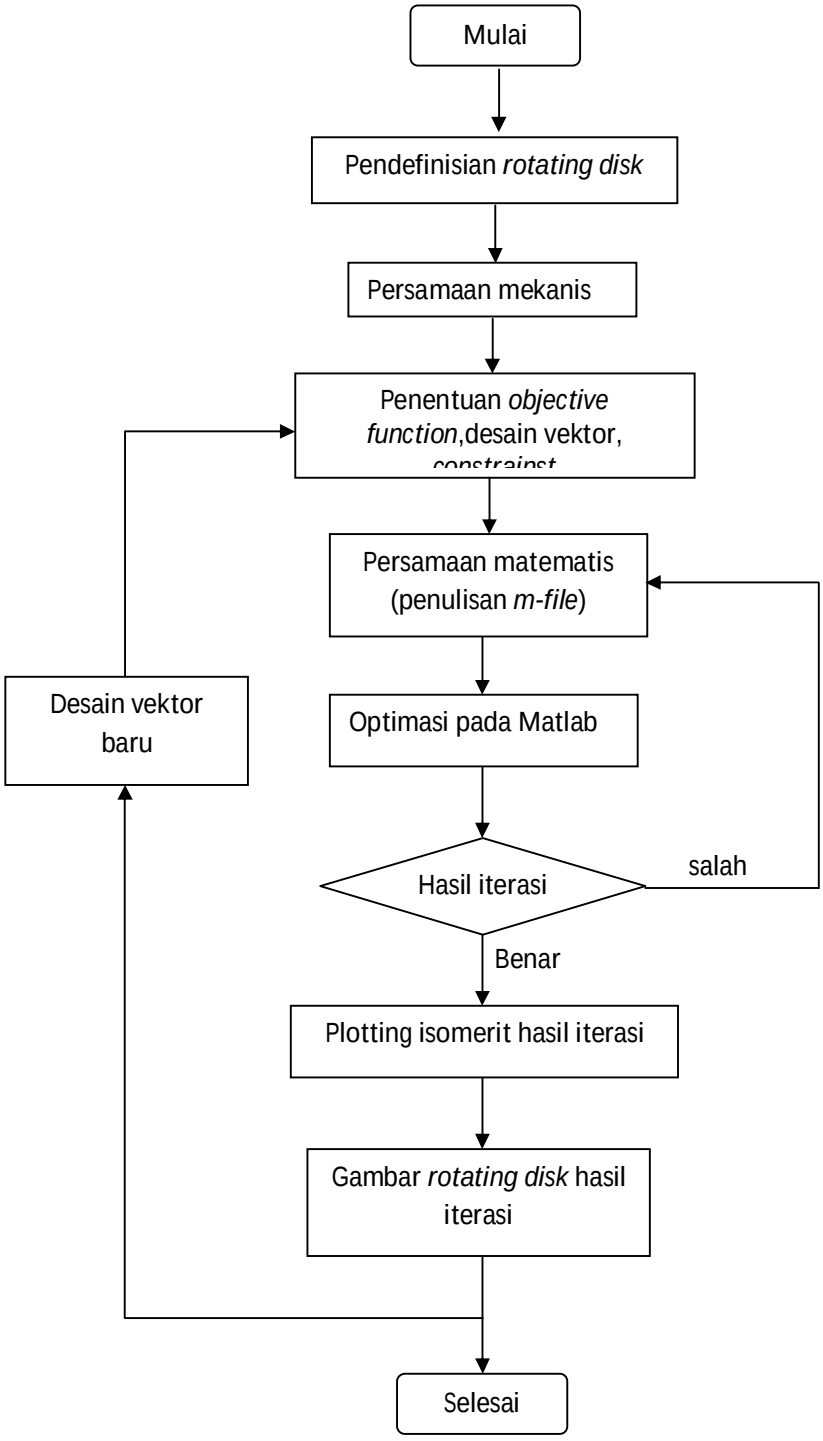
BAB III
METODE ANALISIS

3.1 Langkah Optimasi *Rotating Disk*

Tujuan dari penelitian ini adalah untuk menentukan desain geometri *Rotating Disk* yang sesuai dengan *objective function* yaitu untuk meminimalkan tegangan tangensial maksimum dan rata-rata menggunakan *optimization toolbox* pada Matlab.Untuk itu kita perlu mengetahui persamaan mekanis dari sebuah *rotating disk* kemudian mengubahnya menjadi persamaan matematis pada *m-file.*Adapun langkah-langkah yang dilakukan untuk mendapatkan hasil optimasi adalah sebagai berikut :

a. Menentukan parameter *rotating disk* yang akan dioptimasi.
b. Menentukan persamaan mekanis *rotating disk* bertingkat.
c. Menentukan *objective function*, variabel desain, serta batasan-batasan yang akan digunakan.
d. Menuliskan persamaan *objective function* dan *constraints* pada *m-file*.
e. Melakukan optimasi pada Matlab.
f. Plot variabel hasil optimasi.
g. Menggambar *rotating disk* hasil optimasi.

Pada penelitian ini akan dilakukan beberapa kali proses optimasi dengan variabel desain yang berbeda-beda. Hal ini untuk memperlihatkan bahwa optimasi yang dilakukan mendapatkan hasil yang sesuai dengan *objective function* dan untuk melihat perbedaan geometri *rotating disk* dengan masing masing variabel desain.



