EECS 367 Lab
KinEval IK Control Flow and Parameters

Administrative

Assignment 5 released

Due Wednesday, November 11 at 11:59pm

Pull stencil update from upstream!

Lab Takeaways

1. Assignment 5 goals
2. KinEval overview
3. KinEval walkthrough

→ How to start Assignment 5

Inverse Kinematics Overview

		Assignment 5: Inverse Kinematics
6	All	Manipulator Jacobian
3	All	Gradient descent with Jacobian transpose*
3	All	Jacobian pseudoinverse
6	Grad	Euler angle conversion

Features assigned to all sections

Feature assigned to grad section only

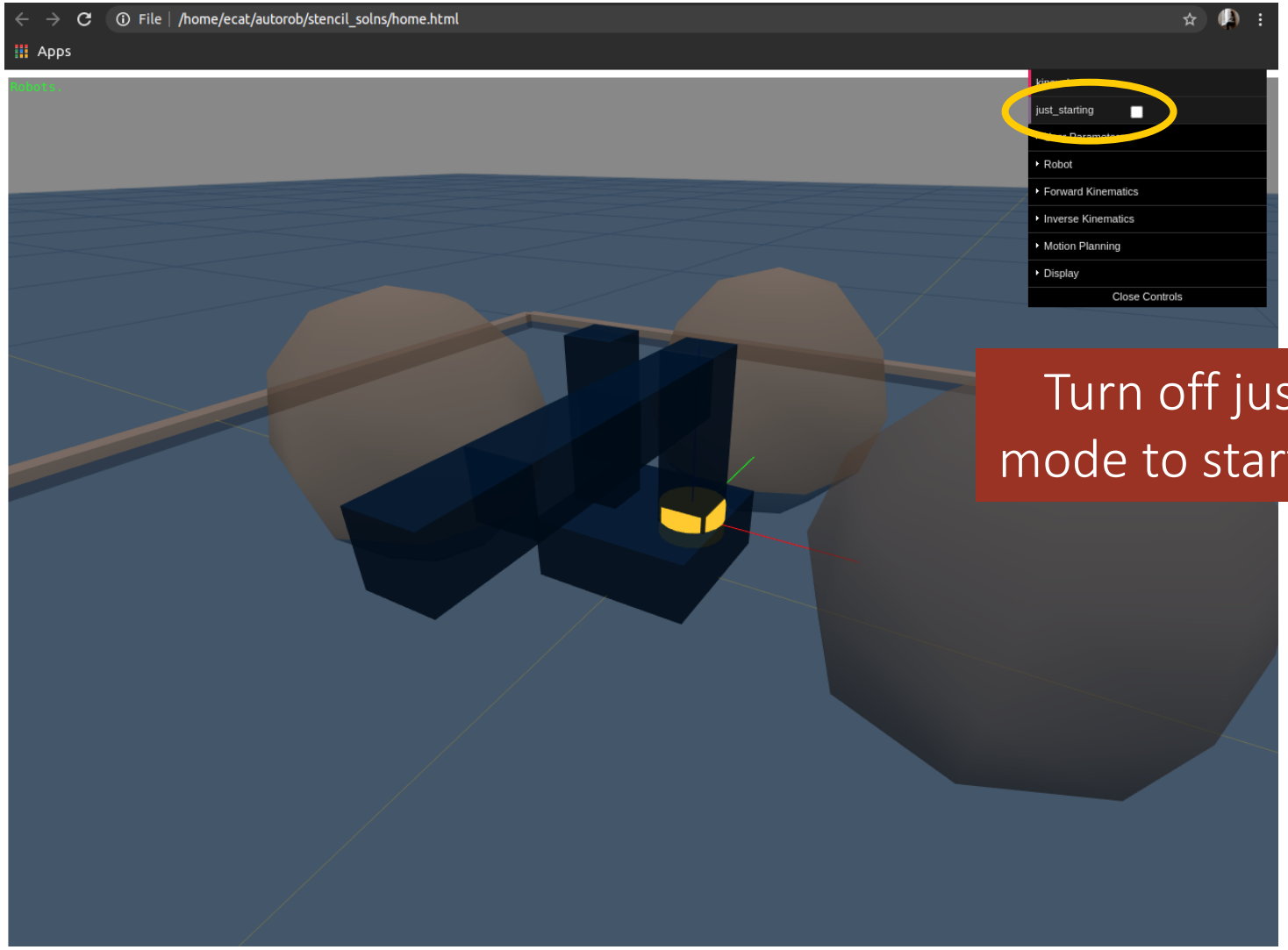
* Undergrad section will implement gradient descent for **position only**, but grad section will implement gradient descent for **position and orientation**

