LAPORAN TUGAS BESAR

IF-5162 / Metode Numerik Lanjut

Penyelesaian Persoalan Numerik

Dipersiapkan oleh:

Gugy Lucky Khamdani / 23517041

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | Nomor Dokumen | | Halaman |
| *IF5162/23517041* | | *27* |
| *-* | *-* | *23 Mei 2018* |

# Daftar Isi

[Daftar Isi 2](#_Toc514849574)

[1 Pendahuluan 3](#_Toc514849575)

[2 Pencarian Akara pada Persamaan Nirlanjar 3](#_Toc514849576)

[3 Pencarian Solusi Sistem Persamaan Lanjar 6](#_Toc514849577)

[4 Interpolasi Polinom 9](#_Toc514849578)

[5 Integrasi Numerik 12](#_Toc514849579)

[6 Persamaan Diferensial Biasa 13](#_Toc514849580)

[7 Referensi 14](#_Toc514849581)

[*8 Source Code* 15](#_Toc514849582)

[8.1 Pencarian Akar pada Persamaan Nirlanjar 15](#_Toc514849583)

[8.2 Pencarian Solusi Sistem Persamaan Lanjar 17](#_Toc514849584)

[8.3 Interpolasi Polinom 20](#_Toc514849585)

[8.4 Integrasi Numerik 22](#_Toc514849586)

[8.5 Persamaan Diferensial Biasa 24](#_Toc514849587)

# Pendahuluan

Laporan ini berisi hasil implementasi, penjelasan dan analisis untuk penyelesaian berbagai macam persoalan numerik. Laporan ini dibuat sebagai syarat kelulusan untuk mata kuliah IF5162 Metode Numerik Lanjut. Setiap subbab selanjutnya akan berisi penjelasan mengenai hal-hal berikut.

1. Deskripsi persoalan numerik yang diselesaikan.
2. *Source code* yang sebenarnya berada di bab terpisah (bab 8)
3. *Screenshot* hasil eksekusi program berupa grafik
4. Diskusi dan analisis hasil.

# Pencarian Akara pada Persamaan Nirlanjar

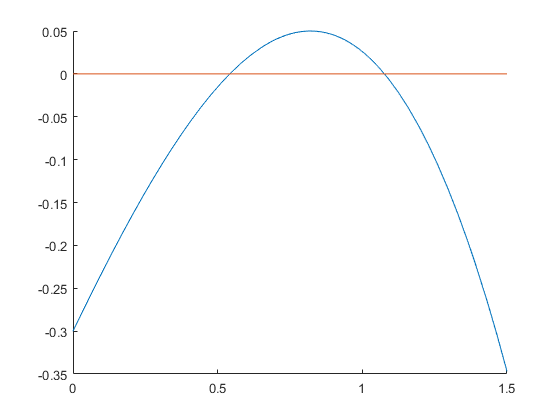
Ada dua persoalan yang harus dikerjakan pada permasalahan ini. Solusi sejati didapatkan dengan menggunakan Symbolab­[1].

1. Tentukan akar-akar pada persamaan (1) fungsi nirlanjar dari fungsi

didapatkan dua akar sejati dari persamaan tersebut yaitu

…

…



Gambar 2.1 Grafik persamaan (1)

