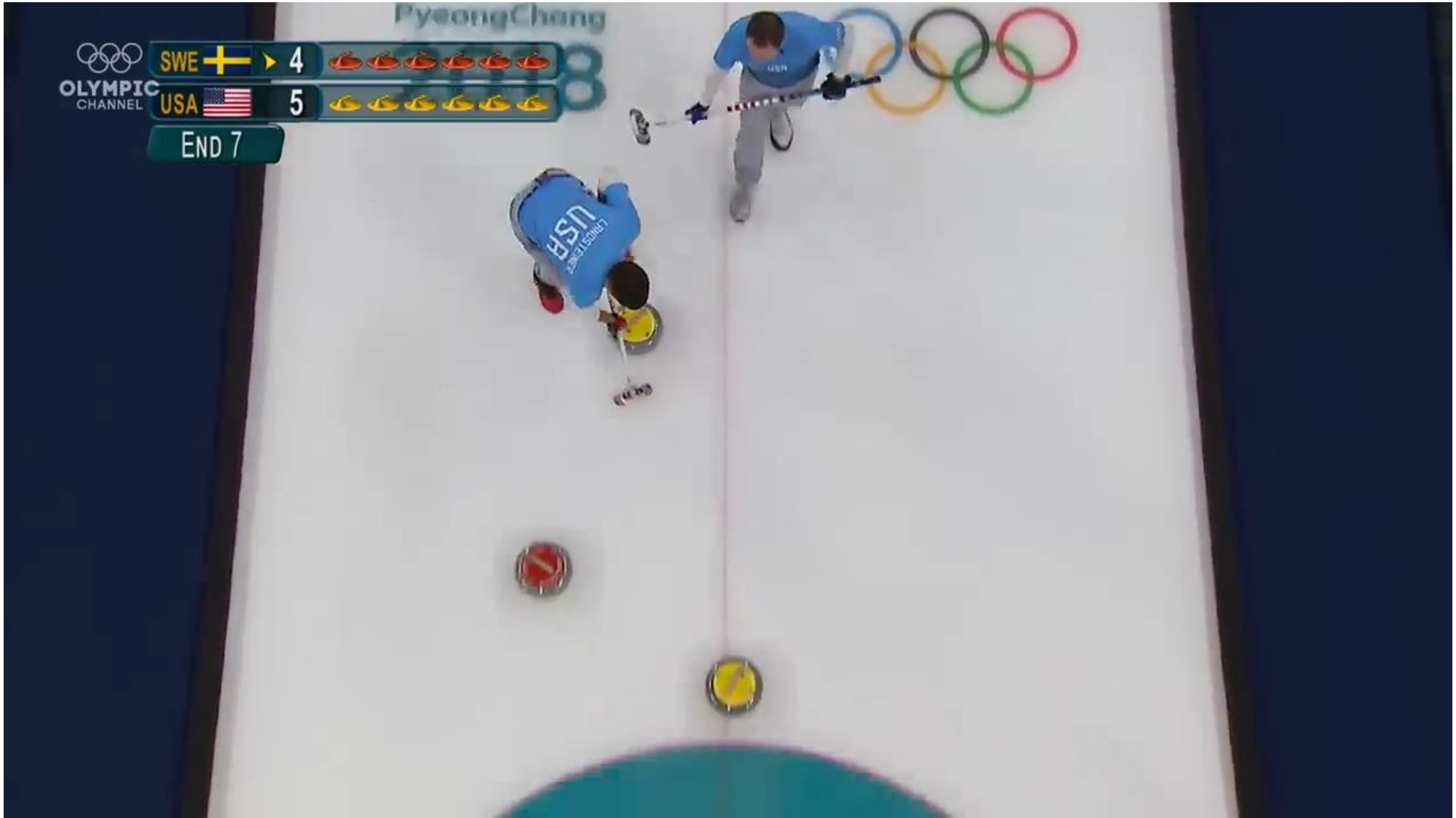
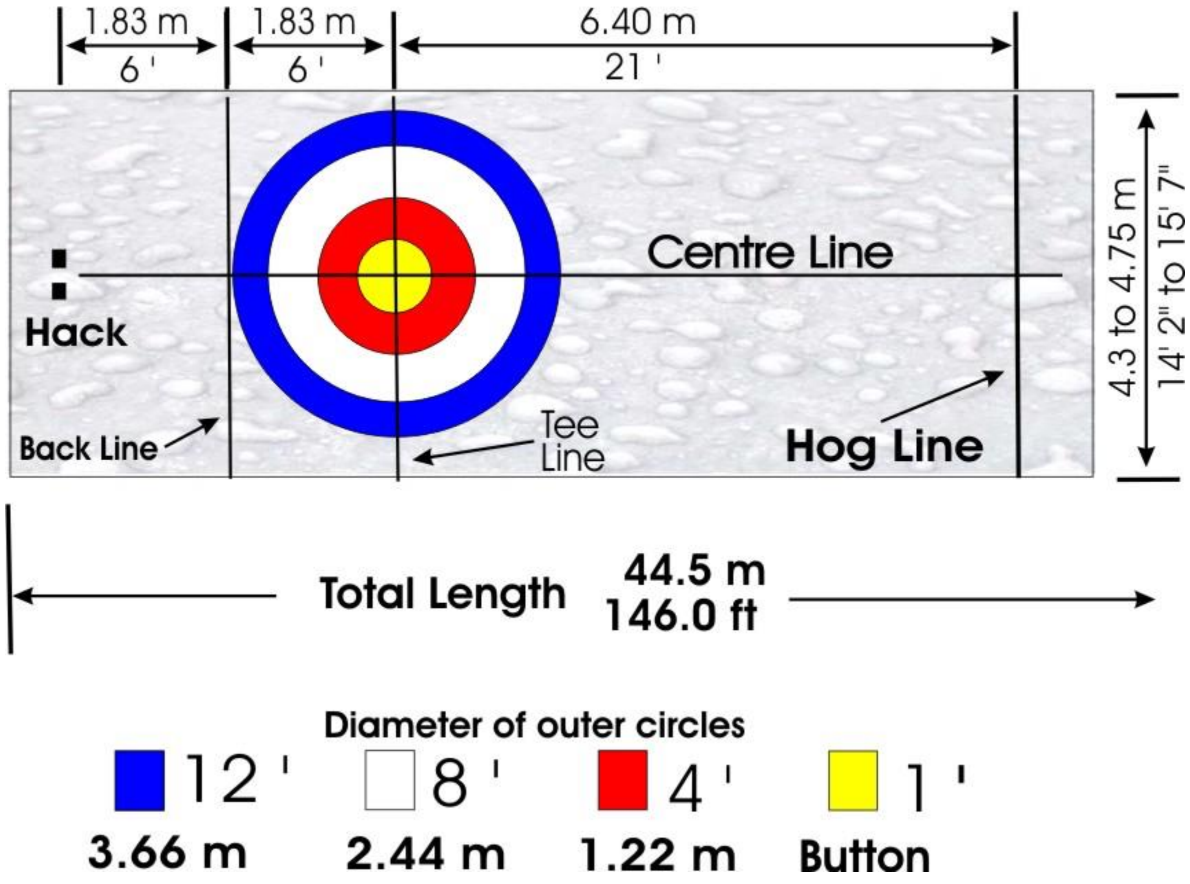
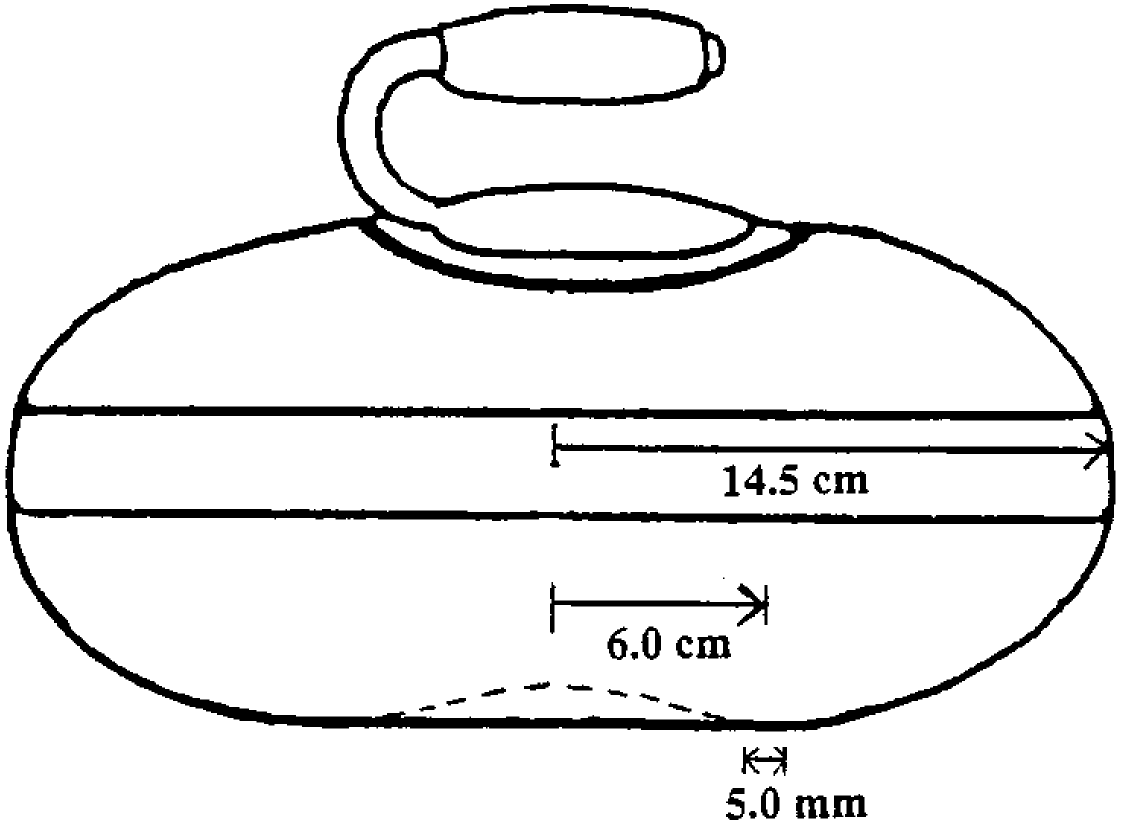
Video of Interaction