Demo

Running IK

Start by loading home.html

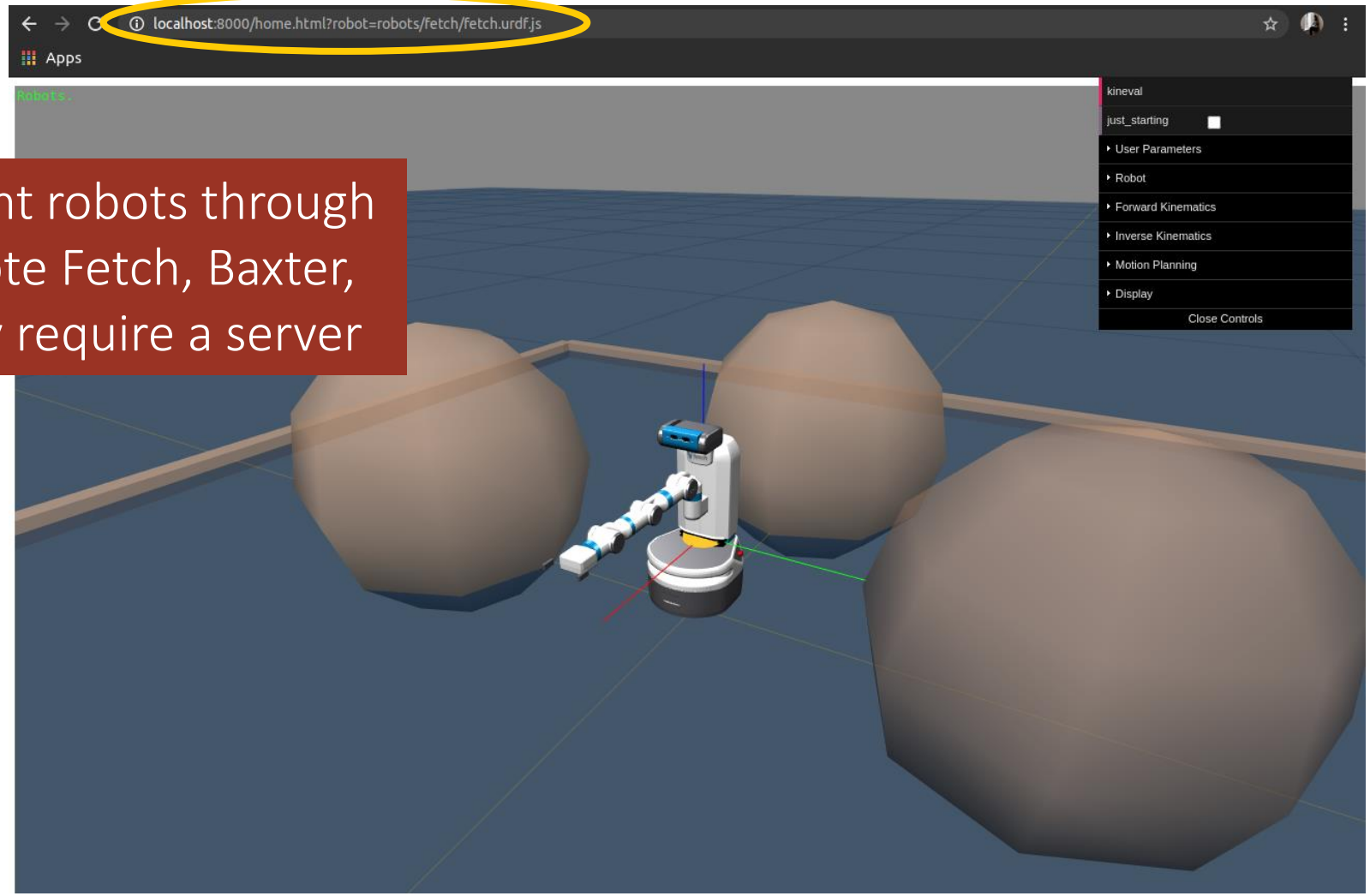
Running IK



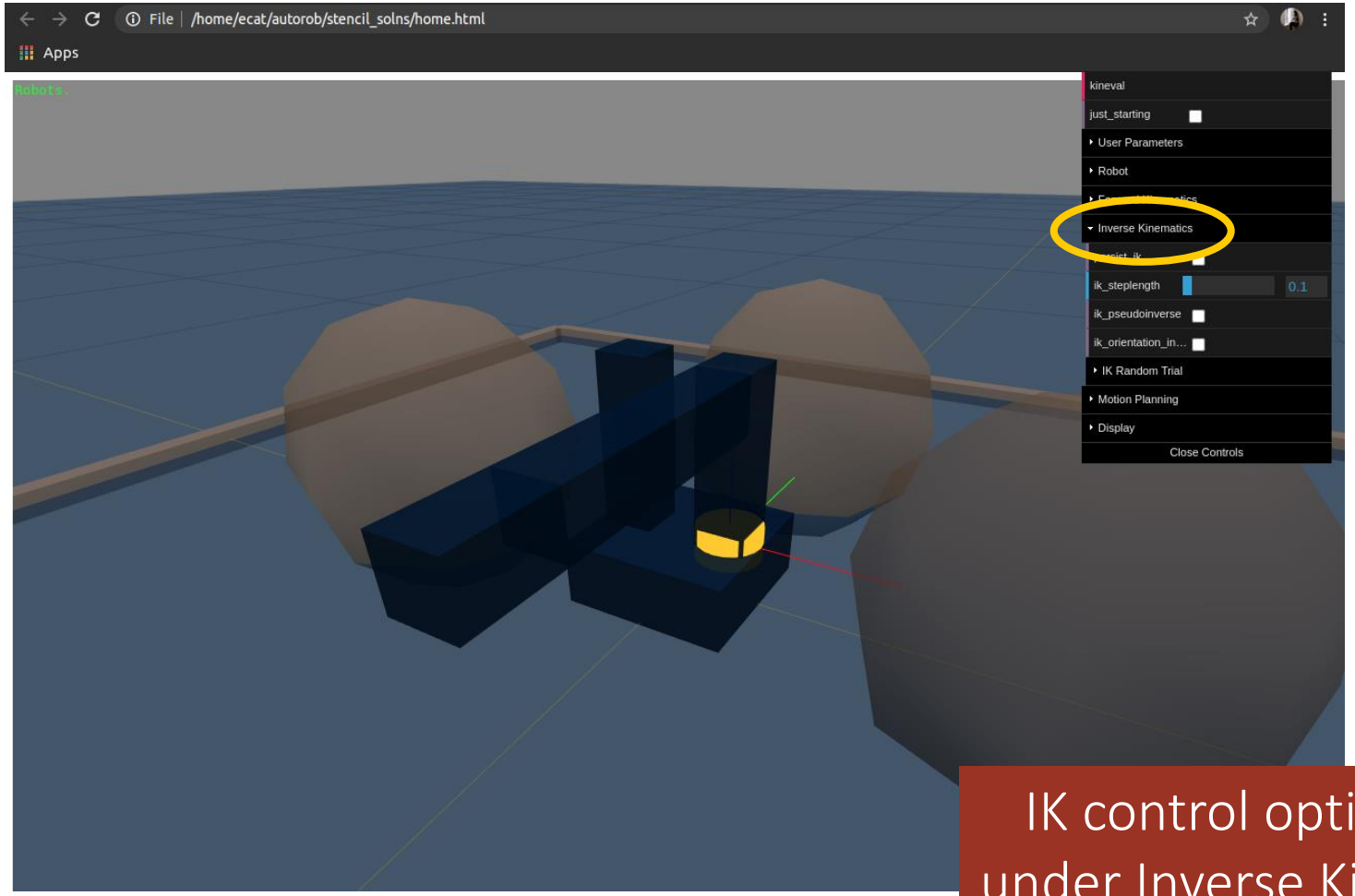
Turn off just_starting mode to start running FK

Running IK

Load different robots through the URL; note Fetch, Baxter, Sawyer may require a server

