Math 586 Homework 5
Due June 2
By Marvyn Bailly (Github: MarvynB)

Problem 1 *Consider solving*

$$\begin{cases} u_t + u_{xxx} = 0, \quad -1 < x < 1 \\ u(x,0) = \eta(x), \\ u(-1,t) = u(1,t), \\ u_x(-1,t) = u_x(1,t), \\ u_{xx}(-1,t) = u_{xx}(1,t). \end{cases}$$

This is the linear KdV (Airy) equation with periodic boundary conditions.

- *Use a second-order accurate centered difference and the trapezoid method as a time-stepper. Can you see dispersive quantization? Use* $\eta(x) = 1$ *if* $-1/2 < x < 1/2$ *and* $\eta(x) = 0$ *otherwise.*
- *Prove that the method is Lax-Richtmyer stable. Discuss whether or not it is convergent with this initial condition.*

Solution.
Consider the Airy equation with periodic boundary conditions given by

$$\begin{cases} u_t + u_{xxx} = 0, \quad -1 < x < 1 \\ u(x,0) = \eta(x), \\ u(-1,t) = u(1,t), \\ u_x(-1,t) = u_x(1,t), \\ u_{xx}(-1,t) = u_{xx}(1,t). \end{cases}$$

(a) We first wish to solve the problem using a second-order accurate centered difference and the trapezoid method as a time stepper. We begin by discretizing in space using a centered finite difference of the form

$$U_j'(t) = -\left(\frac{-U_{j-2}(t) + 2U_{j-1}(t) - 2U_{j+1}(t) + U_{j+2}(t)}{2h^3}\right)[U_j].$$

Then applying the periodic boundary conditions we have that

$$\begin{aligned} U_0 &= U_{m+1} \\ U_{-1} &= U_m \\ U_{-2} &= U_{m-1}. \end{aligned}$$

Then at $j = 0$ we have

$$U_0' = \frac{-U_{-2} + 2U_{-1} - 2U_1 + U_2}{2h^3} = \frac{-U_{m-1} + 2U_m - 2U_1 + U_2}{2h^3},$$

and at $j = 1$ we have

$$U_1' = \frac{-U_{-1} + 2U_0 - 2U_2 + U_3}{2h^3} = \frac{-U_m + 2U_{m+1} - 2U_2 + U_3}{2h^3},$$

and at $j = 2$ we have

$$U_2' = \frac{-U_0 + 2U_1 - 2U_3 + U_4}{2h^3} = \frac{-U_{m+1} + 2U_1 - 2U_3 + U_4}{2h^3},$$

and continuing this process we find

$$[U_j'] = -\frac{1}{2h^3} D_3 [U_j],$$

where

$$D_3 = \begin{bmatrix} 0 & -2 & 1 & & & -1 & 2 \\ 2 & 0 & -2 & 1 & & & -1 \\ -1 & 2 & 0 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 2 & 0 & -2 & 1 \\ 1 & & & -1 & 2 & 0 & -2 \\ -2 & 1 & & & -1 & 2 & 0 \end{bmatrix}$$

Then applying the trapezoid method to time step yields

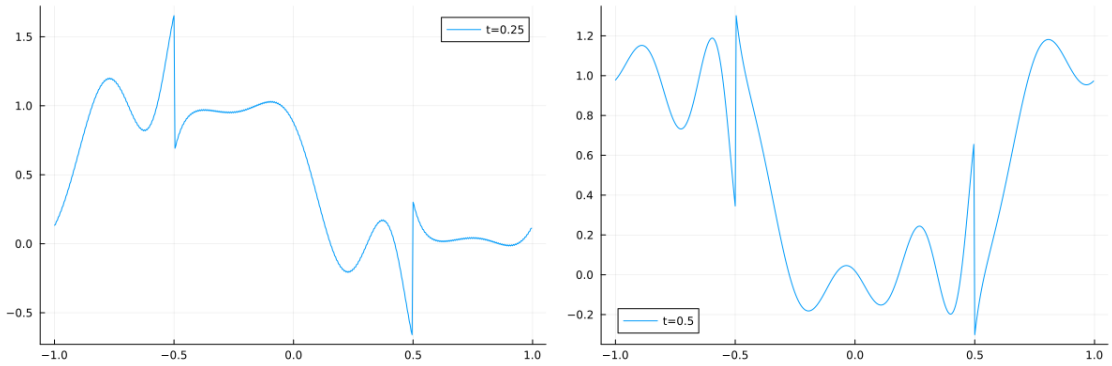
$$\left(I + \frac{k}{4h^3} D_3\right)[U_j^{n+1}] = \left(I - \frac{k}{4h^3} D_3\right)[U_j^n].$$

We implement this method in Julia as follows:

```
function solve_airy(m,inti,stop)
    L = 1
    h = 2/(1+m)
    k = 0.01
    xs = -1:h:1-h
    h = L*h
    xs = L*xs

    D3 = diagm(1 => fill(-2,m), -1 => fill(2,m), -2 => fill(-1,m-1), 2 =>
        fill(1,m-1)) |> sparse
    D3[1,m] = -1
    D3[1,m+1] = 2
```

