Math 228B: Homework 1

1a): Show that the 9-point Laplacian has the truncation error

$$\frac{1}{12}h^2\nabla^4 u + \mathcal{O}(h^4).$$

First, note that the usual finite difference for the second derivative in 1D can be calculated easily:

$$f(\pm h) = f(0) \pm f'(0)h + \frac{1}{2}f''(0)h^2 \pm \frac{1}{6}f^{(3)}h^3 + \frac{1}{24}f^{(4)}(0)h^4 + \mathcal{O}(h^5)$$

So that (since the odd degree terms cancel)

$$\frac{f(h) - 2f(0) + f(-h)}{h^2} = f''(0) + \frac{1}{12}f^{(4)}(0)h^2 + \mathcal{O}(h^4)$$

It easily follows that in 2D, the 5 point Laplacian has

$$\nabla_5^2 f = (\partial_x^2 + \partial_y^2)f(0) + \frac{1}{12}h^2(\partial_x^4 + \partial_y^4)f(0) + \mathcal{O}(h^4)$$

by summing the usual finite differences in each direction.

Taking the hint, we subtract off the usual 5-point Laplacian from the 9 point to get the following stencil:

$$\frac{1}{6h^2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline\end{array}$$

Now, we want to show that this finite difference can be written as

$$\frac{1}{6}h^2\partial_x^2\partial_y^2 f(0) + \mathcal{O}(h^4).$$

We Taylor expand $f(\pm h, 0)$, $f(0, \pm h)$ and $f(\pm_1 h, \pm_2 h)$. Note that by the symmetry in x and y of the stencil (and because $\partial_x\partial_y = \partial_y\partial_x$) these first two are symmetric. Further, by the symmetry of our stencil, any odd power terms in h will cancel (due to the $\pm h$), so we will cheerfully neglect them. The vertical and horizontal terms are very similar to the usual finite difference:

$$\begin{aligned}-2(f(h,0) + f(-h,0) + f(0,h) + f(0,-h)) = \\ -8f(0) - 2(\partial_x^2 + \partial_y^2)f(0)h^2 - \frac{1}{3}(\partial_x^4 + \partial_y^4)f(0)h^4 + \mathcal{O}(h^6)\end{aligned}$$

The diagonals are a bit different. We use a multivariate Taylor expansion

$$f(\pm_1 h, \pm_2 h) = \sum_{i+j\leq 4} \frac{1}{i!j!}\partial_x^i \partial_y^j f(0)(\pm_1 h)^i (\pm_2 h)^j + \mathcal{O}(h^5).$$

Now consider a term where i (or j) is odd. Then $\pm_1$ appears in this term which means that when we add the plus and minus versions (with the same coefficients), they will cancel! In particular we are left with only even terms after summing:

$$\begin{aligned} &f(h,h) + f(-h,h) + f(h,-h) + f(-h,-h) = \\ &\qquad 4f(0) + 2(\partial_x^2 + \partial_y^2)f(0)h^2 + \frac{1}{3}(\partial_x^4 + \partial_y^4)f(0)h^4 + \partial_x^2\partial_y^2 f(0)h^4 + \mathcal{O}(h^6). \end{aligned}$$

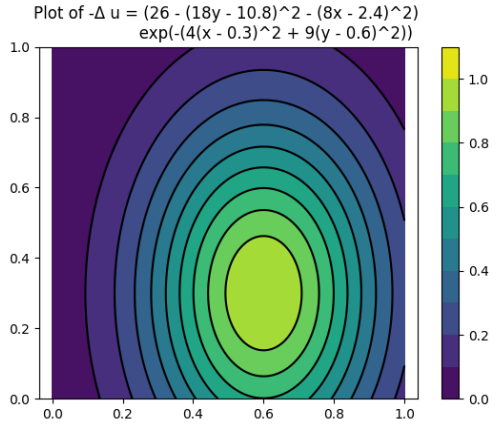
Adding these together as well as the center term $4f(0)$ gives (after dividing by our factor)