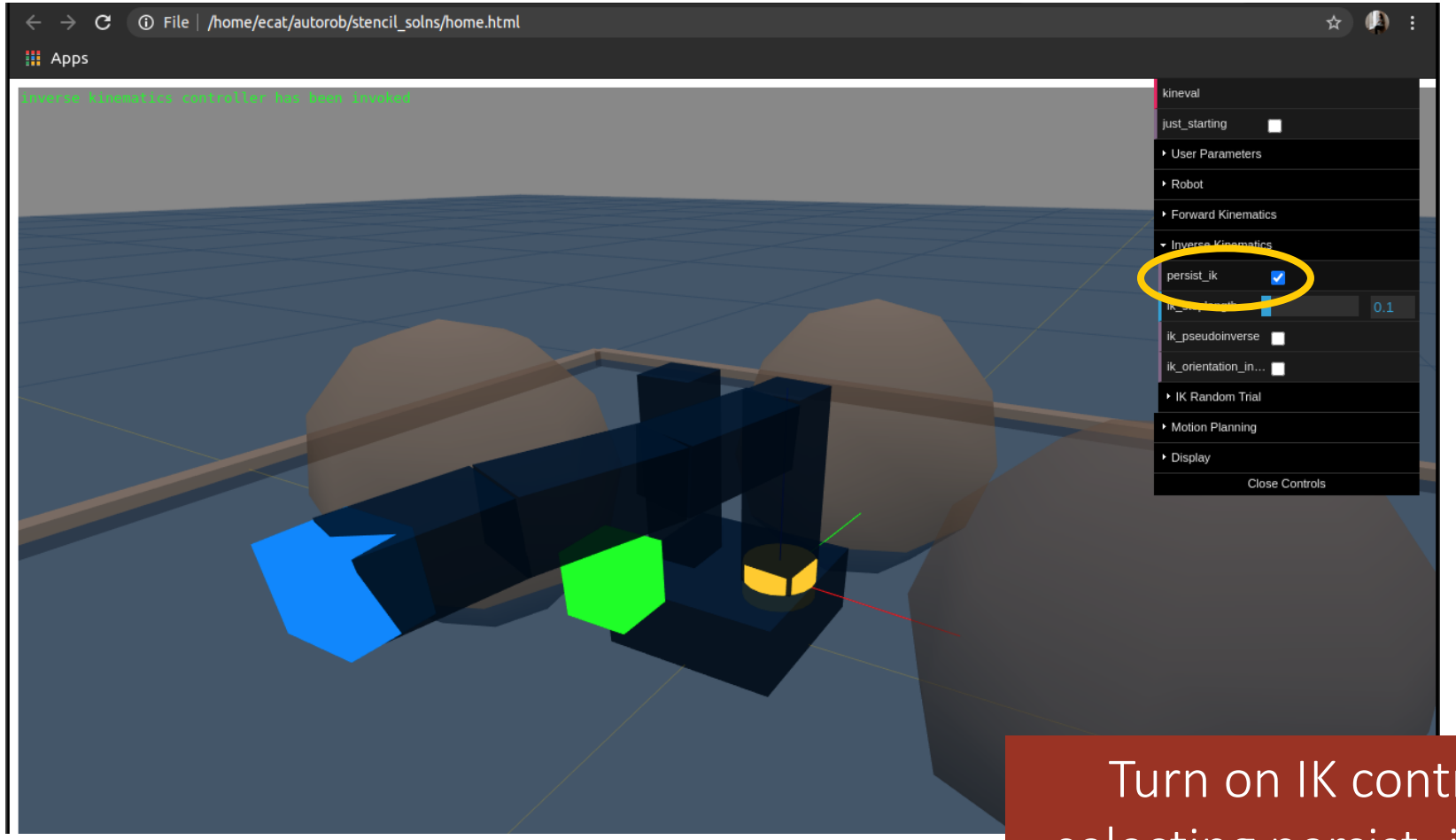


Running IK



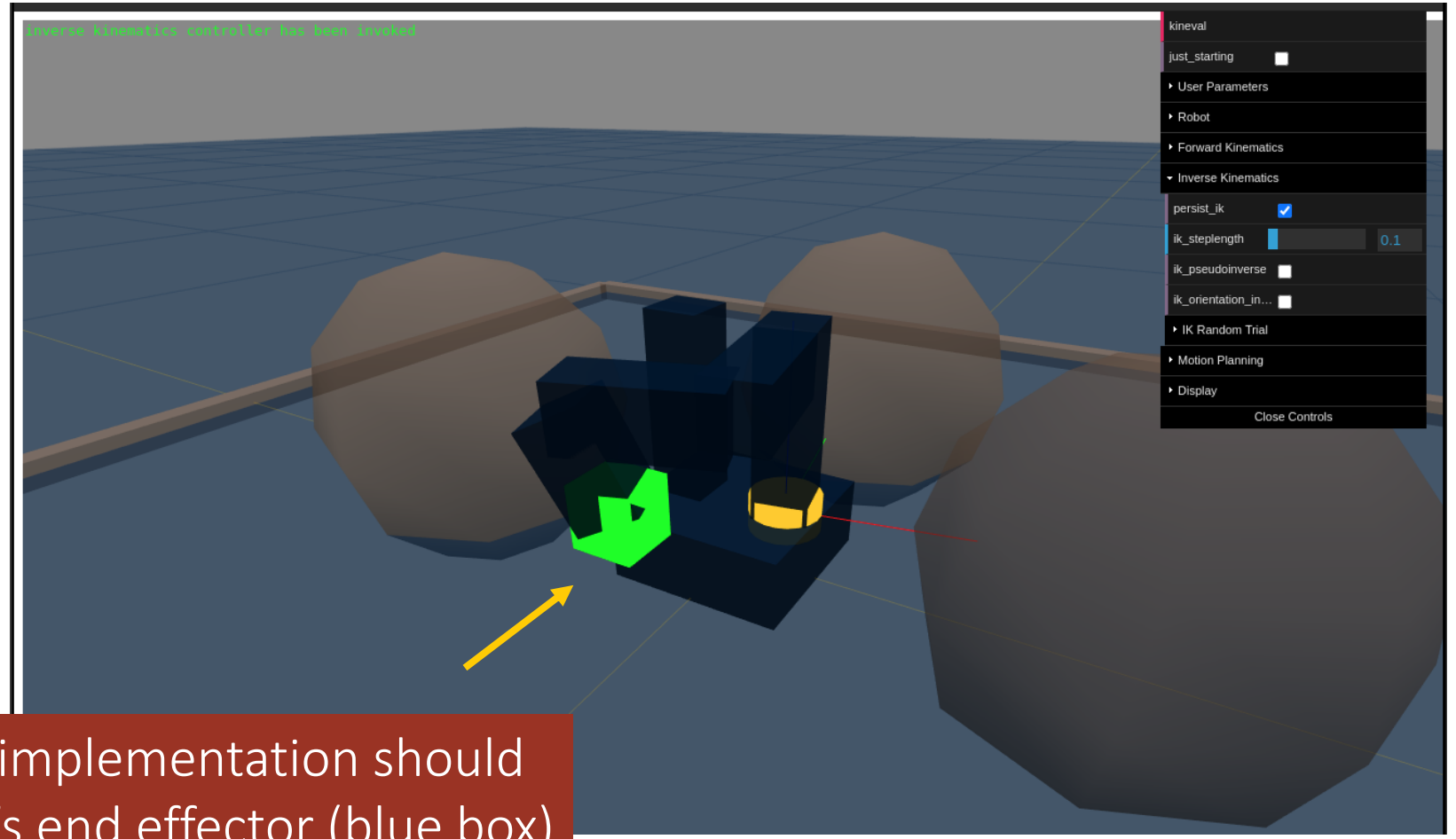
IK control options are under Inverse Kinematics in user interface

