NAME: THOMAS AMO KYEIMIAH

STUDENT ID: 261121054

COURSE: ATOC 558 PROJECT
13th APRIL 2023

---

## QUESTION 1

Given

$$\frac{\partial u}{\partial t} = -g\frac{\partial \eta}{\partial x} \tag{1}$$
$$\frac{\partial v}{\partial t} = -g\frac{\partial \eta}{\partial y} \tag{2}$$
$$\frac{\partial \eta}{\partial t} = -H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \tag{3}$$

Taking $\partial_x$ for (1), $\partial_y$ for (2) and $\partial_t$ for (3). We get;

$$\frac{\partial^2 u}{\partial t\partial x} = -g\frac{\partial^2 \eta}{\partial x^2} \tag{4}$$
$$\frac{\partial^2 v}{\partial t\partial y} = -g\frac{\partial^2 \eta}{\partial y^2} \tag{5}$$
$$\frac{\partial^2 \eta}{\partial t^2} = -H\left(\frac{\partial^2 u}{\partial x\partial t} + \frac{\partial^2 v}{\partial y\partial t}\right) \tag{6}$$

Substitute (4) and (5) into (6). We get:

$$\frac{\partial^2 \eta}{\partial t^2} = gH\left(\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2}\right) \tag{7}$$

Given waveform solution: $\eta = \eta_0 e^{i(kx+ly-\omega t)}$ and substituting it into (7):

$$i^2\omega^2\eta_0 e^{i(kx+ly-\omega t)} = gH\left(i^2k^2\eta_0 e^{i(kx+ly-\omega t)} + i^2l^2\eta_0 e^{i(kx+ly-\omega t)}\right) \tag{8}$$

Now, we can solve for the dispersion relation:

$$-\omega^2 = -gH\left(k^2+l^2\right) \tag{9}$$

$$\boxed{\boldsymbol{\omega = \pm\sqrt{gH\left(k^2+l^2\right)}}} \tag{10}$$

This is the dispersion relation for the non-rotating linearized shallow-water equations.

The positive and negative signs in the dispersion relation represent **two physical modes** traveling in opposite directions. These two modes have the same frequency and wavelength, but they differ in their direction of propagation. Therefore, for each combination of k and l, there are two physical modes with opposite directions of propagation for the non-rotating linearized shallow-water equations.

**Phase velocity for the two modes**:

$$c_{p+} = \frac{\omega_+}{\kappa} = \frac{\sqrt{gH\left(k^2+l^2\right)}}{\sqrt{k^2+l^2}} = \sqrt{\frac{gH\left(k^2+l^2\right)}{k^2+l^2}} = \sqrt{gH}$$

$$c_{p-} = \frac{\omega_-}{\kappa} = -\frac{\sqrt{gH\left(k^2+l^2\right)}}{\sqrt{k^2+l^2}} = -\sqrt{\frac{gH\left(k^2+l^2\right)}{k^2+l^2}} = -\sqrt{gH}$$

Note$=\kappa^2 = k^2 + l^2$

**Group velocity for the two modes**:

$$c_{g+} = \left(c_{gx^+}, c_{gy^+}\right) = \left(\frac{\partial \omega_+}{\partial k}, \frac{\partial \omega_+}{\partial l}\right) = \left(\frac{gHk}{\sqrt{gH\left(k^2+l^2\right)}}, \frac{gHl}{\sqrt{gH\left(k^2+l^2\right)}}\right) = \left(\frac{ghk}{\omega_+}, \frac{ghl}{\omega_+}\right)$$

$$c_{g-} = \left(c_{gx^-}, c_{gy^-}\right) = \left(\frac{\partial \omega_-}{\partial k}, \frac{\partial \omega_-}{\partial l}\right) = \left(\frac{-gHk}{\sqrt{gH\left(k^2+l^2\right)}}, \frac{-gHl}{\sqrt{gH\left(k^2+l^2\right)}}\right) = \left(\frac{-gHk}{\omega_-}, \frac{-ghl}{\omega_-}\right)$$

To check for dispersive or non dispersive, Lets take the magnitude of either $c_g$ so we can compare it to magnitude of $c_p$

$$|c_g| = \sqrt{\left(\frac{ghk}{\omega}\right)^2 + \left(\frac{gHk}{\omega}\right)^2} = \sqrt{gH} = c$$

$$|c_p| = \sqrt{gH} = c$$

Since

$$|c_g| = |c_p|,$$

the waves are **non dispersive**. Non-dispersive waves are waves in which all the different frequencies travel at the same speed.

## QUESTION 2

From (7) we got:

$$\frac{\partial^2\eta}{\partial t^2} = gH\left(\frac{\partial^2\eta}{\partial x^2} + \frac{\partial^2\eta}{\partial y^2}\right) \tag{11}$$

Taking Fourier transform with respect to space( x and y)

$$\frac{\partial^2\hat{\eta}}{\partial t^2} = gH\left(i^2k^2\hat{\eta} + i^2l^2\hat{\eta}\right) \tag{12}$$

$$\frac{\partial^2\hat{\eta}}{\partial t^2} = -gH\left(k^2 + l^2\right)\hat{\eta} \tag{13}$$

$$\frac{\partial^2\hat{\eta}}{\partial t^2} + gH\left(k^2 + l^2\right)\hat{\eta} = 0 \tag{14}$$

Let $\hat{\eta} = e^{rt}$ and substituting it into (14)

$$r^2e^{rt} + gH\left(k^2 + l^2\right)e^{rt} = 0 \tag{15}$$

$$r^2 + gH\left(k^2 + l^2\right) = 0 \tag{16}$$

$$r = \pm gH\left(k^2 + l^2\right), \omega = gH\left(k^2 + l^2\right) \tag{17}$$

$$r = \pm\omega \tag{18}$$

Thus

$$\hat{\eta} = Ae^{\omega t} + Be^{-\omega t} \tag{19}$$

Given initial condition: $\eta\left(x, y, 0\right) = f(x, y)$ Taking the fourier transform of the initial conditio, we get:

$$\hat{\eta}\left(k, l, 0\right) = \hat{f}(k, l) \tag{20}$$

From (19), the general solution is:

$$\hat{\eta}\left(k, l, \omega\right) = A\left(k, l\right)e^{\omega t} + B\left(k, l\right)e^{-\omega t} \tag{21}$$

At t=0:

$$\hat{\eta}\left(k, l, 0\right) = A\left(k, l\right)e^{\omega(0)} + B\left(k, l\right)e^{-\omega(0)} \tag{22}$$

$$\hat{\eta}\left(k, l, 0\right) = A\left(k, l\right) + B\left(k, l\right) \tag{23}$$