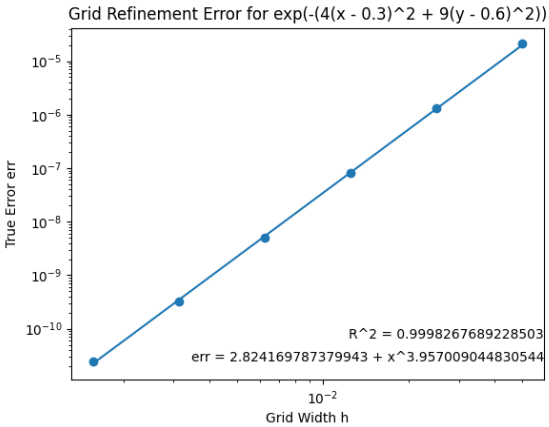
$$\frac{1}{6}h^2\partial_x^2\partial_y^2 f(0) + \mathcal{O}(h^4).$$

Finally, adding this to the 5 point Laplacian gives

$$\nabla_9^2 f = \nabla^2 f + \frac{1}{12}h^2(\partial_x^4 + 2\partial_x^2\partial_y^2 + p_y^4)f + \mathcal{O}(h^4) = \nabla^2 f + \frac{1}{12}h^2\nabla^4 f + \mathcal{O}(h^4)$$

1b): Modify the Julia function `assemblePoisson` to the 9 point Laplacian, and use the f_{ij} from LeVeque. Perfrom a grid refinement study to verify that fourth order accuracy is achieved.






```
"""
    Computes the discretization Au = b of -Δu = f on the unit square with boundary
    condition u = g to 4th order in h = 1/n (the grid width)
    Optionally setting useadhoclaplace = false, giving the true laplacian Lap_f of f
    uses that instead of using a 5 point scheme to estimate it.

    We achieve 4th order by using the 9 point Laplacian with error
    -Δ_9 u = -Δu - h^2/12 Δ^2 u + O(h^4)
    Since -Δu = f (for the true solution) we see that the scheme
    -Δ_9 u = f + h^2/12 * Δf has order O(h^4), even if we use an order h^2
    scheme to estimate Δf (which is done below)
"""
function assemblePoisson(n, f, g, useadhoclaplace =true, Lap_f = 0)
    h = 1.0/n
    N = (n+1)^2
    x = h* (0:n)
    y = x

    if useadhoclaplace
        Lap_f = adhocLaplace(f, h)
    else
        Lap_f = (x,y) -> h^2 * Lap_f(x,y)
    end

    umap = reshape(1:N, n+1, n+1)
    A = Tuple{Int64, Int64, Float64}[]  #matrix of the finite difference as (row, col,
```

```
                                        #row corresponds to output = forcing
                                        #col corresponds to input = u
    b = zeros(N)

    for j = 1:n+1
        for i = 1:n+1
            row = umap[i,j]
            if i == 1 || i == n+1 || j == 1 || j == n+1
                # On the boundary need to set u(x_i, y_j) = g(x_i, y_j)
                push!(A, (row, row, 1.0)) # 1 on the diagonal
                b[row] = g(x[i], y[j]) # value u should be
            else
                # In the interior use the 9 point stencil multiplied through by 6 h^2
                # Center
                push!(A, (row, row, 20.0))

                # Diagonals
                push!(A, (row, umap[i-1,j-1], -1.0))
                push!(A, (row, umap[i+1,j-1], -1.0))
                push!(A, (row, umap[i-1,j+1], -1.0))
                push!(A, (row, umap[i+1,j+1], -1.0))

                # Edges
                push!(A, (row, umap[i-1,j], -4.0))
                push!(A, (row, umap[i+1,j], -4.0))
                push!(A, (row, umap[i,j-1], -4.0))
                push!(A, (row, umap[i,j+1], -4.0))

                # Forcing
                # Note that we already multiplied by h^2 so
                # really the laplace f term has h^4
                b[row] = 6 * h^2 * f(x[i], y[j]) + h^2/2.0 * Lap_f(x[i],y[j])
            end
        end
    end

    A = sparse((x->x[1]).(A), (x->x[2]).(A), (x->x[3]).(A), N, N)

    return A, b, x, y
end

"""
    Computes the 5 point Laplacian of a function f with grid size h, to avoid loss of
```

```
    precision we avoid dividing by h^2
"""
function adhocLaplace(f, h)
    # for h sufficiently small we are losing some precision here by
    # dividing by h^2 when # we're gonna multiply it by h^2 later
    # anyway so we simply don't
    return (x,y) -> -4.0 * f(x,y) + f(x-h,y) + f(x+h, y) + f(x, y-h) + f(x,y+h)
end
```

2a): Introduce a mapping T to transform the original domain Ω (with coordinates x, y) to the unit square $\hat{\Omega}$ (with coordinates ξ, η). Derive the equations and the boundary conditions in the new domain.