- Gold Medal Match in the 2018 PyeongChang Olympics



Curling Info:





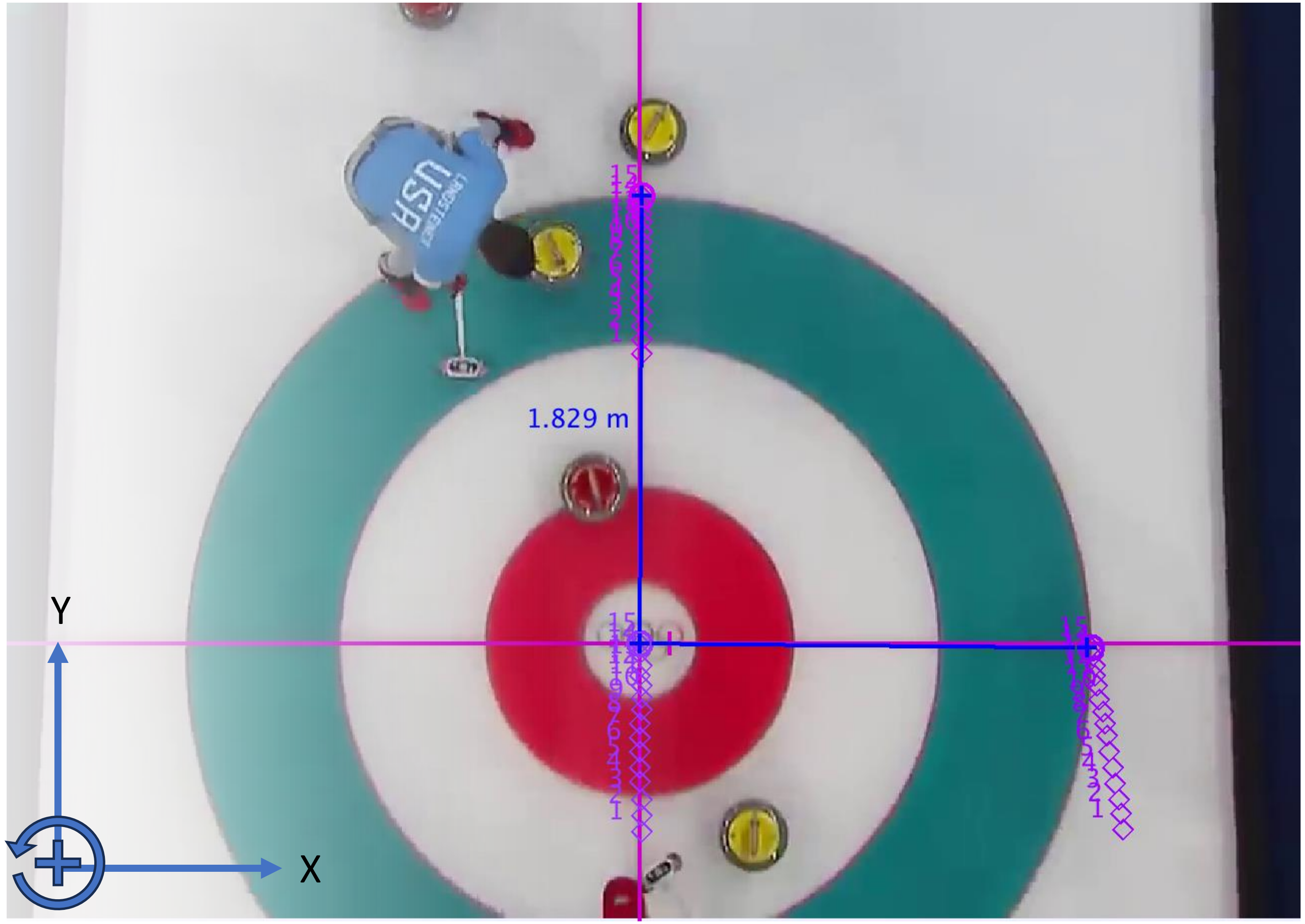
$M \sim 20\text{kg}$

Assumptions

- Geometry of Curling Stones – simple cylindrical disc
- Frictional Force on Ice – constant and very small (neglect for most!)
- Neglect of Air Drag
- Center of Mass – center of disc
- Coefficient of Restitution – material constant

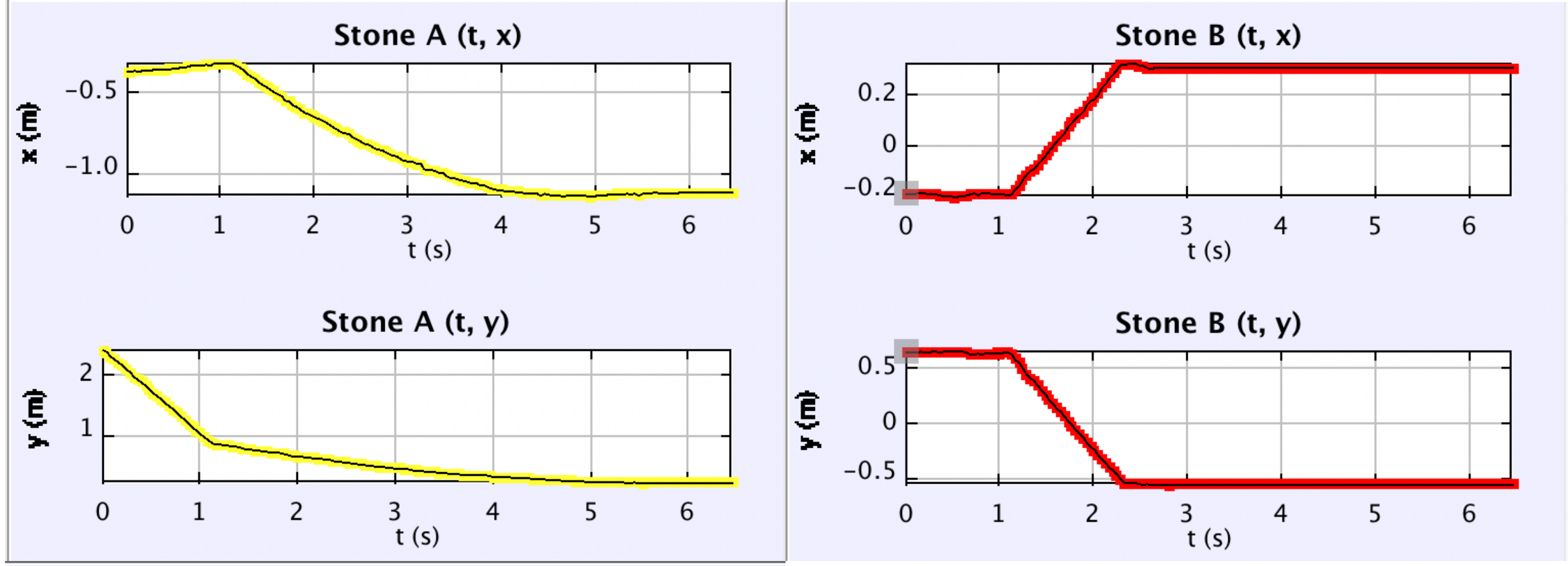
Tracker Analysis: Axis

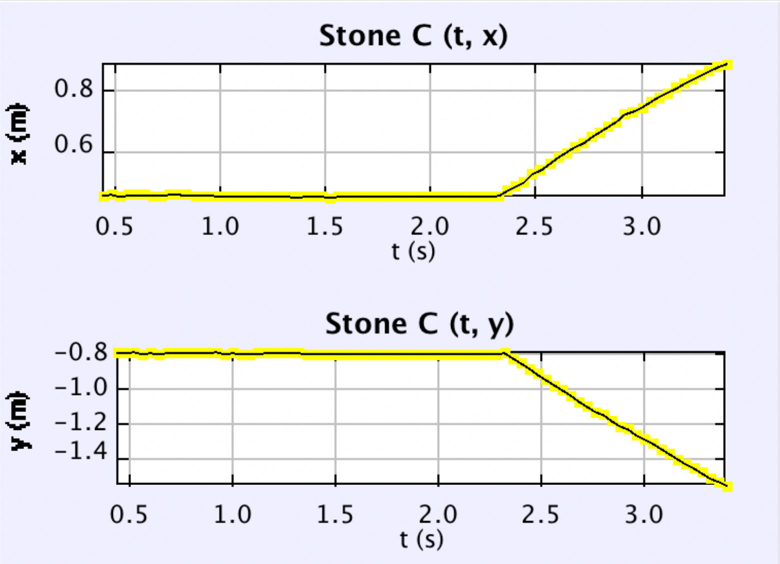
- Camera moves and zooms out
- Relative frame, changing position and scale
- Track 3 points:
  - Circle center, top edge, right edge
- Affix origin to circle center
- Affix calibration sticks to origin + circle edges



