Tugas Besar TF2202

Selesaikan soal-soal berikut ini dengan menggunakan Scilab.

1 Perbandingan akurasi beberapa metode untuk ODE

Carilah solusi numerik dari persamaan diferensial berikut

$$y''(t) + y(t) = 0 \tag{1}$$

dengan syarat awal

$$y(0) = 0, \quad y'(0) = 1 \tag{2}$$

pada interval $0 \leq t \leq 10$.
Bandingkan solusi yang diperoleh dengan solusi analitik:

$$y(t) = \sin(t) \tag{3}$$

Gunakan menggunakan metode-metode berikut ini untuk mencari solusi numeriknya.

- Euler
- Euler dengan prediktor-korektor (Runge-Kutta orde-2)
- Runge-Kutta orde-4

Dari solusi numerik yang didapatkan, buatlah (1) plot antara y dan y' dan (2) plot antara t dan y.

Solusi

Metode Euler file (file: `ode_euler.sce`)

```
function [t,y] = ode_euler(f,tspan,y0,N)

  if (~exists("N", "local")) | (N <= 0)
    N = 100
  end

  if ~exists("tspan","local")
    y0 = 0
  end

  h = (tspan(2) - tspan(1))/N
  t = tspan(1) + [0:N]'*h

  // check y0, transpose if needed
  if size(y0,1) ~= 1
    if size(y0,2) > 1
      y0 = y0'
    end
  end

  Ndim = size(y0,2)
  y = zeros(N+1,Ndim)

  // initial value
  y(1,:) = y0

  // Euler's algorithm here
  for k = 1:N
```

```
    y(k+1,:) = y(k,:) + h*f(t(k),y(k,:))
  end

endfunction
```

Metode Euler dengan prediktor-korektor: (file: `ode_euler_PC.sce`)

```
function [t,y] = ode_euler_PC(f,tspan,y0,N)

  if (~exists("N", "local")) | (N <= 0)
    N = 100
  end

  if ~exists("tspan","local")
    y0 = 0
  end

  h = (tspan(2) - tspan(1))/N
  t = tspan(1) + [0:N]'*h

  // check y0, transpose if needed
  if size(y0,1) ~= 1
    if size(y0,2) > 1
      y0 = y0'
    end
  end

  Ndim = size(y0,2)
  y = zeros(N+1,Ndim)

  // initial value
  y(1,:) = y0

  for k = 1:N
    yk1 = y(k,:) + h*f( t(k), y(k,:) )
    y(k+1,:) = y(k,:) + 0.5*h*( f(t(k),y(k,:)) + f(t(k)+h,yk1) )
  end

endfunction
```

Metode Runge-Kutta orde 4 (file: `ode_RK4.sce`)

```
function [t,y] = ode_RK4(f,tspan,y0,N)
  //
  if (~exists("N", "local")) | (N <= 0)
    N = 100
  end
  //
  if ~exists("tspan","local")
    y0 = 0
  end
  // check y0, transpose if needed
  if size(y0,1) ~= 1
    if size(y0,2) > 1
      y0 = y0'
    end
  end
  //
  Ndim = size(y0,2)
  y = zeros(N+1,Ndim)
  //
  y(1,:) = y0
  h = (tspan(2) - tspan(1))/N
```

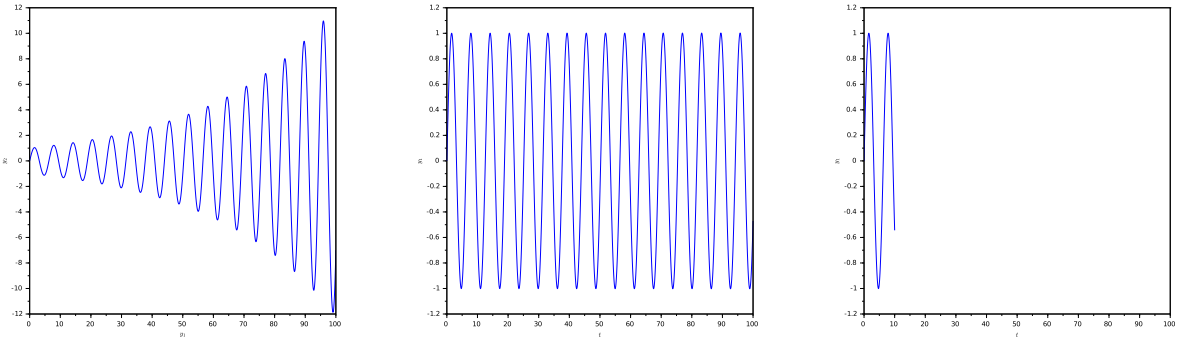
```
  t = tspan(1) + [0:N]'*h
  h2 = h/2
  // Runge-Kutta 4th-order algorithm here
  for k=1:N
    f1 = h*f(t(k),y(k,:))
    f2 = h*f(t(k)+h2,y(k,:)+f1/2)
    f3 = h*f(t(k)+h2,y(k,:)+f2/2)
    f4 = h*f(t(k)+h,y(k,:)+f3)
    y(k+1,:) = y(k,:) + (f1 + 2*(f2+f3) + f4)/6
  end
endfunction
```

