LAPORAN PRAKTIKUM

Modul 01: Metode Beda Hingga Turunan Pertama

Dosen Pengampu:

Dr. Nurjanna Joko Trilaksono, S.Si., M.Si.

Disusun oleh:

Fardhan Indrayesa (NIM: 12821046)

Kelompok : 04

PROGRAM STUDI METEOROLOGI

FAKULTAS ILMU DAN TEKNOLOGI KEBUMIAN

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

1. Tujuan Praktikum

a. Menyusun program FORTRAN untuk menghitung turunan pertama dari suatu fungsi atau data variabel medan.
b. Menyusun program FORTRAN untuk menentukan solusi dari suatu persamaan model adveksi.
c. Menyusun program FORTRAN untuk menghitung hasil integrasi numerik dari suatu persamaan gelombang sederhana berdasarkan skema Euler (maju), mundur, dan Leap Frog.
d. Memahami permasalahan kriteria kestabilan.

2. Langkah Kerja

Tugas 4.1 bagian a:

- Membuat direktori "Modul 1" di Ubuntu (melalui Mobaxterm) dan mengunduh folder "deriv" dari Drive Praktikum. Lalu, menyimpannya di direktori "Modul 1".
- Mengubah file *subroutine* eksternal (ddx1.f, ddx2.f, dan ddx4.f) menjadi *object file* dengan perintah
  `gfortran -c nama_file.f`
- *Compile* program utama bersama *object file* dengan perintah
  `gfortran deriv.f ddx1.o ddx2.o ddx4.o -o deriv.exe`
- Eksekusi program dengan perintah
  `./deriv.exe`
- Membuka Jupyter Notebook untuk mem-plot profil tekanan terhadap ketinggian.
- Menulis program .ipynb untuk menghasilkan plot profil tekanan terhadap ketinggian.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xarray as xr

data = pd.read_table("deriv/data.txt", sep=",")
z = [0., 1000., 2000., 3000., 4000., 5000., 6000., 7000., 8000., 9000.]
data["z"] = z
p = data["p(z)"]

fig = plt.figure(figsize=(14,8), dpi=300)
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.plot(z, p)
ax.set_xlabel('z', fontsize=14)
ax.set_ylabel('p', fontsize=14)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)
ax.set_title('Plot Profil Tekanan p(z) Terhadap Ketinggian z', fontsize=15)
fig.savefig("output/no41a.png", dpi=300)
```

- Dihasilkan plot profil tekanan terhadap ketinggian.
- Analisis plot profil tekanan terhadap ketinggian.

Tugas 4.1 bagian b:

- Dari program .ipynb yang sama, olah data hasil turunan untuk menghasilkan nilai eror terhadap nilai analitiknya.

- Menulis program untuk menghasilkan plot nilai turunan terhadap ketinggian dan eror setiap turunan terhadap ketinggian.

```python
# Plot dp/dz terhadap ketinggian
ddx1_f = data["ddx1_f"]
ddx1_b = data["ddx1_b"]
ddx2 = data["ddx2"]
ddx4 = data["ddx4"]
deriv = data["dp/dz"]

fig2 = plt.figure(figsize=(14,8), dpi=300)
ax2 = fig2.add_axes([0.1, 0.1, 0.8, 0.8])

ax2.plot(z, deriv, color = "orange", label = "analytic")
ax2.plot(z, ddx1_f, color = "red", label = "ddx1_f")
ax2.plot(z, ddx1_b, color = "blue", label = "ddx1_b")
ax2.plot(z, ddx2, color = "green", label = "ddx2", marker = "o")
ax2.plot(z, ddx4, color = "black", label = "ddx4", marker = "s")

ax2.set_xlabel('z', fontsize=14)
ax2.set_ylabel('dp/dz', fontsize=14)

ax2.tick_params(axis='x', labelsize=12)
ax2.tick_params(axis='y', labelsize=12)

ax2.legend(loc=2, fontsize = 15) # upper left corner

ax2.set_title('Plot Profil Turunan p(z) (dp/dz) Terhadap Ketinggian z', fontsize=15)
fig2.savefig("output/no41b_dpdz.png", dpi=300)
```

```python
# Plot eror terhadap ketinggian
true = data["dp/dz"]
true1_f = np.zeros(10)
true1_b = np.zeros(10)
true2 = np.zeros(10)
true4 = np.zeros(10)

for i in range(10):
    true1_f[i] = abs((ddx1_f[i] - true[i])/true[i])*100
    true1_b[i] = abs((ddx1_b[i] - true[i])/true[i])*100
    true2[i] = abs((ddx2[i] - true[i])/true[i])*100
    true4[i] = abs((ddx4[i] - true[i])/true[i])*100

fig2 = plt.figure(figsize=(10,8), dpi = 300)
ax2 = fig2.add_axes([0.1, 0.1, 0.8, 0.8]) # Blue group
ax21 = ax2.twinx() # Red group

# X axis
ax2.set_xlabel('z (m)', fontsize=14)
ax2.tick_params(axis='x', labelsize=12)

# Blue group
ax2.plot(z, true1_f, color = "blue", label = "error ddx1_f")
ax2.plot(z, true1_b, color = "cyan", label = "error ddx1_b")

ax2.set_ylabel('Eror relatif (%)', fontsize=14, color="blue")
ax2.tick_params(axis='y', labelsize=12, labelcolor = "blue")

# Red group
ax21.plot(z, true2, color = "red", label = "error ddx2")
ax21.plot(z, true4, color = "orange", label = "error ddx4")

ax21.set_ylabel('Eror relatif (%)', fontsize=14, color = "red")
ax21.tick_params(axis='y', labelsize=12, labelcolor = "red")

# Legends
ax2.legend(loc=6, fontsize = 12) # Blue group
ax21.legend(loc=7, fontsize = 12) # Red group

ax2.set_title('Plot Nilai True Eror dp/dz Terhadap Nilai Analitik', fontsize=15)
fig2.savefig("output/no41b_err.png", dpi=300)
```

- Dihasilkan plot nilai turunan terhadap ketinggian dan eror setiap turunan terhadap ketinggian.
- Analisis hasil plot.

Tugas 4.2 bagian a:

- Menggunakan program ftbs.f90 yang sudah disediakan, *compile* file tersebut dengan perintah
  `gfortran ftbs.f90 -o ftbs.exe`
- Eksekusi program dengan perintah
  `./ftbs.exe`
- Diperoleh nilai CFL dan data berekstensi .dat yang berisi data dengan nilai berturut-turut x, t, dan u.
- Dari program .ipynb yang sama (atau boleh membuat baru), baca data berekstensi .dat tersebut. Lalu, membuat program untuk menampilkan plot 3D dari data tersebut, dengan sumbu x, t, dan u.

```python
# 4.2a
advec = np.loadtxt("advection/ftbs.dat")
advec = pd.DataFrame(advec)
advec.columns = ["x", "t", "u"]
x = advec["x"]
t = advec["t"]
u = advec["u"]

fig3 = plt.figure(figsize=(8, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advec)):
    if i%61 == 0:
        ax3.plot3D(x[i:i+61], t[i:i+61], u[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTBS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.01666\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42a.png", dpi=300)
```

- Dihasilkan plot 3D dari data file .dat.

- Analisis hasil plot.

  Tugas 4.2 bagian b:

- Langkahnya sama seperti bagian a. Namun, nilai variabel dt pada program diubah sesuai perintah soal.

```
inisiasi parameter
a = 300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.02 !s (Langkah waktu)
pi = 3.14
```

- *Compile* program, lalu running program tersebut.
- Diperoleh nilai CFL dan data dalam bentuk file .dat.
- Setelah itu, plot masing-masing data berdasarkan nilai dt yang berbeda.