1. Tentukan akar-akar pada persamaan (2) fungsi nirlanjar dari fungsi

didapatkan lima akara sejati dari persamaan tersebut yaitu

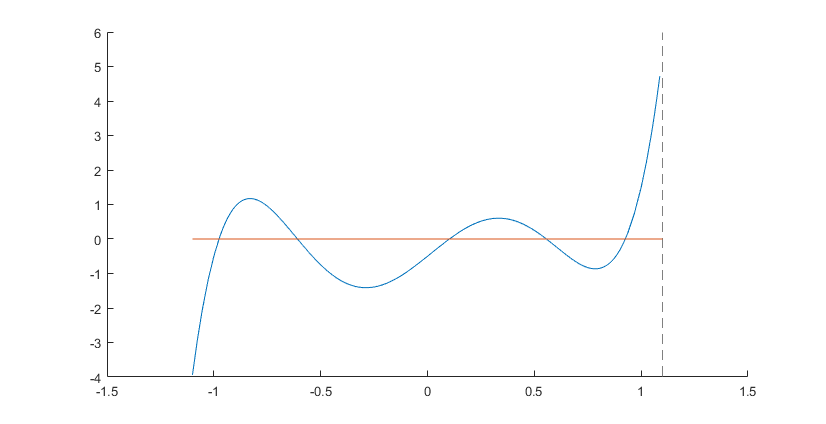
…

…

…

…

…



Gambar 2.2 Grafik persamaan (2)

Metode numerik yang digunakan pada permasalahan ini adalah metode bagidua dan metode Newton-Raphson.

Hasil akar-akar yang didapatkan jika menggunakan metode bagidua untuk persamaan (1).

|  |  |  |
| --- | --- | --- |
| **Batas awal** | **Batas akhir** | **Nilai akar** |
| 0,5 | 0,6 | 0.524999999254942 |
| 1 | 1,1 | 1.049999999254942 |

Hasil akar-akar yang didapatkan jika menggunakan metode bagidua untuk persamaan (2).

|  |  |  |
| --- | --- | --- |
| **Batas awal** | **Batas akhir** | **Nilai akar** |
| -1,2 | -0,8 | -1.000000000745058 |
| -0,8 | -0,4 | -0.700000000745058 |
| 0 | 0,3 | 0.074999999441206 |
| 0,3 | 0,7 | 0.499999999254942 |
| 0,7 | 1,0 | 0.849999999441206 |

Selanjutnya permasalahan diselesaikan dengan menggunakan metode Newton-Raphson. Untuk menggunakan metode ini diperlukan turunan dari persamaannya.

Turunan pertama dari persamaan (1) adalah sebagai berikut.

Turunan pertama dari persamaan (2) adalah sebagai berikut.

Selanjutnya adalah hasil akar-akar yang didapatkan jika menggunakan metode Newton-Raphson untuk persamaan (1).

|  |  |
| --- | --- |
| **Titik awal** | **Nilai akar** |
| 0 | 0.541948113962968 |
| 1 | 1.076464034742759 |

hasil akar-akar yang didapatkan jika menggunakan metode Newton-Raphson untuk persamaan (2).

|  |  |
| --- | --- |
| **Titik awal** | **Nilai akar** |
| -1 | -0.975831349590963 |
| -0.5 | -0.608611856725739 |
| 0 | 0.102140267149237 |
| 0.5 | 0.556352728918709 |
| 1 | 0.925950210248757 |

# Pencarian Solusi Sistem Persamaan Lanjar

Ada dua persoalan yang harus dikerjakan dalam permasalahan ini.

1. Tentukan solusi sistem persamaan lanjar pada persamaan (3) berikut.

Solusi sejati dari sistem persamaan lanjar persamaan (3) tersebut adalah sebagai berikut.

1. Tentukan solusi sistem persamaan lanjar berikut.

*A* merupakan matriks Hilbert berukuran 10x10, dengan yang setiap elemennya dapat dihitung gengan menggunakan rumus berikut.

yang dalam hal ini i merupakan nilai baris dan j merupakan nilai kolom, dengan indeks dimulai dari angka 1.

Matriks Hilbert tersebut bisa ditulis sebagai sebuah persamaan (4)

solusi sejati dari sistem persamaan lanjar (4) adalah sebagai berikut.

Metode numerik yang digunakan pada permasalahan ini adalah metode Gauss-Jordan, metode dekomposisi LU Gauss, dan metode lelaran Jacobi.

