



REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN CIPTAAN

Dalam rangka perlindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan : EC00201813443, 22 Mei 2018

Pencipta

Nama : **Dr. Julian Hernadi, M.Si.**
Alamat : Tempel Wirogunan UH 3/905 RT 46 RW 11 Tahunan Umbulharjo , Yogyakarta , Di Yogyakarta, 55167
Kewarganegaraan : Indonesia

Pemegang Hak Cipta

Nama : **Universitas Muhammadiyah Ponorogo**
Alamat : Jalan Budi Utomo Nomor 10 , Ponorogo, Jawa Timur, 63471
Kewarganegaraan : Indonesia
Jenis Ciptaan : **Program Komputer**
Judul Ciptaan : **Kumpulan M- File MATLAB Untuk Implementasi Metode Numerik Pada Komputer**

Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia : 15 Maret 2018, di Ponorogo

Jangka waktu perlindungan : Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.

Nomor pencatatan : 000109023

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL



Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001

KUMPULAN M-FILE MATLAB UNTUK IMPLEMENTASI METODE NUMERIK PADA KOMPUTER

Disusun oleh:

Julan HERNADI

Prodi Pendidikan Matematika Universitas Muhammadiyah Ponorogo

Jalan Budi Utomo 10 Ponorogo

email: julan_hernadi@yahoo.com

Karya ini berupa program komputer dalam bahasa Matlab yang digunakan untuk menjalankan berbagai algoritma pada metode numerik dengan menggunakan komputer. Pada karya ini terdapat 87 m-file Matlab dengan nama sesuai apa yang tertulis pada Daftar Isi.

Daftar Isi

1.	Membangkitkan galat aproksimasi faktorial dengan formula Stirling	5
2.	Melihat dampak galat pembulatan komputer pada aproksimasi bilangan e	5
3.	Aproksimasi nilai e dengan deret Taylor	5
4.	Aproksimasi limit mendekati nol	5
5.	Metode bagidua dengan kriteria stopping logaritma	6
6.	Metode bagidua dengan kriteria stopping nilai mutlak fungsi	6
7.	Metode secant	7
8.	Metode false-position	7
9.	Metode Newton persamaan taklinear satu variabel	8
10.	Metode titik-tetap (fixed point)	8
11.	Konstruksi polinomial Lagrange	9
12.	Konstruksi polinomial interpolasi dengan metode Lagrange	9
13.	Koefisien polinomial interpolasi dengan formula Newton	9
14.	Membangun polinomial interpolasi dengan formula Newton	10
15.	Membangun matriks hasil terbagi Newton	10
16.	Membangun polinomial interpolasi melalui data tabel	10
17.	Membangun matriks hasil bagi Hermite	11
18.	Membangun polinomial interpolasi Hermite	11
19.	Membangun polinomial sepotong-sepotong	11
20.	Membangun polinomial spline kubik	12
21.	Membangun polinomial B-spline derajat nol sampai derajat 4	13
22.	Membangun interpolasi B-spline	14
23.	Membangun spline kardinal orde 0 sampai 4	15
24.	Membangun derivatif pertama spline kardinal orde 1 sampai 4	15
25.	Membangun interpolasi spline kardinal	15
26.	Aproksimasi derivatif pertama dengan tiga formula selisih hingga	16
27.	Aproksimasi derivatif pertama fungsi terdiferensial di mana-mana	16
28.	Aproksimasi derivatif pertama fungsi pada array tertentu	17
29.	Aproksimasi derivatif pertama fungsi dengan metode operasi array	17
30.	Aproksimasi derivatif pertama fungsi melalui data tabular	17

31.	Mendeteksi sensitivitas aproksimasi dengan formula selisih terbagi.....	18
32.	Aproksimasi derivatif dengan metode Richardson.....	18
33.	Metode kuadatur dasar	19
34.	Simulasi galat metode kuadatur dasar	19
35.	Metode midpoint bersusun	19
36.	Metode Simpson bersusun	19
37.	Metode trapesium bersusun.....	20
38.	Integral data tabular dengan metode tarpesium	20
39.	Implementasi integral Gauss order 4.....	20
40.	Implementasi integral Gauss bersusun order 4.....	21
41.	Metode integral Romberg.....	21
42.	Metode substitusi mundur untuk menyelesaikan SPL.....	22
43.	Metode substitusi maju untuk menyelesaikan SPL	22
44.	Membentuk SPL ke dalam bentuk segitiga atas	22
45.	Metode eliminasi Gauss.....	23
46.	Metode eliminasi Gauss dengan strategi pivoting parsial	23
47.	Metode eliminasi Gauss dengan strategi pivoting skala parsial	24
48.	Metode eliminasi Gauss dengan strategi pivoting total	25
49.	Metode dekomposisi LU tanpa pivoting	26
50.	Metode dekomposisi LU dengan pivoting.....	26
51.	Metode Newton sistem persamaan taklinear kriteria stopping 1	27
52.	Metode Newton sistem persamaan taklinear kriteria stopping 2	27
53.	Metode Newton dengan Jacobian diaproksimasi	28
54.	Metode Chord standar.....	29
55.	Metode Chord modifikasi	29
56.	Metode Broyden	30
57.	Metode penurunan tercuram (steepest descent)	31
58.	Metode Euler standar	32
59.	Metode Euler selisih mundur	32
60.	Metode Euler perbaikan	33
61.	Metode Runge-Kutta order 4	33

62.	Metode Runge-Kutta order 4 untuk sistem	34
63.	Metode Runge-Kutta order 4 untuk persamaan tingkat tinggi	34
64.	Metode Adam-Basfort order 4	35
65.	Metode Adam-Basfort order 4 dengan memori	35
66.	Metode prediktor-korektor	36
67.	Metode shooting linear	37
68.	Metode shooting taklinear dengan secant	37
69.	Metode shooting taklinear dengan Newton	37
70.	Metode selisih hingga untuk MSB linear	39
71.	Metode selisih hingga untuk MSB taklinear.....	39
72.	Metode Ritz.....	41
73.	Metode residu terbobot	42
74.	Fungsi basis b-spline kubik.....	43
75.	Metode kolokasi titik.....	43
76.	Metode golden-section untuk optimasi.....	44
77.	Metode parabolik berjenjang untuk optimasi	45
78.	Metode parabolik golden-section untuk optimasi	45
79.	Metode Fibonacci untuk optimasi	46
80.	Metode Newton untuk optimasi	47
81.	Metode Secant untuk optimasi	47
82.	Metode penurunan tercuram (steepest-descent) untuk optimasi	48
83.	Metode Newton untuk optimasi multivariabel.....	48
84.	Metode konjugat gradien untuk optimasi multivariabel	49
85.	Metode kuasi-Newton untuk optimasi multivariabel.....	50
86.	Metode BFGS untuk optimasi multivariabel	50
87.	Metode simpleks untuk program linear	51

1. Membangkitkan galat aproksimasi faktorial dengan formula Stirling

```
n = 15;
eks = zeros(n,1); ap=zeros(n,1);
err_mutlak=zeros(n,1); err_rel=zeros(n,1);
for k = 1:n
    eks(k) = factorial(k);
    ap(k) = sqrt(2*pi*k)*(k/exp(1))^k;
    err_mutlak(k)=abs(ap(k)-eks(k));
    err_rel(k)=err_mutlak(k)/eks(k);
end
[eks ap err_mutlak err_rel]
```

2. Melihat dampak galat pembulatan komputer pada aproksimasi bilangan e

```
m = 17;
ap = zeros(m,1);
err = zeros(m,1);
for k = 1: m
    n = 10^k;
    ap(k) = (1+1/n)^n;
    err(k)=abs(ap(k)-exp(1));
end
[ap err]
```

3. Aproksimasi nilai e dengan deret Taylor

```
m = 17;
ap = zeros(m,1);
err = zeros(m,1);
ap(1) = 1+1;
err(1)=abs(exp(1)-ap(1));
for k = 2:m
    ap(k)=ap(k-1)+1/factorial(k);
    err(k)=abs(ap(k)-exp(1));
end
[ap err]
```

4. Aproksimasi limit mendekati nol

```
m = 16;
ap = zeros(m,1);
```

```

err = zeros(m,1);
for k = 1: m
n = 10^(-k);
ap(k) = (exp(n)-1)/n;
err(k)= abs(ap(k)-1);
end
[ap err]

```

5. Metode bagidua dengan kriteria stopping logaritma

```

function [akar,langkah]=bagidua1(a,b,tol)
%INPUT : titik ujung interval a, b, toleransi = tol
%OUTPUT : akar : barisan akar aproksimasi
% langkah : banyaknya langkah yang dibutuhkan.
%cek adanya akar di dalam [a,b]
if feval('fun',a)*feval('fun',b)>0
    error('akarnya tidak terjamin ada dalam interval ini')
end
langkah = 0; akar = [];
N=ceil(log2((b-a)/tol));%banyak iterasi minimal yang dibutuhkan
%loop (iterasi):
for k=1:N
    langkah = langkah+1;
    p = (a+b)/2;
    akar=[akar;p];
fa = feval('fun',a); fp = feval('fun',p);
if sign(fa)*sign(fp)<0
    a = a; b = p;
else
    a = p; b = b;
end
end
%fungsi f yang diberikan pada f(x)=0
function y = fun(x)
c=3; y = x-exp(-x/c);

```

6. Metode bagidua dengan kriteria stopping nilai mutlak fungsi

```

function [akar,langkah]=bagidua2(a,b,tol)
%cek adanya akar di dalam [a,b]
if feval('fun',a)*feval('fun',b)>0
error('akarnya tidak terjamin ada dalam interval ini')
end
langkah = 0; akar = [];
p = (a+b)/2; fp = feval('fun',p);
while abs(fp) > tol

```

```

langkah = langkah+1;
p = (a+b)/2;
akar=[akar;p];
fa = feval('fun',a); fp = feval('fun',p);
if sign(fa)*sign(fp)<0
a = a; b = p;
else
a = p; b = b;
end
end
%fungsi yang diberikan
function y = fun(x)
c=3; y = x-exp(-x/c);

```

7. Metode secant

```

function [akar,langkah]=secant3(a,b,tol)
p_1 = a; p0 = b;
f_1 = feval('fun',p_1); f0 = feval('fun',p0);
langkah = 0; akar=[];
while abs(p_1-p0) > tol
    langkah = langkah+1;
    p = p0-f0*(p0-p_1)/(f0-f_1);
    akar = [akar;p];
    p_1 = p0; p0 = p;
    f_1 = feval('fun',p_1);
    f0 = feval('fun',p0);
end
%fungsi yang diberikan
function y = fun(x)
%c=3; y = x-exp(-x/c);
y=1000*x*exp(x)-1025*x-435;

```

8. Metode false-position

```

function [akar,langkah]=falsepos3(a,b,tol)
fa = feval('fun',a); fb = feval('fun',b);
p = a-fa*(b-a)/(fb-fa); fp = feval('fun',p);
langkah = 1; akar = [p];
p_1=a;p0=b;
while abs(p_1-p0) > tol
    langkah = langkah+1;
    if sign(fa)*sign(fp)<0
        a = a; b = p;
    else

```



```

        a = p; b = b;
    end
    fa = feval('fun',a); fb = feval('fun',b);
    p = a-fa*(b-a)/(fb-fa);
    akar = [akar;p];
    p_1=p0; p0=p; fp = feval('fun',p);
end
%fungsi yang diberikan
function y = fun(x)
%y = (x^3-4*x^2+x+1)*sin(3*x);
y=1000*x*exp(x)-1025*x-435;
%y=tan(pi*x)-6;

