High End Simulation in Practice: Molecular Dynamics Part 3

Severin Strobl, Harald Köstler

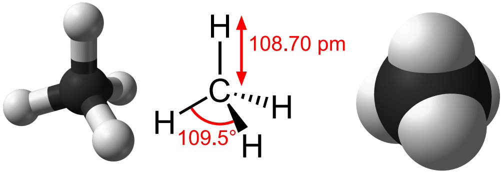
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

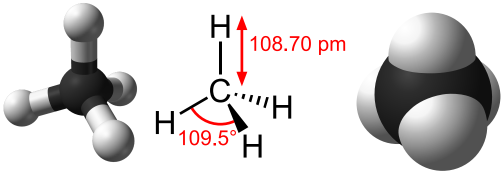
MSS

Outline

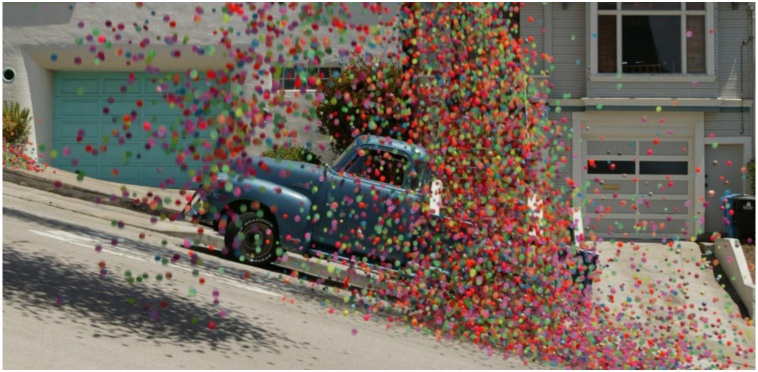
- In the previous lectures, we have ignored the **rotational degrees of freedom** of our particles.
- In MD, we can keep ignoring them if we build our molecules using proper **atomic bonding forces**:



- The problem is that these bonds vibrate at really high speeds, requiring really small time steps to solve.
- We spend all our time on the boring bonds, not the interesting molecules.



- But if we could freeze our bonds and make the molecule **rigid**, we could just ignore the bond vibrations and use a big time step.
- The problem is that the molecule is not spherical, so we need to track the rotation of the molecule in the simulation and take into account **torque**!



- If we want to simulate macroscopic systems, we need rotation even if our particles are spherical.
- **Friction** is everywhere at the macro scale, it causes balls to roll and parked cars not to.

Image taken from Sony Bravia advert.

- Also, if you want to program the more fun kind of simulations, friction can really help add to the dynamics.

Screen-shot of in-game play of Angry Birds by Rovio.

Outline

Algorithm Outline

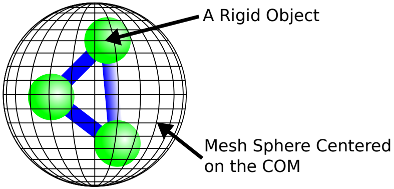
- Before, when we looked at the simplest systems, we solved Newton's equation of motion:

$$\mathbf{F}_i = m_i\,\mathbf{a}_i = \dot{\mathbf{v}}_i = \ddot{\mathbf{x}}_i$$

- Here we had a position, velocity, and acceleration/force to work with.
- But what are the variables for the rotational degrees of freedom?

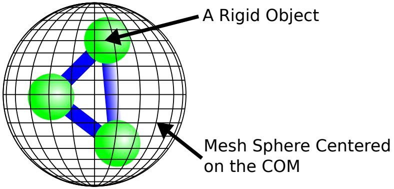
Body Fixed Coordinate System

- If we want to track the orientation of a particle, we need to first define the **body-fixed coordinate system**.
- This is some defined “initial orientation” of the particle.



- For reference, the initial orientation is chosen such that the inertia tensor is diagonal in this frame of reference.