Hasil perhitungan solusi dari persamaan (3) dengan menggunakan metode Gauss-Jordan adalah sebagai berikut

Hasil perhitungan solusi dari persamaan (3) dengan menggunakan metode dekomposisi LU Gauss adalah sebagai berikut

Hasil perhitungan solusi dari persamaan (3) dengan menggunakan metode lelaran Jacobi adalah sebagai berikut

Selanjutnya adalah hasil perhitungan solusi dari persamaan (4) dengan menggunakan metode Gauss-Jordan adalah sebagai berikut

Hasil perhitungan solusi dari persamaan (4) dengan menggunakan metode dekomposisi LU Gauss adalah sebagai berikut

Hasil perhitungan solusi dari persamaan (4) dengan menggunakan metode lelaran Jacobi tidak membuahkan hasil karena matriks Hilbert tidak dominan secara diagonal sehingga lelarannya tidak konvergen.

# Interpolasi Polinom

Ada dua persoalan yang harus dikerjakan pada permasalahan ini.

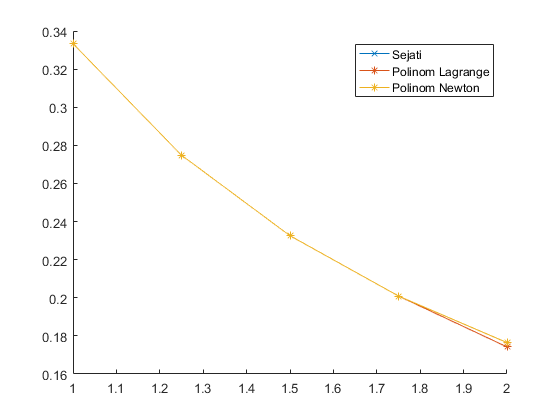
1. Gunakan metode interpolasi berderajat 4 dan 5 untuk menghitung nilai dari persamaan (5) berikut

pada selang [1, 2]

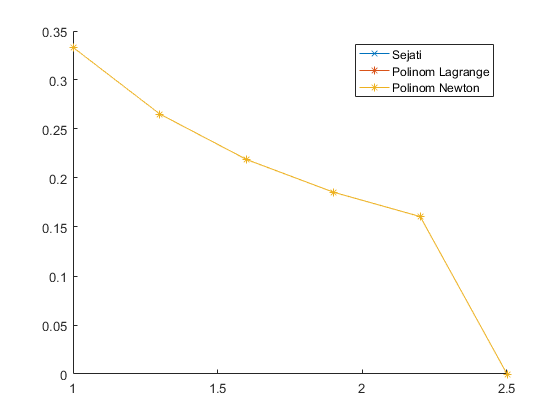
1. Gunakan metode interpolasi berderajat 5 untuk menghitung nilai dari fungsi yang memiliki nilai sesuai dengan tabel berikut.

|  |  |  |
| --- | --- | --- |
| **i** | **x** | **f(x)** |
| 1 | 0.5 | 1.143 |
| 2 | 1 | 1.000 |
| 3 | 1.5 | 0.828 |
| 4 | 2 | 0.667 |
| 5 | 2.5 | 0.533 |
| 6 | 3 | 0.428 |

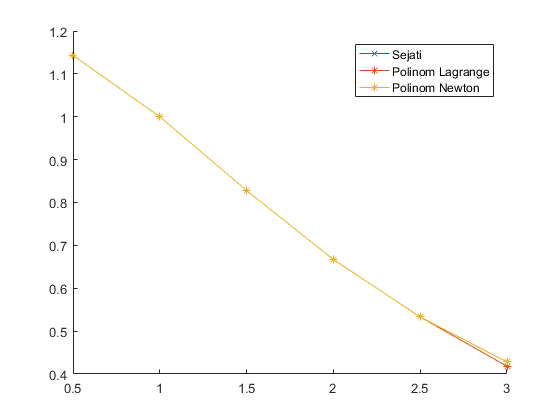
Metode numerik yang digunakan pada permasalahan ini adalah metode Lagrange dan metode Newton.



Gambar 4.1 Grafik hasil interpolasi untuk persoalan 3a dengan orde 4



Gambar 4.2 Grafik hasil interpolasi untuk persoalan 3a dengan orde 5



Gambar 4.3 Grafik hasil interpolasi untuk persoalan 3b dengan orde 5

Hasil galat dalam perhitungan interpolasi dengan x=1.7 untuk permasalahan 3a dengan menggunakan metode Lagrange orde 4.