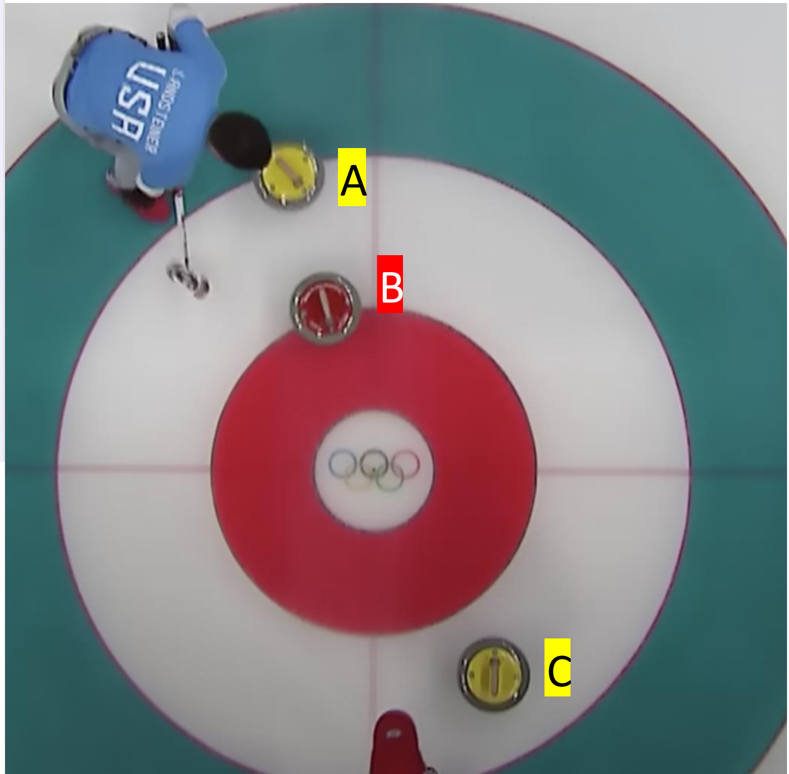
Tracker Analysis: Motion!

- Track all 3 stones at CoM
- Tracker takes care of our relative frame!
- Collision 1 (A+B) @ 1.120s
- Collision 2 (B+C) @ 2.320s

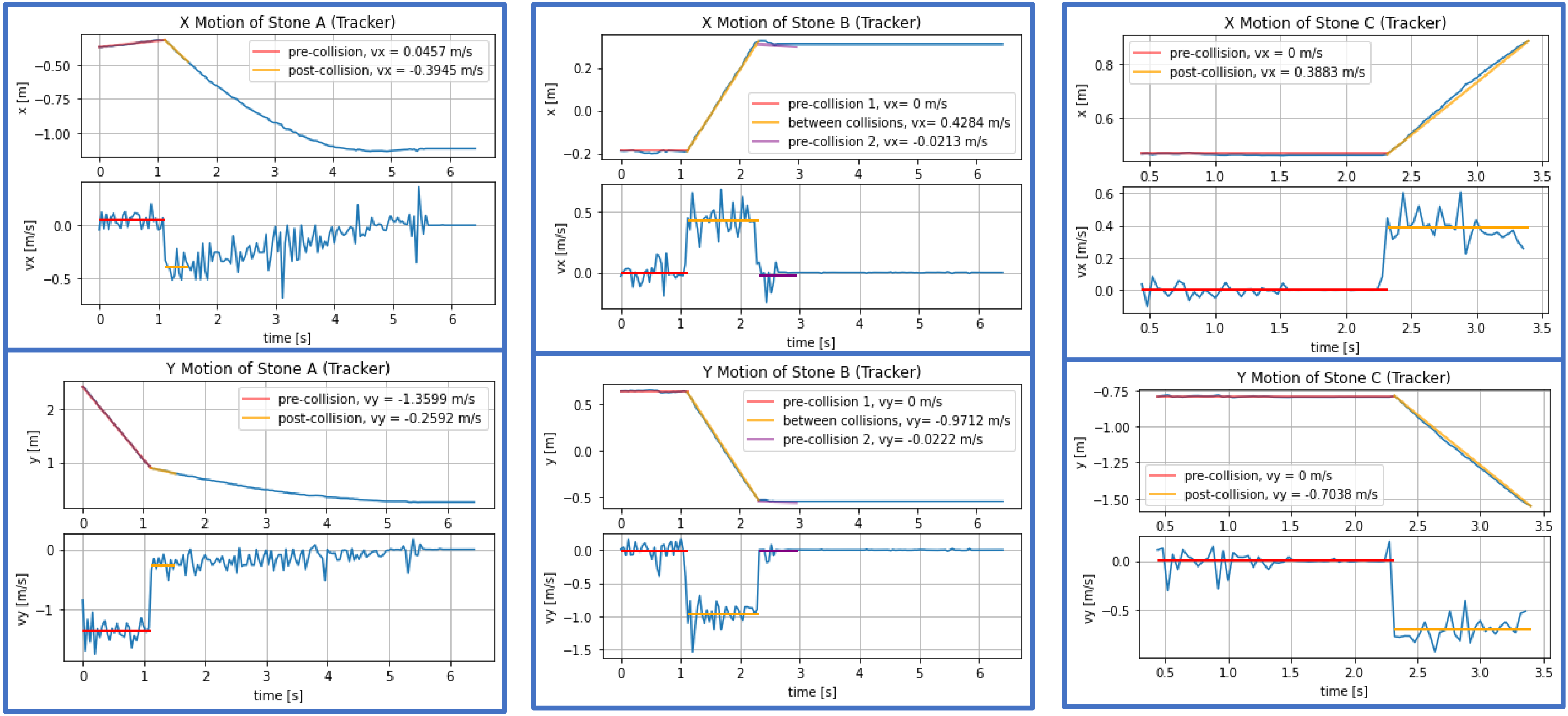




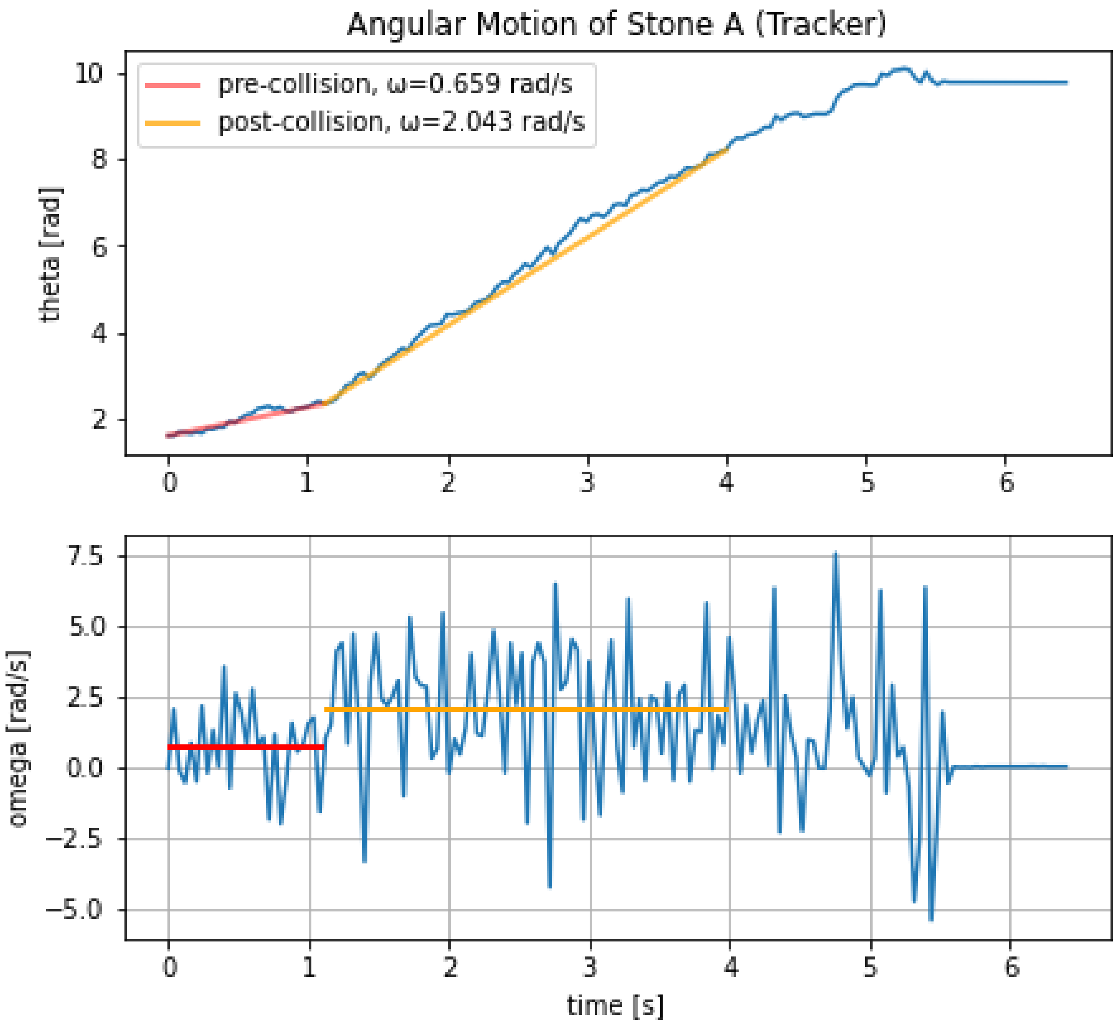
Note Stone C is out of frame until about 0.5s, and leaves frame at about 4s.