Gambar 3.1 Diagram alir proses optimasi *rotating disk*

3.2 Optimasi *Rotating Disk* pada Matlab

Konsep dasar pencarian nilai optimum pada proses optimasi menggunakan metode numerik adalah melalui prosedur berikut :

a. Menentukan nilai tebakan awal X_1

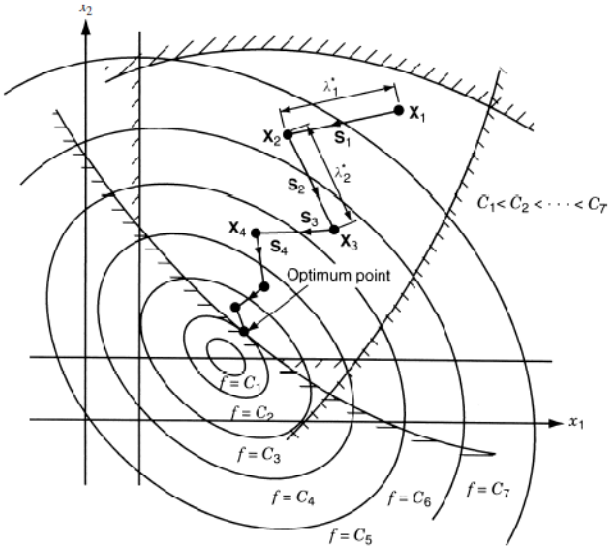
b. Tentukan arah pencarian S_i (i = 1 untuk nilai awal) yang menunjukan nilai optimum secara umum

c. Tentukan panjang langkah pada arah S_i (λ_1*)

d. Temukan nilai tebakan baru X_{i+1}

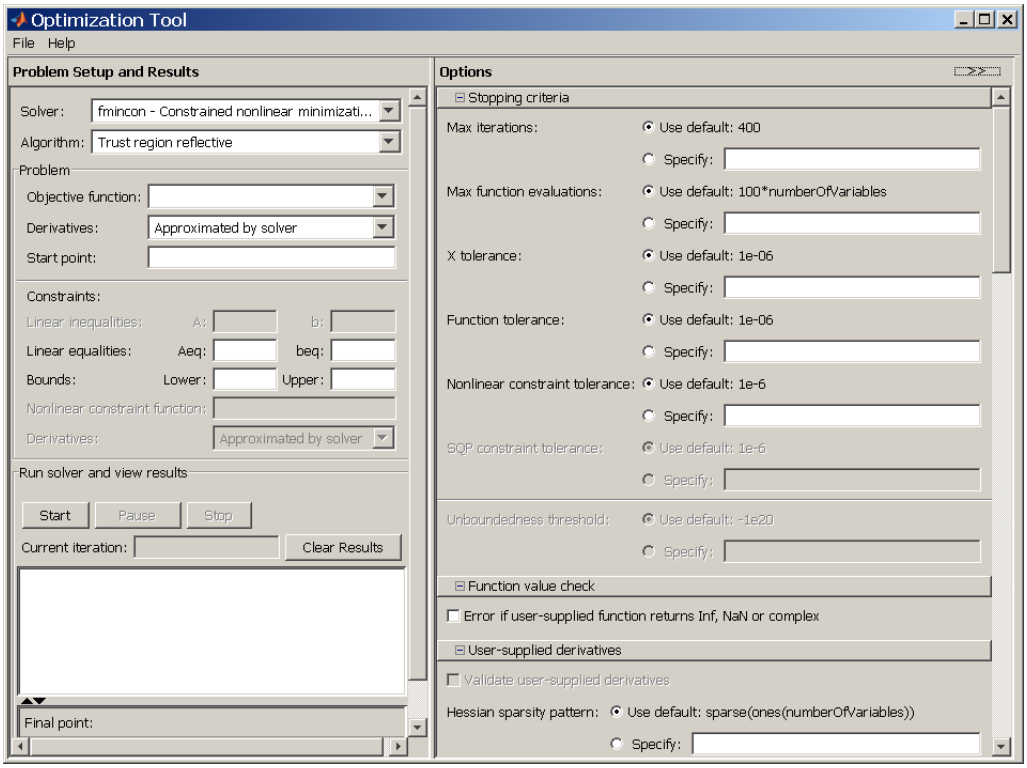
$$X_{i+1} = X_i + S_i \lambda_1^*$$

e. Tes apakah nilai X_{i+1} merupakan nilai optimum, jika iya hentikan prosedur pencarian. Jika tidak tentukan nilai $i = i + 1$ yang baru dan ulangi langkah 2.



Gambar 3.2 Proses iterasi untuk mendapatkan nilai optimum secara numerik [8]

Matlab mempunyai sebuah *toolbox* yang berisi *routine-routine* untuk memecahkan permasalahan optimasi menggunakan beberapa metode yang disebut dengan *optimization toolbox*. Dengan *optimization toolbox* ini kita hanya perlu menentukan metode (*solver*) yang akan digunakan, kemudian memanggil *objective function* dan fungsi batas. Selanjutnya *optimization toolbox* ini akan menjalankan *routine* optimasi sesuai dengan metode yang kita pilih dan menampilkan hasilnya.



