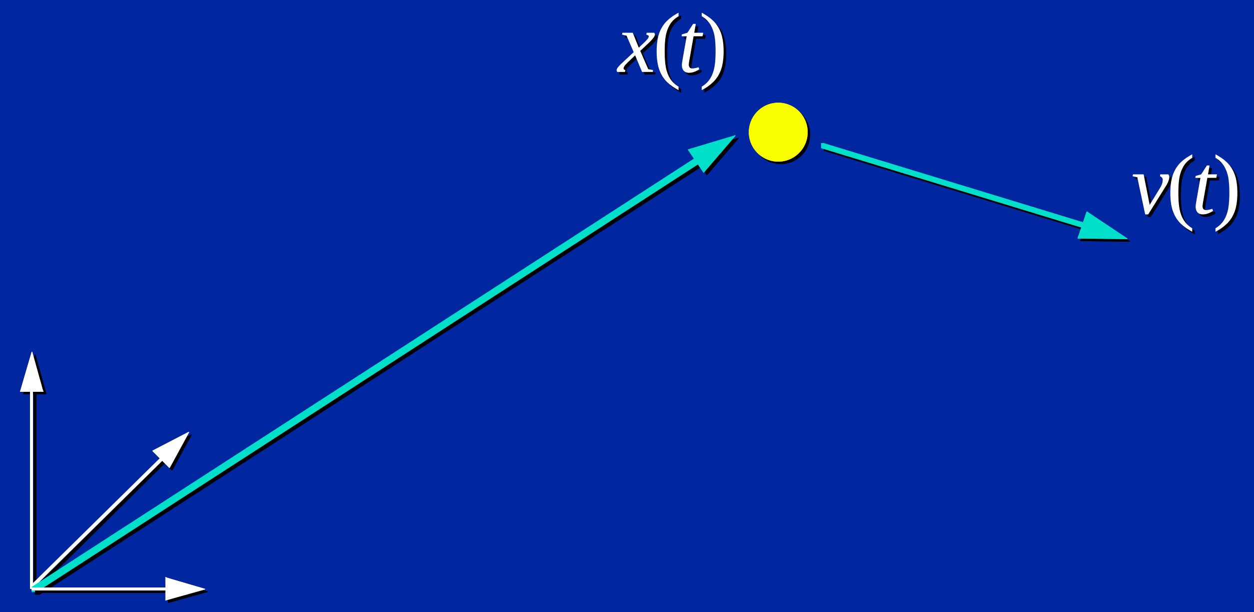
Rigid Body Simulation

David Baraff
Robotics Institute and
School of Computer Science

Carnegie Mellon

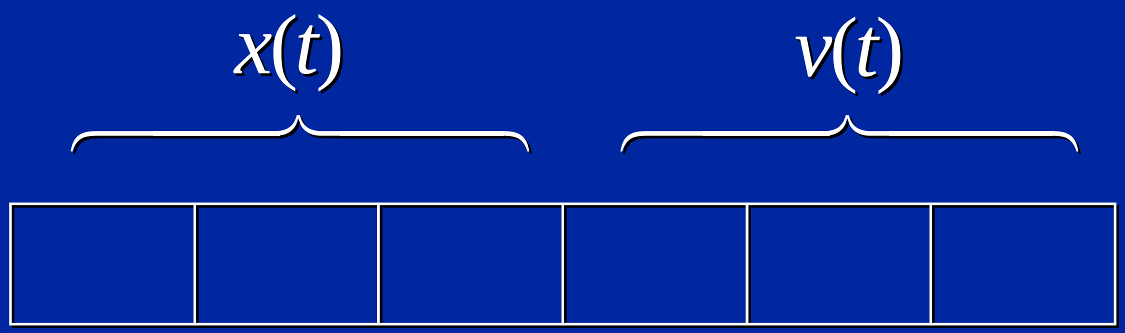
Particle Motion



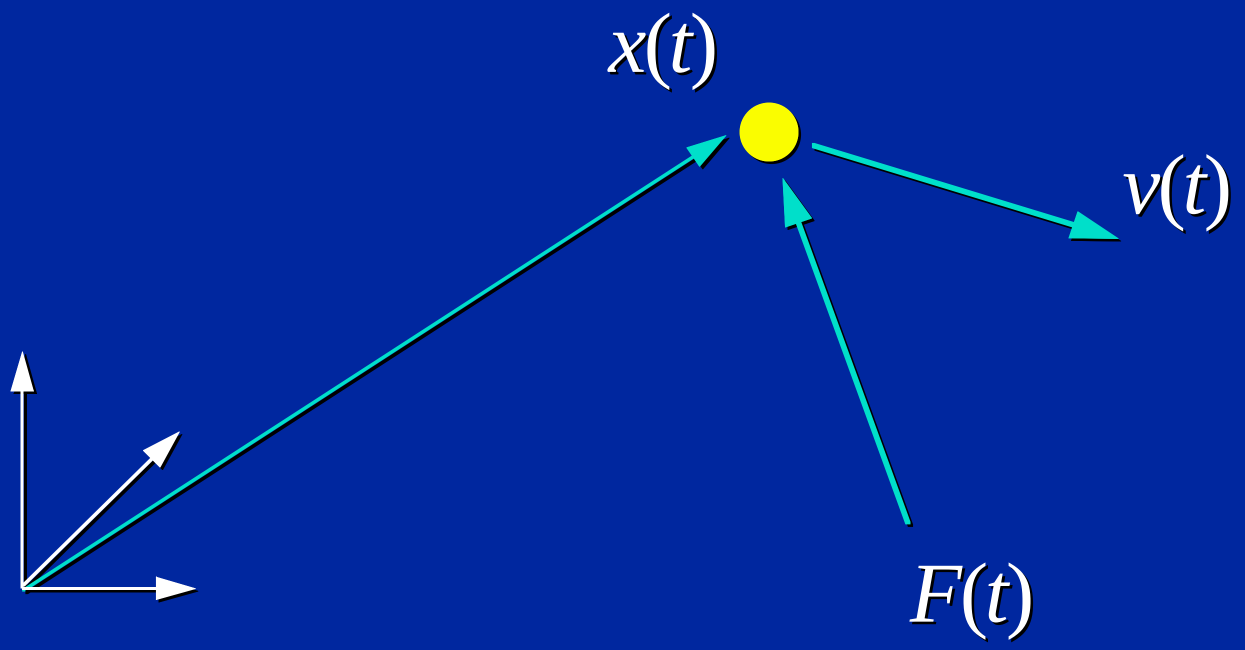
Particle State

$$\mathbf{Y} = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$

$\mathbf{Y} =$



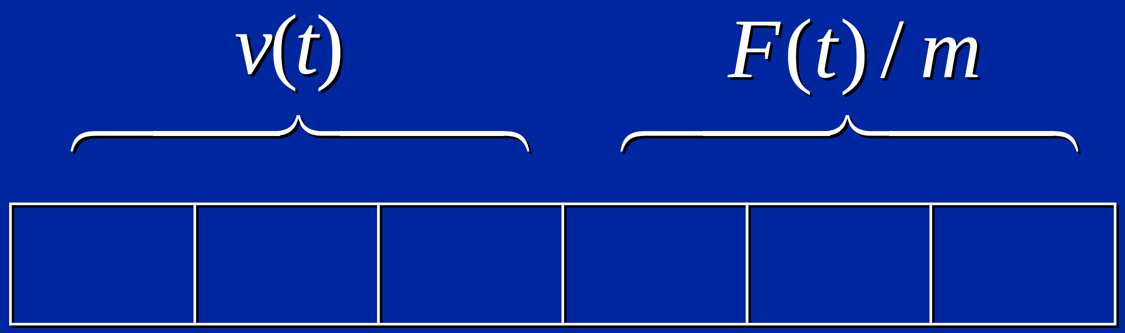
Particle Dynamics



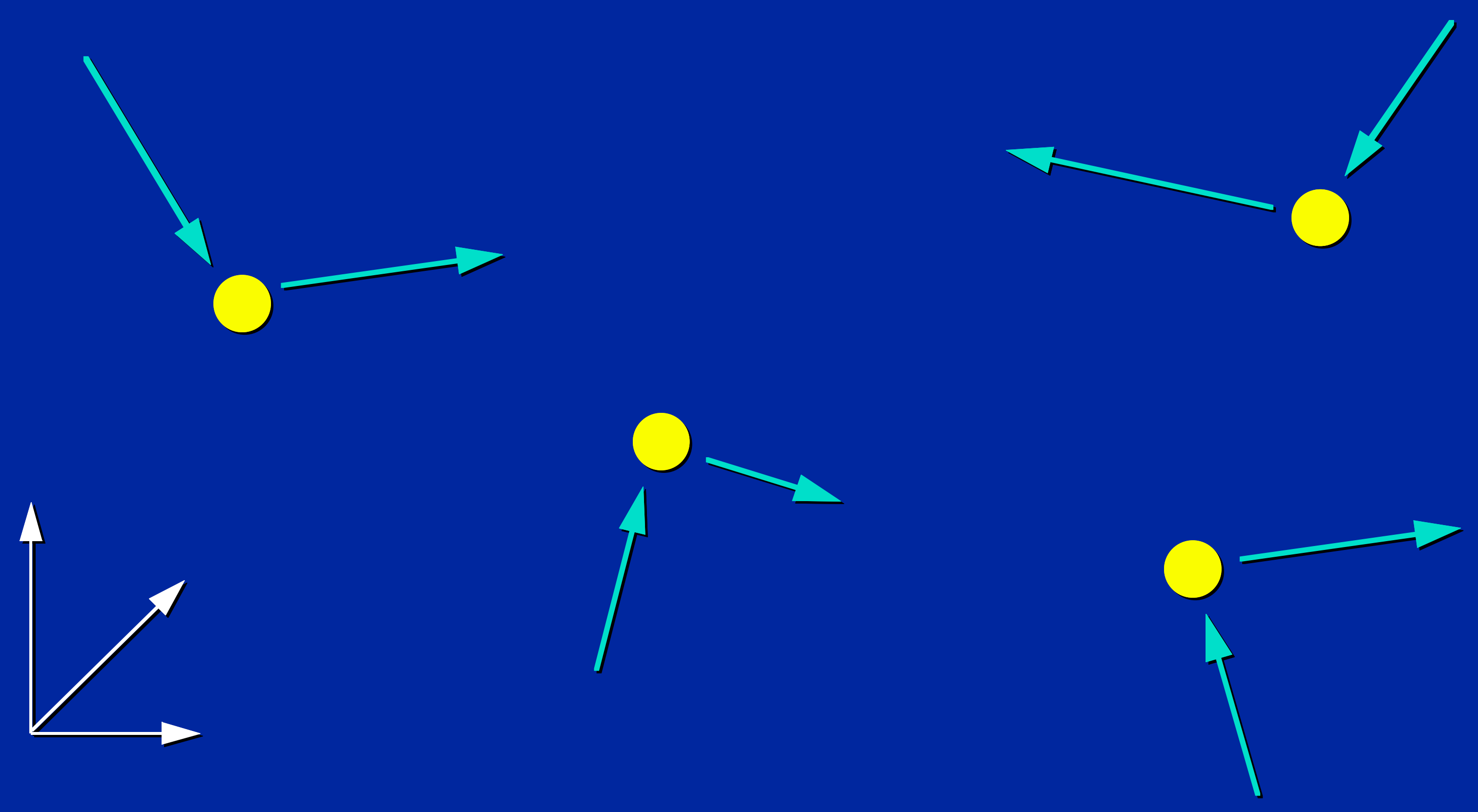
State Derivative

$$\frac{d}{dt}\mathbf{Y} = \frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ F(t)/m \end{pmatrix}$$