```
# 4.2b
# \Delta t = 0.01666
advecb1 = np.loadtxt("advection/ftbs.dat")
advecb1 = pd.DataFrame(advecb1)
advecb1.columns = ["x", "t", "u"]
xb1 = advecb1["x"]
tb1 = advecb1["t"]
ub1 = advecb1["u"]

fig3 = plt.figure(figsize=(8, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecb1)):
    if i%61 == 0:
        ax3.plot3D(xb1[i:i+61], tb1[i:i+61], ub1[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTBS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.01666\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42b1.png", dpi=300)
```

```
# 4.2b
# \Delta t = 0.0075
advecb2 = np.loadtxt("advection/ftbsb.dat")
advecb2 = pd.DataFrame(advecb2)
advecb2.columns = ["x", "t", "u"]
xb2 = advecb2["x"]
tb2 = advecb2["t"]
ub2 = advecb2["u"]

fig3 = plt.figure(figsize=(8, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecb2)):
    if i%61 == 0:
        ax3.plot3D(xb2[i:i+61], tb2[i:i+61], ub2[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTBS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.0075\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42b2.png", dpi=300)
```

```python
# 4.2b
# \Delta t = 0.02
advecb3 = np.loadtxt("advection/ftbsb2.dat")
advecb3 = pd.DataFrame(advecb3)
advecb3.columns = ["x", "t", "u"]
xb3 = advecb3["x"]
tb3 = advecb3["t"]
ub3 = advecb3["u"]

fig3 = plt.figure(figsize=(9, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecb3)):
    if i%61 == 0:
        ax3.plot3D(xb3[i:i+61], tb3[i:i+61], ub3[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTBS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.02\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42b3.png", dpi=300)
```

- Analisis perbedaan hasil plot.

  Tugas 4.2 bagian c:

- Salin program ftbs.f90 menjadi ftfs.90 dengan perintah

  `cp ftbs.f90 ftfs.f90`

- Modifikasi persamaan FTBS menjadi FTFS.

```
        do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
          ! loop waktu
          ! syarat batas di boundary grid x = 1 dan grid x = nx
          f(1,it) = 0
          f(nx,it) = 0
          do 102 ix = 2, nx-1
            ! loop ruang
            f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix,it-1))
102     continue
101     continue
```

- Sesuaikan nama file output (.dat)

```
open(1001,file='ftfs3.dat', status='unknown')
```

- *Compile* program, lalu running program tersebut.
- Diperoleh nilai CFL dan data dalam bentuk file .dat.
- Lakukan lagi untuk nilai a yang berbeda.

```
inisiasi parameter
a = -300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.01666 !s (Langkah waktu)
pi = 3.14
```

- Diperoleh file .dat dengan nilai a yang berbeda.
- Plot masing-masing data dalam koordinat 3D.

```python
# 4.2c
# FTFS
advecf = np.loadtxt("advection/ftfs.dat")
advecf = pd.DataFrame(advecf)
advecf.columns = ["x", "t", "u"]
xf = advecf["x"]
tf = advecf["t"]
uf = advecf["u"]

fig3 = plt.figure(figsize=(9, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecf)):
    if i%61 == 0:
        ax3.plot3D(xf[i:i+61], tf[i:i+61], uf[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTFS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.01666\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42c.png", dpi=300)
```

```python
# 4.2c
# FTFS, a = -300 m/s
advecf3 = np.loadtxt("advection/ftfs3.dat")
advecf3 = pd.DataFrame(advecf3)
advecf3.columns = ["x", "t", "u"]
xf3 = advecf3["x"]
tf3 = advecf3["t"]
uf3 = advecf3["u"]

fig3 = plt.figure(figsize=(10, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecf3)):
    if i%61 == 0:
        ax3.plot3D(xf3[i:i+61], tf3[i:i+61], uf3[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTFS dengan $\Delta x = 5\ m$, $\Delta t = 0.01666\ s$, dan a = $-300$ m/s',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
ax3.view_init(30, -45)
fig3.savefig("output/no42c3.png", dpi=300)
```

- Bandingkan dan analisis hasil plot.

  Tugas 4.2 bagian d:

- Salin program ftbs.f90 menjadi ftcs.f90/ctcs.f90 dengan perintah

  `cp ftbs.f90 ftcs.f90`

  `cp ftbs.f90 ctcs.f90`

- Modifikasi persamaan menjadi FTCS/CTCS.

```
 perhitungan FTCS
 do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
   ! loop waktu
   ! syarat batas di boundary grid x = 1 dan grid x = nx
   f(1,it) = 0
   f(nx,it) = 0
   do 102 ix = 2, nx-1
     ! loop ruang
     f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix-1,it-1))
 continue
 continue
```

```
perhitungan CTCS
do 101 it = 2, nt-1 ! prediksi untuk langkat waktu t = 2 s.d nt
  ! loop waktu
  ! syarat batas di boundary grid x = 1 dan grid x = nx
  f(1,it) = 0
  f(nx,it) = 0
  do 102 ix = 2, nx-1
    ! loop ruang
    f(ix,it+1) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix-1,it-1))
continue
continue
```

- Sesuaikan nama file output (.dat)

```
open(1001,file='ftfs.dat', status='unknown')
```

```
open(1001,file='ctcs.dat', status='unknown')
```

- *Compile* program, lalu runnig program tersebut.
- Diperoleh nilai CFL dan data dalam bentuk file dengan ekstensi .dat.
- Lakukan lagi untuk nilai a yang berbeda.

```
inisiasi parameter
a = -300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.01666 !s (Langkah waktu)
pi = 3.14
```

- Diperoleh file .dat dengan nilai a yang berbeda.
- Plot masing-masing data dalam koordinat 3D.

```
# 4.2d
# FTCS
advecc = np.loadtxt("advection/ftcs.dat")
advecc = pd.DataFrame(advecc)
advecc.columns = ["x", "t", "u"]
xc = advecc["x"]
tc = advecc["t"]
uc = advecc["u"]

fig3 = plt.figure(figsize=(9, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecc)):
    if i%61 == 0:
        ax3.plot3D(xc[i:i+61], tc[i:i+61], uc[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTCS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.01666\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42d_FTCS.png", dpi=300)
```

```
# 4.2d
# FTCS, a = -300 m/s
advecc3 = np.loadtxt("advection/ftcs3.dat")
advecc3 = pd.DataFrame(advecc3)
advecc3.columns = ["x", "t", "u"]
xc3 = advecc3["x"]
tc3 = advecc3["t"]
uc3 = advecc3["u"]

fig3 = plt.figure(figsize=(10, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecc3)):
    if i%61 == 0:
        ax3.plot3D(xc3[i:i+61], tc3[i:i+61], uc3[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u FTCS dengan $\Delta x = 5\ m$ $\Delta t = 0.01666\ s$ dan a = $-300$ m/s',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42d_FTCS3.png", dpi=300)
```

```
# 4.2d
# CTCS
advecct = np.loadtxt("advection/ctcs.dat")
advecct = pd.DataFrame(advecct)
advecct.columns = ["x", "t", "u"]
xct = advecct["x"]
tct = advecct["t"]
uct = advecct["u"]

fig3 = plt.figure(figsize=(10, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecct)):
    if i%61 == 0:
        ax3.plot3D(xct[i:i+61], tct[i:i+61], uct[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u CTCS dengan $\Delta x = 5\ m$ dan $\Delta t = 0.01666\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42d_CTCS.png", dpi=300)
```

```python
# 4.2d
# CTCS, a = -300 m/s
advecct3 = np.loadtxt("advection/ctcs3.dat")
advecct3 = pd.DataFrame(advecct3)
advecct3.columns = ["x", "t", "u"]
xct3 = advecct3["x"]
tct3 = advecct3["t"]
uct3 = advecct3["u"]

fig3 = plt.figure(figsize=(10, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(advecct3)):
    if i%61 == 0:
        ax3.plot3D(xct3[i:i+61], tct3[i:i+61], uct3[i:i+61], "blue")
ax3.set_title(r'Solusi Adveksi u CTCS dengan $\Delta x = 5\ m$, $\Delta t = 0.01666\ s$, dan a = $-300$ m/s',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no42d_CTCS3.png", dpi=300)
```

- Bandingkan dan analisis hasil plot.

  Tugas 4.3:

- Buka grads di Ubuntu dengan perintah `grads`
- Buka file .ctl.
- Set latitude 15 dan set ketinggian 500.