Gambar 3.3*Optimization toolbox* pada Matlab

Terdapat beberapa metode atau *solver* pada *optimization toolbox,* namun pada tugas akhir ini hanya akan digunakan fungsi *fminimax* dan *fmincon*. Kedua fungsi ini menggunakan *Gradient-Based-Method* dalam memecahkan suatu permasalahan optimasi. Suatu fungsi akan mencapai nilai optimum jika turunan dari fungsi tersebut bernilai nol. Pada metode ini untuk mencari nilai optimum dari suatu fungsi pertama-tama harus diberikan nilai tebakan awal. Berdasarkan nilai tebakan awal tersebut maka pencarian nilai optimum akan dilakukan, pencarian nilai optimum ini juga didasarkan dari turunan fungsi yang akan dioptimisasi. Kelemahan dari metode ini adalah nilai optimum yang dihasilkan tergantung dari tebakan awal yang diberikan.Sehingga pada fungsi yang memiliki banyak titik puncak, jika tebakan yang diberikan tidak akurat maka yang didapatkan adalah nilai optimum lokal dan bukan nilai optimum global dari fungsi tersebut. Selain itu metode ini bekerja berdasarkan turunan dari fungsi yang akan dioptimisasi. Suatu fungsi akan memiliki turunan jika fungsi tersebut kontinu, sedangkan pada permasalahan yang ada terkadang terdapat fungsi yang tidak kontinu.

A. *fminimax solver*

Fungsi *fminimax* digunakan untuk meminimalkan nilai terburuk (*worst-case value*) dari suatu fungsi multivariabel dengan output lebih dari satu menggunakan tebakan awal dengan batasan maupun tanpa batasan tertentu. Masalah seperti ini dikenal dengan *minimax problem.*Penggunaan metode ini dapat dilakukan dengan menuliskan *objective function* tersebut dalam *m-file* dan dijalankan dengan *optimization toolbox* di matlab.

Persamaan matematis dalam *minimax problem* adalah sebagai berikut :

$$\min_{x} \max_{\{F_1\}} \{F_{(x)}\} such that \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \,.\, x \leq b \\ Aeq \,.\, x = beq \\ lb \leq x \leq ub \end{cases}$$

Dimana x, *b, beq, lb*, dan *ub* merupakan vektor, *A* dan *Aeq* merupakan matrik, *c(x),ceq(x),* dan *F(x),* adalah fungsi dengan nilai output berupa vektor. *F(x), ceq(x),* dan *F(x)* dapat berupa fungsi linier maupun nonlinier.

Syntax fungsi *fminimax* adalah sebagai berikut :

x = fminimax(fun,x0)

x = fminimax(fun,x0,A,b)

x = fminimax(fun,x0,A,b,Aeq,beq)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)

keterangan :

x = nilai hasil optimasi,

fun = nama *objective function*,

x0 = nilai tebakan awal,

A, b = matrix dan vektor sebagai *linear inequality constraints.A.x ≤ b*

Aeq, beq= matrix dan vektor sebagai *linear equality constraints. Aeq.x = beq*

lb, ub=lower bound dan *upper bound*, merupakan nilai batas bawah dan atas dari x

nonlcon= nama fungsi *nonlinear constraint.*

Fungsi *fminimax* ini dipilih karena dianggap sesuai untuk menyelesaikan *objective function* pertama yaitu untuk meminimalkan tegangan tangensial maksimum.*fminimax* ini akan mencari nilai terburuk (dalam hal ini nilai terbesar dari tegangan tangensial) dari suatu fungsi dengan output lebih dari satu, kemudian meminimalkannya.

B. *fmincon solver*

fmincon berfungsi untuk meminimalkan suatu fungsi skalardengan beberapabatasandan variabel menggunakan nilai tebakan awal. Penggunaan metode ini dapat dilakukan dengan mengetikkan *objective function* tersebut dalam *m-file* matlab dan dijalankan dengan *optimization toolbox* di matlab.

Persamaan matematis dalam metode ini adalah sebagai berikut:

$$\min_x f(x) subject to \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A.x \leq b \\ Aeq.\ x = beq \\ lb \leq x \leq ub, \end{cases}$$

Dimana x, *b, beq, lb,* dan *ub* merupakan vektor, *A* dan *Aeq* merupakan matrik, *c(x),ceq(x),* dan *F(x),* adalah fungsi dengan nilai output berupa vektor. *F(x), ceq(x),* dan *F(x)* dapat berupa fungsi linier maupun nonlinear.

Syntax fungsi *fminimax* adalah sebagai berikut :

x = fmincon(fun,x0,A,b)

x = fmincon(fun,x0,A,b,Aeq,beq)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2, ...)