Contoh hasil solusi:

```
exec("ode_euler.sce",-1)
exec("ode_euler_PC.sce",-1)
exec("ode_RK4.sce",-1)

// definisi ODE
function f = dy(t,y)
  f(1) =  y(2)
  f(2) = -y(1)
  f = f'
endfunction

tspan = [0 10]
y0 = [0 1]

method = "RK4" // pilih metode

h = 0.05
N = (tspan(2) - tspan(1))/h

if method == "RK4"
  [t,y] = ode_RK4(dy,tspan,y0,N)
elseif method == "euler"
  [t,y] = ode_euler(dy,tspan,y0,N)
elseif method == "euler_PC"
  [t,y] = ode_euler_PC(dy,tspan,y0,N)
else
  error("method is unknown")
end

clf()
plot( y(:,1), y(:,2), 'b' )
// xmin, ymin, xmax, ymax
xlabel('$y_1$')
ylabel('$y_2$')
if method == "euler"
  square(-15,-15,15,15)
  xs2pdf( gcf(), "images/soal_01_ode_euler_y1_y2.pdf" )
elseif method == "euler_PC"
  square(-1.5,-1.5,1.5,1.5)
  xs2pdf( gcf(), "images/soal_01_ode_euler_PC_y1_y2.pdf" )
elseif method == "RK4"
  square(-1.5,-1.5,1.5,1.5)
  xs2pdf( gcf(), "images/soal_01_ode_RK4_y1_y2.pdf" )
end

clf()
plot( t, y(:,1), 'b')
xlabel('$t$')
ylabel('$y_1$')
```

```
if method == "euler"
  xs2pdf( gcf(), "images/soal_01_ode_euler_t_y1.pdf")
elseif method == "euler_PC"
  set(gca(),"data_bounds",[0,100,-1.2,1.2])
  xs2pdf( gcf(), "images/soal_01_ode_euler_PC_t_y1.pdf")
elseif method == "RK4"
  set(gca(),"data_bounds",[0,100,-1.2,1.2])
  xs2pdf( gcf(), "images/soal_01_ode_RK4_t_y1.pdf")
end
```

Hasil visualisasi solusi



Gambar 1: Hasil solusi numerik x vs y_1, berturut-turut dari kiri ke kanan: Euler, Euler dengan prediktor-korektor dan Runge-Kutta orde 4

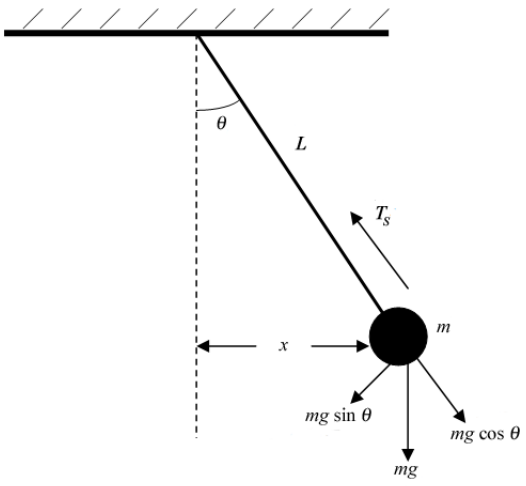
2 Gerakan pendulum

Gerak suatu pendulum dapat dinyatakan dengan persamaan diferensial:

$$\theta''(t) = -\frac{g}{L}\sin(\theta(t)) - k\theta'(t) \tag{4}$$

dengan syarat awal

$$\theta(0) = \theta_0, \qquad \theta'(0) = 0 \tag{5}$$



Gambar 2: Pendulum sederhana

Dalam persamaan tersebut θ menyatakan simpangan pendulum, θ_0 menyatakan simpangan awal pendulum, g menyatakan percepatan gravitasi, L menyatakan panjang benang pendulum, dan $k\theta'$ menyatakan suku redaman (gesekan) yang berbanding lurus dengan kecepatan θ' (k adalah bilangan positif).