We will use the mapping

$$T(x,y) = (\xi(x,y), \eta(x,y)) = \left(\frac{x}{B/2 + A/Hy}, y/H \right)$$

To derive the equations we note that

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} = \frac{1}{B/2 + A/Hy}\frac{\partial}{\partial \xi}$$

$$\frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} = -\frac{A}{H}\frac{x}{(B/2 + A/Hy)^2}\frac{\partial}{\partial \xi} + \frac{1}{H}\frac{\partial}{\partial \eta}$$

It will be helpfull to have these entirely in terms of the unit square:

$$\frac{\partial}{\partial x} = \frac{1}{B/2 + A\eta}\frac{\partial}{\partial \xi}$$

$$\frac{\partial}{\partial y} = -\frac{A}{H}\frac{\xi}{B/2 + A\eta}\frac{\partial}{\partial \xi} + \frac{1}{H}\frac{\partial}{\partial \eta}$$

And in matrix form (in the standard basis $a\frac{\partial}{\partial x} + b\frac{\partial}{\partial y}$ is equal to)

$$dT \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \frac{1}{B/2+A\eta} & -\frac{A}{H}\frac{\xi}{B/2+A\eta} \\ 0 & \frac{1}{H} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

To calculate the Laplacian in $\hat{\Omega}$, we will need to make sense of second partials:

$$\begin{aligned} -\left(\frac{\partial}{\partial x}\right)^2 &= -\left(\frac{1}{B/2 + A\eta}\frac{\partial}{\partial \xi}\right)\left(\frac{1}{B/2 + A\eta}\frac{\partial}{\partial \xi}\right) \\ &= -\frac{1}{(B/2 + A\eta)^2}\left(\frac{\partial}{\partial \xi}\right)^2 \end{aligned}$$

$$\begin{aligned} -\left(\frac{\partial}{\partial y}\right)^2 &= -\left(-\frac{A}{H}\frac{\xi}{B/2 + A\eta}\frac{\partial}{\partial \xi} + \frac{1}{H}\frac{\partial}{\partial \eta}\right)\left(-\frac{A}{H}\frac{\xi}{B/2 + A\eta}\frac{\partial}{\partial \xi} + \frac{1}{H}\frac{\partial}{\partial \eta}\right) \\ &= -\left(\frac{A}{H}\frac{\xi}{B/2 + A\eta}\right)^2\left(\frac{\partial}{\partial \xi}\right)^2 + 2\frac{A}{H^2}\frac{\xi}{B/2 + A\eta}\frac{\partial}{\partial \xi}\frac{\partial}{\partial \eta} - \frac{1}{H^2}\left(\frac{\partial}{\partial \eta}\right)^2 \\ &\quad - \xi\left(\frac{A}{H}\frac{1}{B/2 + A\eta}\right)^2\frac{\partial}{\partial \xi} - \left(\frac{A^2}{H^2}\frac{\xi}{(B/2 + A\eta)^2}\right)\frac{\partial}{\partial \xi} \end{aligned}$$

It follows that the Laplacian in the square is the sum of these.

For computing the flowrate it will be useful to have an expression for $dx \wedge dy$ in the new coordinates. For this we use the determinant

$$d\xi \wedge d\eta = \det dT dx \wedge dy = \frac{1}{H}\frac{1}{B/2 + A\eta} dx \wedge dy$$

So that

$$\int_\Omega u(x,y)dxdy = \int_{\hat{\Omega}} u(\xi,\eta)H(B/2 + A\eta)d\xi d\eta$$

2b): In the transformed domain, write down second-order accurate finite-difference schemes for the discretization of all the derivative terms in the interior with Neumann boundary conditions on the top and left (use the left condition for the upper left corner) and Dirichlet for the other two (use these for the remaining corners). Also show how to evaluate the integral in the approximate flowrate $\hat{Q}$ numerically in the computational domain.