$$\frac{d}{dt}\mathbf{Y} =$$



Multiple Particles



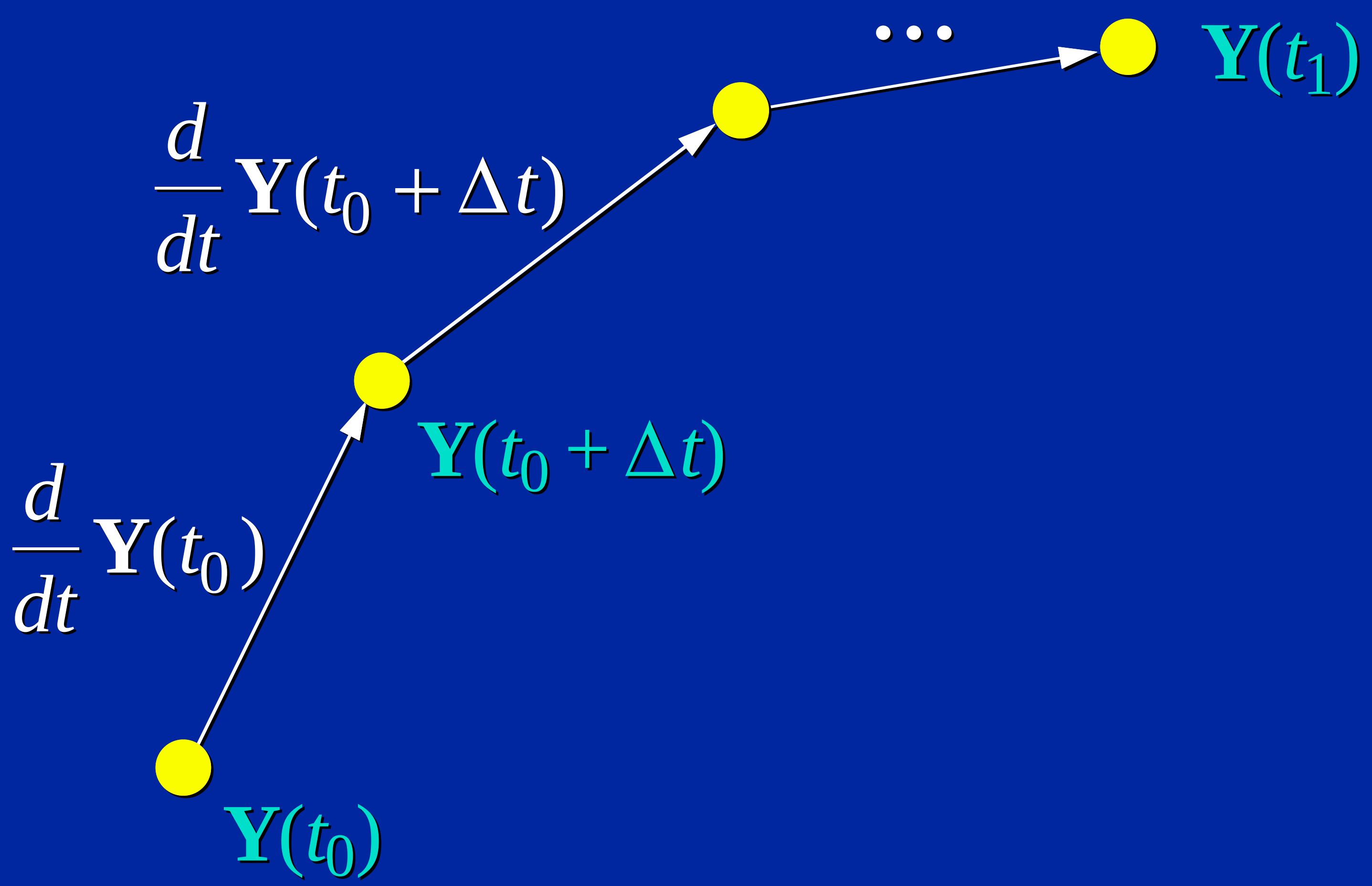
State Derivative

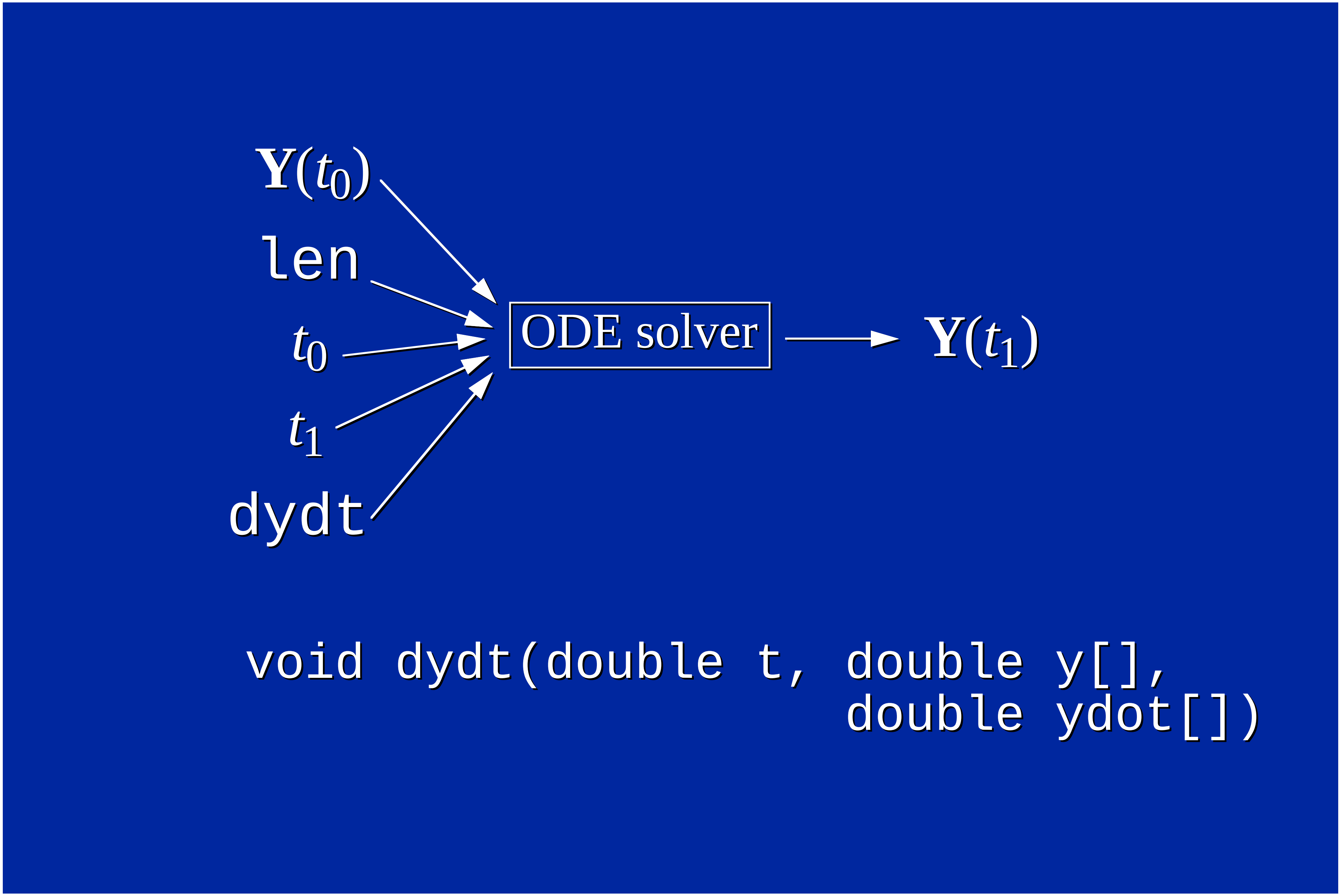
$$\frac{d}{dt}\mathbf{Y} = \frac{d}{dt}\begin{pmatrix} x_1(t) \\ v_1(t) \\ \vdots \\ x_n(t) \\ v_n(t) \end{pmatrix} = \begin{pmatrix} v_1(t) \\ F_1(t)/m_1 \\ \vdots \\ v_n(t) \\ F_n(t)/m_n \end{pmatrix}$$

$$\frac{d}{dt}\mathbf{Y} =$$

		••• $6n$ elements •••		

ODE solution

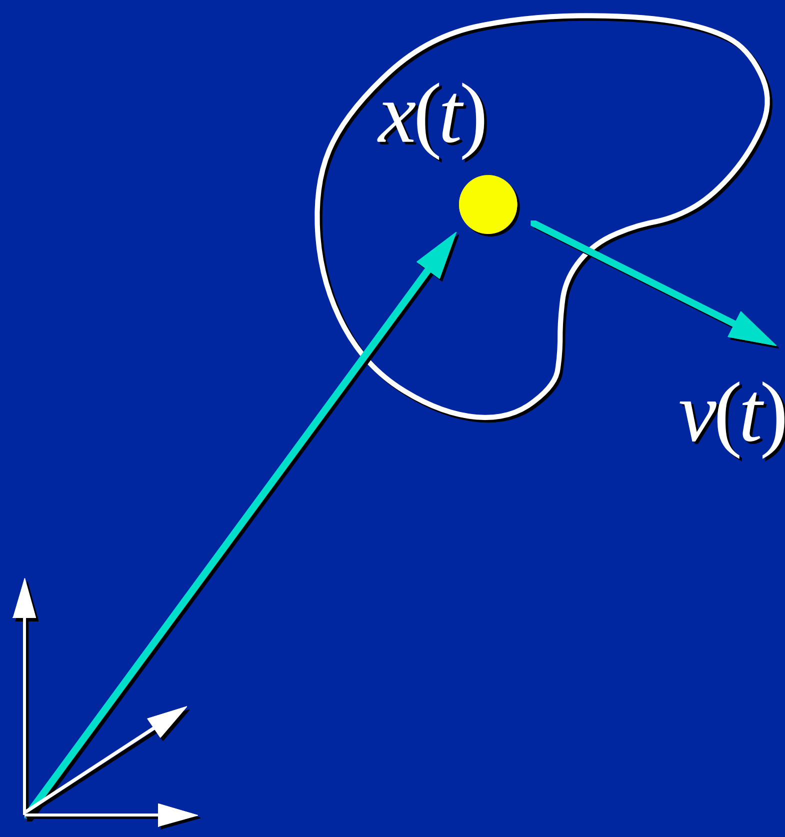



Y(t0)
len
t0
t1
dydt
ODE solver
Y(t1)
void dydt(double t, double y[],
double ydot[])

dydt

$$\mathbf{Y}(t) = \begin{bmatrix} x_1(t) \\ v_1(t) \\ \vdots \\ x_n(t) \\ v_n(t) \end{bmatrix} \qquad \frac{d}{dt}\mathbf{Y}(t) = \begin{bmatrix} v_1(t) \\ F_1(t)/\,m_1 \\ \vdots \\ v_n(t) \\ F_n(t)/m_n \end{bmatrix}$$

Rigid Body State

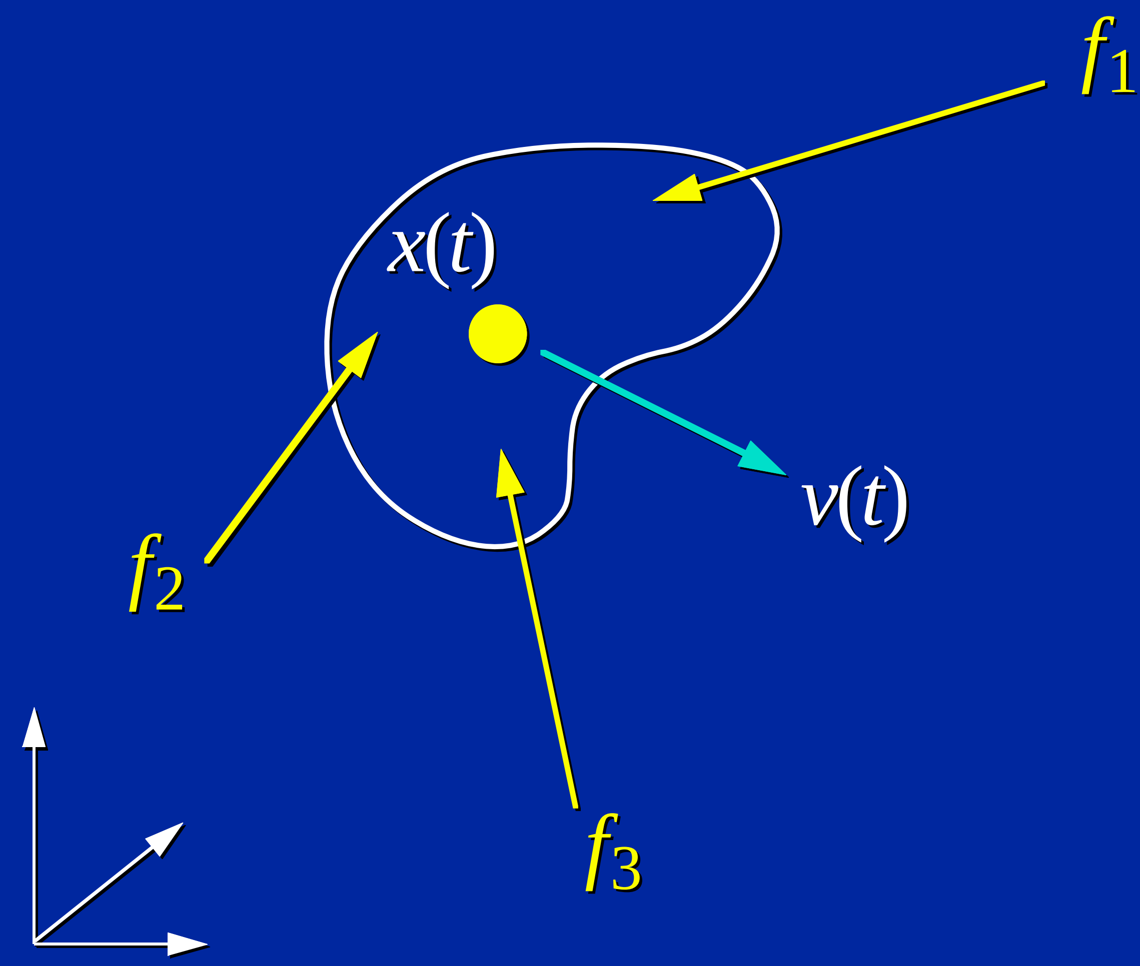


$$\mathbf{Y} = \begin{pmatrix} x(t) \\ ? \\ v(t) \\ ? \end{pmatrix}$$

Rigid Body Equation of Motion

$$\frac{d}{dt}\mathbf{Y} = \frac{d}{dt}\begin{pmatrix} x(t) \\ ? \\ Mv(t) \\ ? \end{pmatrix} = \begin{pmatrix} v(t) \\ ? \\ F(t) \\ ? \end{pmatrix}$$

Net Force



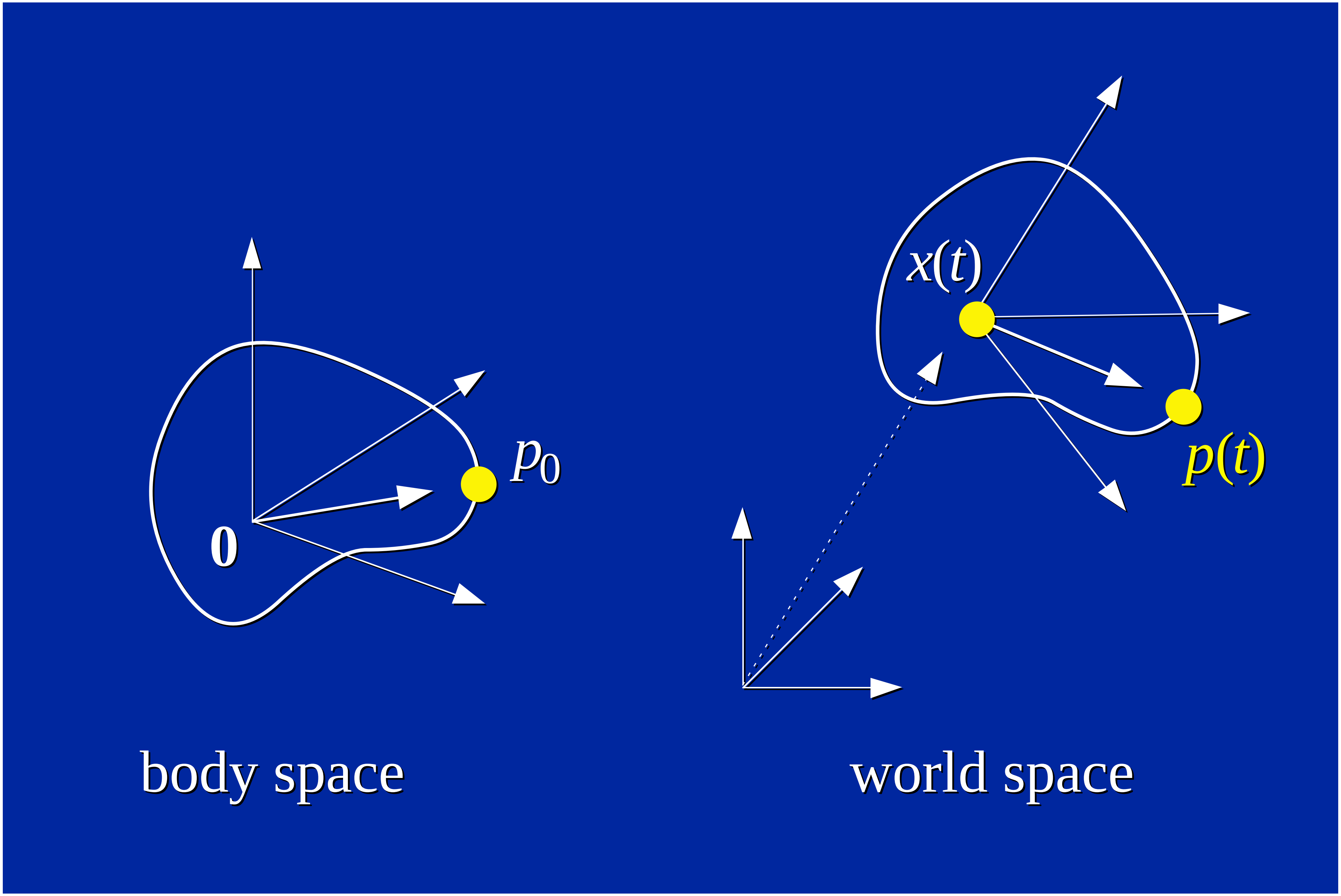
$$F(t) = \sum f_i$$

Orientation

We represent orientation as a rotation matrix[†] $R(t)$. Points are transformed from body-space to world-space as:

$$p(t) = R(t)p_0 + x(t)$$

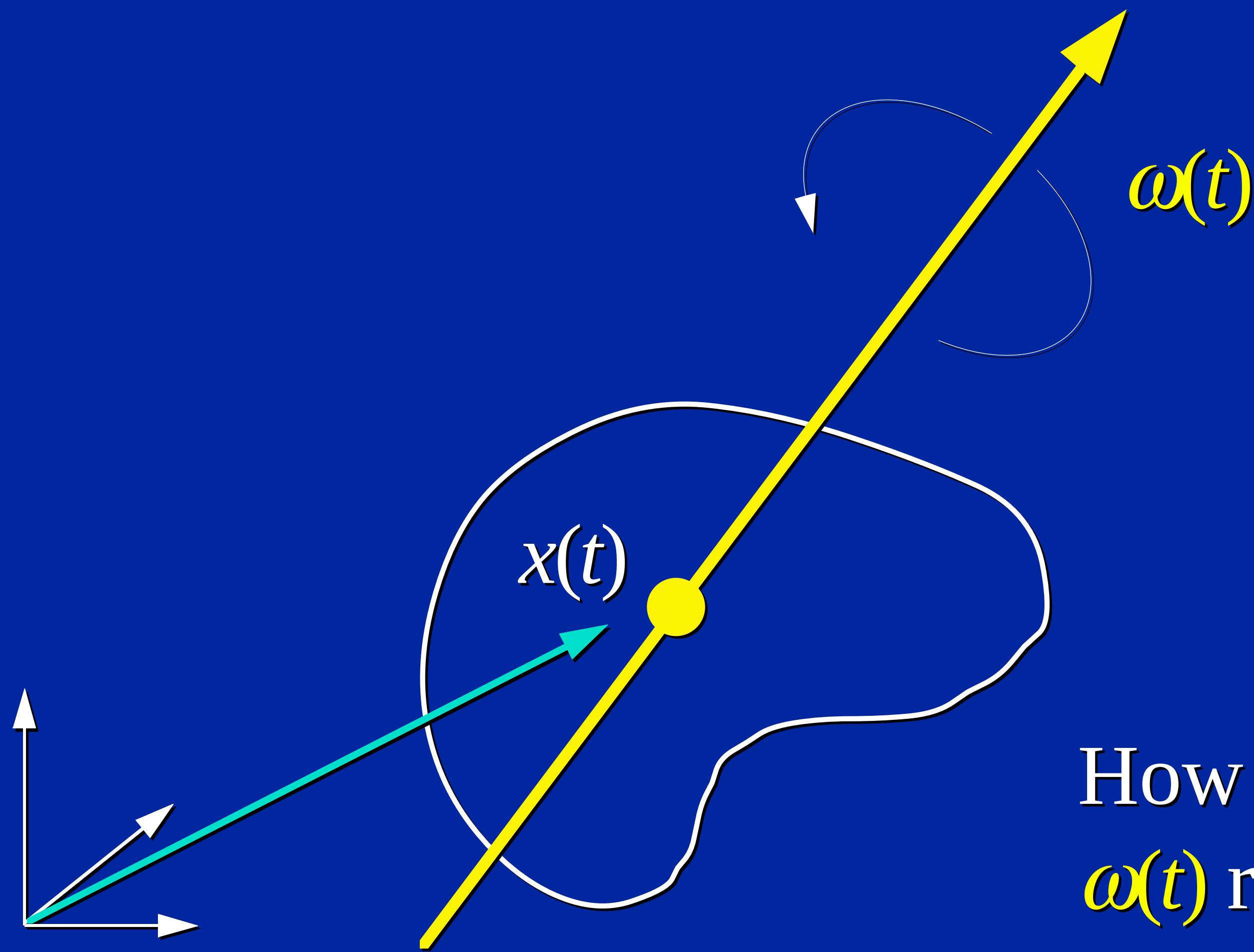
[†]He's lying. Actually, we use quaternions.


p_0
0
body space
$x(t)$
$p(t)$
world space

Angular Velocity

We represent angular velocity as a vector $\omega(t)$, which encodes both the axis of the spin and the speed of the spin.

Angular Velocity Definition



How are $R(t)$ and $\omega(t)$ related?

Angular Velocity

$\dot{R}(t)$ and $\omega(t)$ are related by

$$\frac{d}{dt}R(t) = \begin{pmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{pmatrix} R(t)$$

($\omega(t)^*$ is a shorthand for the above matrix)

Rigid Body Equation of Motion

$$\frac{d}{dt}\mathbf{Y} = \frac{d}{dt}\begin{pmatrix} x(t) \\ R(t) \\ Mv(t) \\ < \omega(t) > \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t)^{*} R(t) \\ F(t) \\ ? \end{pmatrix}$$

Need to relate $\dot{\omega}(t)$ and mass distribution to $F(t)$.

Inertia Tensor

$$I(t) = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}$$

diagonal terms†

$$I_{xx} = M \int_V (y^2 + z^2)\, dV$$

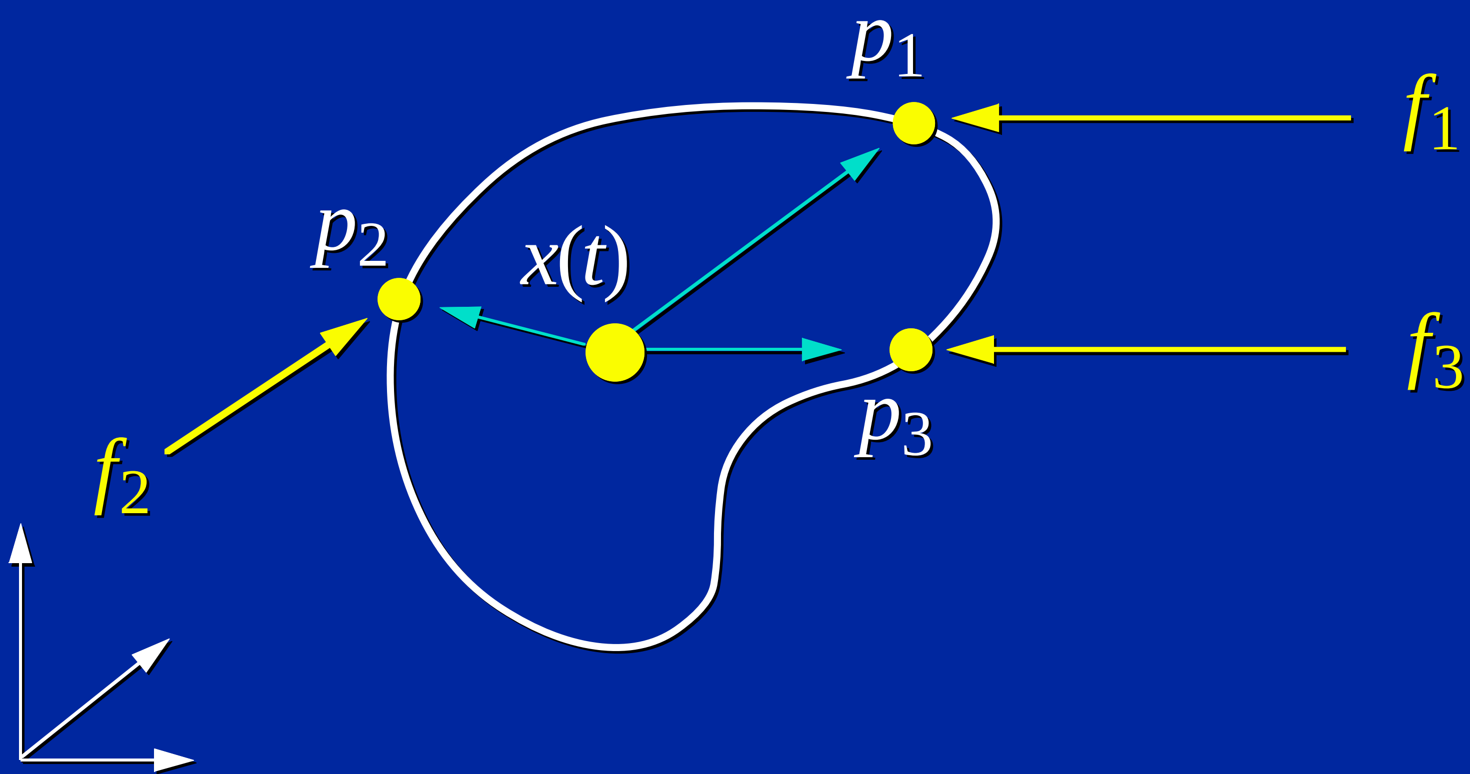
off-diagonal terms†

$$I_{xy} = -M \int_V xy\, dV$$

---

†Integrals are precomputed.

Net Torque



$$\tau(t) = \sum (p_i - x(t)) \times f_i$$

Rigid Body Equation of Motion

$$\frac{d}{dt}\mathbf{Y} = \frac{d}{dt}\begin{pmatrix} x(t) \\ R(t) \\ \boxed{Mv(t)} \\ \boxed{I(t)\omega(t)} \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t)^{*} R(t) \\ F(t) \\ \tau(t) \end{pmatrix}$$

$P(t)$ – linear momentum

$L(t)$ – angular momentum

What's in the Course Notes

1. Implementation of `dydt` for rigid bodies (bookkeeping, data structures, computations)
2. Quaternions – derivations and code
3. Miscellaneous formulas and examples
4. Derivations for force and torque equations, center of mass, inertia tensor, rotation equations, velocity/acceleration of points

Constraints

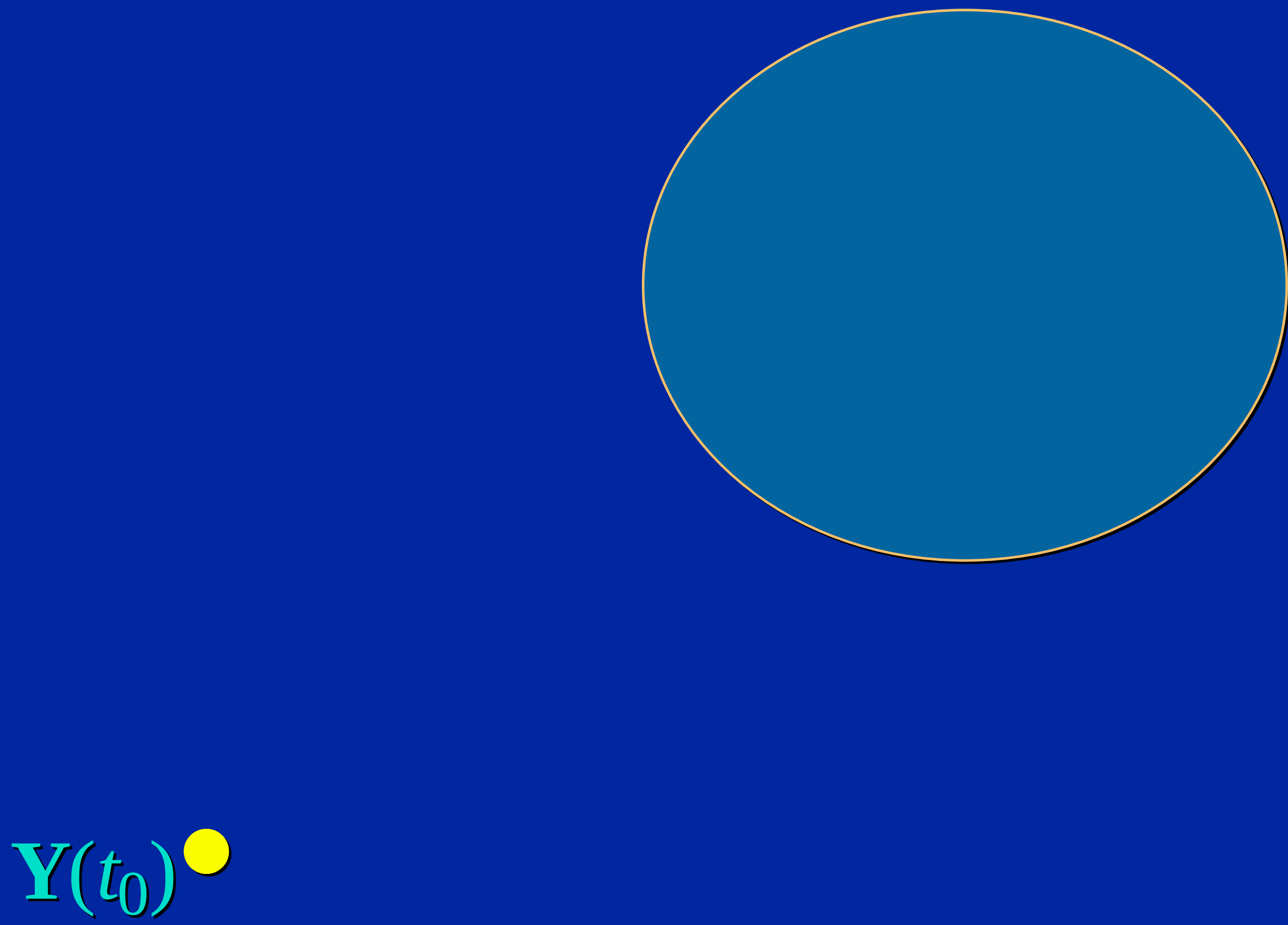
We want rigid bodies to behave as solid objects, and not inter-penetrate. By applying constraint forces between contacting bodies, we prevent interpenetration from occurring. We need to:

a) Detect interpenetration

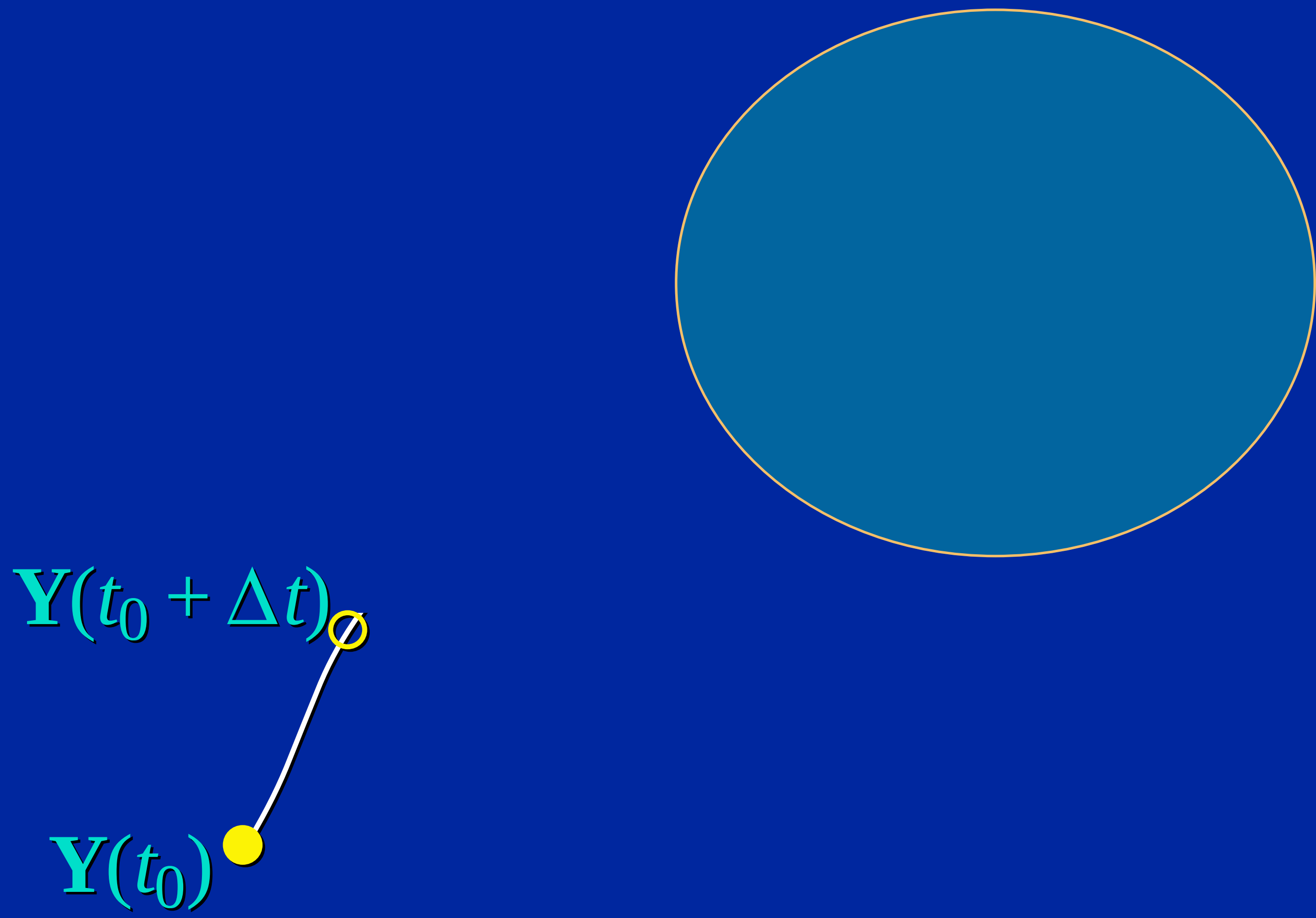
b) Determine contact points

c) Compute constraint forces

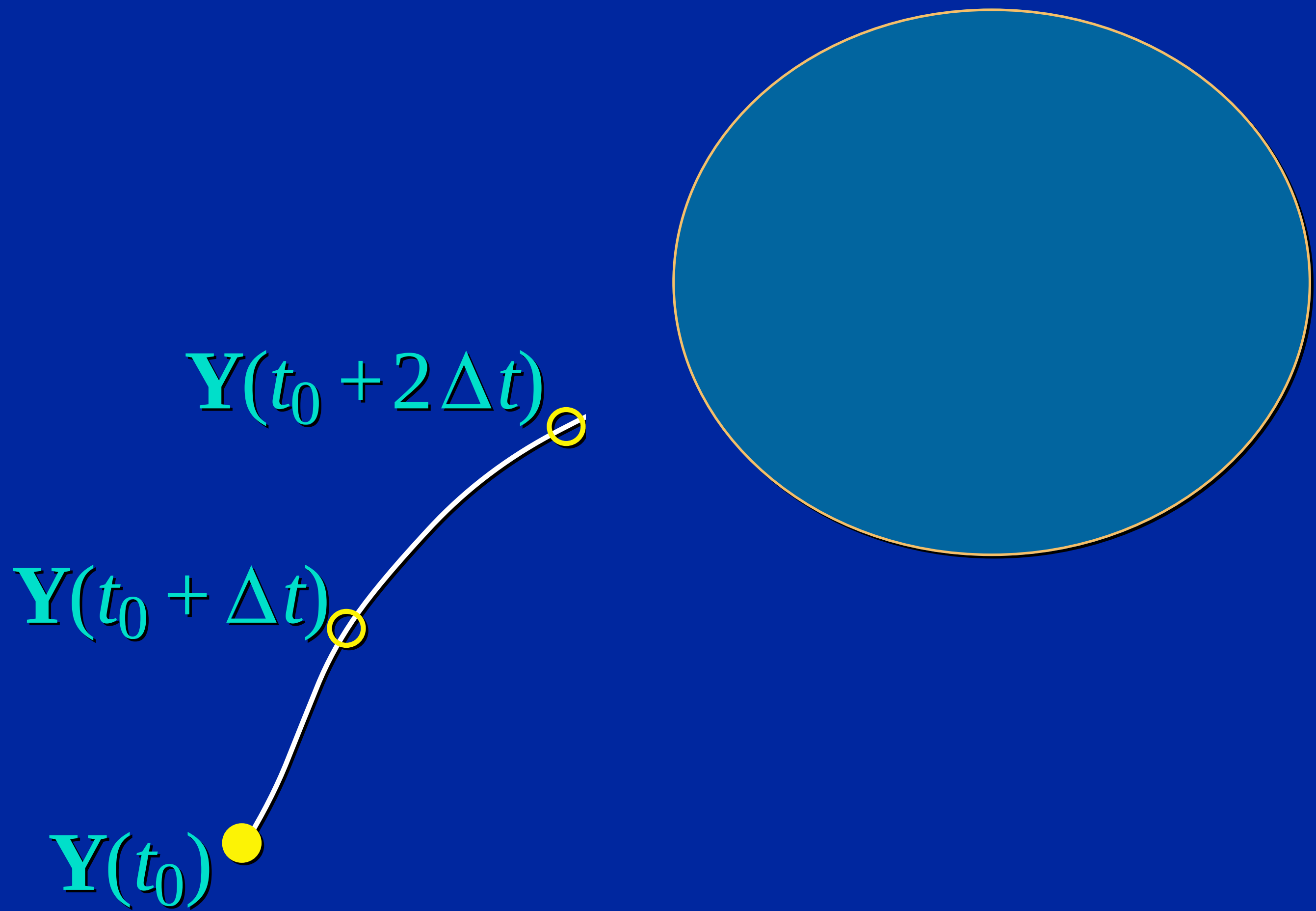
Simulations with Collisions



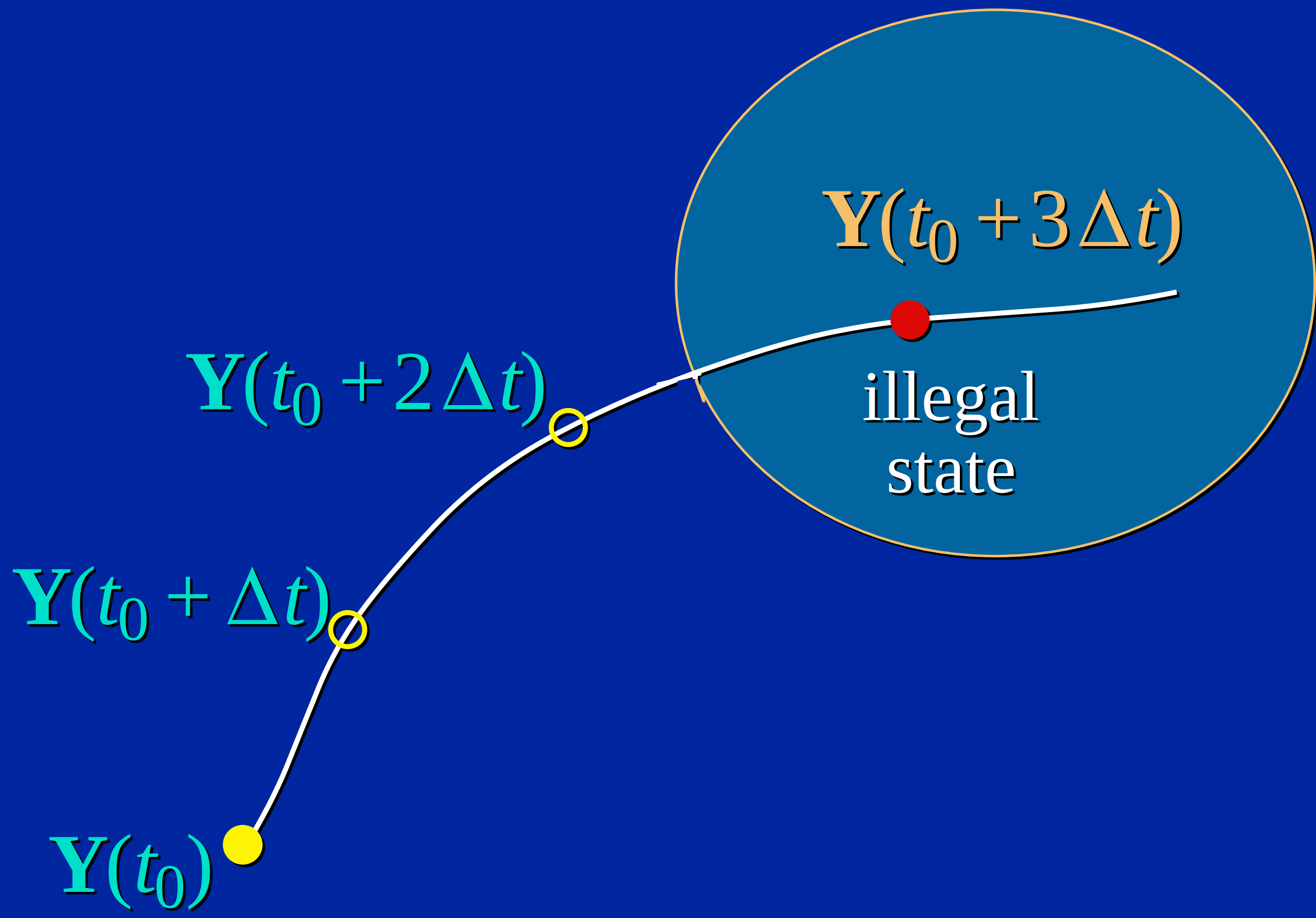
Simulations with Collisions



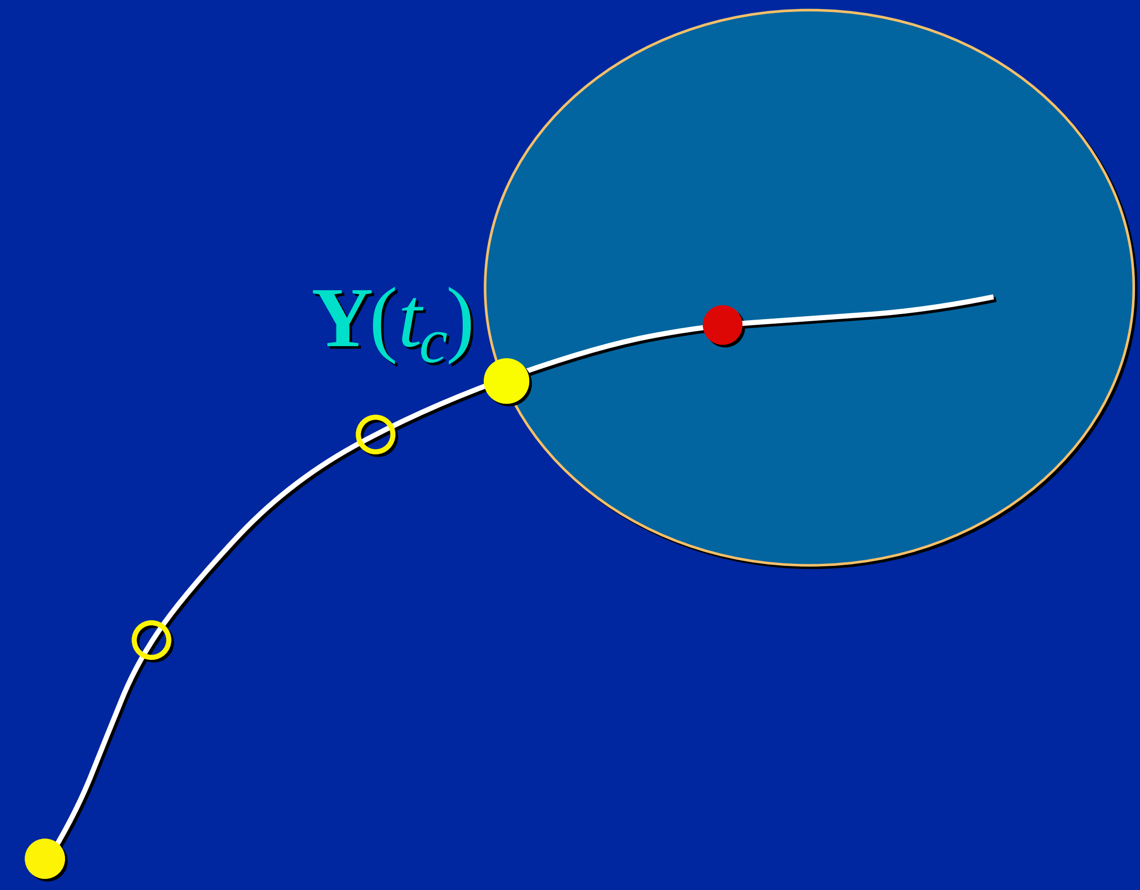
Simulations with Collisions



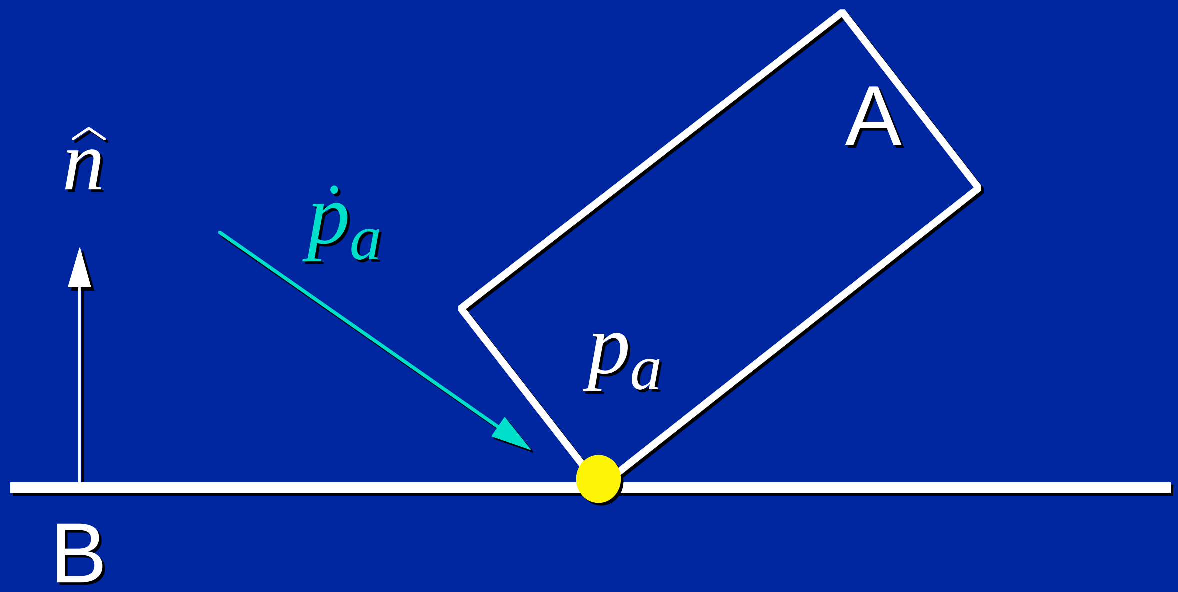
An Illegal State Y