(a) Cari solusi $\theta(t)$ untuk kasus $k = 0$ untuk simpangan awal $\theta_0 = 0.1, \pi/5$ dan 3.0.

(b) Masih untuk kasus $k = 0$, tentukan periode osilasi T sebagai fungsi dari simpangan awal u_0.

(c) Carilah solusi $u(t)$ untuk kasus $k = 0.1, 0.3, 0.5$ dan 0.8 untuk simpangan awal yang sama.

3 Persamaan Schrodinger (metode shooting)

Persamaan Schrodinger independen-waktu pada 1 dimensi dapat dinyatakan sebagai berikut.

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}\psi}{\mathrm{d}x^2} + V(x)\psi(x) = E\psi(x) \tag{6}$$

Dengan menggunakan unit atomik, kita dapat mengambil $\hbar = 1$ dan $m = 1$, dan persamaan (6) dapat ditulis menjadi:

$$\psi'' = 2\left[V(x) - E\right]\psi \tag{7}$$

Untuk solusi keadaan terikat (bound states), nilai E hanya dapat memiliki nilai yang diskrit. Selain itu, untuk bound states fungsi gelombang dibatasi dengan syarat

$$\lim_{x\to\infty}\psi(x) = 0, \qquad \lim_{x\to-\infty}\psi(x) = 0 \tag{8}$$

Selain itu, fungsi gelombang biasanya juga dinyatakan dalam bentuk ternormalisasi:

$$\int_{-\infty}^{\infty}\psi^*(x)\psi(x)\,\mathrm{d}x = 1 \tag{9}$$

Untuk potensial yang simetrik terhadap $x = 0$, secara matematis dapat ditulis $V(-x) = V(x)$. Contoh potential simetrik yang akan dibahas adalah potensial harmonik

$$V(x) = \frac{1}{2}x^2 \tag{10}$$

Dengan demikian, kita bisa mendapatkan solusi pada $(-\infty, \infty)$ hanya dengan solusi pada $(0, \infty)$. Ingat lagi dari kuliah mekanika kuantum bahwa, untuk potensial harmonik, nilai E yang diperbolehkan adalah:

$$E = \frac{(n+1)}{2}, \qquad n = 0, 1, 2, \ldots \tag{11}$$

Kita dapat mengelompokkan solusi ini menjadi dua kelompok:

- solusi ganjil ($n = 1, 3, 5, \ldots$), dengan fungsi gelombang $\psi(x) = -\psi(-x)$
- solusi genap ($n = 0, 2, 4, \ldots$), dengan fungsi gelombang $\psi(x) = \psi(-x)$

Persamaan (7) dapat diselesaikan dengan menggunakan metode standard seperti Runge-Kutta orde-4. Metode lain yang sering digunakan adalah metode Numerov. Metode ini biasa digunakan untuk menyelesaikan persamaan diferensial orde dua tanpa suku dengan turunan pertama, seperti pada persamaan (7). Metode ini dapat dituliskan dalam bentuk:

$$u_m = 1 - \frac{1}{6}h^2\left[V(x_m) - E\right] \tag{12}$$

$$\psi_{m+1} = \frac{(12 - 10u_m)\,\psi_m - u_{m-1}\psi_{m-1}}{u_{m+1}} \tag{13}$$

Untuk mengaplikasikan metode shooting, persamaan nilai batas harus dikonversi menjadi permasalahan syarat awal:

$$\psi(0) = \psi_0, \qquad \psi'(0) = 0, \qquad n = 0, 2, 4, \ldots \tag{14}$$

$$\psi(0) = 0, \qquad \psi'(0) = \psi_0', \qquad n = 1, 3, 5, \ldots \tag{15}$$

Nilai dari ψ_0 dan ψ_0' dapat dipilih dengan nilai sembarang bukan nol. Untuk soal ini kita akan ambil $\psi_0 = 1$ dan $\psi_0' = 1$.