In the interior we approximate the usual way:

$$\partial_\xi u(\xi,\eta) = \frac{1}{2h}\big(u(\xi+h,\eta) - u(\xi-h,\eta)\big) + \mathcal{O}(h^2)$$

$$\partial_\xi^2 u(\xi,\eta) = \frac{1}{h^2}\big(u(\xi+h,\eta) - 2u(\xi,\eta) + u(\xi-h,\eta)\big) + \mathcal{O}(h^2)$$

$$\partial_\xi\partial_\eta u(\xi,\eta) = \frac{1}{4h^2}\big(u(\xi+h,\eta+h) - u(\xi-h,\eta+h) - u(\xi+h,\eta-h) + u(\xi-h,\eta-h)\big) + \mathcal{O}(h^2)$$

And similarly for η. Below we use the notation $\xi_i = h(i-1)$, $\eta_j = h(j-1)$, (I'm going to catch the indexing by 1 issue here) and $u_{ij} = u(\xi_i,\eta_j)$. Then, carefully counting we have that, $-\nabla^2$ in the computational coordinates becomes

$$\begin{aligned}
1 = Du_{ij} = {} & \frac{2}{h^2}\left(\frac{1}{(B/2+A\eta_j)^2} + \left(\frac{A\xi_i}{H(B/2+A\eta_j)}\right)^2 + \frac{1}{H^2}\right)u_{ij} \\
& + \frac{1}{4h^2}2\frac{A}{H^2}\frac{\xi_i}{B/2+A\eta_j}\left(u_{(i+1)(j+1)} - u_{(i+1)(j-1)} - u_{(i-1)(j+1)} + u_{(i-1)(j-1)}\right) \\
& + \left(\frac{-1}{h^2}\frac{1}{(B/2+A\eta_j)^2} + \frac{-1}{h^2}\left(\frac{A}{H}\frac{\xi_i}{B/2+A\eta_j}\right)^2 + \frac{-1}{2h}2\xi_i\left(\frac{A}{H}\frac{1}{B/2+A\eta_j}\right)^2\right)u_{(i+1)j} \\
& + \left(\frac{-1}{h^2}\frac{1}{(B/2+A\eta_j)^2} + \frac{-1}{h^2}\left(\frac{A}{H}\frac{\xi_i}{B/2+A\eta_j}\right)^2 + \frac{1}{2h}2\xi_i\left(\frac{A}{H}\frac{1}{B/2+A\eta_j}\right)^2\right)u_{(i-1)j} \\
& + \frac{-1}{h^2}\frac{1}{H^2}\big(u_{i(j+1)} + u_{i(j-1)}\big)
\end{aligned}$$

Simplifying this some gives

$$\begin{aligned}
1 = Du_{ij} = {} & \frac{2}{h^2}\left(\frac{1+A^2/H^2\xi_i^2}{(B/2+A\eta_j)^2} + \frac{1}{H^2}\right)u_{ij} \\
& + \frac{1}{2h^2}\frac{A}{H^2}\frac{\xi_i}{B/2+A\eta_j}\left(u_{(i+1)(j+1)} - u_{(i+1)(j-1)} - u_{(i-1)(j+1)} + u_{(i-1)(j-1)}\right) \\
& + \left(\frac{-1}{h^2}\frac{1+A^2/H^2\xi_i^2}{(B/2+A\eta_j)^2} + \frac{-1}{h}\frac{A^2/H^2\xi_i}{(B/2+A\eta_j)^2}\right)u_{(i+1)j} \\
& + \left(\frac{-1}{h^2}\frac{1+A^2/H^2\xi_i^2}{(B/2+A\eta_j)^2} + \frac{1}{h}\frac{A^2/H^2\xi_i}{(B/2+A\eta_j)^2}\right)u_{(i-1)j} \\
& + \frac{-1}{h^2}\frac{1}{H^2}\big(u_{i(j+1)} + u_{i(j-1)}\big)
\end{aligned}$$

