Eric Braun 10121660

13eb20@queensu.ca

CISC 472

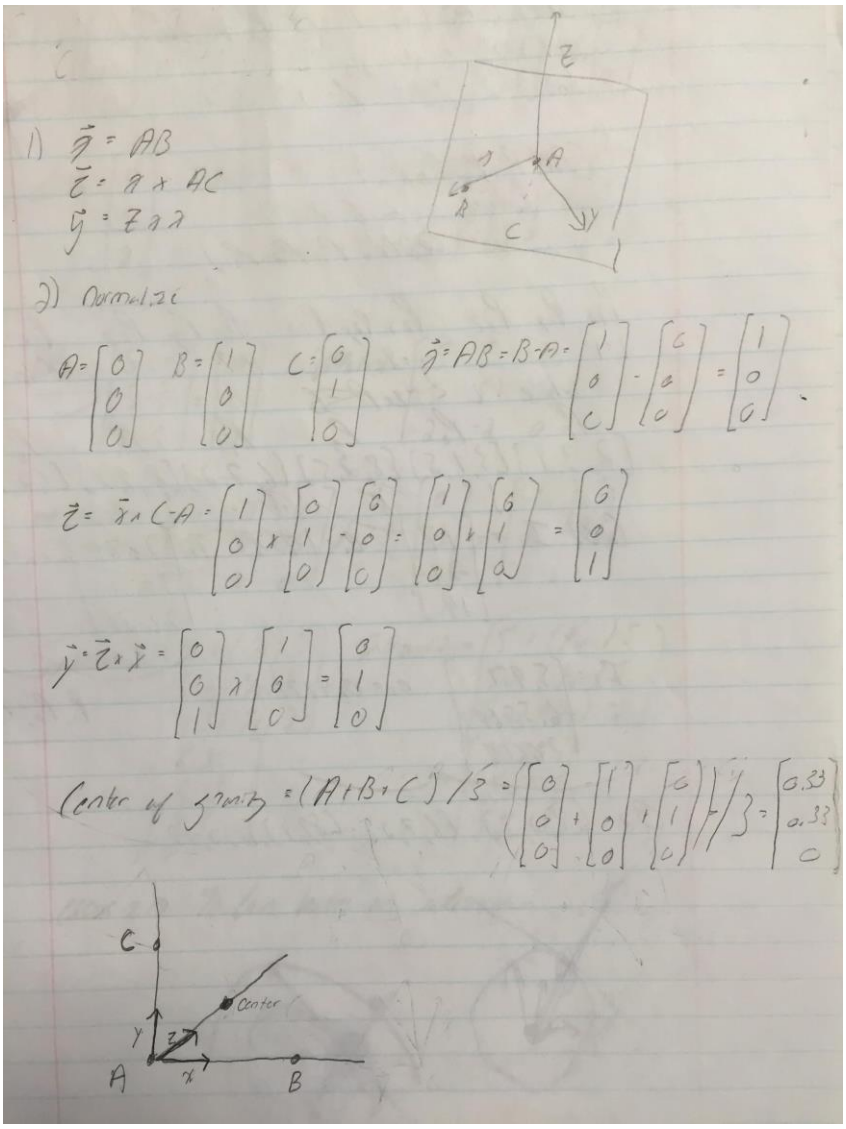
Assignment 1

Marker Registration

1. Cartesian System

Procedure to create a Cartesian system using three fiducial markers:

- x = the vector from A to B: B – A
- z = the cross product between x and AC: cross(x, C-A)
- y = the cross product between z and x: cross(z, x)
- I choose to use the center of gravity as the center point
    - center = (A + B + C)/3

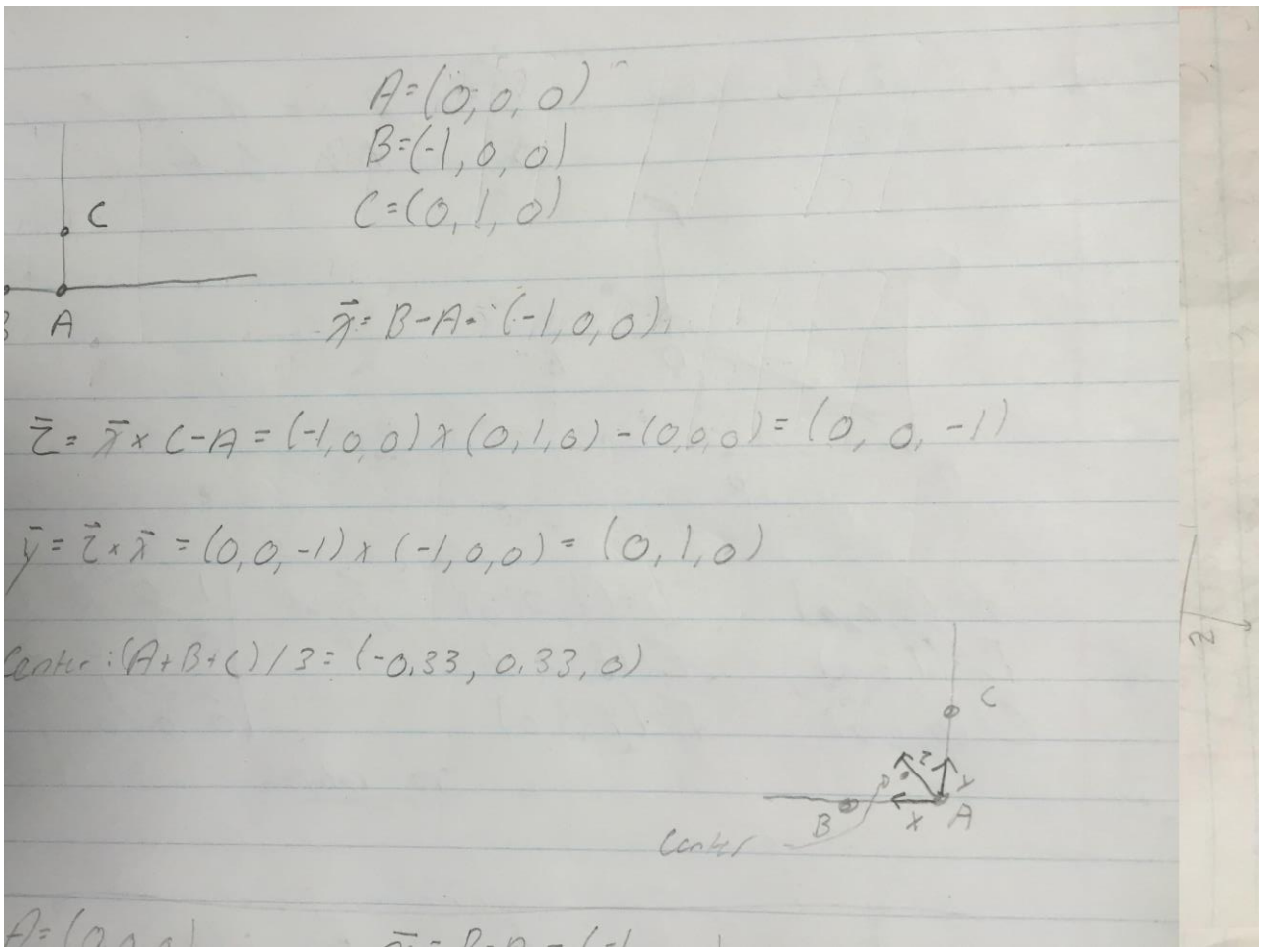


Command Window

File Edit Debug Desktop Window Help

```
>> [x,y,z,c] = CartesianSystem([0,0,0],[1,0,0],[0,1,0])

x =

     1     0     0


y =

     0     1     0


z =

     0     0     1


c =

    0.3333    0.3333         0

>>
```