```
ga-> set lat 15
LAT set to 14 14
ga-> set lev 500
LEV set to 500 500
```

- Tulis data angin zonal (u) pada file .dat

```
ga-> set gxout fwrite
ga-> set fwrite angin.dat
FWrite file name = angin.dat
FWwrite byte order is little_endian; format is stream
Fwrite replacing an existing file
ga-> d u
Replacing file angin.dat.
Wrote 73 of 73 elements to angin.dat as Stream Little_Endian
ga-> disable fwrite
```

- Ganti nilai syarat awal dengan data angin zonal.

```
memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
OPEN(1000, FILE="angin.dat", FORM="UNFORMATTED", ACCESS="STREAM")
DO i = 1, nx, 1
    READ(1000, IOSTAT = ierror) u
    IF (ierror /= 0) EXIT
    f(i, 1) = u
END DO
CLOSE(1000)
```

- Sesuaikan parameter dengan koordinat asli bumi.

```
inisiasi parameter
pi = 3.14
lintang = 14*pi/180
bujur = 5*pi/180
a = 300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 24*60*60 ! s (Rentang waktu simulasi)   diubahhh, dari hari ke jam
Lx = 2*pi*6371000*cos(lintang) ! m (Panjang domain simulasi)
dx = 6371000*cos(lintang)*bujur ! m (Jarak antar grid)
dt = 1798 !s (Langkah waktu)
```

- Tambahkan persamaan FTBS untuk titik awal dan akhir.

```
perhitungan FTBS
do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
  ! loop waktu
  do 102 ix = 1, nx
    ! loop ruang
    if (ix == 1) then
        f(ix, it) = f(ix,it-1)-((a*dt)/dx)*(f(ix,it-1)-f(nx,it-1))
    else
        f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix,it-1)-f(ix-1,it-1))
    end if
continue
continue
```

- Sesuaikan nama file output (.dat).

```
open(1001,file='hasil.dat', status='unknown')
```

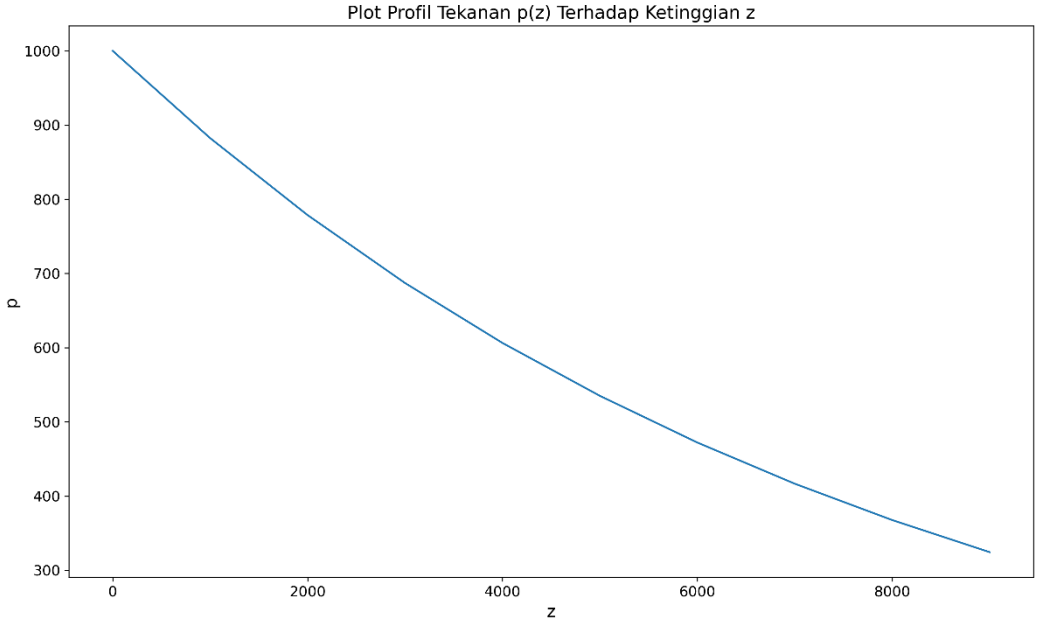
- *Compile* program, lalu runnig program tersebut.
- Diperoleh nilai CFL dan data dalam bentuk file dengan ekstensi .dat.
- Plot data hasil turunan ke dalam koordinat 3D.

```
# 4.3
# FTBS, a = 300 m/s
model3 = np.loadtxt("model/hasil.dat")
model3 = pd.DataFrame(model3)
model3.columns = ["x", "t", "u"]
xm3 = model3["x"]
tm3 = model3["t"]
um3 = model3["u"]

fig3 = plt.figure(figsize=(9, 8), dpi=300)
ax3 = fig3.add_subplot(1,1,1, projection = "3d")

