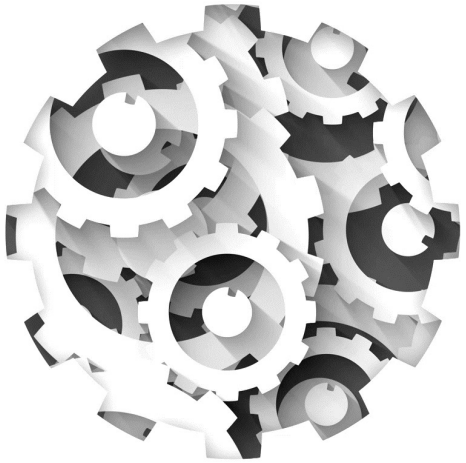
Phase transition in active spinners

A May 6 Presentation

Arya K. Rajan, Jimmy Gonzalez Nunez, Subhaya Bose

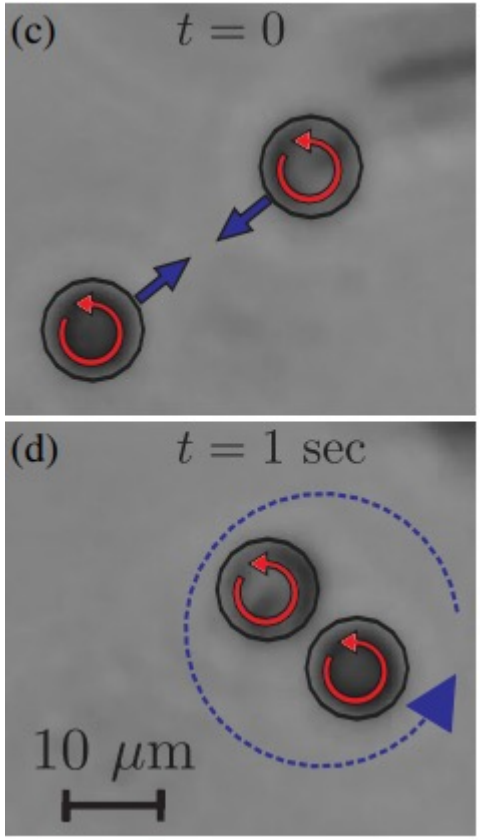
UC MERCED

Physics 230

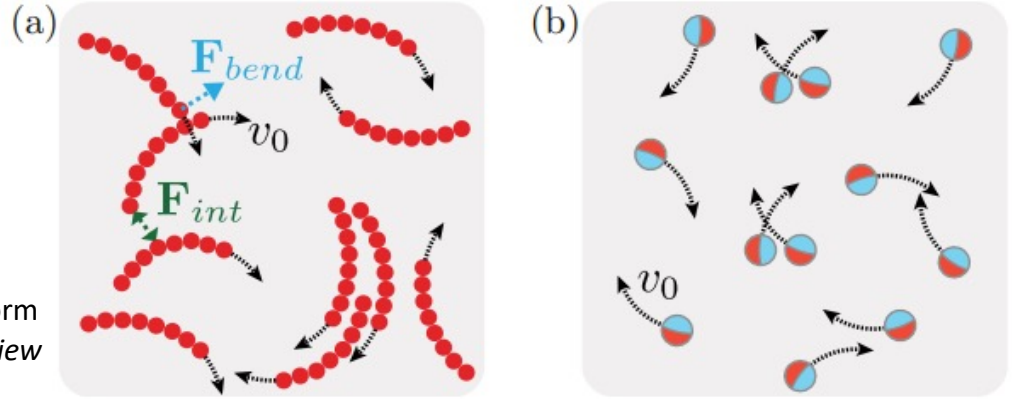


Background

- Extensive study of particles with propulsion.
- Particles with active rotation not explored as much.
- Examples – spinning microorganisms, treadmilling proteins, microtubule aggregates, shaken chiral grains, chiral colloids, liquid crystals, synthetic cilia driven by rotatory motors.



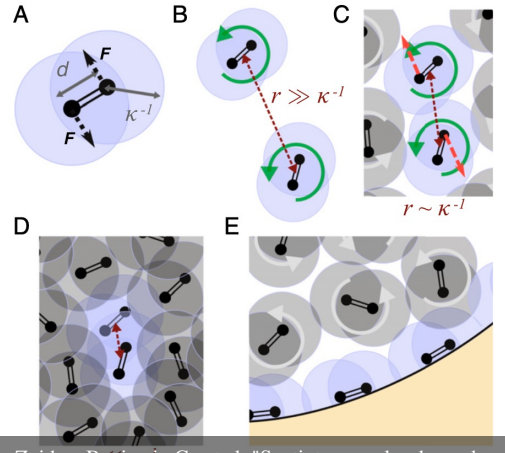
Petroff, Alexander P., Xiao-Lun Wu, and Albert Libchaber. "Fast-moving bacteria self-organize into active two-dimensional crystals of rotating cells." *Physical review letters* 114.15 (2015): 158102.