```

9. Metode Newton persamaan taklinear satu variabel

```

function [akar,langkah]=newton3(p0,tol)
%INPUT: iterasi awal p0 dan toleransi tol
%OUTPUT: akar:barisan aproksimasi, langkah: banyak langkah
fp = feval('fun',p0); dfp = feval('dfun',p0);
p = p0-fp/dfp;
akar =[p0;p]; langkah = 1;
while abs(p-p0)>tol
    langkah = langkah+1;
    p0=p;
    fp = feval('fun',p0);dfp = feval('dfun',p0);
    p = p0-fp/dfp;
    akar = [akar;p];
end
%fungsi yang diketahui
function y = fun(x)
y=(4/3)*exp(2-x/2).*(1+log(x)./x);
%derivatifnya
function dy = dfun(x)
dy = (4/3)*(-0.5*exp(2-x/2)*(1+log(x)/x)+exp(2-x/2)*((1-
log(x))/x^2));

```

10. Metode titik-tetap (fixed point)

```

function [x,langkah]=titiktetap3(x0,tol)
x1=feval('fun_iter',x0);
x=[x0;x1];langkah=1;
while abs(x0-x1)>tol
    langkah = langkah+1;
    xp=feval('fun_iter',x1);
    x=[x;xp];
    x0=x1;x1=xp;

```

```

end
%fungsi iterasi
function y=fun_iter(x);
y=(x+exp(-x))/2;

```

11. Konstruksi polinomial Lagrange

```

function y = lagr1(x,xdata,k)
%mendefinsikan fungsi Lagrange ke-k dengan xdata sbg node.
n=length(xdata);
%cek indeks
if k>n
    error('indeks k melebihi banyak data')
end
xx=xdata;xx(k)=[];%membuang komponen ke k xdata
d=prod(xdata(k)-xx); %suku penyebut pada definisi polinomial
Lagrange
y = 1;
for k=1:n-1;
    ya=(x-xx(k));
    y=y.*ya;
end
y=y/d;

```

12. Konstruksi polinomial interpolasi dengan metode Lagrange

```

function y = poli_inter1(x,xdata,ydata)
if length(xdata)~=length(ydata)
    error('panjang xdata dan ydata harus sama')
end
n = length(xdata);
y = 0;
for k = 1:n;
    yp = ydata(k)*lagr1(x,xdata,k);
    y = y + yp;
end

```

13. Koefisien polinomial interpolasi dengan formula Newton

```

function a=newton_div(xdata,ydata)
x=xdata;y=ydata;n=length(x)-1;
a=zeros(n+1,1);
a(1)=y(1);
for k = 2:n+1
    w=1;p=0;
    for j = 1:k-1

```

```

        p = p+a(j)*w;
        w = w*(x(k)-x(j));
    end
a(k)=(y(k)-p)/w;
end

```

14. Membangun polinomial interpolasi dengan formula Newton

```

function y=newton_poly(x,xdata,ydata)
%panggil dulu koefisiennya
a=newton_div(xdata,ydata);
n=length(xdata);
y=a(1); yy=1;
for k=2:n;
    p=(x-xdata(k-1));
    yy=yy.*p;
    y=y+a(k)*yy;
end

```

15. Membangun matriks hasil terbagi Newton

```

function D = hbagi_tabel(xdata,ydata)
%menghasilkan semua nilai hasil terbagi dlm bentuk matriks
n=length(xdata);
D=zeros(n,n);
D(:,1)=ydata';
for j=2:n
    v=D(j-1:end,j-1);
    a=v(2:end)-v(1:end-1);
    b=xdata(j:end)-xdata(1:end-j+1);
    b=b'; d=a./b;
    D(j:end,j)=d;
end

```

16. Membangun polinomial interpolasi melalui data tabel

```

function y = poli_tabel(x,xdata,ydata)
D = hbagi_tabel(xdata,ydata);
n=length(xdata);
y = D(1,1);
yy=1;
for k=2:n;
    p=(x-xdata(k-1));
    yy=yy.*p;
    y=y+D(k,k)*yy;
end

```

17. Membangun matriks hasil bagi Hermite

```
function D = hbagi_hermite(xdata,ydata,dydata)
%input: pasangan data (xdata,ydata) dan (xdata,dydata)
n=length(xdata);
zdata=zeros(1,2*n);
zdata(1:2:end-1)=xdata;
zdata(2:2:end)=xdata;
zydata=zeros(1,2*n);
zydata(1:2:end-1)=ydata;
zydata(2:2:end)=ydata;
D=zeros(2*n,2*n);
D(:,1)=zydata';
D(2:2:end,2)=dydata';
dy=ydata(2:end)-ydata(1:end-1);
dx=xdata(2:end)-xdata(1:end-1);
d=dy./dx;
D(3:2:end-1,2)=d';
for j=3:2*n
    v=D(j-1:end,j-1);
    a=v(2:end)-v(1:end-1);
    b=zdata(j:end)-zdata(1:end-j+1);b=b';
    d=a./b;
    D(j:end,j)=d;
end
```

18. Membangun polinomial interpolasi Hermite

```
function y = poli_hermite(x,xdata,ydata,dydata)
%mendefinisikan polinomial Hermite
D = hbagi_hermite(xdata,ydata,dydata);
n=length(xdata);
zdata=zeros(1,2*n);
zdata(1:2:end-1)=xdata;
zdata(2:2:end)=xdata;
y = D(1,1); yy=1;
for k=2:2*n;
    p=(x-zdata(k-1));
    yy=yy.*p;
    y=y+D(k,k)*yy;
end
```

19. Membangun polinomial sepotong-sepotong

```

function p = poli_potong(x, xdata, ydata, m, d)
%cek kecocokan banyak data
n=length(xdata)-1;
if n~=m*d
error('banyak xdata harus sama dengan md+1')
end
Qd=zeros(m, length(x));
for k = 1:m
xkdata=xdata(d*k-d+1:d*k+1);
ykdata=ydata(d*k-d+1:d*k+1);
D = hbagi_tabel(xkdata, ykdata);
y = poli_tabel(x, xkdata, ykdata);
Qd(k, :)=y;
end
p=0;
for k=1:m-1
p0= Qd(k, :).*(x>=xdata(d*k-d+1)&x<xdata(d*k+1));
p=p+p0;
end
sm=Qd(m, :).*(x>=xdata(d*m-d+1)&x<=xdata((d*m+1)));
p = p+sm;

```

20. Membangun polinomial spline kubik

```

function [koef, s]=spline_kubik(x, xdata, ydata, dy0, dyn)
n=length(xdata)-1;
h = xdata(2:end)-xdata(1:end-1);
a = ydata;
%pendefinisikan SPL Mc=a
M=zeros(n+1);
d1=[h(1:n)];
d0=[2*h(1) 2*(h(1:n-1)+h(2:n)) 2*h(n)];
d_1=h(1:n); M=diag(d_1, -1)+diag(d0, 0)+diag(d1, 1);
alpha=zeros(n+1, 1);
alpha(1)=(3/h(1))*(a(2)-a(1))-3*dy0;
for k=2:n
    alpha(k)=(3/h(k))*(a(k+1)-a(k))-(3/h(k-1))*(a(k)-a(k-1));
end
alpha(n+1)=3*dyn-(3/h(n))*(a(n+1)-a(n));
c = M\alpha;%koefisien potongan polinomial
%menghitung koefisien b_k dan d_k
b=zeros(n, 1);
for k = 1:n
    b(k)=(1/h(k))*(a(k+1)-a(k))-(h(k)/3)*(2*c(k)+c(k+1));
end
d=zeros(n, 1);
for k=1:n

```

```

        d(k)=(1/(3*h(k)))*(c(k+1)-c(k));
end
%mendefinisikan potongan polinomial S_k
sk=zeros(n,length(x));
for k=1:n
    sk(k,:)=a(k)+b(k)*(x-xdata(k))+c(k)*(x-
        xdata(k)).^2+d(k)*(x-xdata(k)).^3;
end
s=0;
for k=1:n-1
    s0=sk(k,:).*(x>=xdata(k)&x<xdata(k+1));
    s = s+s0;
end
s0=sk(n,:).*(x>=xdata(n)&x<=xdata(n+1));
s = s+s0;
koef=[a(1:end-1)' b c(1:end-1) d];

```

21. Membangun polinomial B-spline derajat nol sampai derajat 4

```

function p=bs0(x,xdata,i)
n=length(xdata);
if (i<=0||i>=n);
    error('indeks i sudah keluar rentang yang ada')
end
p=1.*(x>=xdata(i)&x<xdata(i+1));

```

```

function p=bs1(x,xdata,i)
n=length(xdata);
if (i<=0||i>=n-1);
    error('indeks i sudah keluar rentang yang ada')
end
k=1;
v1=(x-xdata(i))/(xdata(i+k)-xdata(i));
v2=(x-xdata(i+1))/(xdata(i+1+k)-xdata(i+1));
p = v1.*bs0(x,xdata,i)+(1-v2).*bs0(x,xdata,i+1);

```

```

function p=bs2(x,xdata,i)
n=length(xdata);
if (i<=0||i>=n-2);
    error('indeks i sudah keluar rentang yang ada')
end
k=2;
v1=(x-xdata(i))/(xdata(i+k)-xdata(i));
v2=(x-xdata(i+1))/(xdata(i+1+k)-xdata(i+1));
p = v1.*bs1(x,xdata,i)+(1-v2).*bs1(x,xdata,i+1);

```

```

function p=bs3(x,xdata,i)

```

```

n=length(xdata);
if (i<=0||i>=n-3);
    error('indeks i sudah keluar rentang yang ada')
end
k=3;
v1=(x-xdata(i))/(xdata(i+k)-xdata(i));
v2=(x-xdata(i+1))/(xdata(i+1+k)-xdata(i+1));
p = v1.*bs2(x,xdata,i)+(1-v2).*bs2(x,xdata,i+1);

```

```

function p=bs4(x,xdata,i)
n=length(xdata);
if (i<=0||i>=n-4);
    error('indeks i sudah keluar rentang yang ada')
end
k=4;
v1=(x-xdata(i))/(xdata(i+k)-xdata(i));
v2=(x-xdata(i+1))/(xdata(i+1+k)-xdata(i+1));
p = v1.*bs3(x,xdata,i)+(1-v2).*bs3(x,xdata,i+1);

```

22. Membangun interpolasi B-spline

```

function b=bspline_interp(x,leftnode,node,rightnode,ydata,dfa,dfb)
%input: x: domain di mana fungsi didefinisikan, node:node
interior,
% leftnode dan rightnode: masing-masing node tambahan di a dan b
% ydata: nilai f di titik-titik node, dfa=f'(a), dfb=f'(b).
n = length(node)-1; xdata=[leftnode node rightnode];
%mendefinisikan matriks M
M=zeros(n+1,n+3);
for k=1:n+3
    M(:,k)=bs3(node,xdata,k);
end
%aproksimasi derivatif basis di a dan b
v0=zeros(1,n+3); h=0.0001;
for k=1:3
    v0(k)=(bs3(node(1)+h,xdata,k)-bs3(node(1)- . . .
h,xdata,k))/(2*h);
end
v1=zeros(1,n+3);
for k=n+3:-1:n+1
    v1(k)=(bs3(node(end)+h,xdata,k)-bs3(node(end)- . . .
h,xdata,k))/(2*h);
end
M1=[v0;M;v1];
%mendefinisikan vektor f1
f1=[dfa;ydata';dfb];
%menghitung koefisien c_k

