Assignment 4
Numerical solution of time-dependent problems

Numerical Analysis For PDE's (WI3730TU)
Kaloyan Yanchev

October 2024

1. Upper bound on the maximum eigenvalue

   1.1. The general formula of some row of A for arbitrary coordinates is given by:
   $f_{i,j} = -\frac{k_{i-0.5,j}}{h_x^2}u_{i-1,j} - \frac{k_{i,j-0.5}}{h_y^2}u_{i,j-1} + (\frac{k_{i-0.5,j}}{h_x^2} + \frac{k_{i,j-0.5}}{h_y^2} + \frac{k_{i+0.5,j}}{h_x^2} + \frac{k_{i,j+0.5}}{h_y^2})u_{i,j} - \frac{k_{i+0.5,j}}{h_x^2}u_{i+1,j} - \frac{k_{i,j+0.5}}{h_y^2}u_{i,j+1}$
   Diagonal value: $\frac{k_{i-0.5,j}}{h_x^2} + \frac{k_{i,j-0.5}}{h_y^2} + \frac{k_{i+0.5,j}}{h_x^2} + \frac{k_{i,j+0.5}}{h_y^2}$
   Off-diagonal values: $-\frac{k_{i-0.5,j}}{h_x^2}; -\frac{k_{i,j-0.5}}{h_y^2}; -\frac{k_{i+0.5,j}}{h_x^2}; -\frac{k_{i,j+0.5}}{h_y^2}$

   1.2. Gerschgorin's Lemma: $|\lambda_r - a_{r,r}| \leq \sum_{m\neq r}^{n} |a_{r,m}|$
   $=> -\sum_{m\neq r}^{n} |a_{r,m}| \leq \lambda_r - a_{r,r} \leq \sum_{m\neq r}^{n} |a_{r,m}|$
   Since we are looking at the upper bound, we can ignore the first inequality and focus on the second one only.
   $\lambda_r - a_{r,r} \leq \sum_{m\neq r}^{n} |a_{r,m}|$
   $\lambda_r \leq a_{r,r} + \sum_{m\neq r}^{n} |a_{r,m}|$
   We can now substitute the values of the r^{th} row of A derived in the previous question. However, we first have to note that $a_{r,r} = \sum_{m\neq r}^{n} |a_{r,m}|$ (diagonally dominant). Then we get: $\lambda_r \leq 2a_{r,r}$.
   $\lambda_r \leq 2(\frac{k_{i-0.5,j}}{h_x^2} + \frac{k_{i,j-0.5}}{h_y^2} + \frac{k_{i+0.5,j}}{h_x^2} + \frac{k_{i,j+0.5}}{h_y^2})$
   If we consider the whole matrix A: $\lambda_{max} \leq \max_r 2a_{r,r}$.
   $\lambda_{max} \leq 2\max_{i,j}(\frac{k_{i-0.5,j}}{h_x^2} + \frac{k_{i,j-0.5}}{h_y^2} + \frac{k_{i+0.5,j}}{h_x^2} + \frac{k_{i,j+0.5}}{h_y^2})$
   Without any further information about the problem, this inequality cannot be further simplified without introducing a marginalization error.

2. Diffusion equation
   $\frac{\partial u}{\partial t} - \nabla \cdot (k\nabla u) = f, (x,y) \in \Omega, t \in (0,T]$
   $u(x,y,t) = 0, (x,y) \in \partial\Omega$
   $u(x,y,0) = 0, (x,y) \in \Omega$
   $f(x,y) = e^{-\alpha(x-x_1)^2-\alpha(y-y_1)^2} + e^{-\alpha(x-x_2)^2-\alpha(y-y_2)^2} + e^{-\alpha(x-x_3)^2-\alpha(y-y_3)^2} + e^{-\alpha(x-x_4)^2-\alpha(y-y_4)^2}$
   $$k(x,y) = \begin{cases} 0.1 \text{ if } x < L_x/2, y < L_y/2; \\ 0.4 \text{ if } x < L_x/2, y \geq L_y/2; \\ 0.7 \text{ if } x \geq L_x/2, y \geq L_y/2; \\ 1.0 \text{ if } x \geq L_x/2, y < L_y/2; \end{cases}$$
   where