(a) Buatlah program untuk menghitung solusi numerik dari persamaan 7 dengan menggunakan metode Numerov. Perhatikan bahwa untuk mendapatkan nilai ψ_{m+1} kita memerlukan dua nilai sebelumnya, yaitu ψ_m dan ψ_{m-1}, sedangkan dari deskripsi masalah kita biasanya hanya memiliki informasi nilai awal ψ_0 dan ψ_0'. Untuk mencari nilai ψ_1 kita dapat menggunakan metode Euler-Cromer:

$$\psi_1 = \psi_0 + \psi_0'\Delta x + 2(V(x_0) - E)(\Delta x)^2 \tag{16}$$

Gunakan nilai solusi analitik $E = 0.5$ dan disekitarnya misalnya $E = 0.499$ dan 0.501 serta bandingkan hasilnya numerik dari ψ yang didapatkan dengan ψ eksak, yaitu $\psi = \exp(-x^2/2)$. Anda dapat mengaproksimasi solusi pada selang $0 \le x < \infty$ dengan solusi selang $0 < x < 5$, misalnya dengan asumsi bahwa fungsi gelombang eksak telah memiliki nilai nol pada $x = 5$. Gunakan juga syarat awal yang sesuai untuk $E = 0.5$ (solusi genap), yaitu $\psi_0 = 1$ dan $\psi_0' = 0$.

(b) Gunakan metode shooting dengan cara memvariasikan nilai E untuk mencari nilai E yang mungkin dalam rentang $0 \le E \le 15$. Gunakan syarat awal yang sesuai untuk solusi ganjil dan genap.

4 Persamaan Schrodinger (nilai eigen)

Metode alternatif yang dapat digunakan untuk menyelesaikan persamaan Schrodinger independen-waktu adalah dengan menyelesaikan persamaan eigenvalue

$$\left[-\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V(x)\right]\psi(x) = E\psi(x) \tag{17}$$

$$\hat{H}\psi(x) = E\psi(x) \tag{18}$$

Nilai-nilai E yang mungkin dapat dicari dengan cara menghitung nilai eigen dari matriks Hamiltonian

$$H_{ij} = K_{ij} + V_{ij} \tag{19}$$

Dengan K_{ij} dan V_{ij} adalah representasi matriks dari operator kinetik dan potensial. Fungsi eigen terkait adalah fungsi gelombang solusi dari persamaan (7).
Suatu selang $-L < x < L$ dengan L suatu bilangan positif dibagi-bagi kedalam $N-1$ partisi dengan jumlah total titik N, yaitu x_i, $i = 1, 2, \ldots, N$. Potensial $V(x)$ dapat dihitung pada x_i sehingga kita peroleh representasi diskrit dari potensial: $V_i = V_(x_i)$.
Matriks V_{ij} adalah matriks diagonal yang elemen diagonalnya adalah nilai dari $V(x)$ pada titik x_i.
Matriks K_{ij} dapat dapat diaproksimasi dengan cara mengaproksimasi operator $\mathrm{d}^2/\mathrm{d}x^2$ dengan menggunakan beda hingga.

(a) Gunakan beda hingga sentral 3 titik untuk mengaproksimasi matriks K_{ij}

(b) Hitung matriks Hamiltonian dan selesaikan persamaan eigen $H\psi = E\psi$. Anda dapat menggunakan perintah `[psi,D] = spec(H); E = diag(D)` pada Scilab. Bandingkan hasil yang Anda peroleh dengan solusi analitik.

(c) Ulangi (a) dan (b) dengan menggunakan beda hingga sentral 5, 7, dan 9 titik. Bandingkan hasilnya dengan menggunakan Δx yang sama.

5 Persamaan Poisson 2D

Hitung solusi numerik dari persamaan Laplace:

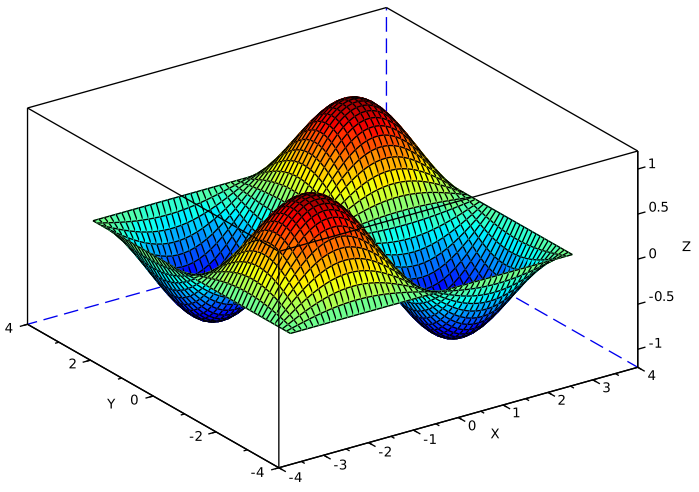
$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = \cos(x+y) - \sin(x-y) \tag{20}$$