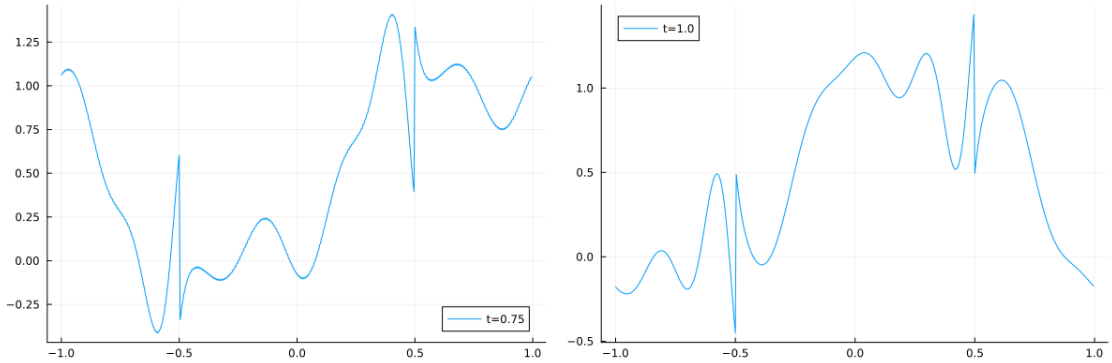
```
    D3[2,m+1] = -1
    D3[m+1,1] = -2
    D3[m+1,2] = 1
    D3[m,1] = 1

    U = inti.(xs)
    for t=k:k:stop
        U =  (I + k/(4*h^3)*D3) \ ((I-k/(4*h^3)*D3)*U)
    end
    return xs, U
end
```

Then using $h = 0.004$ and $k = 0.01$ and plotting the solution at rational numbers of the period length such as $t \in [0.25, 0.50, 0.75, 1.0]$ we get plots seen in Figure 1. Recalling that the initial condition is a step function, we see no signs of dispersive quantization in this system since the solution does not appear to be a composition of step functions as we would expect when dispersive quantization is present.



(a) Solution at $t = 0.25$.

(b) Solution at $t = 0.50$.



(c) Solution at $t = 0.75$.

(d) Solution at $t = 1.0$.

Figure 1: Solutions at $t = [0.25, 0.50, 0.75, 1.0]$ to the linear KdV equation using a second-order accurate method.

(b) To apply Lax-Richtmyer stability analysis, we require $(I + \frac{k}{4h^3}D_3)$ to be invertible. Since D_3 is scew-symmetric, so is $\frac{k}{4h^3}D_3$. We know that skew-symmetric matrices have purely imaginary eigenvalues and thus we know that $I + 4h^3 D_3$ must have nonzero eigenvalues and thus is invertible. So we now consider

$$[U_J^{n+1}] = \left(I + \frac{k}{4h^3}D_3\right)^{-1}\left(I - \frac{k}{4h^3}D_3\right)[U_j^n] = B[U_j^n],$$

and for Lax-Richtmyer stability we require the eigenvalues of B to be less than or equal to 1 with strict inequality for repeated eigenvalues. Now letting $\lambda = ai$ be an arbitrary eigenvalue $\frac{k}{4h^3}D_3$ and noting that the eigenvalues of I are 1, then

$$|\text{eval}(B)| = |(1+\lambda)^{-1}(1-\lambda)| = |(1+ai)^{-1}(1-ai)| = |1+ai|^{-1}|1-ai| = 1.$$

Next, we will show that the eigenvalues of D_3 are unique. Since D_3 encodes a periodic operator then we look for eigenvectors that come from period grid functions of the general form