Running IK



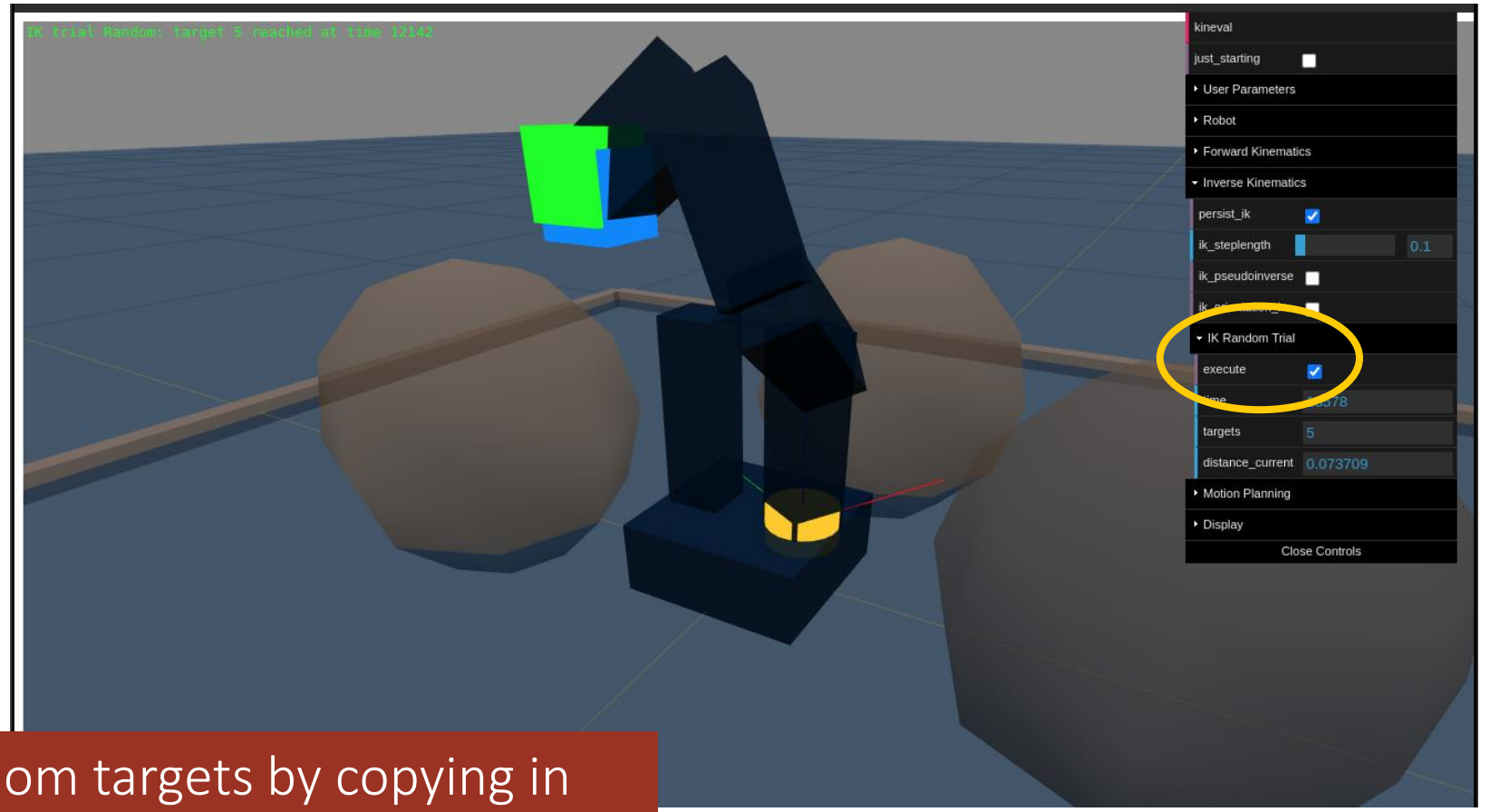
Turn on IK control by selecting persist_ik OR by holding down ‘p’ key