```

```

c=M1\f1;
%mendefinisikan kombinasi linear
b=0;
for k=1:n+3;
    b0=c(k)*bs3(x,xdata,k);
b=b+b0;
end

```

23. Membangun spline kardinal orde 0 sampai 4

```

function phi =spline_kard0(x)
phi = 1.*(x>=0&x<1);

```

```

function phi =spline_kard1(x)
phi=x.*feval('spline_kard0',x)+(2-x).*feval('spline_kard0',x-1);

```

```

function phi =spline_kard2(x)
phi=(x/2).*feval('spline_kard1',x)+(3-x)/2.*feval('spline_kard1',x-1);

```

```

function phi =spline_kard3(x)
phi=(x/3).*feval('spline_kard2',x)+(4-x)/3.*feval('spline_kard2',x-1);

```

```

function phi =spline_kard4(x)
phi=(x/4).*feval('spline_kard3',x)+(5-x)/4.*feval('spline_kard3',x-1);

```

24. Membangun derivatif pertama spline kardinal orde 1 sampai 4

```

function dphi=derkard2(x)
dphi=feval('spline_kard1',x)-feval('spline_kard1',x-1);

```

```

function dphi=derkard3(x)
dphi=feval('spline_kard2',x)-feval('spline_kard2',x-1);

```

```

function dphi=derkard4(x)
dphi=feval('spline_kard3',x)-feval('spline_kard3',x-1);

```

25. Membangun interpolasi spline kardinal

```

function [b,node,ydata]=spline_kardinal3_interp(x,a,b,n)
%input: x: domain di mana fungsi didefinisikan,
% ydata: nilai f di titik-titik node, dfa=f'(a), dfb=f'(b).
%output: b: nilai polinomial interpol pd x, node dan pasangannya
(ydata).
h=(b-a)/n;

```



```

node=a:h:a+n*h;
ydata=fun(node);
xdata=[a-3*h a-2*h a-h node];
dfa=dfun(a);dfb=dfun(b);
%mendefinisikan matriks M
M=zeros(n+3,n+3);
M=diag(ones(n+2,1),-
1)+diag(4*ones(n+3,1),0)+diag(ones(n+2,1),1);
M(1,1)=-1;M(1,2)=0;M(1,3)=1;
M(end,end-2)=-1;M(end,end-1)=0;M(end,end)=1;
%mendefinisikan vektor ruas kanan
f=zeros(n+3,1);
f=6*[(h/3)*dfa ydata (h/3)*dfb]';
%menghitung koefisien c
c=M\f;
%membangun kombinasi linear
b=0;
for k=-3:n-1;
    b0=c(k+4)*spline_kard3((x-xdata(k+4))/h);
b=b+b0;
end
%fungsi yang aproksimasi dan derivatifnya
function y =fun(x)
y=1./(1+x.^2);
function dy =dfun(x)
dy=(-2*x)./(1+x.^2).^2;

```

26. Aproksimasi derivatif pertama dengan tiga formula selisih hingg

```

function [dyf,dyb,dyc]=derfun0(x0,h)
dyf=(feval('fun',x0+h)-feval('fun',x0))/h;%maju
dyb=(feval('fun',x0)-feval('fun',x0-h))/h;%mundur
dyc=(feval('fun',x0+h)-feval('fun',x0-h))/(2*h);%terpusat
%fungsi yang didiferensialkan
function y=fun(x)
y=exp(-x.^2);

```

27. Aproksimasi derivatif pertama fungsi terdiferensial di mana-mana

```

function[x,dermun,dermaj,derpus]=derfun1(a,b,h)
x=a:h:b;
n=length(x);
for k=1:n
    dermun(k)=(feval('fun',x(k))-feval('fun',x(k)-h))/h;
    dermaj(k)=(feval('fun',x(k)+h)-feval('fun',x(k)))/h;

```

```

        derpus(k)=(feval('fun',x(k)+h)-feval('fun',x(k)-h))/(2*h);
end
%fungsi yang didiferensialkan
function y=fun(x)
y=exp(-x.^2);

```

28. Aproksimasi derivatif pertama fungsi pada array tertentu

```

function[x,der]=derfun2(xdata)
x=xdata;
n=length(x);
der=zeros(1,n);%array untuk derivatif
h1=x(2)-x(1);hn=x(n)-x(n-1);%mesh pertama dan terakhir
der(1)=(feval('fun',x(1)+h1)-feval('fun',x(1)))/h1;%maju
der(n)=(feval('fun',x(n))-feval('fun',x(n)-hn))/hn;%mundur
for k=2:n-1 %indeks titik interior
    h1=x(k)-x(k-1);h2=x(k+1)-x(k);
    der(k)=(feval('fun',x(k)+h2)-feval('fun',x(k)-h1))/((h1+h2));
end
%fungsi yang didiferensialkan
function y=fun(x)
y=exp(-x.^2);

```

29. Aproksimasi derivatif pertama fungsi dengan metode operasi array

```

function[x,dermun,dermaj,derpus]=derfun11(a,b,h)
x=a:h:b;
n=length(x);
xx=a-h:h:b+h;
dermun=(feval('fun',x)-feval('fun',xx(1:end-2)))/h;
dermaj=(feval('fun',xx(3:end))-feval('fun',x))/h;
derpus=(feval('fun',xx(3:end))-feval('fun',xx(1:end-2)))/(2*h);
%fungsi yang didiferensialkan
function y=fun(x)
y=exp(-x.^2);

```

30. Aproksimasi derivatif pertama fungsi melalui data tabular

```

function [x derf]=dertab(xdata,ydata)
x=xdata;y=ydata;
n=length(x);
h=x(2:end)-x(1:end-1);
derf(1)=(y(2)-y(1))/h(1);derf(n)=(y(end)-y(end-1))/h(n-1);
for k=2:n-1

```

```

    derf(k)=(y(k+1)-y(k-1))/(h(k-1)+h(k));
end

```

31. Mendeteksi sensitivitas aproksimasi dengan formula selisih terbagi

```

function [h,r,err]=tesderivatif(M)
h=1;a=sqrt(2);
F1=feval('fun3',a);
h=zeros(M,1);h(1)=1;
r=zeros(M,1);
for k=1:M
    F2=feval('fun3',a+h);
    d=F2-F1;
    r(k)=d(k)/h(k);
    h(k+1)=h(k)/2;
end
h(end)=[];
err=abs(r-1/3);%pendefinisian galat
[(1:M)',h,r,err] %menampilkan data keluaran
%fungsi yg didiferensialkan
function y=fun3(x)
y=atan(x);

```

32. Aproksimasi derivatif dengan metode Richardson

```

function der = richardson1(x0,M)
%input x0: titik dimana akan dihitung f'(x0),
        %M: banyak step
%output der:matriks segitiga kumpulan aproksimasi derivatif
h = (1/2).^(0:M); %mendefinisikan array mesh
der = zeros(M+1,M+1); %persiapan matriks segitiga
%inisialisasi kolom pertama matriks
for n = 1:M+1
    der(n,1)= feval(@center,x0,h(n));
end
for k = 2:M+1
    for n = k:M+1
        der(n,k)=der(n,k-1)+(der(n,k-1)-der(n-1,k-1))/(4^(k-1)-1);
    end
end
%formula selisih terpusat
function phi = center(x,h)
phi = (feval(@atan,x+h)-feval(@atan,x-h))/(2*h);

```

33. Metode kuadatur dasar

```
function [M,T,S]=kuad_dasar(a,b)
% input: batas integrasi a dan b
% output: ketiga metoda kuadratur dasar, M untuk midpoint,
% T untuk trapesium dan S untuk Simpson.
M = (b-a)*feval('fun',(a+b)/2);
T = (b-a)/2*(feval('fun',a)+feval('fun',b));
S = (b-a)/6*(feval('fun',a)+4*feval('fun',(a+b)/2)+feval('fun',b));
%fungsi yang diintegralkan
function y = fun(x)
y = x*exp(x^2);
```

34. Simulasi galat metode kuadatur dasar

```
function v=eksp_kdasar(a,b)
% input: batas integrasi a dan b
% output: vektor v seperti di atas
M = (b-a)*feval('fun',(a+b)/2);
T = (b-a)/2*(feval('fun',a)+feval('fun',b));
S = (b-a)/6*(feval('fun',a)+4*feval('fun',(a+b)/2)+feval('fun',b));
eksak = feval('fun1',b)-feval('fun1',a);
v = [eksak M abs(M-eksak) T abs(T-eksak) S abs(S-eksak)];
%fungsi yang diintegralkan
function y = fun(x)
y = x*exp(x^2);
%fungsi hasil integral
function y = fun1(x)
y = 0.5*exp(x^2);
```

35. Metode midpoint bersusun

```
function M=midpoint(a,b,n)
%masukkan n genap
h = (b-a)/n;
x = a:h:b;%mendefinisikan partisi seragam
if(rem(n,2)~=0)%cek n genap.
    error('Anda memasukkan nilai n ganjil, seharusnya genap')
end
node_ganjil = x(2:2:end-1);%mengumpulkan node dengan indeks ganjil.
M = 2*h*sum(feval('fun',node_ganjil));
%fungsi yang diintegralkan
function y = fun(x)
y = x.*exp(x.^2);
```

36. Metode Simpson bersusun

```

function S=simpson(a,b,n)
%masukkan n genap
h = (b-a)/n;
x = a:h:b;
if(rem(n,2)~=0)
    error('Anda memasukkan nilai n ganjil, seharusnya genap')
end
node_ganjil = x(2:2:end-1);
node_genap = x(3:2:end-2);
S = (h/3)*(feval('fun',a)+feval('fun',b)+...
4*sum(feval('fun',node_ganjil))+2*sum(feval('fun',node_genap)));
%fungsi yang diintegalkan
function y = fun(x)
y = x.*exp(x.^2);

```

37. Metode trapesium bersusun

```

function T=trapes(a,b,n)
%masukkan n bulat positif (boleh genap atau ganjil)
h = (b-a)/n;
x = a:h:b;
node_interior = x(2:end-1);
T = (h/2)*(feval('fun',a)+feval('fun',b)+...
2*sum(feval('fun',node_interior)));
%fungsi integral
function y = fun(x)
y = x.*exp(x.^2);

```

38. Integral data tabular dengan metode trapesium

```

function T1 = int_tab(datax,datay)
%input: datax adalah array para x_k, datay adalah array para y_k
a1=datax(2:end);a2=datax(1:end-1);h=a1-a2; %mendefinisikan array h
b1=datay(1:end-1);b2=datay(2:end);yy=b1+b2; % mendefinisikan array yy
T1 = sum((h/2).*yy);

```

39. Implementasi integral Gauss order 4

```

function q = gauss4(a,b)
%input: a dan b sebagai batas integral
%output: q nilai aproksimasi dengan integrasi Gauss
c = [0.8611363116, 0.3399810436, -0.3399810436, -0.8611363116];%absis
t = [0.3478548451, 0.6521451549, 0.6521451549, 0.3478548451];%bobot
x = ((b-a)*t+b+a)/2; % absis hasil transformasi
q = (b-a)/2*sum(c.*feval('fun',x));
%fungsi yang diintegalkan

