

Homework #3

Due date: February 29, 2024

Problem 1 · Develop a general purpose solver for systems of equations of the form

$$\frac{d\mathbf{U}(t)}{dt} = \mathbf{F}(t, \mathbf{U}(t))$$

where $\mathbf{U}(t)$ is a vector of functions of time, and \mathbf{F} is an operator that depends on the governing equations under consideration. Your code should take the following inputs:

- 📥 The ability to choose between periodic or non-periodic domains.
- 📥 The ability to choose the finite-difference spatial derivative operator.
- 📥 The ability to choose the governing equation.
- 📥 The length L of the physical domain, $x \in [0, L]$.
- 📥 The length T of the temporal domain, $t \in [0, T]$.
- 📥 The number N_x of discrete grid points $x_n, n \in \{0, 1, \dots, N_x - 1\}$.
 - ↔ If the domain is periodic, then $\Delta x = L/N_x$ (end-point $x = L$ excluded).
 - ↔ If the domain is not periodic, then $\Delta x = L/(N_x - 1)$ (end-point $x = L$ included).
- 📥 The number N_t of discrete time levels $t_m, m \in \{0, 1, \dots, N_t - 1\}$ (used only for reporting the solution!).
 - ↔ The time levels are separated by $\Delta t = T/(N_t - 1)$ (end-point $t = T$ included).
- 📥 The initial condition $\mathbf{U}(t = 0)$.

It should return:

- 📤 The set of discrete time levels $t_m, m \in \{0, 1, \dots, N_t - 1\}$.
- 📤 The set of solutions $\mathbf{U}(t_m), m \in \{0, 1, \dots, N_t - 1\}$.

You may use the following Python code, or develop your own:

```
1 import numpy as np
2 from scipy.linalg import circulant
3 from scipy.integrate import solve_ivp
4
5 # Operator from HW1
6 def D_operator_periodic(N, L, R, a):
7     first_row = np.zeros(N); first_row[0:L+R+1] = a; first_row = np.roll(first_row, -L)
8     return np.array(circulant(first_row)).transpose()
9
```

```

10 # Functor for the 1D advection equation
11 def LinearAdv1D(t,U,D):
12     # Initialize velocity
13     a = 1
14     # Return F(t,U)
15     return (-a*D)@U
16
17 # General integrator function
18 def Integrator(periodic,operator,problem,L,T,Nx,Nt,U0):
19     ##### Inputs of the function "Integrator" #####
20     #
21     # periodic : boolean flag to select periodicity (options: True or False)
22     # operator : string to select the spatial derivative operator
23     # problem : string to select the governing equations
24     #     L : length of the physical domain, x runs from 0 to L
25     #     T : length of the temporal domain, t runs from 0 to T
26     #     Nx : number of points to use in x
27     #     Nt : number of points to use in t (for reporting the solutions)
28     #     U0 : initial condition
29     #
30     ##### Outputs of the function "Integrator" #####
31     #
32     #     t : the discrete time levels (in a vector of size Nt)
33     #     U : the solutions (in a matrix of size Nt x Nx)
34     #
35     #####
36
37     # Initialize spatial domain
38     x = np.linspace(0, L, Nx, endpoint=(not periodic))
39     dx = x[1] - x[0]
40
41     # Initialize temporal domain
42     t = np.linspace(0, T, Nt, endpoint=True)
43
44     # Construct spatial matrix operator
45     match (operator,periodic):
46         case ('ForwardOrder1FirstDeriv',True): # Periodic 1st-order forward differences
47             D = D_operator_periodic(Nx,0,1,[-1/dx,1/dx])
48         case ('BackwardOrder1FirstDeriv',True): # Periodic 1st-order backward differences
49             D = D_operator_periodic(Nx,1,0,[-1/dx,1/dx])
50         case _:
51             raise Exception("The %s operator '%s' is not yet implement!" % ('periodic' if periodic
52                                     else 'non-periodic', operator))
53
54     # Solve and return solutions!
55     match problem:
56         case 'LinearAdv1D':
57             # Solve initial value problem; see documentation at:
58             # https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html
59             sol = solve_ivp(LinearAdv1D, [0, T], U0, args=(D,), t_eval=t, rtol=1.0e-6, atol=1.0e-6)
60             # Transpose solution vector so that U has the format (Nt x Nx)
61             U = sol.y.transpose()
62             # Return outputs
63             return t, U
64         case _:
65             raise Exception("The case '%s' is not yet implement!" % problem)

```

Problem 2 · Using the code of Problem 1, solve the problem that we have mainly considered in class so far, i.e., the periodic semi-discrete 1D advection with the wavespeed $a = 1$, domain length $L = 1$, final time $T = 10$, $N_x = 50$ discrete points, and the initial condition

$$u_0(x) = \exp\left(\frac{-\left(x - \frac{L}{2}\right)^2}{2\sigma^2}\right), \quad \sigma = \frac{3}{40}.$$

For each scheme listed below:

- (1) Plot the eigenvalue spectrum of the matrix $\mathbf{A} = -a\mathbf{D}$.
- (2) Compare your numerical solution to the exact expected solution at the times $t_m, m \in \{1, 2, \dots, 10\}$.
- (3) Name and explain the different phenomena you observe based on the previously computed eigenvalue spectra and the known properties of the scheme.