Running IK



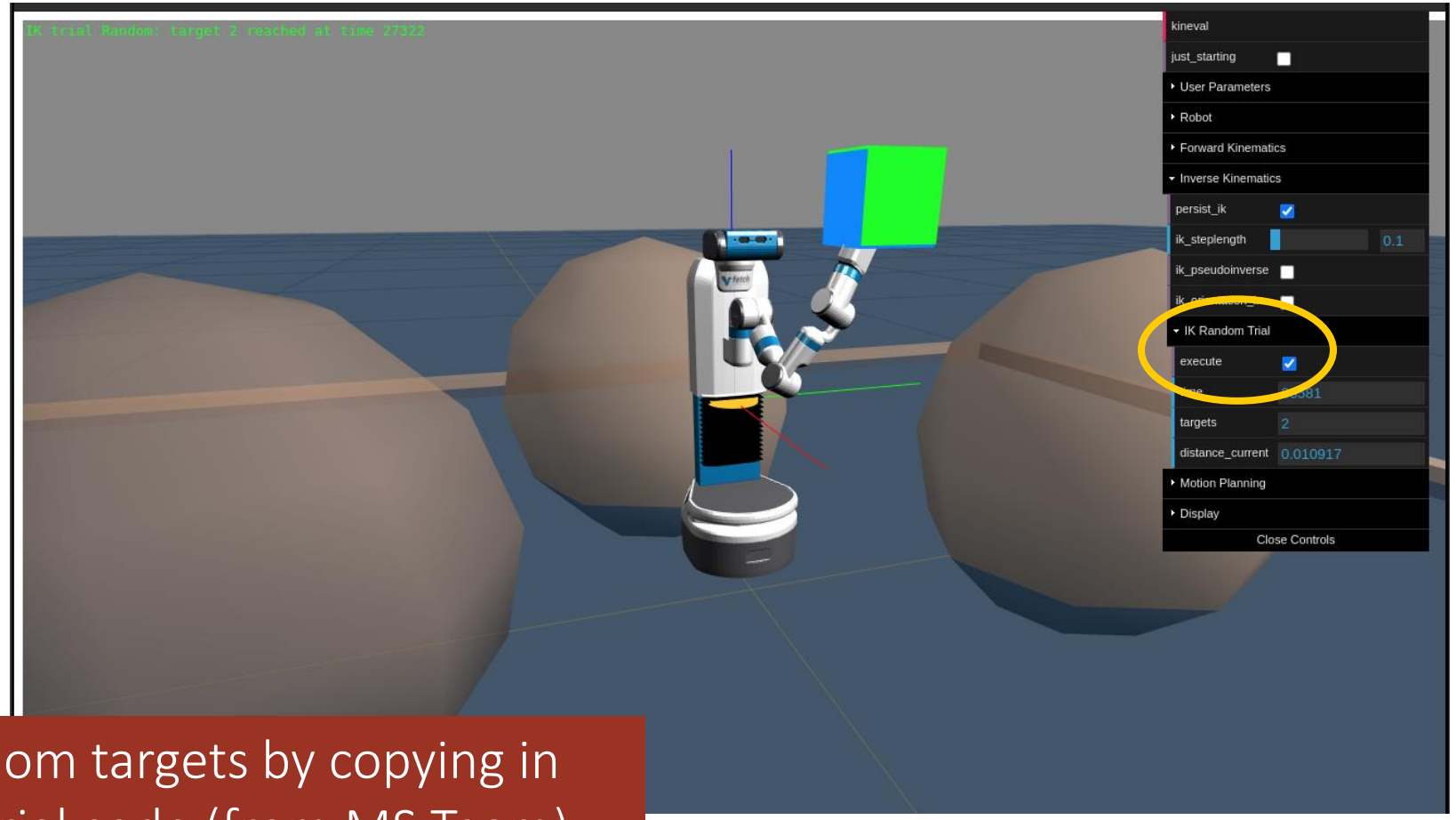
Correct IK implementation should allow robot's end effector (blue box) to reach the target (green box)