keterangan :

x = nilai hasil optimasi,

fun = nama *objective function*,

x0 = nilai tebakan awal,

A, b = matrix dan vektor sebagai *linear inequality constraints.A.x* $\leq$ *b*

Aeq, beq= matrix dan vektor sebagai *linear equality constraints. Aeq.x = beq*

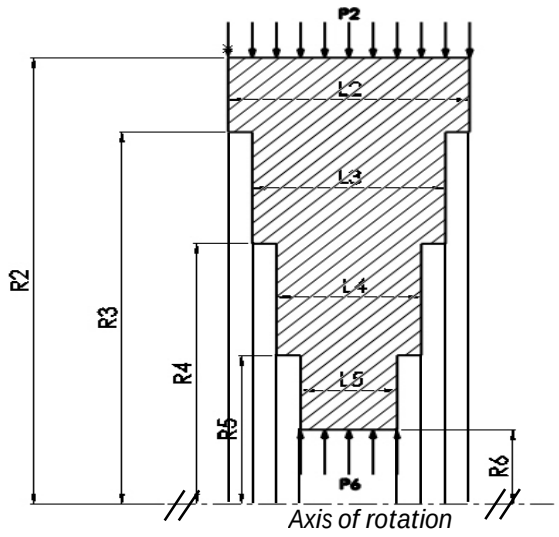
lb, ub = *lower bound* dan *upper bound*, merupakan nilai batas bawah dan atas dari x

nonlcon= nama fungsi *nonlinear constraint*

Fungsi *fmincon* dipilih untuk menyelesaikan *objective function* yang kedua yaitu meminimalkan tegangan tangensial rata-rata. *fmincon* akan menentukan nilai variabel untuk meminimalkan suatu fungsi dengan output skalar bukan vektor dalam hal ini output berupa tegangan tangensial rata-rata. Jika *fmincon* dipakai pada fungsi dengan output berupa vektor (output lebih dari satu), Matlab akan menampilkan hasil berupa eror.

3.2.1 Pendefinisian Permasalahan Optimasi pada Matlab

Proses optimasi *rotating disk* dimulai dengan mendefinisikan desain rotating disk secara umum. Berikut adalah bentuk *rotating disk* yang akan dioptimasi :



Gambar 3.4 Profil potongan melintang *rotating disk* 4 tingkat

Rotating disk berputar dengan kecepatan sudut ω (rad/s) dengan sumbu putar atau *axis of rotation* ditengah (pada porosnya).Tekanan yang bekerja dapat dari dalam maupun dari luar. Dari dalam misalnya diakibatkan tegangan susut pada hasil proses pemasangan poros dengan piringan. Tekanan gas yang bekerja pada sudu turbin dapat mengakibatkan tekanan luar yang bekerja pada *rotating disk*. Karena bentuk piringan simetris maka biasanya pemodelan profilnya hanya ditunjukan pada setengah bagian

saja. Pemodelan ini yang akan kita pakai dalam membentuk model matematisnya. Berikut adalah persamaan matematis *rotating disk* yang akan doptimasi sesuai dengan Gambar 3.4.

Preassigned parameters:

a. Jumlah segmen = 4
b. Material = *High-strength low-alloy steel*
c. Densitasρ = 0.283 lb/in^3
d. Kecepatan putaran$V$ = 100 inches/second
e. Pm = 1001.0 psi
f. Poisson's ratio = 0.3
g. R2 = 6 inches
h. Rm= R6 = 1.0 inch
i. P2 = 0.0 psi
j. L_{min} = 0.6 inches
k. L_{max} = 3.0 inches

Design variables :

L2,L3,L4,L5,R3,R4,R5

Constraints :

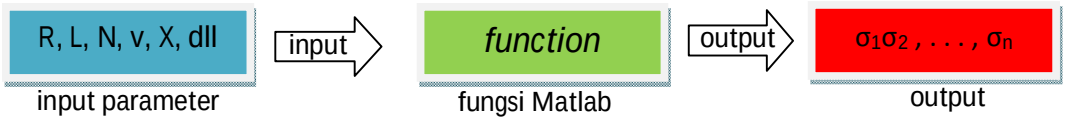
a. $L_{min} \leq Ln \leq L_{max}$
b. R2>R3>R4>R5>R6

Objective function :

a. Minimize the maximum tangential stress (Min σ_t max)
b. Minimize the average tangential stress (Min $\sigma_t$average)

Proses optimasi akan dilakukan berulang-ulang dengan variabel desain yang berbeda, pada tiap optimasi akan ditentukan dua variabel desain yang berbeda dengan satu nilai*objective function,* hal ini ditujukan agar hasil optimasi dapat ditampilkan dalam grafik isomerit atau *contour plot* pada Matlab. Kemudian pada optimasi terakhir akan digunakan semua variabel desain untuk mendapatkan profil *rotating disk* paling optimal sesuai dengan *objective function*.

Untuk mendefinisikan masalah seperti ini pada Matlab, kita perlu menulis sebuah fungsi dalam Matlab dimana fungsi itu menerima input berupa variabel desain baik yang sudah ditetapkan (*preassigned variable*) maupun variabel yang dicari nilainya(*design variable*) dengan output berupa tegangan tangensial pada tiap-tiap antarmuka.



Gambar 3.5 Skema fungsi Matlab

3.2.2 Penulisan *m-file Objective Function*

Agar persamaan mekanik tegangan tangensial pada *rotating disk* dapat dibaca oleh matlab, kita perlu menulis persamaan tersebut dalam bentuk *m-file*. Pada optimasi menggunakan matlab, *objective function* harus dituliskan sebagai sebuah fungsi dengan input berupa variabel desain dan output sebagai parameter yang akan dioptimasi. Semua parameter yang sudah ditentukan (*preassigned parameter*) harus didefinisikan pada fungsi tersebut.Berikut adalah fungsi *objective function* dalam bentuk *m-file.*

A. Pendefinisian nama fungsi beserta input-output

```
function f = ConstructDisk (x)
```

Skrip diatas merupakan pendefinisian awal dari fungsi matlab, dengan f sebagai variabel output dan x sebagai input, dimana x merupakan nilai variabel desain. *Construct disk* mendefinisikan nama fungsi.

