**ROBUST CONTROL UNTUK SISTEM ROTATIONAL/TRANSLATIONAL ACTUATOR**

**LAPORAN TUGAS AKHIR**

**TEKNIK KENDALI LANJUT**

**(TKEE163125)**

****

**Disusun Oleh:**

**Bagas Budhi Permana**

**16/396785/TK/44798**

**Resha Dwika Hefni Al-Fahsi**

**16/394959/TK/44251**

**Vicko Pranowo**

**16/394966/TK/44258**

**PROGRAM STUDI TEKNIK ELEKTRO**

**DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA**

**2018**

**Abstrak**

Laporan tugas akhir ini membahas tentang *robust control* pada sistem *rotational/translational actuator*. Kinematis dari sistem tersebut dirumuskan dari model non-linearnya. Kemudian untuk mempermudah dalam analisis kendali dengan *robust control* dilakukan linearisasi sistem sehingga dapat dicari *state space-*nya. Dilakukan juga analisis *controllable* dan *observable* pada sistem tersebut. Menggunakan LQR untuk mencari nilai matriks *gain* yang tepat untuk kestabilan. Lalu, Melakukan perhitungan dan *norm*. Langkah terakhir setelah mencari *internal stability* dilakukan *coprime factorization* untuk sistem tersebut. Semua simulasi perhitungan pada sistem dilakukan menggunakan Matlab® 1 R2016a.

Copyright © 2018 Bagas Budhi Permana, Resha Dwika Hefni Al-Fahsi, Vicko Pranowo from Universitas Gadjah Mada.

This document may be used for academic purposes with reference to this document.

1Matlab and Simulink are trademarks of MathWorks Inc. USA.

**Daftar Isi**

**Abstrak1**

**Daftar Isi2**

**Bab 1: Pendahuluan3**

**Bab 2: Sistem *Rotational/Translational Actuator*4**

2.1. Linearisasi4

2.2. *Observability* dan *Controlability*5

2.2.1. Program Matlab® 5

**Bab 3: LQR *Controller*6**

3.1. Menghitung *Gain*6

3.1.1. Program Matlab® 6

**Bab 4: dan *Space*7**

4.1. Menghitung dan *norm*7

4.1.1. Program Matlab® 8

**Bab 5: *Internal Stability* dan *Coprime Factorization*9**

5.1. *Internal Stability*9

5.2. *Coprime Factorization*9

5.2. Program Matlab®10

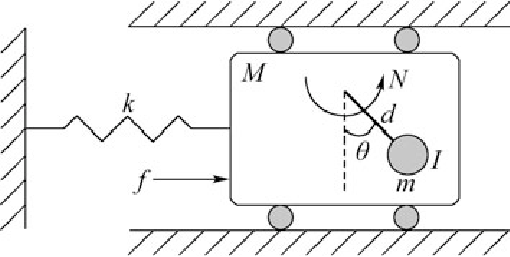
**Daftar Pustaka12**

**A Lampiran Program13**

**Bab 1**

**Pendahuluan**

Tujuan dari pembuatan laporan ini untuk memberikan penjelasan tentang perancangan *robust*   *control* pada sistem *rotational/translational actuator*. Sistem *rotational/translational actuator* atau biasa disingkat RTAC merupakan sebuah sistem berupa gabungan dua aktuator yang salah satunya bergerak secara rotasi dan aktuator lainnya bergerak secara translasi[2]. Dalam praktiknya aktuator tersebut berupa sistem osilasi yaitu pendulum yang berperan dalam gerakan rotasi dan pegas yang berperan dalam gerakan translasi. Sistem RTAC sendiri sering digambarkan dengan sebuah gerobak yang terpasang sistem pegas yang dipasang secara tegak lurus pada dinding dan di dalam gerobak tersebut terdapat pendulum yang berosilasi (bergerak secara rotasi) sehingga akan menyebabkan gerakan translasi pada gerobak[3].



Gambar 1. Pemodelan Sistem RTAC

Pada bab 2, kita mendefinisikan sistem RTAC dalam *state-space*. Sistem RTAC sendiri pada dasarnya merupakan sistem non-linear sehingga diperlukan proses linearisasi untuk mendapatkan bentuk *state-space*-nya. Dengan mengetahui bentuk *state-space* suatu sistem kita dapat memeriksa apakah sistem tersebut *observable* dan *controllable.*

Pada bab 3 dilakukan perhitungan menggunakan LQR untuk mendapatkan nilai matriks *gain* K. Matriks tersebut digunakan untuk membuat sistem stabil dengan membuat sistem menjadi *closed-loop*.