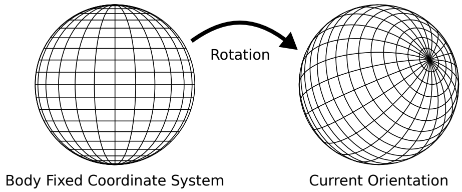
Body Fixed Coordinate System



- All rotations of an object occur around the centre of mass (COM), found by averaging location of the objects mass.
- We'll use this wire mesh sphere to represent all our rotations, but it could represent any shape.

Body Fixed Coordinate System

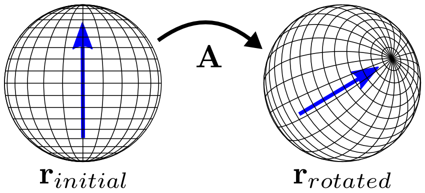
- If we want to describe the orientation of a object/particle, we can then specify it using a rotation from this coordinate system



Rotation Matrices

- Typically in mathematics, rotations are specified using a matrix:

$$\mathbf{r}_{rotated} = \mathbf{A} \cdot \mathbf{r}_{original}$$



- This matrix is the coordinate we need to track if we want to track the orientation in our simulation.

Rotation Matrices

$$\mathbf{r}_{rotated} = \mathbf{A} \cdot \mathbf{r}_{original}$$

- Rotation matrices have some interesting properties:

$\mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{1}$	Standard definition of an inverse matrix.
$\mathbf{A}^{-1} = \mathbf{A}^{T}$	The inverse is equal to the transpose.
$\det \mathbf{A} = 1$	Their determinant is 1...
$\lvert \mathbf{A} \cdot \mathbf{r} \rvert = \lvert \mathbf{r} \rvert$	...so the length of a vector is unchanged.
$\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$	Rotation matrices may be combined...
$\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$	...but the order is important.

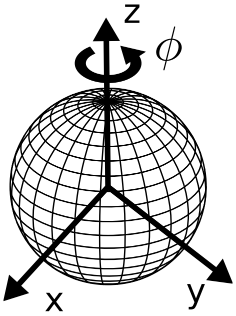
Rotation Matrices

- Do we need to track a whole 3×3 rotation matrix?
- In 3D, we can do it using just three *Euler Angles* (ϕ, θ, ψ), specifying 3 rotations around 3 axes.
- But which axes? In which order?
- Here we're using the y-convention (ZYZ), which is also used in one of the books for this lecture[1].
- So we rotate around the Z axis first, then the Y axis, and finally the Z axis again.
- This is the most confusing part in the literature...
- There are many conventions (XYZ, ZYX,. . .), but they all rotate around the axes of the coordinate system.

[1]T. Pöschel, T. Schwager, "Computational Granular Dynamics"

Rotation Matrices

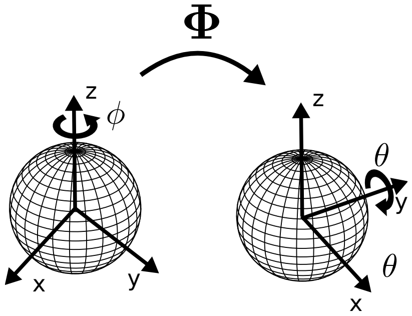
But Z-Y-Z? Twice around the Z axis? How does that work?



The rotation matrices ($\mathbf{\Phi}$, $\mathbf{\Theta}$, $\mathbf{\Psi}$) are applied one after another! We rotate around the rotated Z axis!

Rotation Matrices

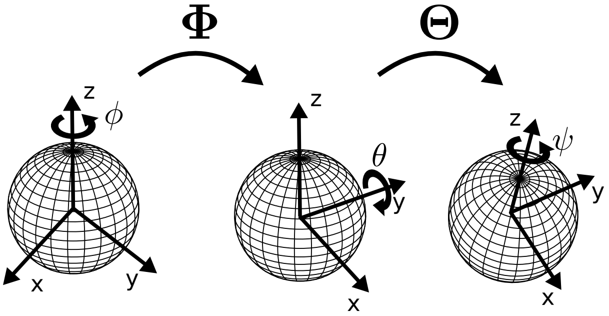
But Z-Y-Z? Twice around the Z axis? How does that work?