Tracker Analysis: Motion (cont.)

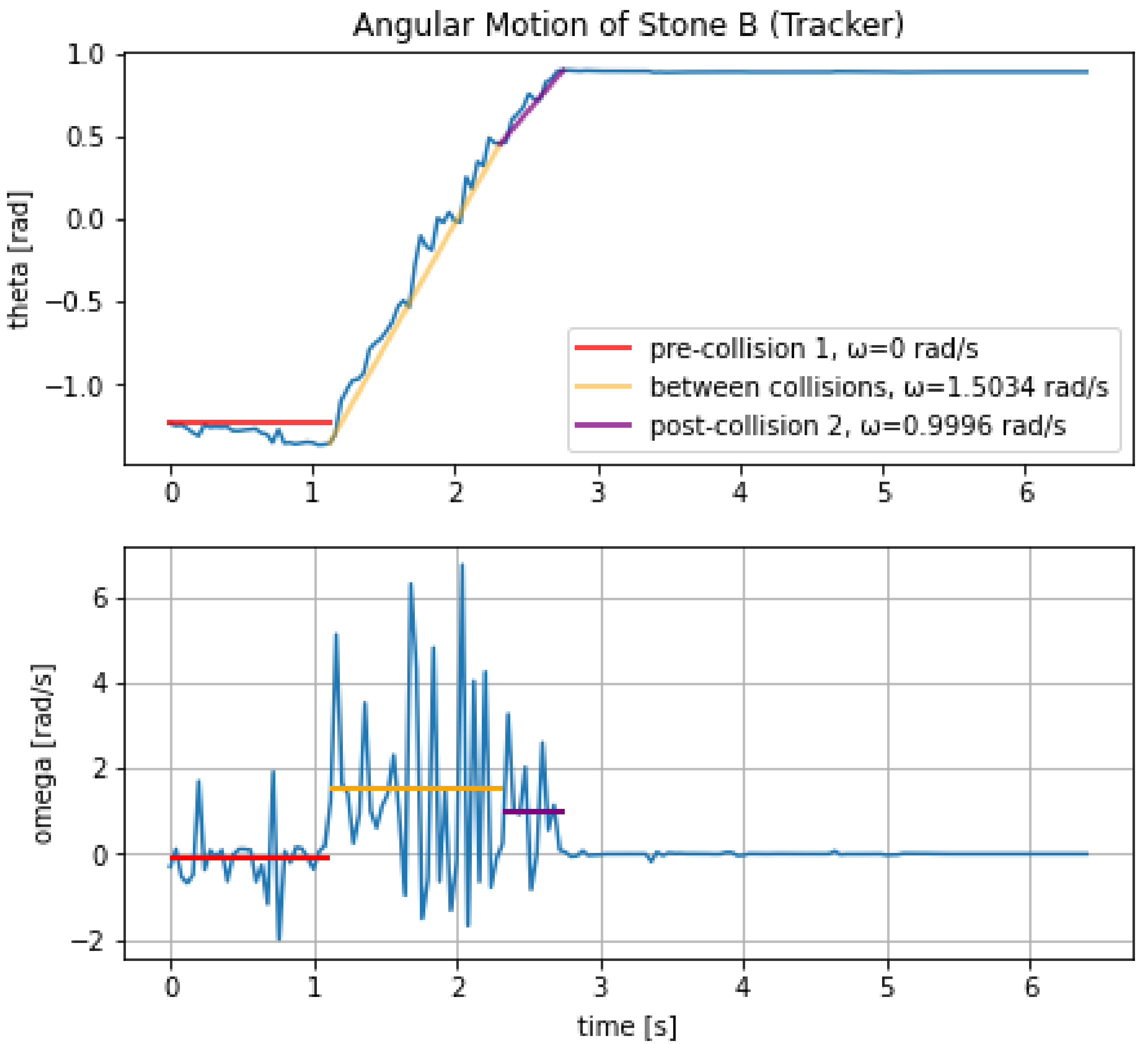


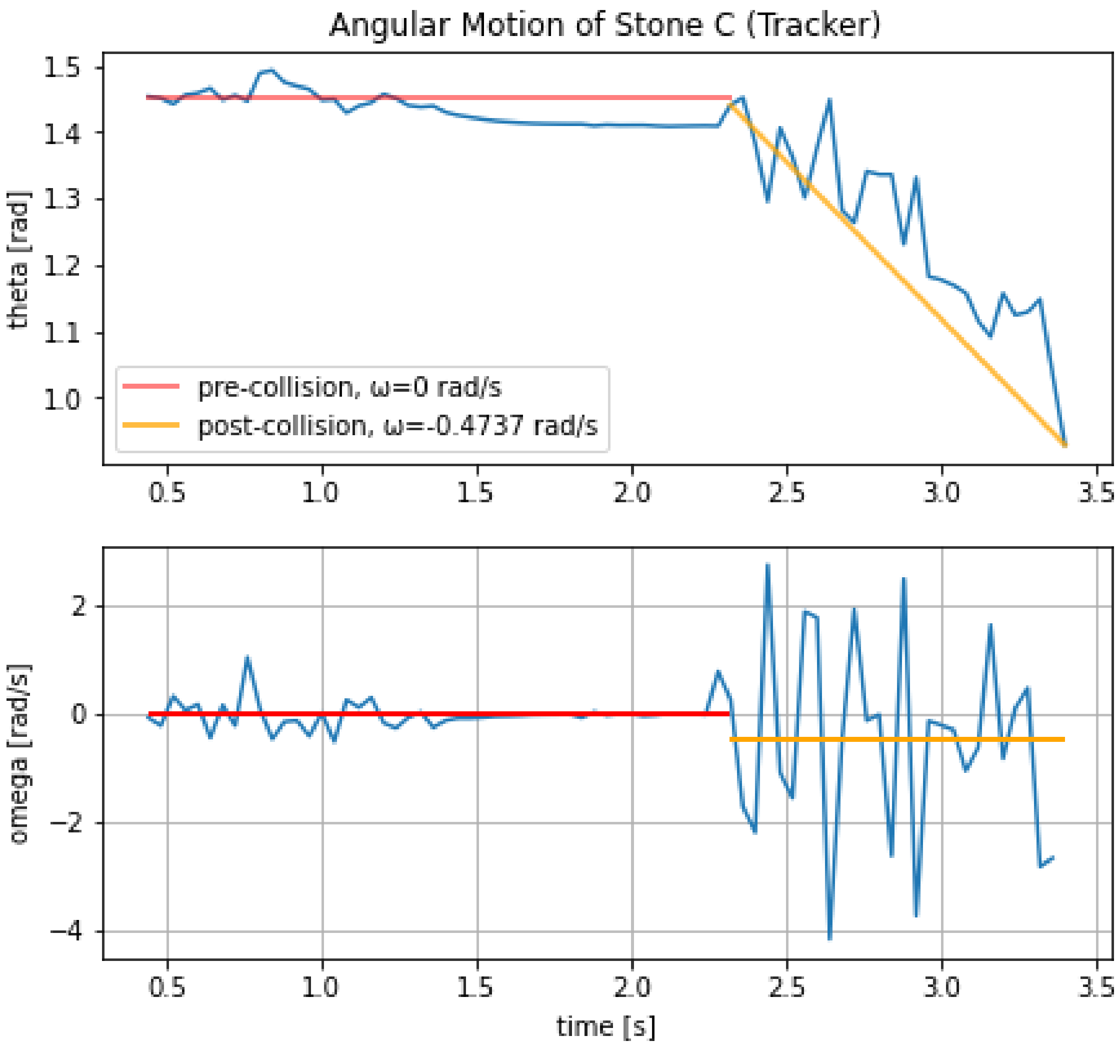
Tracker Analysis: Angular Motion!