Pada bab 4, dengan menganalisis dan *space* sistem RTAC dapat dan *norm* sistem tersebut sehingga dapat diukur seberapa *robust* sistem tersebut.

Pada bab 5 dibahas mengenai *internal stability* dan *coprime factorization.* Untuk mencari *internal stability* sendiri diasumsikan sistem sesuai standar dan merupakan sistem yang *well-posed.* Dengan mengetahui *internal stability* sistem, *coprime factorization* sistem dapat dihitung.

**Bab 2**

**Sistem *Rotational/Translational Actuator***

Seperti yang dijelaskan sebelumnya sistem RTAC merupakan sistem non-linear. Sistem non-linear sendiri memiliki *state* yang *dependent* dengan *state* lainnya. Pada sistem yang akan dibahas, terdapat 4 *state* yang didefiniskan pada *state-space*. Untuk mendapatkan *state-space* sistem tersebut dilakukan linearisasi dengan metode linearisasi Jacobian.

**2.1. Linearisasi**

Linearisasi Jacobian sendiri menggunakan turunan parsial setiap *state* terhadap suatu fungsi yang mendefinisikan salah satu turunan *state* pada sistem.Untuk sistem RTAC didefinisikan *state-space* non-linearnya[4]:

Di mana:

= perpindahan gerobak dari titik setimbang, m

= kecepatan linear gerobak, m/s

= posisi sudut pendulum, rad

= kecepatan sudut pendulum, rad/s

M = massa gerobak, 1.3608 kg

m = massa pendulum, 0.096 kg

r = panjang tali pendulum, 0.0592 m

k = konstanta/kekakuan pegas, 186.3 N/m

F = gaya translasi, gaya gangguan, 5 N

N = torsi pada pendulum, kontrol pada \_\_\_.sistem, 0.41861 Nm

I = inersia pendulum, 0.0002175 kg/m2

Dengan linearisasi Jacobian menggunakan *operating point, =* dan  *=* , didapatkan matriks-matriks *state-space*-nya:

Untuk matriks A-nya:

Sedangkan untuk matriks B-nya:

Dengan matriks *C* = **I4×4**dan *D* = **04×2**. Sehingga bentuk *state-space*, = dan = , dari sistem tersebut telah terbentuk.

**2.2. *Observability* dan *Controlability***

Dengan mengetahui bentuk lengkap dari *state-space* dapat ditentukan *observability* dan *controllability* dari sistem tersebut. Untuk mencari *observability* dari suatu sistem digunakan matriks dan mencari *observability* dari suatu sistem digunakan matriks Syarat sistem *observable* ketika matriks merupakan matriks *full-column rank* dan sistem *controllable* ketika matriks merupakan matriks *full-row rank*.

**2.2.1. Program Matlab®**

Dengan menggunakan Matlab® dapat dicari:

|  |
| --- |
| Ob=obsv(A,C); [n,m]=size(Ob); unob = m-rank(Ob); %observable  if(unob==0)  disp('Given System is Observable.');  else  disp('Given System is Unobservable');  end  Co = ctrb(A,B); [n,m]=size(Co); unco=n-rank(Co); %controllable  if(unco==0)  disp('Given System is Controllable.');  else  disp('Given System is Uncontrollable');  end |

Hasil dari simulasi tersebut didapatkan bahwa sistem *observable* dan *controllable.*

**Bab 3**

**LQR *Controller***

Pada teori *optimal control* suatu sistem dinamis berjalan dengan membuat minimal suatu *cost function*. Sistem dinamis tersebut dimodelkan dalam persamaan diferensial sedangkan *cost function* berupa fungsi kuadrat. Solusi dari persamaan kuadrat tersebut diselesaikan dengan metode LQR atau Linear Qudratic Regulator. Meskipun lingkup sistem yang akan dibahas merupakan *robust control* sedangkan LQR merupakan lingkup *optimal control*, LQR berguna untuk mencari kestabilan dalam sistem.