For the left boundary condition (excluding the bottom left corner) we use a right sided derivative. Note that $\partial_\xi u = 0$ is equivalent to $\partial_x u = 0$ so we don't need to compute the coefficient.

$$0 = \frac{1}{2h}(-3u_{ij} + 4u_{(i+1)j} - 1u_{(i+2)j})$$

For the top boundary (excluding the corner points) we use a centered derivative for ∂_ξ and a left sided derivative for ∂_η

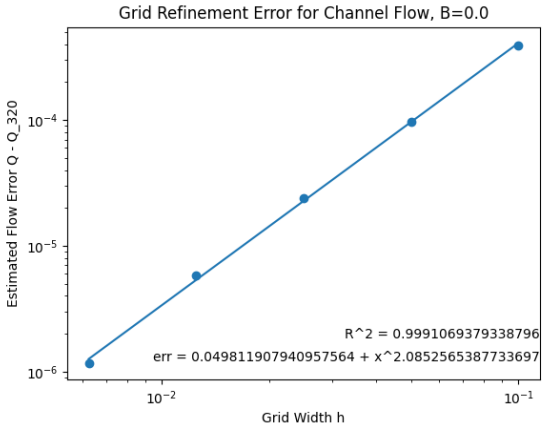
$$0 = -\frac{A/H\xi_i}{B/2 + A\eta_j}\frac{1}{2h}(u_{(i+1)j} - u_{(i-1)j}) + \frac{1}{H}\frac{1}{2h}(3u_{ij} - 4u_{i(j-1)} + 1u_{i(j-2)})$$

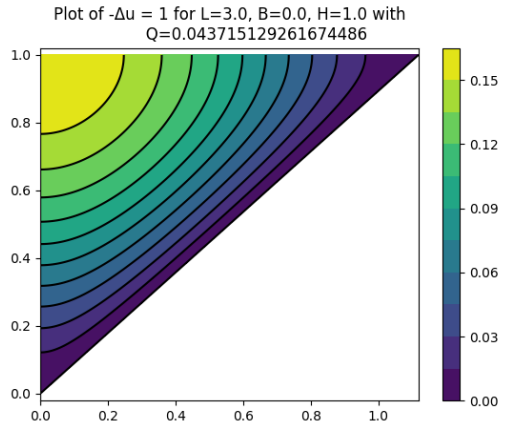
To calculate the flowrate we simply use the trapezoidal sum on $f(\xi, \eta) = H(B/2 + A\eta)u(\xi, \eta)$ which will work out to be

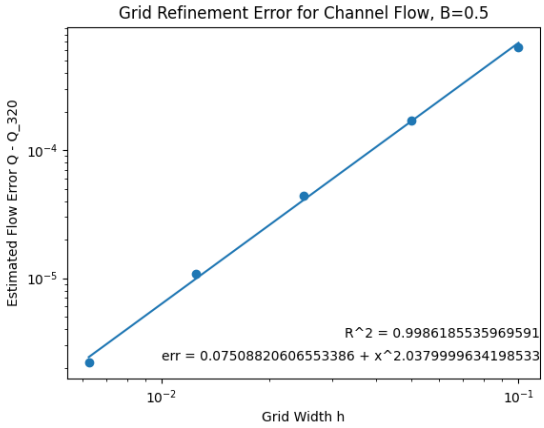
$$\frac{h^2}{4}\left(f(0,0) + \cdots + f(1,1) + 2\sum f(\xi_i, 0) + \cdots + 2\sum f(1, \eta_j) + 4\sum f(\xi_i, \eta_j)\right)$$

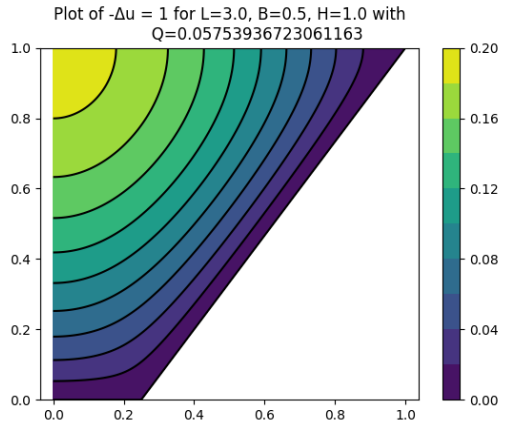
2c): Make convergence plots and calculate convergence rates for the error in the output Q.