   - $\overline{\Omega} = [0, L_x] \times [0, L_y]$ is rectangle with $L_x = 10, L_y = 5$
   - $\alpha = 40$
   - $(x_1, y_1) = (0.25L_x, 0.25L_y)$
   - $(x_2, y_2) = (0.25L_x, 0.75L_y)$
   - $(x_3, y_3) = (0.75L_x, 0.75L_y)$
   - $(x_4, y_4) = (0.75L_x, 0.25L_y)$

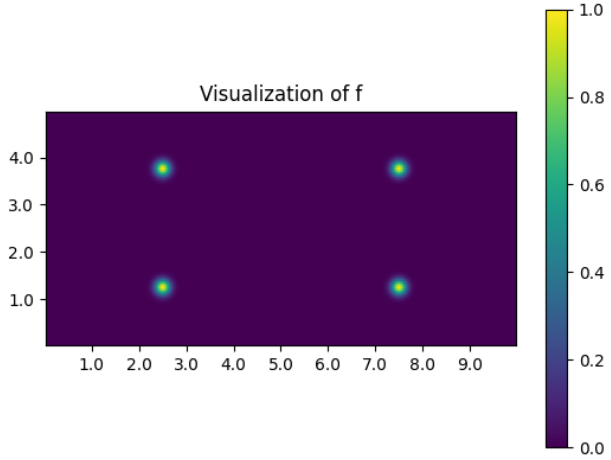
   2.1. Spatial discretization and time integration
   $\mathbf{u}' = -A\mathbf{u} + f$

2.1.1. Forward Euler: $\mathbf{u}^{k+1} = \mathbf{u}^k + h[-A\mathbf{u}^k + \mathbf{f}(t_k, \mathbf{u}^k)] = [I - hA]\mathbf{u}^k + h\mathbf{f}(t_k, \mathbf{u}^k)$
Trapezoidal: $\mathbf{u}^{k+1} = \mathbf{u}^k + \frac{h}{2}[-A\mathbf{u}^k + \mathbf{f}(t_k, \mathbf{u}^k) - A\mathbf{u}^{k+1} + \mathbf{f}(t_{k+1}, \mathbf{u}^{k+1})]$
$=> [I + \frac{h}{2}A]\mathbf{u}^{k+1} = [I - \frac{h}{2}A]\mathbf{u}^k + \frac{h}{2}[\mathbf{f}(t_k, \mathbf{u}^k) + \mathbf{f}(t_{k+1}, \mathbf{u}^{k+1})]$
Since f is time-indifferent then $\mathbf{f}(t_k, \mathbf{u}^k) = \mathbf{f}(t_{k+1}, \mathbf{u}^{k+1})$
$=> \mathbf{u}^{k+1} = [I + \frac{h}{2}A]^{-1}([I - \frac{h}{2}A]\mathbf{u}^k + h\mathbf{f}(t_k, \mathbf{u}^k))$

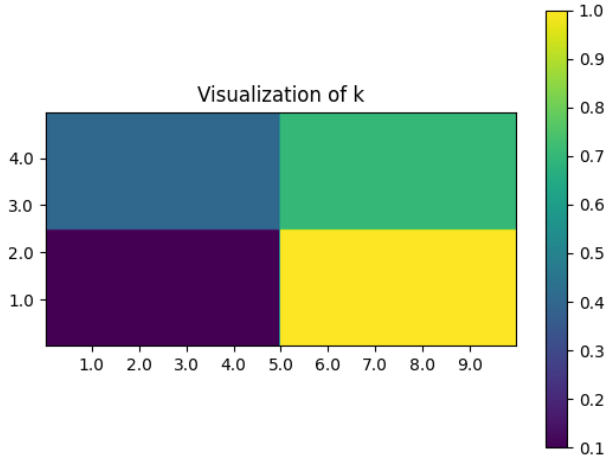
2.1.2. General stability condition: $|1 - h\lambda_j| < 1 => 0 < h < \frac{2}{\lambda_j}$
Using the derivation from 1.2: $0 < h < \frac{2}{\lambda_{max}} => 0 < h < \frac{1}{\max_{i,j}(\frac{k_{i-0.5,j}}{h_x^2} + \frac{k_{i,j-0.5}}{h_y^2} + \frac{k_{i+0.5,j}}{h_x^2} + \frac{k_{i,j+0.5}}{h_y^2})}$