Hasil galat dalam perhitungan interpolasi dengan x=1.7 untuk permasalahan 3a dengan menggunakan metode Lagrange orde 5.

Hasil galat dalam perhitungan interpolasi dengan x=1.7 untuk permasalahan 3a dengan menggunakan metode Newton orde 4.

Hasil galat dalam perhitungan interpolasi dengan x=1.7 untuk permasalahan 3a dengan menggunakan metode Newton orde 5.

# Integrasi Numerik

Ada dua persoalan yang harus dikerjakan pada permasalahan ini. Nilai sejati didapatkan dari Symbolab.

1. Hitunglah nilai dari integral untuk persamaan (6) berikut.

akan tetapi persamaan di atas bersifat singular sehingga harus diubah terlebih dahulu menjadi persamaan berikut.

solusi sejati yang didapatkan dari persamaan (6) adalah 3.1043790178550514369248958246317.

1. Hitunglah nilai dari integral untuk persamaan (7) berikut.

solusi sejati yang didapatkan dari persamaan (7) adalah 2.234314548392391.

Hasil perhitungan solusi dari persamaan (6) dengan menggunakan metode Trapesium dengan n=10 adalah 3.102709619182681.

Hasil perhitungan solusi dari persamaan (6) dengan menggunakan metode Simpson 1/3 dengan n=10 adalah 3.104389328409193.

Hasil perhitungan solusi dari persamaan (7) dengan menggunakan metode Trapesium dengan n=10 adalah 2.233977198685922.

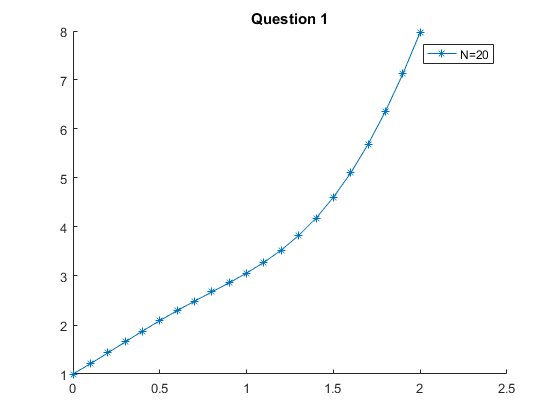
Hasil perhitungan solusi dari persamaan (7) dengan menggunakan metode Simpson 1/3 dengan n=10 adalah 2.234314095019442.

# Persamaan Diferensial Biasa

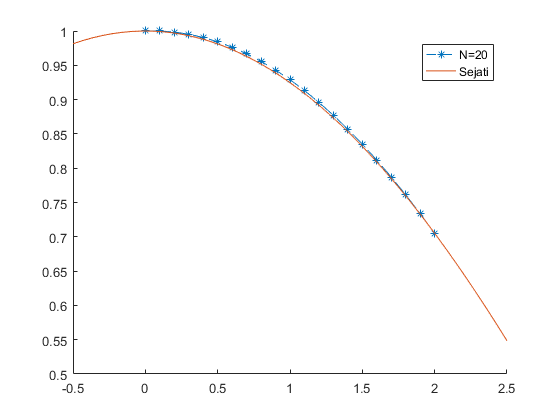
Ada dua persoalan yang harus dikerjakan dalam permasalahan ini.

1. Dengan menggunakan metode numerik, hitunglah y(2) dengan h=0.1 jika diketahui persamaan (8) berikut.
2. Dengan menggunakan metode numerik, hitunglah y(2) dengan h=0.1 jika diketahui persamaan (9) berikut.

Metode numerik yang digunakan adalah metode Runge-Kutta orde 3.