```

```
function y = fun(x);
y = x.*exp(x.^2);
```

40. Implementasi integral Gauss bersusun order 4

```
function q = gauss4_bersusun(a,b,N)
%input: a dan b batas integral dan N banyak subinterval
%output: q nilai aproksimasi dengan integrasi Gauss bersusun
h = (b-a)/N;
xx = a:h:b; % partisi domain
t = [0.8611363116, 0.3399810436, -0.3399810436, -
0.8611363116];%absis
c = [0.3478548451, 0.6521451549, 0.6521451549,
0.3478548451];%bobot
q = 0;
for k = 2: N+1;
    x = (h*(t+1)+2*xx(k-1))/2; % absis hasil transformasi
    q0 = (h/2)*sum(c.*feval('fun',x));
    q = q + q0;
end
%fungsi yang diintegalkan
function y = fun(x);
y = x.*exp(x.^2);
```

41. Metode integral Romberg

```
function R = romberg_int(a,b,M)
%input: batas integral dan level aproksimasi M
%output: matriks segitiga bawah aproksimasi Romberg
R = zeros(M+1,M+1); %matriks untuk menampung nilai aproksimasi
h = b-a; %mesh awal
R(1,1)=0.5*(b-a)*(feval('fun',a)+feval('fun',b));
for n = 1:M
    h = h/2;%penghalusan mesh
    i = 1:2^(n-1);%indeks node
    nodes = a+(2*i-1)*h;%node yang digunakan
    fnode = sum(feval('fun',nodes));
    R(n+1,1)=0.5*R(n,1)+h*fnode;
end
for m = 1:M
    for n=m:M
        R(n+1,m+1)=R(n+1,m) + (R(n+1,m) -R(n,m)) / (4^m-1);
    end
end
%integrand
function y = fun(x)
y = x.*exp(x.^2);
```

42. Metode substitusi mundur untuk menyelesaikan SPL

```
function x=subt_mundur(A,b)
%input: matriks A dan vektor b
%output: vektor penyelesaian x
n = length(b);%ukuran sistem
x=zeros(n,1);%menyiapkan vektor penyelesaian
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
    sum=0;
    for j=k+1:n
        sum=sum+A(k,j)*x(j);
    end
    x(k)=(b(k)-sum)/A(k,k);
end
```

43. Metode substitusi maju untuk menyelesaikan SPL

```
function x=subt_maju(A,b)
%input: matriks A dan vektor b
%output: vektor penyelesaian x
n = length(b);%ukuran sistem
x=zeros(n,1);%menyiapkan vektor penyelesaian
x(1)=b(1)/A(1,1);
for k=2:n
    sum=0;
    for j=1:k-1
        sum=sum+A(k,j)*x(j);
    end
    x(k)=(b(k)-sum)/A(k,k);
end
```

44. Membentuk SPL ke dalam bentuk segitiga atas

```
function [A1,b1]=spl_segitiga_atas(A,b)
A1=A;b1=b;
n=length(b1);
for i=1:n-1
    for j=i+1:n
        if A1(i,i)==0
            error('metode eliminasi gagal')
        end
        lambda=A1(j,i)/A1(i,i);
        for k=i:n
            A1(j,k)=A1(j,k)-lambda*A1(i,k)
        end
        b1(j)=b1(j)-lambda*b1(i)
    end
end
```

```
end
end
```

45. Metode eliminasi Gauss

```
function x=gaussian1(A,b)
[A1,b1]=spl_segitiga_atas(A,b);%menjadikan bentuk segitiga atas
x=subt_mundur(A1,b1);%menerapkan substitusi mundur
```

46. Metode eliminasi Gauss dengan strategi pivoting parsial

```
function x=parsial_pivoting(A,b)
n=length(b);
%membentuk sistem segitiga atas
for i=1:n-1
%pertukaran baris untuk memilih pivot terbesar
A1=A; b1=b;
    for k=i+1:n
        p=i;
        if abs(A(p,i))<abs(A(k,i))
            p=k;
        else
            p=i;
        end
    end
    A1(i,:)=A(p,:);A1(p,:)=A(i,:);
    b1(i)=b(p);b1(p)=b(i);
%eliminasi per kolom
    for j=i+1:n
        lambda=A1(j,i)/A1(i,i);
        for k=i:n
            A1(j,k)=A1(j,k)-lambda*A1(i,k);
        end
    end
    b1(j)=b1(j)-lambda*b1(i);
end
A=A1;b=b1;
end
%menjalankan substitusi mundur
x=zeros(n,1);%menyiapkan vektor penyelesaian
x(n)=b1(n)/A1(n,n);
for k=n-1:-1:1
    sum=0;
    for j=k+1:n
        sum=sum+A1(k,j)*x(j);
    end
    x(k)=(b1(k)-sum)/A1(k,k);
end
```


end

47. Metode eliminasi Gauss dengan strategi pivoting skala parsial

```
function x=skala_parsial_pivoting(A,b)
n=length(b);
s=zeros(n,1);
%faktor skala
for k=1:n
    s(k)=max(abs(A(k,:)));
end
%membentuk sistem segitiga atas
for i=1:n-1
    %pertukaran baris untuk memilih pivot terbesar setelah diskala
    A1=A; b1=b;
    for k=i+1:n
        p=i;
        if abs(A(p,i))/s(i)<abs(A(k,i))/s(k)
            p=k;
        else
            p=i;
        end
    end
    A1(i,:)=A(p,:);A1(p,:)=A(i,:);
    b1(i)=b(p);b1(p)=b(i);
    %eliminasi per kolom
    for j=i+1:n
        lambda=A1(j,i)/A1(i,i);
        for k=i:n
            A1(j,k)=A1(j,k)-lambda*A1(i,k);
        end
        b1(j)=b1(j)-lambda*b1(i);
    end
    A=A1;b=b1;
end
%menjalankan substitusi mundur
x=zeros(n,1);%menyiapkan vektor penyelesaian
x(n)=b1(n)/A1(n,n);
for k=n-1:-1:1
    sum=0;
    for j=k+1:n
        sum=sum+A1(k,j)*x(j);
    end
    x(k)=(b1(k)-sum)/A1(k,k);
end
```

48. Metode eliminasi Gauss dengan strategi pivoting total

```
function x=total_pivoting(A,b)
n=length(b);
mutan=zeros(1,n-1);%menyimpan data kolom yang dipertukarkan.
for i=1:n-1
    d=max(abs(A(i:end,i:end)));%magnitudo terbesar setiap kolom
    pp=find(d==max(d));
    p=pp(1)+i-1;
    mutan(i)=p;
    AA=A;
    AA(:,i)=A(:,p);AA(:,p)=A(:,i);%pertukaran kolom;
    %pertukaran baris untuk memilih pivot terbesar
    t=find(abs(AA(i:end,i))==max(abs(AA(i:end,i)))));
    p1=t(1)+i-1;
    A2=AA; b1=b;
    A2(i,:)=AA(p1,:);A2(p1,:)=AA(i,:);
    b1(i)=b(p1);b1(p1)=b(i);
    %eliminasi per kolom
    for j=i+1:n
        lambda=A2(j,i)/A2(i,i);
        for k=i:n
            A2(j,k)=A2(j,k)-lambda*A2(i,k);
        end
        b1(j)=b1(j)-lambda*b1(i);
    end
    A=A2;b=b1;
end
%menjalankan substitusi mundur
z=zeros(n,1);%menyiapkan vektor penyelesaian
z(n)=b1(n)/A2(n,n);
for k=n-1:-1:1
    sum=0;
    for j=k+1:n
        sum=sum+A2(k,j)*z(j);
    end
    z(k)=(b1(k)-sum)/A2(k,k);
end
%transformasi dengan matriks permutasi
P=eye(n);
for k=1:n-1
    P1=matriks_perm(n,k,mutan(k));
    P=P*P1;
end
x = P*z;
```

49. Metode dekomposisi LU tanpa pivoting

```
function x=dekomp_lu(A,b)
n=length(A);
%membangun matriks L dan U
L=eye(n);U=eye(n);
for k=1:n-1
    R=zeros(n);
    R(k+1:n,k)=A(k+1:n,k)/A(k,k);
    M=eye(n)-R;
    U=M*A;
    L=L+R;
    A=U;
end
%menyelesaikan Ly=b dengan subtitusi maju
y=subt_maju(L,b);
%menyelesaikan Ux=y dengan subtitusi mundur
x=subt_mundur(U,y);
```

50. Metode dekomposisi LU dengan pivoting

```
function x=lu_pivot(A,b)
n=length(A); A
A=A; bb=b;
m=[];%indeks hasil mutasi
for i=1:n-1
    %pertukaran baris untuk memilih pivot terbesar
    t=find(abs(A(i:end,i))==max(abs(A(i:end,i)))));
    p=t(1)+i-1;
    A1=A; b1=b;
    A1(i,:)=A(p,:);A1(p,:)=A(i,:);
    b1(i)=b(p);b1(p)=b(i);
    m=[m;p];
    %eliminasi per kolom
    for j=i+1:n
        lambda=A1(j,i)/A1(i,i);
        for k=i:n
            A1(j,k)=A1(j,k)-lambda*A1(i,k);
        end
        b1(j)=b1(j)-lambda*b1(i);
    end
    A=A1;b=b1;
end
%matriks dan vektor hasil mutasi
A1=AA;b1=bb;%original
```

```

for k=1:n-1
    I=eye(n);
    P=I;
    P(k,:)=I(m(k),:);P(m(k),:)=I(k,:);
    A1=P*A1;b1=P*b1;
end
x=dekomp_lu(A1,b1);

```

51. Metode Newton sistem persamaan taklinear kriteria stopping 1

```

function [akar,langkah]=newton_sistem(p0,tol)
fp=feval('fun',p0); jf=feval('dfun',p0);
if det(jf)==0
    error('metode Newton gagal!')
end
akar=[p0]; langkah=0;
while norm(fp)>tol
    langkah=langkah+1;
    q0=jf\fp;
    p=p0-q0;
    akar=[akar p];
    p0=p;
    fp=feval('fun',p0);
    jf=feval('dfun',p0);
    if det(jf)==0
        error('metode Newton gagal!')
    end
end
%fungsi yang diketahui
function y=fun(x)
y=zeros(2,1);
y(1)=2*x(1)-x(2)+(1/9)*exp(-x(1))-1;
y(2)=-x(1)+2*x(2)+(1/9)*exp(-x(2));
%Jacobiannya
function dy=dfun(x)
dy=zeros(2,2);
dy(1,1)=2-(1/9)*exp(-x(1));dy(1,2)=-1;
dy(2,1)=-1;dy(2,2)=2-(1/9)*exp(-x(2));

```

52. Metode Newton sistem persamaan taklinear kriteria stopping 2

```

function [akar,langkah]=newton_sistem0(p0,tol)
fp=feval('fun',p0); jf = feval('dfun',p0);
if det(jf)==0
    error('metode Newton gagal!')