However, this expression can be further simplified, given the problem information. Most importantly, it should be considered that k has constant value on four rectangles on the domain. Therefore, for large enough number of discretization points on the grid, the values of $k_{i-0.5,j}, k_{i,j-0.5}, k_{i+0.5,j}, k_{i,j+0.5}$ would fall in the same rectangle thus would have the same value and can be substituted by $k_{i,j}$. Thus the stability condition of the Forward-Euler becomes:
$0 < h < \frac{1}{2(\frac{\max_{i,j} k_{i,j}}{h_x^2} + \frac{\max_{i,j} k_{i,j}}{h_y^2})} => 0 < h < \frac{1}{2(\frac{1}{h_x^2} + \frac{1}{h_y^2})}$ $[\lambda_{max} \leq 4(\frac{1}{h_x^2} + \frac{1}{h_y^2})]$

2.2. Implementation and numerical experiments



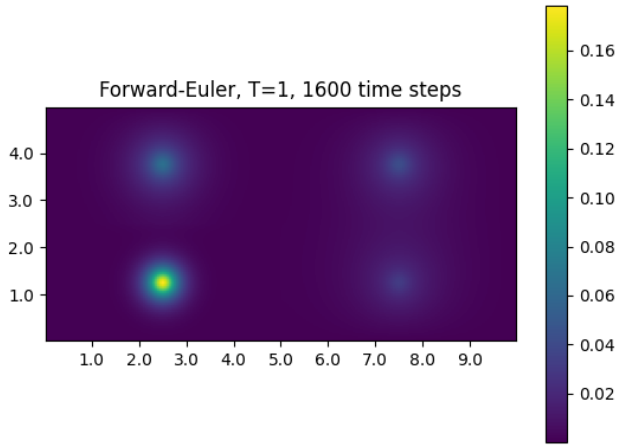
2.2.1.



2.2.2.
```
def solveFE(uStart, tStart, tEnd, Nt):
    h = (tEnd - tStart)/Nt
    A = create2DLFVM(k_grid)
    I = sp.eye((nx - 1) * (ny - 1), (nx - 1) * (ny - 1), format='csc')
    k = I - h * A
    for i in range(0, Nt):
        uStart = k.dot(uStart) + h * fLX
    return uStart
```

```
def solveTR(uStart, tStart, tEnd, Nt):
    h = (tEnd - tStart) / Nt
    A = create2DLFVM(k_grid)
    I = sp.eye((nx - 1) * (ny - 1), (nx - 1) * (ny - 1), format='csc')
    k = I - h / 2 * A
    k1 = I + h / 2 * A
    for i in range(0, Nt):
        uStart = la.spsolve(k1, (k.dot(uStart) + h * fLX))
                    .reshape((nx-1)*(ny-1), 1)
    return uStart
```

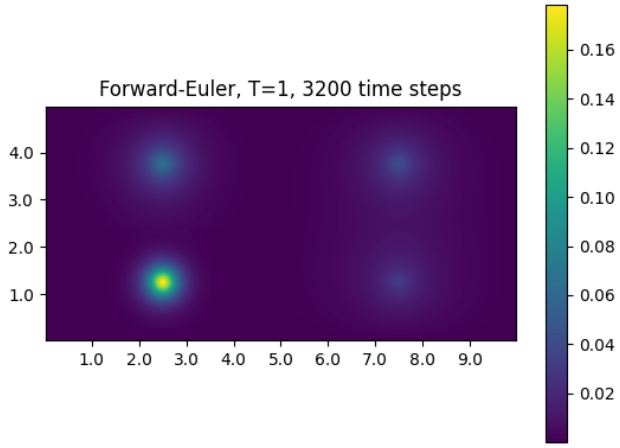
2.2.3. Forward–Euler Number of Time Steps: 1600

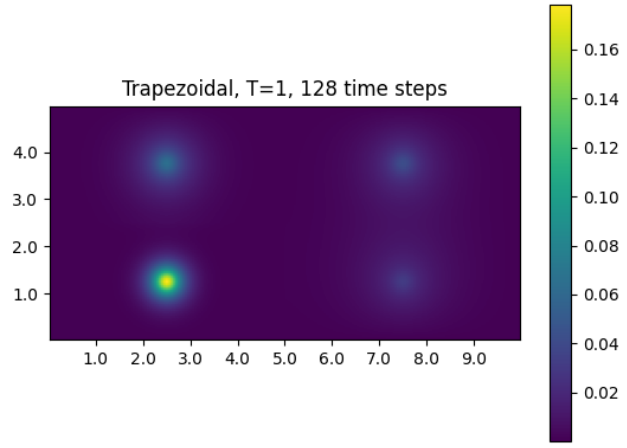


2.2.4.

