**Lab 4 Report**

**November 20, 2015**

Alan Downey

Allen Ewert

Eric Cuevas

MECH 4v95- Introduction To Robotics

Dr. Nicholas Gans

Fall 2015

Department of Mechanical Engineering,  
Erik Jonsson School of Engineering and Computer Science

University of Texas at Dallas

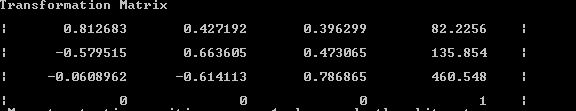


**Analysis**

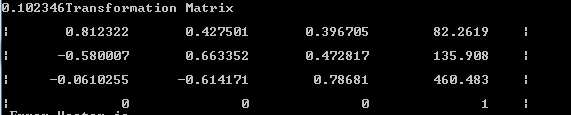
Overall the program performed as we expected, and stopped when it hit our minimum excepted error. The movements it took to get to the goal position seemed sporadic at times, and the larger the requested change became, the more random the movements became. At some points it appeared to move in the complete wrong direction before moving towards the goal position. When we pushed the positions to the limits we did find that it would lock up do to singularity.

**Run 1**

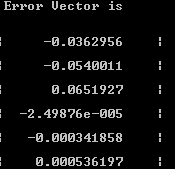
**goal**

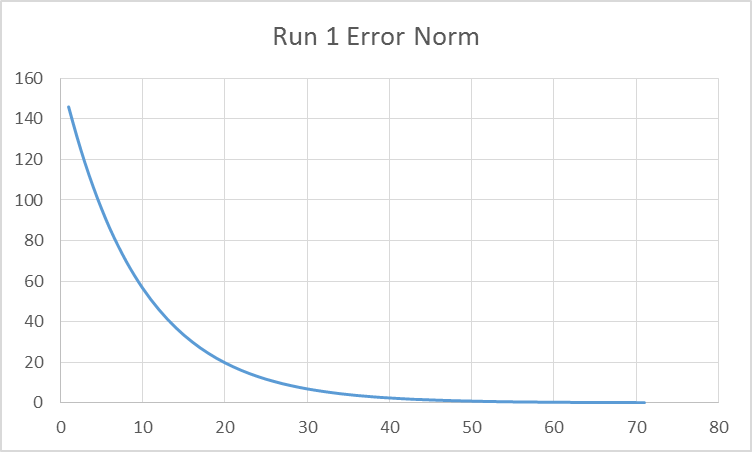


**Final**



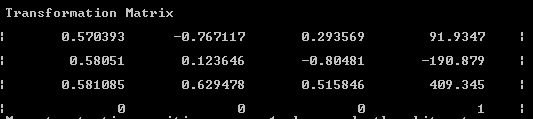
**Error**



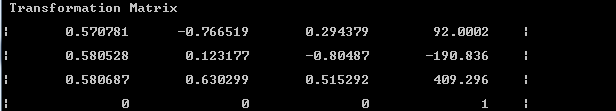
****

**Run 2**

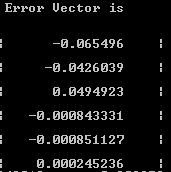
**Goal**

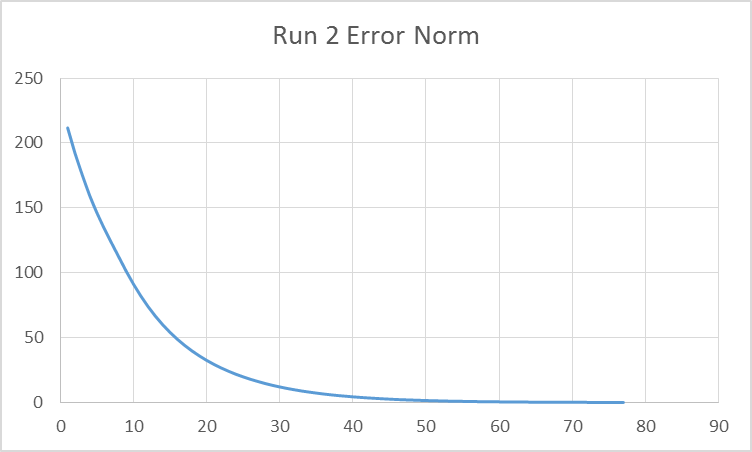


**Final**



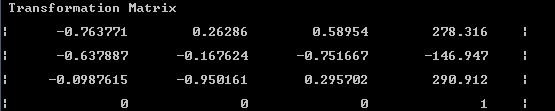
**Error**



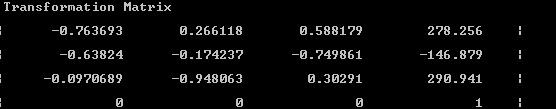