```

```

end
q0=jf\fp;
p1=p0-q0;
akar=[p0];
langkah=0;
while norm(p1-p0)>tol
    langkah=langkah+1;
    akar=[akar p1];
    p0=p1;
    fp=feval('fun',p0);
    jf=feval('dfun',p0);
    if det(jf)==0
        error('metode Newton gagal!')
    end
    q0=jf\fp;
    p1=p0-q0;
end
%fungsi yang diketahui
function y=fun(x)
y=zeros(2,1);
y(1)=x(1)^2-x(2)+0.25; y(2)=x(1)+x(2)^2+0.25;
%Jacobiannya
function dy=dfun(x)
dy=zeros(2,2);
dy(1,1)=2*x(1);dy(1,2)=-1;
dy(2,1)=1;dy(2,2)=2*x(2);

```

53. Metode Newton dengan Jacobian diaproksimasi

```

function [akar,langkah]=newton_sistem1(p0,tol,h)
fp=feval('fun',p0);
%aproksimasi jacobian
jf=jacob_app(p0,h);
if det(jf)==0
    error('metode Newton gagal!')
end
q0=jf\fp;
p1=p0-q0;
akar=[p0]; langkah=0;
while norm(p1-p0)>tol
    langkah=langkah+1;
    akar=[akar p1];
    p0=p1;
    fp=feval('fun',p0);
    jf=jacob_app(p0,h);
    if det(jf)==0

```

```

            error('metode Newton gagal!')
        end
        q0=jf\fp;
        p1=p0-q0;
    end
    %fungsi yang diketahui
    function y=fun(x)
    y=zeros(2,1);
    y(1)=x(1)^2+x(1)*x(2)^3-9;
    y(2)=3*x(1)^2*x(2)-x(2)^3-4;

```

54. Metode Chord standar

```

function [akar,langkah]=chord_standard(p0,tol)
fp=feval('fun',p0);
jf=feval('dfun',p0); %Jacobian di titik awal
if det(jf)==0
    error('metode Newton gagal!')
end
q0=jf\fp;
p1=p0-q0;
akar=[p0];
langkah=0;
while norm(p1-p0)>tol
    langkah=langkah+1;
    akar=[akar p1];
    p0=p1;
    fp=feval('fun',p0);
    q0=jf\fp;
    p1=p0-q0;
end
%fungsi yang diketahui
function y=fun(x)
y=zeros(2,1);
y(1)=2*x(1)-x(2)+(1/9)*exp(-x(1))-1;
y(2)=-x(1)+2*x(2)+(1/9)*exp(-x(2));
%Jacobiannya
function dy=dfun(x)
dy=zeros(2,2);
dy(1,1)=2-(1/9)*exp(-x(1)); dy(1,2)=-1;
dy(2,1)=-1; dy(2,2)=2-(1/9)*exp(-x(2));

```

55. Metode Chord modifikasi

```

function [akar,langkah]=chord_modif(p0,tol,m)
%m: periode memperbarui nilai awal Jacobian

```

```

fp=feval('fun',p0);
jf=feval('dfun',p0); %Jacobian di titik awal
q0=jf\fp;
p1=p0-q0;
akar=[p0];
langkah=0;
d1=1;d2=m;
while norm(p1-p0)>tol
for k=d1:d2
    langkah=langkah+1;
    akar=[akar p1];
    p0=p1;
    fp=feval('fun',p0);
    q0=jf\fp;
    p1=p0-q0;
end
%updating nilai awal Jacobian
d1=m+1;d2=d2+m;
pu=akar(:,d1);
jf=feval('dfun',pu);
end
%fungsi yang diketahui
function y=fun(x)
y=zeros(2,1);
y(1)=2*x(1)-x(2)+(1/9)*exp(-x(1))-1;
y(2)=-x(1)+2*x(2)+(1/9)*exp(-x(2));
%derivatifnya
function dy=dfun(x)
dy=zeros(2,2);
dy(1,1)=2-(1/9)*exp(-x(1));dy(1,2)=-1;
dy(2,1)=-1;dy(2,2)=2-(1/9)*exp(-x(2));

```

56. Metode Broyden

```

function [akar,langkah]=broyden(p0,tol)
%initialiasasi Jacobian
B0=feval('dfun',p0);
f0=feval('fun',p0);
s0=-B0\f0; p1=p0+s0;
akar=[p0]; langkah=0;
while norm(p1-p0)>tol
    langkah=langkah+1;
    akar=[akar p1];
    yk=feval('fun',p1)-feval('fun',p0);
    B1=B0+(yk-B0*s0)*s0'/(s0'*s0);
    fp1=feval('fun',p1);
    s1=-B1\fp1;

```

```

        p2=p1+s1;
        p0=p1;p1=p2;s0=s1;B0=B1;
end
%fungsi F pada F(x)=0
function F=fun(x);
n=length(x);
F=zeros(n,1);
F(1)=3*x(1)-cos(x(2)*x(3))-0.5;
F(2)=x(1)^2-81*(x(2)+0.1)^2+sin(x(3))+1.06;
F(3)=exp(x(1)*x(2))+20*x(3)+(10*pi-3)/3;
%Jacobiannya
function JF=dfun(x)
n=length(x);
JF=zeros(n,n);
JF(1,1)=3;JF(1,2)=x(3)*sin(x(2)*x(3));JF(1,3)=x(2)*sin(x(2)*x(3)
);
JF(2,1)=2*x(1);JF(2,2)=-162*(x(2)+0.1);JF(2,3)=cos(x(3));
JF(3,1)=-x(2)*exp(x(1)*x(2));JF(3,2)=-x(1)*exp(-
x(1)*x(2));JF(3,3)=20;

```

57. Metode penurunan tercuram (steepest descent)

```

function [akar,langkah]=steepest_descent(p0,tol)
akar=[p0];langkah=0;
[g0,z0]=fun(p0);
while norm(g0)>tol
    langkah=langkah+1;
%menentukan parameter lambda
t1=0;t3=1;
[g1,z1]=fun(p0);[g3,z3]=fun(p0-t3*z0);
while g1<g3
    t3=t3/2;
[g3,z3]=fun(p0-t3*z1);
end
t2=t3/2;
[g2,z2]=fun(p0-t2*z1);
xdata=[t1 t2 t3];ydata=[g1 g2 g3];
t=0:0.0001:t3;
y=poli_tabel(t,xdata,ydata);
m=min(y);
i=find(y==m);
lambda=t(i);
p1=p0-lambda*z0;
akar=[akar p1];
[g2,z2]=fun(p1);
g0=g2;p0=p1;z0=z2;

```



```

end
%fungsi objective dan gradiennya
function [g,z]=fun(x)
n=length(x);
F=zeros(n,1);
JF=zeros(n,n);
F(1)=3*x(1)-cos(x(2)*x(3))-0.5;
F(2)=x(1)^2-81*(x(2)+0.1)^2+sin(x(3))+1.06;
F(3)=exp(x(1)*x(2))+20*x(3)+(10*pi-3)/3;
JF(1,1)=3;JF(1,2)=x(3)*sin(x(2)*x(3));JF(1,3)=x(2)*sin(x(2)*x(3)
);
JF(2,1)=2*x(1);JF(2,2)=-162*(x(2)+0.1);JF(2,3)=cos(x(3));
JF(3,1)=-x(2)*exp(x(1)*x(2));JF(3,2)=-x(1)*exp(-
x(1)*x(2));JF(3,3)=20;
g=F(1)^2+F(2)^2+F(3)^2;
grad=2*JF'*F;
z=grad/norm(grad);

```

58. Metode Euler standar

```

function [t,w]=euler11(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
% Output: t:titik mesh, w: aproksimasi pada titik mesh
h=(b-a)/N;
t = a:h:b;t=t';
w = zeros(N+1,1);
w(1)=alpha;
for k=1:N
    w(k+1)=w(k)+h*feval('fun',t(k),w(k));
end
%fungsi ruas kanan function
z=fun(t,y)
z=(y+t)/(y-t);

```

59. Metode Euler selisih mundur

```

function [t,w,wb]=euler_back1(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
% Output: t:titik mesh, w: aproksimasi Euler, wb: aproksimasi
Euler mundur
h=(b-a)/N;
t=a:h:b;t=t';
%euler
w=zeros(N+1,1);

```

```

w(1)=alpha;
for k=1:N
    w(k+1)=w(k)+h*feval('fun',t(k),w(k));
end
%euler back
wb=zeros(N+1,1);
wb(1)=alpha;
tb=[t;b+h];
for k=1:N
    wb(k+1)=wb(k)+h*feval('fun',tb(k+1),w(k+1));
end
%fungsi ruas kanan
function z=fun(t,y)
z=(y+t)/(y-t);

```

60. Metode Euler perbaikan

```

function [t,w]=euler_improve(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
% Output: t:titik mesh, w: aproksimasi pada titik mesh
h=(b-a)/N;
t=a:h:b;t=t';
w=zeros(N+1,1);
w(1)=alpha;
tb=[t;b+h];
for k=1:N
    e=feval('fun',tb(k+1),w(k)+h*feval('fun',tb(k),w(k)));
    w(k+1)=w(k)+(h/2)*(feval('fun',tb(k),w(k))+ e);
end
%fungsi ruas kanan
function z=fun(t,y)
z=(y+t)/(y-t);

```

61. Metode Runge-Kutta order 4

```

function [t,w]=rungekutta4(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
% Output: t:titik mesh, w: aproksimasi pada titik mesh
h=(b-a)/N; t=a:h:b;t=t';
w=zeros(N+1,1);
w(1)=alpha;
for k=1:N
    k1=feval('fun',t(k),w(k));
    k2=feval('fun',t(k)+h/2,w(k)+h*k1/2);

```

```

        k3=feval('fun',t(k)+h/2,w(k)+h*k2/2);
        k4=feval('fun',t(k)+h,w(k)+h*k3/2);
        w(k+1)=w(k)+(h/6)*(k1+2*k2+2*k3+k4);
end
%fungsi ruas kanan
function z = fun(t,y)
z=(y+t)/(y-t);

```

62. Metode Runge-Kutta order 4 untuk sistem

```

function [t,w]=rk4_sistem(a,b,alpha,N)
n=length(alpha);%banyaknya variabel
%Input: a,b: batas interval waktu, y0: vektor nilai awal, n:
banyak variabel,
%N: banyak subinterval
% Output: t:titik mesh, w: matriks aproksimasi pada titik mesh
h=(b-a)/N;
t = a:h:b;
w=zeros(n,N+1);% menyiapkan w_{j,i} matriks aproksimasi.
w(:,1)=alpha;
for k=1:N
    k1=feval('fun',t(k),w(:,k));
    k2=feval('fun',t(k)+h/2,w(:,k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(:,k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(:,k)+h*k3);
    w(:,k+1)=w(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
end
w=w';
%fungsi ruas kanan
function z=fun(t,y)
n=2;%nilai ini menyesuaikan banyaknya variabel.
z=zeros(n,1);
z(1)=y(1)-y(2)+2;z(2)=-y(1)+y(2)+4*t;

```

63. Metode Runge-Kutta order 4 untuk persamaan tingkat tinggi

```

function [t,w]=rk4_order_tinggi(a,b,alpha,N)
n=length(alpha);
%Input: a,b: batas interval waktu, y0: vektor nilai awal, n:
banyak variabel,
%N: banyak subinterval
% Output: t:titik mesh, w: matriks aproksimasi pada titik mesh
h=(b-a)/N;
t = a:h:b;t=t';
w=zeros(n,N+1);% menyiapkan w_{j,i} matriks aproksimas.