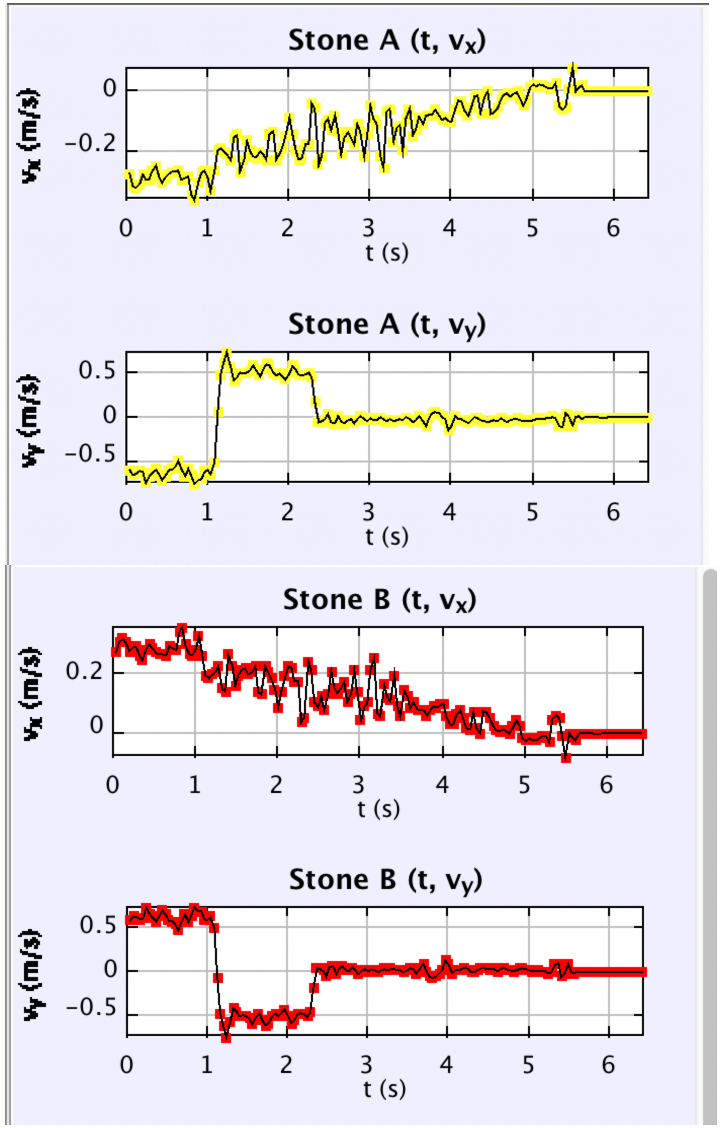


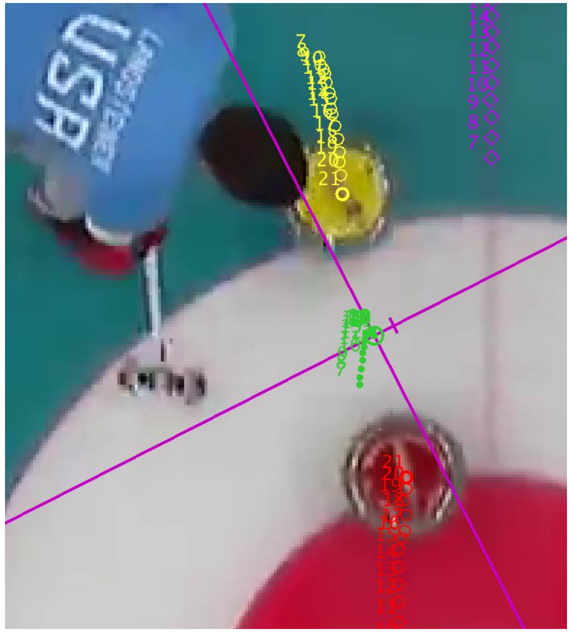
- Set new axes, origin at CoM of each stone
  - Origin moves with stone!
- Track edge of stones, plot angle wrt. origin
- Neglect friction for rotation → find average ω for **select** periods of time (before/after collision)
  - Use Python!

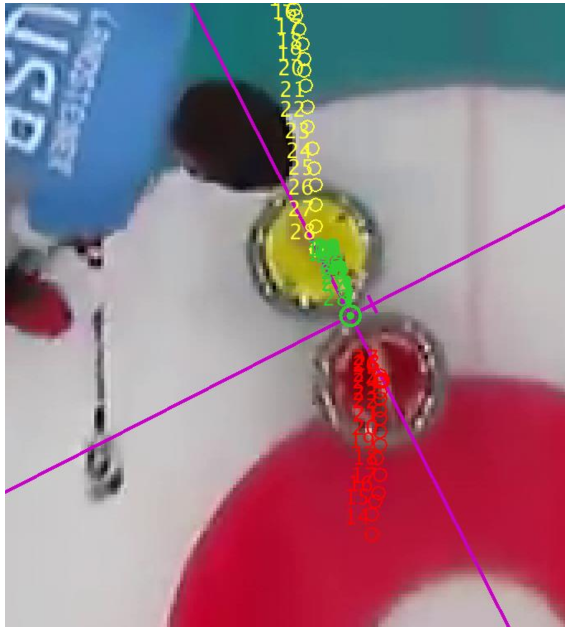
Tracker Analysis: Angular Motion (cont.)









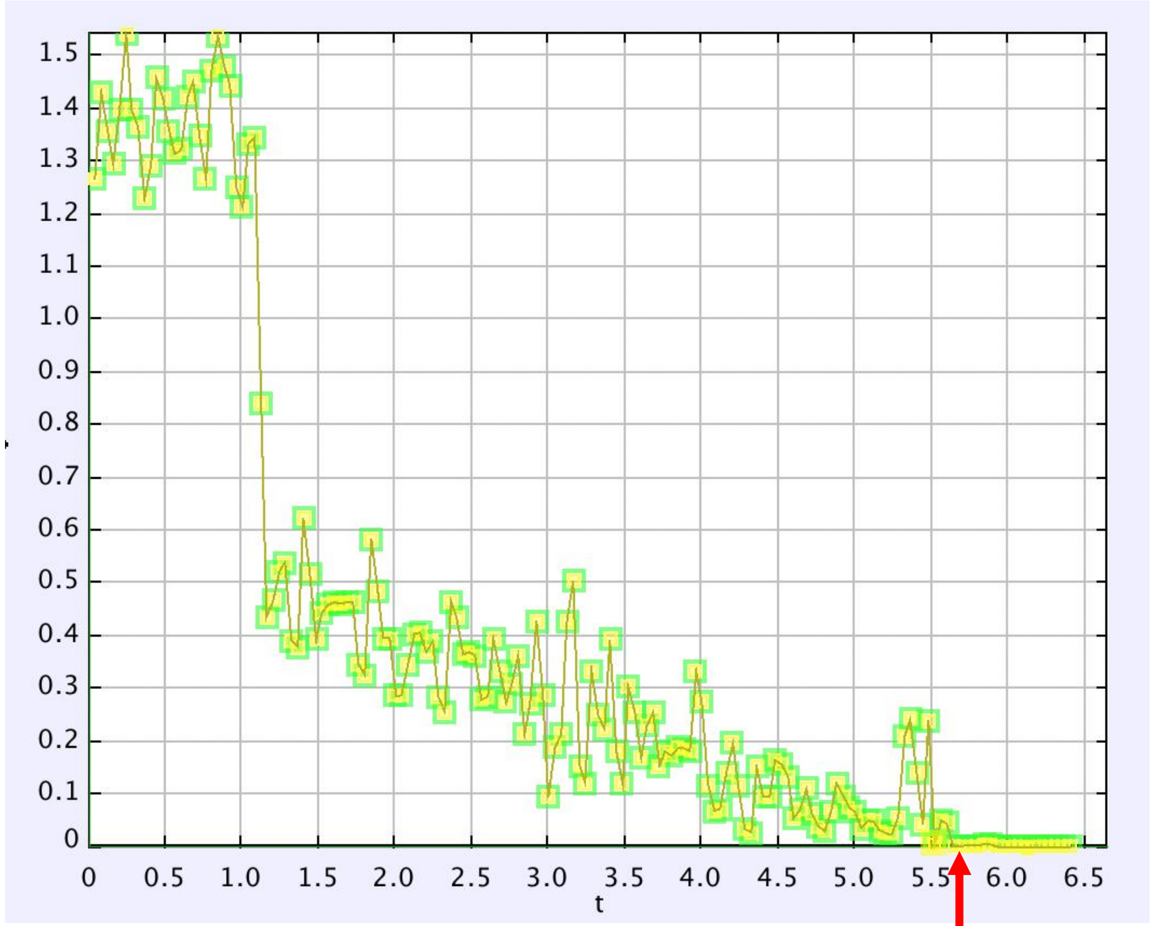


Tracker Analysis: Coefficient of Restitution

- Isolate 1st interaction (A+B)
- Set new axis
- Origin at combined COM
- Y-axis (Normal) runs through the two COM
- Neglect X-component (Tangential)
- Calculated COR: $e = 0.8345$!!!!
- Actual COR*: $e = 0.83 \pm 0.06$

```
COR = (vyB_f_obj - vyA_f_obj) / (vyA_i_obj - vyB_i_obj)
```

* https://www.sciencedirect.com/science/article/pii/S0019103510003465?via%3Dihub#s0040



Columns ▾ Stone A

t (s)	v (m/s)	frame
5.320	0.210	158
5.360	0.242	159
5.400	0.143	160
5.440	4.270E-2	161
5.480	0.241	162
5.520	4.908E-3	163
5.560	5.091E-2	164
5.600	4.519E-2	165
5.640	9.375E-4	166