The rotation matrices ($\mathbf{\Phi}$, $\mathbf{\Theta}$, $\mathbf{\Psi}$) are applied one after another! We rotate around the rotated Z axis!

Rotation Matrices

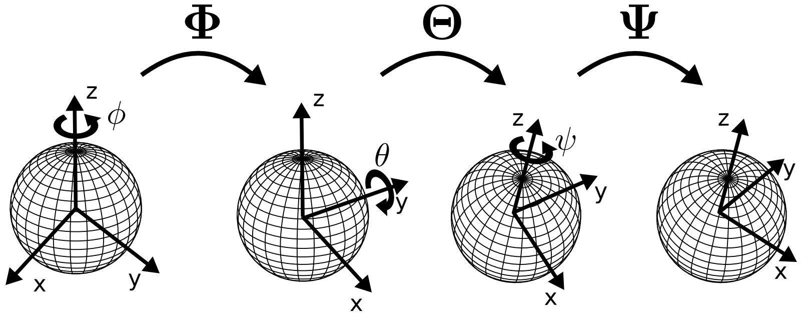
But Z-Y-Z? Twice around the Z axis? How does that work?



The rotation matrices ($\mathbf{\Phi}$, $\mathbf{\Theta}$, $\mathbf{\Psi}$) are applied one after another! We rotate around the rotated Z axis!

Rotation Matrices

But Z-Y-Z? Twice around the Z axis? How does that work?



The rotation matrices (**Φ**, **Θ**, **Ψ**) are applied one after another! We rotate around the rotated Z axis!

Rotation Matrices

- We can generate each individual axis rotation matrix from the individual Euler angles like so:

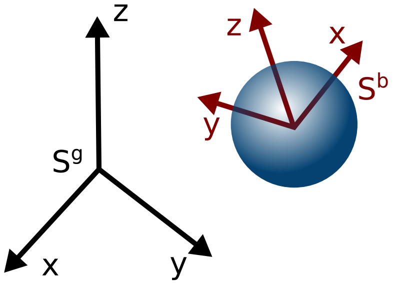
$$\mathbf{\Phi} = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{\Theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$\mathbf{\Psi} = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation Matrices

- For our simulation we need several coordinate systems:
  - A *global* (also: space-fixed or world) coordinate system S^g.
  - A *body-fixed* coordinate system S^b, centered at the COM of a particle and aligned along the particle's principal axes.
- To transform vectors between these two systems, we can use combinations of the rotation matrices $\mathbf{\Phi}$, $\mathbf{\Theta}$, and $\mathbf{\Psi}$.



Rotation Matrices

- To rotate from the body-fixed coordinates to the global coordinates (S^b to S^g), we use

$$\mathbf{A} = \mathbf{\Psi} \cdot \mathbf{\Theta} \cdot \mathbf{\Phi},$$

so we can rotate an arbitrary vector $\mathbf{r}$ using:

$$\mathbf{r}^g = \mathbf{A}\,\mathbf{r}^b = \mathbf{\Psi}\,\mathbf{\Theta}\,\mathbf{\Phi}\,\mathbf{r}^b.$$

- To rotate from global to body-fixed coordinates (S^g to S^b), we have to use the *inverse* rotations:

$$\mathbf{A}^{-1} = \mathbf{\Phi}^{-1} \cdot \mathbf{\Theta}^{-1} \cdot \mathbf{\Psi}^{-1},$$

so

$$\mathbf{r}^b = \mathbf{A}^{-1}\,\mathbf{r}^g = \mathbf{\Phi}^{-1}\,\mathbf{\Theta}^{-1}\,\mathbf{\Psi}^{-1}\,\mathbf{r}^g.$$

Rotation Matrices

- We'd like to just use these three Euler angles (ϕ, θ, ψ) in our simulation, but there is a problem called **Gimbal Lock**.
- When you try to use only three angles, there is a point on one axis where you **lose the use of another axis**!
- This occurs in our system (ZYZ) when $\theta = 0$ (no rotation around the Y axis).
- Then, the $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ rotations both rotate around the same Z axis, so we lose one degree of freedom.