$$[V_j^{(l)}] = [e^{2\pi ijlh}], \quad j = 0, 1, 2, \ldots, m.$$

We then compute

$$\begin{aligned}
D_3[V_j^{(l)}] &= \begin{bmatrix} 0 & -2 & 1 & & & -1 & 2 \\ 2 & 0 & -2 & 1 & & & -1 \\ -1 & 2 & 0 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 2 & 0 & -2 & 1 \\ 1 & & & -1 & 2 & 0 & -2 \\ -2 & 1 & & & -1 & 2 & 0 \end{bmatrix} [V_j^{(l)}] \\
&= \begin{bmatrix} -2V_1^{(l)} + V_2^{(l)} - V_{m-1}^{(l)} + 2V_m^{(l)} \\ 2V_0^{(l)} - 2V_2^{(l)} + V_3^{(l)} - V_m^{(l)} \\ \vdots \\ -V_{j-2}^{(l)} + 2V_{j-1}^{(l)} - 2V_{j+1}^{(l)} + V_{j+2}^{(l)} \\ \vdots \\ V_1^{(l)} - V_{m-3}^{(l)} + 2V_{m-2}^{(l)} - 2V_m^{(l)} \\ -2V_1^{(l)} + V_2^{(l)} - V_{m-2}^{(l)} + 2V_{m-1}^{(l)} \end{bmatrix} \\
&= \begin{bmatrix} -2e^{2\pi ilh} + e^{2\pi i2lh} - e^{2\pi i(m-1)lh} + 2e^{2\pi imlh} \\ 2e^{2\pi i(0)lh} - 2e^{2\pi i2lh} + e^{2\pi i3lh} - e^{2\pi imlh} \\ \vdots \\ -e^{2\pi i(j-2)lh} + 2e^{2\pi i(j-1)lh} - 2e^{2\pi i(j+1)lh} + e^{2\pi i(j+2)lh} \\ \vdots \\ e^{2\pi ilh} - e^{2\pi il(m-3)h} + 2e^{2\pi il(m-2)h} - 2e^{2\pi ilmh} \\ -2e^{2\pi ilh} + e^{2\pi i2lh} - e^{2\pi i(m-2)lh} + 2e^{2\pi i(m-1)lh} \end{bmatrix} \\
&= \begin{bmatrix} e^{2\pi i(0)lh}\left(2e^{2\pi i(-1)lh} - 2e^{2\pi i(1)lh} + e^{2\pi i(2)lh} - e^{2\pi i(-2)lh}\right) \\ e^{2\pi i(1)lh}\left(2e^{2\pi i(-1)lh} - 2e^{2\pi i(1)lh} + e^{2\pi i(2)lh} - e^{2\pi i(-2)lh}\right) \\ \vdots \\ e^{2\pi i(j)lh}\left(2e^{2\pi i(-1)lh} - 2e^{2\pi i(1)lh} + e^{2\pi i(2)lh} - e^{2\pi i(-2)lh}\right) \\ \vdots \\ e^{2\pi i(m-1)}\left(2e^{2\pi il(-1)h} - 2e^{2\pi il(1)h} + e^{2\pi i(2)lh} - e^{2\pi il(-2)h}\right) \\ e^{2\pi i(m)}\left(2e^{2\pi i(-1)lh} - 2e^{2\pi i(1)lh} + e^{2\pi i(2)lh} - e^{2\pi i(-2)lh}\right) \end{bmatrix} \\
&= (4i\sin(-2\pi lh) + 2i\sin(4\pi lh))[V_j^{(l)}].
\end{aligned}$$

Since $h = \frac{2}{m+1}$, the eigenvalues of D_3 are unique. Therefore B has unique eigenvalues satisfiying the Lax-Richtmyer stability condition. For the given initial condition, the

method is not convergent since it is not consistent. We know that the method is not consistent from the part a were we did not observe dispersive quantization. From class, we expect this method to have quantizations and thus the method is not consistent.

□

Problem 2 *Consider solving*

$$\begin{cases} u_t + 3(u^2)_x + u_{xxx} = 0, \quad -L < x < L, \\ u(x,0) = \eta(x), \\ u(-L,t) = u(L,t), \\ u_x(-L,t) = u_x(L,t), \\ u_{xx}(-L,t) = u_{xx}(L,t). \end{cases}$$

This is the KdV equation with periodic boundary conditions.

- *Use a second-order accurate centered difference and the trapezoid method as a time-stepper to solve this problem with*

$$\eta(x) = 4\text{sech}(x)^2, \quad L = 10.$$

  Note that you will need to implement Newton's method for this and use it at each time step.

- *You need not prove this, give some rationale as to why you might hope this method is Lax-Richtmyer stable.*

Solution.
Consider the KdV equation with periodic boundary conditions given by

$$\begin{cases} u_t + 3(u^2)_x + u_{xxx} = 0, \quad -L < x < L, \\ u(x,0) = \eta(x), \\ u(-L,t) = u(L,t), \\ u_x(-L,t) = u_x(L,t), \\ u_{xx}(-L,t) = u_{xx}(L,t). \end{cases}$$

(a) Consider the initial condition

$$\eta(x) = 4\text{sech}(x)^2, \ L = 10.$$

We wish to derive a second-order accurate centered difference with the trapezoid method. We begin by using two centered differences, note that we found D_3 in Question 1 and used the same process to derive D_1 and its coefficient, to discretize in space to get

$$[U_j]' = -6[U_j] \circ \left[\frac{1}{2h} D_1[U_j]\right] - \frac{1}{2h^3} D_3[U_j],$$

where $\circ$ represents the Hadamard product and

$$D_1 = \begin{bmatrix} 0 & 1 & & & -1 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1 \\ 1 & & & -1 & 0 \end{bmatrix},$$

and

$$D_3 = \begin{bmatrix} 0 & -2 & 1 & & & -1 & 2 \\ 2 & 0 & -2 & 1 & & & -1 \\ -1 & 2 & 0 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 2 & 0 & -2 & 1 \\ 1 & & & -1 & 2 & 0 & -2 \\ -2 & 1 & & & -1 & 2 & 0 \end{bmatrix}.$$

Next, we discretize in time using the trapezoid rule to get

$$\begin{aligned} \frac{[U_j^{n+1}] - [U_j^n]}{k} &= \frac{1}{2}\left(-6[U_j^n] \circ \left[\frac{1}{2h} D_1[U_j^n]\right] - \frac{1}{2h^3} D_3[U_j^n]\right) \\ &\quad + \frac{1}{2}\left(-6[U_j^{n+1}] \circ \left[\frac{1}{2h} D_1[U_j^{n+1}]\right] - \frac{1}{2h^3} D_3[U_j^{n+1}]\right), \end{aligned}$$

which we can rewrite to get $F([U_j^{n+1}]) = 0$ as

$$\begin{aligned} F([U_j^{n+1}]) &= \left([U_j^{n+1}] + \frac{3k}{2h}[U_j^{n+1}] \circ (D_1[U_j^{n+1}]) + \frac{k}{4h^3} D_3[U_j^{n+1}]\right) \\ &\quad + \left(-[U_j^n] + \frac{3k}{2h}[U_j^n] \circ (D_1[U_j^n]) + \frac{k}{4h^3} D_3[U_j^n]\right). \end{aligned}$$