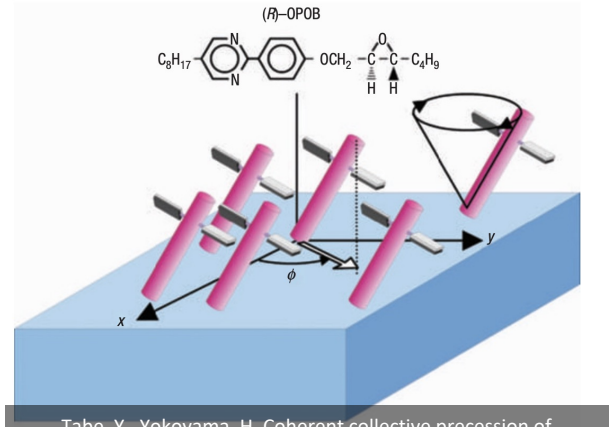
Denk, Jonas, et al. "Active curved polymers form vortex patterns on membranes." *Physical Review Letters* 116.17 (2016): 178301.

Background

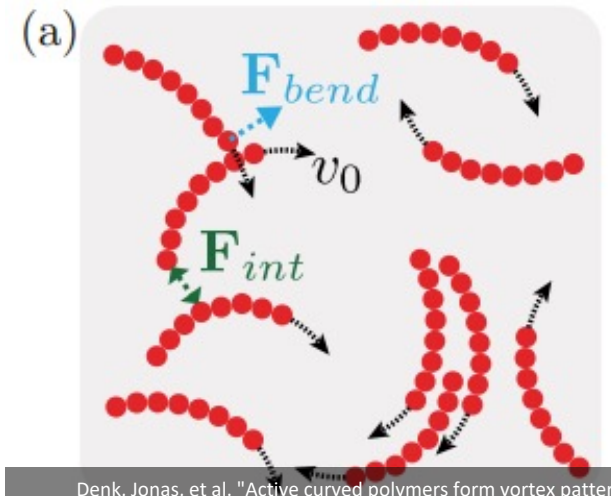
- Active rotation in absence of self propulsion
- Geometric frustration of spinner rotation by interaction – melt under increasing pressure
- Low density – free spinning – monopole like interactions
- Higher density – short range multipole gear like interactions – competition between active rotation and interaction, resulting in melting

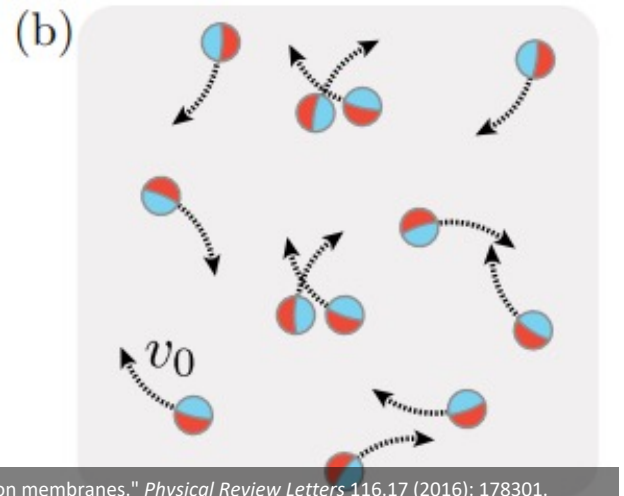


van Zuiden, Benjamin C., et al. "Spatiotemporal order and emergent edge currents in active spinner materials." *Proceedings of the national academy of sciences* 113.46 (2016): 12919-12924.