OVR

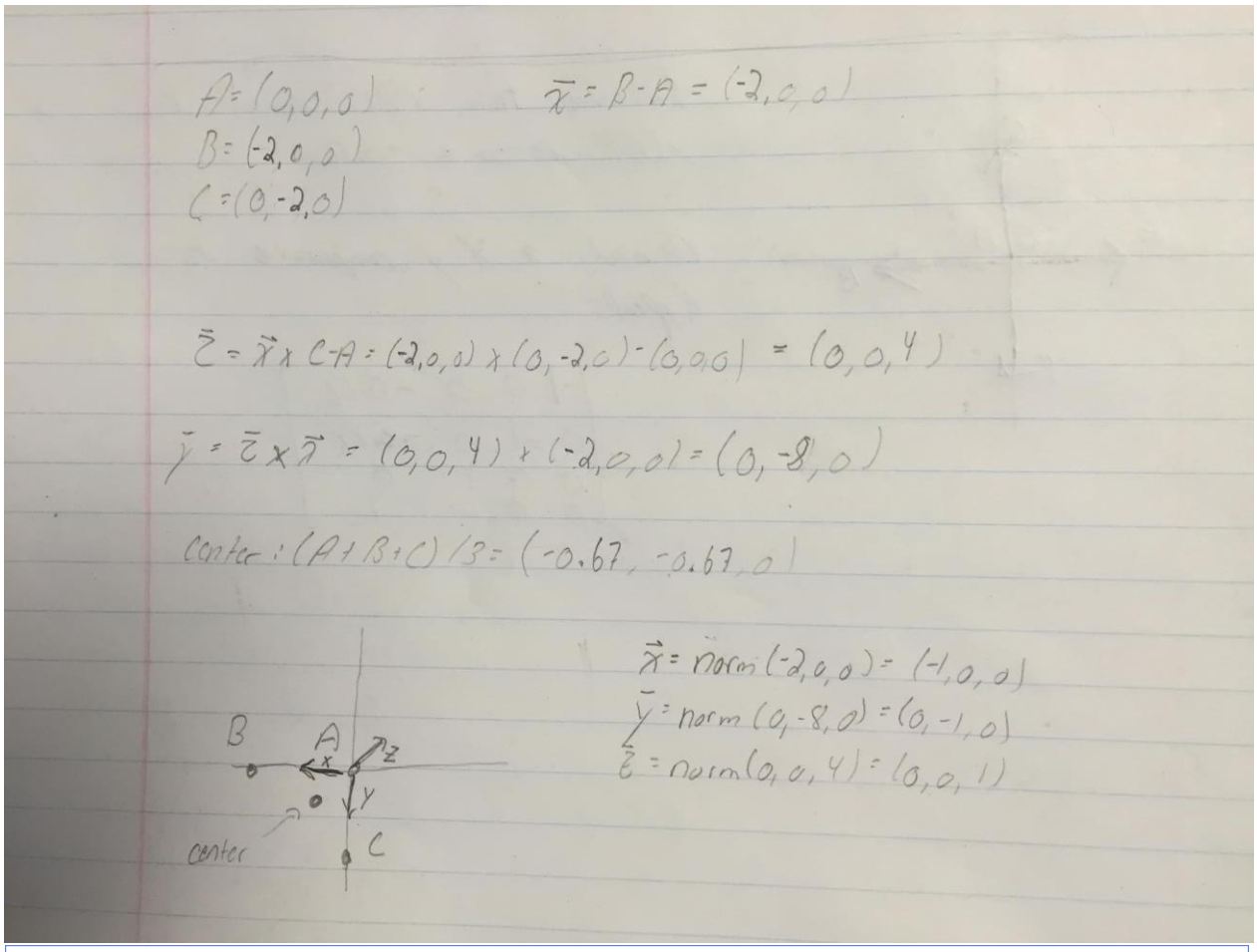


Command Window

File Edit Debug Desktop Window Help

```
>> [x,y,z,c] = CartesianSystem([0,0,0],[-1,0,0],[0,1,0])

x =

    -1     0     0


y =

     0     1     0


z =

     0     0    -1


c =

   -0.3333    0.3333         0

fx >>
```

OVR



Command Window

File Edit Debug Desktop Window Help

```
>> [x,y,z,c] = CartesianSystem([0,0,0],[-2,0,0],[0,-2,0])

x =

    -1     0     0


y =

     0    -1     0

z =

     0     0     1


c =

   -0.6667   -0.6667         0

>>
```

OVR

2. Fiducial Registration

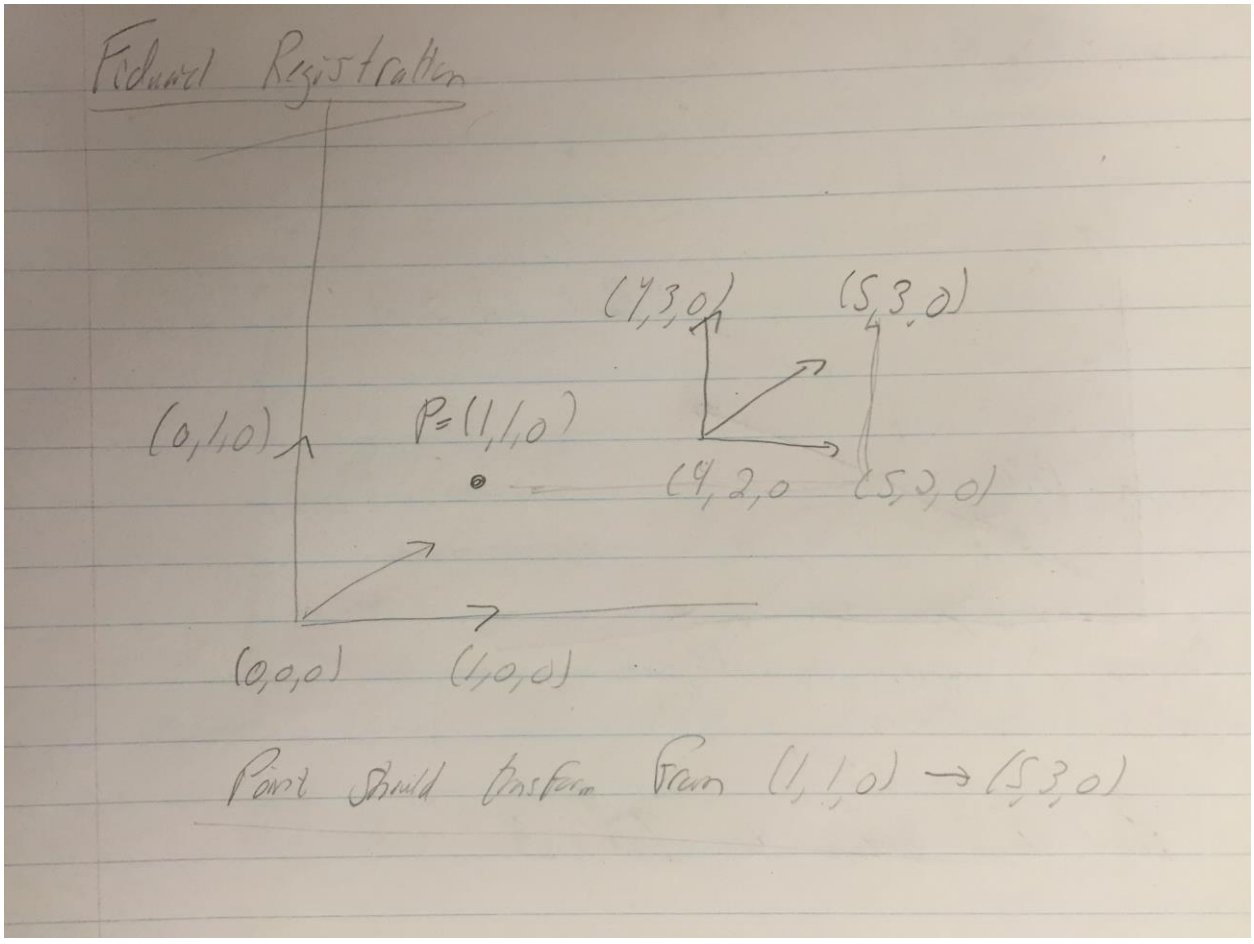
The homogeneous 4x4 frame transformation matrix includes the transformations for scaling, rotating, and translating points in that order.

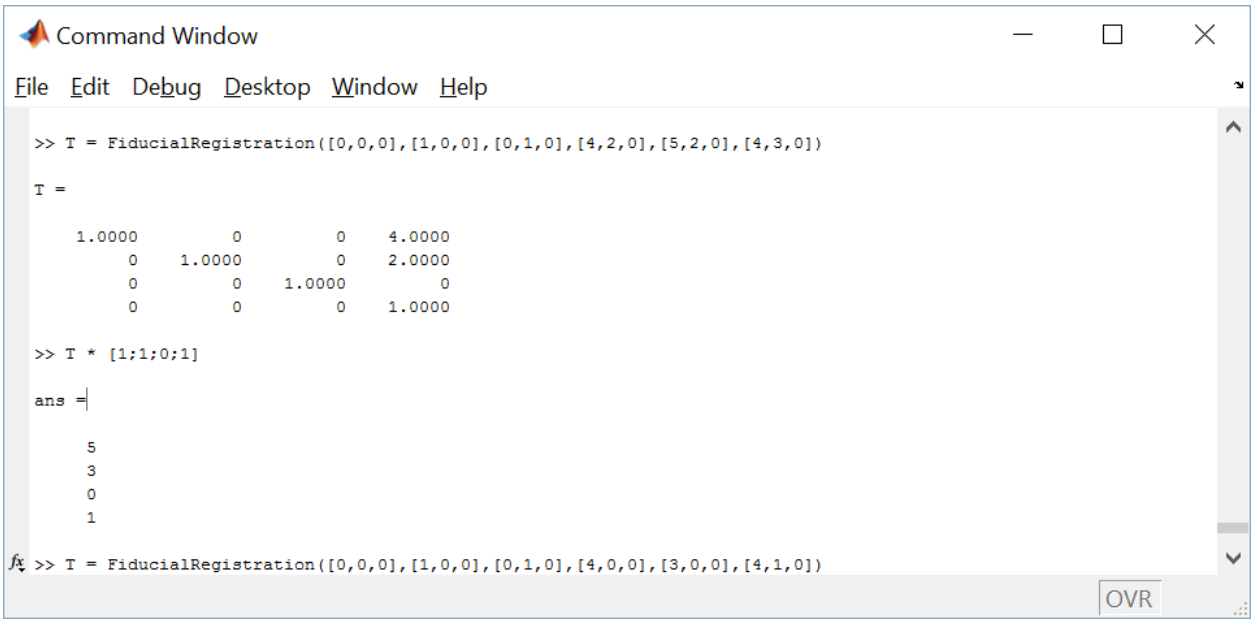
- We can omit scaling because our markers will be perceived as the same size.