Applying the initial condition:

$$\hat{\eta}\left(k, l, 0\right) = \hat{f}(k, l) = A\left(k, l\right) + B\left(k, l\right) \tag{24}$$

$$\hat{f}(k, l) = A\left(k, l\right) + B\left(k, l\right) \tag{25}$$

since both $A$ and $B$ are constant we assume their addition is equal to a constant $C$. Thus, our general solution becomes:

$$\hat{\eta}\left(k, l, \omega\right) = \hat{f}(k, l)\left[e^{\omega t} + e^{-\omega t}\right] \tag{26}$$

Now we find the solution for $\eta\left(x, y, t\right)$ using the inverse Fourier transform:

$$\eta\left(x, y, t\right) = \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\hat{\eta}\left(k, l, \omega\right)e^{i(kx+ly)}\,dk\,dl \tag{27}$$

$$\eta\,(x, y, t) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}(k, l) \left[e^{\omega t} + e^{-\omega t}\right] e^{i(kx+ly)}\, dk\, dl \tag{28}$$

since $2cos(\omega t) = e^{\omega t} + e^{-\omega t}$, the solution becomes

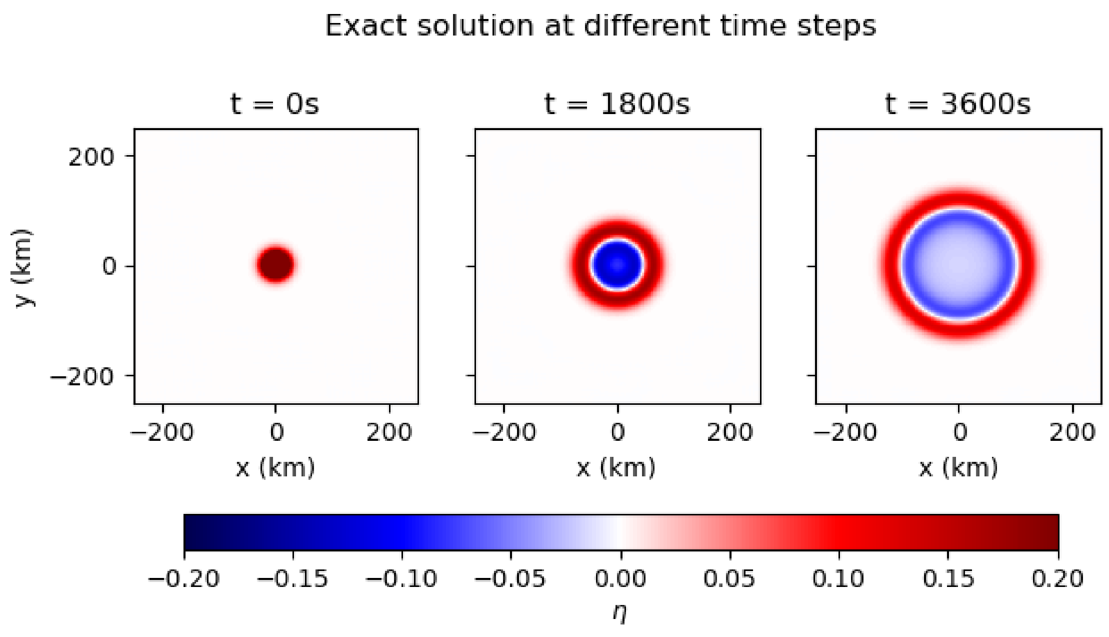
$$\eta\,(x, y, t) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} 2\hat{f}(k, l) cos(\omega t) e^{i(kx+ly)}\, dk\, dl \tag{29}$$

$$\boxed{\boldsymbol{\eta\,(x, y, t) = \tfrac{1}{2(\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}(k, l) cos(\omega t) e^{i(kx+ly)}\, dk\, dl.}} \tag{30}$$

Thus, this is the full analytical solution for $\eta\,(x, y, t)$. It is difficult to solve it physically but computer programming languages has alot of modules and functions to make the mathematics easier.

## QUESTION 3

```
[1]: # Modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import  fftshift,fft2, ifft2, ifftshift
import warnings
warnings.simplefilter('ignore')

# Constants
Lx, Ly = 500000, 500000
ax, ay = 20000, 20000
dx, dy = 2000, 2000
Nx, Ny = Lx/dx, Ly/dy
m = 2
g = 9.81
H = 100

# Domain
x = np.arange(-250000,250000+dx,dx)
y = np.arange(-250000,250000+dx,dy)
x_grid, y_grid = np.meshgrid(x, y)

# Wavenumbers
k=2*np.pi*fftshift(np.fft.fftfreq(len(x), dx))
l=2*np.pi*fftshift(np.fft.fftfreq(len(y), dy))
k_grid, l_grid = np.meshgrid(k,l)

# Dispersion relation
omega = np.sqrt(g*H*(k_grid**2+l_grid**2))

# Initial condition
fxy = np.exp(-(x_grid / ax) ** m) * np.exp(-(y_grid / ay) ** m)
ffxy = fftshift(fft2(fxy))

# Exact solution
time_step = [0,1800,3600]
exact_timestep = np.zeros((3,len(x),len(y)))
exact_solution = np.zeros((3,len(x),len(y)))
for i in range(3):
    exact_timestep[i] = ffxy*np.cos(omega*time_step[i])
    exact_solution[i] = np.real(ifft2(ifftshift(exact_timestep[i])))

# Plot
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(7, 2), sharey=True)

for i, t in enumerate(time_step):
    color_plot = axs[i].pcolormesh(x / 1e3, y / 1e3, exact_solution[i],␣
↪cmap='seismic', vmin=-0.2, vmax=0.2)
    axs[i].set_title(f't = {t}s')
    axs[i].set_xlabel('x (km)')
    if i == 0:
        axs[i].set_ylabel('y (km)')

# Create a colorbar with a separate axis and position it below the subplots
cbar_ax = fig.add_axes([0.165, -0.3, 0.7, 0.1])
fig.colorbar(color_plot, cax=cbar_ax, orientation='horizontal', label='$\eta$')
plt.suptitle('Exact solution at different time steps', y=1.2)
plt.savefig('exact.jpg', dpi=300) #saving plot
plt.show()
```



**QUESTION 4**
A suitable numerical method that i will be using to solve the shallow water equations is the **Leapfrog scheme in time and centered in space scheme**. It is conditionally stable and has 2nd-order accuracy in both time and space.