****

**Run 3**

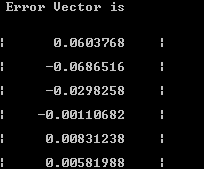
**Goal**

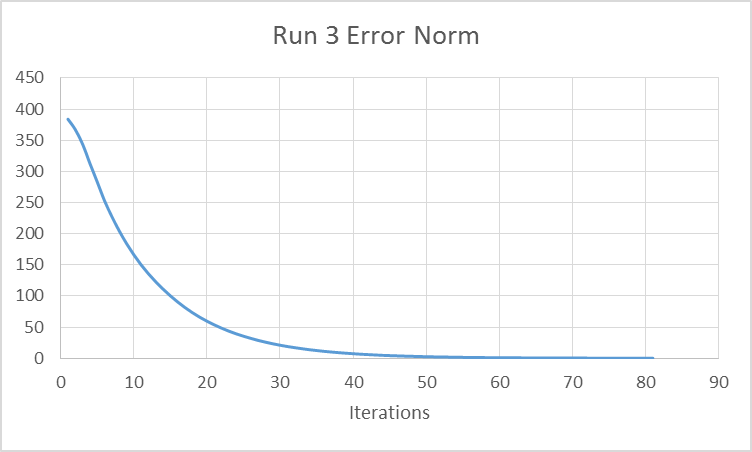


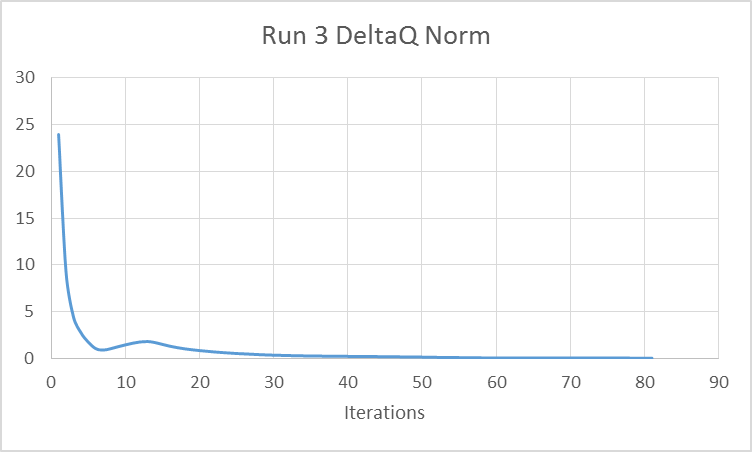
**Final**



**Error Vector**



****

****

**Relevant Code**

**Code in main.cpp:**

//call your functions here

EcReArray current;

current.setToZeros(7, 7);

EcReArray goal;

goal.setToZeros(7, 7);

EcReArray pseudo;

pseudo.setToZeros(7, 6);

EcReArray error;

error.setToZeros(6, 1);

EcReArray deltaq;

deltaq.setToZeros(7, 1);

int advance;

cout << "\n Move to Goal Position, press 1 when ready" << endl;

cin >> advance;

goal = networkComTester.lab3();

cout << "\n Move to current Position, press 1 when ready" << endl;

cin >> advance;

current = networkComTester.lab3();

cout << "\n main check 1" << endl;

double norm = 1.0;

int whilestop = 0;

while (norm > 0.1 && whilestop < 200)

{

cout << "\n beginning of while loop" << endl;

current = networkComTester.lab3();

error[0][0] = goal[7][0] - current[7][0];

error[1][0] = goal[7][1] - current[7][1];

error[2][0] = goal[7][2] - current[7][2];

error[3][0] = goal[7][3] - current[7][3];

error[4][0] = goal[7][4] - current[7][4];

error[5][0] = goal[7][5] - current[7][5];

cout << "error vector \n" << endl;

networkComTester.printMatrix(error, 6, 1);