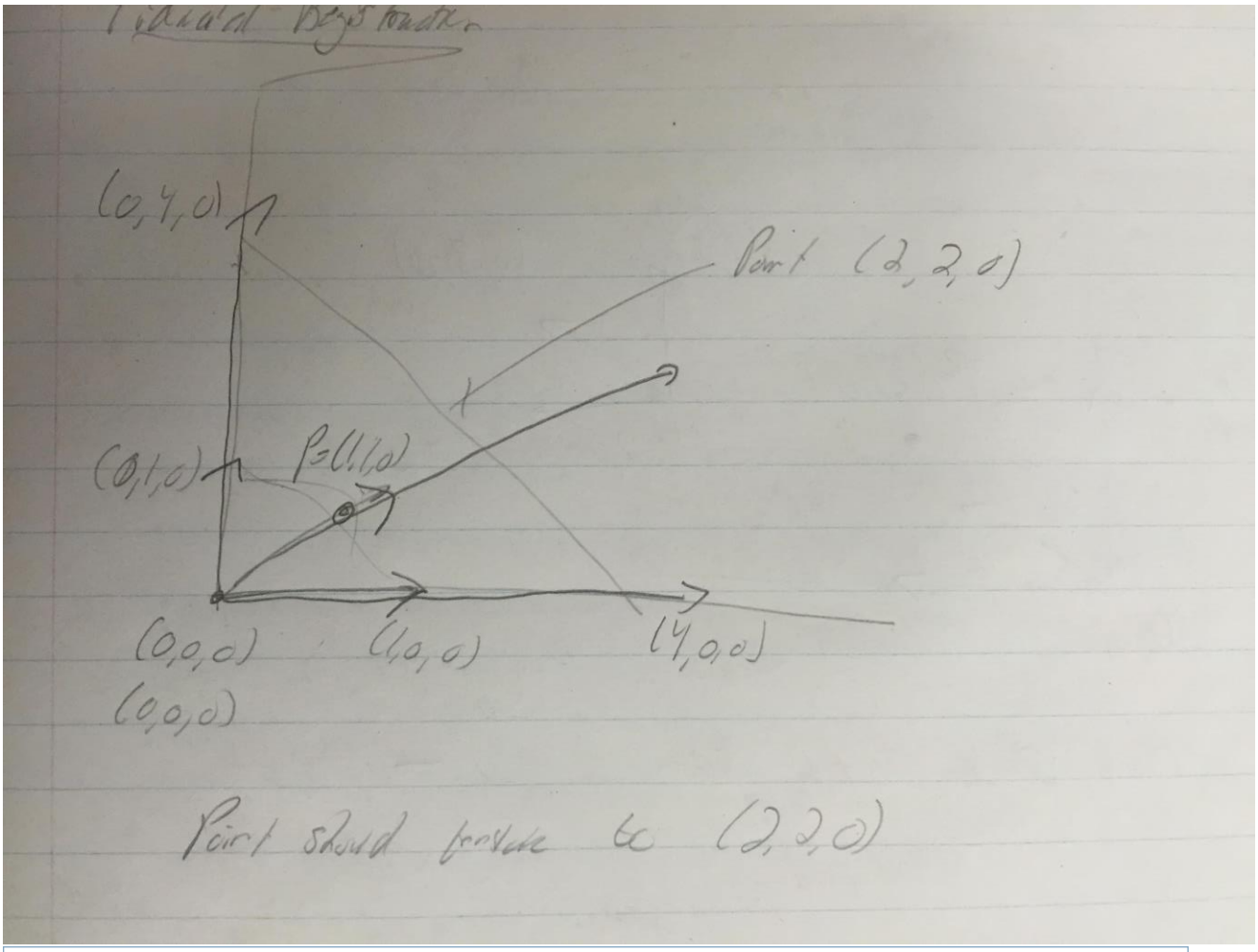
Procedure to create 4x4 transformation matrix using 2 sets of 3 fiducial markers:

- Create 2 orthonormal basis vector systems using function CartesianSystem from question 1.
- Create the rotational matrix between the two system using dot product from notes
- Create translation matrix using difference of centers from notes
- Use the rotational and translational matrix to make 4x4 transformation matrix.

Expect the point to move with relation to the system.




Command Window
File Edit Debug Desktop Window Help
>> T = FiducialRegistration([0,0,0],[1,0,0],[0,1,0],[4,2,0],[5,2,0],[4,3,0])
T =
1.0000 0 0 4.0000
0 1.0000 0 2.0000
0 0 1.0000 0
0 0 0 1.0000
>> T * [1;1;0;1]
ans =
5
3
0
1
>> T = FiducialRegistration([0,0,0],[1,0,0],[0,1,0],[4,0,0],[3,0,0],[4,1,0])
OVR

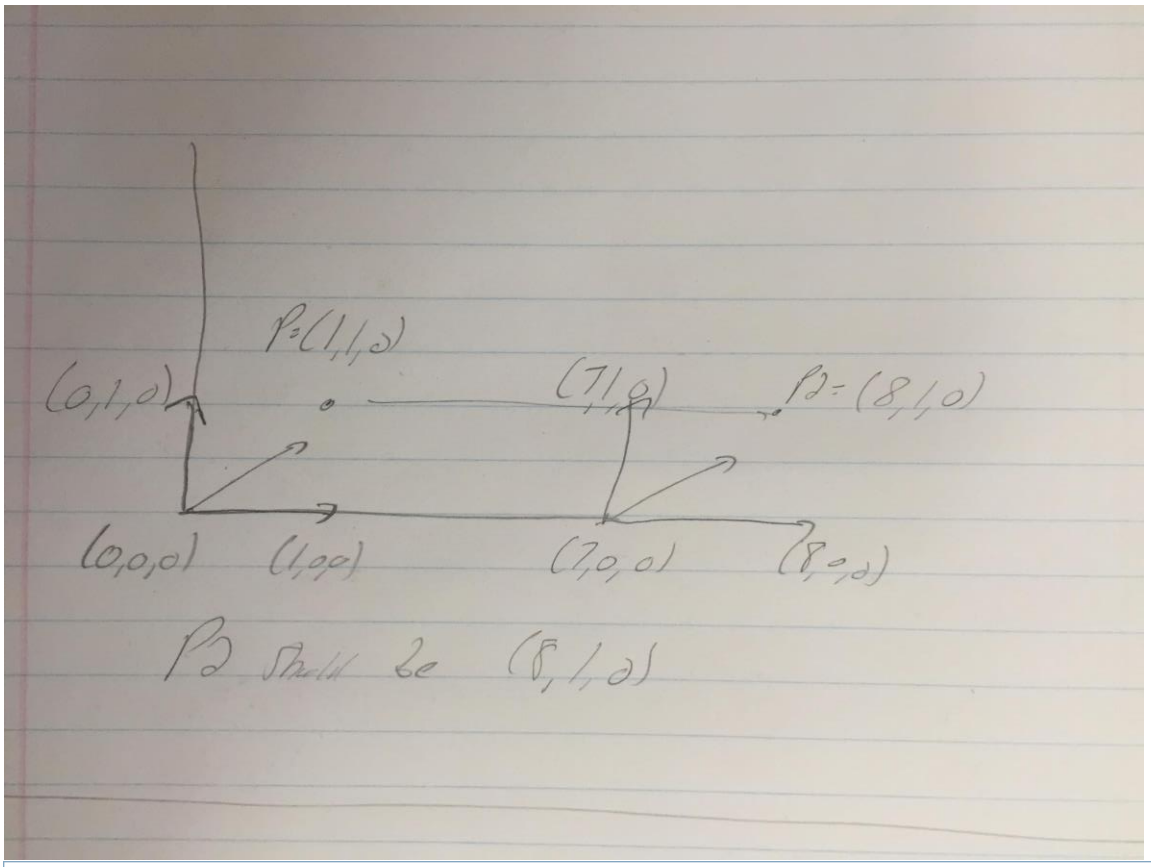


Command Window

File Edit Debug Desktop Window Help

```
>> T = FiducialRegistration([0,0,0],[1,0,0],[0,1,0],[0,0,0],[4,0,0],[0,4,0])

T =

     1     0     0     1
     0     1     0     1
     0     0     1     0
     0     0     0     1

>> T * [1;1;0;1]

ans =

     2
     2
     0
     1

>>
```