pada domain rektangular $-\pi < x < \pi$, $-\pi < y < \pi$ dengan syarat batas

$$u(\pm\pi, y) = 0, \qquad u(x, \pm\pi) = 0 \tag{21}$$

dengan menggunakan metode beda hingga. Gunakan metode Gauss-Seidel untuk menyelesaikan sistem persamaan linear yang dihasilkan. Bandingkan solusi numerik yang diperoleh dengan solusi analitik

$$u(x,y) = \sin(x)\cos(y) \tag{22}$$



Gambar 3: Solusi persamaan Laplace $u(x,y)$

Solusi

Fungsi untuk menyelesaikan persamaan Poisson 2d dengan metode Gauss-Seidel

```
function u = poisson2d(u0, x, y, Nx, Ny, TOL, f)
// Nx, Ny: no of nodes in x and y direction

  MaxIter = 10000 // no. iterative steps

  hx = (x(Nx) - x(1))/(Nx-1)
  kx = 1.0/(hx*hx)

  hy = (y(Ny) - y(1))/(Nx-1)
  ky = 1.0/(hy*hy)

  kxy = 2.0*(kx + ky)

  // Gauss-Seidel algorithm here
  u = u0
  for iter =1:MaxIter
    err = 0.0
    u_old = u
    // loop only for internal nodes
    for j = 2:Ny-1
      for i = 2:Nx-1
        u(i,j) = ( kx*( u(i-1,j) + u(i+1,j) ) + ...
                   ky*( u(i,j-1) + u(i,j+1) ) - f(i,j) ) / kxy
      end
    end
    // calculate error
    err = sum(abs(u - u_old))
    printf("iter = %8d, err = %18.10e\n", iter, err)
    if err < TOL
      printf("Convergence is achieved\n")
      break
    end
  end

endfunction
```

Test program:

```
exec("poisson2d.sce",-1)

// RHS of the equation
function u = func(x,y)
  u = cos(x+y) - cos(x-y)
endfunction

Nx = 50
Ny = 50
x = linspace(-%pi,%pi,Nx)
y = linspace(-%pi,%pi,Ny)

u0 = zeros(Nx,Ny)
// set BC
u0(1,:) = 0.0
u0(:,1) = 0.0
u0(Nx,:) = 0.0
u0(:,Ny) = 0.0

// calculate array for RHS
f = zeros(Nx,Ny)
```

```
for j = 1:Ny
  for i = 1:Nx
    f(i,j) = func( x(i), y(j) )
  end
end

u = poisson2d(u0, x, y, Nx, Ny, 1e-5, f)

surf(x,y,u)
set(gcf(),"color_map",jetcolormap(32))
//colorbar(min(u),max(u))
xs2pdf(gcf(),"poisson2d.pdf")
```

6 Persamaan transfer kalor

Hitung solusi numerik dari

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \tag{23}$$

untuk $0 \leq x \leq 1$ dan $0 \leq t \leq 0.1$ dengan syarat awal

$$u(x,0) = \sin(\pi x) \tag{24}$$

dan syarat batas

$$u(0,t) = 0, \qquad u(1,t) = 0 \tag{25}$$

Gunakan metode Euler eksplisit, Euler implisit dan Crank-Nicolson serta bandingkan hasilnya. Buat juga animasi (dalam format GIF) yang menggambarkan perubatan profil suhu terhadap waktu.