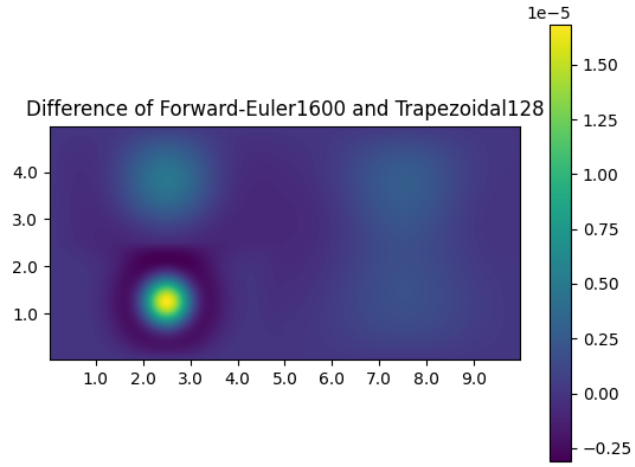
2.2.5. Forward–Euler 1600 Time Steps Execution Time: 0.3653688430786133 s

2.2.6. Forward–Euler 3200 Time Steps Execution Time: 0.6086587905883789 s

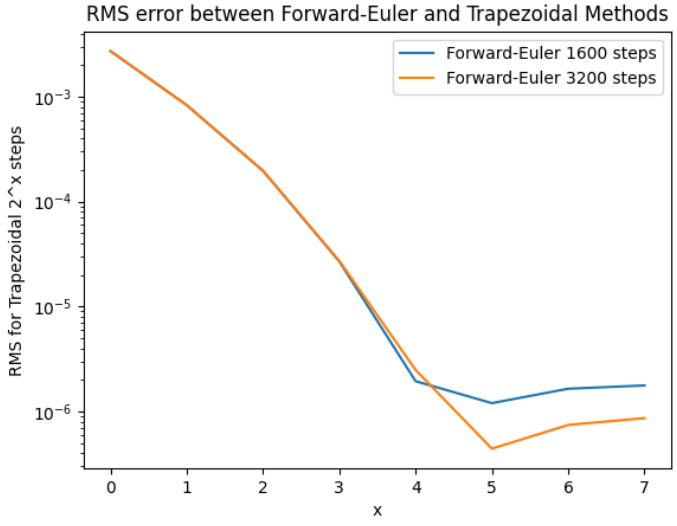




2.2.7.



2.2.8.



2.2.9.

2.2.10. A. Since Trapezoidal method has a global error of $O(h^2)$ and the time step size h is inversely proportional to number of time steps, the approximation with more time steps ($\tilde{u}_8(x, y)$) is more accurate then the one with less time steps ($\tilde{u}_6(x, y)$).

B. While Trapezoidal method has a global error of $O(h^2)$, the Forward-Euler one has a global error of $O(h)$, making it the generally less accurate method. The solution $u_1(x, y)$ is computed using Forward-Euler method with 1600 time steps, therefore has a global error of $O(\frac{1}{1600})$, while the solution $\tilde{u}_8(x, y)$ is computed using Trapezoidal method with $2^7 = 128$

time steps, therefore has a global error of $O(\frac{1}{128^2}) = O(\frac{1}{16384})$. This means that the $\tilde{u}_8(x,y)$ approximation is more accurate than the $u_1(x,y)$ one. Alternatively, it can be said that since Forward-Euler has a $O(h)$ global error and h is inversely proportional to the number of time steps, then the Forward-Euler with more time steps is the more accurate one. Therefore since the RMS error of $\tilde{u}_8(x,y)$ is smaller when compared to $u_2(x,y)$ than when compared to $u_1(x,y)$, then $\tilde{u}_8(x,y)$ is closer to $u_2(x,y)$ than to $u_1(x,y)$ and is therefore more accurate than $u_1(x,y)$.