B. Pendefinisian variabel

Dalam mendefinisikan nilai variabel matlab menggunakan bentuk matrik, tiap variabel yang memiliki nilai lebih dari satu ditulis dalam matlab sebagai matrik dengan jumlah kolom dan baris tertentu. Sebagai contoh, jari-jari lingkaran didefinisikan sebagai matrik R, dimana matrik R merupakan matrik 1 baris dengan jumlah kolom sesuai dengan jumlah R(R = [R1 R2 R3 R4 Rn]).

```
R = [R1 R2 R3 R4 R5 R6]; % jari-jari rotating disk dalam inch
L = [L1 L2 L3 L4 L5 L6]; % ketebalan rotating disk dala inch
V = 1;                   % kecepatan putar (1 = 100 inch/sec^2)
nu = 0.3;                % poisson's ratio
rho = 0.284;             % density material (lb/in^3)
P(2)=0.0;                % tekanan luar (psi)
P(6)=1000.1;             % tekanan dalam / tekanan pada poros (psi)
```

C.Perhitungan awal

Selanjutnya adalah menuliskan persamaan dasar untuk menghitung distribusi tekanan pada tiap *interface* (persamaan 2.10 s/d 2.16).Untuk mempermudah perhitungan digunakan fitur *loop*, fitur ini adalah sebuah proses iterasi atau perhitungan berulang-ulang.

```
for n=1:5,
    %Bn
B(1,n) = (2*(R(n)/R(n+1))^2)/((R(n)/R(n+1))^2-1);
end

for n=2:4,
%An
    A(1,n) = (((3+nu)*rho*10^4)/4)*((R(n)/R(1,2))^2-
(R(n+2)/R(1,2))^2);
%Cn
C(1,n) = 2/((R(n+1)/R(n+2))^2-1);
%Dn
    D(1,n) = (((1-nu)+(1+nu)*(R(n)/R(n+1))^2)/((R(n)/R(n+1))^2-
      1))+(L(n)/L(n+1))*(((1+nu)+(1-
      nu)*(R(n+1)/R(n+2))^2)/((R(n+1)/R(n+2))^2-1));
%Kn
K(1,n) = A(1,n)/C(1,n);
%Qn
Q(1,n) = (B(1,n)/C(1,n))*(L(n-1))/L(n);
%Un
U(1,n) = D(1,n)/C(1,n);
end
```

Skrip di atas akan menghasilkan matrik baru dari matrik sebelumnya melalui perhitungan berulang tiap variabel. Contoh dalam perhitungan B,

```
for n=1:5,
    %Bn
B(1,n) = (2*(R(n)/R(n+1))^2)/((R(n)/R(n+1))^2-1);
end
```

Matlab akan menghitung nilai B berulang sebanyak n-kali, dimana n= 1 s/d 5 (ditulis dalam *m-file* sebagai n=1:5). Dalam tiap kali perhitungan matlab akan memanggil nilai variabel yang berbeda-beda (dalam hal ini R dan L) dari tiap-tiap matriknya sesuai dengan nilai n, kemudian menuliskan hasilnya pada matrik B, dimana, B=[B(1) B(2) B(3) B(4) B(5)].

D. Ekstrapolasi nilai P3

Seperti yang dijelaskan pada bab sebelumnya (persamaan 2.9) untuk menghitung nilai tekanan pada tiap-tiap *interface* dilakukan dengan menebak dua nilai P3 dan

menghitung nilai Pm (P6) untuk masing-masing tebakan kemudian melakukan ekstrapolasi guna mendapatkan harga P3 aktual. Dituliskan pada matlab sebagai berikut,

```
P3_g =(200-rand(1,2)*(200-100));
     % memberikan 2 nilai acak dalam interval 100-200 dan
      mendefinisikannya sebagai matrik P3_g

for n=1:2,
    P4_g=K(2)*V^2-Q(2)*P(2)+U(2)*P3_g(1,:);
    P5_g=K(3)*V^2-Q(3)*P3_g(1,:)+U(3)*P4_g;
    P6_g=K(4)*V^2-Q(4)*P4_g+U(4)*P5_g;
%menghitung nilai P6 untuk masing-masing nilai P3 tebakan
end

P(3) = interp1(P6_g,P3_g,P(6),'linear','extrap');
    %melakukan ekstrapolasi linear dari nilai P6 dan P3 tebakan untuk
    mendapatkan nilai P3 aktual
```

E. Menghitung nilai P untuk tiap-tiap *interface*

Setelah mendapat nilai P3 aktual, dapat dihitung nilai P untuk masing-masing *interface.*

```
P(4) = K(1,2)*V^2-Q(1,2)*P(2)+U(1,2)*P(3);
P(5) = K(1,3)*V^2-Q(1,3)*P(3)+U(1,3)*P(4);
P(6) = K(1,4)*V^2-Q(1,4)*P(4)+U(1,4)*P(5);
```

Pada tahap ini kita telah mendapatkan semua nilai P yang dibutuhkan, matlab menyimpan nilai P pada matrik P = [P1 P2 P3 P4 P5 P6].