Tracker Analysis: Friction

- Because Stone A moves for so long, the effects of the small friction force can be seen!
  - Isolate t=5.520s for full stop.
  - Collision at t=1.120s
  - $\Delta t = 4.40$s
- "Initial" speed (after collision):

$$v_i = \sqrt{(-0.3945)^2 + (-0.2592)^2} = 0.4720\, m/s$$

$$\boldsymbol{v_f = v_i + a\Delta t}$$

$$0 = 0.472\, m/s + a(4.40s)$$

$$a = -0.1073\, m/s^2$$

$$\boldsymbol{F_f = \mu N = ma}$$

$$\mu mg = ma$$

$$\mu = \frac{a}{g} = \frac{-0.1073\, m/s^2}{-9.81\, m/s^2}$$

- Calculated: $\mu_k = 0.0109$
- Actual*: $\mu_k \in [0.006, 0.016]$

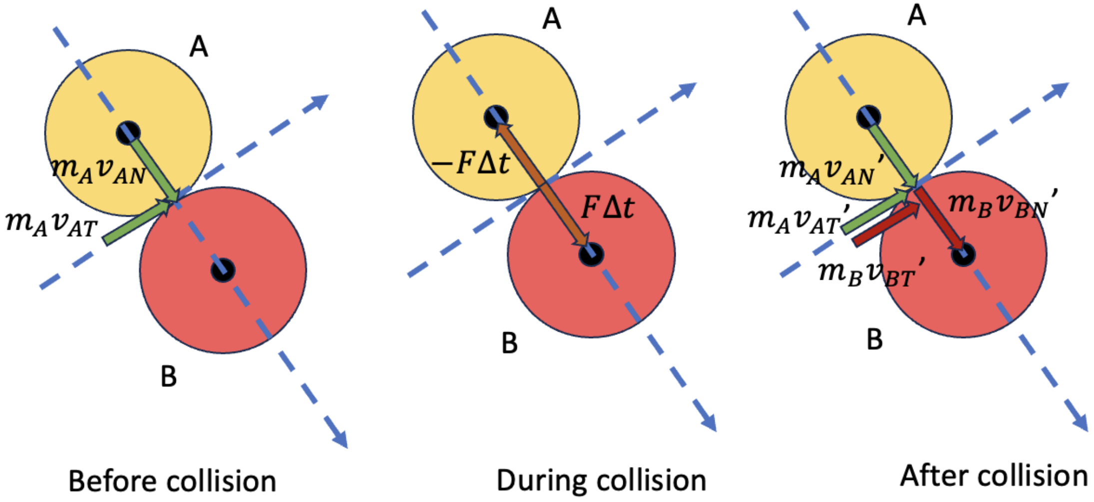
* https://www.mdpi.com/2075-4442/10/10/265#:~:text=It%20can%20be%20seen%20that,film%20on%20the%20ice%20surface.

Python Modeling: Our turn! ☺

- Scrape necessary data from Tracker
  - Initial positions of all stones
  - Initial velocity (trans. and ang.) of stone A
- Write equations of motion
- Use Python script

Python Modeling: Equations of Collision

Let Stone B start from rest (like in the video model)
Note that all velocities shown act on the **point of contact**.



Ignore impulsive force if we use momentum of system! It is internal ☺

Knowns (from Tracker): $v_{AT}, v_{AN}, v_{BT} = v_{BN} = 0$

No change in velocity along tangential (no impulse!):

$v_{AT} = v_{AT}'$ $\quad$ $v_{BT} = v_{BT}' = 0$

Conserve momentum along normal:

$$m_A v_{AN} + m_B v_{BN} = m_A v_{AN}' + m_A v_{BN}'$$

$$v_{AN} = v_{AN}' + v_{BN}'$$

Recall $m_A = m_B$
$v_{BN} = 0$

Use COR along normal:

$$v_{BN}' - v_{AN}' = e(v_{AN} - v_{BN})$$

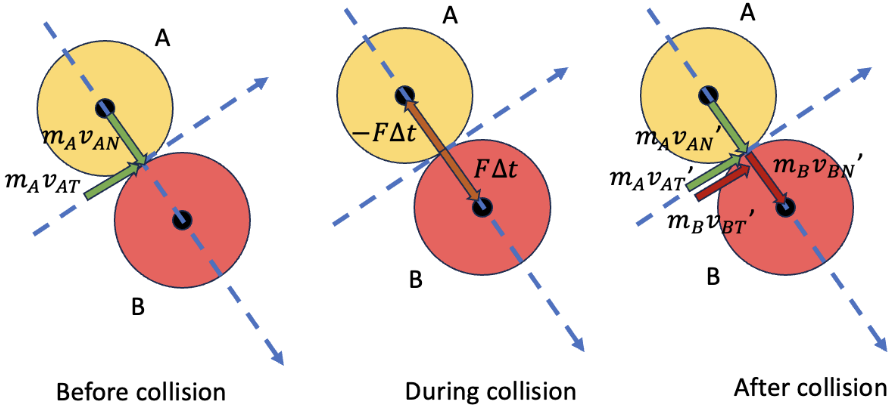
$$v_{BN}' - v_{AN}' = e(v_{AN})$$

4 equations, 4 unknowns!

Python Modeling: What about rotation!?

- All our Tracker velocities are based on COM!
- To get velocity of point of contact:

$$\overrightarrow{v_A} = \overrightarrow{v_G} + (\vec{\omega} \times \vec{r})$$
$$\overrightarrow{v_{An}} = \overrightarrow{v_{Gn}}$$
$$\overrightarrow{v_{At}} = \overrightarrow{v_{Gt}} + r\omega\hat{t}$$



- After the collision, it is necessary to work backwards:

$$\overrightarrow{v_{Gt}'} = \overrightarrow{v_{At}'} - r\omega'\hat{t}$$

- How do we get ω'?

$$I\omega_A + I\omega_B = I\omega_A' + I\omega_B' \qquad\qquad \omega_A = \omega_A' + \omega_B'$$

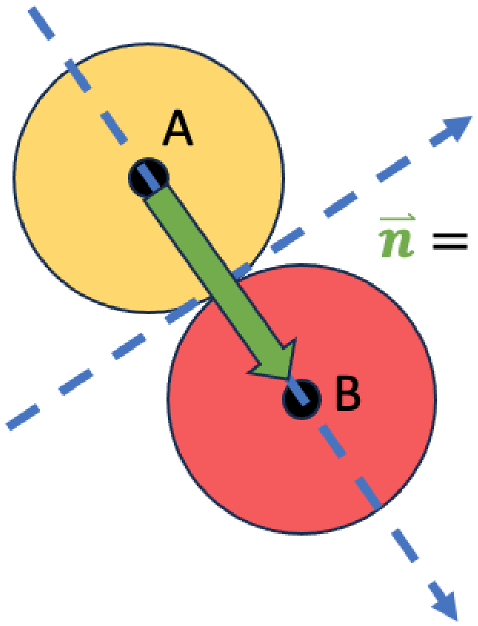
- Here we ran into an issue…
  - We can get total angular momentum of system
  - But we found no good way to split it per stone:

$$I\omega_A + M\Delta t = I\omega_A',$$
$$M = F_f * r$$

But friction depends on the Normal force (impulse), which changes per collision, and means we introduce MORE unknowns!

What we decided:
- Realized our system has too many unknowns to solve.
- Need to give ourselves more info from Tracker!
- We gave ourselves:
  - ω_A' after Col. 1
  - ω_C' after Col. 2

Python Modeling: How do we get our x-y Tracker data to n-t?



$$\vec{n} = (B_x - A_x)\hat{\imath} + (B_y - A_y)\hat{\jmath}$$

```python
# Define the known values (Insert Values)
P1x, P1y = -0.320, 0.894                # Position of COM of Stone A
P2x, P2y = -0.189, 0.629                # Position of COM of Stone B
V1x, V1y, w1 = 0.0457, -1.3599, 0.6586         # Velocities of Stone A
V2x, V2y, w2 = 0, 0, 0                  # Velocities of Stone B


# Calculate normal and tangential Components of Velocity

# Define the normal and tangential vectors
n_vector = (P2x - P1x, P2y - P1y)   #normal vector
mag_n = sqrt(n_vector[0]**2 + n_vector[1]**2)   # Calculate the magnitude of the normal vector

print("Actual distance between the two COM:", mag_n)
print("Theoretical distance between the two COM:", 2*r)
print("")

unit_n = ((n_vector[0] / mag_n) , (n_vector[1] / mag_n)) # Unit normal vector
print("unit n:", unit_n)

V1 = (V1x, V1y) # velocity of stone 1 in vector form


# Calculate normal and tangential components
mag_V1n = ((V1[0] * unit_n[0]) + (V1[1] * unit_n[1]))
V1n = ((mag_V1n * unit_n[0]) , (mag_V1n * unit_n[1])) # Dot product of unit_n and V1 is Vn1
```

- Calculate unit normal vector
  - Position vectors of CoM
    - Due to the geometry of circle
  - Magnitude of normal vector

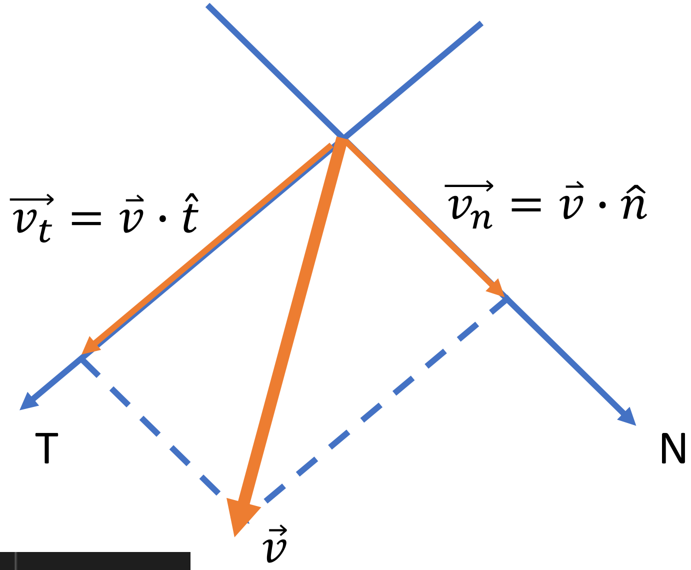
- Calculate normal component
  - Dot product
  - Projection on n-axis

Python Modeling: x-y to n-t (cont.)