Solusi

Metode Euler eksplisit:

```
function [u,x,t] = heat_1d_euler_exp(a,xf,T,initialTemp,bx0,bxf,Nx,Nt)

  dx = xf/Nx
  x  = [0:Nx]'*dx

  dt = T/Nt
  t  = [0:Nt]*dt

  // Set initial condition
  for i = 1:Nx+1
    u(i,1) = initialTemp(x(i))
  end

  for it = 1:Nt+1
    u([1 Nx+1],it) = [bx0(t(it)); bxf(t(it))]
  end

  r  = a*dt/dx/dx
  r1 = 1 - 2*r

  if r > 0.5
    printf("\nheat_1d_euler_exp:\n")
    printf("WARNING: r is larger than 0.5: %f\n", r)
    printf("WARNING: solution is not stable\n\n")
  else
    printf("\nheat_1d_euler:\n")
    printf("r = %f >= 0.5\n", r)
    printf("The solution should be stable\n\n")
  end

  for it = 1:Nt
```

```
    for i = 2:Nx
      u(i,it+1) = r*( u(i+1,it) + u(i-1,it) ) + r1*u(i,it)
    end
  end

endfunction
```

Metode Euler implisit

```
function [u,x,t] = heat_1d_euler_imp(a,xf,T,initialTemp,bx0,bxf,Nx,Nt)

  dx = xf/Nx
  x  = [0:Nx]'*dx

  dt = T/Nt
  t  = [0:Nt]*dt

  for i = 1:Nx+1
    u(i,1) = initialTemp(x(i))
  end

  for it = 1:Nt+1
    u([1 Nx+1],it) = [bx0(t(it)); bxf(t(it))]
  end

  r  = a*dt/dx/dx
  r2 = 1 + 2*r

  // Build matrix A
  for i = 1:Nx-1
    A(i,i) = r2
    if i > 1
      A(i-1,i) = -r
      A(i,i-1) = -r
    end
  end

  // time-stepping, solve linear equation
  for k=2:Nt+1
    b = [r*u(1,k); zeros(Nx-3,1); r*u(Nx+1,k)] + u(2:Nx,k-1);
    u(2:Nx,k) = A\b
  end

endfunction
```

Metode Crank-Nicolson

```
function [u,x,t] = heat_1d_CN(a,xf,T,initialTemp,bx0,bxf,Nx,Nt)

  dx = xf/Nx
  x  = [0:Nx]'*dx

  dt = T/Nt
  t  = [0:Nt]*dt

  for i = 1:Nx+1
    u(i,1) = initialTemp(x(i))
  end

  for it = 1:Nt+1
    u([1 Nx+1],it) = [bx0(t(it)); bxf(t(it))]
  end
```

```
  r  = a*dt/dx/dx
  r1 = 2*(1-r)
  r2 = 2*(1+r)

  // Build matrix A
  A = zeros(Nx-1,Nx-1)
  for i = 1:Nx-1
    A(i,i) = 2*(1 + r)
    if i > 1
      A(i-1,i) = -r
      A(i,i-1) = -r
    end
  end

  // Build matrix B
  B = zeros(Nx-1,Nx-1)
  for i = 1:Nx-1
    B(i,i) = 2*(1 - r)
    if i > 1
      B(i-1,i) = r
      B(i,i-1) = r
    end
  end

  // Time-steping, solve linear equation
  for it = 2:Nt+1
    b = B*u(2:Nx,it-1)
    u(2:Nx,it) = A\b
  end

  endfunction
```

Contoh pemanggilan fungsi:

```
exec("to_string.sce",-1)
exec("heat_1d_euler_exp.sce",-1)
exec("heat_1d_euler_imp.sce",-1)
exec("heat_1d_CN.sce",-1)

// initial condition (function of x)
function T = it0(x)
  T = sin(%pi*x)
endfunction

// boundary condition (function of t)
function T = bx0(t)
  T = 0.0
endfunction

function T = bxf(t)
  T = 0.0
endfunction

function T = analytic_solution(x,t)
  T = sin(%pi*x)*exp(-%pi*%pi*t)
endfunction

function plot_to_png(u,x,t,prefix)
  Nt = length(t)-1
  for it = 1:Nt+1
    clf()
    plot(x,u(:,it))
```

```
    set(gca(), "data_bounds", [0,1,0,1])
    strt = "t = " + string(t(it))
    xstring(0.8,0.9,strt)
    xs2png(gcf(), prefix + to_string(it) + ".png")
    printf("Done output solution for t = %f\n", t(it))
  end
endfunction

a = 1

xf = 1
Nx = 25

T  = 0.1
Nt = 100

// Explicit Euler
[u1,x,t] = heat_1d_euler_exp( a, xf, T, it0, bx0, bxf, Nx, Nt )
plot_to_png(u1,x,t,"TEMP_exp_")

// Using implicit Euler method
[u2,x,t] = heat_1d_euler_imp( a, xf, T, it0, bx0, bxf, Nx, Nt )
plot_to_png(u2,x,t,"TEMP_imp_")

// Using Crank-Nicholson method
[u3,x,t] = heat_1d_CN( a, xf, T, it0, bx0, bxf, Nx, Nt )
plot_to_png(u3,x,t,"TEMP_CN_")

NxNt = Nx*Nt
u_analytic = analytic_solution(x,t)

//How far from the analytical solution?
err1 = norm((u1-u_analytic))/NxNt
err2 = norm((u2-u_analytic))/NxNt
err3 = norm((u3-u_analytic))/NxNt

printf("err1 = %f\n", err1)
printf("err2 = %f\n", err2)
printf("err3 = %f\n", err3)
```

