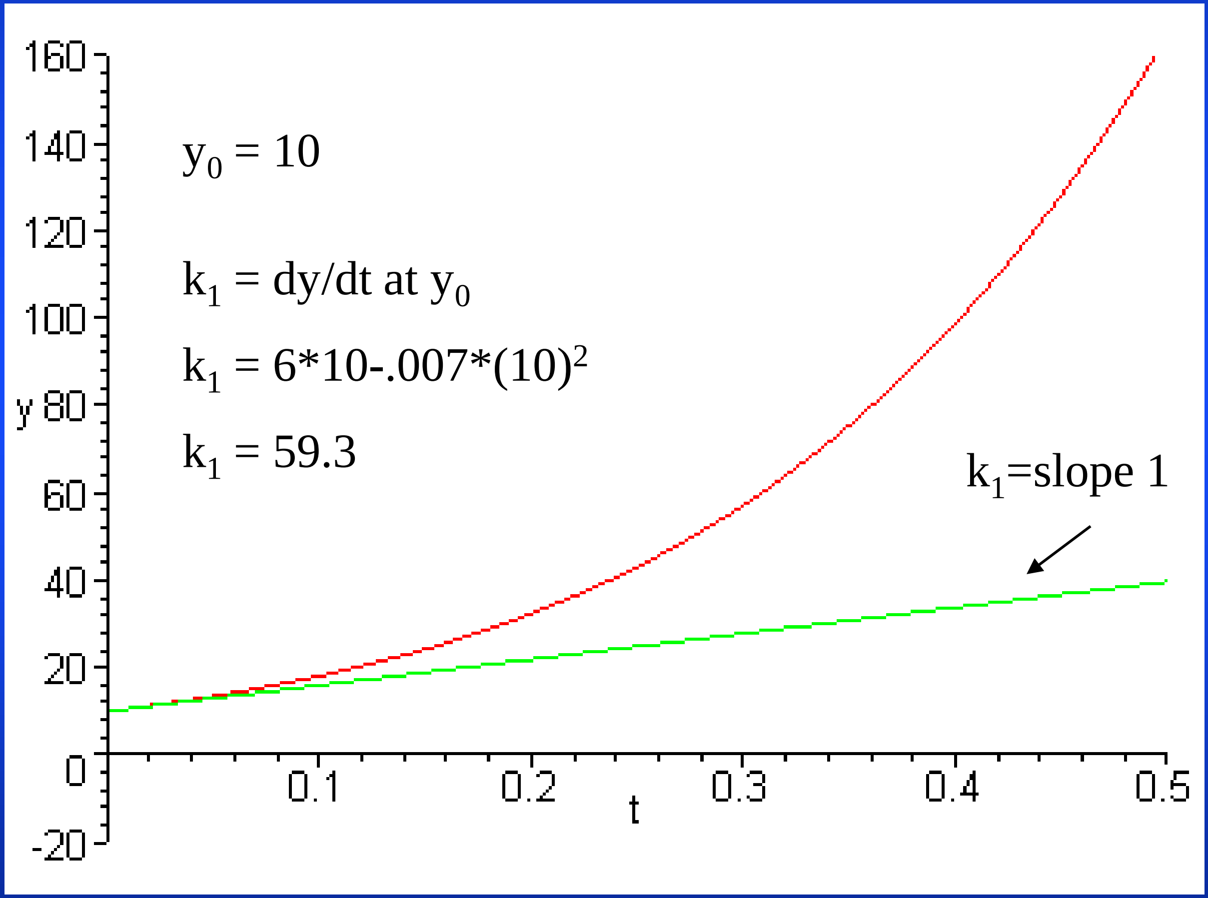
$$f'(t, y) = derivative\ at\ (t, y)$$

$$k_1 = f'(t, y)$$
$$k_2 = f'(t + \Delta t/2, y + k_1 \Delta t/2)$$
$$k_3 = f'(t + \Delta t/2, y + k_2 \Delta t/2)$$
$$k_4 = f'(t + \Delta t, y + k_3 \Delta t)$$

$$y_{t+\Delta} = y_t + \Delta t \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