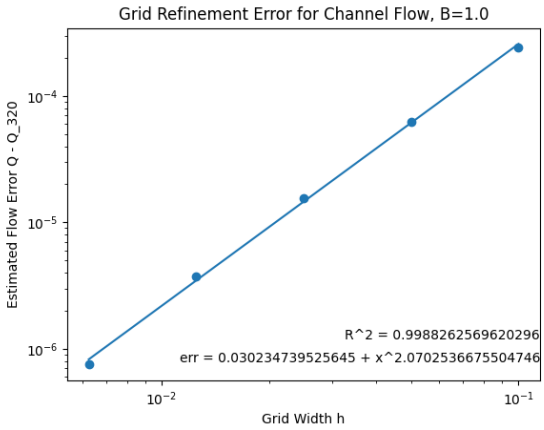
Here are the plots, with linear regressions:

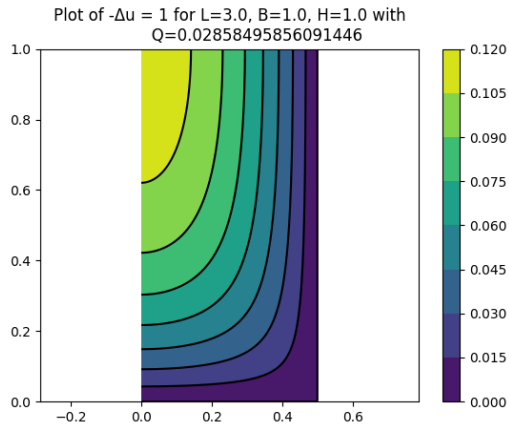












And the code

```
function channelflow(L,B,H,n)
```

```
h = 1.0/n
N = (n+1)^2
xi = h * (0:n)
eta = xi
A = sqrt((L - B)^2/4.0 - H^2)

umap = reshape(1:N, n+1, n+1)
M = Tuple{Int64, Int64, Float64}[]  #matrix of the finite difference as (row, col,
                                    #row corresponds to output = forcing
                                    #col corresponds to input = u
b = zeros(N)
for j = 1:n+1
    for i = 1:n+1
        row = umap[i,j]
        denom = 1.0/(B/2.0 + A*eta[j])
        if i == n+1 || j == 1
            # Dirichlet boundary
            push!(M, (row, row, 1.0)) # 1 on the diagonal
            b[row] = 0.0 # value u should be
        elseif i == 1
            # left boundary (before the upper boundary so the corner will live her
            push!(M, (row, umap[i,j], -3.0))
            push!(M, (row, umap[i+1,j], 4.0))
            push!(M, (row, umap[i+2,j], -1.0))
            b[row] = 0.0 #multiply through by 2h
        elseif j == n+1
            # upper boundary (multiply through by 2hH)
            # xi part
            push!(M, (row, umap[i+1,j], -A*xi[i]*denom))
            push!(M, (row, umap[i-1,j], A*xi[i]*denom))
            # eta part
            push!(M, (row, umap[i,j], 3.0))
            push!(M, (row, umap[i,j-1], -4.0))
            push!(M, (row, umap[i,j-2], 1.0))
            b[row] = 0.0
        else
            # In the interior use the crazy mess we calculated
            # Being a bit more verbose to try to midigate errors
            # Center - contributions from 2nd derivatives
            push!(M, (row, umap[i,j], 2.0/h^2 *((1+A^2/H^2 *xi[i]^2)*denom^2 + 1/H

            # Diagonals - contributions from mixed derivatives
            push!(M, (row, umap[i+1,j+1], 1.0/h^2/2.0 *A/H^2 *xi[i]*denom))
```

```
                push!(M, (row, umap[i+1,j-1], -1.0/h^2/2.0 *A/H^2 *xi[i]*denom))
                push!(M, (row, umap[i-1,j+1], -1.0/h^2/2.0 *A/H^2 *xi[i]*denom))
                push!(M, (row, umap[i-1,j-1], 1.0/h^2/2.0 *A/H^2 *xi[i]*denom))

                # Horizontal edges - xi derivatives
                push!(M, (row, umap[i+1,j], -1.0/h^2*(1.0 + A^2/H^2 * xi[i]^2) * denom
                          - 1.0/h *A^2/H^2 * xi[i] * denom^2))
                push!(M, (row, umap[i-1,j], -1.0/h^2*(1.0 + A^2/H^2 * xi[i]^2) * denom
                          + 1.0/h *A^2/H^2 * xi[i] * denom^2))

                # Vertical edges - eta derivatives
                push!(M, (row, umap[i,j+1], -1.0/h^2/H^2))
                push!(M, (row, umap[i,j-1], -1.0/h^2/H^2))

                # Forcing
                b[row] = 1.0
            end
        end
    end

    M = sparse((x->x[1]).(M), (x->x[2]).(M), (x->x[3]).(M), N, N)

    # The solution reshaped into a square matrix (note we don't change anything here
    # because functions don't transform under change of variables - we just reinterpre
    # grid)
    u = reshape(M \ b, length(xi), length(eta))
    # To get the original coordinates we use the inverse map - note that we can't have
    # x and y independent - we need them to be matrices of the same size as u
    x = [[i*(B/2.0 + A*j) for j in eta] for i in xi]
    y = [[H*j for j in eta] for i in xi]

    f(i,j) = H*(B/2 + A*eta[j])*u[i,j]

    Q = h^2/4.0 * (f(1,1) + f(1,n+1) + f(n+1,1) + f(n+1,n+1)
                   + 2*sum(f(i,1) for i in (2:n)) + 2*sum(f(i,n+1) for i in (2:n))
                   + 2*sum(f(1,j) for j in (2:n)) + 2*sum(f(n+1,j) for j in (2:n))
                   + 4*sum(f(i,j) for i in (2:n) for j in (2:n)))

    return Q, x, y, u
end
```