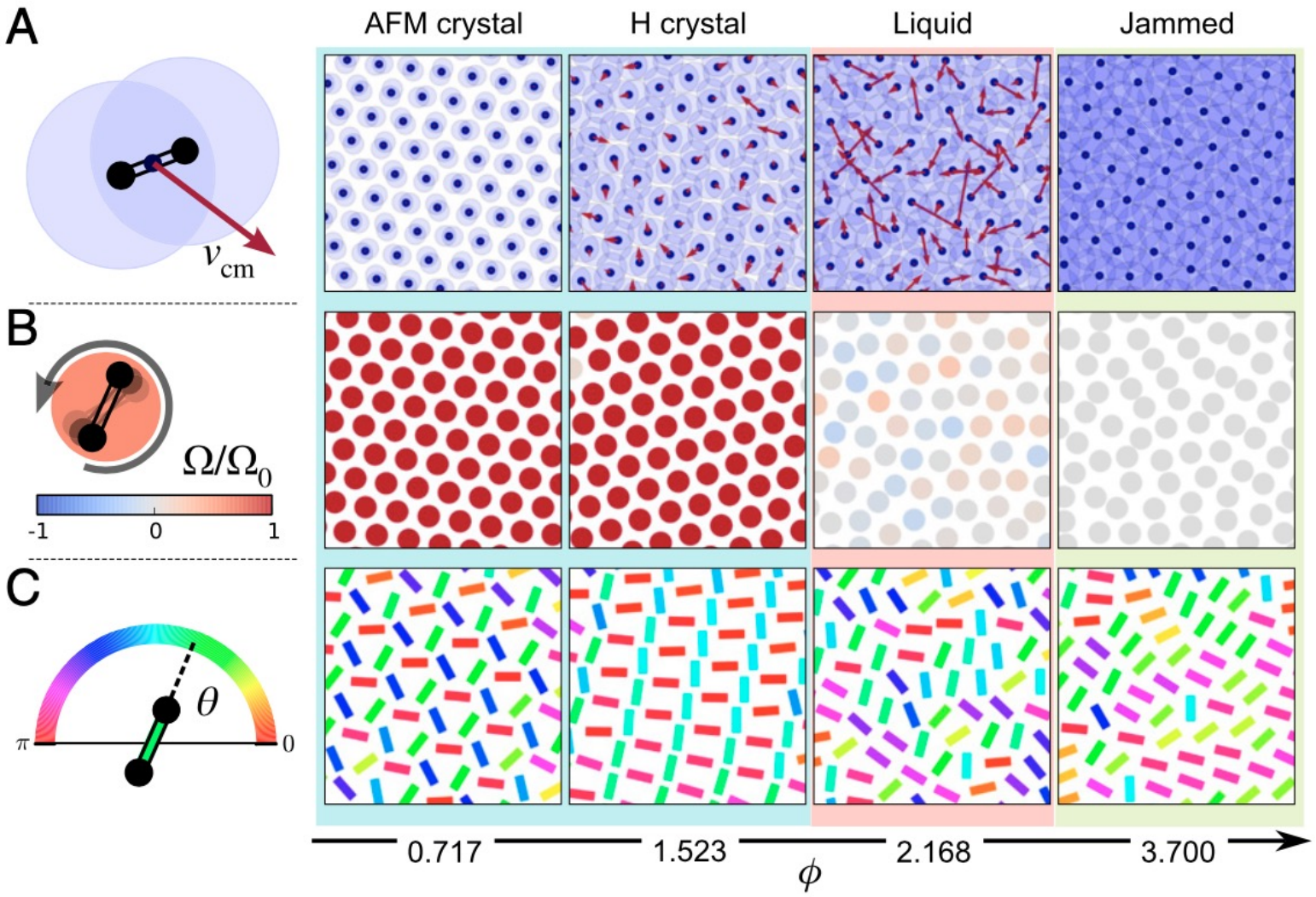
Tabe, Y., Yokoyama, H. Coherent collective precession of molecular rotors with chiral propellers. *Nature Mater* **2,** 806–809 (2003). https://doi.org/10.1038/nmat1017

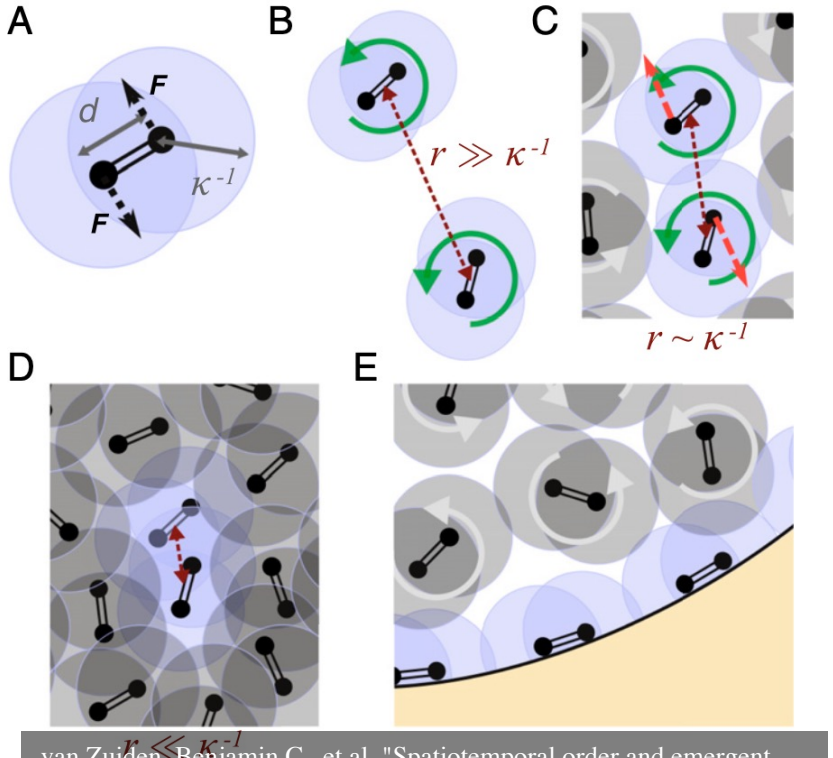




Denk, Jonas, et al. "Active curved polymers form vortex patterns on membranes." *Physical Review Letters* 116.17 (2016): 178301.