7 Persamaan gelombang

Selesaikan PDE berikut dengan menggunakan metode beda hingga. Buat juga visualisasi solusi berupa animasi dari solusi yang didapatkan.

(a) Hitung solusi numerik persamaan gelombang 1d

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} \tag{26}$$

untuk $0 < x < 1$ dan $0 \leq t \leq 1$ dengan syarat awal

$$u(x,0) = \sin(2\pi x)$$
$$u'(x,0) = 0$$

dan syarat batas:

$$u(0,t) = 0, \qquad u(1,t) = 0$$

(b) Selesaikan persamaan gelombang 2d

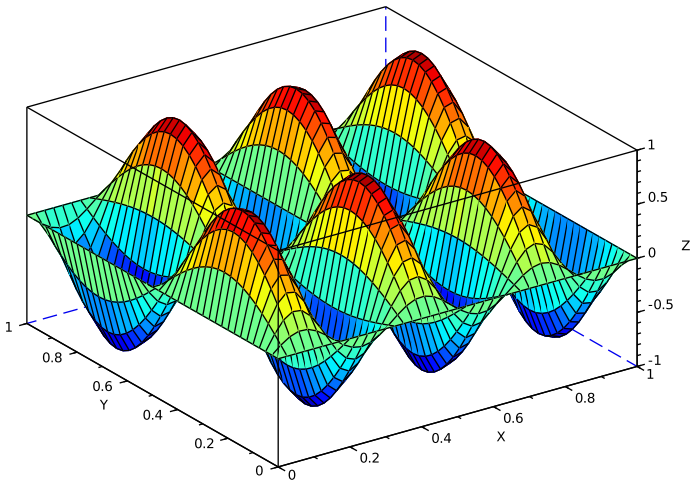
$$\frac{1}{4}\left(\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2}\right) = \frac{\partial^2 u(x,y,t)}{\partial t^2} \tag{27}$$

pada domain rektangular dengan $0 < x < 1$ dan $0 < y < 1$ serta pada selang waktu $0 < t < 2$ dengan syarat awal

$$u(x,y,0) = \sin(6\pi x)\sin(2\pi y)$$

dan syarat batas:

$$u(0,y,t) = 0, \quad u(x,0,t) = 0$$
$$u(1,y,t) = 0, \quad u(x,1,t) = 0$$



Gambar 4: Syarat awal $u(x,y,0)$

Solusi persmaan gelombang 1d

Persamaan gelombang 1d dengan metode beda hingga

```
function [u,x,t] = wave_1d(a,xf,tf,it0,i1t0,bx0,bxf,Nx,Nt)

  dx = xf/Nx
  x = [0:Nx]'*dx

  dt = tf/Nt
  t = [0:Nt]*dt

  r = a*(dt/dx)^2
  r1 = r/2
  r2 = 2*(1 - r)
  if r > 1
    printf("WARNING: propagation will not be stable\n\n")
  end

  // initial conditions
  for i = 1:Nx+1
    u(i,1) = it0(x(i))
  end

  // boundary conditions
  for k = 1:Nt+1
    u(1,k)    = bx0(t(k))
    u(Nx+1,k) = bxf(t(k))
  end

  u(2:Nx,2) = r1*u(1:Nx-1,1) + (1-r)*u(2:Nx,1) + r1*u(3:Nx+1,1) + dt*i1t0(x(2:Nx))

  for k = 3:Nt+1
    u(2:Nx,k) = r*u(1:Nx-1,k-1) + r2*u(2:Nx,k-1) + r*u(3:Nx+1,k-1) - u(2:Nx,k-2)
  end

endfunction
```