- Tangential Component

$$\overrightarrow{v_1} = \overrightarrow{v_{1n}} + \overrightarrow{v_{1t}}$$
$$\overrightarrow{v_{1t}} = \overrightarrow{v_1} - \overrightarrow{v_{1n}}$$

- Unit Tangent Vector
  - Calculate from $\overrightarrow{v_{1t}}$
  - Important for later when converting n-t coordinates back to x-y




```python
V1t = ((V1[0] - V1n[0]) , (V1[1] - V1n[1] + r * w1))
mag_V1t = sqrt(V1t[0]**2 + V1t[1]**2)
unit_t = ((V1t[0] / mag_V1t) , (V1t[1] / mag_V1t))
print("unit t:", unit_t)
print("")

print("V1n:", mag_V1n)
print("V1t:", mag_V1t)
print(sqrt(V1[0]**2 + V1[1]**2), " = ", sqrt(mag_V1n**2 + mag_V1t**2) , "?")
print("")


mag_V2n = 0
mag_V2t = 0
```

Python Modeling: Solving the system of equations

Knowns (from Tracker): $v_{AT}, v_{AN}, v_{BT} = v_{BN} = 0$

No change in velocity along tangential (no impulse!):

$v_{AT} = v_{AT}'$ $\quad$ $v_{BT} = v_{BT}' = 0$

Conserve momentum along normal:

$m_A v_{AN} + m_B v_{BN} = m_A v_{AN}' + m_A v_{BN}'$

$v_{AN} = v_{AN}' + v_{BN}'$

Recall $m_A = m_B$

$v_{BN} = 0$

Use COR along normal:

$v_{BN}' - v_{AN}' = e(v_{AN} - v_{BN})$

$v_{BN}' - v_{AN}' = e(v_{AN})$

4 equations, 4 unknowns!

- 4 equations, 4 unknowns
- Velocity calculated = velocity on point of collision
    - Need to consider angular velocity to determine CoM

```
# Da equations
eq1 = Eq(mag_V1t, mag_V1tf)
eq2 = Eq(mag_V2t, mag_V2tf)
eq3 = Eq(m * mag_V1n + m * mag_V2n , m * mag_V1nf + m * mag_V2nf)
eq4 = Eq(mag_V2nf - mag_V1nf, e * (mag_V1n - mag_V2n))

# Solve the system of equations
solution = solve((eq1, eq2, eq3, eq4), (mag_V1tf, mag_V2tf, mag_V1nf, mag_V2nf))

# Print the solution
print("Tangential and Normal Components:")
print("VAtf =", solution[mag_V1tf])
print("VBtf =", solution[mag_V2tf])
print("VAnf =", solution[mag_V1nf])
print("VBnf =", solution[mag_V2nf])
print("")

# Calculate final velocities using the normal and tangential components
mag_V1tf = solution[mag_V1tf]
mag_V2tf = solution[mag_V2tf]
mag_V1nf = solution[mag_V1nf]
mag_V2nf = solution[mag_V2nf]
```

Python Modeling: Angular Velocity and x-y coordinate system

```python
# Calculate final angular velocities using conservation of angular momentum
# Define the final angular velocities
wtotal = 0.6586
w1f = 2.0433     # Angular velocity of A, given
w2f = wtotal - w1f   # Angular velocity of B

# Display the final angular velocities
print("Total System Angular Velocity:", wtotal)
print("w1f:", w1f)
print("w2f:", w2f)
print("")

# Define the final velocities in terms of initial velocities and final components
V1f = (mag_V1nf * unit_n[0] + (mag_V1tf - (r * w1f)) * unit_t[0] , mag_V1nf * unit_n[1] + (mag_V1tf - (r * w1f)) * unit_t[1])
V2f = (mag_V2nf * unit_n[0] + (mag_V2tf - (r * w2f)) * unit_t[0] , mag_V2nf * unit_n[1] + (mag_V2tf - (r * w2f)) * unit_t[1])

# Display the final velocities
print("Final Velocities:")
print("VAxf =", V1f[0])
print("VAyf =", V1f[1])
print("VBxf =", V2f[0])
print("VByf =", V2f[1])
```

- Determine angular velocity
  - Conservation of angular momentum
  - Aforementioned given ω
- Recall:

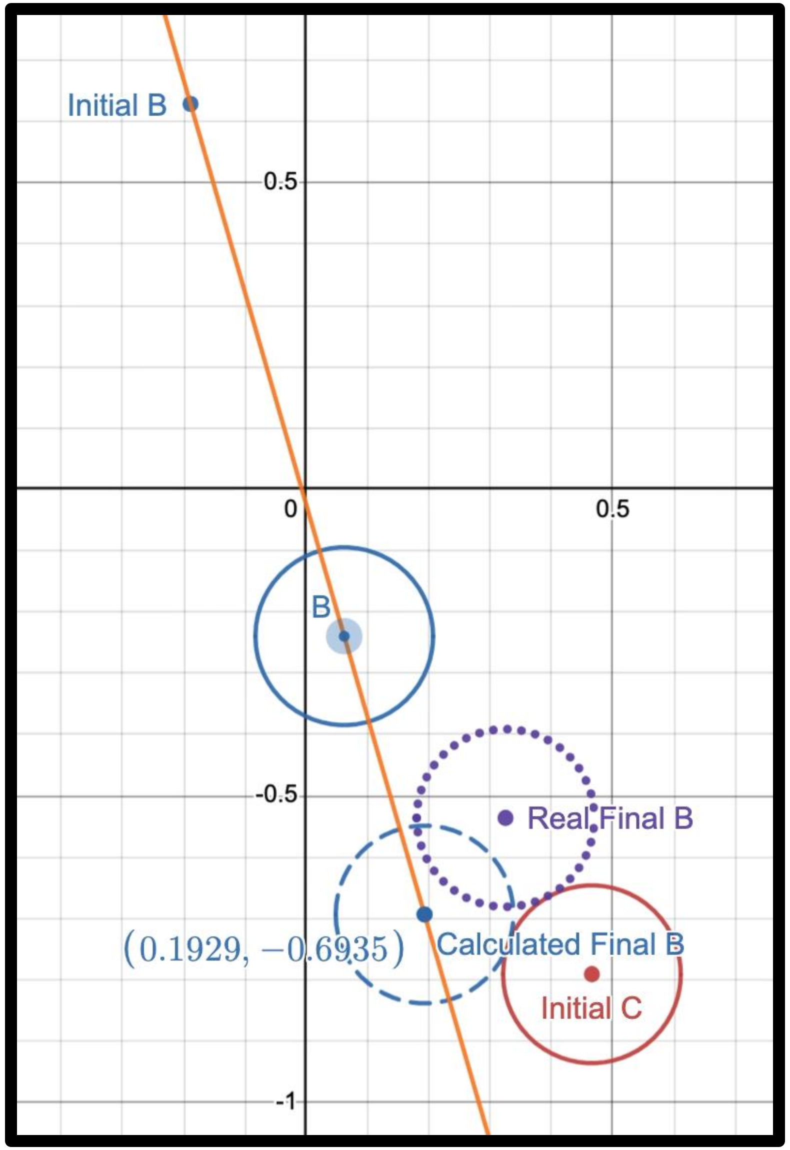
- After the collision, it is necessary to work backwards:

$$\overrightarrow{v_{Gt}}' = \overrightarrow{v_{At}}' - r\omega'\hat{t}$$

- Turn back to x-y coordinates
  - Not difficult; n-t components were written in terms of x-y
  - As most calculations were done with magnitude of n-t components
  - Multiply magnitude of n-t components by unit vectors
  - Add everything together

Python Modeling: B travelling to C

- Utilized final velocity of B
  - Parametrized to determine trajectory:
    - $x = -0.189 + t$
    - $y = 0.629 - 3.46284t$
  - Purple is real, tracker final B pos.
- Go through same process/run same python code for BC interaction



Results:

Table of Tracker Values

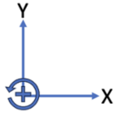
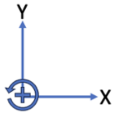
	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s	$-0.333\hat{\imath} + 0.880\hat{\jmath}$	$-0.3945\hat{\imath} - 0.2592\hat{\jmath}$	$2.0433\hat{k}$	$-0.171\hat{\imath} + 0.585\hat{\jmath}$	$0.4284\hat{\imath} - 0.9712\hat{\jmath}$	$1.5034\hat{k}$			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.325\hat{\imath} - 0.536\hat{\jmath}$	$0.4284\hat{\imath} - 0.9712\hat{\jmath}$	$1.5034\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.328\hat{\imath} - 0.536\hat{\jmath}$	$-0.0213\hat{\imath} - 0.0222\hat{\jmath}$	$0.9996\hat{k}$	$0.484\hat{\imath} - 0.822\hat{\jmath}$	$0.3883\hat{\imath} - 0.7038\hat{\jmath}$	$-0.4737\hat{k}$

Table of Python-Calculated Values

	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s		$-0.1734\hat{\imath} - 0.1615\hat{\jmath}$	$2.0433\hat{k}$		$0.3105\hat{\imath} - 1.075\hat{\jmath}$	$-1.3847\hat{k}$			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.1929\hat{\imath} - 0.6935\hat{\jmath}$	$0.3105\hat{\imath} - 1.075\hat{\jmath}$	$-1.3847\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.1929\hat{\imath} - 0.6935\hat{\jmath}$	$-0.2074\hat{\imath} + 0.9170\hat{\jmath}$	$1.132\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	$0.5444\hat{\imath} - 0.2671$	$-0.4737\hat{k}$

Limitations:

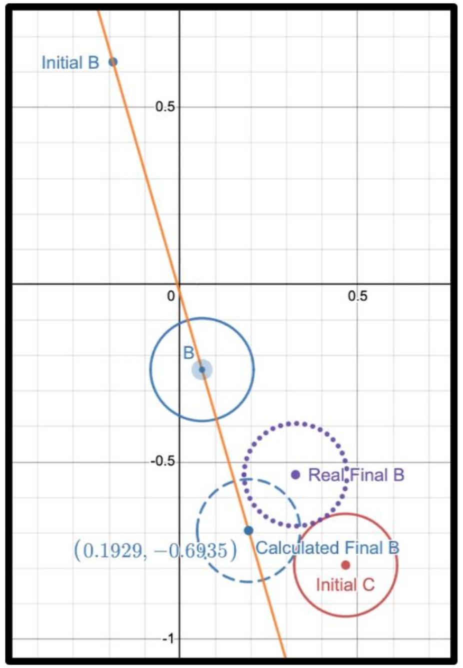
- Tracker values Do NOT obey conservation of angular momentum
  - Torque from uneven ice friction?
- Our linear values are okay to meh. Not so much for angular.