/\*

pseudo = {

current[0][0], current[0][1], current[0][2], current[0][3], current[0][4], current[0][5], current[0][6],

current[1][0], current[1][1], current[1][2], current[1][3], current[1][4], current[1][5], current[1][6],

current[2][0], current[2][1], current[2][2], current[2][3], current[2][4], current[2][5], current[2][6],

current[3][0], current[3][1], current[3][2], current[3][3], current[3][4], current[3][5], current[3][6],

current[4][0], current[4][1], current[4][2], current[4][3], current[4][4], current[4][5], current[4][6],

current[5][0], current[5][1], current[5][2], current[5][3], current[5][4], current[5][5], current[5][6]

};

\*/

pseudo[0][0] = current[0][0];

pseudo[0][1] = current[0][1];

pseudo[0][2] = current[0][2];

pseudo[0][3] = current[0][3];

pseudo[0][4] = current[0][4];

pseudo[0][5] = current[0][5];

pseudo[1][0] = current[1][0];

pseudo[1][1] = current[1][1];

pseudo[1][2] = current[1][2];

pseudo[1][3] = current[1][3];

pseudo[1][4] = current[1][4];

pseudo[1][5] = current[1][5];

pseudo[2][0] = current[2][0];

pseudo[2][1] = current[2][1];

pseudo[2][2] = current[2][2];

pseudo[2][3] = current[2][3];

pseudo[2][4] = current[2][4];

pseudo[2][5] = current[2][5];

pseudo[3][0] = current[3][0];

pseudo[3][1] = current[3][1];

pseudo[3][2] = current[3][2];

pseudo[3][3] = current[3][3];

pseudo[3][4] = current[3][4];

pseudo[3][5] = current[3][5];

pseudo[4][0] = current[4][0];

pseudo[4][1] = current[4][1];

pseudo[4][2] = current[4][2];

pseudo[4][3] = current[4][3];

pseudo[4][4] = current[4][4];

pseudo[4][5] = current[4][5];

pseudo[5][0] = current[5][0];

pseudo[5][1] = current[5][1];

pseudo[5][2] = current[5][2];

pseudo[5][3] = current[5][3];

pseudo[5][4] = current[5][4];

pseudo[5][5] = current[5][5];

pseudo[6][0] = current[6][0];

pseudo[6][1] = current[6][1];

pseudo[6][2] = current[6][2];

pseudo[6][3] = current[6][3];

pseudo[6][4] = current[6][4];

pseudo[6][5] = current[6][5];

cout << "\n \n";

cout << "pseudo vector \n" << endl;

networkComTester.printMatrix(pseudo, 7, 6);

deltaq = pseudo\*error;

EcReArray send;

send.setToZeros(7, 1);

send[0][0] = 0.1 \* deltaq[0][0];

send[1][0] = 0.1 \* deltaq[1][0];

send[2][0] = 0.1 \* deltaq[2][0];

send[3][0] = 0.1 \* deltaq[3][0];

send[4][0] = 0.1 \* deltaq[4][0];

send[5][0] = 0.1 \* deltaq[5][0];

send[6][0] = 0.1 \* deltaq[6][0];

networkComTester.lab4(send);

norm = sqrt((error[0][0] \* error[0][0]) + (error[1][0] \* error[1][0]) + (error[2][0] \* error[2][0]) + (error[3][0] \* error[3][0]) + (error[4][0] \* error[4][0]) + (error[5][0] \* error[5][0]));

cout << "\n norm is" << endl;

cout << norm;

whilestop +=1;

}

cout << "\n end of lab. press 1 to end" << endl;

cin >> advance;

return 1;