To show the above properties of the scheme by mathematical analysis, let's reduce the 2D problem in $(1)-(3)$ to a 1D problem in $x-t$ for simplicity. The 1D shallow water equations can be written in the conservative form

$$\frac{\partial^2 \eta}{\partial t^2} = gH\left(\frac{\partial^2 \eta}{\partial x^2}\right) \tag{31}$$

Let

$$gH = c^2 \tag{32}$$

Discretizing (31) using the scheme gives:

$$\frac{\phi_i^{n+1} - 2\phi_i^n + \phi_i^{n-1}}{(\Delta t)^2} = c^2\left(\frac{\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n}{(\Delta x)^2}\right) \tag{33}$$

To find the accuracy we need to find the truncation error. Let do relevant Taylor series expansion :

$$\psi_i^{n+1} = \psi_i^n + \Delta t\frac{\partial \psi}{\partial t} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} + \frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5) \tag{34}$$

$$\psi_i^{n-1} = \psi_i^n - \Delta t\frac{\partial \psi}{\partial t} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} - \frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5) \tag{35}$$

$$\psi_{i+1}^n = \psi_i^n + \Delta x\frac{\partial \psi}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 \psi}{\partial x^2} + \frac{(\Delta x)^3}{6}\frac{\partial^3 \psi}{\partial x^3} + \frac{(\Delta x)^4}{24}\frac{\partial^4 \psi}{\partial x^4} + \mathcal{O}((\Delta x)^5) \tag{36}$$

$$\psi_{i+1}^n = \psi_i^n - \Delta x\frac{\partial \psi}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 \psi}{\partial x^2} - \frac{(\Delta x)^3}{6}\frac{\partial^3 \psi}{\partial x^3} + \frac{(\Delta x)^4}{24}\frac{\partial^4 \psi}{\partial x^4} + \mathcal{O}((\Delta x)^5) \tag{37}$$

Now lets substitute into (33) and solve it.To simplify matters I will do this one term at a time, beginning with the left term

**LHS** term

$$\begin{aligned}\frac{1}{(\Delta t)^2}\Bigg[&\psi_i^n + \Delta t\frac{\partial \psi}{\partial t} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} + \frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5) - 2\psi_i^n \\ &+ \psi_i^n - \Delta t\frac{\partial \psi}{\partial t} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} - \frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5)\Bigg]\end{aligned} \tag{38}$$

$$\begin{aligned}\frac{1}{(\Delta t)^2}\Bigg[&\cancel{\psi_i^n} + \cancel{\Delta t\frac{\partial \psi}{\partial t}} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} + \cancel{\frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3}} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5) - 2\cancel{\psi_i^n} \\ &+ \cancel{\psi_i^n} - \cancel{\Delta t\frac{\partial \psi}{\partial t}} + \frac{(\Delta t)^2}{2}\frac{\partial^2 \psi}{\partial t^2} - \cancel{\frac{(\Delta t)^3}{6}\frac{\partial^3 \psi}{\partial t^3}} + \frac{(\Delta t)^4}{24}\frac{\partial^4 \psi}{\partial t^4} + \mathcal{O}((\Delta t)^5)\Bigg]\end{aligned} \tag{39}$$

$$\frac{1}{(\Delta t)^2}\left[(\Delta t)^2\frac{\partial^2\psi}{\partial t^2}+\frac{(\Delta t)^4}{12}\frac{\partial^4\psi}{\partial t^4}+\mathcal{O}((\Delta t)^5)\right] \tag{40}$$

$$=\frac{\partial^2\psi}{\partial t^2}+\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial t^4}+\mathcal{O}((\Delta t)^3) \tag{41}$$

Now to the **RHS** term:

$$\begin{aligned}\frac{c^2}{(\Delta x)^2}\Bigg[&\psi_i^n+\Delta x\frac{\partial\psi}{\partial x}+\frac{(\Delta x)^2}{2}\frac{\partial^2\psi}{\partial x^2}+\frac{(\Delta x)^3}{6}\frac{\partial^3\psi}{\partial t^3}+\frac{(\Delta t)^4}{24}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^5)-2\psi_i^n\\&+\psi_i^n-\Delta x\frac{\partial\psi}{\partial x}+\frac{(\Delta x)^2}{2}\frac{\partial^2\psi}{\partial x^2}-\frac{(\Delta x)^3}{6}\frac{\partial^3\psi}{\partial x^3}+\frac{(\Delta x)^4}{24}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^5)\Bigg]\end{aligned} \tag{42}$$

$$\begin{aligned}\frac{c^2}{(\Delta x)^2}\Bigg[&\cancel{\psi_i^n}+\cancel{\Delta x\frac{\partial\psi}{\partial x}}+\frac{(\Delta x)^2}{2}\frac{\partial^2\psi}{\partial x^2}+\cancel{\frac{(\Delta x)^3}{6}\frac{\partial^3\psi}{\partial x^3}}+\frac{(\Delta x)^4}{24}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^5)-2\cancel{\psi_i^n}\\&+\cancel{\psi_i^n}-\cancel{\Delta t\frac{\partial\psi}{\partial x}}+\frac{(\Delta x)^2}{2}\frac{\partial^2\psi}{\partial x^2}-\cancel{\frac{(\Delta x)^3}{6}\frac{\partial^3\psi}{\partial x^3}}+\frac{(\Delta x)^4}{24}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^5)\Bigg]\end{aligned} \tag{43}$$

$$\frac{c^2}{(\Delta x)^2}\left[(\Delta x)^2\frac{\partial^2\psi}{\partial x^2}+\frac{(\Delta x)^4}{12}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^5)\right] \tag{44}$$

$$=c^2\frac{\partial^2\psi}{\partial x^2}+c^2\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta x)^3) \tag{45}$$

Bringing them together, we have

$$t^n=\frac{\partial^2\psi}{\partial t^2}+\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial t^4}-c^2\frac{\partial^2\psi}{\partial x^2}-c^2\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta t)^3)+\mathcal{O}((\Delta x)^3) \tag{46}$$

Thus this is the truncation error:

$$\boxed{\boldsymbol{t^n=\frac{\partial^2\psi}{\partial t^2}-c^2\frac{\partial^2\psi}{\partial x^2}+\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial t^4}-c^2\frac{(\Delta t)^2}{12}\frac{\partial^4\psi}{\partial x^4}+\mathcal{O}((\Delta t)^3)+\mathcal{O}((\Delta x)^3)}} \tag{47}$$

The leading-order errors are thus $(\Delta t)^2$ and $(\Delta x)^2$. Therefore the scheme is second-order accurate in both time and space.