F. Menghitung variabel yang akan dioptimasi

Tahap selanjutnya kita akan menghitung variabel yang akan dioptimasi, pada tugas akhir ini ada dua *objective function* yang akan dihitung. Oleh karena ini kita harus menulis dua *m-file* untuk tiap-tiap *objective function*. Kedua *m-file* ini secara keseluruhan sama, namun berbeda pada pendefinisian nilai f pada fungsinya. Nilai f pada tiap-tiap fungsinya adalah sebagai berikut :

1) Menghitung tegangan tangensial

```
for n=2:5,

E(1,n) = 1/((R(1,n)/R(1,n+1))^2-1);

    F(1,n) = (((((3+nu)*rho)*10^4)/4)*(R(1,n)/R(1,2))^2)+(((1-
             nu)*rho*10^4)/4)*(R(1,n+1)/R(1,2))^2;

tangential_stress(1,n-1)=
    (-(B(1,n)*L(1,n+1)/L(1,n))*P(1,n))+((E(1,n)+(B(1,n)/2)-
     nu/2*(L(1,n)/L(1,n+1)))*P(1,n+1))+F(1,n)*V^2);

End
   f=tangential_stress;
```

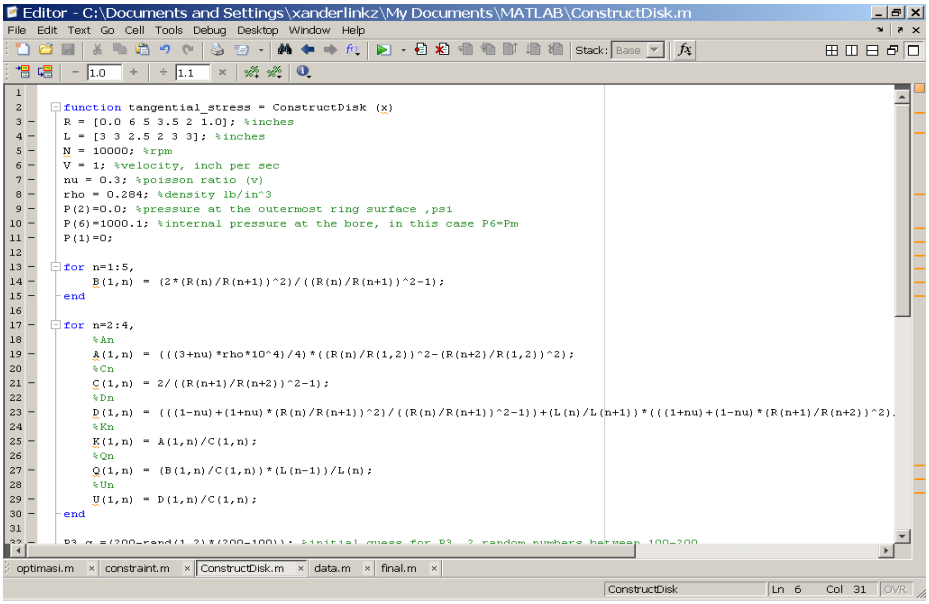
Skrip diatas berdasarkan pada persamaan(2.18) s/d (2.20) dan akan menghasilkan output berupa matrik tegangan tangensial pada tiap-tiap interface yang dituliskan pada matrik [*tangential_stress*].

2)Menghitung tegangan tangensial rata-rata

Untuk menghitung tegangan tangensial rata-rata skrip yang digunakan sama seperti pada *objective function* sebelumnya, hanya saja pada akhir skrip digunakan *command mean* yang artinya menghitung nilai rata-rata pada matrik tersebut.

```
f=mean(tangential_stress);
```

Skrip ini kemudian disimpan dalam bentuk *m-file* dengan nama*ConstructDisk.m* (secara *default* Matlab menyimpan nama file sesuai dengan nama fungsinya), *m-file* secara lengkap dicantumkan pada halaman lampiran.



Gambar 3.6Penulisan *objective function* pada Matlab

3.2.3 Penulisan *Constraint*

Seperti sudah dijelaskan sebelumnya, dalam optimasi terdapat beberapa pembatas-pembatas (*constraints*) yang harus dipenuhi. Pada tugas akhir ini pembatas yang diberikan adalah :

$Lmin \leq Ln \leq Lmax$ (0.6 inch $\leq Ln \leq$ 3.0 inch)

dan

$R2>R3>R4>R5>R6$ (6 inch > R3 > R4 > R5 > 1 inch)

Cara menuliskan *constraint* tersebut dalam *m-file* adalah sebagai berikut :

A. Lmin ≤ Ln ≤ Lmax

Constraint ini menyatakan bahwa masing-masing nilai variabel L harus lebih besar atau sama dengan nilai Lmin dan lebih kecil atau sama dengan nilai Lmax. Pada matlab pembatas seperti ini disebut sebagai *boundary* dan didefinisikan sebagai *lb* dan *ub* (*lower bound* dan *upper bound*).Dimana *lb* dan *ub* berupa vektor pembatas untuk tiap-tiap variabel yang dioptimasi.

```
x0 = [x(1) x(2)];
lb = [0.6 0,6];
ub = [3 3];
```

Tiap kolom pada matrik diatas mewakili batas bawah dan atas tiap-tiap variabel, artinya nilai x(1) mempunyai batas atas kolom pertama matrik lb (0.6) dan batas bawah kolom pertama matrik ub (0.6) demikian juga untuk x(2) mempunyai batas atas dan bawah kolom kedua matrik*lb* dan *ub*, dimana x(1) dan x(2) adalah variabel desain.