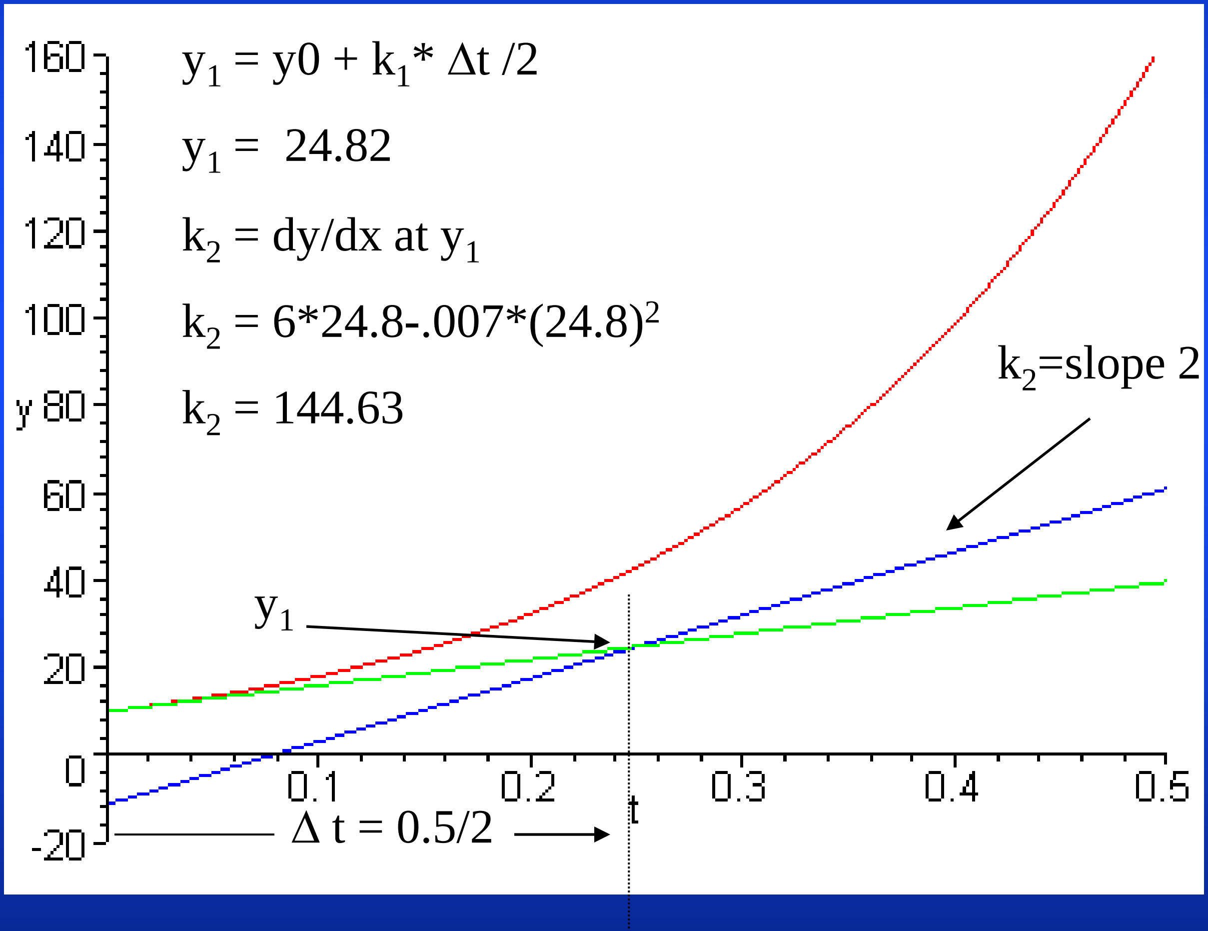


Step 1: Evaluate slope at current value of state variable.