**3.1. Menghitung *Gain***

Pada sistem yang akan dianalisis bukanlah sebuah sistem yang stabil, apabila dicari *eigenvalue*-nya, , memiliki nilai lebih dari 0. Di mana kestabilan sistem diindikasikan bahwa *pole-pole*-nya berada di sisi sebelah kiri bidang y. Dengan adanya nilai positif pada *eigenvalue* matriks *A* maka sistem tersebut bukanlah sistem yang stabil. Dengan menyelesaikan persamaan aljabar Riccati kontinyu waktu:

di mana dan berturut-turut didefinisikan sebagai:

Di mana merupakan sebuah konstanta yang diatur sedemikian sehingga dapat dicari matriks *gain* K yang membuat sistem stabil dengan menghitung:

dengan merupakan solusi dari persamaan aljabar Riccati kontinyu waktu. Sehingga didapatkan matriks *A* baru dengan sistem *closed-loop*:

Sistem akan stabil jika dan hanya jika semua *eigenvalue* dari matriks *A* benar-benar berada di bagian negatif (bernilai negatif).

**3.1.1.** **Program Matlab®**

Dengan menggunakan Matlab® dapat dicari:

|  |
| --- |
| %Mengecek kestabilan (melihat apakah pole berada di half-right plane)  pole = eig(A);  %Membuat stabil dengan menggunakan LQR  Q=C'\*C;  R=0.01\*eye(2);  p=care(A,B,Q,R);  K=inv(R)\*B'\*p;  Anew=A-B\*K; Bnew=B; Cnew=C; Dnew=D;  polenew=eig(Anew); |

Dari hasil simulasi didapatkan semua nilai *eigenvalue*-nya bernilai negatif sehingga sistem menjadi stabil.

**Bab 4**

**dan *Space***

Jika pada *optimal control* kita mengenal adanya LQR maka pada *robust control* terdapat . Tujuan yang paling utama dari sistem kendali tersebut adalah untuk mencapai performa sesuai spesifikasi tertentu sehingga sebagai tambahan dapat memperoleh *internal stability.* Performa pada suatu sistem dapat didefinisikan seberapa besar isyarat *error* yang dapat dikendalikan. Sehingga dengan mengukur *norm* pada suatu sistem kendali kita dapat mengukur seberapa *robust* sistem tersebut[1].

**4.1. Menghitung dan *norm***

Diketahui persamaan *state* suatu sistem:



*norm* sistem didefinisikan sebagai dengan dan berturut-turut adalah *controllability* dan *observability Gramian,* yang keduanya merupakan matriks simetris yang diperoleh dari persamaan Lyapunov:

Untuk nilai sendiri didefinisikan sebagai:

Nilai *norm* sendiri dapat dicari menggunakan *bisection algorithm*. *Bisection algorithm* merupakan algoritma yang bekerja secara perulangan. Untuk mencari nilai *norm* pada setiap perulangan pada algoritma tersebut memeriksa setiap *eigenvalue* dari untuk menentukan apakah ada *eigenvalue* yang benar-benar imajiner. Di mana merupakan *Hamiltonian matrix* yang didefinisikan sebagai

Algoritma tersebut akan konvergen secara linear dan perkiraan nilainya bergantung terhadap akurasi nilai . Sebelum algoritma bekerja didefinisikan nilai sebagai batas bawah dan batas atas, dengan = 0 dan bernilai sangat besar, bisa mendekati tak hingga, untuk mencapai konvergen relatif terhadap . Asumsikan matriks *A* stabil dan nilai .

**Tahap 1:** Hitung batas-batas pada *bisection algorithm*, dan dengan,

**Tahap 2:** Atur . Jika , **berhenti**.

**Tahap 3:** Hitung .

**Tahap 4:** Periksa apakah ada *eigenvalue* yang benar-benar imajiner pada . Jika ada , \_\_\_\_\_\_\_..sebaliknya .

Setelah **berhenti**:

**Tahap 5:**

Fungsi merupakan sebuah fungsi yang mengembalikan nilai maksimum *eigenvalue* dari . Untuk nilai sendiri didefinisikan sebagai:

**4.1.1. Program Matlab**

Dengan menggunakan Matlab® dapat dicari:

|  |
| --- |
| %Perhitungan H2 norm, Hinfinity norm, L2 dan L infinity    G=pck(Anew,Bnew,Cnew,Dnew);  plantbaru=ss(Anew,Bnew,Cnew,Dnew);  %Observability Gramian Matrix  Qgram=gram(cjt(Anew),cjt(Cnew));    %computing H2 and L2  H2=norm(plantbaru,2)  % atau  H2=h2norm(G)  L2=sqrt(trace(cjt(Bnew)\*Qgram\*Bnew))    %hitung h infinity dan l infinity  Hinf = hinfnorm(G,0.0001)  Linf = linfnorm(G,0.0001) |

Dari hasil simulasi didapatkan dan . Karena nilai maka sistem tersebut stabil.

