دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی برق

Kasra Khalafi: 9531306

Final Exam

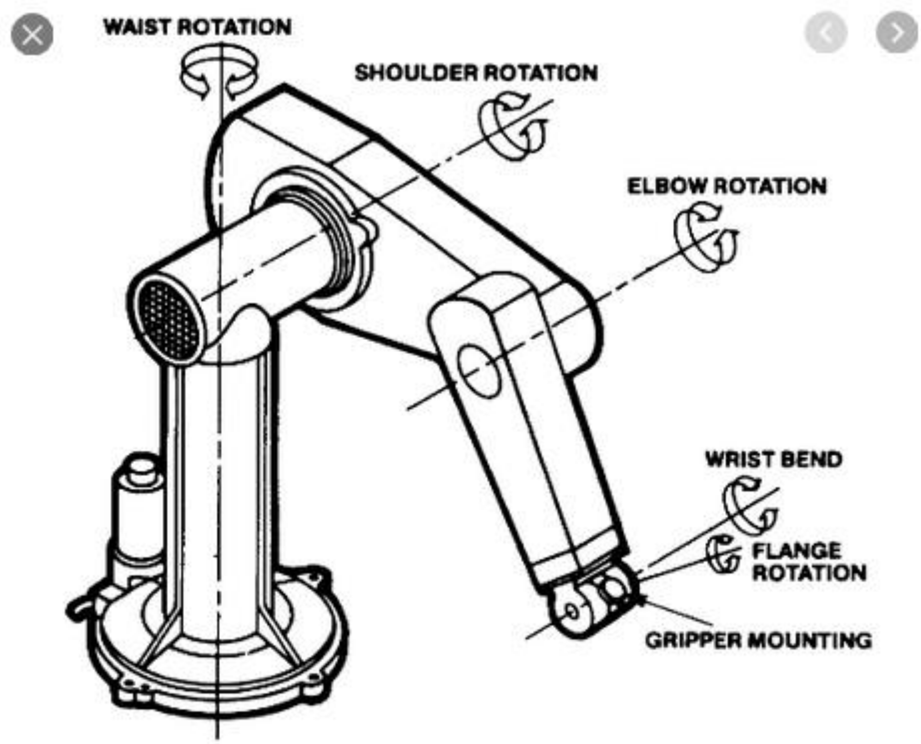
Advanced Robotic

Dr.Iman Sharifi

Maple:

در بخش پیاده سازی عملی در قسمت <خ> سوال، به پیاده سازی یک ربات 3DOF، به بخش ابتدایی ربات PUMA 560 پرداخته شده است.

ربات به صورت زیر می باشد:



همانطور که مشاهده می شود سه joint ابتدایی ربات به صورت revolute بوده در نتیجه ربات پیاده سازی شده به صورت RRR خواهد بود.

پس از محاسبات ربات، ماتریس تبدیل سه جوینت، به صورت زیر خواهد بود:

```
A1 := Matrix([[cos(t1), -sin(t1), 0, 0], [sin(t1), cos(t1), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]) :
A2 := Matrix([[cos(theta[2]), -sin(theta[2]), 0, 0], [0, 0, 1, 0], [-sin(theta[2]),
    -cos(theta[2]), 0, 0], [0, 0, 0, 1]]) :
A3 := Matrix([[cos(theta[3]), -sin(theta[3]), 0, a2], [sin(theta[3]), cos(theta[3]), 0, 0],
    [0, 0, 1, d3], [0, 0, 0, 1]]) :
A1
```

$$\begin{bmatrix} \cos(t1) & -\sin(t1) & 0 & 0 \\ \sin(t1) & \cos(t1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