Gambar 6.1 Grafik lelaran Runge-Kutta orde 3 untuk persoalan 5a



Gambar 6.2 3 Grafik lelaran Runge-Kutta orde 3 untuk persoalan 5a

Hasil perhitungan solusi dari persamaan (8) dengan menggunakan metode Runge-Kutta orde 3 adalah 7.976395838110149.

Hasil perhitungan solusi dari persamaan (9) dengan menggunakan metode Runge-Kutta orde 3 adalah 0.704845560384812, dengan solusi eksaknya adalah 0.705050359540160.

# Referensi

1. Munir, Rinaldi. 2003. Metode Numerik. Bandung: Penerbit Informatika.

# *Source Code*

## Pencarian Akar pada Persamaan Nirlanjar

|  |
| --- |
| addpath(genpath('Functions'))    f = @(x) sin(x) - 0.3.\*exp(x);  f\_aksen = @(x) cos(x) - 0.3.\*exp(x);    hold on  fplot(f, [0, 1.5]);  fplot(@(x)0, [0, 1.5]);  hold off    akar1\_bagiDua = BagiDua(f, 0.5, 0.6);  akar2\_bagiDua = BagiDua(f, 1, 1.1);    akar1\_newtonRaphson = NewtonRaphson(f, f\_aksen, 0);  akar2\_newtonRaphson = NewtonRaphson(f, f\_aksen, 1); |
| addpath(genpath('Functions'))    f = @(x) 16.\*x.^5 - 20.\*x.^3 + x.^2 + 5.\*x - 0.5;  f\_aksen = @(x) 80.\*x.^4 - 60.\*x.^2 + 2.\*x + 5;    hold on  fplot(f, [-1.1, 1.1]);  fplot(@(x)0, [-1.1, 1.1]);  hold off    akar1\_bagiDua = BagiDua(f, -1.2, -0.8);  akar2\_bagiDua = BagiDua(f, -0.8, -0.4);  akar3\_bagiDua = BagiDua(f, 0, 0.3);  akar4\_bagiDua = BagiDua(f, 0.3, 0.7);  akar5\_bagiDua = BagiDua(f, 0.7, 1.0);    akar1\_newtonRaphson = NewtonRaphson(f, f\_aksen, -1);  akar2\_newtonRaphson = NewtonRaphson(f, f\_aksen, -0.5);  akar3\_newtonRaphson = NewtonRaphson(f, f\_aksen, 0);  akar4\_newtonRaphson = NewtonRaphson(f, f\_aksen, 0.5);  akar5\_newtonRaphson = NewtonRaphson(f, f\_aksen, 1); |
| function [ akar ] = BagiDua( f, a, b )  epsilon1 = 0.000000001;  while 1  c = (a + b) / 2;  if (f(a) \* f(b) < 0)  b = c;  else  a = c;  end    if (abs(a-b) < epsilon1)  break  end  end  akar = c;  end |
| function [ akar ] = NewtonRaphson( f, f\_aksen, x )  epsilon1 = 0.000000001;  epsilon2 = 0.000000001;  nmax = 30;    n = 0;  berhenti = false;  x\_prev = 0;  while 1  if (abs(f\_aksen(x)) < epsilon2)  berhenti = true;  else  x\_prev = x;  x = x - f(x)/f\_aksen(x);  n = n + 1;  end    if (abs(x-x\_prev) < epsilon1 || berhenti || n > nmax)  break  end  end    if (berhenti)  akar = Inf;  elseif n > nmax  akar = Inf;  else  akar = x;  end  end |