Apollo 11

A well-known gimbal lock incident happened in the Apollo 11 Moon mission with its Inertial Measurement Unit (IMU).

The engineers were aware of the gimbal lock problem, but the system was designed to simply give up and freeze near the gimbal lock position.

From this point, the spacecraft would have to be manually moved away from the gimbal lock position, and the platform would have to be manually realigned using the stars as a reference.

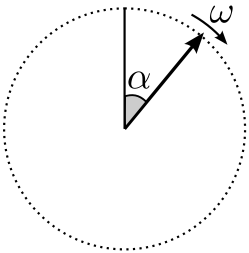
Source: `http://en.wikipedia.org/wiki/Gimbal_lock#Gimbal_lock_on_Apollo_11`

Rotation Matrices

- In reality, our object can still move in all dimensions. . .
- . . . but mathematically, we have encountered a singularity.
- Can we see this singularity in a simpler system?

Single Rotation

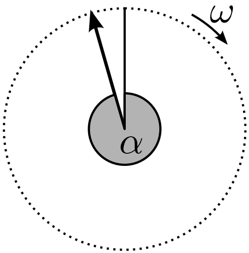
- Take a “simulation” of a clock, a “Gimbal Clock” if you will.



- We have a hand of the clock, at an angle $\alpha \in [0^\circ, 360^\circ)$.
- In our “simulation”, the clock hand is moving with an angular velocity ω.

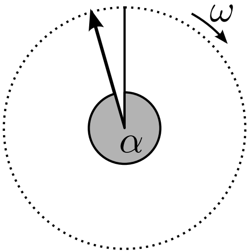
Single Rotation

- Everything runs fine in our simulation, until we approach $\alpha \to 360°$!



- What happens now?
- Keep going? (and have degenerate position to angle mapping?)
- Back to 0?

Single Rotation



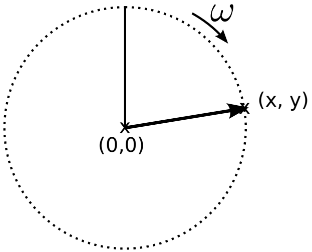
- We could have α suddenly change back to 0°, but then:

$$\lim_{\alpha \to 360^\circ} \dot{\alpha} = -\,\mathrm{sign}(\omega)\,\infty$$

- Our time derivative of α is infinite! We can't use numerical integration here.

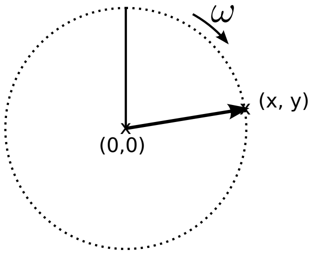
Cartesian Coordinates

- Another solution is to change to another set of coordinates...



- We've actually changed to a simple 2D Cartesian coordinate system.

Cartesian Coordinates



- We now use two variables (x, y) to describe the angle.
- But we now need to relate (x, y) to α.
- We can use the equation of a unit circle:

$$x^2 + y^2 = 1$$

Cartesian Coordinates

- We can map the polar coordinates ($\alpha(t)$) to Cartesian coordinates:

$$x(t) = \sin\alpha(t)$$
$$y(t) = \cos\alpha(t)$$

- The advantage of these coordinates is that there are no singularities for the velocities!

$$\dot{x}(t) = \omega\cos\alpha(t)$$
$$\dot{y}(t) = -\omega\sin\alpha(t)$$

- Mathematically speaking, we embedded our 1D problem into a 2D space to remove singularities.

Quaternions

- What about our original problem? We had three Euler angles (ϕ, θ, ψ) and a singularity (gimbal lock).
- Can we do the same? Lets embed our 3D problem in a 4D space using **quaternions**.