OVR



Command Window

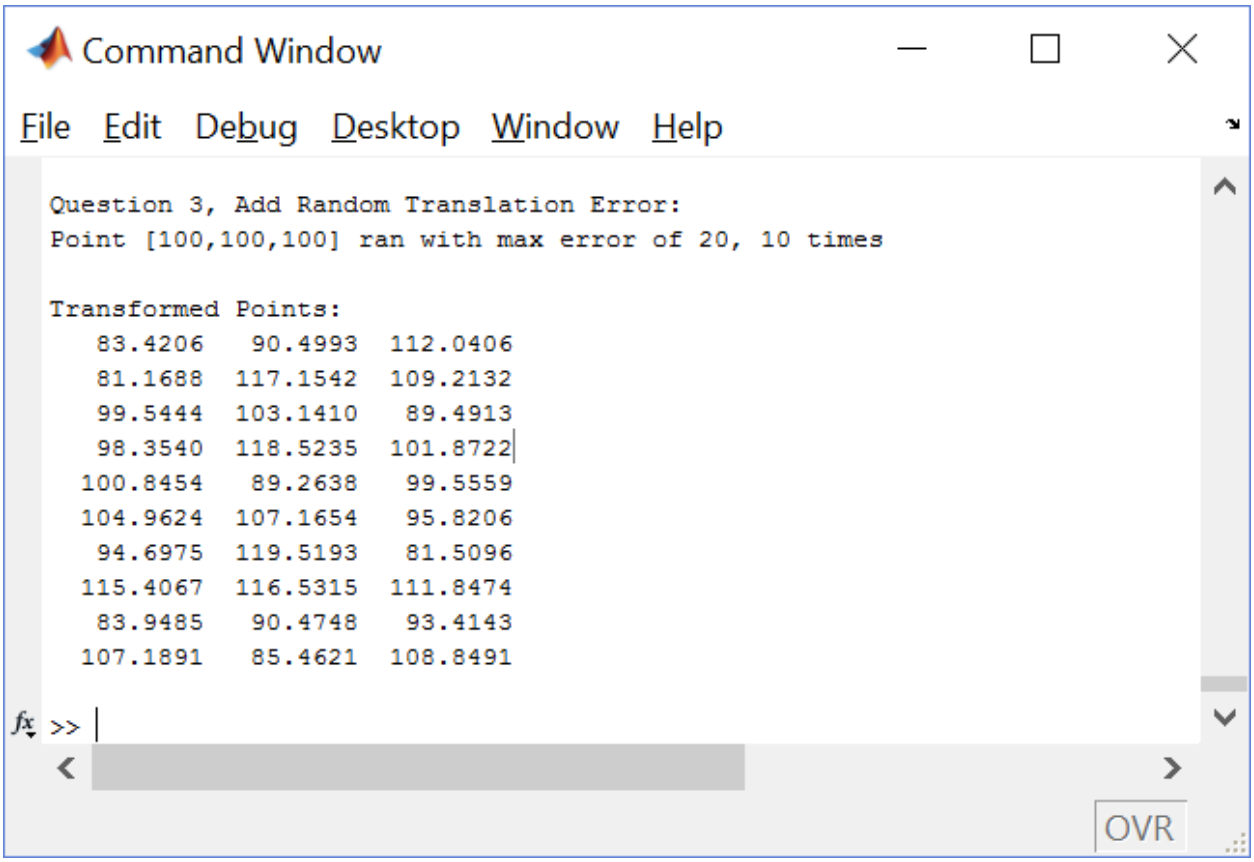
File Edit Debug Desktop Window Help

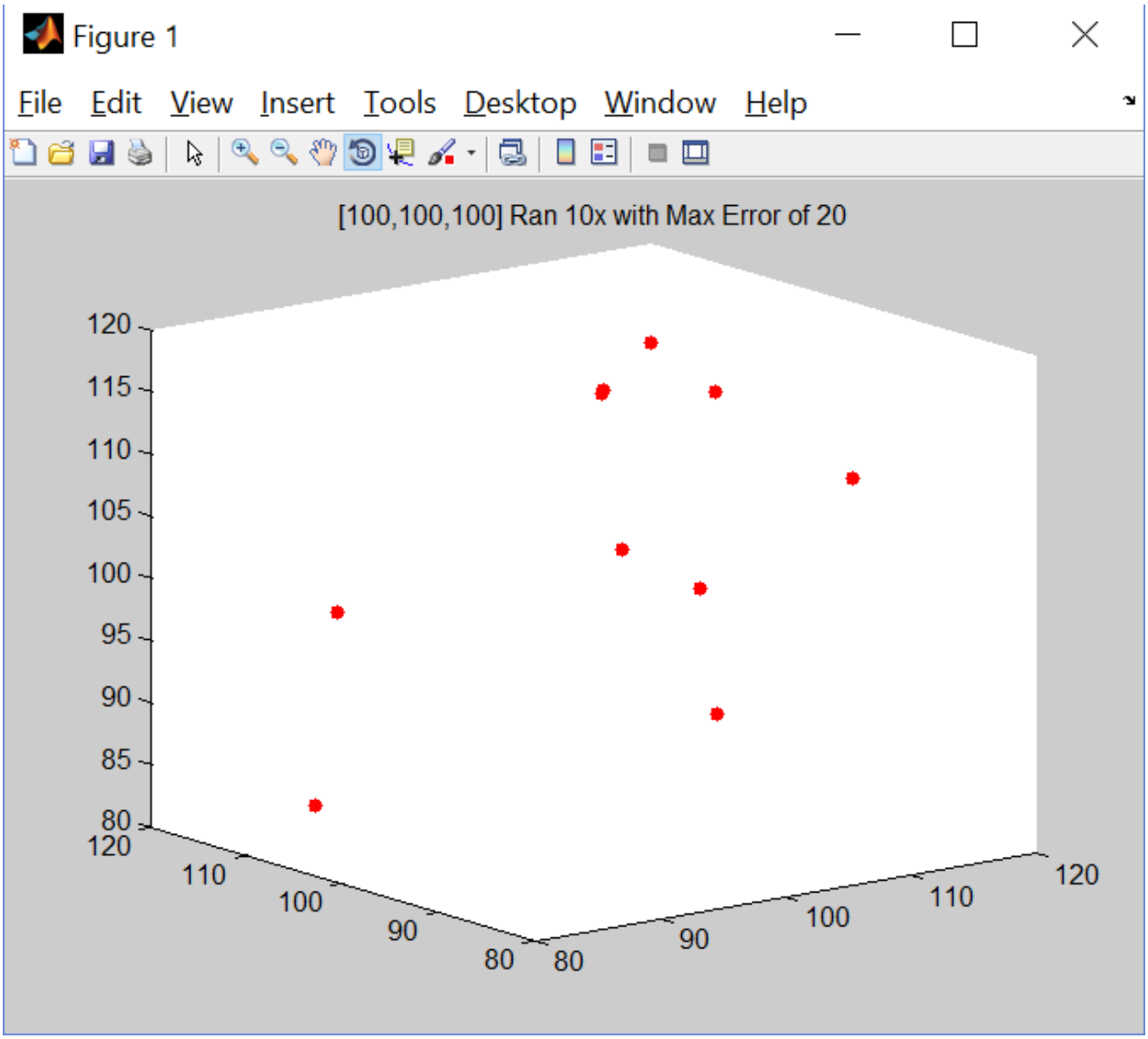
```
>> T = FiducialRegistration([0,0,0],[1,0,0],[0,1,0],[7,0,0],[8,0,0],[7,1,0])

T =

     1     0     0     7
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> T * [1;1;0;1]

ans =

     8
     1
     0
     1

>>
```

OVR

3. Add Random Translation Error

Test 20 cases E max 20 at Point = [100,100,100]:





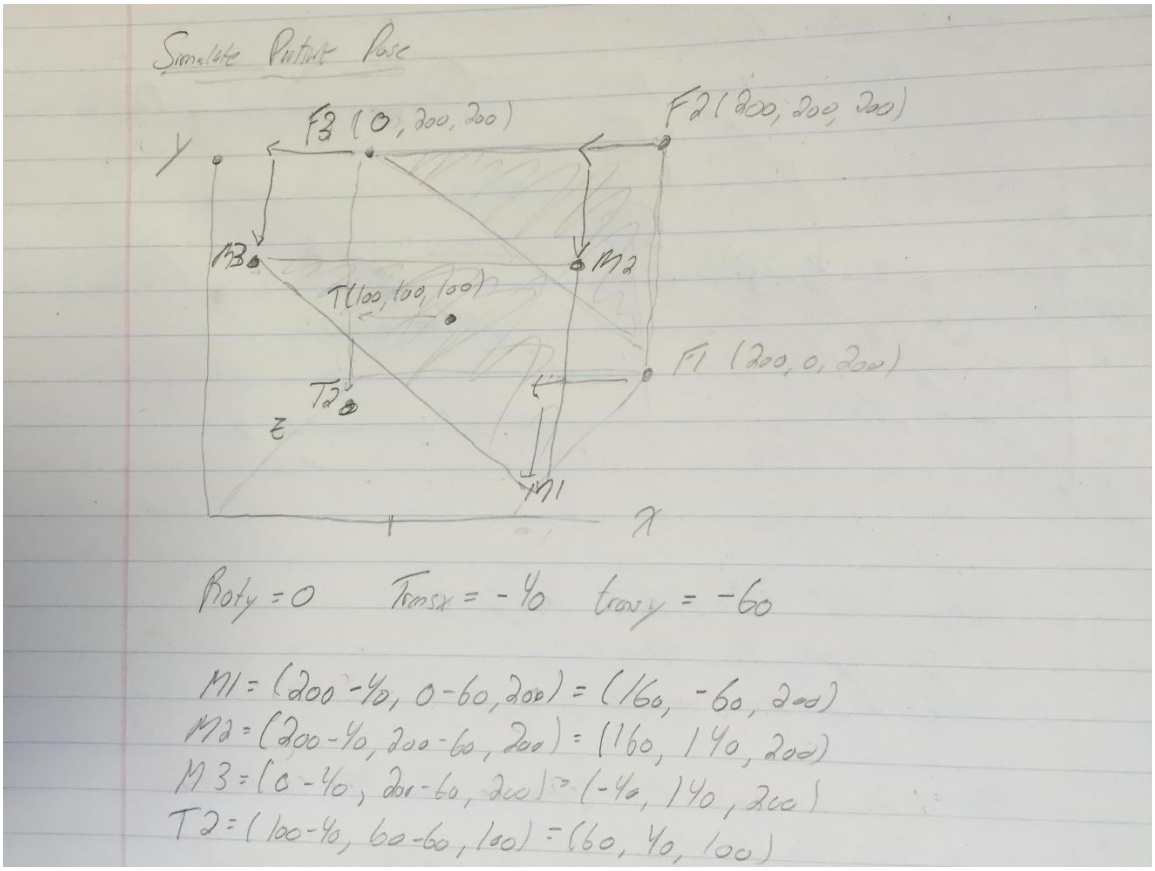
4. Simulate Patient Pose

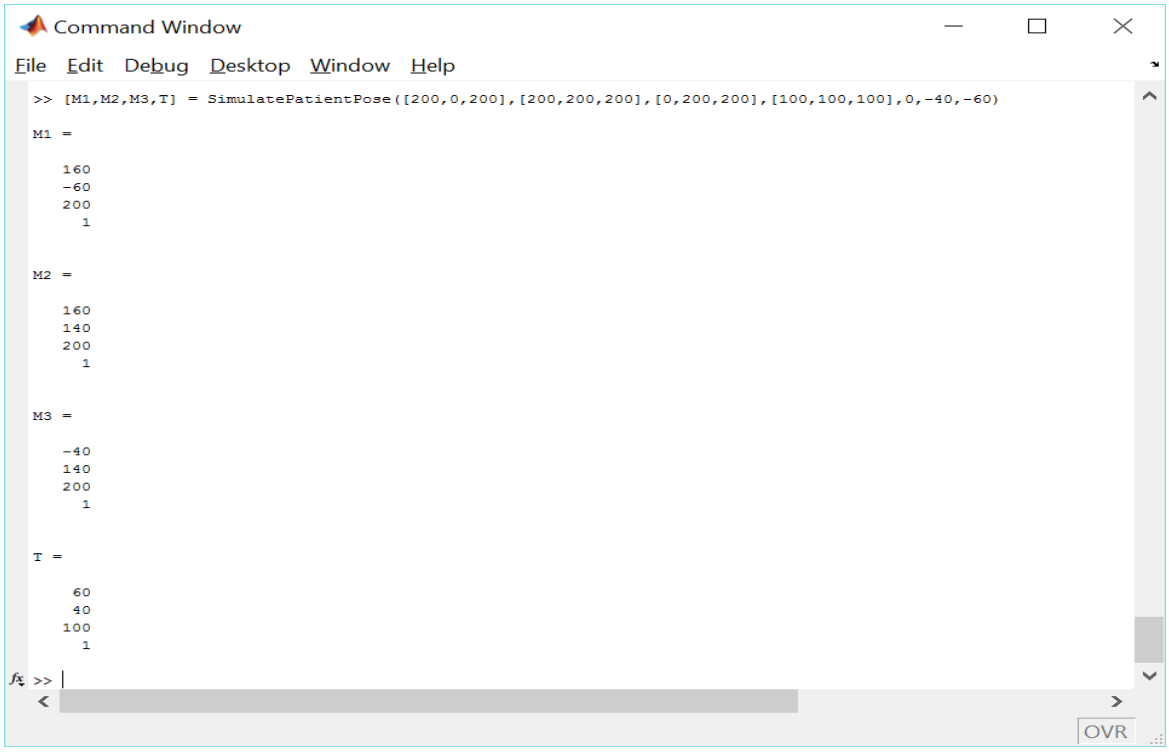
Procedure to create a simulated patient pose given 3 fiducial markers, a target, a scalar rotation around the y-axis, and scalar translations for the x and y axes.

- Convert the degrees to radians
- Implement this radian value into the rotation about the y-axis matrix given in the notes with buffer (see code)
- Create the translation matrix using the identity with the additional column of translated x, and y values and a buffer.
- Add the additional 1 buffer to markers and target
- Multiply the markers and points first by the rotation matrix followed by the translation matrix.

SimulatePatientPose called on Points F1: [200,0,200], F2: [200,200,200], F3: [0,200,200], Target: [100,100,100].

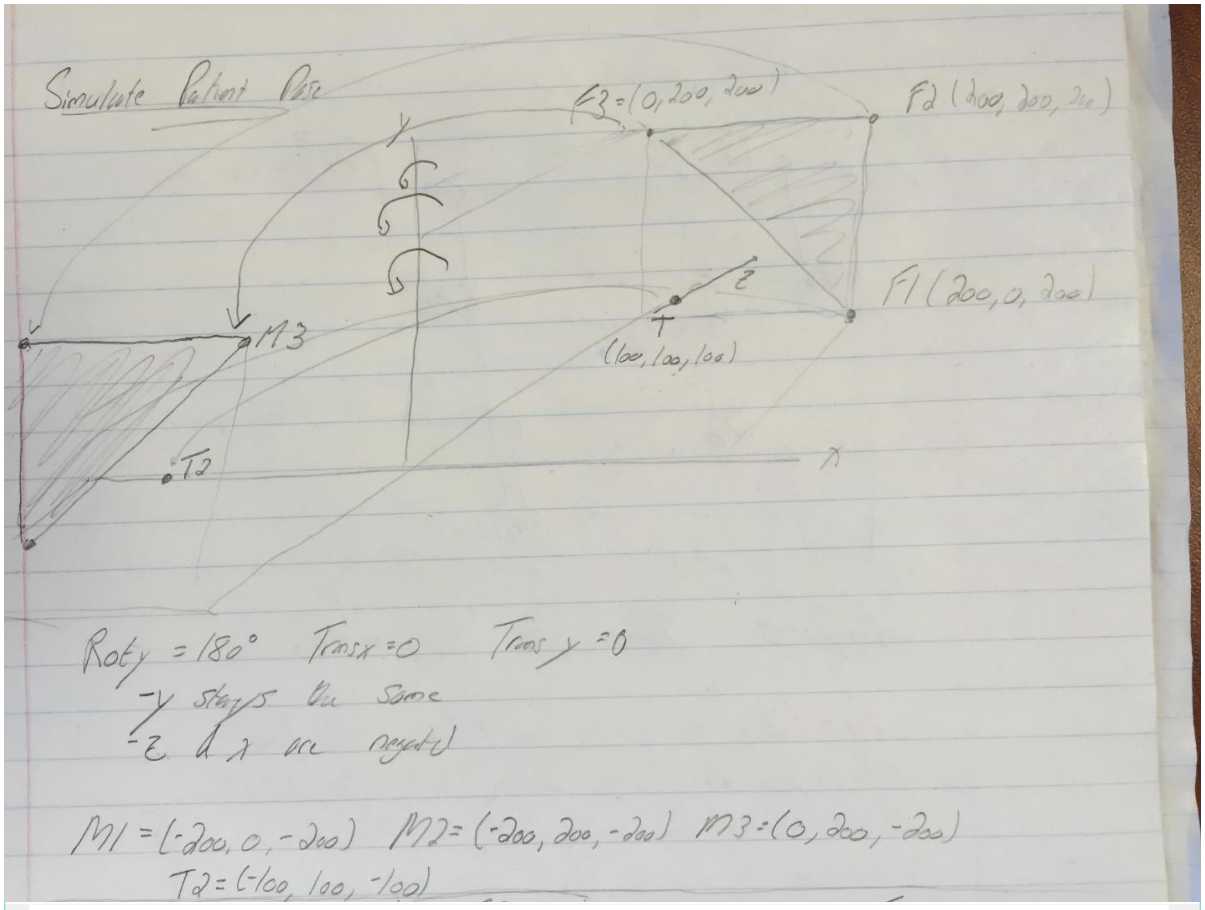
- Roty: 0. Transx: -40: Transy: -60





SimulatePatientPose called on Points F1: [200,0,200], F2: [200,200,200], F3: [0,200,200], Target: [100,100,100].