Next, we will implement Newton's method to solve for U_j^{n+1}. For Newton's method, we require an initial guess and the Jacobian. The Jacobian is given by

$$DF(U_j^n) = I + \frac{k}{4h^3} D_3 + \frac{3k}{2h}\big(\mathrm{diag}(D_1 U_j^n) + \mathrm{diag}(U_j^n) D_1\big).$$

For the initial guess, we will solve the linear case

$$\begin{cases} u_t + u_{xxx} = 0, \quad -1 < x < 1 \\ u(x,0) = \eta(x), \\ u(-1,t) = u(1,t), \\ u_x(-1,t) = u_x(1,t), \\ u_{xx}(-1,t) = u_{xx}(1,t). \end{cases}$$

Now we can implement our second-order method and use Newton's method at each step to approximate the solution to the KdV equation. We implement Newton's method in Julia with the following

```
function Newton(U,U0,m,F,DF)
    count = 0
    New_U = U0
```

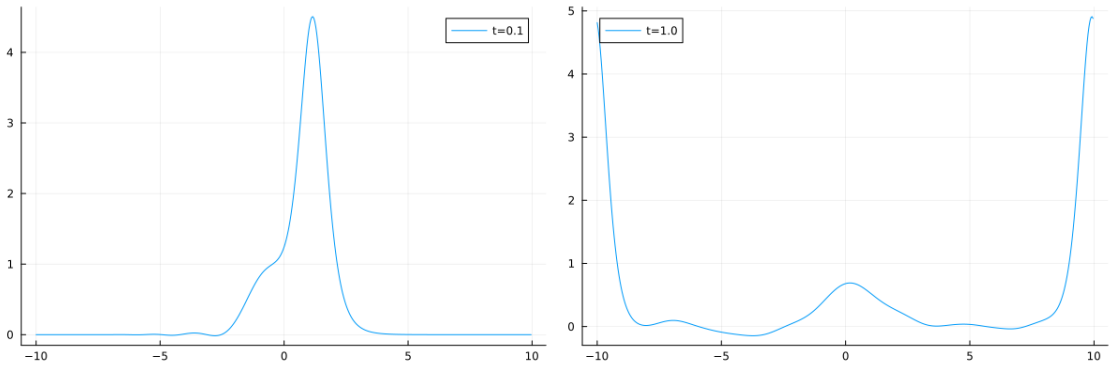
```
    update = fill(100,m+1)
    while maximum(abs.(update)) > 1e-10
        count += 1
        if count == 300
            @printf("No convergence")
        break
        end
        update = DF(New_U)\F(New_U,U)
        New_U = New_U - update
    end
    return New_U
end
```

Using the Newton method, we implement the method to solve KdV as

```
function solveKdV(m,init,stop)
    L = 10
    h = 2/(1+m)
    k = 0.01
    xs = -1:h:1-h
    h = L*h
    xs = L*xs

    D3 = diagm(1 => fill(-2,m), -1 => fill(2,m), -2 => fill(-1,m-1), 2 =>
        fill(1,m-1)) |> sparse
    D3[1,m] = -1
    D3[1,m+1] = 2
    D3[2,m+1] = -1
    D3[m+1,1] = -2
    D3[m+1,2] = 1
    D3[m,1] = 1

    D1 = diagm(1 => fill(1,m),-1 => fill(-1,m)) |> sparse
    D1[1,m+1] = -1
    D1[m+1,1] = 1

    #define things
    f = (y,x) -> (y + (3*k)/(2*h)*y .* (D1*y) + k/(4*h^3)*D3*y) + (-x +
        (3*k)/(2*h)*x .* (D1*x) + k/(4*h^3)*D3*x)
    Df = x -> I + k/(4*h^3)*D3 + (3*k)/(2*h)*(Diagonal(D1 * x) + Diagonal
        (x)*D1)

    U = init.(xs)
```

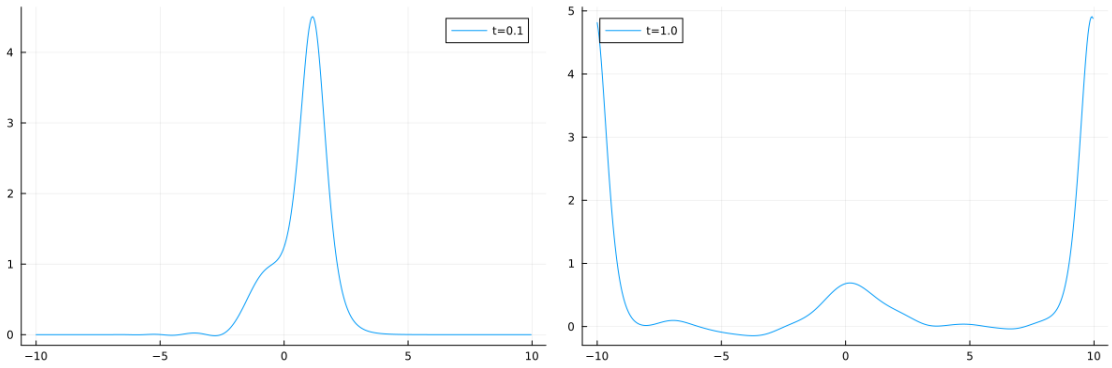
```
    ULinear = init.(xs)

    #run newton's method
    for t=k:k:stop
        #solve the linear case to get init cond
        ULinear = (I + k/(4*h^3)*D3) \ ((I-k/(4*h^3)*D3)*ULinear)
        U = Newton(U,ULinear,m,f,Df)
    end
    return xs,U
end
```