Quaternions

- Quaternions (as their name suggests) have 4 components

$$\mathbf{q} = (q_0, q_1, q_2, q_3)$$

- Just as our 1D problem used the equation of a circle (2D) of radius 1, our 3D problem will use the equation of a **hypersphere** (4D) of radius 1, a unit quaternion!

$$|\mathbf{q}| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$$

- A rotation around an axis $\mathbf{n}$ by an angle α can be expressed using a quaternion as:

$$\mathbf{q}_\alpha = \left(\cos\frac{\alpha}{2}, \mathbf{n}\sin\frac{\alpha}{2}\right)$$

Quaternions

- Just like the angle α for our Gimbal Clock:

$$1 = x^2 + y^2$$

$$x = \sin\alpha \qquad\qquad y = \cos\alpha$$

- We can relate our 3 Euler Angles (ϕ, θ, ψ) to the quaternion components:

$$1 = q_0^2 + q_1^2 + q_2^2 + q_3^2$$

$$q_0 = \cos\frac{\theta}{2}\cos\frac{\phi+\psi}{2} \qquad q_1 = \sin\frac{\theta}{2}\sin\frac{\phi-\psi}{2}$$

$$q_2 = \sin\frac{\theta}{2}\cos\frac{\phi-\psi}{2} \qquad q_3 = \cos\frac{\theta}{2}\sin\frac{\phi+\psi}{2}$$

Quaternions

- But we never need to go back to the Euler angles, from these definitions we can work out all other properties, e.g. the corresponding rotation matrix:

$$\mathbf{A} = 2\begin{pmatrix} \frac{q_0^2+q_1^2-q_2^2-q_3^2}{2} & q_1\,q_2 - q_0\,q_3 & q_1\,q_3 + q_0\,q_2 \\ q_1\,q_2 + q_0\,q_3 & \frac{q_0^2-q_1^2+q_2^2-q_3^2}{2} & q_2\,q_3 - q_0\,q_1 \\ q_1\,q_3 - q_0\,q_2 & q_2\,q_3 + q_0\,q_1 & \frac{q_0^2-q_1^2-q_2^2+q_3^2}{2} \end{pmatrix}$$

- Instead of calculating the rotation matrix from a quaternion and applying this to any vector **r** we want to transform from body to world coordinates, we can also use the quaternions directly.
- For this however we have to introduce some more quaternion algebra.

Quaternion Multiplication

- First we need a basis for the four-dimensional vector space the quaternions correspond to.
- These basis elements are commonly referred to as 1, *i*, *j*, and *k*, with

$$\begin{aligned}\mathbf{1} &= (1,0,0,0)\\ \mathbf{i} &= (0,1,0,0)\\ \mathbf{j} &= (0,0,1,0)\\ \mathbf{k} &= (0,0,0,1)\,.\end{aligned}$$

- The product of the two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ then is defined as:

$$\begin{aligned}\mathbf{q}_1\mathbf{q}_2 &= (a_0,a_1,a_2,a_3)(b_0,b_1,b_2,b_3)\\ &= (a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3,\\ &\quad\; a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2,\\ &\quad\; a_0b_2 - a_1b_3 + a_2b_0 + a_3b_1,\\ &\quad\; a_0b_3 + a_1b_2 - a_2b_1 + a_3b_0).\end{aligned}$$

Quaternion Conjugate and Inverse

- The conjugate $\mathbf{q}^*$ of a quaternion $\mathbf{q}$ is defined as:

$$\mathbf{q}^* = (q_0, q_1, q_2, q_3)^* = (q_0, -q_1, -q_2, -q_3),$$

- and the inverse $\mathbf{q}^{-1}$ as:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{|\mathbf{q}|}.$$

- Note: For a unit quaternion ($|\mathbf{q}| = 1$), as we use it for the orientation of the particles in our simulation, the inverse is obviously equal to the conjugate.