- Roty: 180. Transx: 0: Transy: 0.

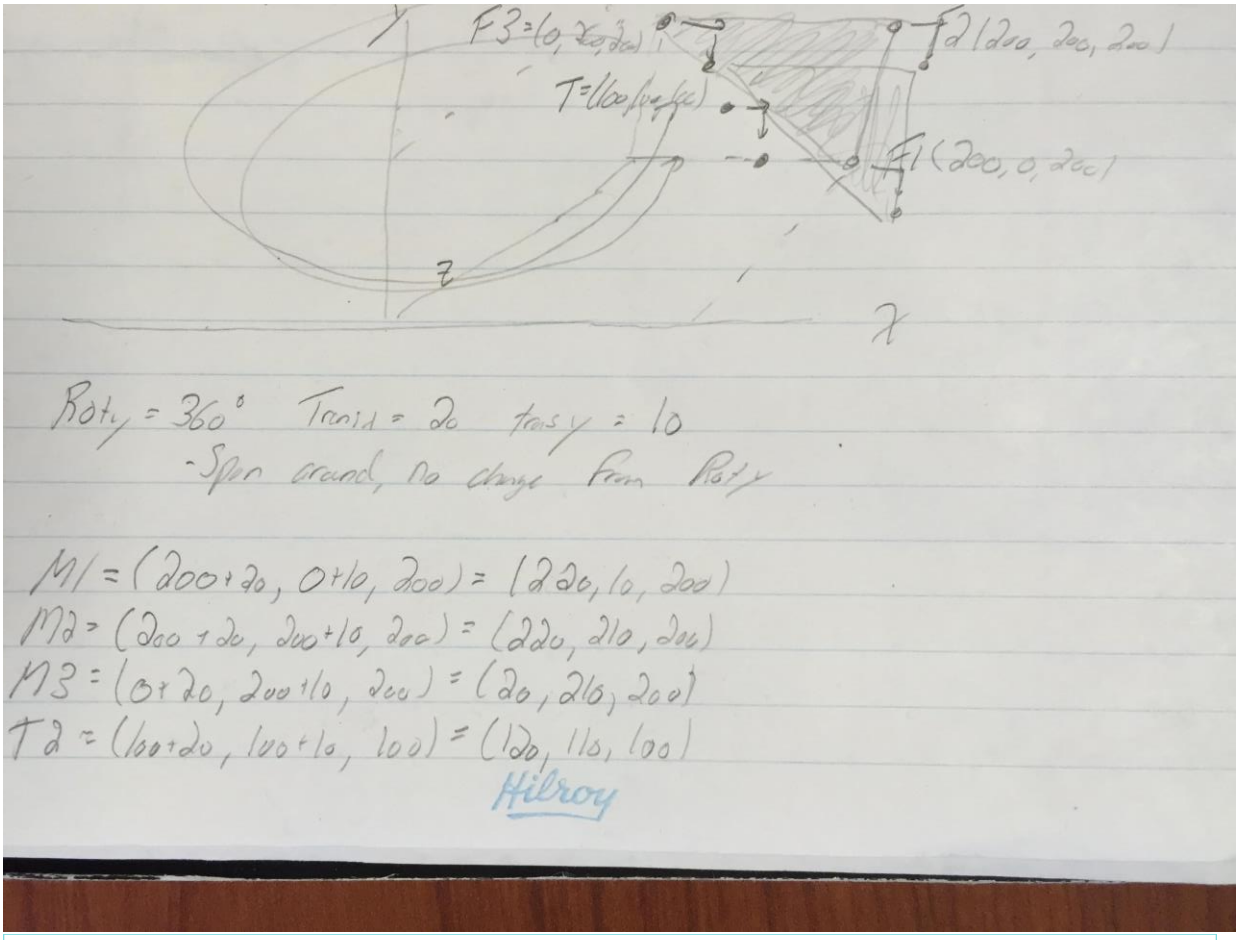



```
>> [M1,M2,M3,T] = SimulatePatientPose([200,0,200],[200,200,200],[0,200,200],[100,100,100],180,0,0)

M1 =

  -200.0000
         0
  -200.0000
     1.0000

M2 =

  -200.0000
   200.0000
  -200.0000
     1.0000

M3 =

    -0.0000
   200.0000
  -200.0000
     1.0000

T =

  -100.0000
   100.0000
  -100.0000
     1.0000

>>
```

OVR

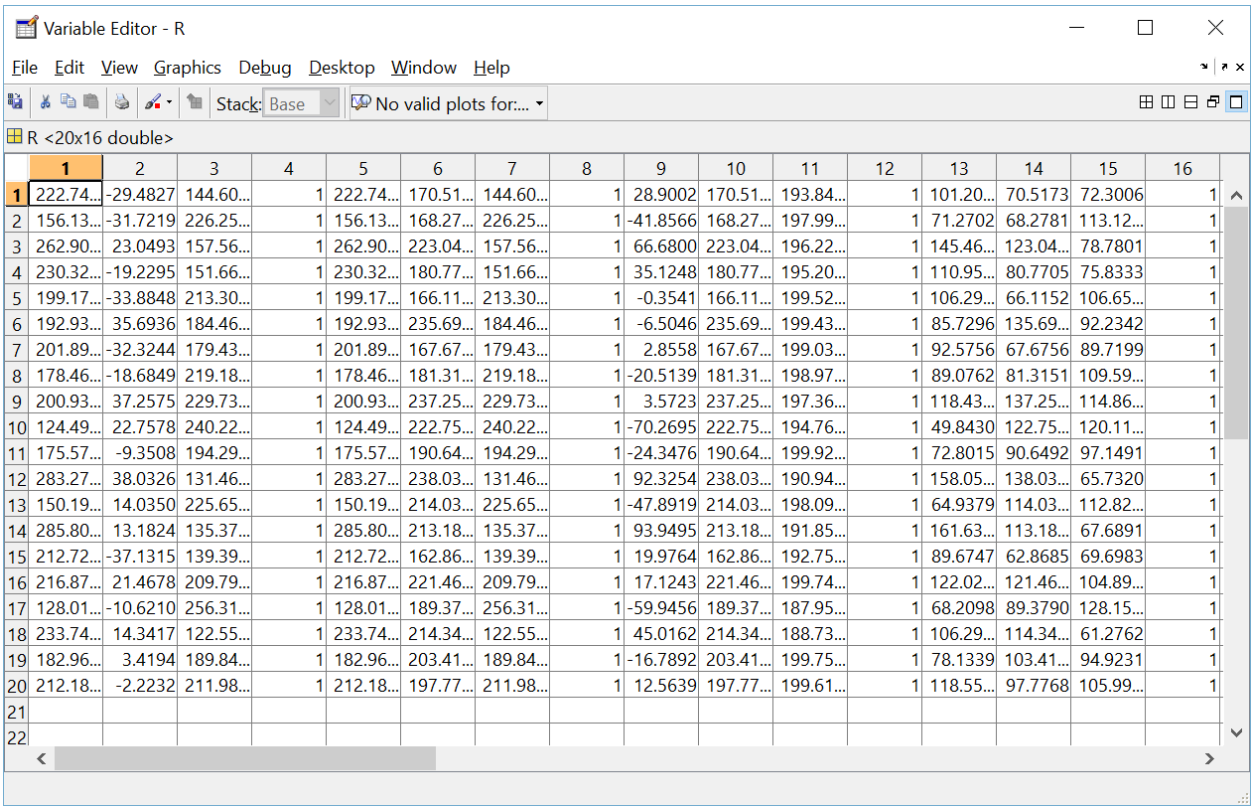
SimulatePatientPose called on points: F1: [200,0,200], F2: [200,200,200], F3: [0,200,200], Target: [100,100,100]

- Roty: 360. Transx: 20. Transy: 10.



5. Simulate Many Patient Poses:

This is the results of running the patient poses with 20 random poses +/- 20 degrees rotation around the y-axis with 0-40 random translation in either direction for both x and y.



Variable Editor - R

File Edit View Graphics Debug Desktop Window Help

Stack: Base | No valid plots for:...