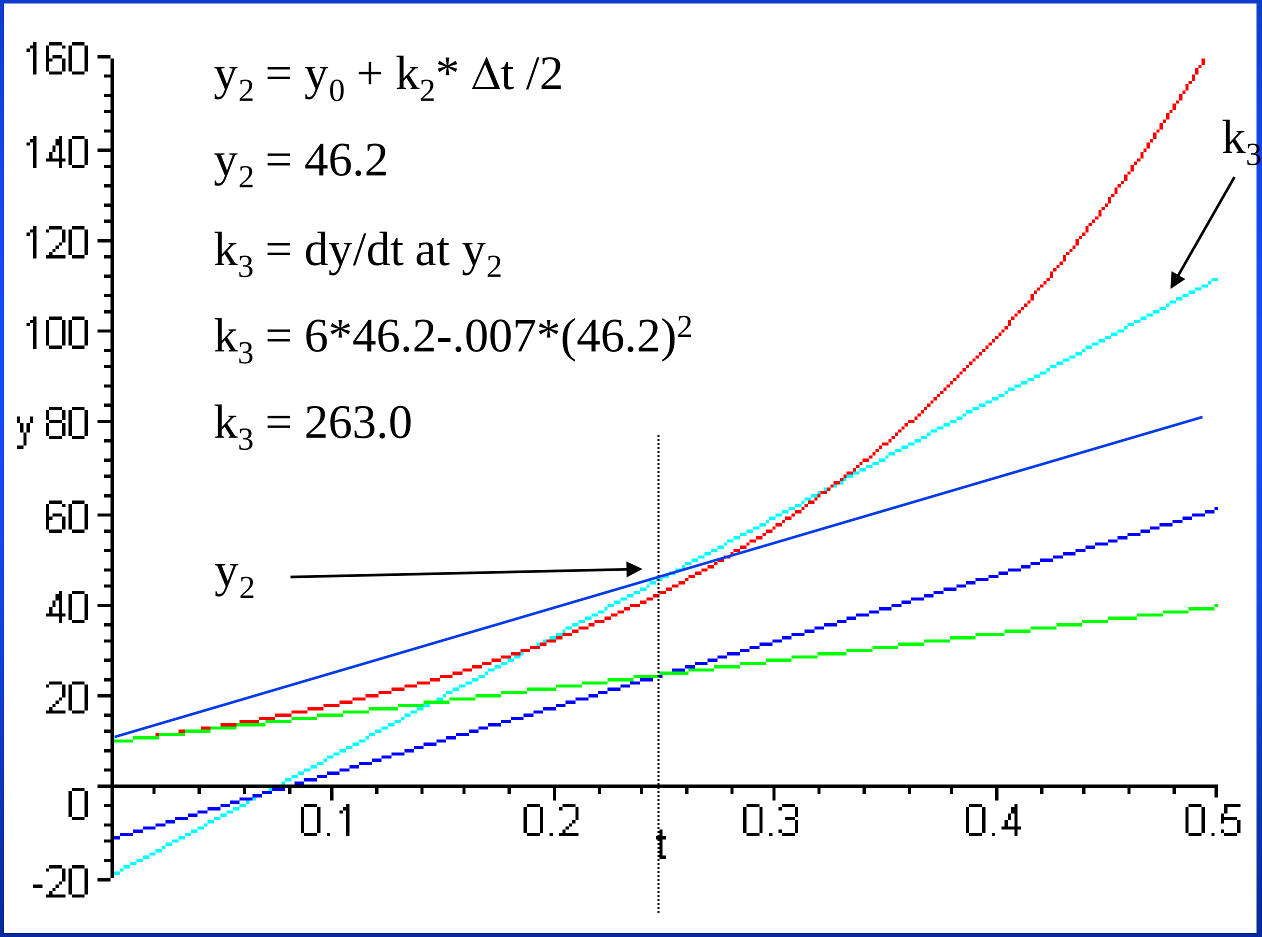
Step 2: Calculate y_1 at $t + \Delta t/2$ using k_1.

Evaluate slope at y_1.



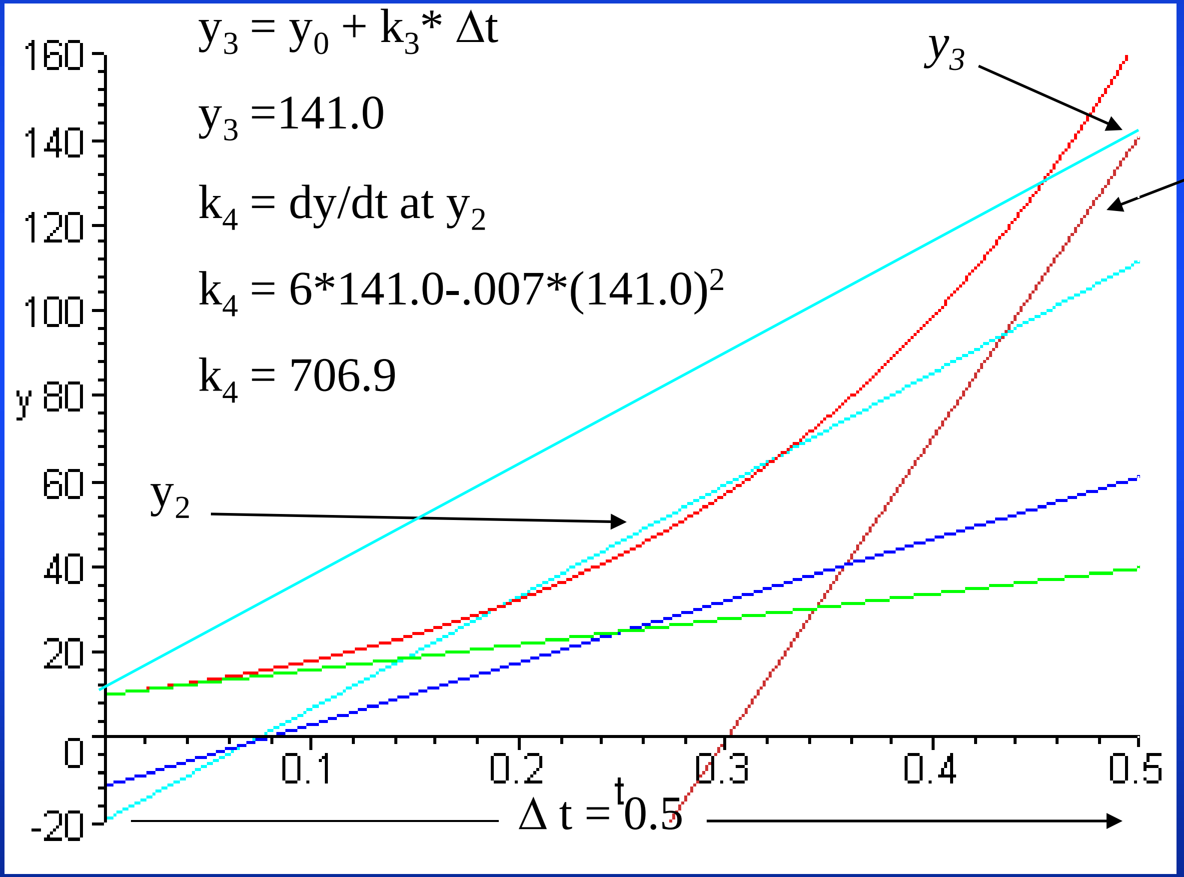
Step 3: Calculate y_2 at $t + \Delta t/2$ using k_2.