## Pencarian Solusi Sistem Persamaan Lanjar

|  |
| --- |
| addpath(genpath('Functions'))    A = [  8 1 3 2;  2 9 -1 -2;  1 3 2 -1;  1 0 6 4  ];    B = [1 2 3 4];    [AGauss, xGauss] = GaussJordan(A,B);    [L, U] = GaussLU(A);  y = ForwardSubstitution(L,B);  xLU = BackwardSubstitution(U,y);    xJacobi = Jacobi(A, B, [0 0 0 0]); |
| addpath(genpath('Functions'))    A = zeros(10,10);  B = [1 0 0 0 0 0 0 0 0 0];    n = size(A, 1);    for i = 1:n  for j = 1:n  A(i,j) = 1/(i+j-1);  end  end    [AGauss, xGauss] = GaussJordan(A,B);    [L, U] = GaussLU(A);  y = ForwardSubstitution(L,B);  xLU = BackwardSubstitution(U,y);    xJacobi = Jacobi(A, B, [0,0,0,0,0,0,0,0,0,0]); |
| function [ x ] = BackwardSubstitution( U, b )  n = size(U, 1);  x(n) = b(n) / U(n,n);  for i = n-1:-1:1  sigma = 0;  for j = i+1:n  sigma = sigma + U(i,j) \* x(j);  end  x(i) = (b(i) - sigma) / U(i,i);  end    end |
| function [ x ] = ForwardSubstitution( L, b )  n = size(L, 1);  x(1) = b(1) / L(1,1);  for i = 2:n  sigma = 0;  for j = 1:i-1  sigma = sigma + L(i,j) \* x(j);  end  x(i) = (b(i) - sigma) / L(i,i);  end    end |
| function [ A, x ] = GaussJordan( A, b )  n = size(A, 1);    for k = 1:n  tampung = A(k,k);  for j = 1:n  A(k,j) = A(k,j) / tampung;  end  b(k) = b(k) / tampung;  for i = 1:n  if i ~= k  m = A(i,k);  for j = 1:n  A(i,j) = A(i,j) - m \* A(k,j);  end  b(i) = b(i) - m \* b(k);  end  end  end    x = b;  end |
| function [ L, U ] = GaussLU( A )  n = size(A, 1);  I = eye(n);    for k = 1:n-1  for i = k+1:n  m = A(i,k) / A(k,k);  I(i,k) = m;  for j = k:n  A(i,j) = A(i,j) - m \* A(k,j);  end  end  end  L = I;  U = A;  end |
| function [ x ] = Jacobi( A, b, x )  n = size(A, 1);  epsilon = 0.0000001;  galat\_relatif = zeros(1, n);    if ~diagonallyDominant(A)  x = ones(1, n) .\* Inf;  return  end    while 1  xlama = x;  for i = 1:n  sigma = 0;  for j = 1:n  if i ~= j  sigma = sigma + A(i,j) \* xlama(j);  end  end  x(i) = (b(i) - sigma) / A(i,i);  galat\_relatif(i) = (x(i) - xlama(i)) / x(i);  end    if (sum(abs(galat\_relatif) < epsilon) == n)  return  end  end  end    function [ is\_it ] = diagonallyDominant( A )  is\_it = false;  n = size(A, 1);    for i = 1:n  sigma = 0;  for j = 1:n  if i ~= j  sigma = sigma + abs(A(i,j));  end  end  if abs(A(i,i)) < sigma  return  end  end  is\_it = true;  end |