برای محاسبه ی ماتریس های تبدیل نسبت به مرکز نیز به صورت زیر در کد پیاده سازی گشت:

$$T[1] := A1 :$$
$$T[2] := simplify(A1.A2) :$$
$$T[3] := simplify(T[2].A3) :$$

ماتریس تبدیل از مرکز جرم ربات نسبت به مبدا مختصات نیز بدین صورت پیاده سازی گشت:

$$TC[1] := simplify(T[1].Matrix([[1, 0, 0, xc1], [0, 1, 0, yc1], [0, 0, 1, zc1], [0, 0, 0, 1]])) :$$
$$TC[2] := simplify(T[2].Matrix([[1, 0, 0, xc2], [0, 1, 0, yc2], [0, 0, 1, zc2], [0, 0, 0, 1]])) :$$
$$TC[3] := simplify(T[3].Matrix([[1, 0, 0, xc3], [0, 1, 0, yc3], [0, 0, 1, zc3], [0, 0, 0, 1]])) :$$

سپس در قسمت بعدی کد به جدا سازی R، Z و O پرداخته می شود.

```
for i from 1 to 3 do
  o[i] := T[i](1..3, 4) :
  oc[i] := TC[i](1..3, 4) :
  z[i] := T[i](1..3, 3) :
  R[i] := T[i](1..3, 1..3) :

  end do:
```

با توجه به اینکه همه ی مفاصل Revolute هستند در نتیجه از فرمول ضرب خارجی برای به دست اوردن ژاکوبین خطی و از Z برای به دست اوردن ژاکوبین دورانی استفاده می شود. ماتریس اینرسی نیز با توجه به اینکه شکل متقارن می باشد در نتیجه ماتریس diagonal خواهد بود:

$$JVC[1] := \langle CrossProduct(z[1], (oc[1] - o[1])) \mid \langle 0, 0, 0 \rangle \mid \langle 0, 0, 0 \rangle \rangle :$$
$$JWC[1] := \langle z[1] \mid \langle 0, 0, 0 \rangle \mid \langle 0, 0, 0 \rangle \rangle :$$
$$JVC[2] := \langle CrossProduct(z[1], (oc[2] - o[1])) \mid CrossProduct(z[2], (oc[2] - o[2])) \mid \langle 0, 0, 0 \rangle \rangle :$$
$$JWC[2] := \langle z[1] \mid z[2] \mid \langle 0, 0, 0 \rangle \rangle :$$
$$JVC[3] := \langle CrossProduct(z[1], (oc[3] - o[1])) \mid CrossProduct(z[2], (oc[3] - o[2])) \mid CrossProduct(z[3], (oc[3] - o[3])) \rangle :$$
$$JWC[3] := \langle z[1] \mid z[2] \mid z[3] \rangle :$$
$$II[1] := Matrix([[Ixx1, 0, 0,], [0, Iyy1, 0], [0, 0, Izz1]]) :$$
$$II[2] := Matrix([[Ixx2, 0, 0,], [0, Iyy2, 0], [0, 0, Izz2]]) :$$
$$II[3] := Matrix([[Ixx3, 0, 0,], [0, Iyy3, 0], [0, 0, Izz3]]) :$$

در قسمت بعد جرم های دلخواه با نام m برای جرم بازو ها در نظر میگیریم. سپس به کمک فرمول زیر که حاصل از مجموع ضرب انرژی جنبشی و دورانی می باشد M یا همان D به دست اورده شد:

for j **from** 1 **to** 3 **do**
$Tvcvc := Transpose(JVC[j]).JVC[j]$:
$part1 := m[j] \cdot Tvcvc$:
$part2 := Transpose(JWC[j]).R[j].II[j].Transpose(R[j]).JWC[j]$:
$M := M + part1 + part2$:

end do:
$M := simplify(M)$:

در مرحله ی بعد شروع به دست اوردن مقادیر نمادهای کریستوفل می پردازیم:

for i **from** 1 **to** 3 **do**
for j **from** 1 **to** 3 **do**
for k **from** 1 **to** 3 **do**
$qi := \mathrm{theta}[i]$:
$qj := \mathrm{theta}[j]$:
$qk := \mathrm{theta}[k]$:
$dkj := M[k, j]$:
$dki := M[k, i]$:
$dij := M[i, j]$:
$aa := \frac{\partial}{\partial q2} dkj$:
$c[i, j, k] := .5 \cdot \left(\frac{\partial}{\partial qi} dkj + \frac{\partial}{\partial qj} dki - \frac{\partial}{\partial qk} dij \right)$:
$c[i, j, k] := simplify(\ c[i, j, k])$:

end do:
end do:
end do:

سپس ماتریس g را با مشق گیری از انرژی پتانسیل به دست می اوریم:

$g := Vector([0, 0, -9.81]):$
$P := 0:$

for q **from** 1 **to** 6 **do**
$P := P + Transpose(g).oc[q] \cdot m[q]:$

end do:

for i **from** 1 **to** 3 **do**
$qi := \text{theta}[i]:$

$$\text{G}[i] := \frac{\partial}{\partial qi} P:$$

end do:

سپس مقادیر تبدیل EndEffector و oee را به دست اورده و ماتریس ژاکوبین کلی را به دست می اوریم.

$Aee := Matrix([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, d4], [0, 0, 0, 1]]):$
$Tee := T[3] \cdot Aee:$
$oee := Tee(1..3, 4):$

$> Jv := \langle CrossProduct(z[1], (oee - o[1])) \mid CrossProduct(z[2], (oee - o[2])) \mid CrossProduct(z[3], (oee - o[3])) \rangle:$
$Jw := \langle z[1] \mid z[2] \mid z[3] \rangle:$

$J := simplify(\langle Jv, Jw \rangle)$

$$J := \begin{bmatrix} (-d4 - d3)\cos(t1) - \sin(t1)\cos(t2)\,a2 & -\cos(t1)\sin(t2)\,a2 & 0 \\ (-d4 - d3)\sin(t1) + \cos(t1)\cos(t2)\,a2 & -\sin(t1)\sin(t2)\,a2 & 0 \\ 0 & -a2\cos(t2) & 0 \\ 0 & -\sin(t1) & -\sin(t1) \\ 0 & \cos(t1) & \cos(t1) \\ 1 & 0 & 0 \end{bmatrix}$$

مقادیر بالا که به دست امده اند در دستگاه مختصات زاویه ای هستند و در صورت سوال از ما خواسته شده که مقادیر را در دستگاه مختصات کارتزین به دست بیاوریم. برای تبدیل از دستگاه مختصات زاویه ای به قطبی از ترانهاده و معکوس ژاکوبین استفاده می شود. در نتیجه از ژاکوبین به دست امده ژاکوبین گرفته شد و در dJ ذخیره گشت.

سپس به کمک کد زیر، از زاویه ای به کارتزین تبدیل های لازم صورت گرفت:

$J\_inv := MatrixInverse(J) :$
$J\_inv\_tra := Transpose(J\_inv) :$
$Jv\_inv := MatrixInverse(Jv, method = pseudo) :$
$Jv\_inv\_tra := Transpose(Jv\_inv) :$
$dJv := dJ(1..3, 1..3) :$
$dJw := dJ(4..6, 1..3) :$

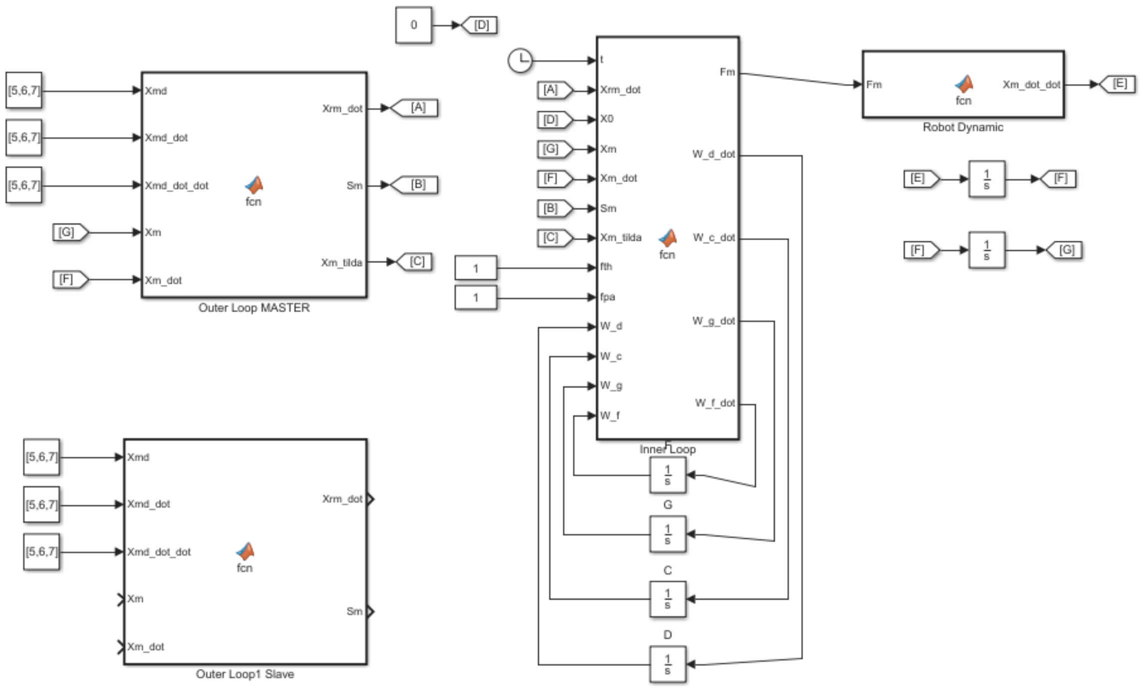
$Mx := Jv\_inv\_tra \cdot M \cdot Jv\_inv :$
$Cx := -Mx \cdot dJv \cdot Jv\_inv + Jv\_inv\_tra \cdot C \cdot Jv\_inv :$
$Gx := Jv\_inv\_tra \cdot G :$