Evaluate slope at y_2.



Step 4: Calculate y_3 at $t + \Delta t$ using k_3.

Evaluate slope at y_3.

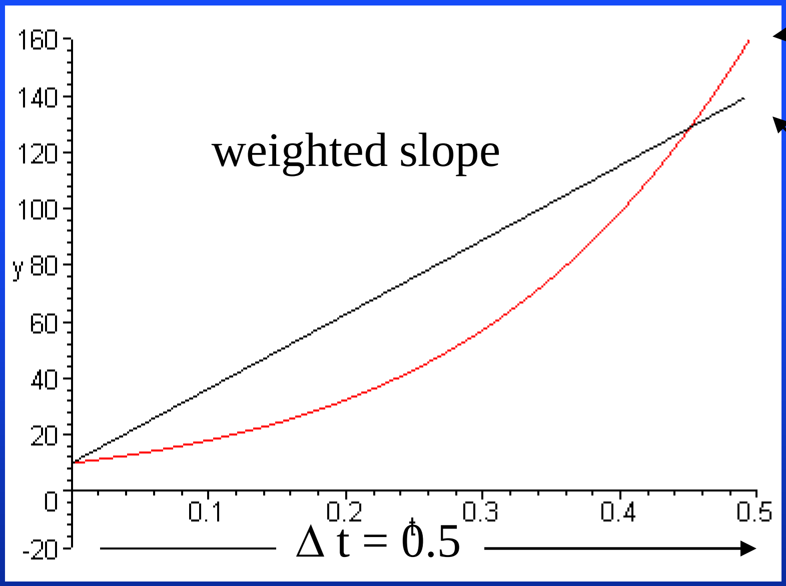


Step 5: Calculate weighted slope.

Use weighted slope to estimate *y* at *t* +Δt

weighted slope = $\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$Y_{t+\Delta} = Y_t + \Delta t \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