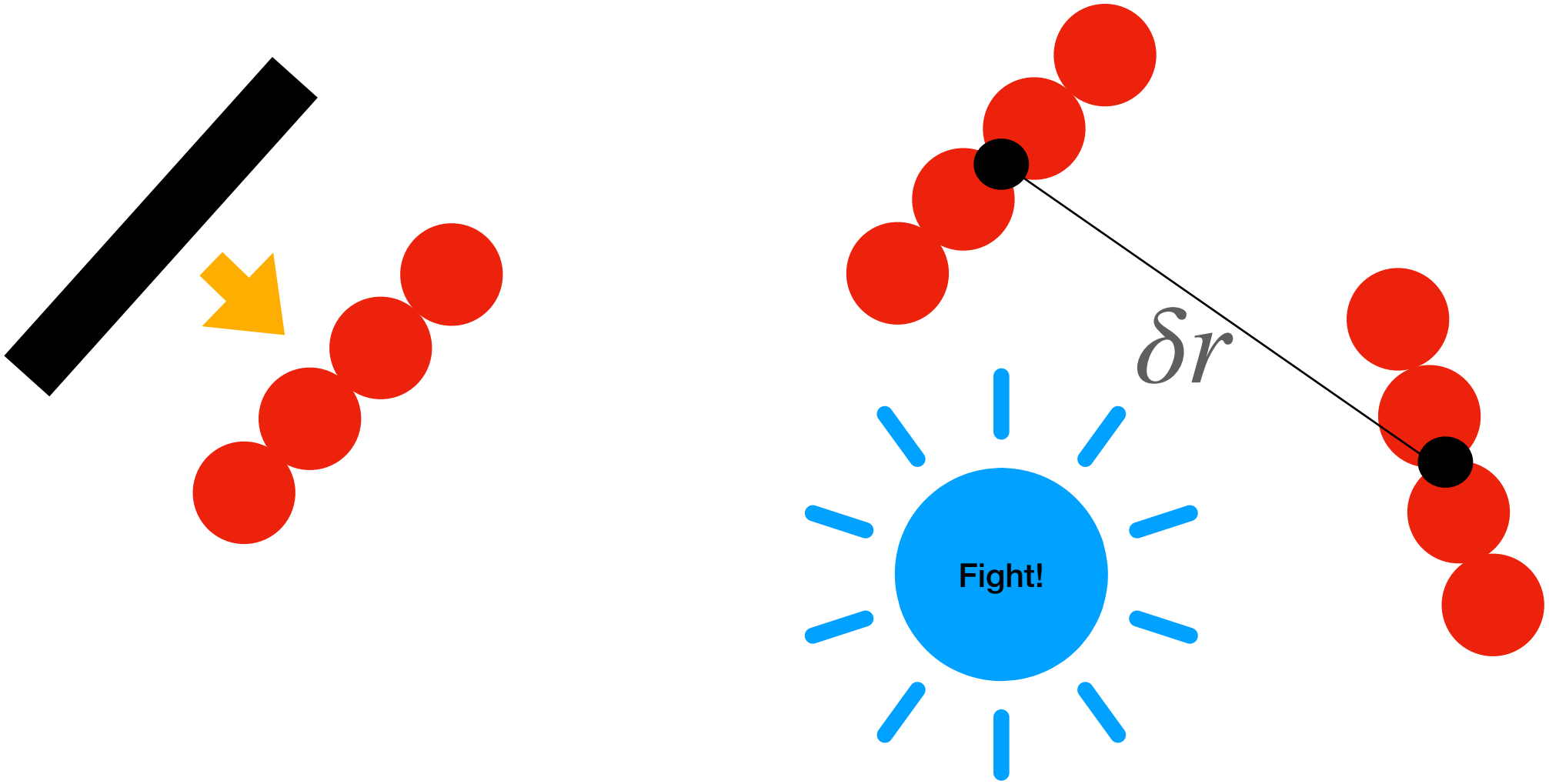
Dimer Interactions



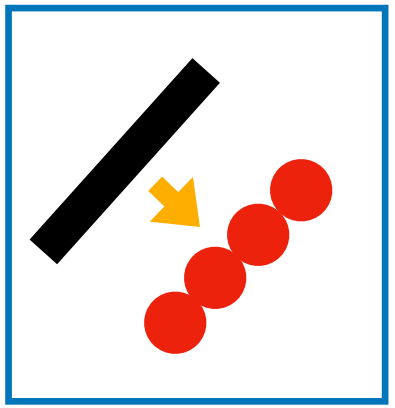


van Zuiden, Benjamin C., et al. "Spatiotemporal order and emergent edge currents in active spinner materials." *Proceedings of the national academy of sciences* 113.46 (2016): 12919-12924.