Now using $h = 0.004$ and $k = 0.01$ we plot the solution at $t \in [0.1, 1, 2, 4]$ to get the figures in Figure 2. The solutions appear to show a solitary wave similar to the initial condition.



(a) Solution at $t = 0.1$.

(b) Solution at $t = 1.0$.



(c) Solution at $t = 2.0$.

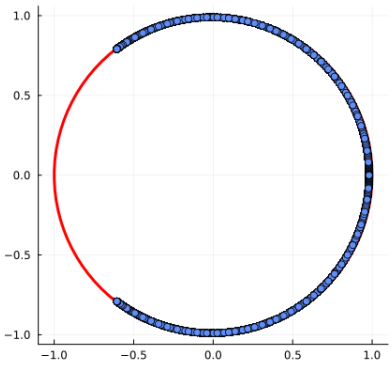
(d) Solution at $t = 4.0$.

Figure 2: Solutions at $t = [0.1, 1.0, 2.0, 4.0]$ to the KdV equation using a second-order accurate method.

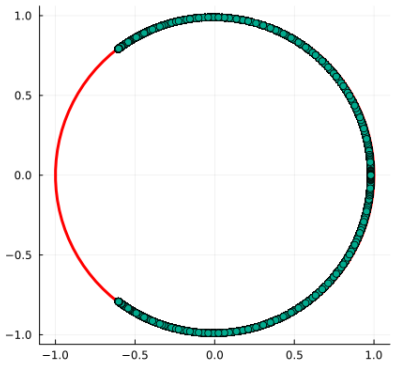
(b) We hope that this second-order method is Lax-Richtmyer stable when studying the eigenvalues of the Jacobian used in Newton's method. Since we are time-stepping using the trapezoid rule, we will map the eigenvalues of the Jacobian using $z = k\lambda$ where $z \mapsto \frac{1-z}{1+z}$. For the trapezoid method to be stable, we expect the eigenvalues to lay within the unit circle. To gain an understanding of the eigenvalues' behavior during the solving process, we compute, using Julia's `eigvals` function, and graph the eigenvalue of each Jacobian during Newton's method and apply the above mapping. We modified the Newton's method presented above as following

```
function Newton(U,U0,m,F,DF)
    count = 0
    New_U = U0
    update = fill(100,m+1)
    while maximum(abs.(update)) > 1e-10
        count += 1
        if count == 300
            @printf("No convergence")
        break
        end
        update = DF(New_U)\F(New_U,U)
        New_U = New_U - update
        #compute evals
        evals = eigvals(Df(New_U) |> Matrix)
        bur = z -> (1-z)/(1+z)
        lambdamap = bur.(k*evals)
        graph = plot!(real(lambdamap),imag(lambdamap),seriestype=:scatter
            ,xaxis=[-1.1,1.1],legend=false)
    end
    return New_U
end
```

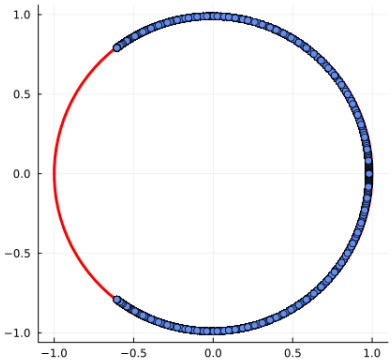
Running the same problem presented in the previous part for $t \in [0.15, 0.30, 0.6, 1.20]$, we compute the eigenvalues of 60, 120, 242, and 489 Jacobians and plotting their values we see that the eigenvalues all lay along the unit circle and do not extend far outside, as seen in Figure 3. We thus expect this method to be Lax-Richtmyer stable due to the consistency of eigenvalues laying on and near the unit disk.



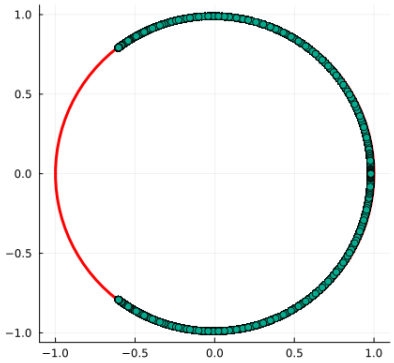
(a) Mapped eigenvalues of 60 Jacobains at $t = 0.15$.



(b) Mapped eigenvalues of 120 Jacobains at $t = 0.30$.



(c) Mapped eigenvalues of 242 Jacobains at $t = 0.60$.



(d) Mapped eigenvalues of 489 Jacobains at $t = 1.20$.