```

```

w(:,1)=alpha;
for k=1:N
    k1=feval('fun',t(k),w(:,k));
    k2=feval('fun',t(k)+h/2,w(:,k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(:,k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(:,k)+h*k3);
    w(:,k+1)=w(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
end
w=w';
%fungsi ruas kanan
function z=fun(t,y)
n=2;
z=zeros(n,1);
z(1)=y(2);
z(2)=(2/t)*y(2)-(2/t^2)*y(1)+t*log(t);

```

64. Metode Adam-Basfort order 4

```

function [t,w]=ab4(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
%Output: t:titik mesh, w: aproksimasi pada titik mesh
h=(b-a)/N; t=a:h:b;t=t';
w=zeros(N+1,1);
w(1)=alpha; %inisialisasi menggunakan RK-4
for k=1:3
    k1=feval('fun',t(k),w(k));
    k2=feval('fun',t(k)+h/2,w(k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(k)+h*k3);
    w(k+1)=w(k)+(h/6)*(k1+2*k2+2*k3+k4);
end
%metode AB-4
for k=4:N
    w(k+1)=w(k)+(h/24)*(55*feval('fun',t(k),w(k))-
    59*feval('fun',t(k-1),...
    w(k-1))+37*feval('fun',t(k-2),w(k-2))- . . .
    9*feval('fun',t(k-3),w(k-3)));
end
%fungsi ruas kanan
function z=fun(t,y)
z=y-t^2+1;

```

65. Metode Adam-Basfort order 4 dengan memori

```

function [t,w]=ab4_1(a,b,alpha,N)

```

```

h=(b-a)/N;
t=a:h:b;t=t';
w=zeros(N+1,1);
w(1)=alpha; %inisialisasi menggunakan RK-4
for k=1:3
    k1=feval('fun',t(k),w(k));
    k2=feval('fun',t(k)+h/2,w(k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(k)+h*k3);
    w(k+1)=w(k)+(h/6)*(k1+2*k2+2*k3+k4);
end
%metode AB-4
f4=feval('fun',t(4),w(4));f3=feval('fun',t(3),w(3));
f2=feval('fun',t(2),w(2));f1=feval('fun',t(1),w(1));
for k=4:N
    w(k+1)=w(k)+(h/24)*(55*f4-59*f3+37*f2-9*f1);
    f1=f2;f2=f3;f3=f4;f4=feval('fun',t(k+1),w(k+1));
end
%fungsi ruas kanan
function z = fun(t,y)
z=y-t^2+1;

```

66. Metode prediktor-korektor

```

function [t,w]=prediktor_korektor(a,b,alpha,N)
%Input: a,b: batas interval waktu, y0: nilai awal, N: banyak
subinterval
%Output: t:titik mesh, w: aproksimasi pada titik mesh
h=(b-a)/N;
t=a:h:b;t=t';
w=zeros(N+1,1);
w(1)=alpha;
%inisialisasi menggunakan RK-4
for k=1:3
    k1=feval('fun',t(k),w(k));
    k2=feval('fun',t(k)+h/2,w(k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(k)+h*k3);
    w(k+1)=w(k)+(h/6)*(k1+2*k2+2*k3+k4);
end
f4=feval('fun',t(4),w(4));f3=feval('fun',t(3),w(3));
f2=feval('fun',t(2),w(2));f1=feval('fun',t(1),w(1));
for k=4:N
    wp=w(k)+(h/24)*(55*f4-59*f3+37*f2-9*f1);%metode AB-4
    fp=feval('fun',t(k+1),wp);
    w(k+1)=w(k)+(h/24)*(9*fp+19*f4-5*f3+f2);%metode AM-3
    f1=f2;f2=f3;f3=f4;f4=feval('fun',t(k+1),w(k+1));

```

```

end
%fungsi ruas kanan
function z = fun(t,y)
z=y-t^2+1;

```

67. Metode shooting linear

```

function [t,w1,w2,w]=shootinglinear(a,b,alpha,beta,N)
%penyelesaian MNA pertama:
[t,w]=rk4_sistem(a,b,[alpha;0],N); w1=w(:,1);
%penyelesaian MNA kedua:
[t,w]=rk4_sistem(a,b,[0;1],N);
w2=w(:,1);
%kombinasi linear:
w=w1+(beta-w1(end)/w2(end))*w2;

```

68. Metode shooting taklinear dengan secant

```

function [t,ss,w]=shootingtaklinear_secant(a,b,alpha,beta,N,s0,s1,tol)
%input: batas interval a,b; nilai batas alpha dan beta; banyak
titik mesh N,
%inisialisasi s0 dan s1, dan tol pada metode secant
ss=[s0 s1];
%penyelesaian MNA pertama:
[t,w]=rk4_sistem(a,b,[alpha;s0],N);
w0n=w(end,1);
%penyelesaian MNA kedua:
[t,w]=rk4_sistem(a,b,[alpha;s1],N);
w1n=w(end,1);
%pembaruan pertama:
s=s1-(w1n-beta)*(s1-s0)/(w1n-w0n);
ss=[ss s];
while abs(s-s1)>tol
    w0n=w1n;s0=s1;s1=s;
    [t,w]=rk4_sistem(a,b,[alpha;s1],N);
    w1n=w(end,1);
    s=s1-(w1n-beta)*(s1-s0)/(w1n-w0n);
    ss=[ss s];
end
%penyelesaian "tembakan" MNA terkahir:
[t,w]=rk4_sistem(a,b,[alpha;s],N);
w=w(:,1);

```

69. Metode shooting taklinear dengan Newton

```

function [t,ss,w]=shootingtaklinear_newton(a,b,alpha,beta,N,s0,tol)

```

```

h=(b-a)/N;
t = a:h:b;t=t';
%menyelesaikan MNA pertama:
w=zeros(2,N+1);
w(:,1)=[alpha s0];
for k=1:N
    k1=feval('fun',t(k),w(:,k));
    k2=feval('fun',t(k)+h/2,w(:,k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(:,k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(:,k)+h*k3);
    w(:,k+1)=w(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
end
w=w';
w0n=w(end,1);w1n=w(end,2);
%menyelesaikan MNA kedua
d=zeros(2,N+1);
d(:,1)=[0 s0];
for k=1:N
global fy fy1
fy=w0n;fy1=w1n;
    k1=feval('fun1',t(k),d(:,k));
    k2=feval('fun1',t(k)+h/2,d(:,k)+h*k1/2);
    k3=feval('fun1',t(k)+h/2,d(:,k)+h*k2/2);
    k4=feval('fun1',t(k)+h,d(:,k)+h*k3);
    d(:,k+1)=d(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
end
d=d';d0n=d(end,1);
%updating pertama:
s1=s0-(w0n-beta)/d0n;
ss=[s0 s1];
while abs(s1-s0)>tol
    w=zeros(2,N+1);
    s0=s1;
    w(:,1)=[alpha s0];
for k=1:N
    k1=feval('fun',t(k),w(:,k));
    k2=feval('fun',t(k)+h/2,w(:,k)+h*k1/2);
    k3=feval('fun',t(k)+h/2,w(:,k)+h*k2/2);
    k4=feval('fun',t(k)+h,w(:,k)+h*k3);
    w(:,k+1)=w(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
end
w=w';
w0n=w(end,1);w1n=w(end,2);
d=zeros(2,N+1);
    d(:,1)=[0 1];
for k=1:N
global fy fy1

```

```

        fy=w0n;fy1=w1n;
        k1=feval('fun1',t(k),d(:,k));
        k2=feval('fun1',t(k)+h/2,d(:,k)+h*k1/2);
        k3=feval('fun1',t(k)+h/2,d(:,k)+h*k2/2);
        k4=feval('fun1',t(k)+h,d(:,k)+h*k3);
        d(:,k+1)=d(:,k)+(h/6)*(k1+2*k2+2*k3+k4);
    end
d=d';d0n=d(end,1);
%updating berikutnya:
s1=s0-(w0n-beta)/d0n;
ss=[ss s1];
end
%fungsi ruas kanan
function z=fun(t,y)
z=[y(2);(32+2*t^3-y(1)*y(2))/8];
function z=fun1(t,z)
global fy fy1
z=[z(2);(-fy1*z(1)-fy*z(2))/8];

```

70. Metode selisih hingga untuk MSB linear

```

function [t,w]=selisih_hingga(a,b,alpha,beta,h)
t=a:h:b;t=t';
tin=t(2:end-1);%mesh interior
N=length(tin);
%mendefinisikan matriks A
A=zeros(N,N);
a=1+(h/2)*feval('p',tin(2:N));
b=-2-h^2*feval('q',tin(1:N));
c=1-(h/2)*feval('p',tin(1:N-1));
A=diag(a,-1)+diag(b,0)+diag(c,1);
%mendefinisikan vektor f
f=zeros(N,1);
f(1)=-h^2*feval('r',tin(1))-(1+(h/2)*feval('p',tin(1)))*alpha;
f(2:N-1)=-h^2*feval('r',tin(2:N-1)); w=A\f;
f(N)=-h^2*feval('r',tin(N))-(1-(h/2)*feval('p',tin(N)))*beta;
w=[alpha;w;beta];%penggabungan syarat batas
%fungsi yang diketahui p(t), q(t), dan r(t)
function y=p(x)
y=2*ones(size(x));
function y=q(x)
y=-1*ones(size(x));
function y=r(x)
y=x-x.*(exp(x));

```

71. Metode selisih hingga untuk MSB taklinear


```

function [t,w]=selisih_hingga_taklinear(a1,b1,alpha,beta,h,iter)
%input: batas interval [a1,b1], nilai batas "alpha" dan "beta",
meshsize
%"h",banyak iterasi"iter", output: titik mesh "t" dan aproksimasi "w".
tt=a1:h:b1; t=tt(2:end-1)';N=length(t);
%inisialisasi:
w=alpha+(beta-alpha)*(t-a1)/(b1-a1);
for k=1:iter
%mendefinisikan Jacobian:
    J=zeros(N,N);
    a=zeros(1,N-1);
    for i=2:N-1
        a(i-1)=-1-0.5*h*feval('fdy',t(i),[w(i),(w(i+1)-w(i-1))]/(2*h)]);
    end
    a(N-1)=-1-0.5*h*feval('fdy',t(N),[w(N),(beta-w(N-1))]/(2*h)]);
    b=zeros(1,N);
    for i=2:N-1
        b(i)=2+h^2*feval('fy',t(i),[w(i),(w(i+1)-w(i-1))]/(2*h)]);
    end
    b(1)=2+h^2*feval('fy',t(1),[w(1),(w(2)-alpha)/(2*h)]);
    b(N)=2+h^2*feval('fy',t(N),[w(N),(beta-w(N-1))]/(2*h)]);
    c=zeros(1,N-1);
    for i=2:N-1
        c(i)=-1+0.5*h*feval('fdy',t(i),[w(i),(w(i+1)-w(i-1))]/(2*h)]);
    end
    c(1)=-1+0.5*h*feval('fdy',t(1),[w(1),(w(2)-alpha)/(2*h)]);
    J=diag(a,-1)+diag(b,0)+diag(c,1);
%mendefinisikan fungsi pada sistem persamaan taklinear:
F=zeros(N,1);
F(1)=2*w(1)-w(2)+h^2*feval('odefun',t(1),[w(1),(w(2)-alpha)/(2*h)])-alpha;
for i=2:N-1
    F(i)=-w(i-1)+2*w(i)-w(i+1)+h^2*feval('odefun',t(i),...
        [w(i),(w(i+1)-w(i-1))]/(2*h)]);
end
F(N)=-w(N-1)+2*w(N)+h^2*feval('odefun',t(N),[w(N),(beta-w(N-1))]/(2*h)])-beta;
%menyelesaikan SPL Jq=F:
q=J\F;
%updating w:
w=w-q;
end
t=[a1;t;b1];w=[alpha;w;beta];%penggabungan syarat batas