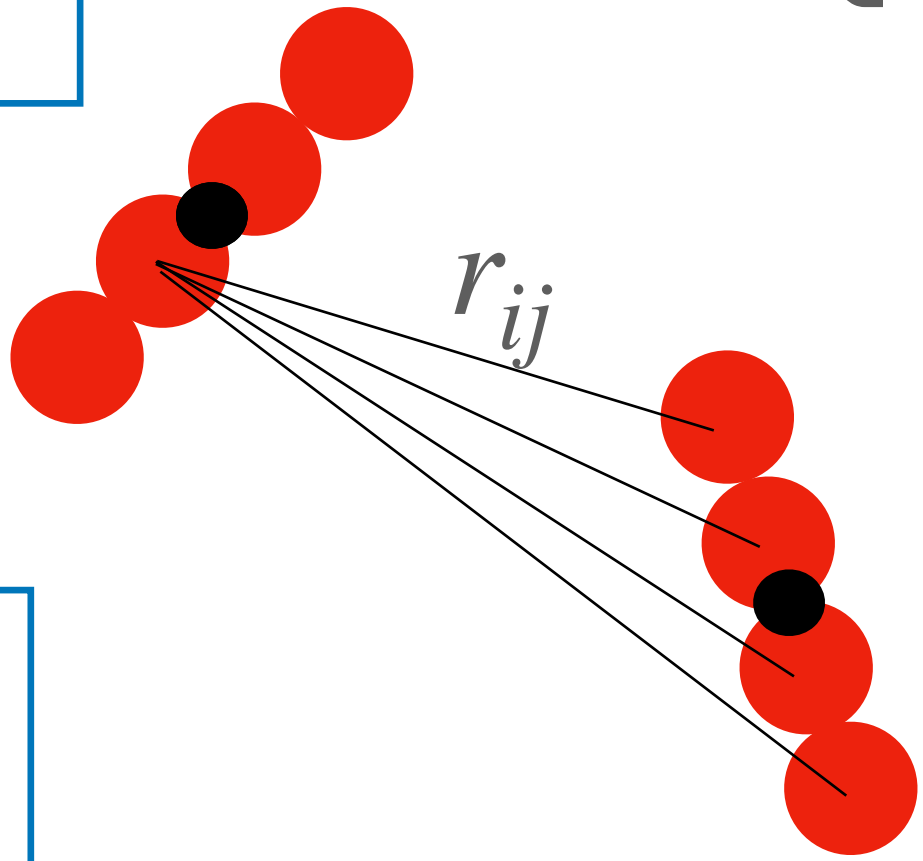
Modeling Active Spinners



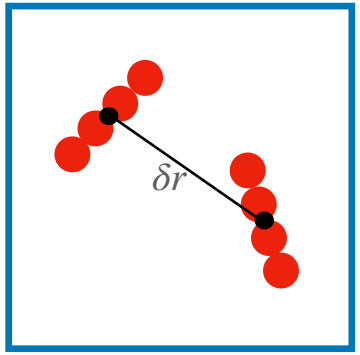
Modeling Active Spinners



$$F = \begin{cases} f_{spring}(r_{ij}) = k(\sigma - r_{ij}), & r_{ij} < \sigma \\ f_{yukawa}(r_{ij}) = e^{-\frac{r}{\sigma}} \dfrac{1 + \sigma r}{r^2}, & \sigma < r_{ij} < 4\sigma \end{cases}$$



$$f^i_{\alpha\beta} = \sum_j f(r_{ij})\hat{r}_{ij}$$



Internal Torque - א

External Force – $F_{Spring} = -k\Delta \mathrm{r}$

$F_{Yukawa} = b\,\exp(-\kappa r)$

Forces on active spinning rods

- Each rod is a stiff chain of beads.
- Rods have an internal torque which drives them to spin at uniform angular velocity.
- steric and short-range forces from other particles responsible for reorientation of particles.

Modeling Active Spinners

Langevin equations for over damped system

$$\frac{\partial \theta_\alpha}{\partial \tau} = \Omega_0 + \frac{\sigma^2 \gamma}{\gamma_\Omega} \sum_i \vec{r}^{\,i}_\alpha \times f^i_{\alpha\beta}$$

$$\frac{\partial \vec{r}_\alpha}{\partial \tau} = \sum_i f^i_{\alpha\beta}$$

- Each rod is considered to be a stiff chain of beads.
- Rods have an internal torque which drives them to spin at uniform angular velocity.
- Contact or hydrodynamic radius from other particles responsible for reorientation of particles.