R <20x16 double>

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	222.74...	-29.4827	144.60...	1	222.74...	170.51...	144.60...	1	28.9002	170.51...	193.84...	1	101.20...	70.5173	72.3006	1
2	156.13...	-31.7219	226.25...	1	156.13...	168.27...	226.25...	1	-41.8566	168.27...	197.99...	1	71.2702	68.2781	113.12...	1
3	262.90...	23.0493	157.56...	1	262.90...	223.04...	157.56...	1	66.6800	223.04...	196.22...	1	145.46...	123.04...	78.7801	1
4	230.32...	-19.2295	151.66...	1	230.32...	180.77...	151.66...	1	35.1248	180.77...	195.20...	1	110.95...	80.7705	75.8333	1
5	199.17...	-33.8848	213.30...	1	199.17...	166.11...	213.30...	1	-0.3541	166.11...	199.52...	1	106.29...	66.1152	106.65...	1
6	192.93...	35.6936	184.46...	1	192.93...	235.69...	184.46...	1	-6.5046	235.69...	199.43...	1	85.7296	135.69...	92.2342	1
7	201.89...	-32.3244	179.43...	1	201.89...	167.67...	179.43...	1	2.8558	167.67...	199.03...	1	92.5756	67.6756	89.7199	1
8	178.46...	-18.6849	219.18...	1	178.46...	181.31...	219.18...	1	-20.5139	181.31...	198.97...	1	89.0762	81.3151	109.59...	1
9	200.93...	37.2575	229.73...	1	200.93...	237.25...	229.73...	1	3.5723	237.25...	197.36...	1	118.43...	137.25...	114.86...	1
10	124.49...	22.7578	240.22...	1	124.49...	222.75...	240.22...	1	-70.2695	222.75...	194.76...	1	49.8430	122.75...	120.11...	1
11	175.57...	-9.3508	194.29...	1	175.57...	190.64...	194.29...	1	-24.3476	190.64...	199.92...	1	72.8015	90.6492	97.1491	1
12	283.27...	38.0326	131.46...	1	283.27...	238.03...	131.46...	1	92.3254	238.03...	190.94...	1	158.05...	138.03...	65.7320	1
13	150.19...	14.0350	225.65...	1	150.19...	214.03...	225.65...	1	-47.8919	214.03...	198.09...	1	64.9379	114.03...	112.82...	1
14	285.80...	13.1824	135.37...	1	285.80...	213.18...	135.37...	1	93.9495	213.18...	191.85...	1	161.63...	113.18...	67.6891	1
15	212.72...	-37.1315	139.39...	1	212.72...	162.86...	139.39...	1	19.9764	162.86...	192.75...	1	89.6747	62.8685	69.6983	1
16	216.87...	21.4678	209.79...	1	216.87...	221.46...	209.79...	1	17.1243	221.46...	199.74...	1	122.02...	121.46...	104.89...	1
17	128.01...	-10.6210	256.31...	1	128.01...	189.37...	256.31...	1	-59.9456	189.37...	187.95...	1	68.2098	89.3790	128.15...	1
18	233.74...	14.3417	122.55...	1	233.74...	214.34...	122.55...	1	45.0162	214.34...	188.73...	1	106.29...	114.34...	61.2762	1
19	182.96...	3.4194	189.84...	1	182.96...	203.41...	189.84...	1	-16.7892	203.41...	199.75...	1	78.1339	103.41...	94.9231	1
20	212.18...	-2.2232	211.98...	1	212.18...	197.77...	211.98...	1	12.5639	197.77...	199.61...	1	118.55...	97.7768	105.99...	1
21																
22																

Rows 1- 4 are the first marker, 5-8 are the second, 9 -12 the third and 13 – 16 is the simulated target points.

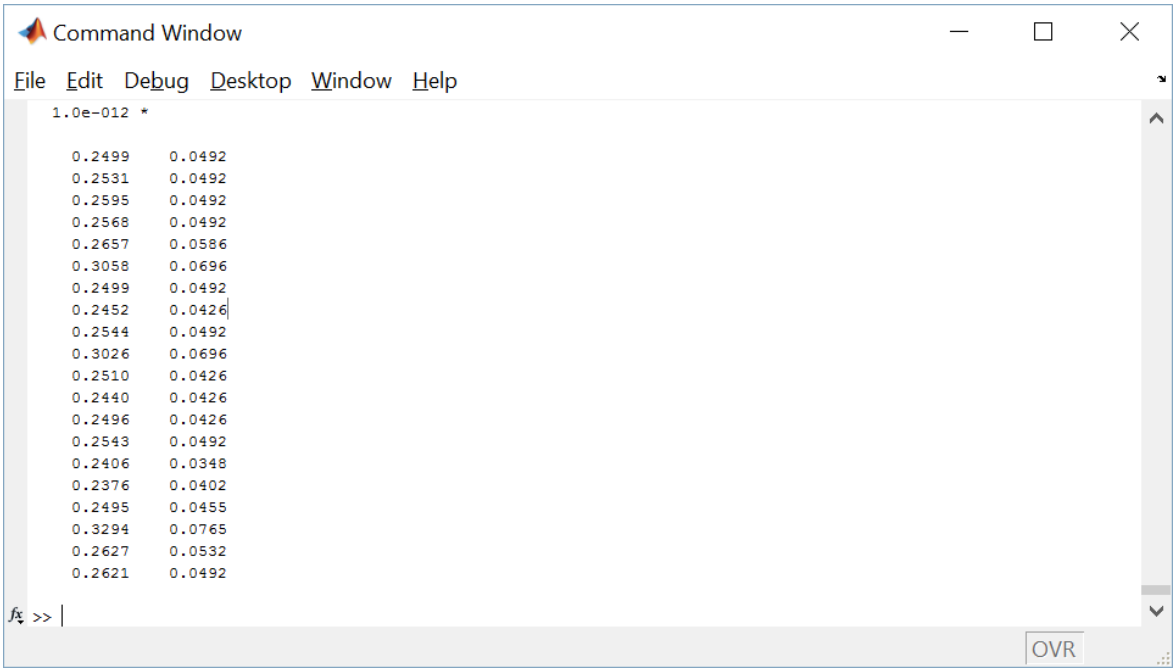
6. Registration Correctness:

To check registration correctness I looked at the FRE (fiducial registration error) and TRE (target registration error). To compute these:

- I ran the fiducial markers and target through SimulateManyPatientPoses with random translational error between 0 and 40 in either direction of both x and y, and random rotation about the y axis by 20 degrees.
- I then took these simulated markers and ran them through my registration check function
    - RegistrationCheck.m
- This would create a 4x4 transformation matrix, twice, between fiducial points and simulated points then run the fiducial points through the matrix to map them to simulated frame
- It would then check the distance between:
    - A: the sum of the distances between fiducial and simulated points for FRE

- B: the distance between simulated and transformed target for TRE

This is the output of running the simulated points through the registration check:

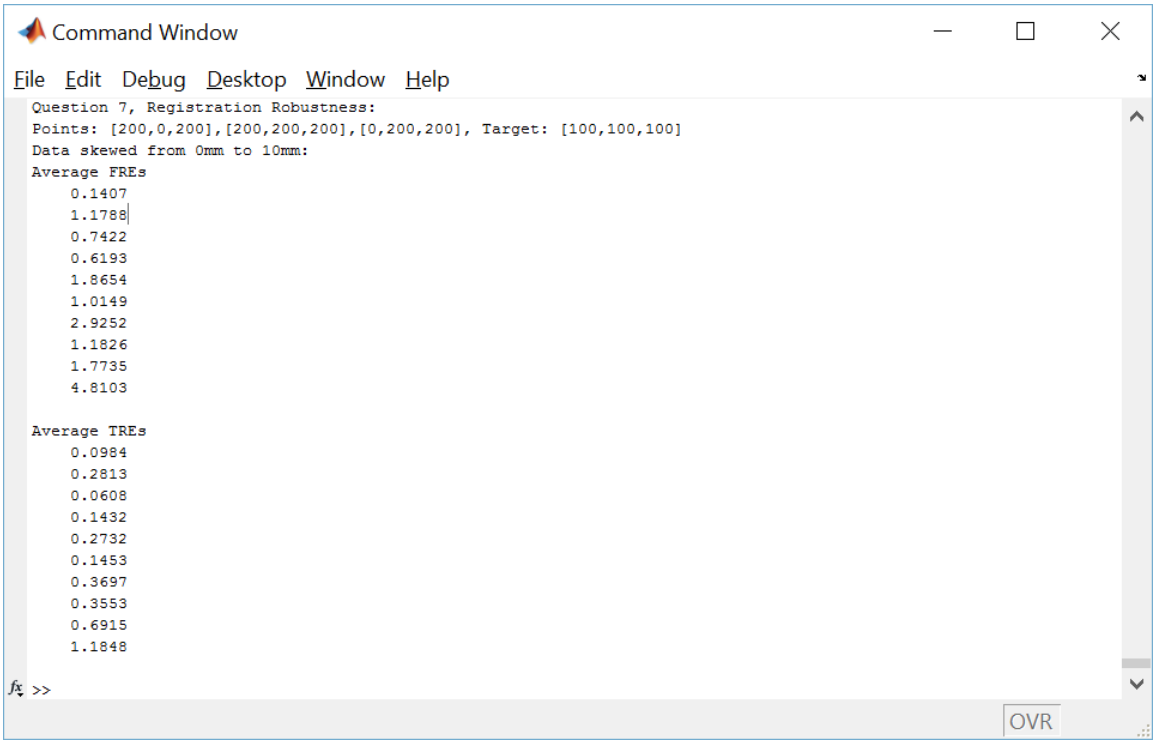


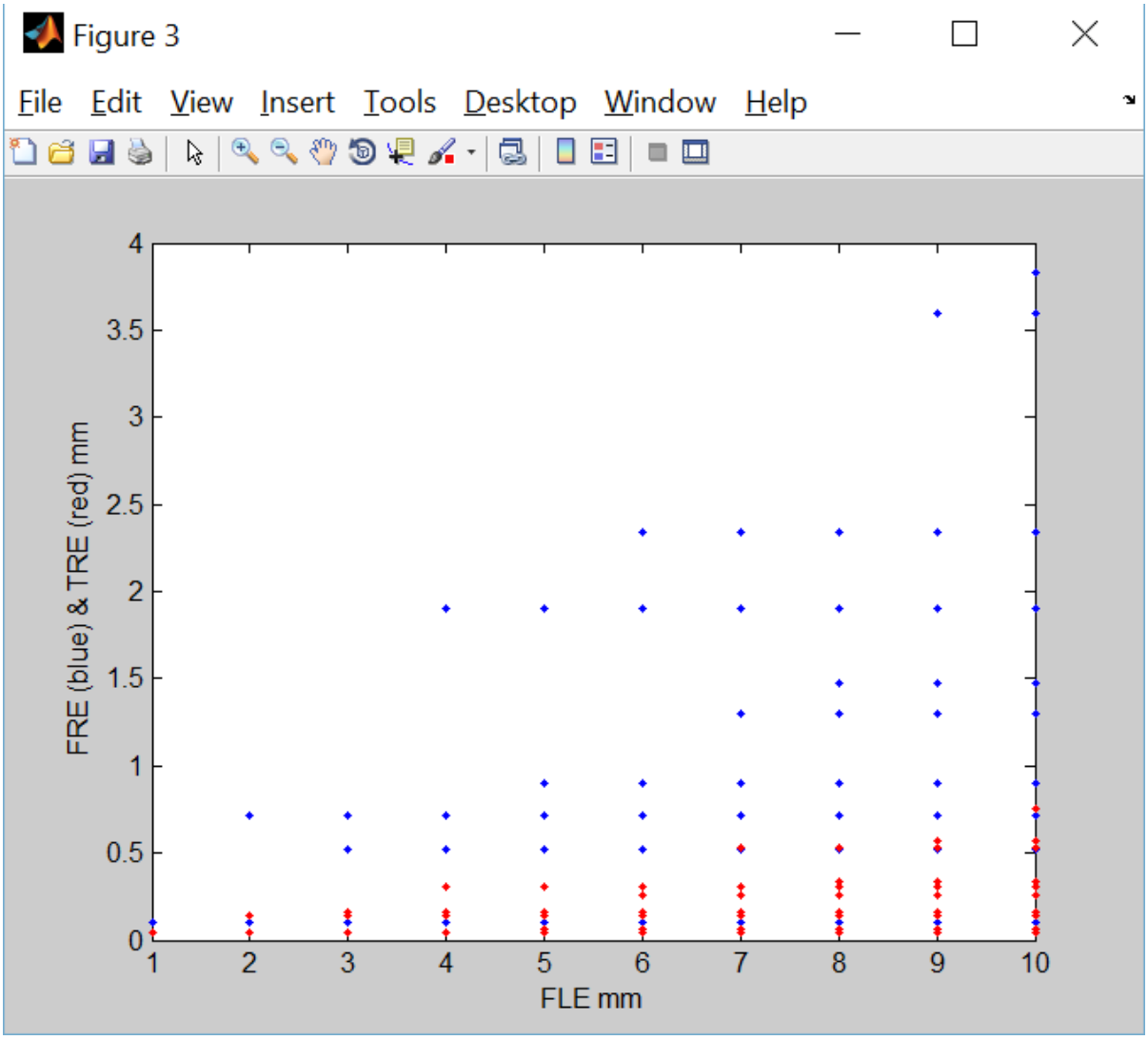
Where the left column is FRE and the right is the TRE.

7. Registration Robustness

To test if my registration is robust I added random translation error using question 3 then called the simulated points and registration check to see how the FRE and TRE were affected:

This is the output from Emax 0mm to 10mm;



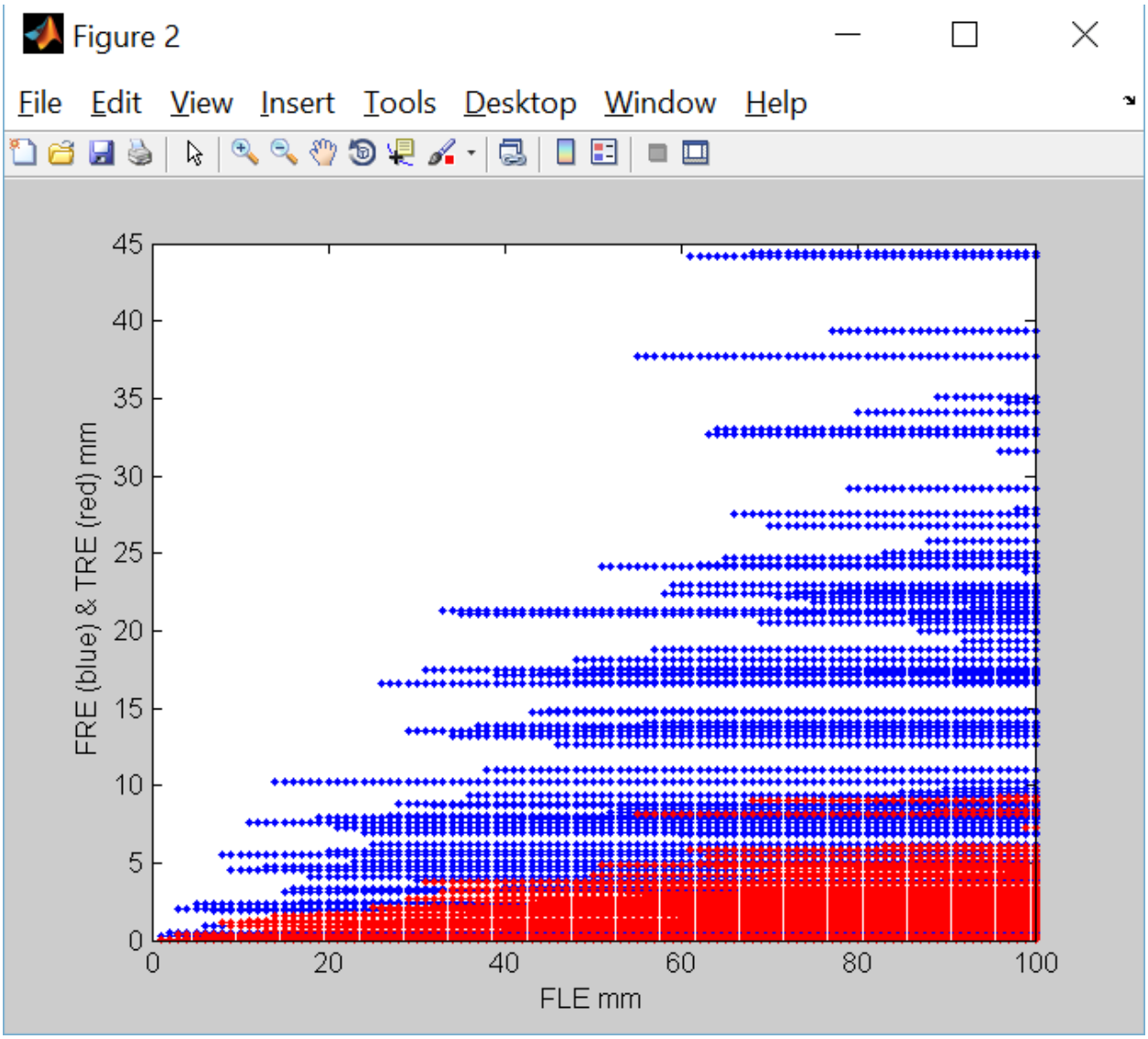


As we can see, both the fiducial and target error increased as we increased fiducial translation error which was to be expected.

FRE (blue) increased significantly greater than TRE (red), this is probably because we are directly applying error to the fiducial markers, and not the target.

We can predict that as we continue to increase the FLEmax, that the errors will continue to grow.

Both errors seem to increasing in a linear patter, however it is hard to tell what they may do to greater FLEmax.



Here is a figure of a FLEmax from 0 – 100.

We can definitely now see that FRE is increasing much more than TRE.

TRE almost looks to have leveled out around 10 mm error. This may be the max error we can achieve from only altering the fiducial points.