Rotations using Quaternions

- Assume we have an unit quaternion $\mathbf{q}$ describing the rotation.
- We then express the vector $\mathbf{x}_{\text{original}} = (x_0, x_1, x_2)$ we want to rotate as a quaternion $\mathbf{r} = (0, x_0, x_1, x_2)$.

$$\mathbf{r}' = \mathbf{q}\mathbf{r}\mathbf{q}^*$$

- The rotated vector $\mathbf{x}_{\text{rotated}}$ can then be extracted from the resulting quaternion $\mathbf{r}'$ as $\mathbf{x}_{\text{rotated}} = (r'_1, r'_2, r'_3)$.
- Now that we know how to express rotations using quaternions, we just need to determine how the orientation changes in time in our simulation.

Outline

Algorithm Outline

- Back to where we were, the old equation of motion was

$$\mathbf{F}_i = m_i\,\mathbf{a}_i = m_i\,\dot{\mathbf{v}}_i = m_i\,\ddot{\mathbf{x}}_i$$

- Now we need to add another equation which describes the time evolution of the orientation:

$$\boldsymbol{\tau}_i = \frac{\mathrm{d}\mathbf{L}_i}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{I}_i\boldsymbol{\omega}_i}{\mathrm{d}t}$$

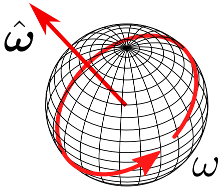
  where $\boldsymbol{\tau}_i$ is the torque acting on particle i, $\mathbf{L}_i$ is the particle's angular momentum, $\mathbf{I}_i$ is the particle's moment of inertia tensor, and $\boldsymbol{\omega}_i$ is the angular velocity.

Algorithm Outline

$$\boldsymbol{\tau}_i = \frac{\mathrm{d}\mathbf{L}_i}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{I}_i\boldsymbol{\omega}_i}{\mathrm{d}t}$$

- The torque is the angular equivalent of force.
- The inertia tensor $\mathbf{I}_i$ is the angular equivalent of mass.
- The angular velocity $\boldsymbol{\omega}_i$ is the angular equivalent of velocity!

Angular Velocity



- The **angular velocity**, ω_i, describes the rate change of the orientation $\mathbf{q}_i$ of a particle i.
- The length of $\boldsymbol{\omega}_i$ is the speed of rotation $\omega_i = |\boldsymbol{\omega}_i|$.
- And the direction defines the axis of rotation $\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i / \omega_i$.

Orientation

- It can be shown, that the rate of change of the particle's orientation $\mathbf{q}$ can directly be calculated from the angular velocity $\boldsymbol{\omega}$ as:

$$\dot{\mathbf{q}} = \frac{1}{2}\left(0, \boldsymbol{\omega}\right)\mathbf{q}.$$

- This equation will need to be integrated using a numerical integrator (just like we did for the position $\mathbf{x}$).
- After each integration step, you should renormalize the quaternion to combat numerical error:

$$\mathbf{q} \rightarrow \frac{\mathbf{q}}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} = \frac{\mathbf{q}}{|\mathbf{q}|}$$

Inertia Tensor

$$\boldsymbol{\tau}_i = \frac{\mathrm{d}\mathbf{I}_i \boldsymbol{\omega}_i}{\mathrm{d}t}$$

- The inertia tensor $\mathbf{I}_i$ is, in general, a time dependent quantity.
- In general, we would have to solve the above equation in the "body-fixed frame," as here the inertia tensor is constant and diagonal.
- But for simplicity, we will only consider spherically symmetric objects.
- Here, the inertia tensor is time independent and always simple:

$$\mathbf{I}_i = \begin{pmatrix} I_i & 0 & 0 \\ 0 & I_i & 0 \\ 0 & 0 & I_i \end{pmatrix} = I_i \, \mathbf{1} \qquad (1)$$

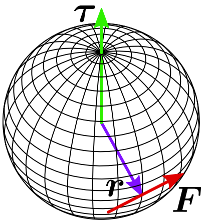
where I_i is the scalar moment of inertia of particle i.

Angular Torque

- We only have the torque left to define:

$$\boldsymbol{\tau}_i = I_i \frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t}$$

- The torque acting on an object is the moment of a force which is causing the angular velocity to change.