Running IK



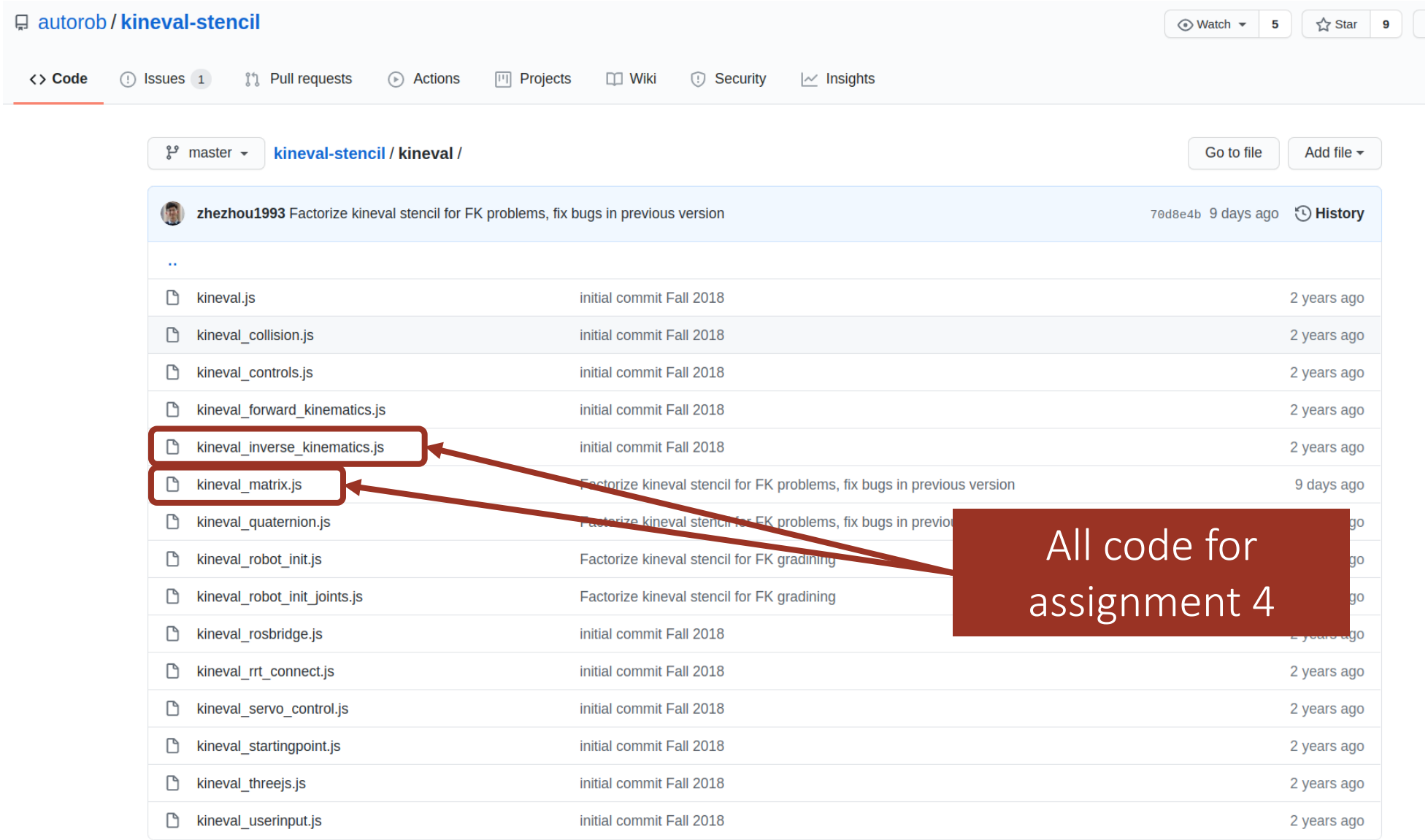
Test random targets by copying in random trial code (from MS Team) then selecting Random Trial->execute

Running IK



Test random targets by copying in random trial code (from MS Team) then selecting Random Trial->execute

KinEval Overview



kineval_inverse_kinematics.js

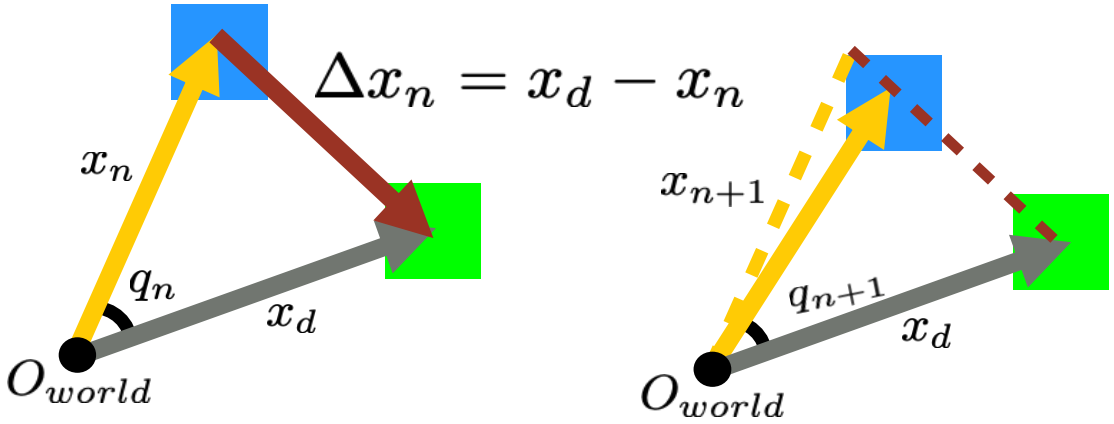
kineval_inverse_kinematics.js

```
19  kineval.robotInverseKinematics = function robot_inverse_kinematics(endeffector_target_world, endeffector_joint, endeffector_position_local) {
20
21      // compute joint angle controls to move location on specified link to Cartesian location
22      if ((kineval.params.update_ik)||(kineval.params.persist_ik)) {
23          // if update requested, call ik iterator and show endeffector and target
24          kineval.iterateIK(endeffector_target_world, endeffector_joint, endeffector_position_local);
25          if (kineval.params.trial_ik_random.execute)
26              kineval.randomizeIKtrial();
27          else // KE: this use of start time assumes IK is invoked before trial
28              kineval.params.trial_ik_random.start = new Date();
29      }
30
31      kineval.params.update_ik = false; // clear IK request for next iteration
32  }
33
34  kineval.randomizeIKtrial = function randomIKtrial () {
35
36      // update time from start of trial
37      cur_time = new Date();
38      kineval.params.trial_ik_random.time = cur_time.getTime()-kineval.params.trial_ik_random.start.getTime();
39
40      // STENCIL: see instructor for random time trial code
41  }
42
43  kineval.iterateIK = function iterate_inverse_kinematics(endeffector_target_world, endeffector_joint, endeffector_position_local) {
44
45      // STENCIL: implement inverse kinematics iteration
46
```