}

**Code in ecNetworkCommunicationTest.cpp:**

**EcReArray EcNetworkCommunicationTester::lab3()**

**{**

**cout << "\n beginning of lab3" << endl;**

**float c = 3.141592 / 180;**

**EcRealVector joints;**

**joints.resize(8);**

**EcBoolean retVal = EcTrue;**

**retVal &= getJointValues(joints);**

**float a0 = 0;**

**float alpha0 = 90\*c;**

**float d0 = 118;**

**float theta0 = joints[0]+90\*c;**

**EcReArray A0;**

**A0.setToZeros(4, 4);**

**A0[0][0] = cos(theta0);**

**A0[0][1] = -sin(theta0)\*cos(alpha0);**

**A0[0][2] = sin(theta0)\*sin(alpha0);**

**A0[0][3] = a0 \*cos(theta0);**

**A0[1][0] = sin(theta0);**

**A0[1][1] = cos(theta0) \*cos(alpha0);**

**A0[1][2] = -cos(theta0) \*sin(alpha0);**

**A0[1][3] = a0 \* sin(theta0);**

**A0[2][0] = 0;**

**A0[2][1] = sin(alpha0);**

**A0[2][2] = cos(alpha0);**

**A0[2][3] = d0;**

**A0[3][0] = 0;**

**A0[3][1] = 0;**

**A0[3][2] = 0;**

**A0[3][3] = 1;**

**float a1 = 0;**

**float alpha1 = -90\*c;**

**float d1 = 0;**

**float theta1 = joints[1];**

**EcReArray A1;**

**A1.setToZeros(4, 4);**

**A1[0][0] = cos(theta1);**

**A1[0][1] = -sin(theta1)\*cos(alpha1);**

**A1[0][2] = sin(theta1)\*sin(alpha1);**

**A1[0][3] = a1 \*cos(theta1);**

**A1[1][0] = sin(theta1);**

**A1[1][1] = cos(theta1) \*cos(alpha1);**

**A1[1][2] = -cos(theta1) \*sin(alpha1);**

**A1[1][3] = a1 \* sin(theta1);**

**A1[2][0] = 0;**

**A1[2][1] = sin(alpha1);**

**A1[2][2] = cos(alpha1);**

**A1[2][3] = d1;**

**A1[3][0] = 0;**

**A1[3][1] = 0;**

**A1[3][2] = 0;**

**A1[3][3] = 1;**

**float a2 = 0;**

**float alpha2 = 90\*c;**

**float d2 = 140.8;**

**float theta2 = joints[2];**

**EcReArray A2;**

**A2.setToZeros(4, 4);**

**A2[0][0] = cos(theta2);**

**A2[0][1] = -sin(theta2)\*cos(alpha2);**

**A2[0][2] = sin(theta2)\*sin(alpha2);**

**A2[0][3] = a2 \*cos(theta2);**

**A2[1][0] = sin(theta2);**

**A2[1][1] = cos(theta2) \*cos(alpha2);**

**A2[1][2] = -cos(theta2) \*sin(alpha2);**

**A2[1][3] = a2 \* sin(theta2);**

**A2[2][0] = 0;**

**A2[2][1] = sin(alpha2);**

**A2[2][2] = cos(alpha2);**

**A2[2][3] = d2;**

**A2[3][0] = 0;**

**A2[3][1] = 0;**

**A2[3][2] = 0;**

**A2[3][3] = 1;**

**float a3 = 71.4;**

**float alpha3 = -90\*c;**

**float d3 = 0;**

**float theta3 = joints[3]+90\*c;**

**EcReArray A3;**

**A3.setToZeros(4, 4);**

**A3[0][0] = cos(theta3);**

**A3[0][1] = -sin(theta3)\*cos(alpha3);**

**A3[0][2] = sin(theta3)\*sin(alpha3);**

**A3[0][3] = a3 \*cos(theta3);**

**A3[1][0] = sin(theta3);**

**A3[1][1] = cos(theta3) \*cos(alpha3);**

**A3[1][2] = -cos(theta3) \*sin(alpha3);**

**A3[1][3] = a3 \* sin(theta3);**

**A3[2][0] = 0;**

**A3[2][1] = sin(alpha3);**

**A3[2][2] = cos(alpha3);**

**A3[2][3] = d3;**

**A3[3][0] = 0;**

**A3[3][1] = 0;**

**A3[3][2] = 0;**

**A3[3][3] = 1;**

**float a4 = 71.4;**

**float alpha4 = 90\*c;**

**float d4 = 0;**

**float theta4 = joints[4];**

**EcReArray A4;**

**A4.setToZeros(4, 4);**

**A4[0][0] = cos(theta4);**

**A4[0][1] = -sin(theta4)\*cos(alpha4);**

**A4[0][2] = sin(theta4)\*sin(alpha4);**

**A4[0][3] = a4 \*cos(theta4);**

**A4[1][0] = sin(theta4);**

**A4[1][1] = cos(theta4) \*cos(alpha4);**

**A4[1][2] = -cos(theta4) \*sin(alpha4);**

**A4[1][3] = a4 \* sin(theta4);**

**A4[2][0] = 0;**

**A4[2][1] = sin(alpha4);**

**A4[2][2] = cos(alpha4);**

**A4[2][3] = d4;**

**A4[3][0] = 0;**

**A4[3][1] = 0;**

**A4[3][2] = 0;**

**A4[3][3] = 1;**

**float a5 = 0;**

**float alpha5 = -90\*c;**

**float d5 = 0;**

**float theta5 = joints[5]-90\*c;**

**EcReArray A5;**

**A5.setToZeros(4, 4);**

**A5[0][0] = cos(theta5);**

**A5[0][1] = -sin(theta5)\*cos(alpha5);**

**A5[0][2] = sin(theta5)\*sin(alpha5);**

**A5[0][3] = a5 \*cos(theta5);**

**A5[1][0] = sin(theta5);**

**A5[1][1] = cos(theta5) \*cos(alpha5);**

**A5[1][2] = -cos(theta5) \*sin(alpha5);**

**A5[1][3] = a5 \* sin(theta5);**

**A5[2][0] = 0;**

**A5[2][1] = sin(alpha5);**

**A5[2][2] = cos(alpha5);**

**A5[2][3] = d5;**

**A5[3][0] = 0;**

**A5[3][1] = 0;**

**A5[3][2] = 0;**

**A5[3][3] = 1;**

**float a6 = 0;**

**float alpha6 = 0;**

**float d6 = 100;**

**float theta6 = joints[6]-90\*c;**

**EcReArray A6;**

**A6.setToZeros(4, 4);**

**A6[0][0] = cos(theta6);**

**A6[0][1] = -sin(theta6)\*cos(alpha6);**

**A6[0][2] = sin(theta6)\*sin(alpha6);**

**A6[0][3] = a6 \*cos(theta6);**

**A6[1][0] = sin(theta6);**

**A6[1][1] = cos(theta6) \*cos(alpha6);**

**A6[1][2] = -cos(theta6) \*sin(alpha6);**

**A6[1][3] = a6 \* sin(theta6);**

**A6[2][0] = 0;**

**A6[2][1] = sin(alpha6);**

**A6[2][2] = cos(alpha6);**

**A6[2][3] = d6;**

**A6[3][0] = 0;**

**A6[3][1] = 0;**

**A6[3][2] = 0;**

**A6[3][3] = 1;**

**EcReArray final;**

**final.setToZeros(4, 4);**

**final = A0\*A1\*A2\*A3\*A4\*A5\*A6;**

**cout << "\n lab3 check 1" << endl;**

**EcReArray T10;**

**EcReArray T20;**

**EcReArray T30;**

**EcReArray T40;**

**EcReArray T50;**

**EcReArray T60;**

**EcReArray T70;**

**T10.setToZeros(4, 4);**

**T20.setToZeros(4, 4);**

**T30.setToZeros(4, 4);**

**T40.setToZeros(4, 4);**

**T50.setToZeros(4, 4);**

**T60.setToZeros(4, 4);**

**T70.setToZeros(4, 4);**

**T10 = A0;**

**T20 = A0\*A1;**

**T30 = A0\*A1\*A2;**

**T40 = A0\*A1\*A2\*A3;**

**T50 = A0\*A1\*A2\*A3\*A4;**

**T60 = A0\*A1\*A2\*A3\*A4\*A5;**

**T70 = A0\*A1\*A2\*A3\*A4\*A5\*A6;**

**EcVector Z0,Z1,Z2,Z3,Z4,Z5,Z6,Z7;**

**Z0.setToZero(); Z1.setToZero(); Z2.setToZero(); Z3.setToZero(); Z4.setToZero(); Z5.setToZero(); Z6.setToZero(); Z7.setToZero();**

**EcVector O0, O1, O2, O3, O4, O5, O6, O7;**

**O0.setToZero(); O1.setToZero(); O2.setToZero(); O3.setToZero(); O4.setToZero(); O5.setToZero(); O6.setToZero(); O7.setToZero();**