**Bab 5**

***Internal Stability* dan *Coprime Factorization***

Pada bagian ini akan dibahas mengenai *internal stability* dan *coprime factorization* pada sistem. Untuk mengecek *internal* *stability* diasumsikan sistem tersebut sesuai standar dan merupakan sistem yang *well-posed*[1]. *Coprime factorization* sistem dapat dihitung ketika sistem mencapai *internal stability*.

**5.1. *Internal Stability***

*Internal stability* menunjukkan kestabilan seluruh *transfer function* masukan ke keluaran ketika sistem diberikan *feedback controller* . Sistem dikatakan *internally stable* jika dan hanya jika sistem tersebut *well-posed* dan:

1. Jumlah *pole* RHP dari

Dari pernyataan di atas, diperlukan *transfer matrix controller ,* namun yang didapat adalah matriks *state closed loop* yang didapat dari *controller state feedback* LQR. Lalu dapat dicari *transfer matrix controller*  jika diketahui *transfer matrix* sistem *closed-loop* dan *transfer matrix plant* *:*

Sehingga didapat digunakan untuk persamaan (ii).

**5.2. *Coprime Factorization***

*Coprime factorization* digunakan untuk menguji adanya *pole-zero cancellation* agar sistem terjamin *internal stability*. *Pole-zero cancellation* pada *pole* RHP dapat membuat bagian yang tidak stabil menjadi tidak terdeteksi. Sistem yang difaktorisasi dikatakan *coprime* sehingga tidak terjadi *pole-zero cancellation.* Sebuah sistem dikatakan *coprime* jika dapat difaktorisasi menjadi RCF (*Right Coprime Factorization*) maupun LCF (*Left Coprime Factorization*) . Sedemikian hingga:



*Coprime factorization* dari *transfer matrix* dapat diberikan representasi *feedback*. Representasi *state feedback* dapat langsung dijadikan bentuk RCF dengan:



Kemudian diuji apakah .

**5.3. Program Matlab®**

Dengan menggunakan Matlab® dapat dicari:

|  |
| --- |
| %% Internal Stability  syms s real% Symbolic variable 's'    % Open Loop System  P = minreal(openloop) %Reduce to minimum order  for i=1:1:4 % Convert transfer function to symbolic  for j=1:1:2  symP(i,j) = tf2sym(P,i,j);  symG\_cl(i,j) = tf2sym(closedloop,i,j);  end  end    %% Dari persamaan mencari controller K jika diketahui system closed loop G\_cl dan plant P  %% K = [(G(s)^-1)P(s) - I] P(s)^-1  symK = simplify((pinv(symG\_cl)\*symP\*pinv(symP)- pinv(symP)));  symK\_hat = -symK;  for i=1:1:2 % Convert symbolic back to transfer function  for j=1:1:4  K\_hat(i,j) = syms2tf(symK\_hat(i,j));  end  end  K\_hat    %% RHP poles test  symPK\_hat = symP\*symK\_hat; % P(s)K\_hat(s)  for i=1:1:4 % Convert symbolic back to transfer function  for j=1:1:4  PK\_hat(i,j) = syms2tf(symPK\_hat(i,j));  end  end  %% (I-P\*K\_hat)^-1  symIStfmatrix = simplify(inv(eye(size(symP\*symK\_hat)))-symP\*symK\_hat)  for i=1:1:4 % Convert symbolic back to transfer function  for j=1:1:4  IStfmatrix(i,j) = syms2tf(symIStfmatrix(i,j));  end  end    if(isstable(IStfmatrix)==0)  disp('Not Internally Stable')  else  disp('Internally Stable')  end  %% Coprime Factorization  syms s real  F = -lqr(A, B, Q, R); %gain F sehingga A+BF stabil  %% M(s)  AM = A+B\*F;  BM = B;  CM = F'\*F;  DM = eye(size(D));  ssM = ss(AM,BM,CM,DM); % state space of system M  tfM = tf(ssM) % transfer function of system M    %% N(s)  AN = A+B\*F;  BN = B;  CN = C+D\*F;  DN = D;  ssN = ss(AN,BN,CN,DN); % state space of system N  tfN = tf(ssN) % transfer function of system N    %Convert to Symbolic  for i=1:1:4  for j=1:1:2  symN(i,j) = tf2sym(tfN,i,j);  symM(i,j) = tf2sym(tfM,i,j);  end  end    % diuji apakah P=N\*M^-1  symPcop = symN\*pinv(symM)    %Convert symbolic back to transfer function  for i=1:1:4  for j=1:1:4  Pcop(i,j) = syms2tf(symPcop(i,j));  end  end  Pcop |

