Final Project

5.3 Dynamic Manipulation

David Nichol (dan1)

Alden Higgins (jah17)

Sam Tormey (srt7)

Problem

The problem we are trying to solve is how to simulate a series of joints that are linked together. In other words, given a list of joints linked together, our program applies torques in response to gravity in order to move the joints such that each ends up in their own respective goal state. This problem is significant because in many robotic applications the robots will be formed not as a single box entity but as a chain of joints that are used to manipulate objects. Hence, this joint simulation solves a problem that is vital for robotic movement.

Our Approach

Our project solution is centered over the ODE that we created for the joint manipulator. In our program, we follow the math outlined in Leon Zlajpah's paper “Dynamic simulation of n-R planar manipulators.” By converting the mathematical model to C++, we are able to solve for the qdots of each joint.

In order to represent the dynamic space, we created a compund state space with one SO2 state space and 1 Real Vector State Space for each joint. This means that we will have n * (SO2 + RealVectorStateSpace) spaces, or 2n total spaces.

In order to reach the goal states, we apply RRT to our controls and keep iterating until we have found our final goal states for each joint.

Experiments Conducted

We tested our program by calculating some simple joints by hand, such as 1 and 2 joint simulations with no torque, and walked through the sections of math with the debugger to ensure that each section was calculating the correct value. For higher dimensions, we used our visualizer to look at what was happening with the joints to verify that the way that the joints moved was realistic and probable.

We experimented mostly on 1, 2, and 3 joints, and saw that the joints would eventually reach their designed spots. While most of this we could easily see in the simulator, we could also see it quantitatively, as the final states that were printed to console would reach the intended goal states.

We wrote the output from these experiments as a list of angles. We then read them into our visualizer that was written as a Matlab graph. This graph shows the motions of these joints over time.

In our experiments, we assumed that each joint had an equal weight and mass, however, this program is easily extensible such that if we wanted to add different lengths or weights of joints, minimal code will have to be changed.

Analysis

We saw the joints very quickly approach the goal states whenever we ran our visualizations. For end states that were hanging down, the joints would fall down due to gravity. For joints that were on the other side of the start states (which we typically tested as every joint pointed straight to the right), the joints would swing around. Quantitatively speaking, for example, we saw a joint with a start state SO2: 0, Real Vector 0 and a goal state SO2: -3.13, Real Vector 0 move like so, with the first number on each line being the angle of the joint and the second number being the current angular velocity of the joint:

0 0

```
-0.0735617 -1.4707
-0.293454 -2.91769
-0.652703 -4.22775
-1.1263 -5.15463
-1.66115 -5.41387
-2.18319 -4.90733
-2.6231 -3.81899
-2.9375 -2.4423
-3.10889 -0.980906
-3.13342 0.490485
```

To give a 2 joint example, start states SO2: 0,0, Real Vector 0,0, and an end state SO2: -3.13 1.27 Real Vector 0,0 move like so (again the indexes start as joint angles and alternate between angular velocities):

0 0 0 0

```
-0.272271 -5.40538 -0.209142 -4.14032
-1.03446 -9.28227 -0.779587 -6.70775
-1.96667 -8.55981 -1.40442 -5.20265
-2.64771 -4.87026 -1.78006 -2.39396
-2.93303 -0.858263 -1.91166 -0.350505
```