for i in range(len(model3)):
    if i%73 == 0:
        ax3.plot3D(xm3[i:i+73], tm3[i:i+73], um3[i:i+73], "blue")
ax3.set_title(r'Solusi u FTBS dengan $\Delta x = 539459.81\ m$ dan $\Delta t = 1798\ s$',
              fontsize=15, fontweight ='bold')
ax3.set_xlabel('x (m)', fontsize=12, fontweight ='bold')
ax3.set_ylabel('t (s)', fontsize=12, fontweight ='bold')
ax3.set_zlabel('u', fontsize=12, fontweight ='bold')
plt.tight_layout()
fig3.savefig("output/no43_FTBS3.png", dpi=300)
```

- Analisis hasil plot.

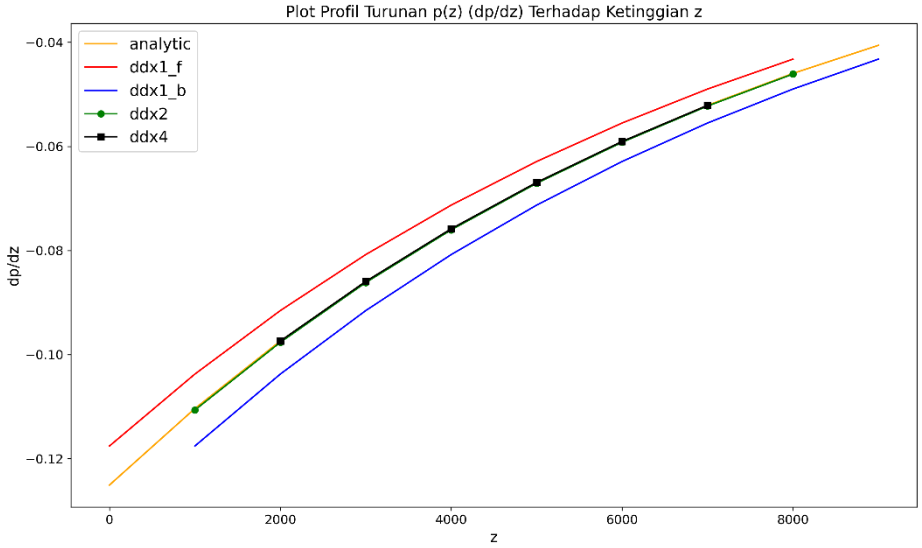
3. Hasil dan Analisis

Tugas 4.1 bagian a:



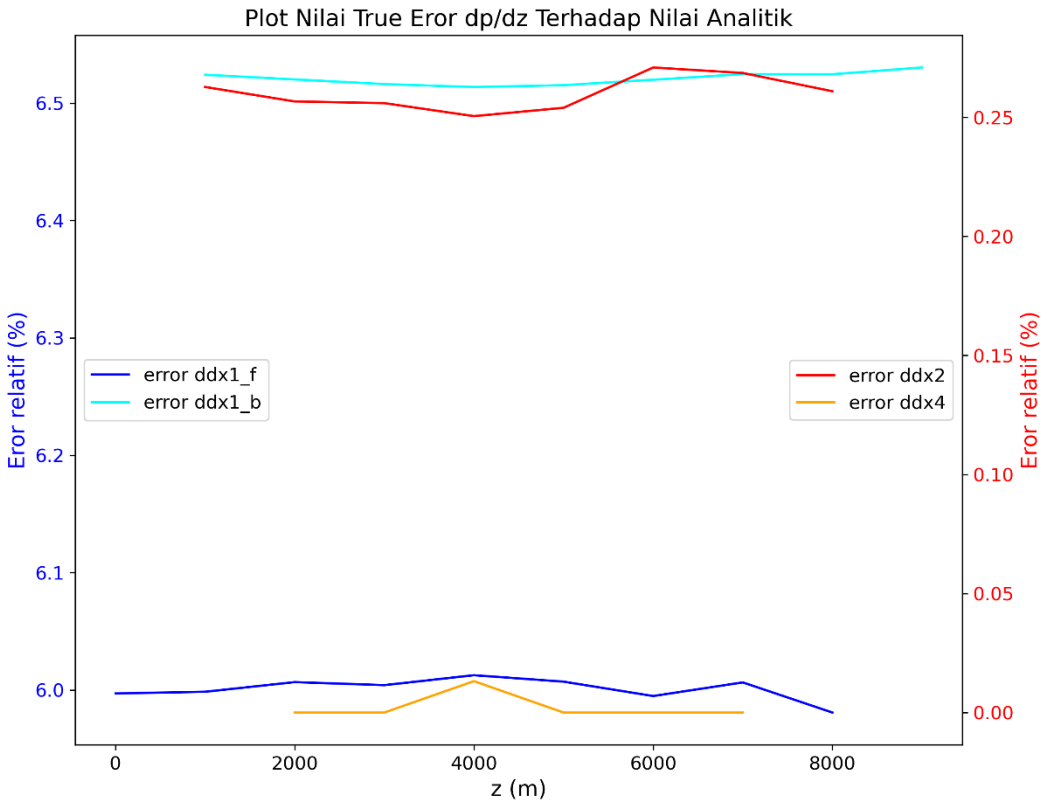
Gambar tersebut menunjukkan profil tekanan terhadap ketinggian yang dihitung berdasarkan persamaan profil tekanan eksponensial. Berdasarkan gambar tersebut, tekanan menurun secara eksponensial seiring bertambahnya ketinggian. Tekanan tertinggi berada pada permukaan bumi, yaitu 1000 hPa. Tekanan terendah berada pada ketinggian 10000 m, yaitu sekitar 320 hPa. Tekanan yang menurun ini dapat disebabkan oleh kandungan udara di atmosfer berkurang seiring bertambahnya ketinggian. Kandungan udara yang berkurang menyebabkan densitas udara di atmosfer bagian atas mengecil, sehingga gaya tekan di atmosfer bagian atas cenderung lemah.

Tugas 4.1 bagian b:



Gambar tersebut menunjukkan hasil turunan setiap metode terhadap ketinggian. Berdasarkan hasil plot tersebut, terlihat bahwa metode yang paling dekat dengan hasil analitik (kuning) adalah metode turunan pertama orde dua/beda tengah (ddx2) dan turunan pertama

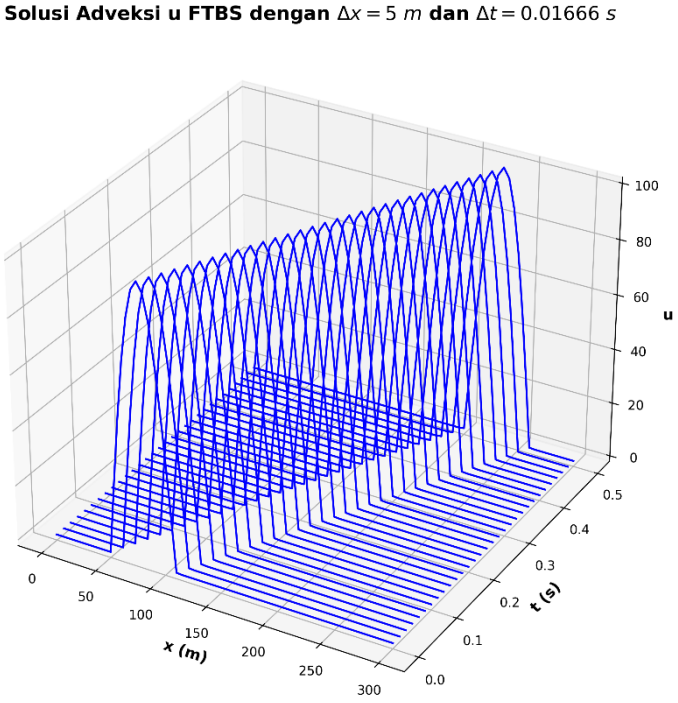
orde keempat (ddx4). Metode yang mendekati nilai analitik lainnya, yaitu beda maju turunan pertama orde satu (ddx1_f) dan beda mundur turunan pertama orde satu (ddx1_b). Dalam menghitung turunan pertama beda maju dibutuhkan satu nilai tekanan di titik saat ini dan di titik selanjutnya, sehingga hasil plotnya terlihat terpotong di titik 8000 m (merah). Dalam menghitung turunan pertama beda mundur dibutuhkan satu nilai tekanan di titik sebelumnya dan di titik saat ini, sehingga hasil plotnya terlihat terpotong di titik 1000 m (biru). Dalam menghitung turunan pertama orde kedua/beda tengah, dibutuhkan satu nilai tekanan di titik sebelumnya dan satu nilai tekanan di titik selanjutnya, sehingga hasil plotnya terlihat terpotong di titik 1000 m dan 8000 m (hijau). Dalam menghitung turunan pertama orde keempat dibutuhkan dua nilai tekanan di titik sebelumnya dan nilai tekanan di titik selanjutnya, sehingga hasil plotnya terlihat terpotong di titik 2000 m dan 7000 m (hitam). Dari sini dapat diketahui bahwa (untuk kasus ini), hasil turunan akan semakin akurat (mendekati nilai analitik) apabila orde turunannya semakin tinggi. Selain itu, nilai turunan yang bernilai negatif menunjukkan bahwa plot profil tekanan monoton turun seiring bertambahnya ketinggian. Nilai turunan yang menaik dengan perlahan dan bernilai negatif menandakan bahwa nilai tekanan menurun dengan lambat seiring bertambahnya ketinggian.



Gambar di atas menunjukkan nilai eror relatif dari hasil keempat turunan terhadap nilai analitiknya. Hasil plot tersebut dibagi menjadi dua kelompok. Kelompok merah (merah dan oranye) dan kelompok biru (biru dan cyan). Kelompok merah membaca nilai eror menggunakan sumbu y di sebelah kanan (dengan label berwarna merah), sedangkan kelompok biru membaca nilai eror menggunakan sumbu y di sebelah kiri (dengan label berwarna biru). Berdasarkan hasil plot tersebut terlihat bahwa metode yang memiliki nilai eror terkecil adalah metode turunan pertama orde keempat (oranye). Sedangkan, metode yang memiliki nilai eror terbesar adalah metode beda mundur turunan pertama orde pertama. Nilai eror metode beda