Figure 3: Mapped eigenvalues of Jacobian for solutions at $t \in [0.15, 0.30, 0.6, 1.20]$ to the KdV equation using a second-order accurate method.

□

Problem 3 *Consider solving the small-dispersion (semi-classical) focusing NLS equation*

$$\begin{cases} i\epsilon u_t + \frac{\epsilon^2}{2} u_{xx} + |u|^2 u = 0, \quad -\infty < x < \infty, \\ u(x,0) = A(x) e^{iS(x)/\epsilon}, \\ A(x) = -\operatorname{sech}(x), \\ S(x) = -\mu \log\cosh(x), \quad \mu = 0.1. \end{cases}$$

Use the Fourier exponential integrator with Runge–Kutta 4 to solve this on $[-L, L]$ for sufficiently large L. Use $\epsilon = 0.1$ and $\epsilon = 0.05$ and produce a contour plot of the squared modulus of the solution for $t \in [0, 4]$ for both values of ϵ. Argue that you have chosen L sufficiently large and have chosen a sufficiently large number of Fourier modes.

Solution. Consider solving the small-dispersion (semi-classical) focusing NLS equation

$$\begin{cases} i\epsilon u_t + \frac{\epsilon^2}{2} u_{xx} + |u|^2 u = 0, \quad -\infty < x < \infty, \\ u(x,0) = A(x) e^{iS(x)/\epsilon}, \\ A(x) = -\operatorname{sech}(x), \\ S(x) = -\mu \log\cosh(x), \quad \mu = 0.1. \end{cases}$$

Rearranging the equation to get

$$u_t = \frac{\epsilon i}{2} u_{xx} + \frac{i}{\epsilon} |u|^2 u,$$

where i denotes the imaginary unit. We wish to use the Fourier exponential integrator with RK4 to solve this on $[-L, L]$ for sufficiently large L. We begin by defining

$$\mathcal{F}_N, \mathcal{F}_N^{-1},$$

to be the discrete Fourier transform and its inverse, tailored to the interval $[-L, L]$. That is, if

$$f(x) = \frac{1}{\sqrt{2L}} \sum_{j=-N_-}^{N_+} \check{c}_j e^{ij\frac{\pi}{L}x},$$

then

$$\mathcal{F}_N\left(\begin{bmatrix} \\ f(\check{x}_l) \\ \\ \end{bmatrix}\right) = \begin{bmatrix} \\ c_j \\ \\ \end{bmatrix},$$

$$\mathcal{F}_N^{-1}\left(\begin{bmatrix} \\ c_j \\ \\ \end{bmatrix}\right) = \begin{bmatrix} \\ f(\check{x}_l) \\ \\ \end{bmatrix},$$

where $\check{x}_l = -L + 2L\left(\frac{l-1}{N}\right), l = 1, 2, \cdots, N$ and $N_+ = \lfloor \frac{N}{2} \rfloor$ and $N_- = \lfloor \frac{N-1}{2} \rfloor$. We also define

$$\mathcal{D}_N = \frac{i\pi}{L}\text{diag}(-N_-, -N_- + 1, \cdots, N_+).$$

Now we can discretize the nonlinear term $|u|^2 u$ as

$$\mathcal{A}\left(\begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix}\right) = \frac{i}{\epsilon}\mathcal{F}_N\left(\left|\mathcal{F}_n^{-1}\left(\begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix}\right)\right|^2 \circ \mathcal{F}_n^{-1}\left(\begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix}\right)\right),$$

where $\circ$ denotes that Hadamard product. So, we rewrite the PDE as

$$\begin{bmatrix} \\ \check{c}_j(t)' \\ \\ \end{bmatrix} = \frac{\epsilon i}{2} D_N^2 \begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix} + \mathcal{A}\left(\begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix}\right).$$

Now letting

$$\begin{bmatrix} \\ v_j(t) \\ \\ \end{bmatrix} = e^{\frac{-\epsilon i}{2} D_N^2 t} \begin{bmatrix} \\ \check{c}_j \\ \\ \end{bmatrix},$$

we get

$$\begin{bmatrix} \\ v_j(t)' \\ \\ \end{bmatrix} = e^{\frac{-\epsilon i}{2} D_N^2 t} \mathcal{A}\left(e^{\frac{\epsilon i}{2} D_N^2 t} \begin{bmatrix} \\ v_j(t) \\ \\ \end{bmatrix}\right).$$

Hoping that

$$\begin{bmatrix} \\ u(\check{x}_j, t) \\ \\ \end{bmatrix} \approx \mathcal{F}_N^{-1}\left(\begin{bmatrix} \\ \check{c}_j(t) \\ \\ \end{bmatrix}\right),$$