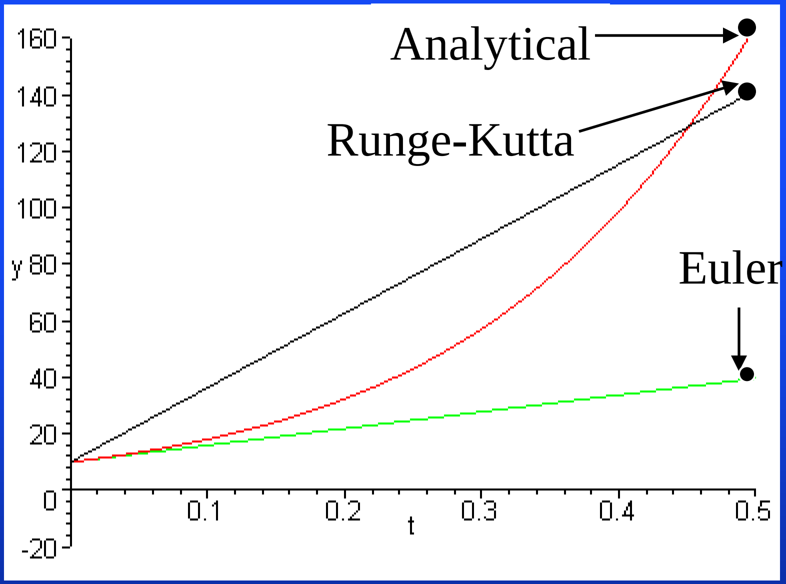


true value

estimated value

Conclusions

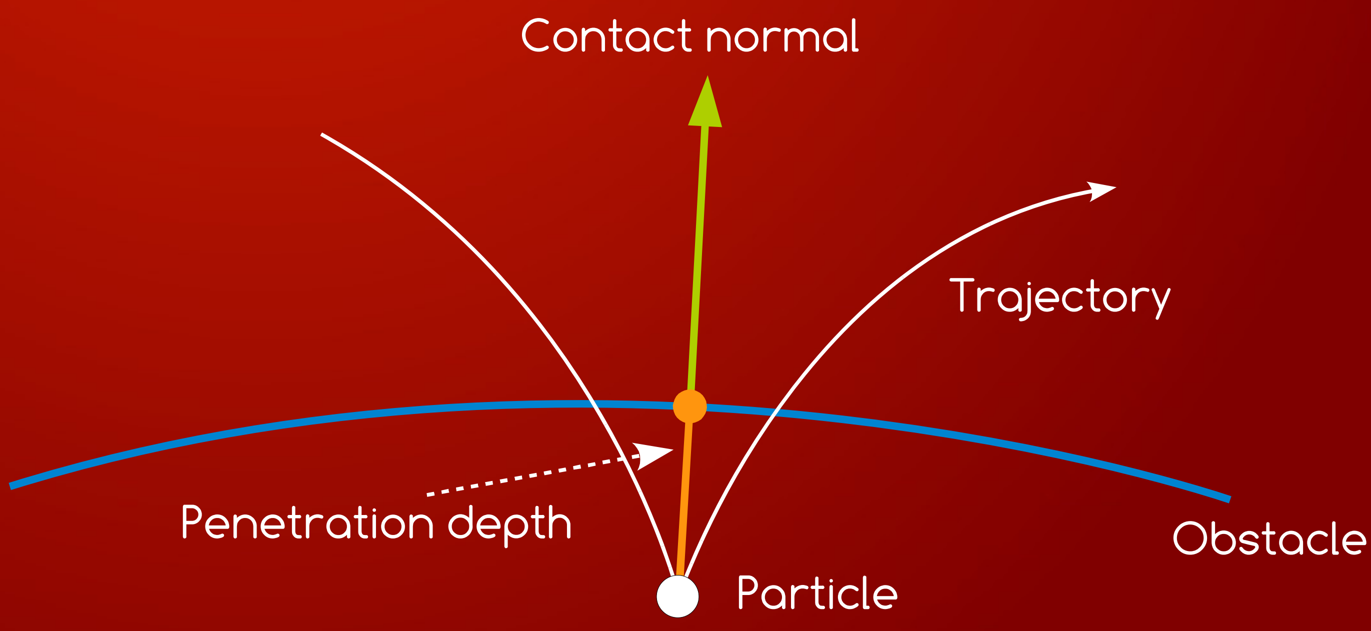
- ***4th order Runge-Kutta offers substantial improvement over Eulers.***
- ***Both techniques provide estimates, not "true" values.***
- ***The accuracy of the estimate depends on the size of the step used in the algorithm.***




Particle Obstacle
Collision Detection

Collision Scenario

- Particle-obstacle contact info
  - **Penetration depth (d):** minimal distance to separate particle from obstacle
  - **Contact normal (n):** direction vector along which we can get particle out of obstacle (by moving about penetration depth)



Newton's Impact Model

$$u_n(t^+) = -e_n u_n(t^-)$$

- Pre-collision relative normal velocity: $u_n(t^-)$
- Post-collision relative normal velocity: $u_n(t^+)$
- Coefficient of restitution: $0 <= e_n <= 1$
- Plastic collisions: $e_n == 0$
- Elastic collisions: $e_n == 1$

Impulse based Collision Resolution

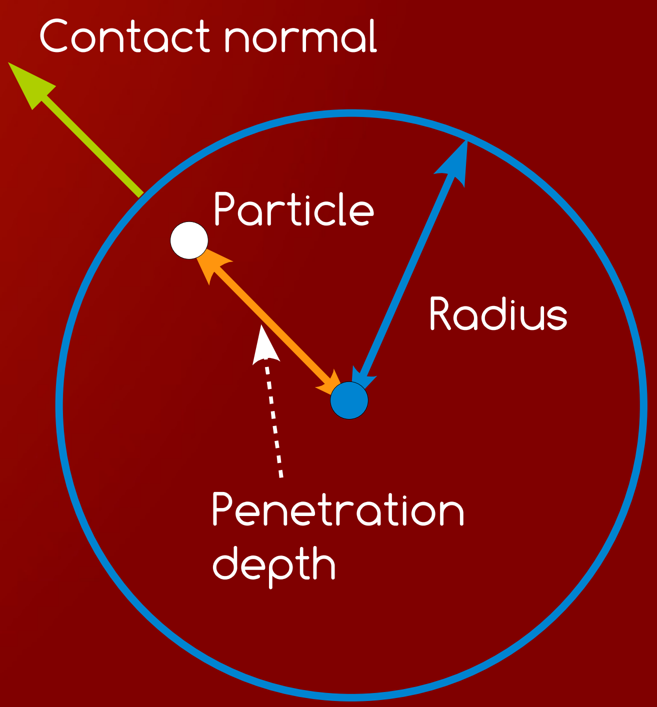
- Collision Impulse: Time integral of repulsive forces acting on bodies during collision