C. The same error analysis methodology can be applied here, however, the solution $\tilde{u}_6(x,y)$ is computed using Trapezoidal method with $2^5 = 32$ time steps. This means that its global error $O(h^2) = O(\frac{1}{32^2}) = O(\frac{1}{1024})$. This is a bigger error than the $O(\frac{1}{1600})$ global error of the solution $u_1(x,y)$ computed using Forward-Euler method with 1600 time steps, therefore the $u_1(x,y)$ approximation is more accurate than the $\tilde{u}_6(x,y)$ one. However, big O is an upper-bound error analysis, therefore it is not suitable to analyse specific solutions. The RMS errors can once again be analyzed to conclude accuracy of specific approximations having in mind that Forward-Euler with more time steps produces more accurate approximation. The RMS of $\tilde{u}_6(x,y)$ when compared to $u_1(x,y)$ is higher than when compared to $u_2(x,y)$, therefore $\tilde{u}_6(x,y)$ is closer to $u_2(x,y)$ than $u_1(x,y)$, thus more accurate than $u_1(x,y)$.

3. Wave equation
$\frac{\partial^2 u}{\partial t^2} - \nabla \cdot (k \nabla u) = f, (x,y) \in \Omega, t \in (0,T]$
$u(x,y,t) = 0, (x,y) \in \partial\Omega$
$u(x,y,0) = 0, (x,y) \in \Omega$
$\frac{\partial u}{\partial t}(x,y,0) = 0, (x,y) \in \Omega$
$f(x,y,t) = sin(\omega t) e^{-\alpha(x-x_1)^2-\alpha(y-y_1)^2} + e^{-\alpha(x-x_2)^2-\alpha(y-y_2)^2} + e^{-\alpha(x-x_3)^2-\alpha(y-y_3)^2} + e^{-\alpha(x-x_4)^2-\alpha(y-y_4)^2}$

$$k(x,y) = \begin{cases} 0.1 \text{ if } x < L_x/2, y < L_y/2; \\ 0.4 \text{ if } x < L_x/2, y \geq L_y/2; \\ 0.7 \text{ if } x \geq L_x/2, y \geq L_y/2; \\ 1.0 \text{ if } x \geq L_x/2, y < L_y/2; \end{cases}$$

where

- $\overline{\Omega} = [0, L_x] \times [0, L_y]$ is rectangle with $L_x = 10, L_y = 5$
- $\alpha = 40$
- $(x_1, y_1) = (0.25L_x, 0.25L_y)$
- $(x_2, y_2) = (0.25L_x, 0.75L_y)$
- $(x_3, y_3) = (0.75L_x, 0.75L_y)$
- $(x_4, y_4) = (0.75L_x, 0.25L_y)$
- $\omega = 4\pi$

3.1. Spatial discretization and time integration
$\mathbf{u}'' = -A\mathbf{u} + f$

3.1.1. $\mathbf{u}^0 = \mathbf{0}$
$\mathbf{u}^1 = \mathbf{u}^0 + h_t \mathbf{u}^{0'} + h_t^2 \mathbf{u}^{0''} + O(h_t^3)$ (Taylor series expansion)
$\mathbf{u}^1 = \mathbf{u}^0 + h_t \mathbf{u}^{0'} + h_t^2(-A\mathbf{u}^0 + \mathbf{f}^0) + O(h_t^3)$ $\mathbf{u}^{0'} = \mathbf{0}$ $\mathbf{f}^0 = sin(0)\mathbf{g}(x,y) = \mathbf{0}$
$=> \mathbf{u}^1 = \mathbf{0} + O(h_t^3)$
$\mathbf{u}^{k+1} = 2\mathbf{u}^k - \mathbf{u}^{k-1} + h_t^2 \mathbf{u}^{k''} + O(h_t^2) = 2\mathbf{u}^k - \mathbf{u}^{k-1} + h_t^2(-A\mathbf{u}^k + \mathbf{f}^k) + O(h_t^2)$

3.1.2. General CFL Stability Condition: $\frac{h_t^2 \lambda_{max}}{4} \leq 1$
As derived in 2.1.1: $\lambda_{max} \leq 4(\frac{1}{h_x^2} + \frac{1}{h_y^2})$
$=> h_t^2(\frac{1}{h_x^2} + \frac{1}{h_y^2}) \leq 1$
$h_t \leq \sqrt{\frac{1}{\frac{1}{h_x^2} + \frac{1}{h_y^2}}}$

3.2. Implementation and numerical experiments

3.2.1.
```
def stepWave(u0Start, u1Start, tStart, dt):
    f = sourcefunc(x, y, tStart+dt).transpose()
    fLX = np.reshape(f, ((nx - 1) * (ny - 1), 1))
    return 2*u1Start - u0Start + dt**2*(-A.dot(u1Start)+fLX)
```