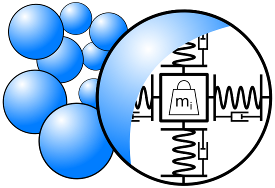
- The torque is proportional to the force **F** and the distance the force is acting from the object **r**:

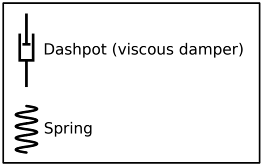
$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

- So we only need a model for some torque forces in the system and we're done.

Frictional Contact Model

- Here we will consider two particles in contact.
- The normal force is given by the Spring-Dashpot model.





$$\mathbf{F}_n^{SD}(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j) = K\,\boldsymbol{\xi} + \gamma\,\dot{\boldsymbol{\xi}}$$

where

$$\boldsymbol{\xi} = \hat{\mathbf{x}}_{ij}\,(\sigma - |\mathbf{x}_{ij}|)\,\Theta\,(\sigma - |\mathbf{x}_{ij}|)$$
$$\dot{\boldsymbol{\xi}} = -\hat{\mathbf{x}}_{ij}\,(\hat{\mathbf{x}}_{ij} \cdot \mathbf{v}_{ij})\,\Theta\,(\sigma - |\mathbf{x}_{ij}|)$$

Frictional Contact Model

- There is now an additional frictional force to consider. For this force we will use the model by Haff and Werner.
- First, we need to define what the relative surface velocity $\mathbf{v}_{ij}^{surface}$ at the contact point is.

$$\mathbf{v}_{ij}^{surface} = \mathbf{v}_{ij} - \hat{\mathbf{x}}_{ij}\left(\hat{\mathbf{x}}_{ij} \cdot \mathbf{v}_{ij}\right) + \hat{\mathbf{x}}_{ij} \times \left(R_i\,\boldsymbol{\omega}_{\mathbf{i}} + R_j\,\boldsymbol{\omega}_{\mathbf{j}}\right)$$

where R_i is the radius of particle i.

- The first term removes the normal component of the velocity from the relative velocity.
- The second line adds on the surface velocity due to particle rotation.

Frictional Contact Model

- With the relative surface velocity defined, we can define the tangential force:

$$\mathbf{F}_t = -\hat{\mathbf{v}}_{ij}^{surface} \min\left(\gamma^t \left|\mathbf{v}_{ij}^{surface}\right|, \mu \left|\mathbf{F}_n\right|\right)$$

  where μ is the coefficient of friction and γ^t controls the velocity dependent friction term.
- This force is used twice, it is added to the normal force to give the linear force between two particles:

$$\mathbf{F}_{ij}\left(\mathbf{r}_{ij}, \mathbf{v}_{ij}\right) = \mathbf{F}_n\left(\mathbf{r}_{ij}, \mathbf{v}_{ij}\right) + \mathbf{F}_t\left(\mathbf{r}_{ij}, \mathbf{v}_{ij}\right)$$

- And again to calculate the torque acting on the particles:

$$\boldsymbol{\tau}_{ij}\left(\mathbf{r}_{ij}, \mathbf{v}_{ij}\right) = -R_i\, \hat{\mathbf{r}}_{ij} \times \mathbf{F}_t\left(\mathbf{r}_{ij}, \mathbf{v}_{ij}\right)$$

Summary

- For every particle, we need a quaternion $\mathbf{q}$, an angular velocity $\boldsymbol{\omega}$ and a torque $\boldsymbol{\tau}$.
- At each time step, we sum up the normal and tangential forces on every particle i for $\mathbf{F}_i$, and we also sum up the torque $\boldsymbol{\tau}_i$ added by each tangential force.
- And we integrate the angular acceleration using

$$\frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t} = I_i^{-1}\boldsymbol{\tau}_i.$$

- We then integrate the change in orientation using

$$\dot{\mathbf{q}} = \frac{1}{2}\,(0, \boldsymbol{\omega})\,\mathbf{q}.$$