$$\mathbf{j}(t) = \int_t^{t+h} \mathbf{f}(a)\,da$$

- Impulses cause direct change of velocity: $\Delta\mathbf{u} = M^{-1}\mathbf{j}$
- $\Delta\mathbf{u} = \Delta\mathbf{u}_1 - \Delta\mathbf{u}_2 = M_1^{-1}\mathbf{j} - M_2^{-1}\mathbf{j} = (M_1^{-1} - M_2^{-1})\,\mathbf{j} = K\mathbf{j}$
- $\Delta\mathbf{u}_n = \mathbf{n}^T K\mathbf{j} = \mathbf{n}^T\mathbf{u}(t+h) - \mathbf{n}^T\mathrm{u}(t) = -e_n\mathbf{n}^T\mathbf{u}(t) - \mathbf{n}^T\mathrm{u}(t) = -(1+e_n)\mathbf{n}^T\mathrm{u}(t)$
- $j = -(1+e_n)\mathbf{n}^T\mathrm{u}(t) \,/\, \mathbf{n}^T(M_1^{-1} - M_2^{-1})\mathbf{n}$
- $\mathbf{u}_1$ += $\mathbf{jn}$; $\mathbf{u}_2$ -= $\mathbf{jn}$;

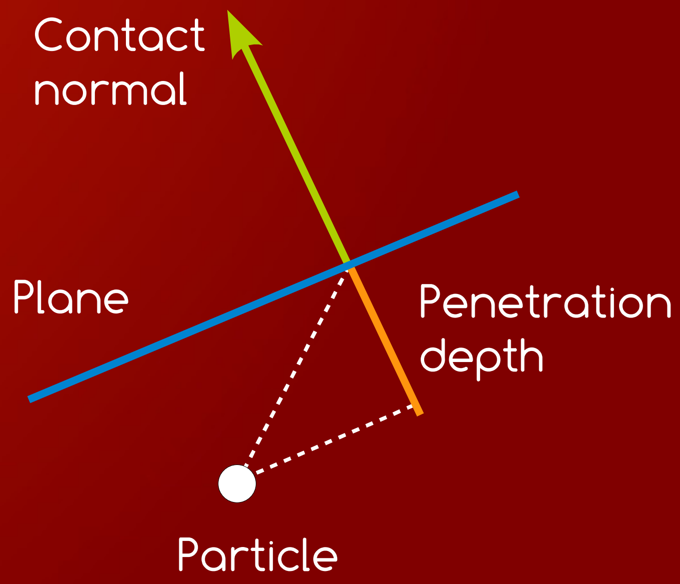
Particle – Sphere Collisions

- Particle - Sphere Model
  - Particle position: $p=(x,y,z)$
  - Sphere Center: $c=(x,y,z)$
  - Sphere Radius: r
- Penetration depth: $d = |p - c| - r$
- Contact normal: $n = norm(p - c)$