Backing up to the Collision Time

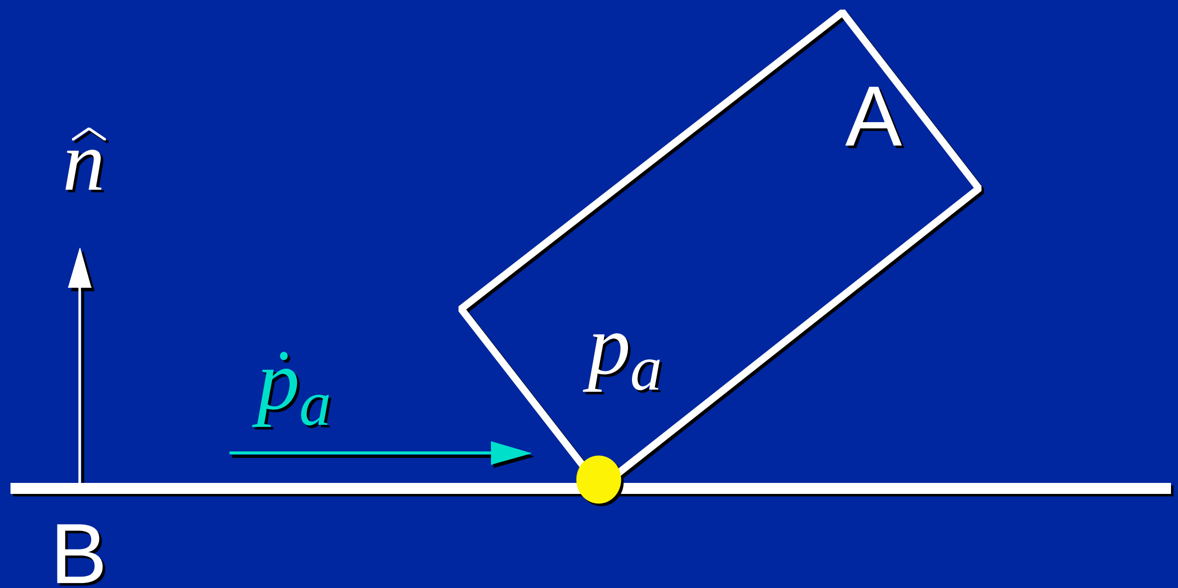


Colliding Contact



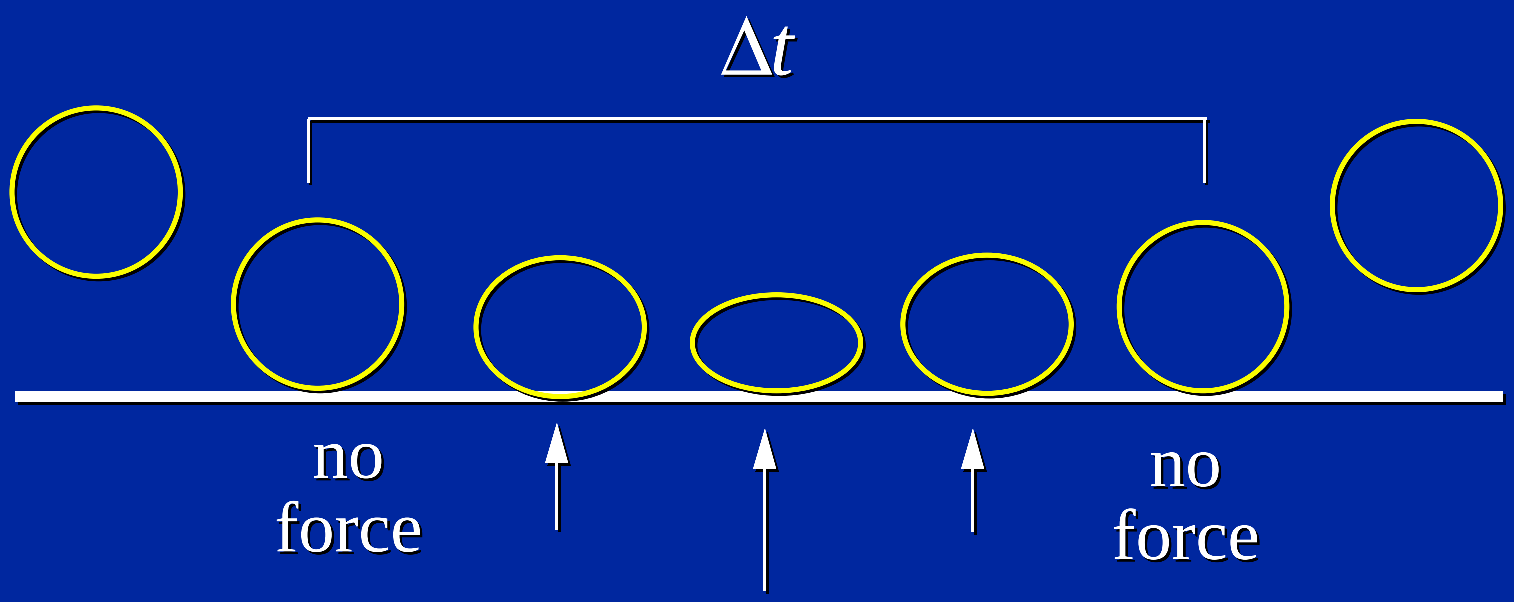
$$\hat{n} \bullet \dot{p}_a < 0$$

Resting Contact

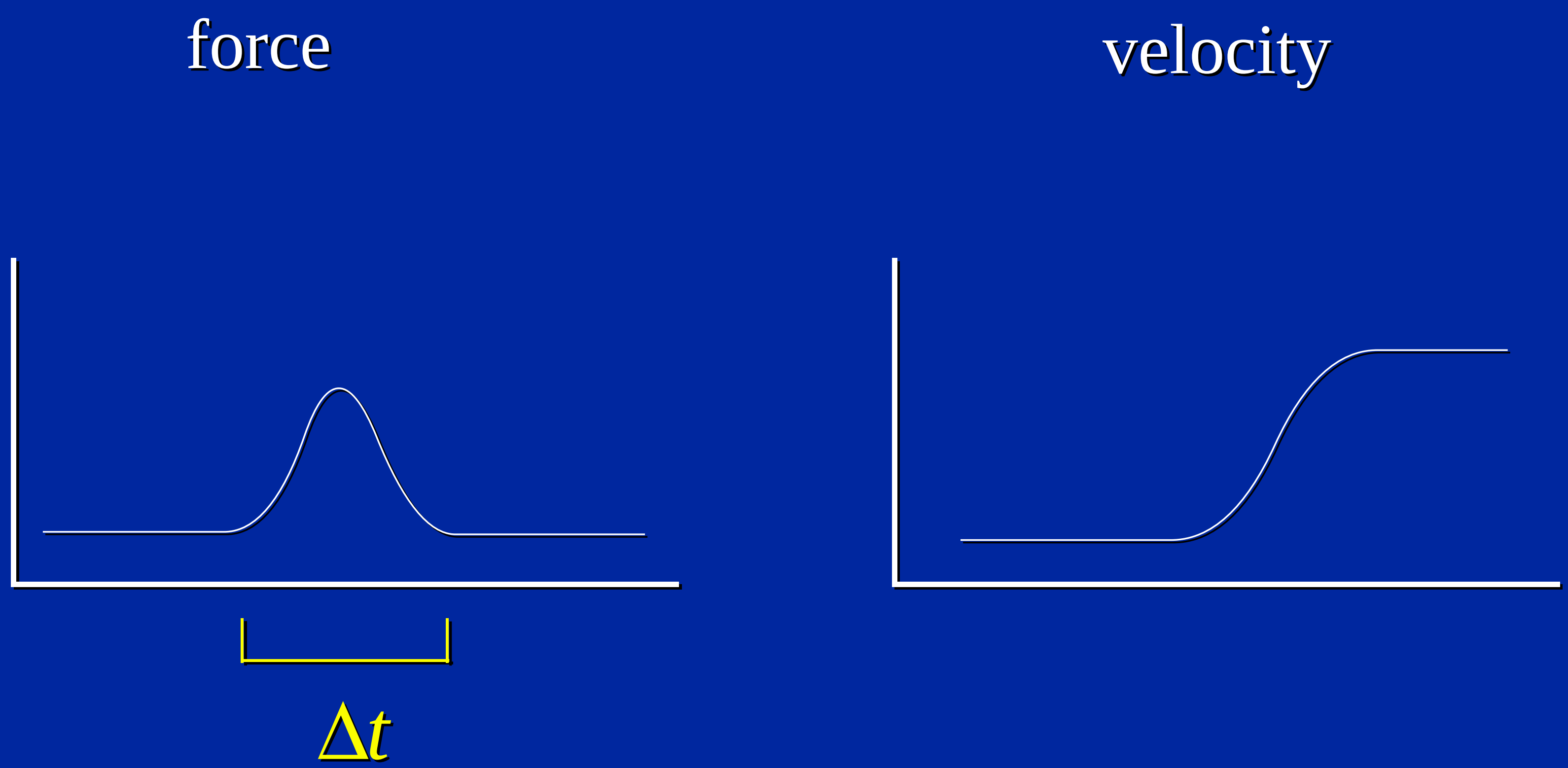


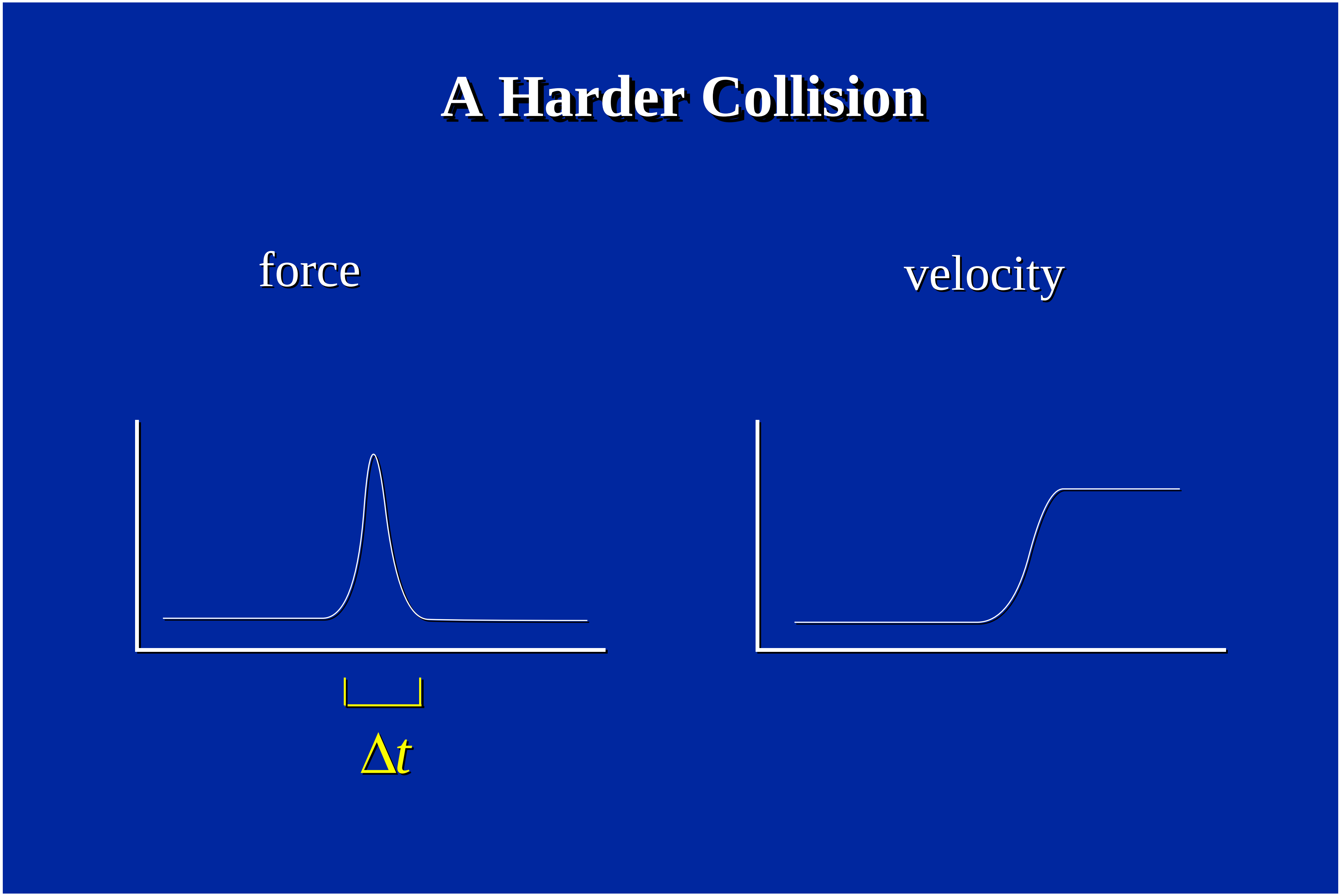
$$\hat{n} \bullet \dot{p}_a = 0$$

Collision Process

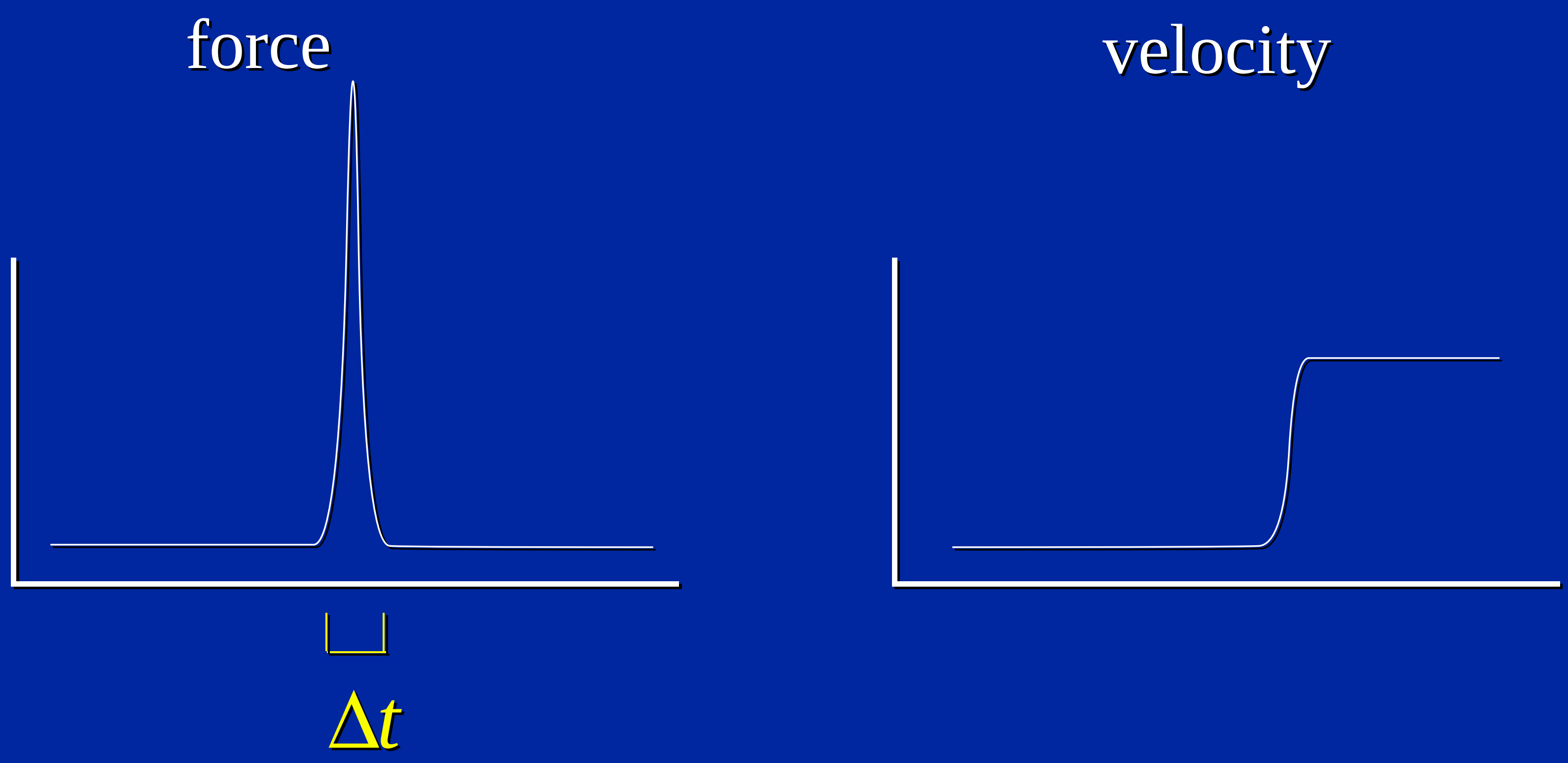


A Soft Collision

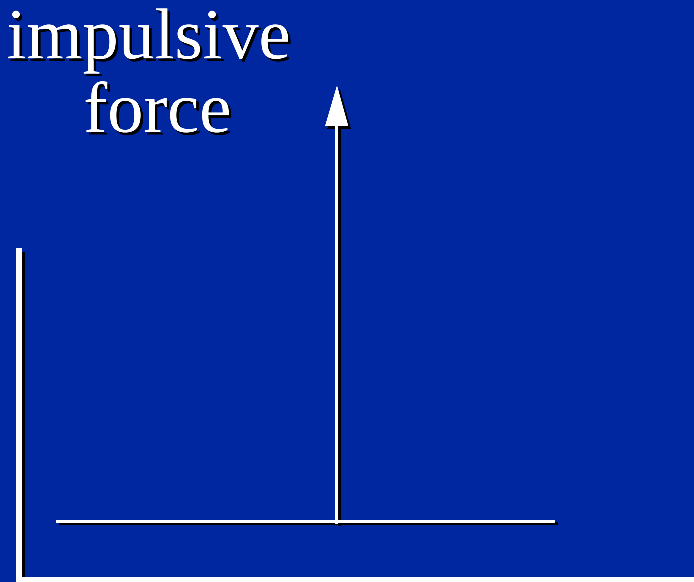