**Stability:** The stability condition is obtained through a Von Neumann analysis. Let

$$\phi_i^n=A_k^n e^{ikj\Delta x} \tag{48}$$

Substitute (48) into (33):

$$\frac{A_k^{n+1}e^{ikj\Delta x}-2A_k^n e^{ikj\Delta x}+A_k^{n-1}e^{ikj\Delta x}}{(\Delta t)^2}=c^2\left(\frac{A_k^n e^{i(j+1)k\Delta x}-2A_k^n e^{ikj\Delta x}+A_k^n e^{i(j-1)k\Delta x}}{(\Delta x)^2}\right) \tag{49}$$

$$\frac{A_k - 2 + A_k^{-1}}{(\Delta t)^2} = c^2\left(\frac{e^{ik\Delta x} - 2 + e^{-ik\Delta x}}{(\Delta x)^2}\right) \tag{50}$$

$$A_k - 2 + A_k^{-1} = \frac{c^2(\Delta t)^2}{(\Delta x)^2}\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right) \tag{51}$$

$$A_k^2 - 2A_k + 1 = \frac{c^2(\Delta t)^2}{(\Delta x)^2}\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right)A_k \tag{52}$$

$$A_k^2 - 2\left[1 - \frac{c^2(\Delta t)^2}{(\Delta x)^2}\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right)\right]A_k + 1 = 0 \tag{53}$$

We know:

$$CFL = \mu = \frac{(c\Delta t)}{(\Delta x)} \tag{54}$$

$$2\cos(k\Delta x) = e^{ik\Delta x} + e^{-ik\Delta x} \tag{55}$$

substituting (54) and (55), we get:

$$A_k^2 - 2\left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]A_k + 1 = 0 \tag{56}$$

The roots of $A_k^2 - 2\left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]A_k + 1 = 0$ are $A_k^2 = \left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right] \pm \sqrt{\left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]^2 - 1}$. Everything depends on that square root giving an imaginary number when $\left[1 - \mu^2 + \mu^2\cos k\Delta x\right]^2 \leq 1$:

If $\left[1-\mu^2+\mu^2\cos(k\Delta x)\right]^2 \leq 1$, then $A_k = \left[1-\mu^2+\mu^2\cos(k\Delta x)\right] \pm i\sqrt{1 - \left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]^2}$ has $|A_k|^2 = \left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]^2 + (1 - \left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]^2) = 1$. The CFL condition $\mu \leq 1$ does produce $\left[1 - \mu^2 + \mu^2\cos(k\Delta x)\right]^2 \leq 1$ and the scheme is stable:

If $\mu \leq 1$, then $\left|1 - \mu^2 + \mu^2\cos k\Delta x\right| \leq (1 - \mu^2) + \mu^2 = 1$. Thus since $\cos k\Delta x$ ranges from -1 or 1, we need to ensure $\mu \leq 1$ which means the solution is stable if:

$$\boxed{\boldsymbol{\mu^2 = \frac{c^2(\Delta t)^2}{(\Delta x)^2} \leq 1}} \tag{57}$$

This shows the scheme is conditionally stable. Applying this idea to the original $2 - d$ form of the equation.

The $2-d$ will have the form:

$$A_k^2 - 2\left(1 - \left(\frac{c\Delta t}{\Delta x}\right)^2 (1-\cos k\Delta x) - \left(\frac{c\Delta t}{\Delta y}\right)^2 (1-\cos l\Delta y)\right) A_k + 1 = 0. \tag{58}$$

Both roots must have $|A_k| = 1$ for stability. This still requires $-1 \leq 2\left(1 - \left(\frac{c\Delta t}{\Delta x}\right)^2 (1-\cos k\Delta x) - \left(\frac{c\Delta t}{\Delta y}\right)^2 (1-\cos l\Delta y)\right) \leq 1$. When the cosines are $-1$ (the dangerous value), we find the stability condition:

$$-1 \leq 1 - 2\left(\frac{c\Delta t}{\Delta x}\right)^2 - 2\left(\frac{c\Delta t}{\Delta y}\right)^2 \tag{59}$$

needs

$$\left(\frac{c\Delta t}{\Delta x}\right)^2 + \left(\frac{c\Delta t}{\Delta y}\right)^2 \leq 1 \tag{60}$$

Thus the stability for the $2-d$ of our equation is:

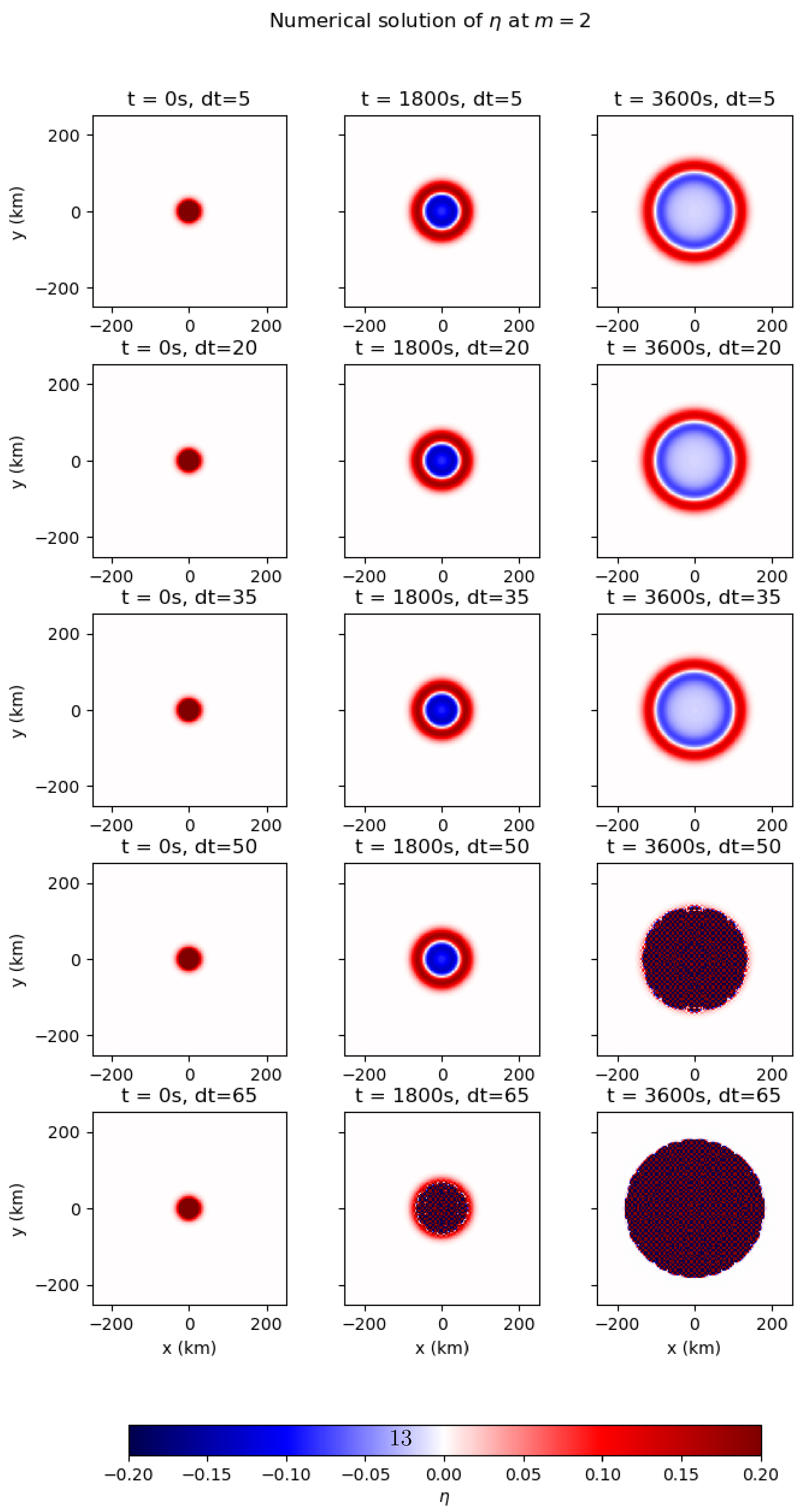
$$\boxed{\boldsymbol{\left(\frac{c\Delta t}{\Delta x}\right)^2 + \left(\frac{c\Delta t}{\Delta y}\right)^2 \leq 1}} \tag{61}$$

Given $\Delta x = 2000m, \Delta y = 2000m$, for $\Delta x = \Delta y$, it means the grid is a square, thus base on (61), this implies

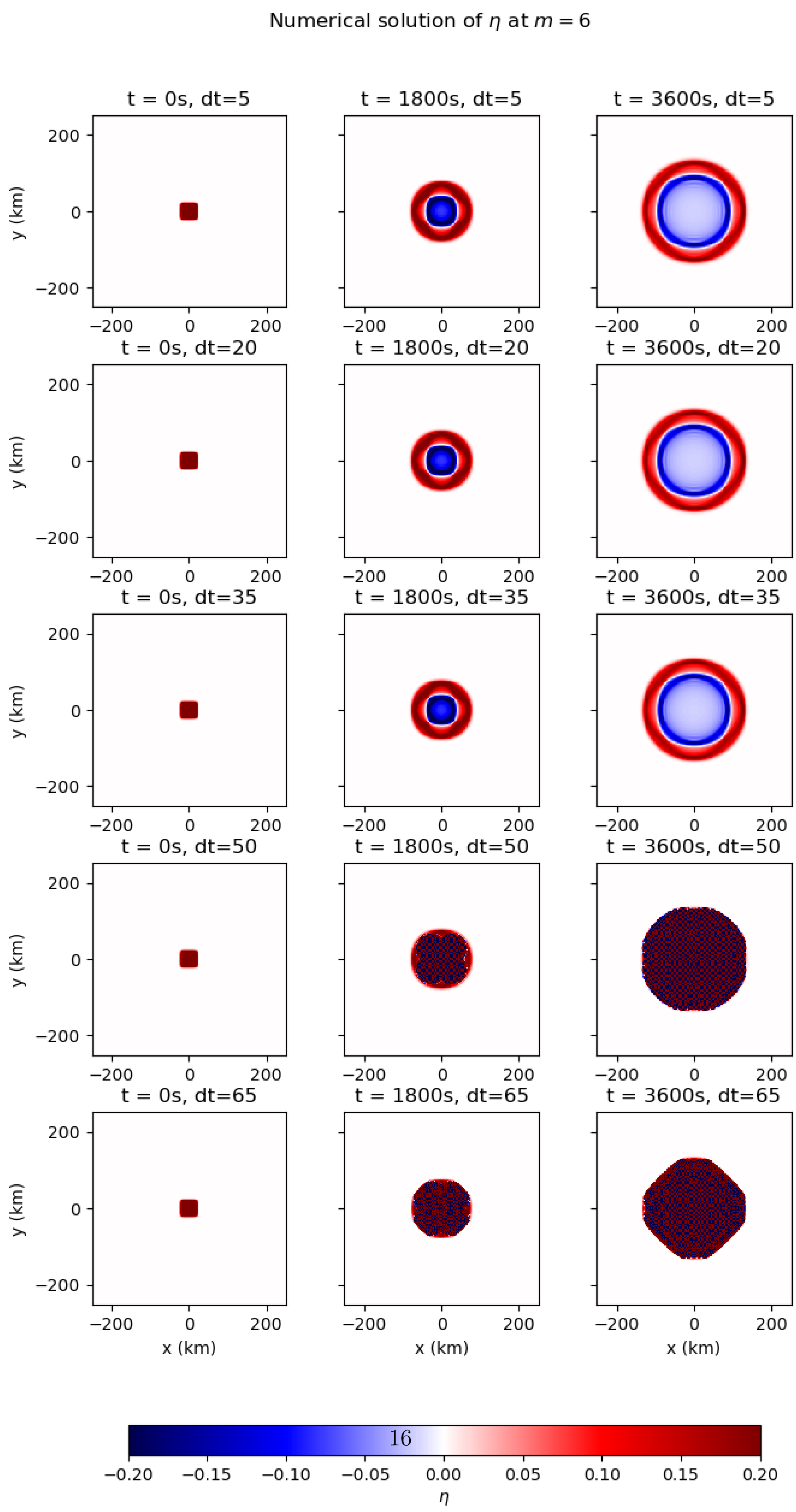
$$\boxed{\Delta t \leq \frac{\Delta x}{c\sqrt{2}}} \tag{62}$$