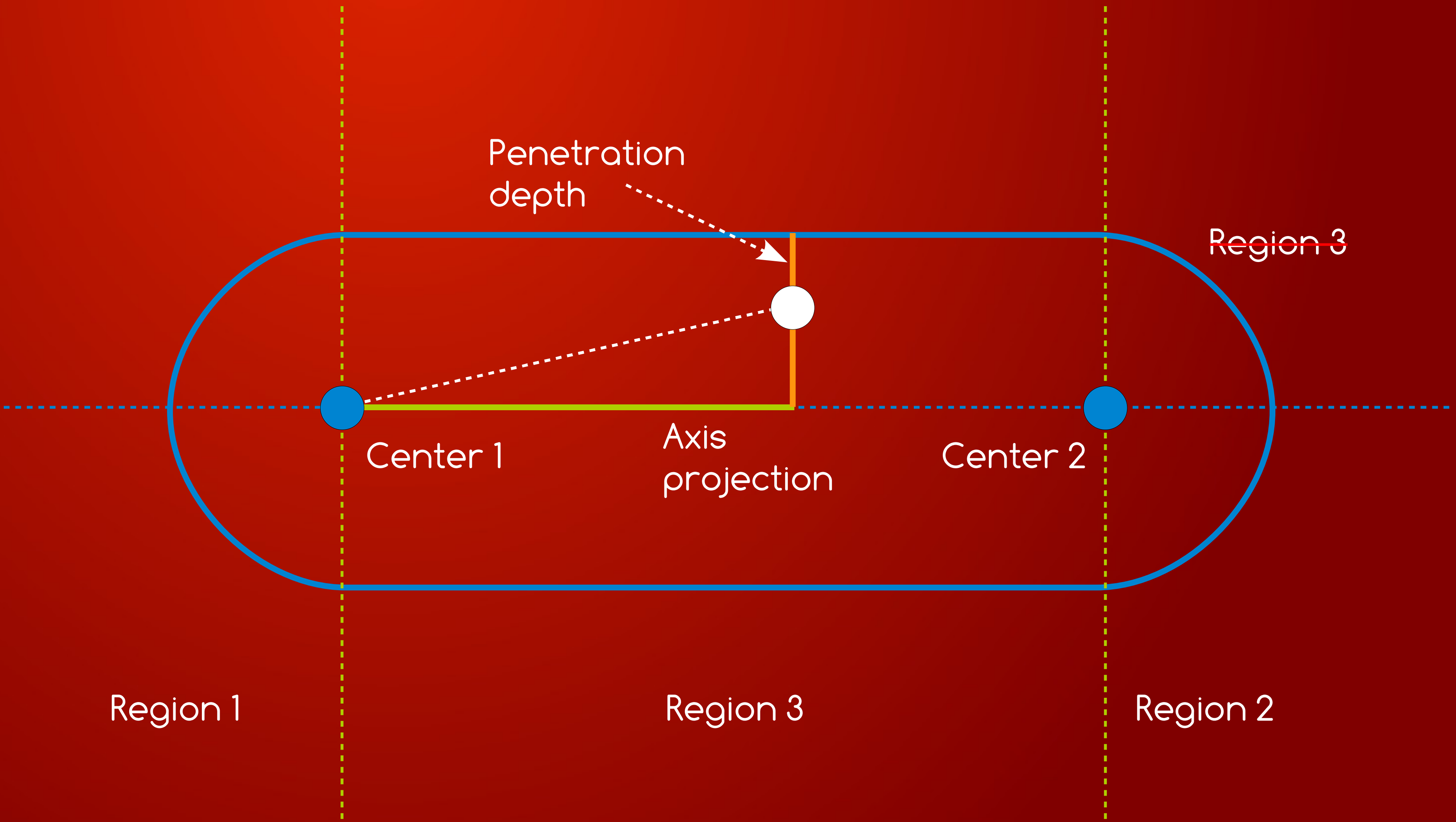


Particle – Plane Collisions

- Particle - Plane Model
  - Particle position: $p = (x,y,z)$
  - Plane origin: $o=(x,y,z)$
  - Plane normal: $m=(x,y,z)$; $|m|==1$
- Penetration depth: $d = m^T(p-c)$
- Contact normal: $n = m$



Particle - Capsule Collisions



Particle - Capsule Collisions

- Particle - Capsule Model
  - Particle position: $p = (x,y,z)$
  - Center1/2: $c1/2 = (x,y,z)$
  - Radius: r
- Algorithm:
  - Detect Voronoi Region (1,2,3)
  - In region 1/2: Compute sphere penetration
  - In region 3: Compute point-line distance
- Voronoi detection: Project $(p-c1)$ onto $(c2-c1)$
  - $f = (c2-c1)^T(p-c1)$
  - Region1 $(f<0)$; Region2 $(0<f<F)$; Region3 $(f>=F)$
  - $F = (c2-c1)^2$

Particle - Capsule Collisions

- Point - Center1 Case
  - Penetration depth: $d = |p - c1| - r$
  - Contact normal: $n = norm(p - c1)$
- Point - Center2 Case
  - Penetration depth: $d = |p - c2| - r$
  - Contact normal: $n = norm(p - c2)$
- Point – Axis Case
  - $u=norm(c2-c1)$; $v=(p-c1)$; $e=u^T v$; $f=v^T v$; $g^2=f - e^2$
  - Penetration depth: $d = r – g$
  - Penetration normal: $n = norm(v - eu)$


Practical design of
Particle System

Particle System

- Particle System
  - A set of similar particles – e.g. rendered with similar material
  - Store in array bag structure
- Particle
  - Has lifetime, physical and material properties
  - During simulation lifetime is decremented until $< 0 \rightarrow$ dead
  - Dead particles are reused for newly emitted particles
- Obstacles: Objects in the scene used as colliders
  - Sphere, boxes, planes, capsules...