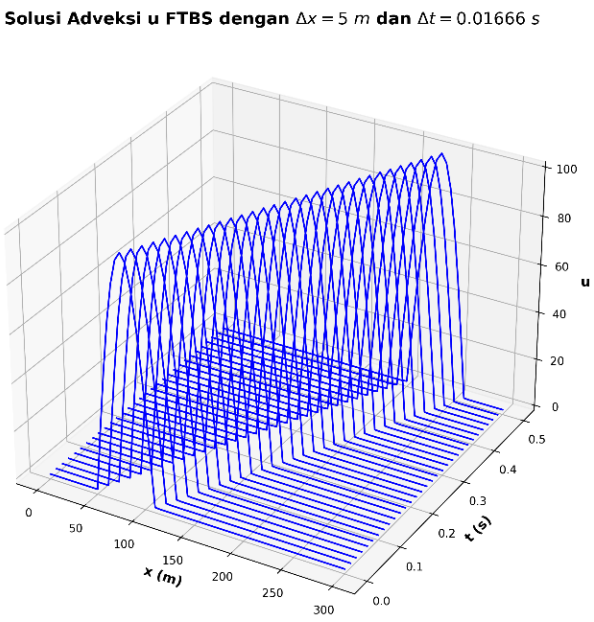
mundur (cyan) lebih besar daripada nilai eror beda maju (biru) karena selisih nilai tekanan dengan titik sebelumnya lebih besar daripada selisih nilai tekanan dengan titik setelahnya. Nilai eror yang dihasilkan cenderung berfluktuasi karena hasil eror dihitung berdasarkan proposi mutlak dari hasil selisih nilai turunan terhadap nilai analitiknya. Nilai eror ini mempengaruhi hasil turunan yang diperoleh dari metode beda hingga. Semakin besar nilai eror, maka hasil turunan menjauhi nilai analitik. Sebaliknya, semakin kecil nilai eror, maka hasil turunan mendekati nilai analitik. Metode beda mundur menghasilkan eror yang lebih besar daripada metode beda maju dan metode beda tengah orde dua menghasilkan nilai eror yang lebih besar daripada metode beda tengah orde keempat.

Tugas 4.2 bagian a:

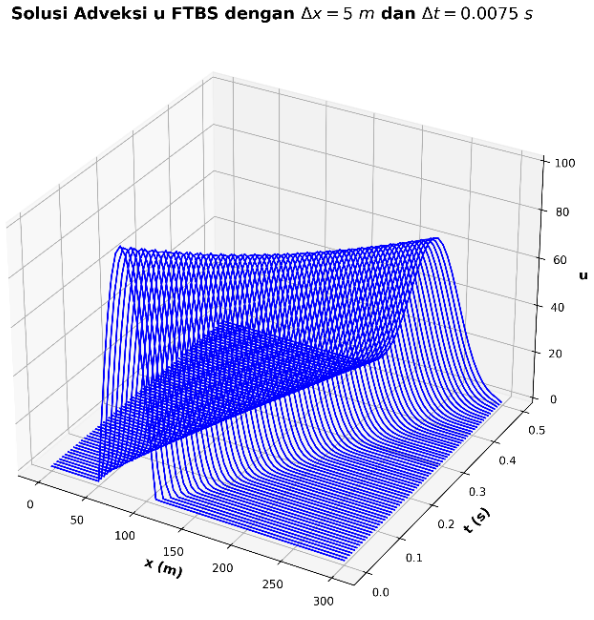


Gambar di atas menunjukkan nilai adveksi gangguan u terhadap posisi x dan waktu t. Adveksi gangguan u semakin bergeser ke kanan seiring bertambahnya waktu t. Hal ini dapat terjadi karena gangguan tersebut memiliki magnitudo positif ($a = 300\, m/s$), sehingga menyebabkan arah adveksi ke arah kanan. Nilai kestabilan (CFL) untuk kasus ini adalah sekitar 0.9996. Nilai CFL yang mendekati 1 dikatakan stabil karena nilai dari solusi adveksi memiliki nilai yang tidak terlalu besar dengan nilai dari waktu t sebelumnya, sehingga hasil dari solusi adveksi yang diperoleh cenderung memiliki nilai yang sama terhadap waktu t sebelumnya.

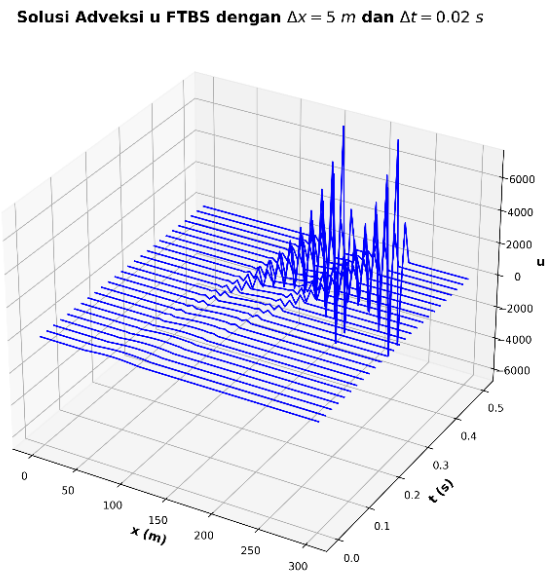
Tugas 4.2 bagian b:



Nilai kestabilan (CFL) untuk kasus ini adalah sekitar 0.9996. Nilai CFL yang mendekati 1 dikatakan stabil karena nilai dari solusi adveksi memiliki nilai yang tidak terlalu besar dengan nilai adveksi dari waktu t sebelumnya, sehingga hasil dari solusi adveksi yang diperoleh cenderung memiliki nilai yang sama terhadap waktu t sebelumnya.

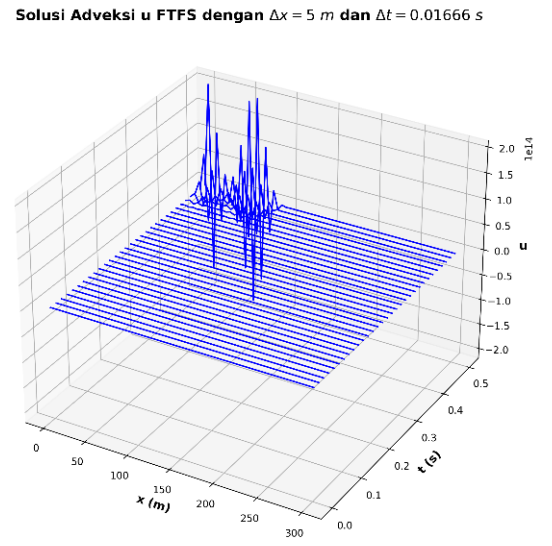


Nilai kestabilan (CFL) untuk kasus ini adalah sekitar 0.45. Nilai CFL ini masih dikatakan stabil karena nilai dari solusi adveksi ini memiliki nilai yang lebih kecil dan tidak terlalu besar dengan nilai adveksi dari waktu t sebelumnya, sehingga hasil dari solusi adveksi yang diperoleh cenderung menurun terhadap nilai adveksi pada waktu t sebelumnya.

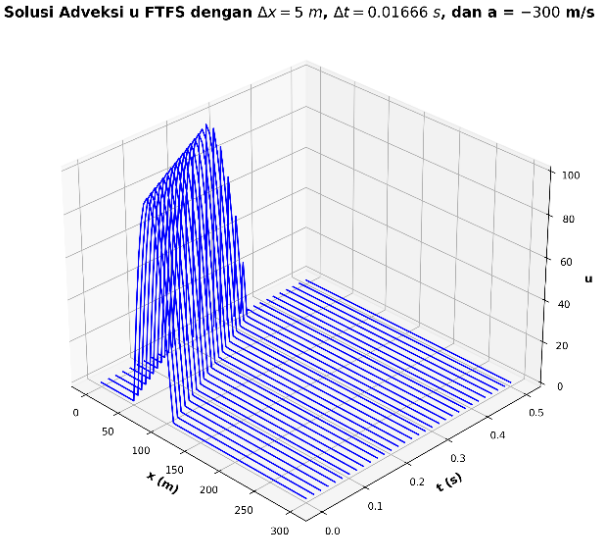


Nilai kestabilan (CFL) untuk kasus ini adalah sekitar 1.2. Nilai CFL ini dikatakan tidak stabil karena nilai dari solusi adveksi ini memiliki nilai yang bertambah besar seiring bertambahnya waktu t, sehingga menyebabkan solusi yang "meledak" pada plot adveksi. Selain itu, gangguan menjalar ke arah kanan karena nilai a yang lebih besar dari nol (positif).

Tugas 4.2 bagian c:

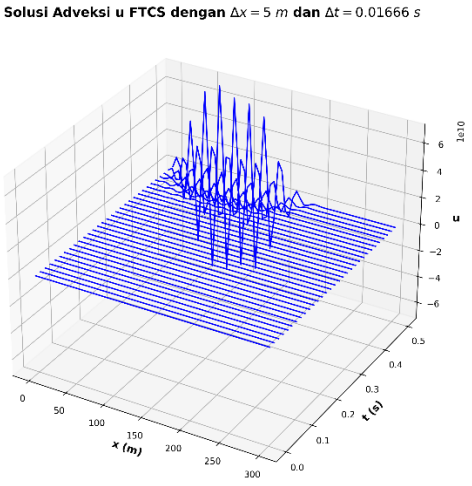


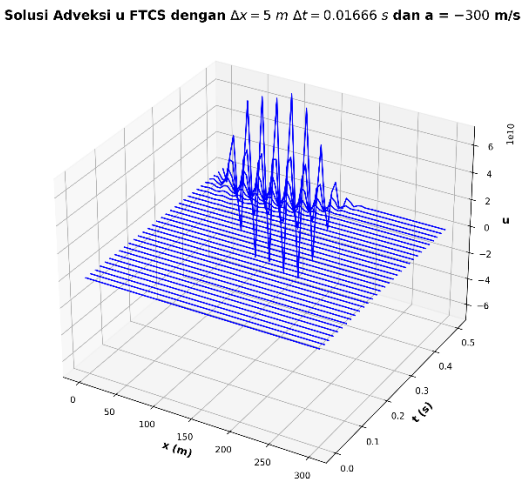