## QUESTION 5

[2]:
```python
# Modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fftshift, fft2, ifft2, ifftshift

# Constants
Lx, Ly = 500000, 500000
ax, ay = 20000, 20000
dx, dy = 2000, 2000
Nx, Ny = Lx / dx, Ly / dy
m = 2
g = 9.81
H = 100

# Domain
x = np.arange(-250000, 250000 + dx, dx)
y = np.arange(-250000, 250000 + dx, dy)
x_grid, y_grid = np.meshgrid(x, y)

# Wavenumbers
k = 2 * np.pi * fftshift(np.fft.fftfreq(len(x), dx))
l = 2 * np.pi * fftshift(np.fft.fftfreq(len(y), dy))
k_grid, l_grid = np.meshgrid(k, l)

# Dispersion relation
omega = np.sqrt(g * H * (k_grid ** 2 + l_grid ** 2))

# Initial condition
fxy = np.exp(-(x_grid / ax) ** m) * np.exp(-(y_grid / ay) ** m)
ffxy = fftshift(fft2(fxy))
dt = [5, 20, 35, 50, 65]
num_rows = len(dt)
num_cols = 3
subplot_size = 2.5  # You can change this value for different subplot sizes
fig, axs = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(subplot_size *␣
↪num_cols, subplot_size * num_rows), sharey=True)
for item in range(len(dt)):
    mu_x = np.sqrt(g * H) * dt[item] / dx
    mu_y = np.sqrt(g * H) * dt[item] / dy
    t = np.arange(0, 3601, dt[item])
    Numerical_solution = np.zeros((len(x), len(y), len(t)))
    Numerical_solution[:, :, 0] = fxy
    Numerical_solution[:, :, 1] = fxy

    for n in range(1, len(t) - 1):
        for i in range(1, len(x) - 1):
            for j in range(1, len(y) - 1):
                Numerical_solution[i, j, n + 1] = (
                    2 * Numerical_solution[i, j, n]
                    - Numerical_solution[i, j, n - 1]
                    + mu_x ** 2 * (
                        Numerical_solution[i + 1, j, n]
                        + Numerical_solution[i - 1, j, n]
                        - 2 * Numerical_solution[i, j, n]
                    )
                    + mu_y ** 2 * (
                        Numerical_solution[i, j + 1, n]
                        + Numerical_solution[i, j - 1, n]
                        - 2 * Numerical_solution[i, j, n]
                    )
                )

    time_step = [0, 1800, 3600]
    new_dt = dt[item]

    for index, data in enumerate(time_step):
        color_plot = axs[item, index].pcolormesh(x / 1e3, y / 1e3,␣
 ↪Numerical_solution[:,:, int(data/new_dt)],
                                        cmap='seismic', vmin=-0.2, vmax=0.2)
        axs[item, index].set_title(f't = {data}s, dt={new_dt}')
        if index == 0:
            axs[item, index].set_ylabel('y (km)')
        if item == 4:
            axs[item, index].set_xlabel('x (km)')

# Create a colorbar with a separate axis and position it below the subplots
cbar_ax = fig.add_axes([0.165, 0.01, 0.7, 0.02])
fig.colorbar(color_plot, cax=cbar_ax, orientation='horizontal', label='$\eta$')
plt.subplots_adjust(hspace=0.3, wspace=0.3, top=0.9) # Adjust the top value to␣
 ↪create more space
plt.suptitle('Numerical solution of $\eta$ at $m=2$', y=0.97) # Increase the y␣
 ↪value to move the title upwards
plt.savefig('m2.jpg', dpi=300, bbox_inches='tight')  # Add bbox_inches='tight'
plt.show()
```


Numerical solution of $\eta$ at $m = 2$
t = 0s, dt=5
t = 1800s, dt=5
t = 3600s, dt=5
t = 0s, dt=20
t = 1800s, dt=20
t = 3600s, dt=20
t = 0s, dt=35
t = 1800s, dt=35
t = 3600s, dt=35
t = 0s, dt=50
t = 1800s, dt=50
t = 3600s, dt=50
t = 0s, dt=65
t = 1800s, dt=65
t = 3600s, dt=65
y (km)
x (km)
200
0
−200
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
$\eta$

[3]:
```
# Modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fftshift, fft2, ifft2, ifftshift

# Constants
Lx, Ly = 500000, 500000
ax, ay = 20000, 20000
dx, dy = 2000, 2000
Nx, Ny = Lx / dx, Ly / dy
m = 6
g = 9.81
H = 100

# Domain
x = np.arange(-250000, 250000 + dx, dx)
y = np.arange(-250000, 250000 + dx, dy)
x_grid, y_grid = np.meshgrid(x, y)

# Wavenumbers
k = 2 * np.pi * fftshift(np.fft.fftfreq(len(x), dx))
l = 2 * np.pi * fftshift(np.fft.fftfreq(len(y), dy))
k_grid, l_grid = np.meshgrid(k, l)

# Dispersion relation
omega = np.sqrt(g * H * (k_grid ** 2 + l_grid ** 2))

# Initial condition
fxy = np.exp(-(x_grid / ax) ** m) * np.exp(-(y_grid / ay) ** m)
ffxy = fftshift(fft2(fxy))
dt = [5, 20, 35, 50, 65]
num_rows = len(dt)
num_cols = 3
subplot_size = 2.5  # You can change this value for different subplot sizes
fig, axs = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(subplot_size *␣
↪num_cols, subplot_size * num_rows), sharey=True)
for item in range(len(dt)):
    mu_x = np.sqrt(g * H) * dt[item] / dx
    mu_y = np.sqrt(g * H) * dt[item] / dy
    t = np.arange(0, 3601, dt[item])
    Numerical_solution = np.zeros((len(x), len(y), len(t)))
    Numerical_solution[:, :, 0] = fxy
    Numerical_solution[:, :, 1] = fxy
    for n in range(1, len(t) - 1):
        for i in range(1, len(x) - 1):
            for j in range(1, len(y) - 1):
                Numerical_solution[i, j, n + 1] = (
                    2 * Numerical_solution[i, j, n]
                    - Numerical_solution[i, j, n - 1]
                    + mu_x ** 2 * (
                        Numerical_solution[i + 1, j, n]
                        + Numerical_solution[i - 1, j, n]
                        - 2 * Numerical_solution[i, j, n]
                    )
                    + mu_y ** 2 * (
                        Numerical_solution[i, j + 1, n]
                        + Numerical_solution[i, j - 1, n]
                        - 2 * Numerical_solution[i, j, n]
                    )
                )

    time_step = [0, 1800, 3600]
    new_dt = dt[item]

    for index, data in enumerate(time_step):
        color_plot = axs[item, index].pcolormesh(x / 1e3, y / 1e3,␣
↪Numerical_solution[:,:, int(data/new_dt)],
                                        cmap='seismic', vmin=-0.2, vmax=0.2)
        axs[item, index].set_title(f't = {data}s, dt={new_dt}')
        if index == 0:
            axs[item, index].set_ylabel('y (km)')
        if item == 4:
            axs[item, index].set_xlabel('x (km)')

# Create a colorbar with a separate axis and position it below the subplots
cbar_ax = fig.add_axes([0.165, 0.01, 0.7, 0.02])
fig.colorbar(color_plot, cax=cbar_ax, orientation='horizontal', label='$\eta$')
plt.subplots_adjust(hspace=0.3, wspace=0.3, top=0.9) # Adjust the top value to␣
↪create more space
plt.suptitle('Numerical solution of $\eta$ at $m=2$', y=0.97) # Increase the y␣
↪value to move the title upwards
plt.savefig('m6.jpg', dpi=300, bbox_inches='tight')  # Add bbox_inches='tight'
plt.show()
```