we employ RK4 as a time stepper. The method is implemented in Julia as follows:

```
mfftshift = x -> circshift(fftshift(x), isodd(length(x)) ? 1 : 0)
mfft = x -> fftshift(fft(fftshift(x),1)) # fft(x,1) is used so that
mifft = x -> mfftshift(ifft(mfftshift(x),1))
mgrid = (n,L) -> -L .+ 2*L*(0:n-1)/n
diffvec = (L,m,j) -> ((-floor(m/2):1:floor((m-1)/2))*(1im*pi/L)).^j

function rk4(F,k,t,c)
    f1 = k*F(c,t)
    f2 = k*F(c + .5*f1, t + .5*k)
    f3 = k*F(c + .5*f2, t + .5*k)
    f4 = k*F(c + f3, t + k)
    return c + 1/6.0*(f1 + 2.0*f2 + 2.0*f3 + f4)
end
```

```
function solve_shrody(eps,k,T,L,m)
    mu = 0.1
    S(x) = mu * log.(cosh.(x))
    A = x -> - sech.(x)
    eta = x -> A.(x).*exp.(1im*S.(x)/eps)

    X = mgrid(m,L)
    c = mfft(eta(X))
    U = mifft(c)
    cl = [-2,2]

    d2 = (eps* 1im)/(2)*diffvec(L,m,2)

    N = v -> (1im/eps)*v.*abs2.(v)

    F = (v,tau) -> exp.(-d2*tau).*(mfft(N(mifft(exp.(d2*tau).*v))))


    n = convert(Int64,ceil(T/k))

    ts = 0:k:T
    z = zeros(size(X)[1],n+1)

    for i = 2:n+1
        c = exp.(d2*k).*rk4(F,k,0.0,c)
        U = mifft(c)
        z[:,i] = abs2.(U)
    end
    return ts,X,z
end
```

To find an L that is sufficiently large for the method, we require the solution, note that throughout our work, 'solution' refers to the modulus squared of the approximate solution, to go to zero at the edges of the domain. Let's pick $L = 25$ and using $h = 0.001$ and $t \in [0, 4]$, we compute the maximum value along the edges throughout time using the z matrix the code generates. In Julia, we implement this as

```
z = solve_shrody(epsilon,0.001,4,25,m)[3]
max_l = maximum(z[1,:])
max_r = maximum(z[end,:])
```

Running the code for $\epsilon = 0.1$ and $\epsilon = 0.05$ we find that:

	$\epsilon = 0.1$	$\epsilon = 0.05$
`max_l`	3.28135217179093e-11	5.658338560671543e-7
`max_r`	2.4169893980594823e-11	2.971566893335267e-7

Therefore $L = 25$ seems like a reasonable choice for both $\epsilon = 0.1$ and $\epsilon = 0.05$. On later thought, we notice that the boundary conditions are periodic and thus the computation of `max_r` is not needed.

Next, we will find a sufficient number of Fourier modes m. Let's pick $m = 2^{15}$. To check if this choice of m is reasonable, we compute the solution with $L = 25, h = 0.001, t \in [0, 4]$ and $m = 2^{15}$ and $m/2$. Then comparing the maximum difference between the solutions using the following Julia code

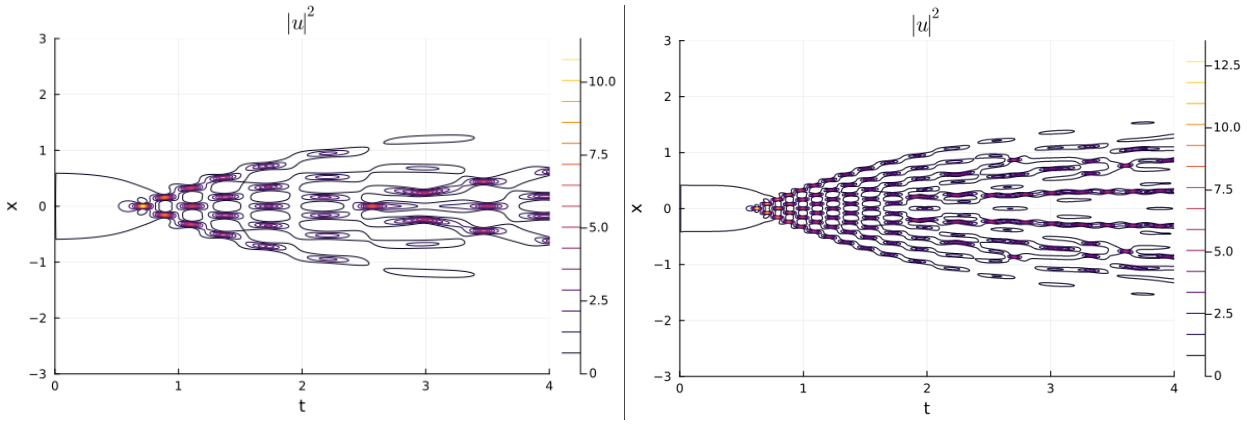
```
z = solve_shrody(epsilon,0.001,4,2^15)
z_half = solve_shrody(epsilon,0.001,4,2^15/2)
dif = z_half - z[1:2:end,:]
maxdif = maximum(dif)
```

Running this method for $\epsilon = 0.1$ and $\epsilon = 0.05$ we get that the max difference is:

	$\epsilon = 0.1$	$\epsilon = 0.05$
`maxdif`	6.266898111562114e-11	4.624698917155001e-7

Since the error between the solution with m modes and $m/2$ modes is small, and thus we argue that our choice of m is sufficiently large.

Now that we have found sufficient L and m values, we compute the solution using $h = 0.001, L = 25, m = 2^15,$ and $t \in [0, 4]$ to get the contour plots presented in figure 4. We note that looking at the contour plots we once again see that that our choice of L is sufficient.