```

```

%fungsi ode
function z=odefun(t,y)
z=(32+2*t^3-y(1)*y(2))/8;
%fungsi derivatif
function z=fy(t,y)
z=-y(2)/8;
function z=fdy(t,y)
z=-y(1)/8;

```

72. Metode Ritz

```

function u=sol_ritz1(x)
a=1;b=2;alpha=0.5;beta=log(2);
N=12; xx=linspace(a,b,N); h=xx(2:end)-xx(1:end-1);
m=(xx(1:end-1)+xx(2:end))/2;
n=N-2;
%mendefinisikan matriks K dan vektor f
K=zeros(n);f=zeros(n,1);
for i=1:n
    I1=fun1(m(i))/h(i);
    I2=(xx(i+1)-m(i))*(m(i)-xx(i))*fun2(m(i))/h(i);
    I3=fun1(m(i+1))/h(i+1);
    I4=(m(i)-xx(i))^2*fun2(m(i))/h(i);
    I5=(xx(i+2)-m(i+1))^2*fun2(m(i+1))/h(i+1);
    I6=(xx(i+2)-m(i+1))*(m(i+1)-xx(i+1))*fun2(m(i+1))/h(i+1);
    I7=(m(i)-xx(i))*fun3(m(i))/h(i);
    I8=(xx(i+2)-m(i+1))*fun3(m(i+1))/h(i+1);
    K(i,i)=I1*(1/h(i))^2+I3*(1/h(i+1))^2+I4*(1/h(i))^2+ . . .
    I5*(1/h(i+1))^2;
    f(i)=I7/h(i)+I8/h(i+1);
end
for i=1:n-1
    I1=fun1(m(i))/h(i);
    I2=(xx(i+1)-m(i))*(m(i)-xx(i))*fun2(m(i))/h(i);
    I3=fun1(m(i+1))/h(i+1);
    I4=(m(i)-xx(i))^2*fun2(m(i))/h(i);
    I5=(xx(i+2)-m(i+1))^2*fun2(m(i+1))/h(i+1);
    I6=(xx(i+2)-m(i+1))*(m(i+1)-xx(i+1))*fun2(m(i+1))/h(i+1);
    I7=(m(i)-xx(i))*fun3(m(i))/h(i);
    I8=(xx(i+2)-m(i+1))*fun3(m(i+1))/h(i+1);
    K(i,i+1)=-I3*(1/h(i+1))^2+I6*(1/h(i+1))^2;
end
for i=2:n
    I1=fun1(m(i))/h(i);
    I2=(xx(i+1)-m(i))*(m(i)-xx(i))*fun2(m(i))/h(i);
    I3=fun1(m(i+1))/h(i+1);
    I4=(m(i)-xx(i))^2*fun2(m(i))/h(i);

```

```

I5=(xx(i+2)-m(i+1))^2*fun2(m(i+1))/h(i+1);
I6=(xx(i+2)-m(i+1))*(m(i+1)-xx(i+1))*fun2(m(i+1))/h(i+1);
I7=(m(i)-xx(i))*fun3(m(i))/h(i);
I8=(xx(i+2)-m(i+1))*fun3(m(i+1))/h(i+1);
K(i,i-1)=-(1/h(i))^2*I1+(1/h(i))^2*I2;
end
c=K\f;
u1=0;
for i=1:n
    phi=(1/h(i))*(x-xx(i)).*(x>xx(i)&x<=xx(i+1))+...
        (1/h(i+1))*(xx(i+2)-x).*(x>xx(i+1)&x<=xx(i+2));
    u0=c(i)*phi;
    u1=u1+u0;
end
u=u1+(1/(b-a))*(beta*(x-a)+alpha*(b-x));
%fungsi-fungsi yang diketahui (p(x), q(x), dan h(x))
function y=fun1(x)
y=4*x;
function y=fun2(x)
y=2/x^2;
function y=fun3(x)
y=-22/x^2+40/x^3-4/x^4+(2/x^2)*log(x)+4*(log(2)-0.5)+...
(4/x^2)*((x-1)*log(2)+(2-x)*(0.5));

```

73. Metode residu terbobot

```

function [z,dz]=fun2018(x)
%menghitung koefisien a_{i,k} dengan metode Simpson
a11=(1/6)*(f1(0)*df1(0)*df1(0)+4*f1(0.5)*df1(0.5)*df1(0.5)+...
f1(1)*df1(1)*df1(1));
a12=(1/6)*sqrt(2)*(df1(0)*df1(0)+4*df1(0.5)*df1(0.5)+df1(1)*df1(1));
a13=(1/6)*((f1(0)*df2(0)+f2(0)*df1(0))*df1(0)+4*(f1(0.5)*df2(0.5)+...
f2(0.5)*df1(0.5))*df1(0.5)+(f1(1)*df2(1)+f2(1)*df1(1))*df1(1));
a14=(1/6)*sqrt(2)*(df2(0)*df1(0)+4*df2(0.5)*df1(0.5)+df2(1)*df1(1));
a15=(1/6)*(f2(0)*df2(0)*df1(0)+4*f2(0.5)*df2(0.5)*df1(0.5)+...
f2(1)*df2(1)*df1(1));
a16=(1/6)*(f1(0)+4*f1(0.5)+f1(1));
a21=(1/6)*(f1(0)*df1(0)*df2(0)+4*f1(0.5)*df1(0.5)*df2(0.5)+...
f1(1)*df1(1)*df2(1));
a22=(1/6)*sqrt(2)*(df1(0)*df2(0)+4*df1(0.5)*df2(0.5)+df1(1)*df2(1));
a23=(1/6)*((f1(0)*df2(0)+f2(0)*df1(0))*df2(0)+4*(f1(0.5)*df2(0.5)+...
f2(0.5)*df1(0.5))*df2(0.5)+(f1(1)*df2(1)+f2(1)*df1(1))*df2(1));
a24=(1/6)*sqrt(2)*(df2(0)*df2(0)+4*df2(0.5)*df2(0.5)+df2(1)*df2(1));
a25=(1/6)*(f2(0)*df2(0)*df2(0)+4*f2(0.5)*df2(0.5)*df2(0.5)+...
f2(1)*df2(1)*df2(1));
a26=(1/6)*(f2(0)+4*f2(0.5)+f2(1));
%mendefinisikan fungsi taklinear
z=zeros(2,1);

```

```

z(1)=a11*x(1)^2+a12*x(1)+a13*x(1)*x(2)+a14*x(2)+a15*x(2)^2+a16;
z(2)=a21*x(1)^2+a22*x(1)+a23*x(1)*x(2)+a24*x(2)+a25*x(2)^2+a26;
%mendefinisikan Jacobian
dz=zeros(2,2);
dz(1,1)=2*a11*x(1)+a12+a13*x(2);dz(1,2)=a13*x(1)+a14+2*a15*x(2);
dz(2,1)=2*a21*x(1)+a22+a23*x(2);dz(2,2)=a23*x(1)+a24+2*a25*x(2);
%fungsi basis dan derivatifnya
function y=f1(x)
y=1-x;
function dy=df1(x)
dy=-1;
function y=f2(x)
y=1-x^2;
function dy=df2(x)
dy=-2*x;

```

74. Fungsi basis b-spline kubik

```

function [y,dy,ddy]=bspline3_basis(x,xdata,i)
h=xdata(2)-xdata(1);
x1=(x-xdata(i))/h;
y=bspline3(x1);
dy=dbspline3(x1)/h;
ddy=ddbbspline3(x1)/h^2;
%formula fungsi induk, 1st derivatif, dan 2nd derivatifnya
function y=bspline3(x)
y=(x+2).^3/6.*(x>=-2&x<-1)+(-0.5*x.^3-x.^2+2/3).*(-1<=x&x<0)+...
(0.5*x.^3-x.^2+2/3).*(x>=0&x<1)+(1/6)*(2-x).^3.*(x>=1&x<2);
function y=dbspline3(x)
y=(x+2).^2/2.*(x>=-2&x<-1)+(-1.5*x.^2-2*x).*(-1<=x&x<0)+...
(1.5*x.^2-2*x).*(x>=0&x<1)+(-0.5)*(2-x).^2.*(x>=1&x<2);
function y=ddbbspline3(x)
y=(x+2).*(x>=-2&x<-1)+(-3*x-2).*(-1<=x&x<0)+...
(3*x-2).*(x>=0&x<1)+(2-x).*(x>=1&x<2);

```

75. Metode kolokasi titik

```

function u=kolokasi1(x)
xdata=-0.1:0.1:1.1;%knot
N=length(xdata);
xx=linspace(0,1,N);%titik kolokasi
%mendefinisikan SPL dan menyelesaikannya
K=zeros(N,N);
f=zeros(N,1);
for k=2:N-1

```

```

        for i=1:N
            [y, dy, ddy]=bspline3_basis(xx(k), xdata, i);
            K(k, i)=-ddy+dy+y;
        end
        f(k)=feval('fun1', xx(k));
    end
    for i=1:N
        [y, dy, ddy]=bspline3_basis(xx(1), xdata, i);
        K(1, i)=y;
    end
    for i=1:N
        [y, dy, ddy]=bspline3_basis(xx(end), xdata, i);
        K(end, i)=y;
    end
    f(1)=0; f(end)=0;
    c=K\f;
    %mendefinisikan kombinasi linear
    u=0;
    for i=1:N
        [y, dy, ddy]=bspline3_basis(x, xdata, i);
        u0=c(i)*y;
        u=u+u0;
    end
    %fungsi ruas kanan
    function y=fun1(x)
    y=(x-1)*exp(x)-exp(1)*(x+1);

```

76. Metode golden-section untuk optimasi satu variabel

```

function [x, fmin, langkah]=golden_section(a, b, tol)
a0=a; b0=b;
p=(-1+sqrt(5))/2;%rasio emas
it=0;
m=[];
while (b0-a0)>tol
    it=it+1;
    x1=a0+(1-p)*(b0-a0); x2=a0+p*(b0-a0);
    f1=feval('fun', x1); f2=feval('fun', x2);
    if f1<f2
        b0=x2;
    else
        a0=x1;
    end
    m0=(a0+b0)/2;
    m=[m; m0];
end

```

```

fmin=feval('fun',m(end));langkah=it;
%fungsi yang diminimumkan
function y=fun(x)
y=0.5-x.*exp(-x.^2);

```

77. Metode parabolik berjenjang untuk optimasi satu variabel

```

function [x,langkah]=metode_parabolik(node,tol)
xdata=node;
ydata=feval('fun',xdata);
D = hbagi_tabel(xdata,ydata);
a=D(3,3);b=D(2,2)-D(3,3)*(xdata(1)+xdata(2));
x0=-b/(2*a); %nilai minimum fungsi kuadrat
x=[x0];
langkah=1;
while abs(xdata(3)-x0)>tol
    langkah=langkah+1;
    xdata(1)=xdata(2);xdata(2)=xdata(3);xdata(3)=x0;
    ydata=feval('fun',xdata);
    D = hbagi_tabel(xdata,ydata);
    a=D(3,3);b=D(2,2)-D(3,3)*(xdata(1)+xdata(2));
    x0=-b/(2*a);
    x=[x;x0];
end
%fungsi yang diminimumkan
function y=fun(x)
y=0.5-x.*exp(-x.^2);