**Z0 = { 0, 0, 1 };**

**Z1 = { T10[0][2], T10[1][2], T10[2][2] };**

**cout << "Z0: " << Z0 << endl;**

**Z2 = { T20[0][2], T20[1][2], T20[2][2] };**

**cout << "\nZ1: " << endl;**

**Z3 = { T30[0][2], T30[1][2], T30[2][2] };**

**Z4 = { T40[0][2], T40[1][2], T40[2][2] };**

**Z5 = { T50[0][2], T50[1][2], T50[2][2] };**

**Z6 = { T60[0][2], T60[1][2], T60[2][2] };**

**Z7 = { T70[0][2], T70[1][2], T70[2][2] };**

**O0 = { 0,0,0 };**

**O1 = { T10[0][3], T10[1][3], T10[2][3] };**

**O2 = { T20[0][3], T20[1][3], T20[2][3] };**

**O3 = { T30[0][3], T30[1][3], T30[2][3] };**

**O4 = { T40[0][3], T40[1][3], T40[2][3] };**

**O5 = { T50[0][3], T50[1][3], T50[2][3] };**

**O6 = { T60[0][3], T60[1][3], T60[2][3] };**

**O7 = { T70[0][3], T70[1][3], T70[2][3] };**

**EcVector C0, C1, C2, C3, C4, C5, C6;**

**C0.setToZero(); C1.setToZero(); C2.setToZero(); C3.setToZero(); C4.setToZero(); C5.setToZero(); C6.setToZero();**

**EcVector S0, S1, S2, S3, S4, S5, S6;**

**S0.setToZero(); S1.setToZero(); S2.setToZero(); S3.setToZero(); S4.setToZero(); S5.setToZero(); S6.setToZero();**

**S0 = O7-O0;**

**S1 = { O7[0] - O1[0], O7[1] - O1[1], O7[2] - O1[2] };**

**S2 = { O7[0] - O2[0], O7[1] - O2[1], O7[2] - O2[2] };**

**S3 = { O7[0] - O3[0], O7[1] - O3[1], O7[2] - O3[2] };**

**S4 = { O7[0] - O4[0], O7[1] - O4[1], O7[2] - O4[2] };**

**S5 = { O7[0] - O5[0], O7[1] - O5[1], O7[2] - O5[2] };**

**S6 = { O7[0] - O6[0], O7[1] - O6[1], O7[2] - O6[2] };**

**C0 = Z0.cross(S0);**

**C1 = Z1.cross(S1);**

**C2 = Z2.cross(S2);**

**C3 = Z3.cross(S3);**

**C4 = Z4.cross(S4);**

**C5 = Z5.cross(S5);**

**C6 = Z6.cross(S6);**

**cout << "\n o0 is:" << O0 << endl;**

**cout << "\n o6 is:" << O6 << endl;**

**cout << "\n C0 is:" << C0 << endl;**

**cout << "\n S0 is:"<< S0 << endl;**

**cout << "\n Z0 is:"<< Z0 << endl;**

**/\*Jacob = {**

**C0[0], C1[0], C2[0], C3[0], C4[0], C5[0], C6[0] ,**

**C0[1], C1[1], C2[1], C3[1], C4[1], C5[1], C6[1] ,**

**C0[2], C1[2], C2[2], C3[2], C4[2], C5[2], C6[2] ,**

**Z0[0], Z1[0], Z2[0], Z3[0], Z4[0], Z5[0], Z6[0] ,**

**Z0[1], Z1[1], Z2[1], Z3[1], Z4[1], Z5[1], Z6[1] ,**

**Z0[2], Z1[2], Z2[2], Z3[2], Z4[2], Z5[2], Z6[2]**

**};**

**\*/**

**EcReArray Jacob;**

**Jacob.setToZeros(6, 7);**

**Jacob[0][0] = C0[0];**

**Jacob[0][1] = C1[0];**

**Jacob[0][2] = C2[0];**

**Jacob[0][3] = C3[0];**

**Jacob[0][4] = C4[0];**

**Jacob[0][5] = C5[0];**

**Jacob[0][6] = C6[0];**

**Jacob[1][0] = C0[1];**

**Jacob[1][1] = C1[1];**

**Jacob[1][2] = C2[1];**

**Jacob[1][3] = C3[1];**

**Jacob[1][4] = C4[1];**

**Jacob[1][5] = C5[1];**

**Jacob[1][6] = C6[1];**

**Jacob[2][0] = C0[2];**

**Jacob[2][1] = C1[2];**

**Jacob[2][2] = C2[2];**

**Jacob[2][3] = C3[2];**

**Jacob[2][4] = C4[2];**

**Jacob[2][5] = C5[2];**

**Jacob[2][6] = C6[2];**

**Jacob[3][0] = Z0[0];**

**Jacob[3][1] = Z1[0];**

**Jacob[3][2] = Z2[0];**