Emitters

- Particle emitter: Creates new particles
  - Particle emit rate: How many particles are emitted per sec
  - Particle initial values: Particle initialization before emission.
  - Custom (physical) and geometrical properties
- Common emitters
  - Point emitter: Emit particle from point
  - Sphere emitter: Emit particles inside volume (on surface)
  - Box emitter: Emit particles inside generic box
  - Cone emitter: Emit particles inside a cone
  - And many more...

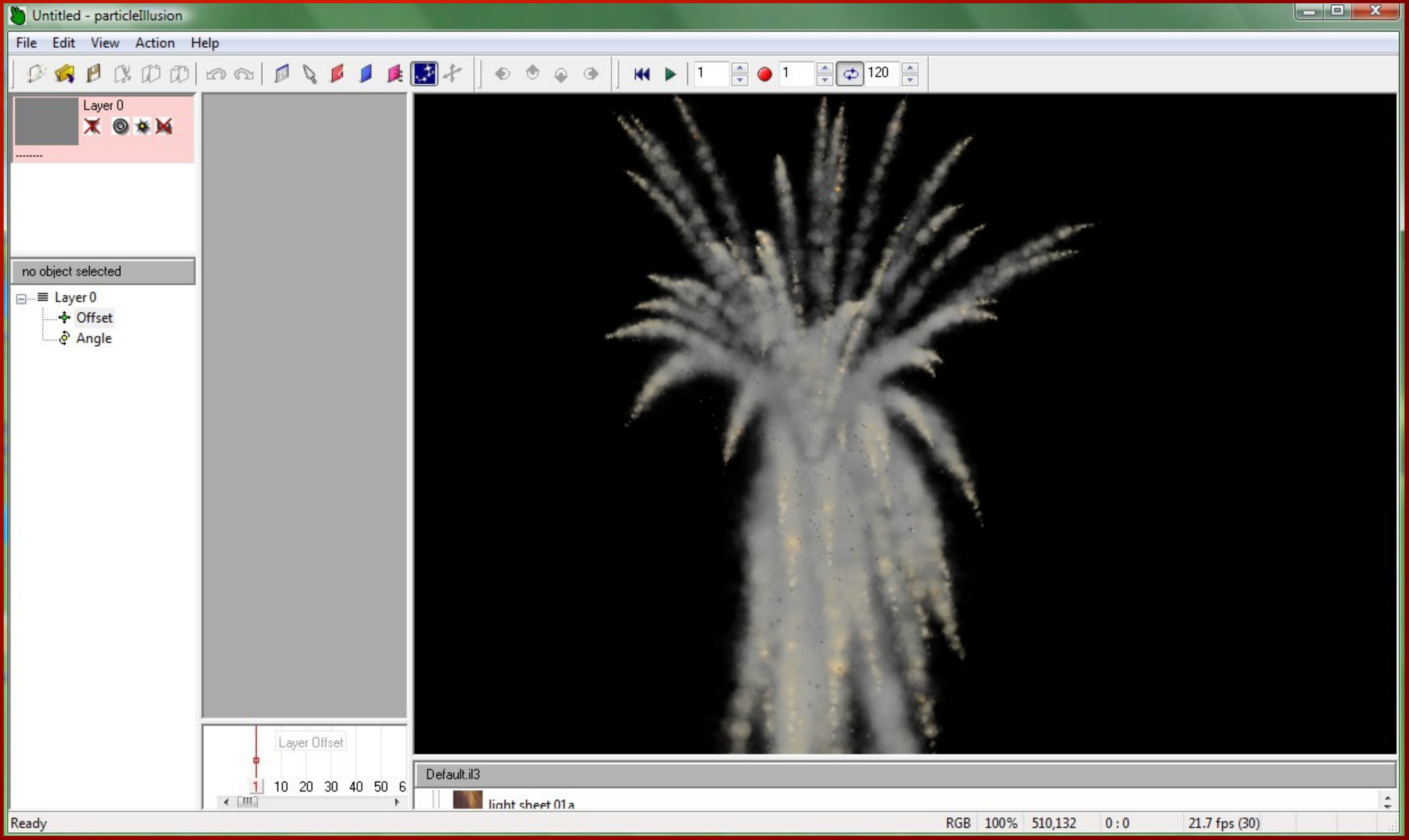
Attractors

- Particle attractor:
  - Is a generic description of forces attracting close particles
- Common attractors:
  - Linear drag: wind, gravity, user drag
  - Vortex drag: rotational force field
  - Distance magnets: obstacles acts like magnets

Demos / tools / libs

Demos / Tools / Libs

- Particle Illusion (http://www.wondertouch.com/)



The End