Dari hasil simulasi didapat bahwa sistem ini tidak *internally stable* karena

memiliki *pole* yang berada di *right half plane* dan tidak *coprime*akibat hasil faktorisasi dan yang berupa  menghasilkan hasil yang tidak sama dengan *plant* awal . Pada dasarnya sistem ini tidak dapat dicari *coprime factorization*-nya karena matriks memiliki ukuran yang tidak sesuai dengan matriks pada faktorisasi sehingga diganti dengan .

**Daftar Pustaka**

1. Zhou, Kemin, and John Comstock Doyle. *Essential of Robust Control*. Vol. 104. Upper Saddle River, NJ: Prentice Hall, 1998.
2. Tavakoli, Mahdi, Hamid D. Taghirad, and Mehdi Abrishamchian. "Identification and robust H∞ control of the rotational/translational actuator system." *International Journal of Control, Automation, and Systems* 3.3 (2005): 387-396.
3. Adlgostar, R., H. Azimian, and H. D. Taghirad. "Robust H∞, H 2/H∞ controller for rotational/translational actuator (RTAC)." *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*. IEEE, 2006.
4. Rosales, Andres, et al. "Controller designed by means of numeric methods for a benchmark problem: RTAC (Rotational Translational Actuator)." *Electronics, Robotics and Automotive Mechanics Conference, 2006*. Vol. 1. IEEE, 2006.

**Appendix A**

**Lampiran Program**

Berikut lampiran program yang digunakan untuk simulasi sistem RTAC ini. Program dibuat menggunakan Matlab® dan tersedia secara online di https://github.com/reshalfahsi/robust-control-rtacsystem. Program tersebut kami bagi menjadi 3 bagian yaitu dua buah program fungsi bantuan dan sebuah program utama. Program utama kami beri nama rtacSystem.m sedangkan dua buah fungsi bantuan kami beri nama masing-masing tf2sym.m dan syms2tf.m.