**Jacob[3][3] = Z3[0];**

**Jacob[3][4] = Z4[0];**

**Jacob[3][5] = Z5[0];**

**Jacob[3][6] = Z6[0];**

**Jacob[4][0] = Z0[1];**

**Jacob[4][1] = Z1[1];**

**Jacob[4][2] = Z2[1];**

**Jacob[4][3] = Z3[1];**

**Jacob[4][4] = Z4[1];**

**Jacob[4][5] = Z5[1];**

**Jacob[4][6] = Z6[1];**

**Jacob[5][0] = Z0[2];**

**Jacob[5][1] = Z1[2];**

**Jacob[5][2] = Z2[2];**

**Jacob[5][3] = Z3[2];**

**Jacob[5][4] = Z4[2];**

**Jacob[5][5] = Z5[2];**

**Jacob[5][6] = Z6[2];**

**cout << "\n Jacob is " << endl;**

**printMatrix(Jacob, 6, 7);**

**EcReArray JT;**

**JT.setToZeros(7, 6);**

**JT = Jacob.transpose();**

**cout << "\n JT is \n" << endl;**

**printMatrix(JT, 7, 6);**

**EcReArray JJT;**

**JJT.setToZeros(6, 6);**

**JJT = Jacob\*JT;**

**cout << "\n JJT is \n" << endl;**

**printMatrix(JJT, 6, 6);**

**EcReArray JJTI;**

**JJTI.setToZeros(6, 6);**

**JJTI = JJT;**

**JJTI.invertSquareMatrix();**

**cout << "\n JJTI is \n" << endl;**

**printMatrix(JJTI, 6, 6);**

**EcReArray JP;**

**JP.setToZeros(7, 6);**

**JP = JT\*JJTI;**

**cout << "\n JP is \n" << endl;**

**printMatrix(JP, 7, 6);**

**cout << "\n lab3 check 2" << endl;**

**float theta;**

**float place;**

**place = (final[0][0] + final[1][1] + final[2][2] - 1.0) / 2.0;**

**theta = acos(place);**

**if (theta == 0){**

**theta = 0.00000001; // Eliminates divide by zero error for unitK**

**}**

**EcRealVector unitk;**

**unitk.resize(3);**

**unitk[0] = 1 / (2 \* sin(theta))\*(final[2][1] - final[1][2]);**

**unitk[1] = 1 / (2\*sin(theta))\*(final[0][2] - final[2][0]);**

**unitk[2] = 1 / (2\*sin(theta))\*(final[1][0] - final[0][1]);**

**EcRealVector position;**

**position.resize(6);**

**position[0] = final[0][3];**

**position[1] = final[1][3];**

**position[2] = final[2][3];**

**position[3] = unitk[0] \* theta;**

**position[4] = unitk[1] \* theta;**

**position[5] = unitk[2] \* theta;**

**/\***

**cout << "\n JP is " << endl;**

**printMatrix(JP, 7, 6);**

**cout << "\n JP00 is " << endl;**

**cout << JP[0][0];**

**cout << "\n JP11 is " << endl;**

**cout << JP[1][1];**

**cout << "\n JP44 is " << endl;**

**cout << JP[4][4];**

**cout << "\n JP55 is " << endl;**

**cout << JP[5][5];**

**cout << "\n unitk is " << endl;**

**printVector(unitk, 3);**

**cout << "\n theta is " << endl;**

**cout << theta;**

**\*/**

**EcReArray lab3ret;**

**lab3ret.setToZeros(8, 6);**

**cout << "\n lab3 check 3" << endl;**

**/\***

**lab3ret = {**

**JP[0][0], JP[0][1], JP[0][2], JP[0][3], JP[0][4], JP[0][5], JP[0][6],**

**JP[1][0], JP[1][1], JP[1][2], JP[1][3], JP[1][4], JP[1][5], JP[1][6],**

**JP[2][0], JP[2][1], JP[2][2], JP[2][3], JP[2][4], JP[2][5], JP[2][6],**

**JP[3][0], JP[3][1], JP[3][2], JP[3][3], JP[3][4], JP[3][5], JP[3][6],**

**JP[4][0], JP[4][1], JP[4][2], JP[4][3], JP[4][4], JP[4][5], JP[4][6],**