$$U(\vec{r}) = A\frac{e^{-r/\lambda}}{r}$$

Psuedo-Code

The Implementation

```
class Particle:
    # center of mass
    # position of beads
    # orientation

def distance(ptcla, ptclb, cut_off, BoxL):
    # distance between particls


def interacting_pairs(ptcls, idxs, cut_off, BoxL):
    # get unique pairs of idices
    # remove pairs whose dist > cut_off


def unitvector(v):
    return v / (v ** 2).sum() ** 0.5

def nodeRepulsion(ptcla, ptclb, lamb, idx, BoxL, dt):
    # node repulsion of ith bead from all beads in particle b
```

```
def unitvector(v):
    return v / (v ** 2).sum() ** 0.5


def crossprod_3(rx, ry, fx, fy):
    return rx * fy - ry * fx

def update(ptcl, sigma, alpha, dt, BoxL, D_T, D_R, rat_gam):
    # get change in center of mass
    # get torque
    # get updated positions and angle

def interact(ptcla, ptclb, lamb, BoxL, dt):
    nodeRepulsion(ptcla, ptclb, lamb, i, BoxL, dt)


def metric(x, BoxL):
    # distance between points on a donut

def force(r, lamb, dt):
    # spring force if r < sigma
    # yukawa force if sigma < r < 4 *sigma

def separation(rx, ry):
    #  r0, hatr0x, hatr0y
```

Psuedo-Code

The Implementation

```
for t in range(t_range):
    for a, b in interacting_pairs:
        interact(a, b)
    ptcls = [update(ptcl) for ptcl in ptcls]
```

Psuedo-Code

The Implementation

Implementing Constraints for Individual Particles

```
class Particle:
    def __init__(self, x, y, sigma, theta, BoxL, omega=1.0, vx=1., vy=1.):
        self.cmx = np.mod(x, BoxL)
        self.cmy = np.mod(y, BoxL)
        self.vx = vx
        self.vy = vy
        self.theta = theta
        self.omega = omega
        self.rsx = np.mod(array([-1.5, -0.5, 0.5, 1.5]) * sigma * cos(theta) +
        x, BoxL)
        self.rsy = np.mod(array([-1.5, -0.5, 0.5, 1.5]) * sigma * sin(theta) +
        y, BoxL)
        self.fx = array([0.0, 0.0, 0.0, 0.0])
        self.fy = array([0.0, 0.0, 0.0, 0.0])
```

Psuedo-Code

The Implementation

Implementing Forces between Particles

```
def nodeRepulsion(ptcla, ptclb, lamb, idx, BoxL, dt):
    rx = -(ptclb.rsx - ptcla.rsx[idx])
    ry = -(ptclb.rsy - ptcla.rsy[idx])
    rx = fromiter((metric(x, BoxL) for x in rx), rx.dtype, count=4)
    ry = fromiter((metric(y, BoxL) for y in ry), ry.dtype, count=4)
    r0, hatr0x, hatr0y = separation(rx, ry)
    f0 = fromiter((force(r, lamb, dt) for r in r0), r0.dtype, count=4)
    f0xs = f0 * hatr0x
    f0ys = f0 * hatr0y
    ptcla.fx[idx] += f0xs.sum()
    ptclb.fx -= f0xs
    ptcla.fy[idx] += f0ys.sum()
    ptclb.fy -= f0ys
```

Psuedo-Code

The Implementation

Implementing Pairwise Interaction between Closely Spaced Particles

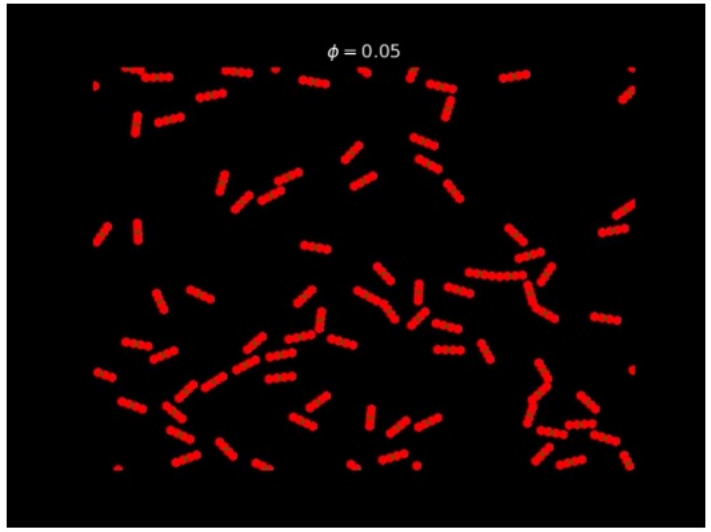
```
def interacting_pairs(ptcls, idxs, cut_off, BoxL):
    # get unique pairs of idices
    cmbtns = combinations(idxs, 2)
    # remove pairs whose dist > cut_off
    return (x for x in cmbtns if distance(ptcls[x[0]], ptcls[x[1]], cut_off,
    BoxL))
```