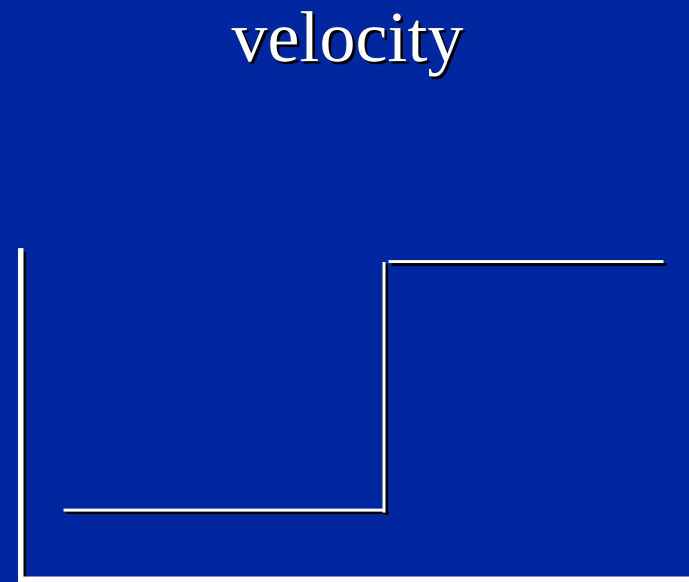

A Harder Collision
force
velocity
Δt

A Very Hard Collision



A Rigid Body Collision

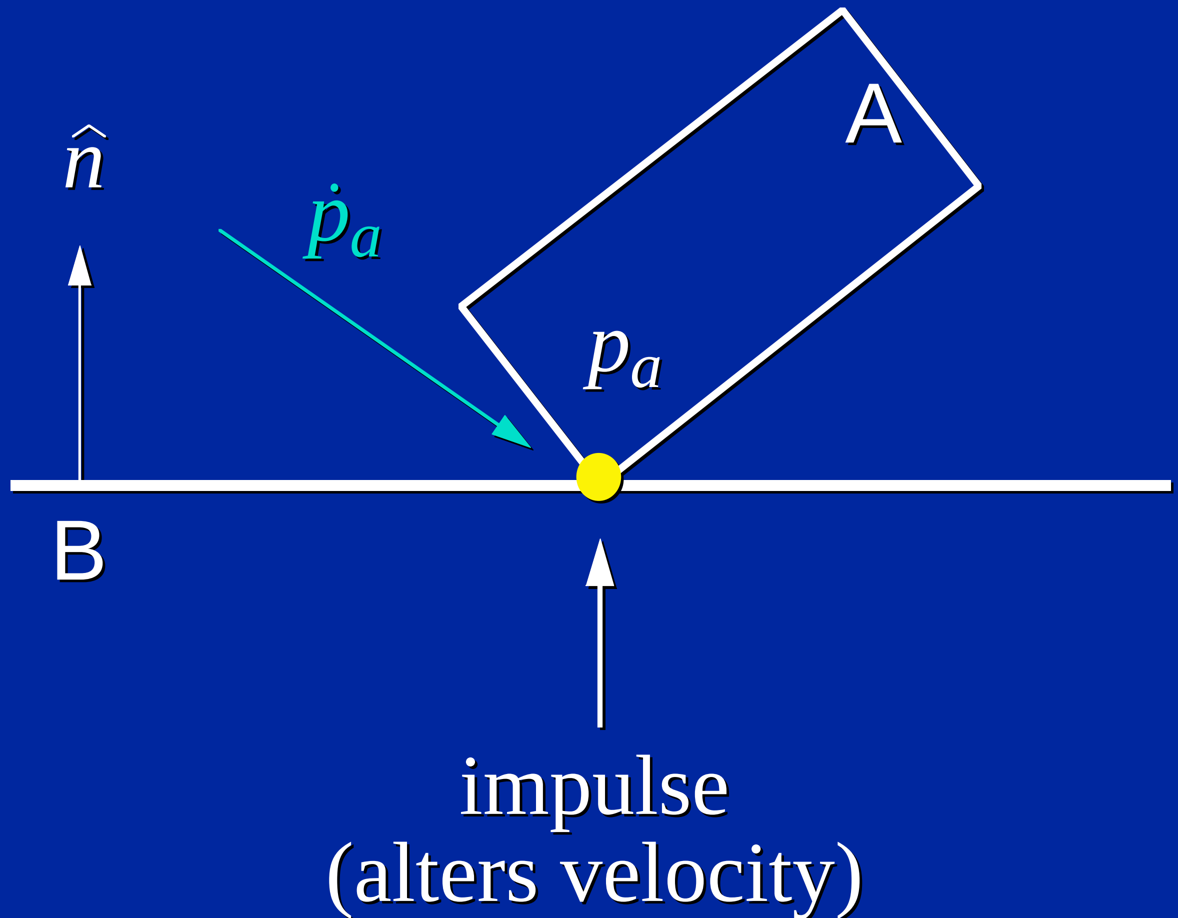


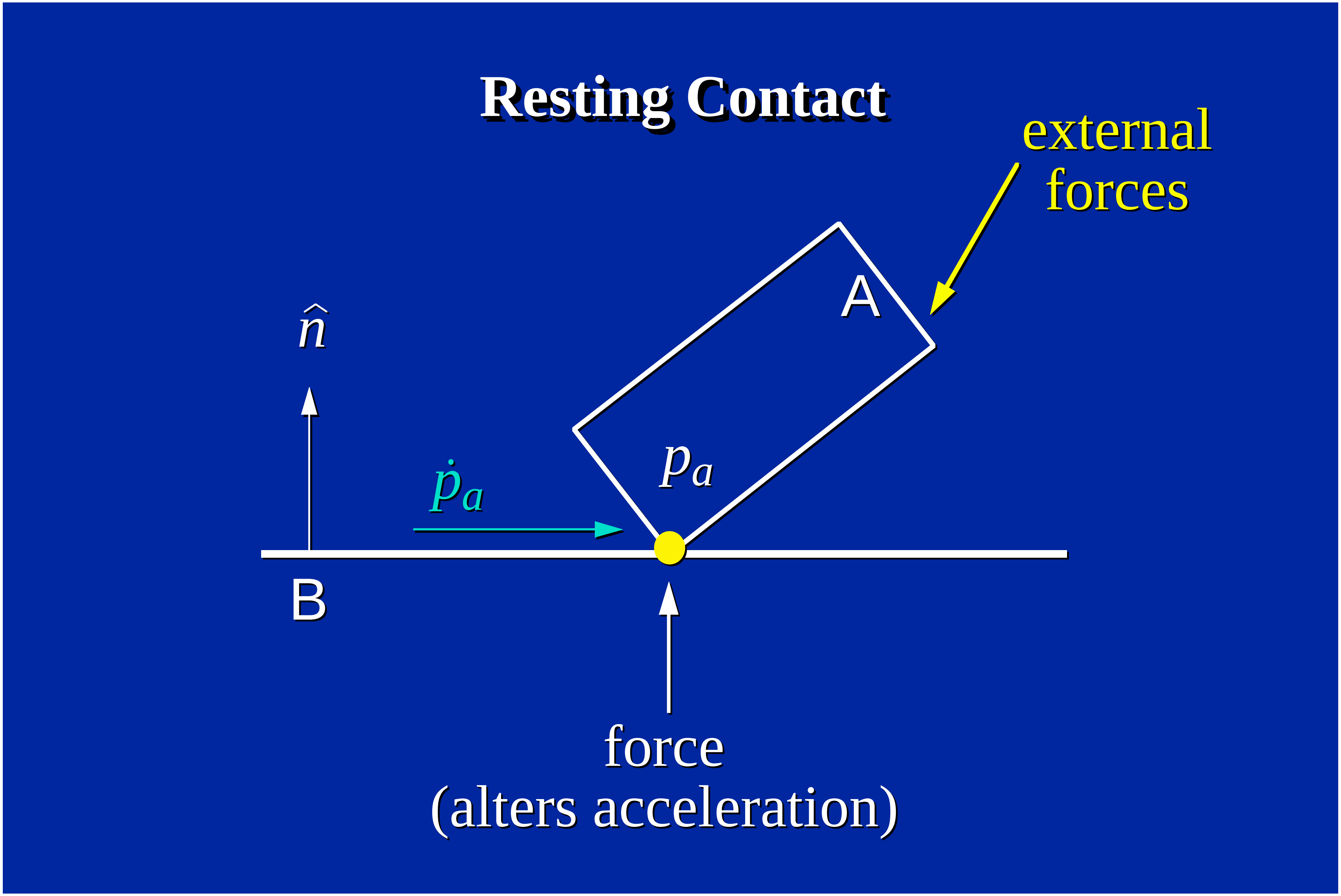


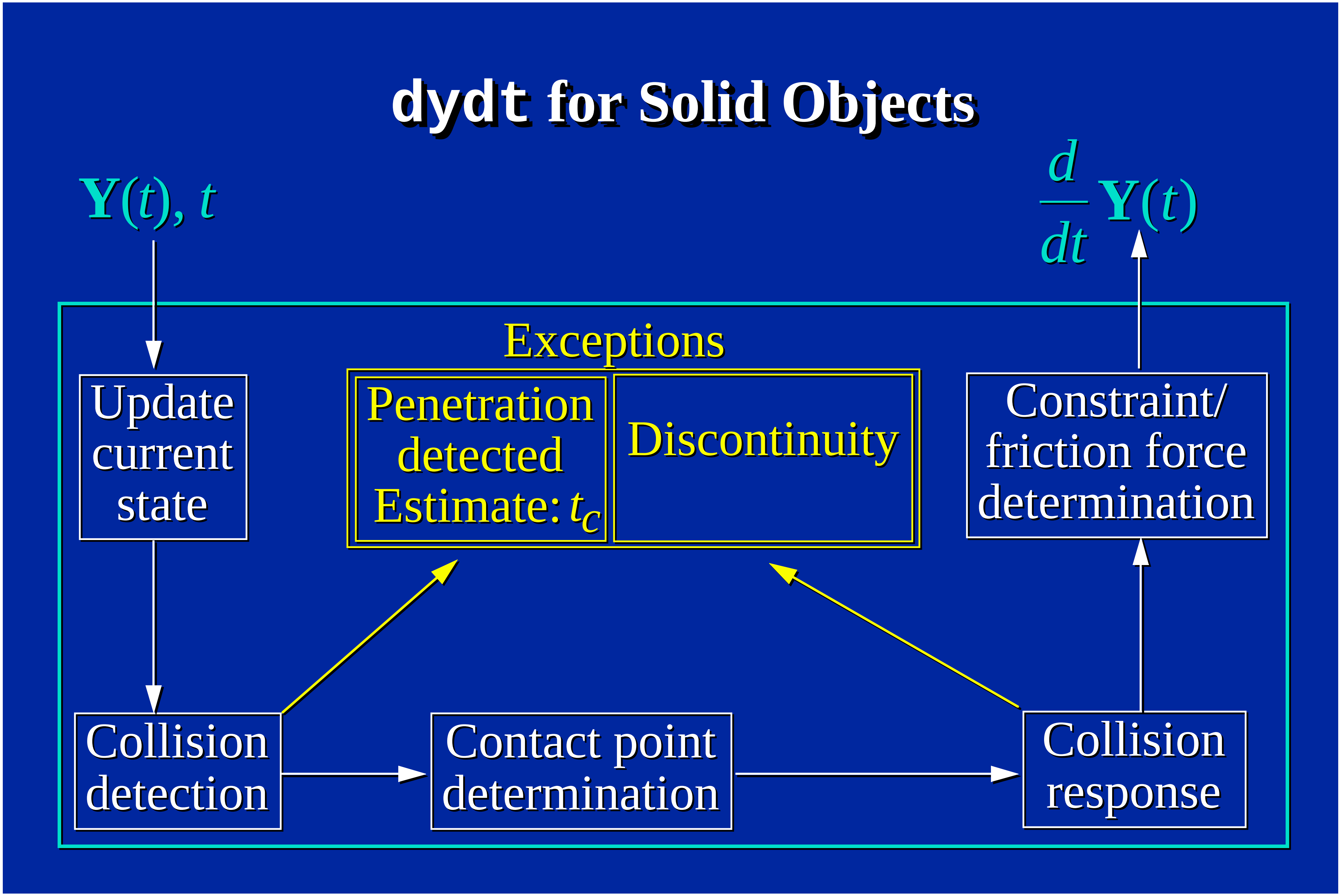
$$f_{imp} = \infty$$

$$\Delta t = 0$$

Colliding Contact




Resting Contact
external forces
A
$\hat{n}$
$\dot{p}_a$
p_a
B
force
(alters acceleration)