## Interpolasi Polinom

|  |
| --- |
| addpath(genpath('Functions'))    f = @(x) (1 + x) / (1+2\*x+3\*x^2);    x = linspace(1,2,5);  y = zeros(1, 5);  for i = 1:5  y(i) = f(x(i));  end    hold on  plot(x,y, '-x');    ynew = zeros(1, 5);  for i = 1:5  ynew(i) = PolinomLagrange(x, y, x(i), 4);  end  plot(x, ynew, '-\*');    ynew = zeros(1, 5);  for i = 1:5  ynew(i) = PolinomNewton(x, y, x(i), 4);  end  plot(x, ynew, '-\*');    legend('Sejati', 'Polinom Lagrange', 'Polinom Newton')  hold off |
| addpath(genpath('Functions'))    x = [0.5 1 1.5 2 2.5 3];  y = [1.143 1 0.828 0.667 0.533 0.428];    hold on  plot(x,y, '-x');    ynew = zeros(1, 6);  for i = 1:6  ynew(i) = PolinomLagrange(x, y, x(i), 5);  end  plot(x, ynew, '-\*');    ynew = zeros(1, 6);  for i = 1:6  ynew(i) = PolinomNewton(x, y, x(i), 5);  end  plot(x, ynew, '-\*');    legend('Sejati', 'Polinom Lagrange', 'Polinom Newton')  hold off |
| function [ ynew ] = PolinomNewton( x, y, xnew, n )  ST = zeros(n+1);    % bikin ST dulu  for k = 1:n+1  ST(k,1) = y(k);  end  for k = 2:n+1  for i = 1:n+2-k  ST(i,k) = (ST(i+1,k-1) - ST(i,k-1)) / (x(i+k-1)-x(i));  end  end    ynew = ST(1,1);  for i = 2:n+1  suku = ST(1,i);  for k = 1:i-1  suku = suku \* (xnew - x(k));  end  ynew = ynew + suku;  end    end |
| function [ ynew ] = PolinomLagrange( x, y, xnew, n )  ynew = 0;  for i = 1:n  pi = 1;  for j = 1:n  if i ~= j  pi = pi \* (xnew - x(j)) / (x(i) - x(j));  end  end  ynew = ynew + y(i) \* pi;  end  end |

## Integrasi Numerik

|  |
| --- |
| addpath(genpath('Functions'))    f = @(u) (-2 \* exp(1-u^2)) / sqrt(2-u^2);  a = 1;  b = 0;    IT = IntegrasiTrapesium(f, a, b, 50);  ISimpson = SimpsonSepertiga(f, a, b, 500); |
| addpath(genpath('Functions'))    f = @(x, y) sqrt(x + y);  ax = 1;  bx = 2;  ay = 0;  by = 1.5;    IT = IntegrasiTrapesiumGanda(f, ax, bx, ay, by, 10);  ISimpson = SimpsonSepertigaGanda(f, ax, bx, ay, by, 10); |
| function [ I ] = IntegrasiTrapesium( f, a, b, n )  h = (b - a) / n;  x = a;  I = f(a) + f(b);  sigma = 0;  for r = 1:n-1  x = x + h;  sigma = sigma + 2 \* f(x);  end  I = (I + sigma) \* h / 2;  end |
| function [ I ] = IntegrasiTrapesiumGanda( f, ax, bx, ay, by, n )  h = (by - ay) / n;  y = ay;  I = IntegrasiTrapesium(f, ay, ax, bx, n) + IntegrasiTrapesium(f, by, ax, bx, n);  sigma = 0;  for r = 1:n-1  y = y + h;  sigma = sigma + 2 \* IntegrasiTrapesium(f, y, ax, bx, n);  end  I = (I + sigma) \* h / 2;  end    function [ I ] = IntegrasiTrapesium( f, y, a, b, n )  h = (b - a) / n;  x = a;  I = f(a, y) + f(b, y);  sigma = 0;  for r = 1:n-1  x = x + h;  sigma = sigma + 2 \* f(x, y);  end  I = (I + sigma) \* h / 2;  end |
| function [ I ] = SimpsonSepertiga( f, a, b, n )  h = (b - a) / n;  x = a;  I = f(a) + f(b);  sigma = 0;  for r = 1:n-1  x = x + h;  if mod(r, 2) == 1  sigma = sigma + 4 \* f(x);  else  sigma = sigma + 2 \* f(x);  end  end  I = (I + sigma) \* h / 3;  end |
| function [ I ] = SimpsonSepertigaGanda( f, ax, bx, ay, by, n )  h = (by - ay) / n;  y = ay;  I = SimpsonSepertiga(f, ay, ax, bx, n) + SimpsonSepertiga(f, by, ax, bx, n);  sigma = 0;  for r = 1:n-1  y = y + h;  if mod(r, 2) == 1  sigma = sigma + 4 \* SimpsonSepertiga(f, y, ax, bx, n);  else  sigma = sigma + 2 \* SimpsonSepertiga(f, y, ax, bx, n);  end  end  I = (I + sigma) \* h / 3;  end    function [ I ] = SimpsonSepertiga( f, y, a, b, n )  h = (b - a) / n;  x = a;  I = f(a, y) + f(b, y);  sigma = 0;  for r = 1:n-1  x = x + h;  if mod(r, 2) == 1  sigma = sigma + 4 \* f(x, y);  else  sigma = sigma + 2 \* f(x, y);  end  end  I = (I + sigma) \* h / 3;  end |