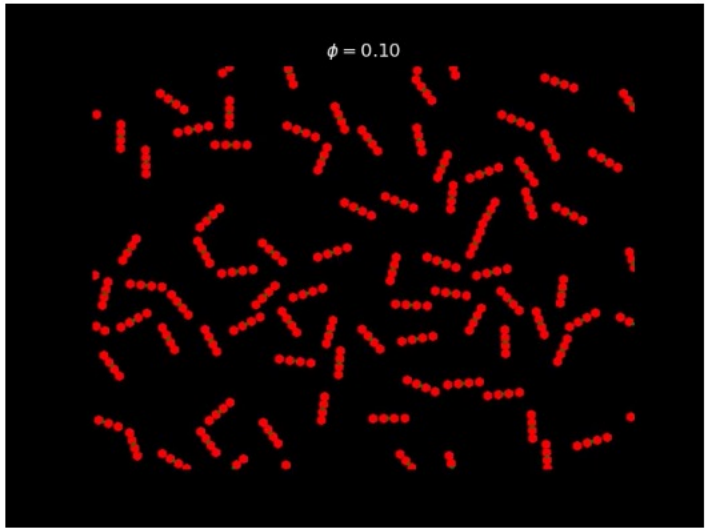
Self-spinning, Particles repel each other with a Yukawa interaction

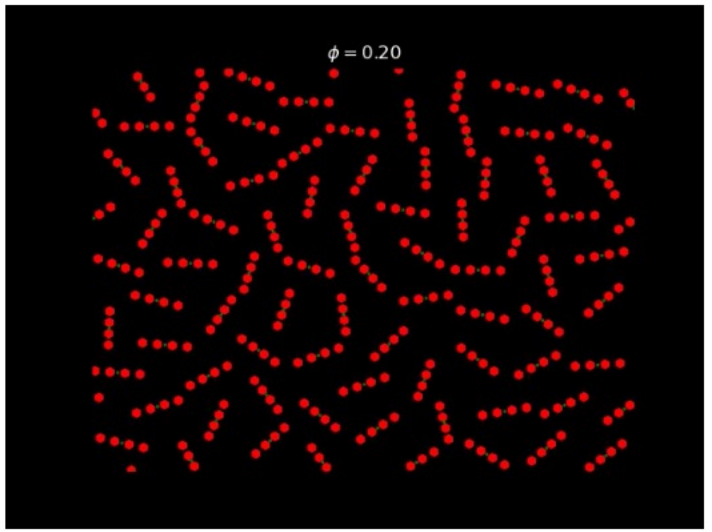
Two particles free to translate
as well as rotate.

Particle interaction and orientation in small packing fraction

- Active spinners attain steady state rotational speed caused by background friction
- Lower packing fraction leads to separation of the particles and non correlated rotation
  ($\phi = 0.05\ , 0.10\ , 0.20$)