B. R2>R3>R4>R5>R6

Constraint seperti ini termasuk dalam *linear inequality constraint*, dalam menangani *constraint* sepertiini pada Matlab perlu dirubah dalam bentuk $A \cdot x \leq b$.*A*adalah matrik yang terdiri atas *m* dan *n*, dimana m adalah jumlah *constraints* pada variabel *X* dengan *n* komponen dan *b* berupa vektor. Untuk lebih jelasnya perhatikan contoh berikut,

Misal kita mempunyai *linear inequalities* berikut sebagai pembatas :

$$x_1 + x_3 \leq 4$$
$$2x_2 - x_3 \geq -2$$
$$x_1 - x_2 + x_3 - x_4 \geq 9$$

Maka m = 3, dan n = 4

Perlu diperhatikan, bahwa matlab mengenali suatu *inequalities constraint* hanya dalam bentuk operator kurang dari sama dengan (≤). Untuk mendapatkan bentuk kurang dari sama dengan,*inequalities* dikalikan dengan -1, sehingga menjadi bentuk sebagai berikut,

$$x_1 + x_3 \leq 4$$

$-2x_2 + x_3 \leq 2$

$-x_1 + x_2 - x_3 + x_4 \leq -9$

Maka matrik*A* dan vektor *b* :

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & -2 & 1 & 0 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ 2 \\ -9 \end{bmatrix}$$

Sama halnya dengan contoh di atas, *constraint* pada tugas akhir ini dapat dituliskan sebagai berikut:

Dengan R2 = 6.0, R6 = 1.0,

$6.0 > R3 > R4 > R5 > 1.0$

Atau dapat ditulis:

R3 < 6.0

R4 < R3

R5 <R4

R5 > 1.0

Seperti yang sudah dijelaskan sebelumnya, matlab hanya mengenali *constraint* dalam bentuk operator kurang dari sama dengan, tetapi karena nilai tiap-tiap R tidak boleh sama maka kita perlu memberi batas dengan mengubah batasan sebagai berikut:

$R3 \leq 5.9$	⇨	$R3 \leq 5.9$
$R4 \leq R3-0.1$		$R4 - R3 \leq -0.1$
$R5 \leq R4-0.1$		$R5 - R4 \leq -0.1$
$R5 \geq 1.1$		$-R5 \leq -1.1$

Jika R3 = x_1, R4 = x_2, dan R5 = x_3, maka matrik*A* dan vektor *b* :

$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

$$b = \begin{bmatrix} 5.9 \\ -0.1 \\ -0.1 \\ -1.1 \end{bmatrix}$$

Dituliskan pada *m-file* sebagai,

```
A = [1 0 0; -1 1 0; 0 -1 1; 0 0 -1];
b = [5.9;-0.1;-0.1;-1.1];
```

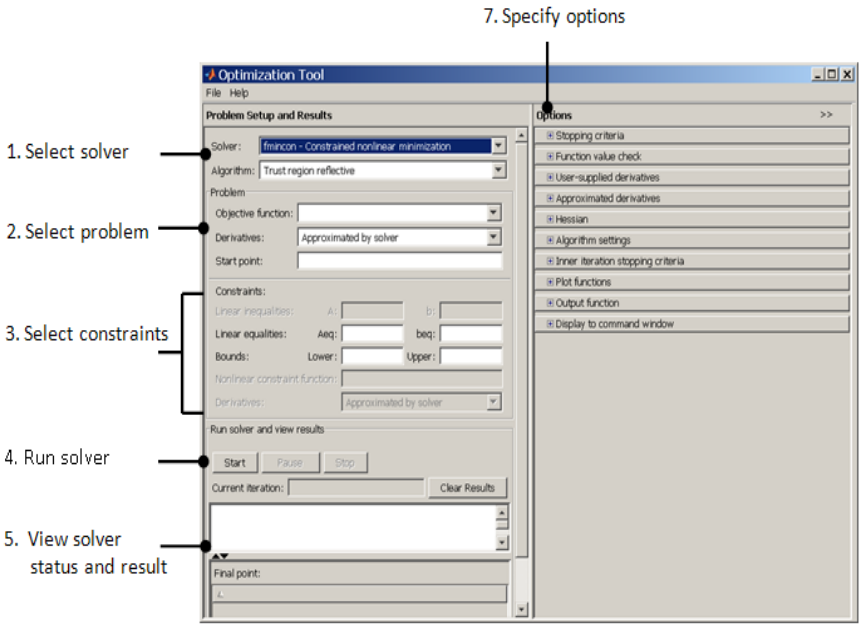
Nilai *A* dan *b* ini yang nanti akan dipanggil dalam proses optimasi sebagai *linear inequalities constraints.*