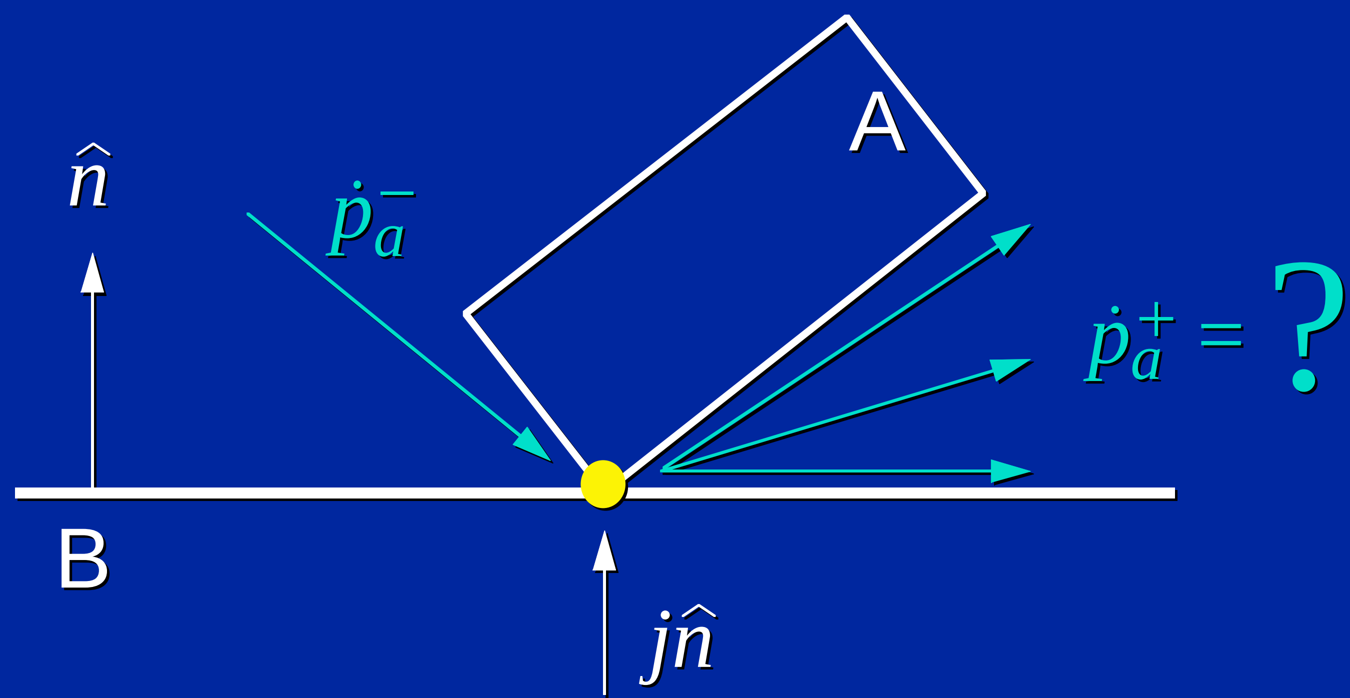

dydt for Solid Objects
Y(t), t
d/dt Y(t)
Exceptions
Update current state
Penetration detected Estimate: t_c
Discontinuity
Constraint/ friction force determination
Collision detection
Contact point determination
Collision response

In the Course Notes – Collision Detection

Bounding box check between *n* objects: yes, you *can* avoid $O(n^2)$ work. Don't even settle for $O(n \log n)$ – insist on an $O(n)$ algorithm!

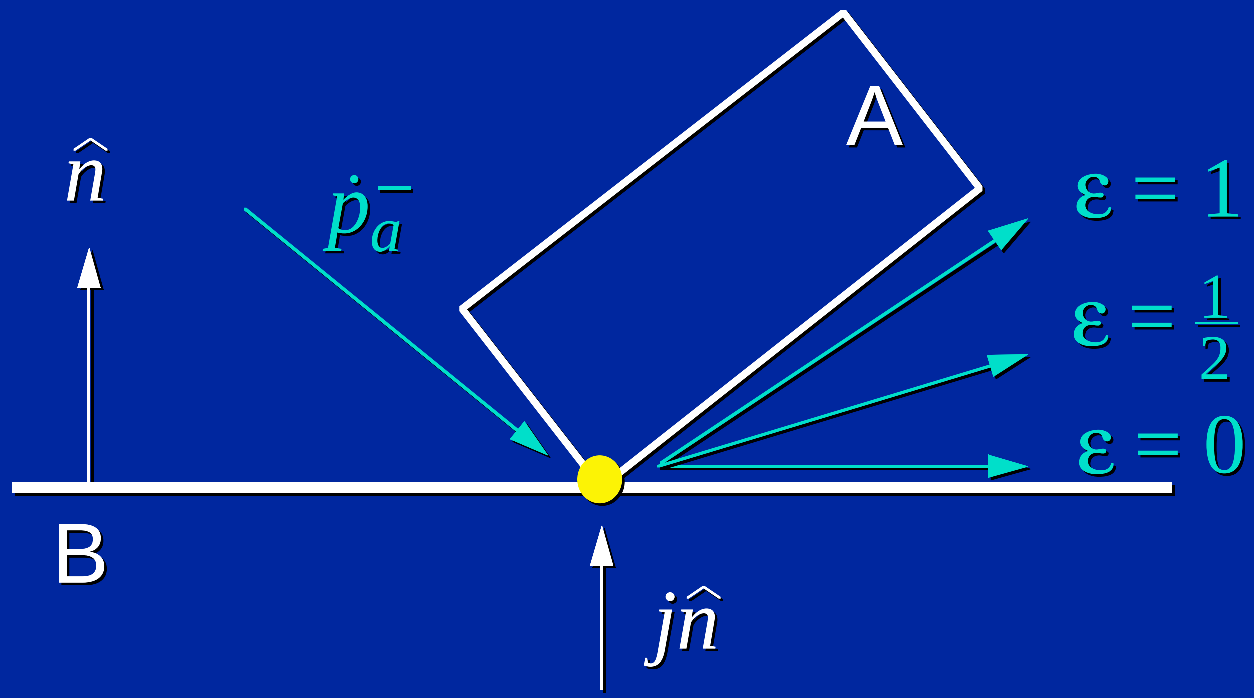
A coherence based collision detection strategy for convex polyhedra: it's simple, efficient and (relatively) easy to program.

Computing Impulses



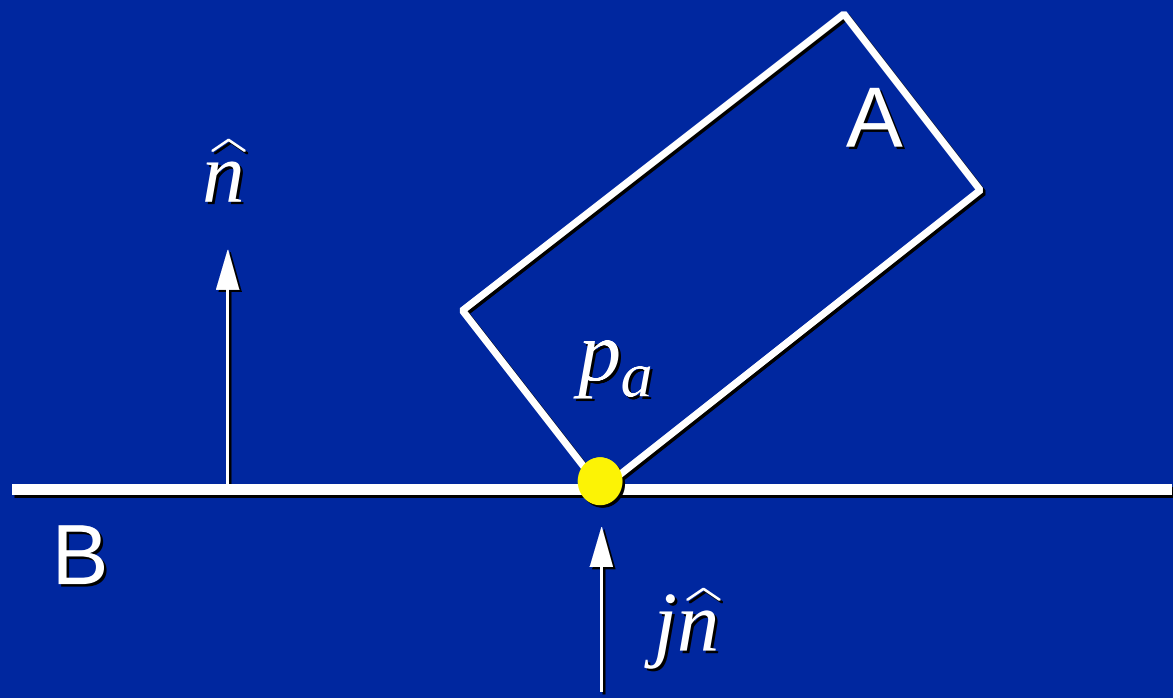
Coefficient of Restitution

$$\hat{n} \cdot \dot{p}_a^+ = -\varepsilon\left(\hat{n} \cdot \dot{p}_a^-\right)$$



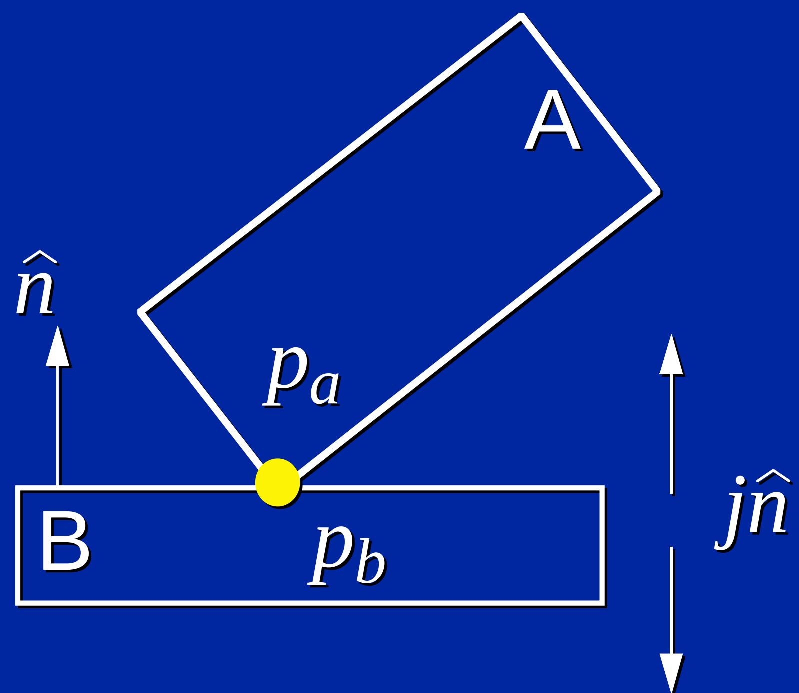
Computing j

$$\hat{n} \bullet \dot{p}_a^{+} = -\varepsilon\left(\hat{n} \bullet \dot{p}_a^{-}\right) \longrightarrow \boxed{cj + b = d}$$



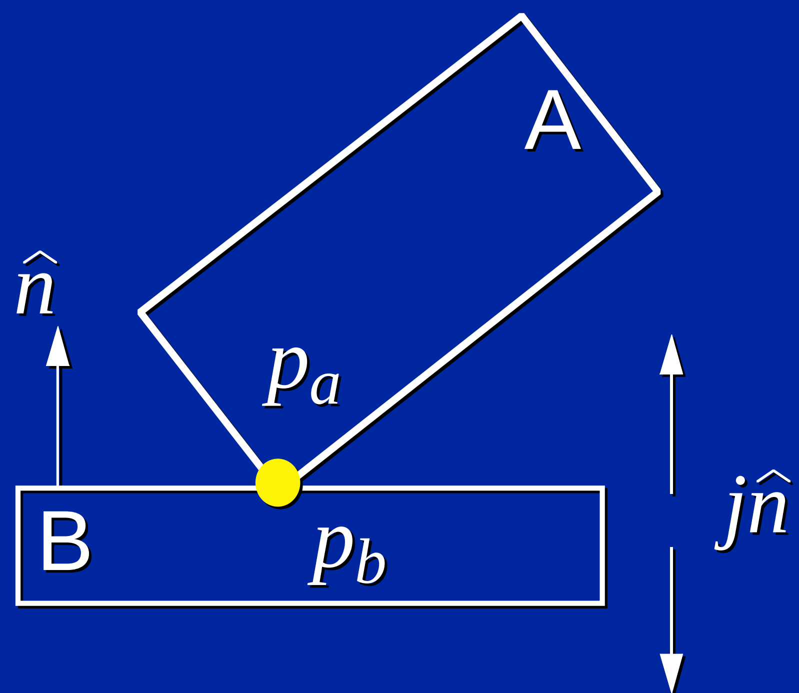
Computing j

$$\hat{n} \bullet (\dot{p}_a^+ - \dot{p}_b^+) = -\varepsilon\left(\hat{n} \bullet (\dot{p}_a^- - \dot{p}_b^-)\right)$$