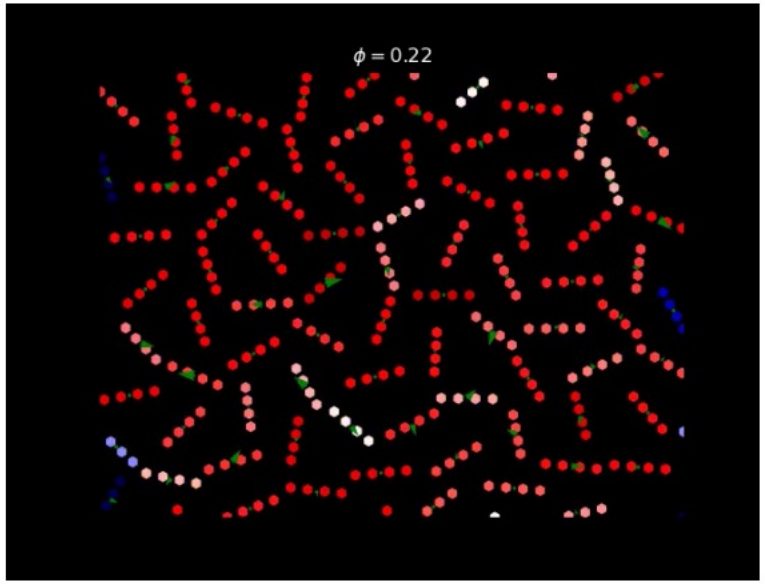


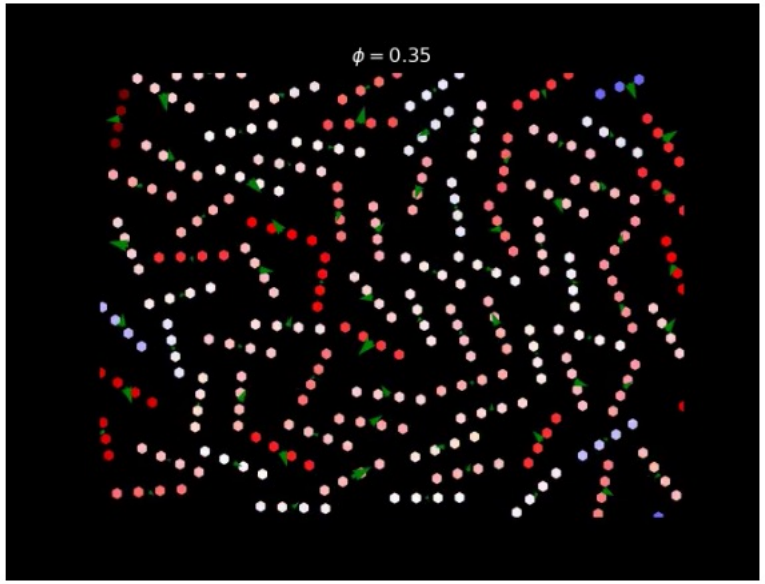




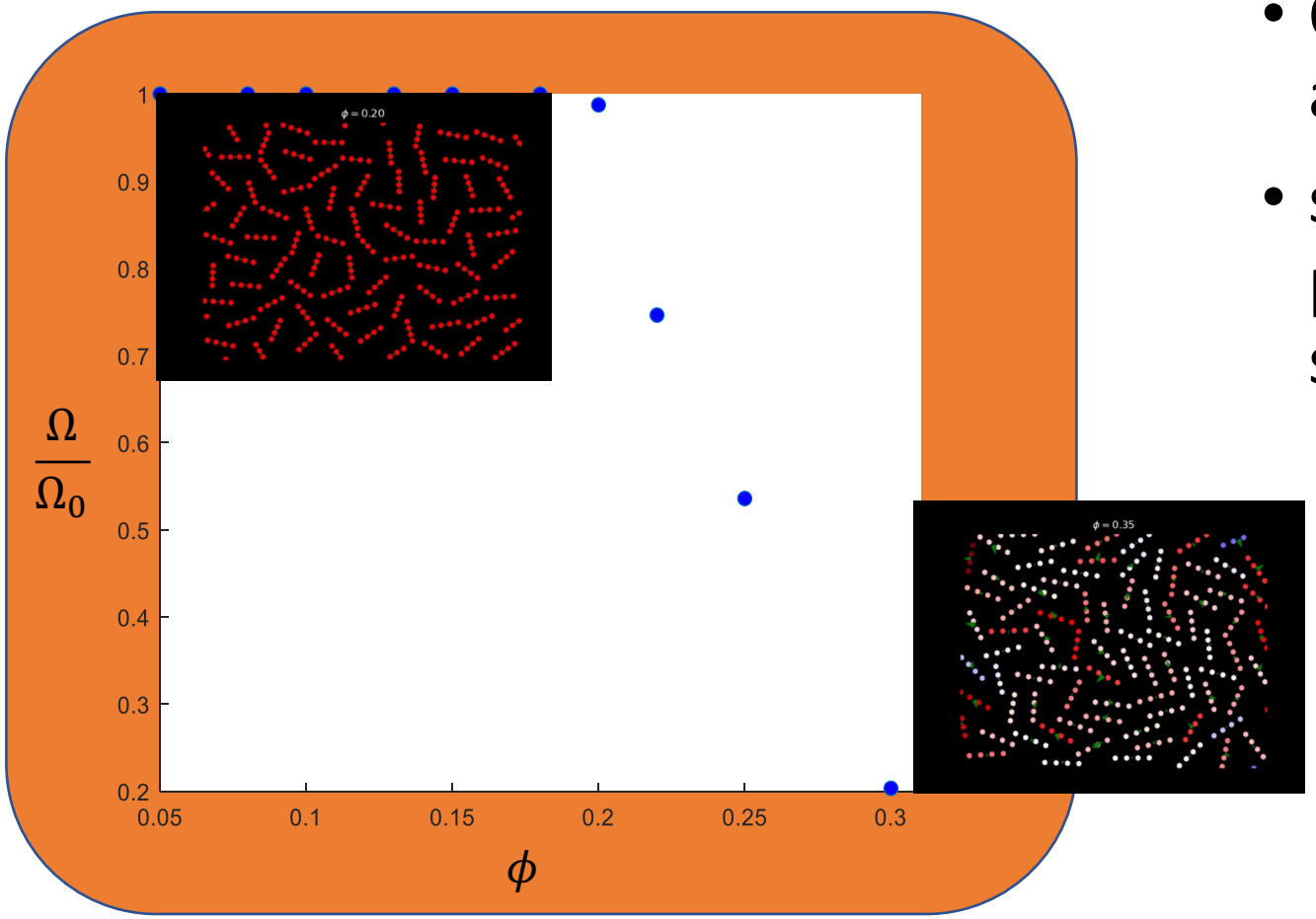
Particle interaction and orientation in higher packing fraction

- As separations become comparable with the screening length, adjacent particles still rotate past each other but experience interaction forces that depend on their instantaneous orientations
- Torque from nearby particles decrease the rotational speed for higher packing fraction. ($\phi = 0.22\ , 0.35$)





Phase Transition in Angular Rotation Speed



- Collisions reduce directed angular rotation
- sharp transition is observed at packing fraction 0.22 where the system starts to melt.

Summary

- Rods slide past each other due to active rotation and repelling force from nearby rods
- Active rotation of rods at high packing fraction results in ordering of rods.

Future work

- Explore even higher packing fraction ($\phi = 0.4 - 0.7$)
- Check order parameter for better understanding of different phases and phase transition.