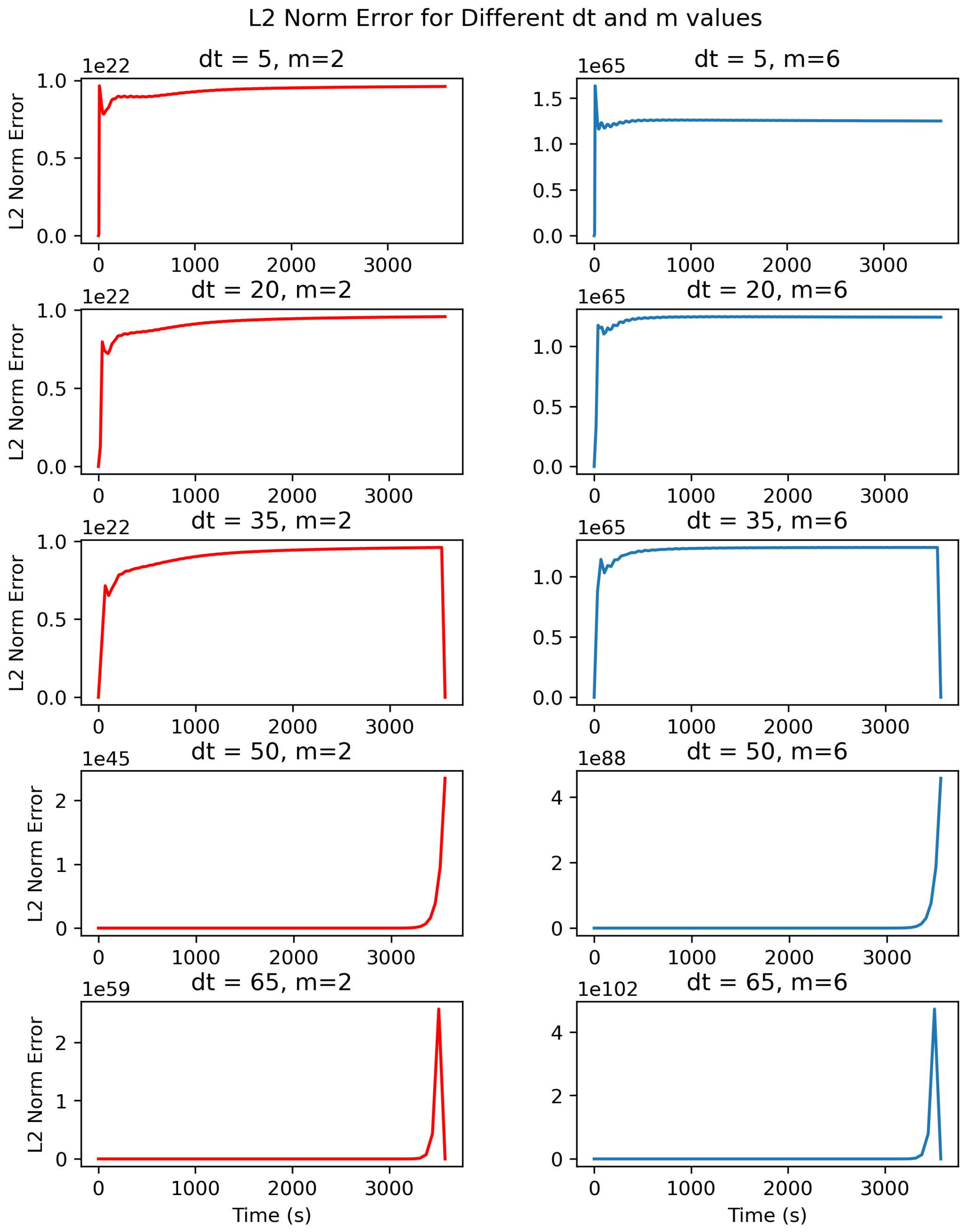

Numerical solution of $\eta$ at $m = 6$
t = 0s, dt=5
t = 1800s, dt=5
t = 3600s, dt=5
t = 0s, dt=20
t = 1800s, dt=20
t = 3600s, dt=20
t = 0s, dt=35
t = 1800s, dt=35
t = 3600s, dt=35
t = 0s, dt=50
t = 1800s, dt=50
t = 3600s, dt=50
t = 0s, dt=65
t = 1800s, dt=65
t = 3600s, dt=65
y (km)
x (km)
200
0
−200
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
$\eta$

From the spatial plots above, it can be seen that as the time step (dt) increases, the numerical simulation becomes less accurate and unstable and this is due to the larger time intervals between each update. From the stability equation for the scheme in $2-d$, it was shown that $\Delta t \leq \frac{\Delta x}{c\sqrt{2}}$, Solving this using the values given for $c^2 = gH$ and $\Delta x$, we get approximately $45.2s$. Thus this prove the reason why at $\Delta t = 5, \Delta t = 20 and \Delta t = 35$, there plots is the same as compared to the exact because it obeys the condition for the scheme to be stable. As $\Delta t$ increases above $45s$, the scheme deviates from the exact and it can be seen from the plot of $\Delta t = 50$and $\Delta t = 65$. The numerical solution is an approximation of the continuous solution, and the accuracy depends on the size of the time step. When the time step is too large, the method might fail to capture the dynamics of the system, leading to deviations from the analytical solution. Similarly, as the parameter 'm' increases, the initial condition becomes more localized and concentrated, resulting in steeper gradients. Steeper gradients might require smaller time steps and spatial discretization to capture the dynamics accurately. With larger 'm' values, the numerical solution might not be able to resolve the sharp features of the problem, and hence, it deviates from the analytical solution.