Contoh pemanggilan fungsi:

```
exec("wave_1d.sce",-1)
exec("to_string.sce",-1)

// initial condition
function y = it0(x)
  //y = x.*(1-x)
  y = sin(2*%pi*x)
endfunction

function y = i1t0(x)
  y = 0
endfunction

function y = bx0t(t)
  y = 0
endfunction

function y = bxft(t)
  y = 0
endfunction

a = 1
xf = 1
M = 100
T = 1
N = 100;
[u,x,t] = wave_1d(a,xf,T,it0,i1t0,bx0t,bxft,M,N);

for n = 1:N
  clf()
  plot(x,u(:,n))
  set(gca(),"data_bounds", [0 xf -1 1])
  xs2png(gcf(), "TEMP_wave_" + to_string(n) + ".png")
end
```

Dari file `TEMP_wave_*` dapat dibuat file animasi dengan dalam format GIF.

Solusi persamaan gelombang 2d

```
function [u,x,y,t] = wave_2d(a,D,T,it0,i1t0,bxyt,Mx,My,N)

  dx = ( D(2) - D(1) )/Mx
  x  = D(1) + [0:Mx]*dx

  dy = ( D(4) - D(3) )/My
  y  = D(3) + [0:My]'*dy
  dt = T/N
  t  = [0:N]*dt

  // Initialization
  u  = zeros(My+1,Mx+1)
  ut = zeros(My+1,Mx+1)

  for j=2:Mx
    for i=2:My
      u(i,j)  = it0(x(j),y(i))
      ut(i,j) = i1t0(x(j),y(i))
    end
  end

  adt2 = a*dt*dt
  rx = adt2/(dx*dx)
```

```
ry = adt2/(dy*dy)
rxy1 = 1 - rx - ry
rxy2 = rxy1*2

u_1 = u

for k = 0:N

  t = k*dt

  // should not have spatial (x and y) dependence
  for i = 1:My+1
    u(i,1) = bxyt(x(1),y(i),t)
    u(i,Mx+1) = bxyt(x(Mx+1),y(i),t)
  end

  for j = 1:Mx+1
    u(1,j) = bxyt(x(j),y(1),t)
    u(My+1,j)= bxyt(x(j),y(My+1),t)
  end

  if k==0 // starting condition

    for i=2:My
      for j=2:Mx
        u(i,j) = 0.5*(rx*(u_1(i,j-1) + u_1(i,j+1)) + ...
                 ry*(u_1(i-1,j) + u_1(i+1,j))) + rxy1*u(i,j) + dt*ut(i,j)
      end
    end

  else // propagation

    for i=2:My
      for j=2:Mx
        u(i,j) = rx*(u_1(i,j-1) + u_1(i,j+1)) + ...
                 ry*(u_1(i-1,j) + u_1(i+1,j)) + rxy2*u(i,j) - u_2(i,j)
      end
    end

  end

  u_2 = u_1
  u_1 = u

  // printing stuff's
  clf()
  surf(x,y,u)
  set(gca(), "data_bounds", [0 1 0 1 -1 1])
  set(gcf(),"color_map",jetcolormap(32))
  f = gcf()
  f.figure_size = [800,1200]
  fc = f.children
  fc.rotation_angles = [77.75,-161.75]
  if k == 0
    xs2pdf(gcf(),"TEMP_wave2d_" + to_string(k) + ".pdf")
    //xs2png(gcf(),"TEMP_wave2d_" + to_string(k) + ".png")
  else
    //xs2png(gcf(),"TEMP_wave2d_" + to_string(k) + ".png")
    xs2pdf(gcf(),"TEMP_wave2d_" + to_string(k) + ".pdf")
  end

end
```

```
endfunction
```

Contoh pemanggilan fungsi

```
exec("to_string.sce",-1)
exec("wave_2d.sce",-1)

function z = it0(x,y)
  kx = 2*%pi
  ky = 2*%pi
  z = sin(3*kx*x)*sin(ky*y)
endfunction

function z = i1t0(x,y)
  z = 0.0
endfunction

function z = bxyt(x,y,t)
  z = 0.0
endfunction

a = 0.25
D = [0 1 0 1]
T = 2
Mx = 40
My = 40
N = 100

[u,x,y,t] = wave_2d(a,D,T,it0,i1t0,bxyt,Mx,My,N)
```

Dari file `TEMP_wave2d_*` dapat dibuat animasi.