## Persamaan Diferensial Biasa

|  |
| --- |
| addpath(genpath('Functions'))    ut = @(x, y) cos(pi \* x) + y;    x = [5, 10, 20, 40];  y = zeros(1, length(x));    figure(1);  hold on;  for i = 1:length(x)  y(i) = RK3(ut, 0, 1, 2, 2/x(i));  end    title('Question 1')  legend('N=5', 'N=10','N=20', 'N=40')  hold off;    figure(2)  plot(x, y, '-o')  title('hasil'); |
| addpath(genpath('Functions'))    yt = @(x, y, z) z;  zt = @(x, y, z) 0.05\*z - 0.15\*y;    exact = @(x) exp(x ./ 40) .\* (cos((sqrt(239) .\* x) ./ 40) - ((1 ./ sqrt(239)) .\* sin((sqrt(239) .\* x) ./ 40)));  y\_exact = exact(2);    t = [10, 20, 40, 80, 160, 320, 640];  y = zeros(1, length(t));  z = zeros(1, length(t));  galat\_y = zeros(1, length(t));    figure(1)  hold on;  for i = 1:length(t)  [y(i), z(i)] = twoRK3(yt, zt, 0, 1, 0, 2, 2/t(i));  galat\_y(i) = y\_exact - y(i);  end    fplot(exact, [-0.5, 2.5]);    figure(2)  plot(t, galat\_y, '-o')  title('galat'); |
| function [ y\_RK3 ] = RK3( f, x0, y0, b, h )  % menghitung nilai y(b) pada PDB  % y'=f(x,y); y(x0)=y0  % dengan metode Runge-Kutta orde 3  n = (b-x0)/h;  x = zeros(1, uint8(n));  y = zeros(1, uint8(n));  x(1) = x0;  y(1) = y0;    for r = 1:n  k1 = h \* f(x(r),y(r));  k2 = h \* f(x(r) + h/2, y(r) + k1/2);  k3 = h \* f(x(r) + h, y(r) - k1 + 2\*k2);  y(r+1) = y(r) + (k1+4\*k2+k3)/6;  x(r+1) = x(r) + h;  end    y\_RK3 = y(n+1);  plot(x, y, '-\*');  end |
| function [ x\_RK3, y\_RK3 ] = twoRK3( f1, f2, t0, x0, y0, b, h )  % menghitung nilai x(b) dan y(b) pada PDB  % x'=f(t,x,y); x(t0)=x0  % y'=f(t,x,y); y(t0)=y0  % dengan metode Runge-Kutta orde 3  n = (b-t0)/h;  t = zeros(1, uint8(n));  x = zeros(1, uint8(n));  y = zeros(1, uint8(n));  t(1) = t0;  x(1) = x0;  y(1) = y0;    for r = 1:n  k1 = h \* f1(t(r), x(r), y(r));  l1 = h \* f2(t(r), x(r), y(r));  k2 = h \* f1(t(r) + h/2, x(r) + k1/2, y(r) + k1/2);  l2 = h \* f2(t(r) + h/2, x(r) + l1/2, y(r) + l1/2);  k3 = h \* f1(t(r) + h, x(r) - k1 + 2\*k2, y(r) - k1 + 2\*k2);  l3 = h \* f2(t(r) + h, x(r) - l1 + 2\*l2, y(r) - l1 + 2\*l2);    x(r+1) = x(r) + (k1+4\*k2+k3)/6;  y(r+1) = y(r) + (l1+4\*l2+l3)/6;    t(r+1) = t(r) + h;  end    x\_RK3 = x(n+1);  y\_RK3 = y(n+1);  plot(t, x, '--\*');  end |