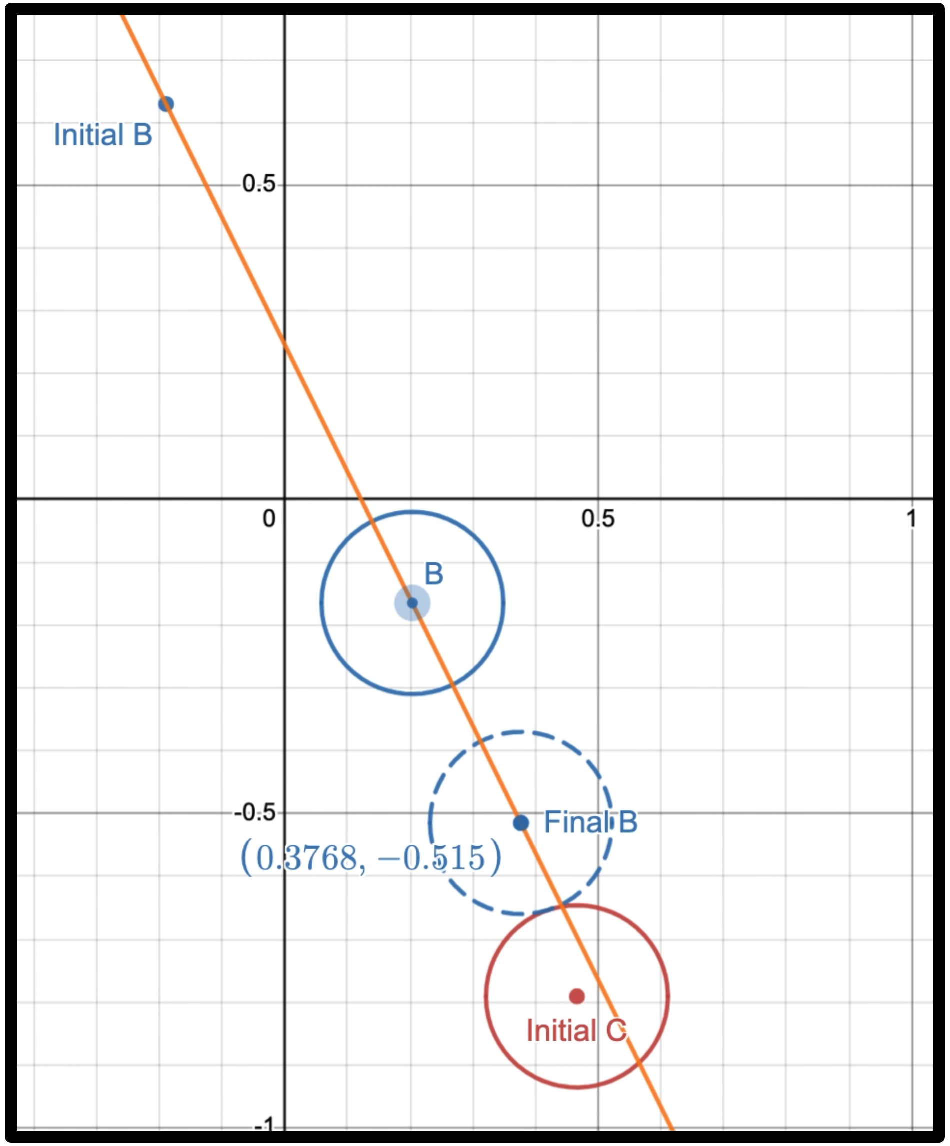
- **After collision 1**
  - Directions consistent

- **After collision 2**
  - Difference in position
  - Sign changes in expected velocity AND angular velocity!

Python Modeling: B travelling to C, no angular momentum

- Utilized final velocity of B from new python that neglects angular motion
  - Parametrized to determine trajectory
    - $x = -0.189 + t$
    - $y = 0.629 - 2.022t$
  - Purple is real final B
- Previous one $\rightarrow$





Results (cont.):

- Decided to try modeling as point mass, since the angular motion was so unpredictable
  - Know this will cause error, but curious!

Table of Tracker Values

	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s	$-0.333\hat{\imath} + 0.880\hat{\jmath}$	$-0.3945\hat{\imath} - 0.2592\hat{\jmath}$	$2.0433\hat{k}$	$-0.171\hat{\imath} + 0.585\hat{\jmath}$	$0.4284\hat{\imath} - 0.9712\hat{\jmath}$	$1.5034\hat{k}$			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.325\hat{\imath} - 0.536\hat{\jmath}$	$0.4284\hat{\imath} - 0.9712\hat{\jmath}$	$1.5034\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.328\hat{\imath} - 0.536\hat{\jmath}$	$-0.0213\hat{\imath} - 0.0222\hat{\jmath}$	$0.9996\hat{k}$	$0.484\hat{\imath} - 0.822\hat{\jmath}$	$0.3883\hat{\imath} - 0.7038\hat{\jmath}$	$-0.4737\hat{k}$

Table of Python-Calculated Values Assuming Point Masses

	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s		$-0.4568\hat{\imath} - 0.2478\hat{\jmath}$	$2.0433\hat{k}$		$0.5025\hat{\imath} - 1.0166\hat{\jmath}$	???			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.3768\hat{\imath} - 0.5150\hat{\jmath}$	$0.5025\hat{\imath} - 1.0166\hat{\jmath}$	??$\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.3768\hat{\imath} - 0.5150\hat{\jmath}$	$-0.04006\hat{\imath} + 0.09320\hat{\jmath}$	??	$0.466\hat{\imath} - 0.791\hat{\jmath}$	$0.4681\hat{\imath} - 0.8466\hat{\jmath}$	$-0.4737\hat{k}$

Collision 1 values are significantly better when modelling as stones as point masses
By Collision 2, error has propagated, but still a lot better; errors compounded from r and v.
Implies that the error from unexpected torque is greater than the error from ignoring instantaneous relative velocity

Results (cont.):

	Experimental	Actual
Coefficient of Restitution	0.8345	0.83 ± 0.06
Coefficient of Kinetic Friction	0.0109	[0.006, 0.016]

Table of Python-Calculated Values

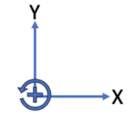
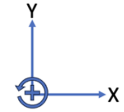
	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s		$-0.1734\hat{\imath} - 0.1615\hat{\jmath}$	$2.0433\hat{k}$		$0.3105\hat{\imath} - 1.075\hat{\jmath}$	$-1.3847\hat{k}$			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.1929\hat{\imath} - 0.6935\hat{\jmath}$	$0.3105\hat{\imath} - 1.075\hat{\jmath}$	$-1.3847\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.1929\hat{\imath} - 0.6935\hat{\jmath}$	$-0.2074\hat{\imath} + 0.9170\hat{\jmath}$	$1.132\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	$0.5444\hat{\imath} - 0.2671$	$-0.4737\hat{k}$

Table of Python-Calculated Values Assuming Point Masses

	STONE A			STONE B			STONE C		
	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)	$\vec{r}$ (m)	$\vec{v}$ (m/s)	$\vec{\omega}$ (rad/s)
Collision 1 (A+B) Frame 53 t = 1.120s	$-0.320\hat{\imath} + 0.894\hat{\jmath}$	$0.0457\hat{\imath} - 1.3599\hat{\jmath}$	$0.6586\hat{k}$	$-0.189\hat{\imath} + 0.629\hat{\jmath}$	0	0			
Post-Collision 1 (A+B) Frame 54 t = 1.160s		$-0.4568\hat{\imath} - 0.2478\hat{\jmath}$	$2.0433\hat{k}$		$0.5025\hat{\imath} - 1.0166\hat{\jmath}$	???			
Collision 2 (B+C) Frame 83 t = 2.320s				$0.3768\hat{\imath} - 0.5150\hat{\jmath}$	$0.5025\hat{\imath} - 1.0166\hat{\jmath}$	$??\hat{k}$	$0.466\hat{\imath} - 0.791\hat{\jmath}$	0	0
After Collision 2 (B+C) Frame 84 t = 2.360s				$0.3768\hat{\imath} - 0.5150\hat{\jmath}$	$-0.04006\hat{\imath} + 0.09320\hat{\jmath}$	??	$0.466\hat{\imath} - 0.791\hat{\jmath}$	$0.4681\hat{\imath} - 0.8466\hat{\jmath}$	$-0.4737\hat{k}$