Gambar di atas adalah solusi adveksi u metode FTFS dengan nilai CFL 0.9996. Nilai CFL ini tidak stabil karena menghasilkan solusi adveksi yang bertambah besar seiring bertambahnya waktu t, sehingga menyebabkan solusi yang "meledak" pada plot adveksi. Selain itu, solusi yang diperoleh tidak sama dengan metode FTBS karena metode FTBS dihitung berdasarkan data di titik saat ini dan di titik sebelumnya. Sedangkan, metode FTFS dihitung berdasarkan data di titik saat ini dan di titik setelahnya, sehingga menghasilkan solusi yang berbeda. Metode FTFS menghasilkan solusi yang "meledak" karena nilai CFL yang mendekati 1 sebagai faktor pengali dari selisih adveksi u terhadap jarak, dapat menghasilkan solusi adveksi u membesar atau mengecil dengan sangat cepat.



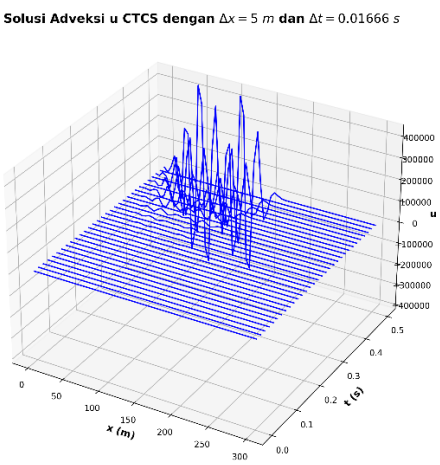
Gambar di atas adalah solusi adveksi u metode FTFS dengan nilai CFL -0.9996 dan nilai $a = -300\, ms^{-1}$. Nilai CFL ini cenderung stabil karena menghasilkan solusi adveksi u yang stabil seiring bertambahnya waktu t. Solusi adveksi ini bergeser ke kiri karena nilai a lebih kecil dari nol (negatif). Solusi adveksi ini dihitung berdasarkan data di titik saat ini dan di titik selanjutnya, serta selisihnya dikalikan dengan CFL hingga menghasilkan solusi yang stabil.

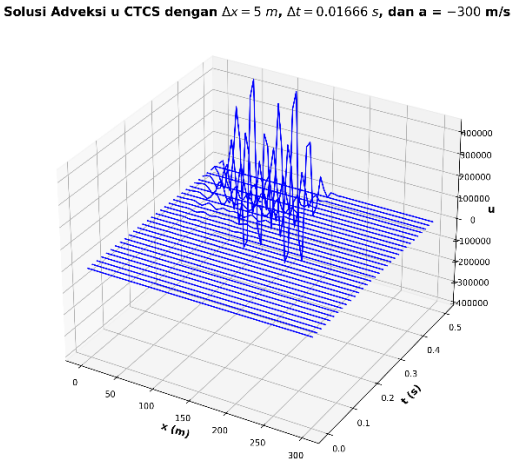
Tugas 4.2 bagian d:





Gambar di atas adalah solusi adveksi u metode FTCS dengan nilai CFL 0.9996 dan –0.9996. Kedua nilai CFL ini menghasilkan solusi yang "meledak" pada metode FTCS, dibandingkan pada metode FTBS dan FTFS. Pada metode FTBS dengan CFL 0.9996 dan metode FTFS dengan CFL -0.9996, solusi yang diperoleh lebih stabil daripada metode FTCS. Hal ini karena pada metode FTCS, titik x yang digunakan adalah titik sebelumnya dan titik setelahnya, sehingga menghasilkan nilai yang lebih berfluktuasi daripada metode FTBS dan FTFS. Sedangkan, pada metode FTFS dengan CFL 0.9996, solusi yang dihasilkan sama-sama tidak stabil.

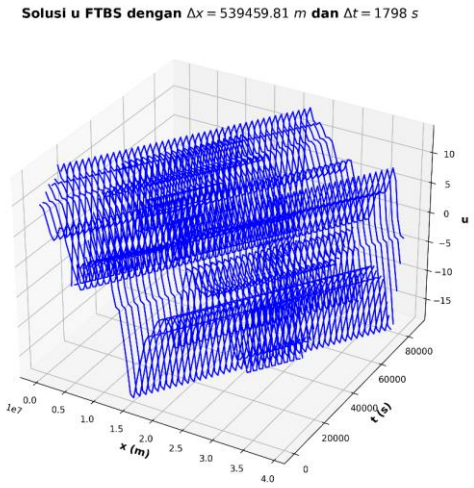




Gambar di atas adalah solusi adveksi u metode CTCS dengan nilai CFL 0.9996 dan -0.9996. Kedua nilai CFL ini menghasilkan solusi yang "meledak" pada metode CTCS, dibandingkan pada metode FTBS dan FTFS. Pada metode FTBS dengan CFL 0.9996 dan metode FTFS dengan CFL -0.9996, solusi yang diperoleh lebih stabil daripada metode CTCS. Hal ini karena pada metode CTCS, titik x dan t yang digunakan adalah titik sebelumnya dan titik setelahnya, sehingga menghasilkan nilai yang lebih berfluktuasi daripada metode FTBS dan FTFS. Sedangkan, pada metode FTFS dengan CFL 0.9996, solusi yang dihasilkan sama-sama tidak stabil.

Kelebihan metode FTBS dan FTFS adalah data yang digunakan dapat lebih sedikit untuk proses perhitungannya. Sedangkan kelemahannya, untuk metode FTBS, tidak dapat menentukan data di awal waktu, untuk metode FTFS, tidak dapat menentukan data di akhir waktu. Hal tersebut dapat diatasi dengan syarat batas. Kelebihan metode FTCS dan CTCS adalah perhitungan yang dihasilkan semakin presisi. Tapi, kelemahan dari metode ini, yaitu data pada waktu selanjutnya harus diketahui terlebih dahulu.

Tugas 4.3:



Gambar di atas adalah solusi adveksi angin zonal u metode FTBS dengan nilai CFL 1.00036538. Nilai CFL ini cenderung stabil karena solusi yang diperoleh tidak mengalami fluktuasi. Karena nilai a lebih besar dari nol (positif), plot solusi u cenderung bergerak ke arah kanan. Nilai Δt yang diperoleh agar jumlah langkah waktu lebih sedikit dan masih memenuhi kestabilan numerik adalah 1798. Nilai ini diperoleh dari nilai a, Δx, dan CFL yang sudah

diketahui (CFL = 1, untuk stabil). Nilai-nilai tersebut disubstitusi pada persamaan CFL, sehingga didapatkan nilai Δt , yaitu sekitar 1798.

4. Kesimpulan

a. Program FORTRAN untuk menghitung turunan pertama dari suatu fungsi atau data variabel medan.

Turunan pertama orde satu:

```
      subroutine DDX1 (a,b,l,dx,index)
c*************************************************************************
c  This subroutine performs the second
c  order finite difference
c
c  input      : a,l,dx
c  output     : b
c  index=1    : forward scheme
c  index=-1   : backward scheme
c*************************************************************************
      real a(l),b(l)
      integer index
c
      if (index.eq.1) then
                l1  = l-1
                do 2110 i = 1, l1
                   b(i)   =  (a(i+1) - a(i))/(dx)
2110       continue
           b(l)=-9999999
           end if

      if (index.eq.-1) then
                do 2111 i = 2, l
                   b(i)   =  (a(i) - a(i-1))/(dx)
2111       continue
           b(1)=-9999999
      end if

c
      return
      end
```

Turunan pertama orde dua:

```
      subroutine DDX2 (a,b,l,dx)
c*************************************************************************
c  This subroutine performs the second
c  order finite difference
c
c  input      : a,l,dx
c  output     : b
c*************************************************************************
      real a(l),b(l)
c
      l1  = l-1
      do 2110 i = 2, l1
         b(i)   =  (a(i+1) - a(i-1))/(2.*dx)
 2110 continue
      b(1) = -9999999
      b(l) = -9999999
c
      return
      end
```

Turunan pertama orde empat:

```
      subroutine DDX4 (a,b,l,dx)
c*************************************************************************
c  This subroutine performs the fourth order
c  finite difference .
c
c  input  : a,l,dx
c  output : b
c*************************************************************************
      real a(l),b(l)
c
      l1 = l-1
      l2 = l-2
      l3 = l-3
      c1 = 4./3.
      c2 = 1./3.
      dx2 = 1./(2.*dx)
      dx4 = 1./(4.*dx)
      do 2120 i = 3, l2
          b(i)  = c1*((a(i+1)-a(i-1))*dx2)
     &          - c2*((a(i+2)-a(i-2))*dx4)
 2120 continue
      b(2)      = -9999999
      b(1)      = -9999999
      b(l)      = -9999999
      b(l1)     = -9999999
c
      return
      end
```

b. Program FORTRAN untuk menentukan solusi dari suatu persamaan model adveksi.

```
program ftbs

! deklarasi variabel
real,allocatable :: x(:), t(:), f(:,:) ! f(nx,nt)
real :: a , dx, dt, Lt, Lx, pi, lintang, bujur, u
integer :: nx, nt, ix, it, ierror

inisiasi parameter
pi = 3.14
lintang = 14*pi/180
bujur = 5*pi/180
a = 300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 24*60*60 ! s (Rentang waktu simulasi)   diubahhh, dari hari ke jam
Lx = 2*pi*6371000*cos(lintang) ! m (Panjang domain simulasi)
dx = 6371000*cos(lintang)*bujur ! m (Jarak antar grid)
dt = 1798 !s (Langkah watu)

Hitung array yg dibutuhkan
nx = int(Lx/dx)+1
nt = int(Lt/dt)+1 ! (Jumlah langkah waktu)

Memasukkan nilai array
allocate (f(nx,nt))
allocate (x(nx))
allocate (t(nt))
do ix = 1, nx
   x(ix) = ix*dx - dx
end do
do it = 1, nt
   t(it) = it*dt - dt
end do

memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
OPEN(1000, FILE="angin.dat", FORM="UNFORMATTED", ACCESS="STREAM")
DO i = 1, nx, 1
    READ(1000, IOSTAT = ierror) u
```

```
            f(i, 1) = u
        END DO
        CLOSE(1000)

!       perhitungan FTBS
        do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
          ! loop waktu
          do 102 ix = 1, nx
            ! loop ruang
            if (ix == 1) then
                f(ix, it) = f(ix,it-1)-((a*dt)/dx)*(f(ix,it-1)-f(nx,it-1))
            else
                f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix,it-1)-f(ix-1,it-1))
            end if
102     continue
101     continue

!       ========== FORMAT PENULISAN DATA ============
200     format(F15.1,2x,F15.3,2x,F10.3)
!       ============================================


        open(1001,file='hasil.dat', status='unknown')

        do 106 it=1,nt
         do 107 ix=1,nx
           write(1001,200) x(ix),t(it), f(ix,it)
           write(*,200) x(ix),t(it), f(ix,it)
107      continue
106     continue

        write(*,*)
        write(*,*) "Courant number = ", (a*dt)/dx
        write(*,*) "dt = ", dt
        write(*,*) "nt = ", nt

        close(1001)

        end program ftbs
```

c. Program FORTRAN untuk menghitung hasil integrasi numerik dari suatu persamaan gelombang sederhana berdasarkan skema Euler (maju), mundur, dan Leap Frog.

Skema Euler (beda maju, FTFS)

```
program ftfs

! deklarasi variabel
real,allocatable :: x(:), t(:), f(:,:) ! f(nx,nt)
real :: a , dx, dt, Lt, Lx, pi
integer :: nx, nt , xi, xj, i , j , ix, it

inisiasi parameter
a = -300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.01666 !s (Langkah waktu)
pi = 3.14

Hitung array yg dibutuhkan
nx = int(Lx/dx)+1
nt = int(Lt/dt)+1 ! (Jumlah langkah waktu)

Memasukkan nilai array
allocate (f(nx,nt))
allocate (x(nx))
allocate (t(nt))
do ix = 1, nx
   x(ix) = ix*dx - dx
end do
do it = 1, nt
   t(it) = it*dt - dt
end do

Batas fungsi
xi = 50 ! m
xj = 110 ! m
i = int(xi/dx)+1
j = int(xj/dx)+1

memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
f(1:i-1,1) = 0

untuk x = i sd x = j bernilai dari suatu fungsi sinus
do 100 ix = i,j
      f(ix,1) = 100*sin(pi*(x(ix)-50.0)/60.0)
100   continue

!     untuk x = j sd x = nx bernilai 0
      f(j+1:nx,1) = 0


!     perhitungan FTFS
      do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
        ! loop waktu
        ! syarat batas di boundary grid x = 1 dan grid x = nx
        f(1,it) = 0
        f(nx,it) = 0
        do 102 ix = 2, nx-1
          ! loop ruang
          f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix,it-1))
102   continue
101   continue