(a) Contour when $\epsilon = 0.1$.

(b) Contour when $\epsilon = 0.05$.

Figure 4: Contour plot of the solution to focusing NLS using Fourier exponential using RK4 where $\epsilon = 0.1$ and $\epsilon = 0.05$.

□

Problem 4 *Consider solving the Kuramoto-Sivashinsky equation with periodic boundary conditions*

$$\begin{cases} u_t + uu_x + u_{xx} + u_{xxxx} = 0, \quad -L < x < L, \\ u(x,0) = \eta(x), \\ u(-L,t) = u(L,t), \\ u_x(-L,t) = u_x(L,t), \\ u_{xx}(-L,t) = u_{xx}(L,t), \\ u_{xxx}(-L,t) = u_{xxx}(L,t). \end{cases}$$

Use the ETDRK4 method using Fourier series to solve this problem with $L = 16\pi$, $\eta(x) = \cos(x/16)(1+\sin(x/16))$. *Create a contour plot of the solution for* $t \in [0, 150]$.

Solution.
Consider the Kuramoto-Sivashinsky equation with periodic boundary conditions

$$\begin{cases} u_t + uu_x + u_{xx} + u_{xxxx} = 0, \quad -L < x < L, \\ u(x,0) = \eta(x), \\ u(-L,t) = u(L,t), \\ u_x(-L,t) = u_x(L,t), \\ u_{xx}(-L,t) = u_{xx}(L,t), \\ u_{xxx}(-L,t) = u_{xxx}(L,t). \end{cases}$$

We will modify the given ETDRK4 method using Fourier series to solve this problem with $L = 16\pi$, $\eta(x) = \cos(x/16)(1+\sin(x/16))$. The modified code looks like

```
init = x -> cos.(x ./ 16) .* (1 .+ sin.(x ./ 16))

L = 16 * pi
N = 2^10
X = mgrid(N,L)
c = mfft(init(X))
D1 = diffvec(L,N,1)
D2 = diffvec(L,N,2)
D4 = diffvec(L,N,4)
U = mifft(c)
cl = [-10,10]
k = 0.1

NF = c -> mfft(-mifft(c).*mifft(D1.*c))

S0 = Diagonal(exp.(-(D2+D4)*k/2))
```

```
S1 = Diagonal(exp.(-(D2+D4)*k))

E0 = k*stable_eval(f0, -(D2+D4)*k, 2) |> Diagonal
E1 = k*stable_eval(f1, -(D2+D4)*k, 2) |> Diagonal
E2 = k*stable_eval(f2, -(D2+D4)*k, 2) |> Diagonal
E3 = k*stable_eval(f3, -(D2+D4)*k, 2) |> Diagonal

ts = 0:k:150
z = zeros(size(X)[1],size(ts)[1])

for i = 2:1500
    aa = S0*c + E0*NF(c)
    bb = S0*c + E0*NF(aa)
    cc = S0*aa + E0*(2*NF(bb)-NF(c))
    c = S1*c + E1*NF(c) + E2*(NF(aa) + NF(bb)) + E3*NF(cc)
    z[:,i] = mifft(c) |> real
end

#create contour plot
contour(ts,X,z)
```

Then using the `contour` function we create the contour plot shown in Figure 5 of the solutions for $t \in [0, 150]$.

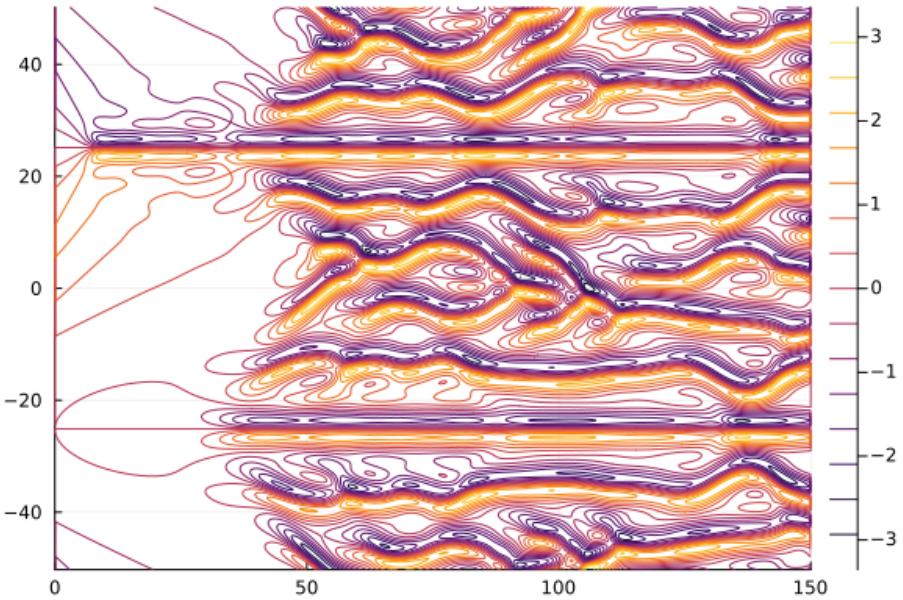


Figure 5: Contour plot of solutions to the Kuramoto-Sivashinsky equation using ETDRK4 for $t \in [0, 150]$.

□