**JP[5][0], JP[5][1], JP[5][2], JP[5][3], JP[5][4], JP[5][5], JP[5][6],**

**position[0], position[1], position[2], position[3], position[4], position[5], 0**

**};**

**\*/**

**lab3ret[0][0] = JP[0][0];**

**lab3ret[0][1] = JP[0][1];**

**lab3ret[0][2] = JP[0][2];**

**lab3ret[0][3] = JP[0][3];**

**lab3ret[0][4] = JP[0][4];**

**lab3ret[0][5] = JP[0][5];**

**lab3ret[1][0] = JP[1][0];**

**lab3ret[1][1] = JP[1][1];**

**lab3ret[1][2] = JP[1][2];**

**lab3ret[1][3] = JP[1][3];**

**lab3ret[1][4] = JP[1][4];**

**lab3ret[1][5] = JP[1][5];**

**lab3ret[2][0] = JP[2][0];**

**lab3ret[2][1] = JP[2][1];**

**lab3ret[2][2] = JP[2][2];**

**lab3ret[2][3] = JP[2][3];**

**lab3ret[2][4] = JP[2][4];**

**lab3ret[2][5] = JP[2][5];**

**lab3ret[3][0] = JP[3][0];**

**lab3ret[3][1] = JP[3][1];**

**lab3ret[3][2] = JP[3][2];**

**lab3ret[3][3] = JP[3][3];**

**lab3ret[3][4] = JP[3][4];**

**lab3ret[3][5] = JP[3][5];**

**lab3ret[4][0] = JP[4][0];**

**lab3ret[4][1] = JP[4][1];**

**lab3ret[4][2] = JP[4][2];**

**lab3ret[4][3] = JP[4][3];**

**lab3ret[4][4] = JP[4][4];**

**lab3ret[4][5] = JP[4][5];**

**lab3ret[5][0] = JP[5][0];**

**lab3ret[5][1] = JP[5][1];**

**lab3ret[5][2] = JP[5][2];**

**lab3ret[5][3] = JP[5][3];**

**lab3ret[5][4] = JP[5][4];**

**lab3ret[5][5] = JP[5][5];**

**lab3ret[6][0] = JP[6][0];**

**lab3ret[6][1] = JP[6][1];**

**lab3ret[6][2] = JP[6][2];**

**lab3ret[6][3] = JP[6][3];**

**lab3ret[6][4] = JP[6][4];**

**lab3ret[6][5] = JP[6][5];**

**lab3ret[7][0] = position[0];**

**lab3ret[7][1] = position[1];**

**lab3ret[7][2] = position[2];**

**lab3ret[7][3] = position[3];**

**lab3ret[7][4] = position[4];**

**lab3ret[7][5] = position[5];**

**//cout << "\n lab3 check 4" << endl;**

**//cout << "\n theta is";**

**//cout << theta;**

**//cout << "\n place is" << endl;**

**//cout << place;**

**return lab3ret;**

**}**

**EcBoolean EcNetworkCommunicationTester::lab4(EcReArray lab4ref)**

**{**

**//cout << "\n lab4ref check \n" << endl;**

**//printMatrix(lab4ref, 7, 1);**

**EcRealVector joints;**

**joints.resize(8);**

**EcBoolean retVal = EcTrue;**

**//cout << "\n moving to new position in lab4" << endl;**

**retVal &= getJointValues(joints);**

**printVector(joints, 8);**

**//printVectorFile(joints, 8, "tempout.txt");**

**//cout << "\n getjointvalues okay" << endl;**

**joints[0] += lab4ref[0][0];**

**joints[1] += lab4ref[1][0];**

**joints[2] += lab4ref[2][0];**

**joints[3] += lab4ref[3][0];**

**joints[4] += lab4ref[4][0];**

**joints[5] += lab4ref[5][0];**

**joints[6] += lab4ref[6][0];**

**joints[7] = 0.01; //gripper always slightly open**

**//cout << "\n joints okay \n" << endl;**

**retVal &= setJointValues(joints); //Send the values to the robot**

**EcSLEEPMS(500);**

**retVal &= getJointValues(joints);**

**printVector(joints, 8);**

**int wait = 0;**

**//cin >> wait;**

**return EcTrue;**

**}**