Implement `iterateIK()` such that each joint along the end effector path gets an update to its `.control` term

Translating the IK Update

IK UPDATE PER JOINT



GENERAL IK UPDATE PROCEDURE

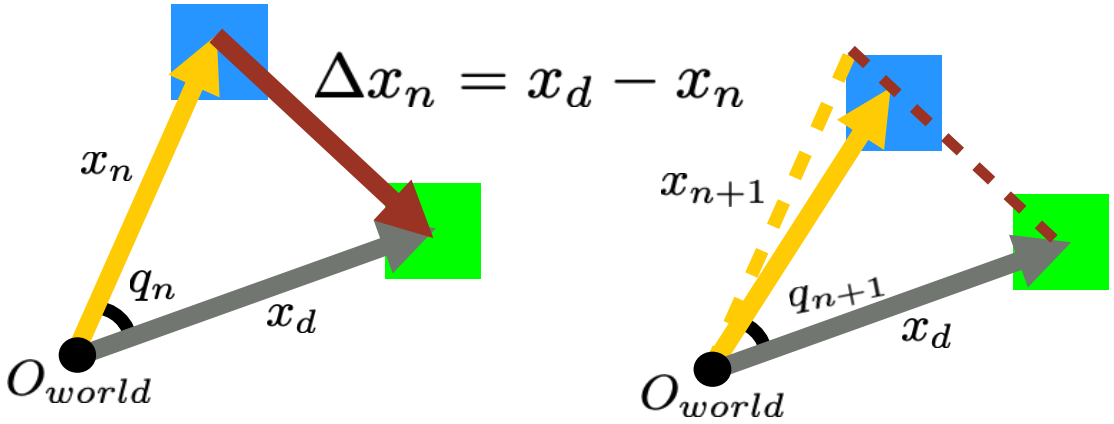
$$\Delta x_n = x_d - x_n$$
$$\Delta q_n = J(q_n)^{-1}\,\Delta x_n$$
$$q_{n+1} = q_n + \gamma \Delta q_n$$

= current end effector position

= desired end effector position

Translating the IK Update

IK UPDATE PER JOINT



GENERAL IK UPDATE PROCEDURE

$$\Delta x_n = x_d - x_n$$
$$\Delta q_n = J(q_n)^{-1}\,\Delta x_n$$
$$q_{n+1} = q_n + \gamma \Delta q_n$$

KINEVAL VARIABLES

$x_d \rightarrow$ `endeffector_target_world`

$q_n \rightarrow$ `robot.joints[…].angle`

$p^{x_n} \rightarrow$ `endeffector_position_local`

$x_n \rightarrow T^O_{x_n} p^{x_n}$

$T^O_{x_n} \rightarrow$ a `.xform`, calculated by FK

$\gamma \rightarrow$ `kineval.params.ik_steplength`

$\Delta x_n \rightarrow$ `robot.dx`

$J(q_n) \rightarrow$ `robot.jacobian`

$\Delta q_n \rightarrow$ `robot.dq`

} Necessary for CI grader!

KinEval IK Parameters

Parameters of `iterate_inverse_kinematics` function:

`endeffector_target_world` – target pose of end effector for IK, has `.position` and `.orientation`

`endeffector_joint` – string name of joint connected to end effector

`endeffector_position_local` – position of end effector with respect to local frame

Global parameters that your code needs to check:

`kineval.params.ik_steplength` – size of step to take along configuration gradient when updating control

`kineval.params.ik_pseudoinverse` – Boolean flag denoting which method to use (Jacobian transpose vs pseudoinverse)

Performance Validation

`kineval.randomizeIKTrial()`

- Source code will be provided on assignment 5 channel in MS Team
- Graduate extension points for reaching at least 100 targets in 60 seconds

Inverse kinematics will react in real time

- Turn on `persist_ik` in the GUI menu or hold down ‘p’ key to turn on
- IK will account for manual adjustments to robot base or joint angles
- Also will react to any modification of the end effector target

Keyboard controls