3a): Show that the method

$$U_j^{n+1} = U_j^n + \frac{k\kappa}{2h^2}[U_{j-1}^n - 2U_j^n + U_{j+1}^n U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}] - k\gamma[(1-\theta)U_j^n + \theta U_j^{n+1}]$$

for the PDE

$$u_t = \kappa u_{xx} - \gamma u$$

(with $\kappa, \gamma > 0$) is $\mathcal{O}(k^p + h^2)$ accurate where $p = 2$ if $\theta = 1/2$ and $p = 1$ otherwise.

We compute the local truncation error by plugging in a true solution to the scheme and Taylor expanding. For the sake of clarity let's deal with one thing at a time:

$$\frac{u(x,t+k) - u(x,t)}{k} = u_t(x,t) + \frac{1}{2}ku_{tt}(x,t) + \mathcal{O}(k^2),$$

$$\begin{aligned}
&-\frac{\kappa}{2h^2}[u(x+h,t) - 2u(x,t) + u(x-h,t) + u(x+h,t+k) - 2u(x,t+k) + u(x-h,t+k)] \\
&\quad = -\frac{\kappa}{2}u_{xx}(x,t) - \frac{\kappa}{2}u_{xx}(x,t+k) + \mathcal{O}(h^2) \\
&\quad = -\kappa u_{xx}(x,t) - \frac{\kappa}{2}ku_{xxt}(x,t) + \mathcal{O}(h^2 + k^2)
\end{aligned}$$

and

$$\gamma[(1-\theta)u(x,t) + \theta u(x,t+k)] = \gamma[u(x,t) + \theta k u_t(x,t)] + \mathcal{O}(k^2)$$

Summing these gives the truncation error (I will drop the (x,t) since we've written it all at one location)

$$\begin{aligned}
\tau &= (u_t - \kappa u_{xx} + \gamma u) + \frac{1}{2}k\frac{\partial}{\partial t}(u_t - \kappa u_{xx} + 2\theta\gamma u) + \mathcal{O}(k^2 + h^2) \\
&= 0 + \frac{1}{2}k(2\theta - 1)u_t + \mathcal{O}(k^2 + h^2)
\end{aligned}$$

Thus, since we expect u_t to not be uniformly 0, we can see that the truncation error is order $k^2 + h^2$ if $\theta = 1/2$ and $k + h^2$ otherwise.