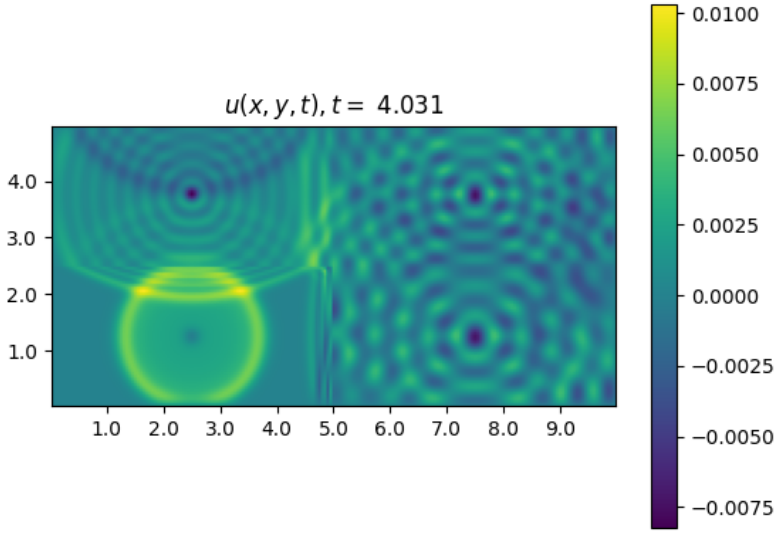
3.2.2. According to derived CFL bound, ht = 0.035, Nt = 114 **for** T = 4.

3.2.3.
```
plt.ion()
fig = plt.figure(3)
plt.clf()
t = 0
prev = np.zeros(((nx - 1) * (ny - 1), 1))
cur = np.copy(prev)
next = np.copy(cur)

# figure initialization
u3arr = np.reshape(next, (ny - 1, nx - 1, 1))
img = plt.imshow(u3arr, extent=(hx / 2, Lx - hx / 2,
                 Ly - hy / 2, hy / 2), interpolation='none')
plt.gca().invert_yaxis()
plt.colorbar(img, orientation='horizontal')
tlt = plt.title(r"$u(x,y,t), t=$" + str(np.round(t, 3)))

def animate(frame):
    global t, prev, cur, next
    t = frame * ht
    if frame == 0:
        prev = np.zeros(((nx - 1) * (ny - 1), 1))
        cur = np.copy(prev)
    next = np.copy(stepWave(prev, cur, t, ht))
    prev = np.copy(cur)
    cur = np.copy(next)
    img.set_array(np.reshape(next, (ny - 1, nx - 1, 1)))
    tlt.set_text(r"$u(x,y,t), t=$" + str(np.round(t + 2 * ht, 3)))
    img.set_clim(next.min(), next.max())
    return img

anim = animation.FuncAnimation(fig, animate, nt-1, repeat=False)
anim.save("animation.gif")
```



3.2.4.

3.2.5. The source function containing 4 distinct points can be seen as wave sources that disperse outwards in all directions. The sine component makes their value oscillating in time as opposed to being a continuous source, which is the reason why their motion represents waves. The coefficient function k in that case is the medium wave propagation coefficient. Given that there

are 4 distinct values of k over 4 distinct regions this can be physically achieved if there are 4 different mediums (materials). Since the boundary condition is 0, this means the waves are not propagating outside the system, but in fact get reflected back. At the beginning, it can be observed how the region with k=0.1 has the slowest spreading wave, while the region with k=1 has the fastest spreading one, as the k=1 medium is more permeable than the k=0.1 one. Furthermore, it can be observed how the wave bounces back from the boundary as the waves do not propagate outside the system. On the other hand, when waves propagate from one region to another, other physical phenomenons can be observed. To begin with, once the wave traverses into another region, a reflection wave is introduced into the source region. Another example is when a wave from a higher k regions enters a region with lower k, it can be observed how the wave's speed slows down and its shape changes from round to oval with its motion direction being the shorter axis. The opposite is also true, but in that case the wave's speed grows bigger and the wave assumes oval form with its motion direction being the longer axis. Last, but not least, inter-wave interactions can also be observed as there are spots where the waves are in-phase and magnify their strength and others where they are completely out of phase cancelling each other out.