**QUESTION 6**

[6]:
```python
def FDCS_AN(dx,dy,dt,m,ffxy,X,Y,g=9.81,H =100):

    T = int(3600/dt)
    T_var = np.arange(0,3601,dt)
    x = np.arange(-250000, 250000 + dx, dx)
    y = np.arange(-250000, 250000 + dx, dy)
    Eta_ana = np.zeros((T,len(X),len(Y)))
    FEta_ts = np.zeros((T,len(X),len(Y)))
    for i in range(T):
        FEta_ts[i] = ffxy*np.cos(omega*T_var[i])
        Eta_ana[i] = np.real(ifft2(ifftshift(FEta_ts[i])))

    CFL_x = np.sqrt(g*H)*dt/dx
    CFL_y = np.sqrt(g*H)*dt/dy


    eta_FD = np.zeros((len(x), len(y), T))
    eta_FD[:, :, 0] = Eta_ana[0]
    eta_FD[:, :, 1] = Eta_ana[0]
    for n in range(1, T-1):
        for i in range(1, len(X)-1):
            for j in range(1, len(Y)-1):
                eta_FD[i,j, n+1] = 2 * eta_FD[i, j, n] - eta_FD[i,j, n-1] +␣
 ↪CFL_x**2 * (eta_FD[i+1, j,n] + eta_FD[i-1, j,n] - 2 * eta_FD[i,j, n]) +␣
 ↪CFL_y**2 * (eta_FD[i, j+1 ,n] + eta_FD[i, j-1 ,n] - 2 * eta_FD[i, j, n])
    return Eta_ana, eta_FD
m1 = 2
m2 = 6
fxy = np.exp(-(x_grid / ax) * m1) * np.exp(-(y_grid / ay) * m1)
fxyy = np.exp(-(x_grid / ax) * m2) * np.exp(-(y_grid / ay) * m2)
ffxy = fftshift(fft2(fxy))
ffxyy = fftshift(fft2(fxyy))

Eta_ana_5 , eta_FD_5 = FDCS_AN(dx,dy,5,m1,ffxy,x,y,g=9.81,H =100)
Eta_ana_20 , eta_FD_20 = FDCS_AN(dx,dy,20,m1,ffxy,x,y,g=9.81,H =100)
Eta_ana_35 , eta_FD_35 = FDCS_AN(dx,dy,35,m1,ffxy,x,y,g=9.81,H =100)
Eta_ana_50 , eta_FD_50 = FDCS_AN(dx,dy,50,m1,ffxy,x,y,g=9.81,H =100)
Eta_ana_65 , eta_FD_65 = FDCS_AN(dx,dy,65,m1,ffxy,x,y,g=9.81,H =100)

Eta_ana_5_ , eta_FD_5_ = FDCS_AN(dx,dy,5,m2,ffxy,x,y,g=9.81,H =100)
Eta_ana_20_ , eta_FD_20_ = FDCS_AN(dx,dy,20,m2,ffxy,x,y,g=9.81,H =100)
Eta_ana_35_ , eta_FD_35_ = FDCS_AN(dx,dy,35,m2,ffxy,x,y,g=9.81,H =100)
Eta_ana_50_ , eta_FD_50_ = FDCS_AN(dx,dy,50,m2,ffxy,x,y,g=9.81,H =100)
Eta_ana_65_ , eta_FD_65_ = FDCS_AN(dx,dy,65,m2,ffxy,x,y,g=9.81,H =100)
```

[42]:
```
import numpy as np
import matplotlib.pyplot as plt

# Assuming you have defined the l2_norm_error function and calculated the errors␣
↪(l2_error_5, l2_error_20, ...)

dt_values = [5, 20, 35, 50, 65]
l2_errors_m2 = [l2_error_5, l2_error_20, l2_error_35, l2_error_50, l2_error_65]
l2_errors_m6 = [l2_error_5_, l2_error_20_, l2_error_35_, l2_error_50_,␣
↪l2_error_65_]

fig, axs = plt.subplots(2, 5, figsize=(18, 6))
fig.suptitle('L2 Norm Error for Different dt Values at m=2 and m=6')

for i, dt in enumerate(dt_values):
    axs[0, i].plot(np.arange(0, 3600, dt), l2_errors_m2[i], label=f'dt = {dt},␣
↪m=2')
    axs[0, i].set_title(f'dt = {dt}, m=2')
    axs[1, i].plot(np.arange(0, 3600, dt), l2_errors_m6[i], label=f'dt = {dt},␣
↪m=6')
    axs[1, i].set_title(f'dt = {dt}, m=6')
    if i == 0:
        axs[0, i].set_ylabel('L2 Norm Error (m=2)')
        axs[1, i].set_ylabel('L2 Norm Error (m=6)')
    axs[1, i].set_xlabel('Time (s)')

plt.show()
plt.suptitle('L2 norm error at m=2 and m=6',) # Increase the y value to move the␣
↪title upwards
plt.subplots_adjust(hspace=0.3, top=0.9) # Adjust the top value to create more␣
↪space
plt.savefig('pics.jpg', dpi=300, bbox_inches='tight')  # Add bbox_inches='tight'
plt.show()
```



Based on the previous answer, we can quantitatively analyze how the L2 error may be affected by the time step (dt) and the parameter 'm': As the time step increases, the L2 error increases because the numerical solution becomes less accurate due to the larger time intervals between each update as explain above in Question 5. The method's accuracy depends on the size of the time step, and a larger time step might fail to capture the dynamics of the system, leading to deviations from the analytical solution. Therefore, a larger dt would result in a higher L2 error. As 'm' increases, the initial condition becomes more localized and concentrated, leading to steeper gradients in the problem. The L2 error increases for larger 'm' values because the numerical solution is not able to resolve the sharp features of the problem accurately with the given spatial discretization (dx and dy). The steepness leads to increased errors in the approximation, and hence a higher L2 error. It can be seen that at m=2, the error across the five $\Delta t$ ranges from $1 \times 10^{22}$ to $1 \times 10^{59}$ while at m=6, the error across the five $\Delta t$ ranges from $1 \times 10^{65}$to $1 \times 10^{102}$ which confirms that as m increases the l2 error increases.

---

Thank you Prof. Daniel Kirshbaum, the class was a great one with good practical assignments and project.