A.1. rtacSystem

|  |
| --- |
| clear all;  %nilai paramter  clear all;  M = 1.3608;  m = 0.096;  R = 0.0592;  I = 2.175\*10^-4;  k = 186.3;    %pembagi di persamaan  h = (M+m)\*(I+0.5\*m\*R^2);    %nilai di matriks A  a11 = 0;  a12 = 1;  a13 = 0;  a14 = 0;  a21 = (-k\*(I+m\*R^2))/h;  a22 = 0;  a23 = -(((m^2)\*(R^3)\*(0.555\*((M+m)\*I+(M\*m\*R^2))+m\*(0.616\*I+0.555\*m\*R^2)))/(h^2));  a24 = (1.57\*I\*m\*R+1.11\*(m^2)\*(R^3))/h;    a31 = 0;  a32 = 0;  a33 = 0;  a34 = 1;    a41 = (0.707\*m\*R\*k)/h;  a42 = 0;  a43 = 0;  a44 = -((0.785\*(m^2)\*(R^2))/h);    %nilai di matriks B  b11 = 0;  b12 = 0;  b21 = (I+m\*R^2)/h;  b22 = (-0.707\*m\*R/h);  b31 = 0;  b32 = 0;  b41 = (-0.707\*m\*R/h);  b42 = (M+m)/h;    %matriks A,B,C, dan D  A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];  B = [0 0; b21 b22; 0 0; b41 b42];  C = eye(4);  D = zeros(4,2);    %================================================================  %observable  Ob=obsv(A,C);  [n,m]=size(Ob);  unob = m-rank(Ob);  if(unob==0)  disp('Given System is Observable.');  else  disp('Given System is Unobservable');  end    %controllable  Co = ctrb(A,B);  [n,m]=size(Co);  unco=n-rank(Co);  if(unco==0)  disp('Given System is Controllable.');  else  disp('Given System is Uncontrollable');  end    %================================================  %Mengecek kestabilan (melihat apakah pole berada di half-right plane)  plantlama=ss(A,B,C,D);  openloop=tf(plantlama)  pole = eig(A);    %Gs Gu decomposition sblm LQR  Glama=pck(A,B,C,D);  [GsLAMA,GuLAMA]=sdecomp(Glama);    %Membuat stabil dengan menggunakan LQR  Q=C'\*C;  R=0.01\*eye(2);  p=care(A,B,Q,R);  K=inv(R)\*B'\*p;    Anew=A-B\*K;  Bnew=B;  Cnew=C;  Dnew=D;  polenew=eig(Anew);    %================================================  %Diperoleh sistem baru yang lebih stabil  %Perhitungan H2 norm, Hinfinity norm, L2 dan L infinity    G=pck(Anew,Bnew,Cnew,Dnew);  plantbaru=ss(Anew,Bnew,Cnew,Dnew);  closedloop=tf(plantbaru);    %Gs Gu decomposition stlh LQR  [Gs,Gu]=sdecomp(G)    %Observability Gramian Matrix  Qgram=gram(cjt(Anew),cjt(Cnew));    %computing H2 and L2  H2=norm(plantbaru,2)  % atau  H2=h2norm(G)  L2=sqrt(trace(cjt(Bnew)\*Qgram\*Bnew))    %hitung h infinity dan l infinity  Hinf = hinfnorm(G,0.0001)  Linf = linfnorm(G,0.0001)    %% Internal Stability  syms s real% Symbolic variable 's'    % Open Loop System  P = minreal(openloop) %Reduce to minimum order    for i=1:1:4 % Convert transfer function to symbolic  for j=1:1:2  symP(i,j) = tf2sym(P,i,j);  symG\_cl(i,j) = tf2sym(closedloop,i,j);  end  end    %% Dari persamaan mencari controller K jika diketahui system closed loop G\_cl dan plant P  %% K = [(G(s)^-1)P(s) - I] P(s)^-1  symK = simplify((pinv(symG\_cl)\*symP\*pinv(symP)- pinv(symP)));  symK\_hat = -symK;  for i=1:1:2 % Convert symbolic back to transfer function  for j=1:1:4  K\_hat(i,j) = syms2tf(symK\_hat(i,j));  end  end  K\_hat    %% RHP poles test  symPK\_hat = symP\*symK\_hat; % P(s)K\_hat(s)  for i=1:1:4 % Convert symbolic back to transfer function  for j=1:1:4  PK\_hat(i,j) = syms2tf(symPK\_hat(i,j));  end  end    %% (I-P\*K\_hat)^-1  symIStfmatrix = simplify(inv(eye(size(symP\*symK\_hat)))-symP\*symK\_hat)  for i=1:1:4 % Convert symbolic back to transfer function  for j=1:1:4  IStfmatrix(i,j) = syms2tf(symIStfmatrix(i,j));  end  end    if(isstable(IStfmatrix)==0)  disp('Not Internally Stable')  else  disp('Internally Stable')  end    %% Coprime Factorization  syms s real  F = -lqr(A, B, Q, R); %gain F sehingga A+BF stabil  %% M(s)  AM = A+B\*F;  BM = B;  CM = F'\*F;  DM = eye(size(D));  ssM = ss(AM,BM,CM,DM); % state space of system M  tfM = tf(ssM) % transfer function of system M    %% N(s)  AN = A+B\*F;  BN = B;  CN = C+D\*F;  DN = D;  ssN = ss(AN,BN,CN,DN); % state space of system N  tfN = tf(ssN) % transfer function of system N    %Convert to Symbolic  for i=1:1:4  for j=1:1:2  symN(i,j) = tf2sym(tfN,i,j);  symM(i,j) = tf2sym(tfM,i,j);  end  end    % diuji apakah P=N\*M^-1  symPcop = symN\*pinv(symM)    %Convert symbolic back to transfer function  for i=1:1:4  for j=1:1:4  Pcop(i,j) = syms2tf(symPcop(i,j));  end  end  Pcop |

A.2. tf2sym

|  |
| --- |
| function tfs = tf2sym(tf,n,m)  syms s real  [num, den] = tfdata(tf(n,m));  tfs = vpa(poly2sym(cell2mat(num),s)/poly2sym(cell2mat(den),s),4);  % pretty(symb)  End |

A.3. syms2tf

|  |
| --- |
| function[ans] = syms2tf(G)  [symNum,symDen] = numden(G); %Get num and den of Symbolic TF  TFnum = sym2poly(symNum); %Convert Symbolic num to polynomial  TFden = sym2poly(symDen); %Convert Symbolic den to polynomial  ans =tf(TFnum,TFden);  end |