$simplify(Jv)$

$$\begin{bmatrix} (-d4 - d3)\cos(t1) - \sin(t1)\cos(t2)\,a2 & -\cos(t1)\sin(t2)\,a2 & 0 \\ (-d4 - d3)\sin(t1) + \cos(t1)\cos(t2)\,a2 & -\sin(t1)\sin(t2)\,a2 & 0 \\ 0 & -a2\cos(t2) & 0 \end{bmatrix}$$

MATLAB

شکل کلی بلوک دیاگرام به صورت زیر است:



قسمت سمت چپ حلقه ی خارجی ما حساب می شود که X_tilda و سایر نوشته ها در حال حساب شدن می باشد.

حلقه ی وسط حلقه ی داخلی می باشد که به محاسبه ی Tau یا در این سوال Fm پرداخته می شود.

حلقه ی داخلی در صورت کلی باید 10 برابر سریع تر از حلقه ی خارجی باشد.

سمت راستی نیز دینامیک ربات می باشد.

کد قسمت حلقه ی خارجی به صورت زیر می باشد:

Outer Loop MASTER

```
function [Xrm_dot, Sm, Xm_tilda] = fcn(Xmd, Xmd_dot, Xmd_dot_dot, Xm, Xm_dot)
l1 = 1;
Xrm_dot = Xmd_dot - l1 * (Xm - Xmd);
Sm = Xm_dot - Xrm_dot;
Xm_tilda = Xmd-Xm;
```

کد حلقه ی داخلی نیز به صورت زیر می باشد:

```
zz = [1;sin(fn*t);sin(2*fn*t)];

Zd = [zz, zeros(6, 1); zeros(3, 1), zz, zeros(3, 1); zeros(6, 1), zz];
Zc = Zd;
Zg = zz;
Zf = zz;

D_hat = W_d' * Zd;
C_hat = W_c' * Zc;
g_hat = W_g' * Zg;
f_hat = W_f' * Zf;

Qd = 100 * eye(9);
Qc = 100 * eye(9);
Qg = 100 * eye(3);
Qf = 100 * eye(3);

P = -(pinv(Mm)*Cm)*Xm_dot - (pinv(Mm)*Km)*(Xm-X0) + pinv(Mm)*(fth-Kf*fpa) + ((lambda3m)^2)*Xm_tilda;
Fm = D_hat * P + C_hat * Xrm_dot + g_hat + f_hat;

W_d_dot = (Sm*P*(Zd')*pinv(Qd))';
W_c_dot = (Sm*(Xrm_dot')*(Zc')*pinv(Qc))';
W_g_dot = (Sm*(Zg')*pinv(Qg))';
W_f_dot = (Sm*(Zf')*pinv(Qf))';
```