And here are the numbers for a 3 joint example, start states SO2: 0,0,0 Real Vector: 0,0,0 and end states SO2: -1.13, -1.57, 1.57 Real Vector: 0, 0, 0:

```
0 0 0 0 0 0

-0.00281943 -0.188103 0.0257995 1.72015 0.0247559 1.65028

-0.0128696 -0.481806 0.0523458 0.0497358 0.0264492 -1.53704

-0.021074 -0.0642728 0.0224904 -2.0409 -0.0356448 -2.60136

-0.0165666 0.370036 -0.0701639 -4.13669 -0.129333 -3.63356

-0.0286216 -1.16849 -0.18469 -3.49471 -0.230212 -3.06722

-0.0863814 -2.6773 -0.279391 -2.80684 -0.311651 -2.32921

-0.170681 -2.93982 -0.368306 -3.09692 -0.359045 -0.800434

-0.262363 -3.16076 -0.463663 -3.23348 -0.357397 0.957119

-0.353822 -2.92124 -0.563093 -3.3656 -0.318389 1.69844

-0.436643 -2.58265 -0.6638 -3.31854 -0.252038 2.78362

-0.520823 -3.02049 -0.771634 -3.82898 -0.196444 0.969369
```

```
-0.616953 -3.36788 -0.890902 -4.07582 -0.190011 -0.462049
-0.721488 -3.57548 -1.0133 -4.03542 -0.218912 -1.37121
-0.818932 -2.90009 -1.13176 -3.81594 -0.244343 -0.246293
-0.894342 -2.10908 -1.2399 -3.36165 -0.229526 1.29572
-0.963258 -2.47299 -1.31164 -1.40003 -0.184324 1.75642
-1.03648 -2.40187 -1.30154 2.08023 -0.131908 1.75535
-1.10008 -1.83297 -1.24588 1.62612 -0.0804592 1.68186
-1.1462 -1.23779 -1.20418 1.15059 -0.0306704 1.64079
-1.17836 -0.904229 -1.17066 1.08099 0.0071395 0.880529
-1.20034 -0.559386 -1.13952 0.991616 0.0221899 0.122922
-1.21186 -0.208184 -1.11136 0.882281 0.0144884 -0.637342
-1.20235 0.840158 -1.09304 0.335463 -0.000243886 -0.351559
-1.16697 1.51199 -1.09794 -0.665642 0.0235516 1.92017
-1.13555 0.579927 -1.12498 -1.13482 -0.00788971 -4.01945
-1.09363 2.21252 -1.21504 -4.85618 -0.0462622 1.46717
-1.02756 2.18141 -1.29509 -0.475556 0.03962 4.23578
-0.976594 1.20509 -1.32054 -1.22643 0.148975 3.02441
-0.955535 0.199245 -1.36885 -1.99321 0.220302 1.73245
-0.939903 0.837703 -1.43011 -2.08121 0.283723 2.48659
-0.905764 1.42717 -1.49319 -2.1159 0.368475 3.13883
-0.855075 1.93697 -1.55662 -2.10652 0.470182 3.60525
-0.821894 0.295362 -1.59249 -0.299792 0.511448 -0.819344
-0.820741 -0.219524 -1.60895 -0.794732 0.51724 1.20625
-0.835595 -0.787843 -1.63972 -1.24376 0.583224 3.1718
-0.877825 -2.03667 -1.62239 2.39484 0.695373 4.30202
-0.921926 -0.904089 -1.57607 0.692567 0.81782 3.82677
-0.931965 0.238467 -1.58084 -1.00931 0.923244 3.17769
-0.939717 -0.746585 -1.61372 -1.18859 1.00743 2.45697
-0.976217 -1.67736 -1.65227 -1.37983 1.07214 1.88368
-1.03321 -2.1121 -1.72172 -3.238 1.11846 1.23277
-1.08501 -1.33296 -1.81928 -3.25004 1.15642 1.3167
-1.11281 -0.515381 -1.91594 -3.18378 1.19826 1.48111
-1.11577 0.318787 -2.0098 -3.06686 1.24548 1.66529
-1.11205 -0.0671975 -2.06437 -0.573462 1.29639 1.73561
-1.13539 -1.52873 -2.00919 4.24489 1.38882 4.41349
-1.18289 -1.60517 -1.89826 3.13226 1.49633 2.75657
```

```
-1.2301 -1.52183 -1.82205 1.93918 1.55424 1.10369
-1.2734 -1.35812 -1.7822 0.716683 1.56257 -0.546184
-1.29147 0.148432 -1.7228 3.24099 1.57993 1.68949
-1.27495 0.956959 -1.67674 -0.176615 1.6013 -0.277019
-1.25651 0.268472 -1.67328 0.404494 1.60734 0.67612
-1.26407 -0.771164 -1.6696 -0.160044 1.61481 -0.17628
-1.27105 0.306758 -1.67028 0.11551 1.61714 0.331694
-1.26576 0.045587 -1.67154 -0.200085 1.62405 0.127744
```

So, as we see, the joints are moving towards and getting very close to their final states.

Concluding Remarks

We enjoyed getting a chance to run through Zlajpah's paper and implement it step by step as well as visualizing the joint movement when everything came together. We ran into some issues that ended up being due to our makefile including a header from the wrong folder, but other than that there wasn't anything to dislike or major hitches in the project. Overall we would rate the project difficulty at a 6, and we spent a total of around 20 hours on it.