```

78. Metode parabolik golden-section untuk optimasi satu variabel

```

function [m,fmin,langkah]=parabolik_golden_section(a,b,tol)
a0=a;b0=b;
p=(-1+sqrt(5))/2;%rasio emas
it=0;
m=[];
while (b0-a0)>tol
    it=it+1;
    x1=a0+(1-p)*(b0-a0);x2=a0+p*(b0-a0);
    f1=feval('fun',x1);f2=feval('fun',x2);
    if f1<f2
        b0=x2;
        xdata=[a x1 x2];
    else
        a0=x1;
        xdata=[x1 x2 b];
    end
end

```

```

ydata=feval('fun',xdata);
D=hbagi_tabel(xdata,ydata);
a=D(3,3);b=D(2,2)-D(3,3)*(xdata(1)+xdata(2));
m0=-b/(2*a);
m=[m;m0];
end
fmin=feval('fun',m(end));
langkah=it;
%fungsi yang diminimumkan
function y=fun(x)
y=0.5-x.*exp(-x.^2);

```

79. Metode Fibonacci untuk optimasi satu variabel

```

function [x,fmin,langkah]=fibonacci(a,b,tol)
%mendefinisikan barisan Fibonacci
F=[1;2];
langkah=2;
while F(end)+F(end-1)<(b-a)/tol;
    langkah=langkah+1;
    F0=F(end)+F(end-1);
    F=[F;F0];
end
N=langkah;
F(N+1)=F(N)+F(N-1);
phi=F(N:-1:1)./F(N+1:-1:2);
x=[];
for k=1:N-1;
    p=phi(k);
    x1=a+(1-p)*(b-a);x2=a+p*(b-a);
    f1=feval('fun',x1);f2=feval('fun',x2);
    if f1<f2
        b=x2;
    else
        a=x1;
    end
    x0=(a+b)/2;
    x=[x;x0];
end
epsilon=1e-2;p=phi(end)+epsilon;
x1=a+(1-p)*(b-a);x2=a+p*(b-a);
f1=feval('fun',x1);f2=feval('fun',x2);
if f1<f2
    a=x2;
else
    b=x1;

```

```

end
x0=(a+b)/2;
x=[x;x0];
fmin=feval('fun',x0);
%fungsi yang diminimumkan
function y=fun(x)
y=0.5-x.*exp(-x.^2);

```

80. Metode Newton untuk optimasi satu variabel

```

function [x,fmin,langkah]=newton_min(x0,tol)
x1=x0-feval('dfun',x0)/feval('ddfun',x0);
x=[x0;x1];
langkah=1;
while abs(x1-x0)>tol
    langkah=langkah+1;
    x0=x1;
    x1=x0-feval('dfun',x0)/feval('ddfun',x0);
    x=[x;x1];
end
fmin=feval('fun',x1);
%fungsi yang diketahui function
y=fun(x)
y=0.5-x.*exp(-x.^2);
function dy=dfun(x);
dy=(2*x.^2-1).*exp(-x.^2);
function ddy=ddfun(x)
ddy=2*x.*(3-2*x.^2).*exp(-x.^2);

```

81. Metode Secant untuk optimasi satu variabel

```

function [x,fmin,langkah]=secant_min(a,b,tol)
x0=a;x1=b;
x=[];
langkah=0;
while abs(x1-x0)>tol
    langkah=langkah+1;
    xx=x1-(x1-x0)*feval('dfun',x1)/(feval('dfun',x1)-
    feval('dfun',x0));
    x=[x;xx];
    x0=x1;x1=xx;
end
fmin=feval('fun',x1);
%fungsi yang diketahui
function y=fun(x)
y=0.5-x.*exp(-x.^2);
function dy=dfun(x);

```



```
dy=(2*x.^2-1).*exp(-x.^2);
```

82. Metode penurunan tercuram (steepest-descent) untuk optimasi multivariabel

```
function [x,fmin,langkah]=steepest_secant(x0,tol)
%iterasi pertama
a=0;b=0.1;tol1=1e-4;
while abs(b-a)>tol1
    dphia=dfun(x0)'*dfun(x0-a*dfun(x0));
    dphib=dfun(x0)'*dfun(x0-b*dfun(x0));
    c=b-(b-a)*dphib/(dphib-dphia);
    a=b;b=c;
end
alpha=c;
x1=x0-alpha*dfun(x0);
langkah=1;
%iterasi ke 2,3, . . .
while norm(x1-x0)>tol
    langkah=langkah+1;
    a=0;b=0.5;
    while abs(b-a)>tol1
        dphia=dfun(x1)'*dfun(x1-a*dfun(x1));
        dphib=dfun(x1)'*dfun(x1-b*dfun(x1));
        c=b-(b-a)*dphib/(dphib-dphia);
        a=b;b=c;
    end
    alpha=c;
    x2=x1-alpha*dfun(x1);
    x0=x1;x1=x2;
end
x=x2;
fmin=fun(x);
%fungsi dan gradien fungsi yang diberikan
function f=fun(x)
f=(x(1)+3)^4+(x(2)-2)^2+(x(3)-4)^4;
function df=dfun(x)
df=[4*(x(1)+3)^3;2*(x(2)-2);4*(x(3)-4)^3];
```

83. Metode Newton untuk optimasi multivariabel

```
function [x,fmin,langkah]=newton_multi_min(x0,tol)
H=feval('ddfun',x0);b=-feval('dfun',x0);
s=H\b;
x1=x0+s;
langkah=1;
while norm(x1-x0)>tol
```

```

    langkah=langkah+1;
    H=feval('ddfun',x1);b=-feval('dfun',x1);
    s=H\b;
    x2=x1+s;
    x0=x1;x1=x2;
end
x=x2;fmin=fun(x);
%fungsi, derivatif pertama (Jacobian), dan Hessian
function f=fun(x)
f=(x(1)+3)^4+(x(2)-2)^2+(x(3)-4)^4;
function df=dfun(x)
df=[4*(x(1)+3)^3;2*(x(2)-2);4*(x(3)-4)^3];
function ddf=ddfun(x)
ddf=zeros(3,3);
ddf(1,1)=12*(x(1)+3)^2;ddf(2,2)=2;ddf(3,3)=12*(x(3)-4)^2;

```

84. Metode konjugat gradien untuk optimasi multivariabel

```

function [x,fmin,langkah]=konjugat_gradien(x0,tol)
%iterasi pertama
a=0;b=0.1;toll=1e-4;
while abs(b-a)>toll
    dphia=dfun(x0)'*dfun(x0-a*dfun(x0));
    dphib=dfun(x0)'*dfun(x0-b*dfun(x0));
    c=b-(b-a)*dphib/(dphib-dphia);
    a=b;b=c;
end
alpha=c;
x1=x0-alpha*dfun(x0);
langkah=1;
%iterasi ke 2,3, . . .
while norm(x1-x0)>tol
    langkah=langkah+1;
    g0=feval('dfun',x0);
    g1=feval('dfun',x1);
    beta=g1'*g1/(g0'*g0);
    s1=-g1-beta*g0;
    a=0;b=0.5;
    while abs(b-a)>toll
        dphia=dfun(x1)'*dfun(x1-a*s1);
        dphib=dfun(x1)'*dfun(x1-b*s1);
        c=b-(b-a)*dphib/(dphib-dphia);
        a=b;b=c;
    end
    alpha=c;
    x2=x1-alpha*s1;

```

```

        x0=x1;x1=x2;
end
x=x2; fmin=fun(x);
%fungsi dan gradien fungsi yang diberikan
function f=fun(x)
f=0.5*x(1)^2+2.5*x(2)^2;
function df=dfun(x)
df=[x(1);5*x(2)];

```

85. Metode kuasi-Newton untuk optimasi multivariabel

```

function [x,fmin,langkah]=dfg_method(x0,tol)
n=length(x0);
A0=eye(n);
langkah=0;
while norm(dfun(x0))>tol
    langkah=langkah+1;
    g0=feval('dfun',x0);d0=-A0*g0;
%metode secant untuk menentukan alpha_k
a=0;b=0.5;tol1=1e-3;
    while abs(b-a)>tol1
        dphia=dfun(x0)'*dfun(x0+a*d0);
        dphib=dfun(x0)'*dfun(x0+b*d0);
        c=b-(b-a)*dphib/(dphib-dphia);
        a=b;b=c;
    end
    alpha=c;
    x1=x0+alpha*d0;
%pembaruan matriks A_k
    g1=feval('dfun',x1);delx=alpha*d0;delg=g1-g0;
    A1=A0+(delx*delx'/(delx'*delg) -
        (A0*delg)*(A0*delg)'/(delg'*A0*delg));
    x0=x1;A0=A1;
end
x=x1;
fmin=fun(x);
%fungsi dan gradien fungsi yang diberikan
function f=fun(x)
f=0.5*[x(1) x(2)]*[4 2;2 2]*[x(1);x(2)]-[x(1) x(2)]*[-1;1];
function df=dfun(x)
df=[4 2;2 2]*[x(1);x(2)]-[-1;1];

```

86. Metode BFGS untuk optimasi multivariabel

```

function [x,fmin,langkah]=bfgs_method(x0,tol)

```

```

n=length(x0);
A0=eye(n);
langkah=0;
while norm(dfun(x0))>tol
    langkah=langkah+1;
    g0=feval('dfun',x0);d0=-A0*g0;
    %metode secant untuk menentukan alpha_k
    a=0;b=0.5;tol1=1e-3;
    while abs(b-a)>tol1
        dphia=dfun(x0)'*dfun(x0+a*d0);
        dphib=dfun(x0)'*dfun(x0+b*d0);
        c=b-(b-a)*dphib/(dphib-dphia);
        a=b;b=c;
    end
    alpha=c;
    x1=x0+alpha*d0;
    %pembaruan matriks A_k
    g1=feval('dfun',x1);delx=alpha*d0;delg=g1-g0;
    A1=A0+(1+delg'*A0*delg/(delg'*delx))*delx*delx'/(delx'*delg)-...
    (A0*delg*delx'+(A0*delg*delx)')/(delg'*delx);
    x0=x1;A0=A1;
end
x=x1;
fmin=fun(x);
%fungsi dan gradien fungsi yang diberikan
function f=fun(x)
f=0.5*[x(1) x(2)]*[5 -3;-3 2]*[x(1);x(2)]-[x(1)
x(2)]*[0;1]+log(pi);
function df=dfun(x)
df=[5 -3;-3 2]*[x(1);x(2)]-[0;1];

```

87. Metode simpleks untuk program linear

```

function Af=simpleks1(A,b,c)
[m,n]=size(A);
A1=[A eye(m) b;-c' zeros(1,m) 0];
nonbasis_indeks=[1:n];
coeff=A1(m+1,1:end-1);%koef pada matriks perluasan kanonik
coeff1=A1(m+1,nonbasis_indeks);%koef variabel nonbasis
while min(coeff1)<0
    t1=min(coeff1);
    q=find(coeff==t1);
    %uji rasio
    b=A1(1:end-1,end);
    r=b./(A1(1:end-1,q)+eps);
    while max(r)<0
        disp('solusi takterbatas')
    end
end

```

```

        break
    end
    t2=min(r);
    p=find(r==t2);
    %OBE A1 menggunakan pivot A1(p,q)
    alpha=A1(p,q);
    A0=A1;
    for i=1:m+1
        if i==p
            A1(i,:)=A0(i,+)/alpha;
        else
            A1(i,:)=A0(i,)-(A0(p,)./alpha)*A0(i,q);
        end
    end
    end
    coeff1=A1(m+1,nonbasis_indeks);
    coeff=A1(m+1,1:end-1);
    end
    Af=A1;

```