!     ========== FORMAT PENULISAN DATA ============
200   format(F5.1,2x,F7.3,2x,F20.3)
!     ============================================


      open(1001,file='ftfs3.dat', status='unknown')

      do 106 it=1,nt
       do 107 ix=1,nx
         write(1001,200) x(ix),t(it), f(ix,it)
         write(*,200) x(ix),t(it), f(ix,it)
107    continue
106   continue

      write(*,*)
      write(*,*) "Courant number = ", (a*dt)/dx

      close(1001)

      end program ftfs
```

Beda mundur (FTBS)

```
program ftbs

! deklarasi variabel
real,allocatable :: x(:), t(:), f(:,:) ! f(nx,nt)
real :: a , dx, dt, Lt, Lx, pi
integer :: nx, nt , xi, xj, i , j , ix, it

inisiasi parameter
a = 300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.02 !s (Langkah waktu)
pi = 3.14

Hitung array yg dibutuhkan
nx = int(Lx/dx)+1
nt = int(Lt/dt)+1 ! (Jumlah langkah waktu)

Memasukkan nilai array
allocate (f(nx,nt))
allocate (x(nx))
allocate (t(nt))
do ix = 1, nx
   x(ix) = ix*dx - dx
end do
do it = 1, nt
   t(it) = it*dt - dt
end do

Batas fungsi
xi = 50 ! m
xj = 110 ! m
i = int(xi/dx)+1
j = int(xj/dx)+1

memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
f(1:i-1,1) = 0

untuk x = i sd x = j bernilai dari suatu fungsi sinus
do 100 ix = i,j
100   continue

!     untuk x = j sd x = nx bernilai 0
      f(j+1:nx,1) = 0


!     perhitungan FTBS
      do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
        ! loop waktu
        ! syarat batas di boundary grid x = 1 dan grid x = nx
        f(1,it) = 0
        f(nx,it) = 0
        do 102 ix = 2, nx-1
          ! loop ruang
          f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix,it-1)-f(ix-1,it-1))
102   continue
101   continue

!     ========== FORMAT PENULISAN DATA ============
200   format(F5.1,2x,F7.3,2x,F10.3)
!     ============================================
201   format("x", 2x, "t", 2x, "u")

      open(1001,file='ftbsb2.dat', status='unknown')

!      write(1001, 201)
      do 106 it=1,nt
       do 107 ix=1,nx
         write(1001,200) x(ix),t(it), f(ix,it)
         write(*,200) x(ix),t(it), f(ix,it)
107    continue
106   continue

      write(*,*)
      write(*,*) "Courant number = ", (a*dt)/dx

      close(1001)

      end program ftbs
```

Beda tengah (FTCS)

```
program ftcs

! deklarasi variabel
real,allocatable :: x(:), t(:), f(:,:) ! f(nx,nt)
real :: a , dx, dt, Lt, Lx, pi
integer :: nx, nt , xi, xj, i , j , ix, it

inisiasi parameter
a = 300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.01666 !s (Langkah waktu)
pi = 3.14

Hitung array yg dibutuhkan
nx = int(Lx/dx)+1
nt = int(Lt/dt)+1 ! (Jumlah langkah waktu)

Memasukkan nilai array
allocate (f(nx,nt))
allocate (x(nx))
allocate (t(nt))
do ix = 1, nx
   x(ix) = ix*dx - dx
end do
do it = 1, nt
   t(it) = it*dt - dt
end do

Batas fungsi
xi = 50 ! m
xj = 110 ! m
i = int(xi/dx)+1
j = int(xj/dx)+1

memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
f(1:i-1,1) = 0

untuk x = i sd x = j bernilai dari suatu fungsi sinus
do 100 ix = i,j
      f(ix,1) = 100*sin(pi*(x(ix)-50.0)/60.0)
100   continue

!     untuk x = j sd x = nx bernilai 0
      f(j+1:nx,1) = 0


!     perhitungan FTCS
      do 101 it = 2, nt ! prediksi untuk langkat waktu t = 2 s.d nt
        ! loop waktu
        ! syarat batas di boundary grid x = 1 dan grid x = nx
        f(1,it) = 0
        f(nx,it) = 0
        do 102 ix = 2, nx-1
          ! loop ruang
          f(ix,it) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix-1,it-1))
102   continue
101   continue

!     ========== FORMAT PENULISAN DATA ============
200   format(F5.1,2x,F7.3,2x,F20.3)
!     ============================================
201   format("x", 2x, "t", 2x, "u")

      open(1001,file='ftcs.dat', status='unknown')

!      write(1001, 201)
      do 106 it=1,nt
       do 107 ix=1,nx
         write(1001,200) x(ix),t(it), f(ix,it)
         write(*,200) x(ix),t(it), f(ix,it)
107    continue
106   continue

      write(*,*)
      write(*,*) "Courant number = ", (a*dt)/dx

      close(1001)

      end program ftcs
```

Leap Frog (CTCS)

```
program ctcs

! deklarasi variabel
real,allocatable :: x(:), t(:), f(:,:) ! f(nx,nt)
real :: a , dx, dt, Lt, Lx, pi
integer :: nx, nt , xi, xj, i , j , ix, it

inisiasi parameter
a = -300 ! m/s (kecepatan/amplitudo adveksi)
Lt = 0.5 ! s (Rentang waktu simulasi)
Lx = 300 ! m (Panjang domain simulasi)
dx = 5 ! m (Jarak antar grid)
dt = 0.01666 !s (Langkah waktu)
pi = 3.14

Hitung array yg dibutuhkan
nx = int(Lx/dx)+1
nt = int(Lt/dt)+1 ! (Jumlah langkah waktu)

Memasukkan nilai array
allocate (f(nx,nt))
allocate (x(nx))
allocate (t(nt))
do ix = 1, nx
   x(ix) = ix*dx - dx
end do
do it = 1, nt
   t(it) = it*dt - dt
end do

Batas fungsi
xi = 50 ! m
xj = 110 ! m
i = int(xi/dx)+1
j = int(xj/dx)+1

memasukkan kondisi awal
untuk x = 1 sd x = i-1 bernilai 0
f(1:i-1,1) = 0

untuk x = i sd x = j bernilai dari suatu fungsi sinus
do 100 ix = i,j
      f(ix,1) = 100*sin(pi*(x(ix)-50.0)/60.0)
100   continue

!     untuk x = j sd x = nx bernilai 0
      f(j+1:nx,1) = 0


!     perhitungan CTCS
      do 101 it = 2, nt-1 ! prediksi untuk langkat waktu t = 2 s.d nt
        ! loop waktu
        ! syarat batas di boundary grid x = 1 dan grid x = nx
        f(1,it) = 0
        f(nx,it) = 0
        do 102 ix = 2, nx-1
          ! loop ruang
          f(ix,it+1) = f(ix,it-1)-((a*dt)/dx)*(f(ix+1,it-1)-f(ix-1,it-1))
102   continue
101   continue

!     ========== FORMAT PENULISAN DATA ============
200   format(F5.1,2x,F7.3,2x,F20.3)
!     ============================================
201   format("x", 2x, "t", 2x, "u")

      open(1001,file='ctcs3.dat', status='unknown')

!      write(1001, 201)
      do 106 it=1,nt
       do 107 ix=1,nx
         write(1001,200) x(ix),t(it), f(ix,it)
         write(*,200) x(ix),t(it), f(ix,it)
107    continue
106   continue

      write(*,*)
      write(*,*) "Courant number = ", (a*dt)/dx

      close(1001)

      end program ctcs
```

d. Solusi dianggap stabil apabila nilai kriteria kestabilan (CFL) lebih kecil atau sama dengan 1. Solusi dianggap tidak stabil apabila nilai kriteria kestabilan (CFL) lebih besar dari 1.