Computing j

$$\hat{n} \cdot (\dot{p}_a^+ - \dot{p}_b^+) = -\varepsilon\left(\hat{n} \cdot (\dot{p}_a^- - \dot{p}_b^-)\right) \longrightarrow cj + b = d$$



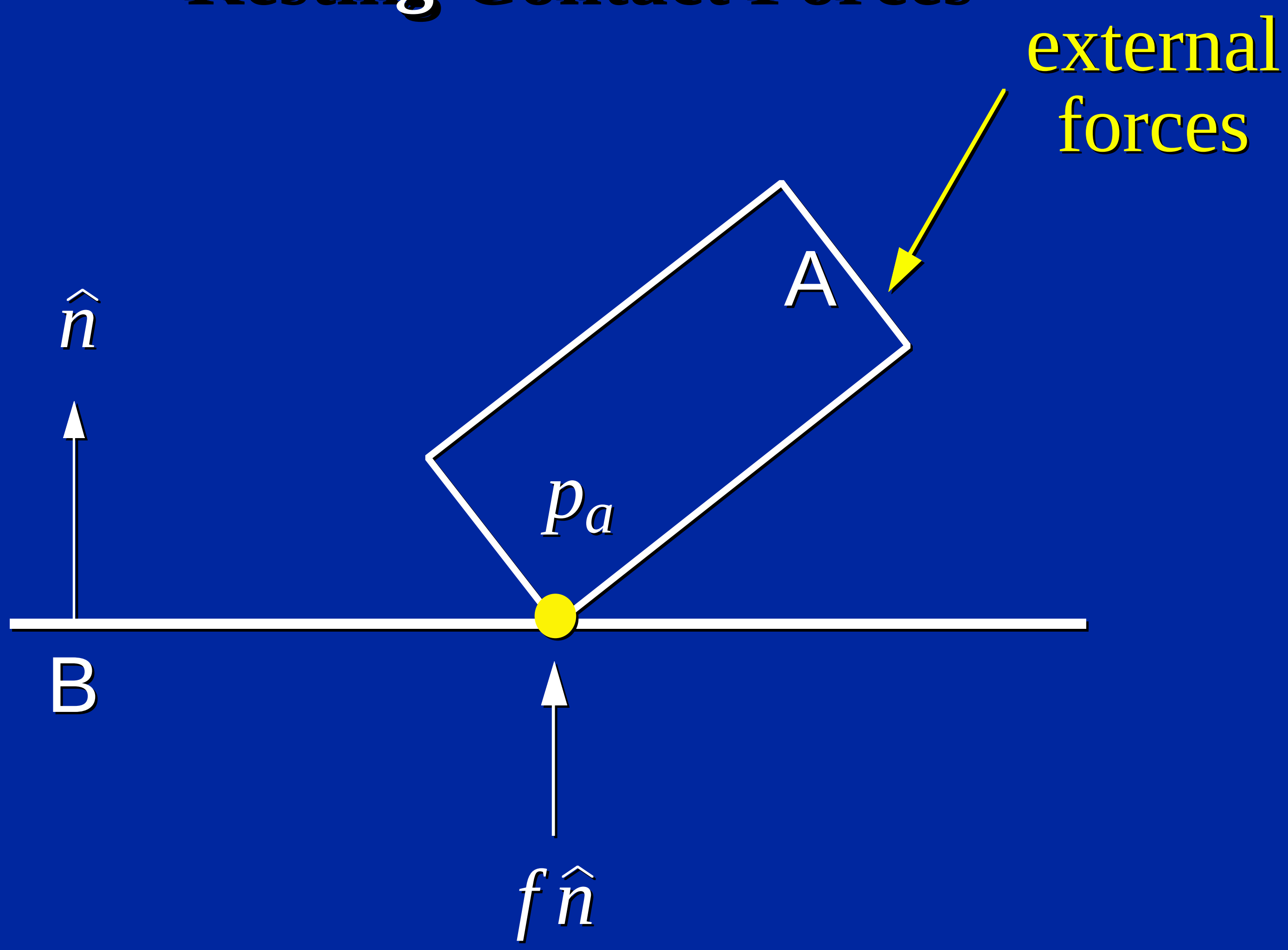
In the Course Notes – Collision Response

Data structures to represent contacts (found by the collision detection phase).

Derivations and code for computing the impulse between two colliding frictionless bodies for a particular coefficient of ε.

Code to detect collisions and apply impulses.

Resting Contact Forces



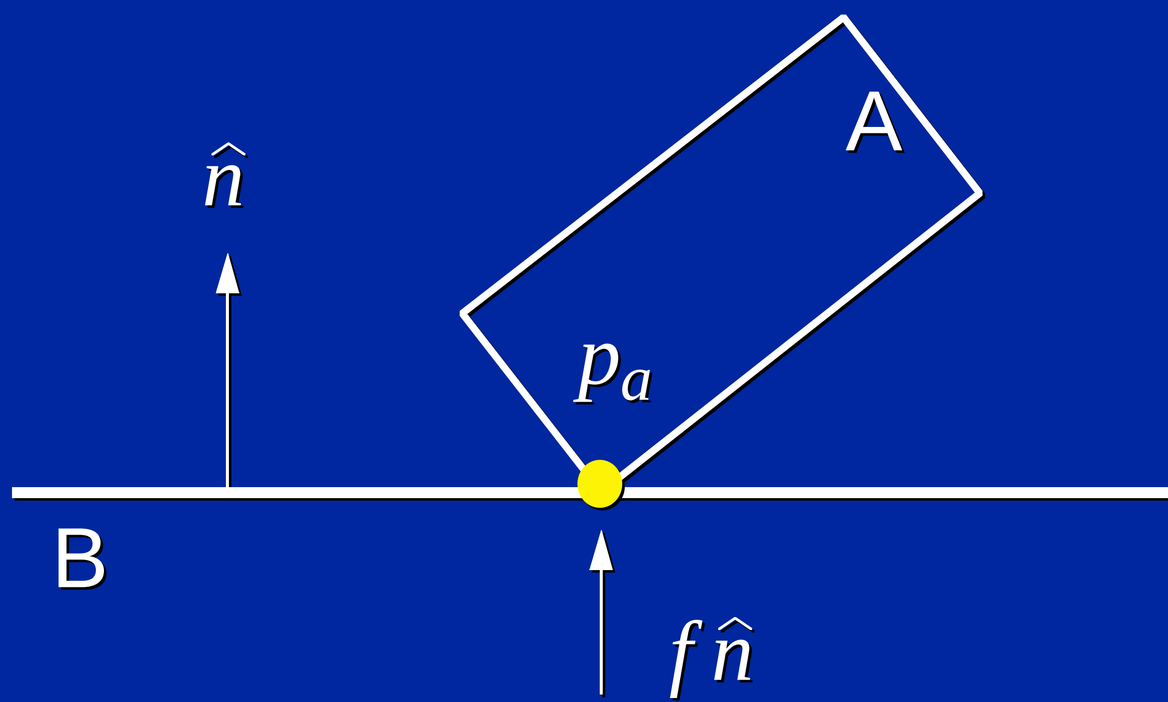
Conditions on the Constraint Force

To avoid inter-penetration, the force strength f must prevent the vertex p_a from accelerating downwards. If B is fixed, this is written as

$$\hat{n} \bullet \ddot{p}_a \geq 0$$

Computing f

$$\hat{n} \cdot \ddot{p}_a \geq 0 \longrightarrow af + b \geq 0$$



Conditions on the Constraint Force

To prevent the constraint force from holding bodies together, the force must be repulsive:

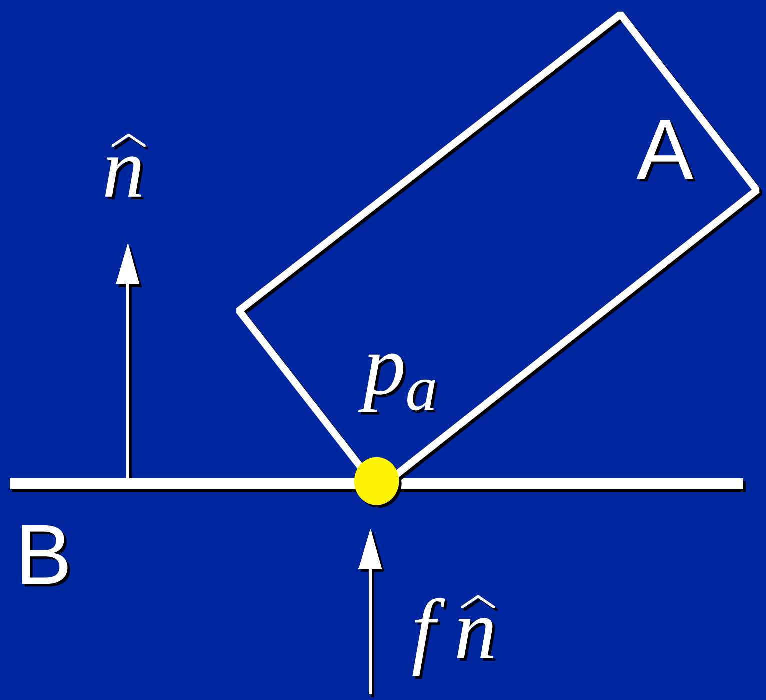
$$f \geq 0$$

Does the above, along with

$$\hat{n} \bullet \ddot{p}_a \geq 0 \longrightarrow af + b \geq 0$$

sufficiently constrain f?

Workless Constraint Force



Either

$$af + b = 0$$
$$f \geq 0$$

or

$$af + b > 0$$
$$f = 0$$

Conditions on the Constraint Force

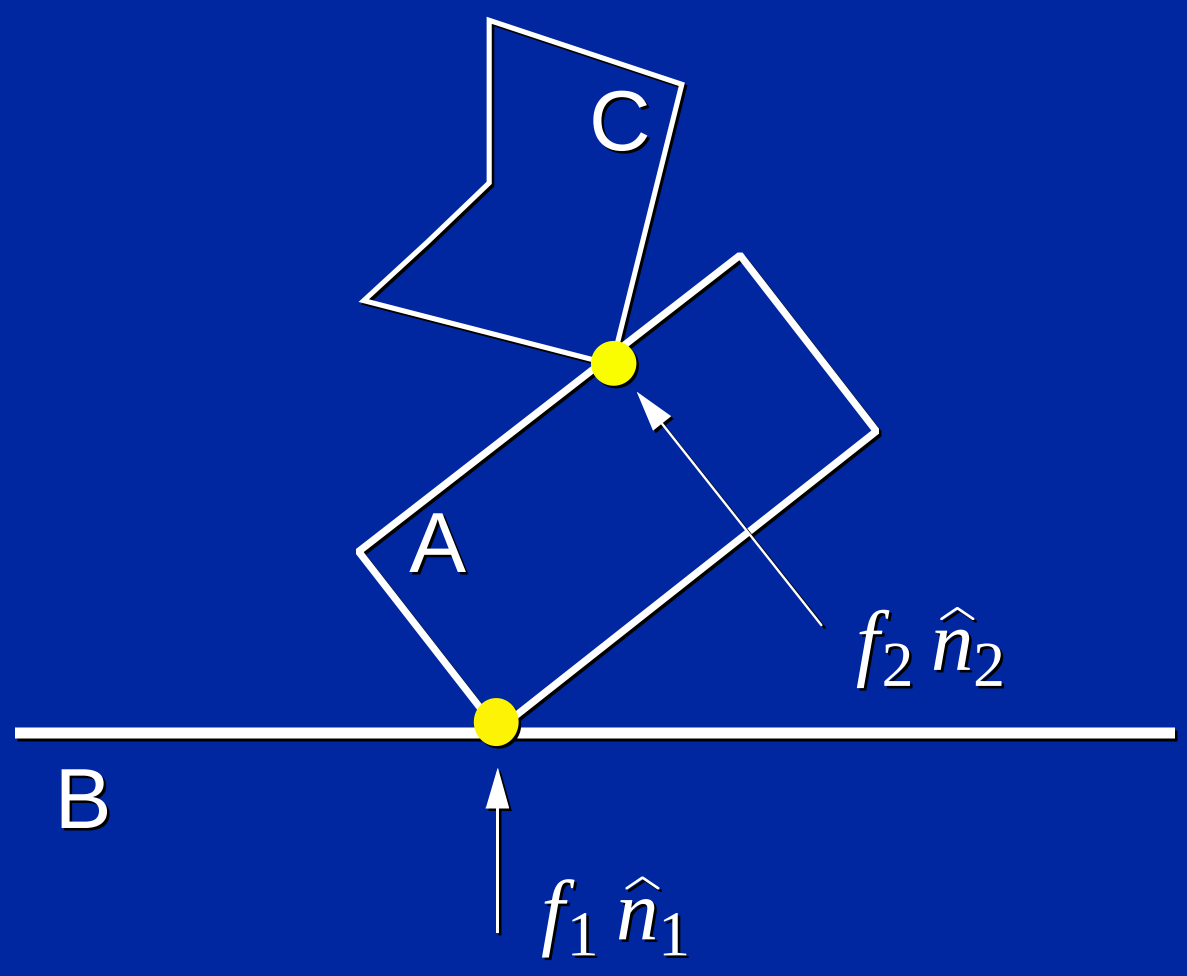
To make f be workless, we use the condition

$$f \cdot (af + b) = 0$$

The full set of conditions is

$$\begin{aligned} af + b &\geq 0 \\ f &\geq 0 \\ f \cdot (af + b) &= 0 \end{aligned}$$

Multiple Contact Points



Conditions on f_1

Non-penetration:

$$a_{11} f_1 + a_{12} f_2 + b_1 \geq 0$$

Repulsive:

$$f_1 \geq 0$$

Workless:

$$f_1 \cdot (a_{11} f_1 + a_{12} f_2 + b_1) = 0$$

Quadratic Program for f_1 and f_2

Non-penetration:

$$a_{11} f_1 + a_{12} f_2 + b_1 \geq 0$$
$$a_{21} f_1 + a_{22} f_2 + b_2 \geq 0$$

Repulsive:

$$f_1 \geq 0$$
$$f_2 \geq 0$$

Workless:

$$f_1 \cdot (a_{11} f_1 + a_{12} f_2 + b_1) = 0$$
$$f_2 \cdot (a_{21} f_1 + a_{22} f_2 + b_2) = 0$$

In the Course Notes – Constraint Forces

Derivations of the non-penetration constraints for contacting polyhedra.

Derivations and code for computing the a_{ij} and b_i coefficients.

Code for computing and applying the constraint forces $f_i \hat{n}_i$.

Quadratic Programs with Equality Constraints

Non-penetration:

$$a_{11} f_1 + a_{12} f_2 + b_1 = 0$$
$$a_{21} f_1 + a_{22} f_2 + b_2 \geq 0$$

Repulsive:

~~$f_1 \geq 0$~~

$$f_2 \geq 0$$

Workless:

$$f_1 \cdot (a_{11} f_1 + a_{12} f_2 + b_1) = 0$$ (free)
$$f_2 \cdot (a_{21} f_1 + a_{22} f_2 + b_2) = 0$$