3.2.4 Proses Optimasi dengan *Optimization Toolbox*

Setelah menuliskan *objective function* dan *constraints* dalam bentuk *m-file* langkah selanjutnya adalah memanggil fungsi tersebut dan menjalankannya pada *optimization toolbox*. Terdapat dua cara untuk menjalankan *optimization toolbox*, yaitu metode GUI(*Graphic User Interface*) dan metode manual.

a. Metode GUI (*Graphic User Interface*)

Metode GUI dilakukan dengan menginputkan perintah *optimtool* pada Matlab *command window*, perintah ini akan memanggil *windowoptimization toolbox.*



Gambar 3.7 *Optimization toolbox*

Keterangan :

1. Menjelaskan tentang jenis *solver* apa yang akan kita gunakan untuk mengoptimasi permasalahan yang akan diselesaikan. Terdapat berbagai macam solver yang dapat kita gunakan. Termasuk diantaranya *fminimax* dan *fmincon*.
2. Menjelaskan tentang *objective function* yang akan diselesaikan. Kita dapat mengetikkan nama *objective function* tersebut sesuai dengan nama yang ada di dalam *m-file*, beserta nilai tebakan awal (*starting point*) yang akan digunakan. Beberapa *solver*, seperti *fmincon dan fminimax* memerlukan nilai tebakan awal agar optimasi dapat berjalan. Jika kita tidak memberikan nilai tebakan awal, maka optimasi ini tidak dapat diselesaikan.
3. Menjelaskan tentang *constraint* yang akan digunakan, termasuk didalamnya memberikan nilai *lower bound* dan *upper bound*, dimana pada tugas ahir ini merupakan nilai L_{min} dan L_{max}.
4. Tombol untuk menjalankan dan menghentikan proses iterasi, termasuk menampilkan jumlah iterasi yang dilakukan.
5. Menjelaskan tentang hasil dari optimasi dan status dari *solver* apakah masih berjalan atau tidak. Semua hasil yang kita simulasikan dapat kita lihat disini dan juga pada *command window* di matlab.
6. Menjelaskan tentang berbagai macam opsi yang dapat kita pilih untuk menentukan *output* dari optimasi tersebut. Opsi-opsi mempunyai berbagai macam pengaruh seperti pada jumlah iterasi maksimal yang kita inginkan, berbagai macam plot hasil optimasi dan sebagainya.

b. Metode Manual

Metode manual dilakukan dengan cara menuliskan routine optimasi pada *m-file*, sesuai dengan sintak untuk masing-masing *solver*. Berikut adalah skrip untuk memulai proses optimasi :

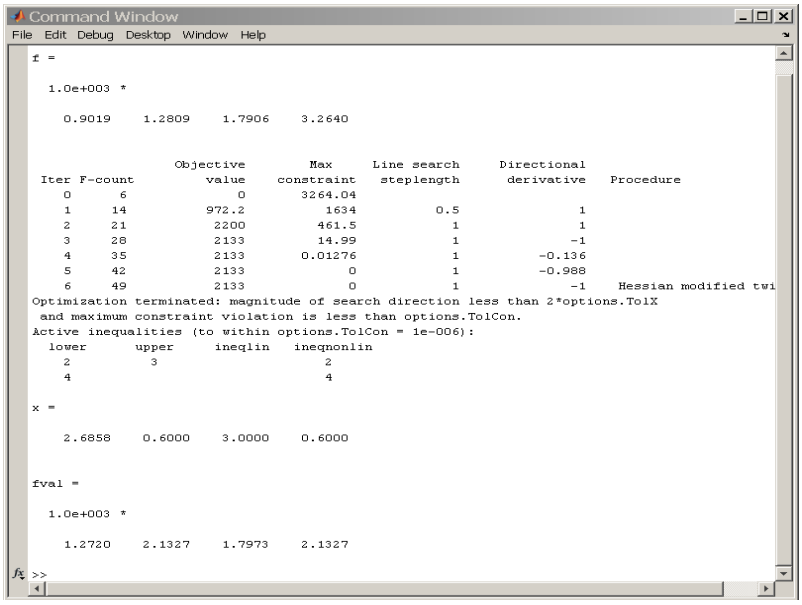
```
x0 = [];% mendeklarasikan sebuah matrik yang berisi tebakan awal untuk
         masing masing variabel desain x0 = [x(1), x(2), x(3), dst],
         dimana x merupakan tebakan awal untuk nilai Ln dan Rn

A = [];% mendeklarasikan matrik A dan b sebagai linear inequality
b = [];% constraints seperti dijelaskan sebelumnya

f=  ConstructDisk(x0),%  memanggil  fungsi  yang  akan  dioptimasi  dan
                         memberikan   nilai   input   x(0)   untuk
```

```
                              membandingkan nilai sebelum optimasi dan
                              sesudah
options = optimset (); % digunakan untuk mendefinisikan opsi-opsi yang
                              akan digunakan.
[x,fval] = fmincon (@ConstructDisk, x0, A, b,[],[],lb,ub,@constraint,
          options), % memanggil fungsi optimasi (fmincon / fminimax)
          untuk
mengoptimasi objective function dengan input dan batasan yang
                        ditentukan.
```

Skrip tersebut disimpan pada *m-file* (disertakan pada halaman lampiran) kemudian dijalankan (*run*) pada Matlab.



Gambar 3.8 Hasil optimasi menggunakan *optimization toolbox*