The list of schemes:

- First-order upwind:

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{u_i - u_{i-1}}{\Delta x}$$

- Second-order central:

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

- Third-order upwind:

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{2u_{i+1} + 3u_i - 6u_{i-1} + u_{i-2}}{6\Delta x}$$

- Fourth-order central:

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x}$$

- Sixth-order Padé (**this scheme only needs to be studied by students taking AE 410/CSE 461 for four credit hours**):

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{u_{i+2} + 28u_{i+1} - 28u_{i-1} - u_{i-2}}{36\Delta x} - \frac{1}{3} \left(\left. \frac{\partial u}{\partial x} \right|_{i-1} + \left. \frac{\partial u}{\partial x} \right|_{i+1} \right)$$

Solution:

Using the code from Problem 1, the linear advection equation was solved on the domain $x \in [0, 1]$ with $N_x = 51$ points for the time frame $t \in [0, 10]$ using $u_0(x) = \exp[-(x - L/2)^2/2\sigma^2]$ as initial condition. For the spatial derivative, four difference schemes have been used and their properties based on an eigenanalysis have been investigated. The spectra for these four schemes can be seen in Fig. 1-5, and the following observations can be made:

- 1st **order upwind**: all eigenvalues are complex and lie within the left half-plane \rightarrow all eigenmodes are oscillatory and stable (damped), resulting in substantial amplitude decrease over time.
- 2nd **order central**: all eigenvalues are purely imaginary \rightarrow all eigenmodes are oscillatory and neutrally stable; substantial dispersion can be observed.
- 3rd **order upwind**: all eigenvalues are complex and lie within the left half-plane and are closer to the imaginary axis than the 1st order upwind scheme \rightarrow all eigenmodes are oscillatory and stable (damped), resulting in some amplitude decrease over time.
- 4th **order central**: all eigenvalues are purely imaginary but have a bigger range than the 2nd order scheme thus resolving higher frequency content \rightarrow all eigenmodes are oscillatory and neutrally stable; dispersion can be observed trailing the pulse.
- 6th **order Padé**: all eigenvalues are purely imaginary but have a bigger range than the 4th order scheme thus resolving higher frequency content \rightarrow all eigenmodes are oscillatory and neutrally stable; no dispersion observed, resulting in very good solution quality.

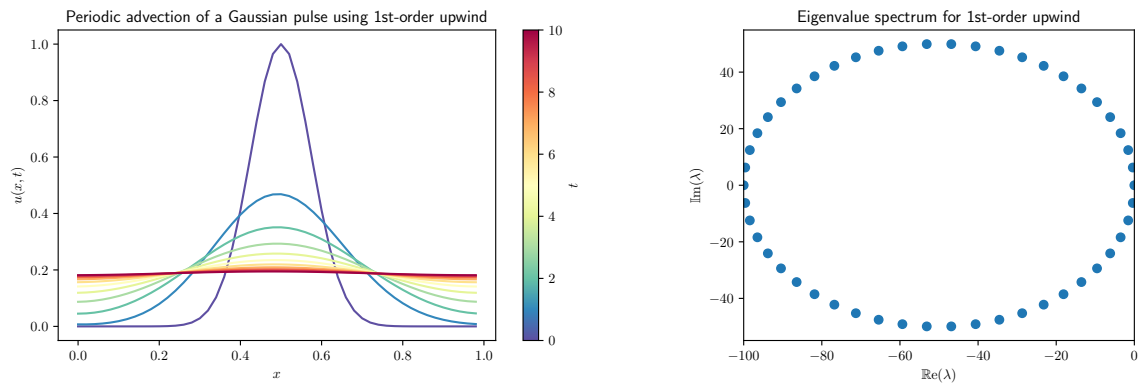


Figure 1: Unsteady solution (left) and spectrum of matrix A (right) for the 1st order upwind scheme

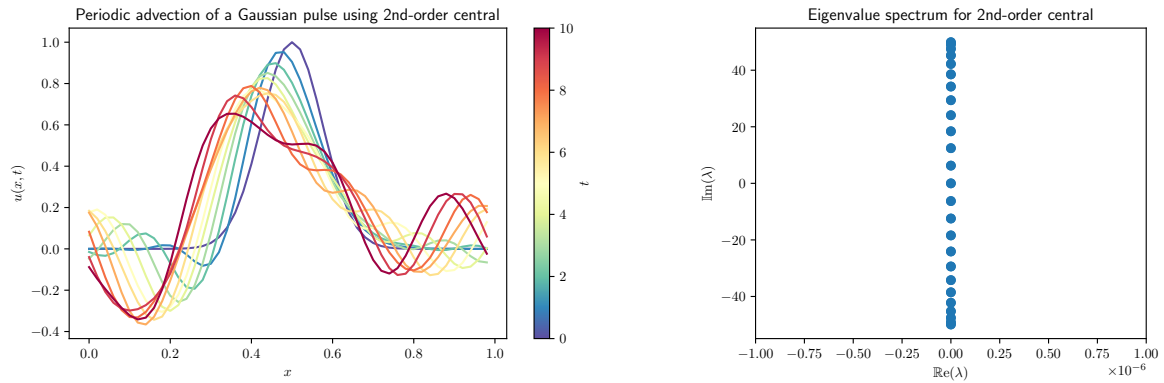


Figure 2: Unsteady solution (left) and spectrum of matrix A (right) for the 2nd order central scheme.

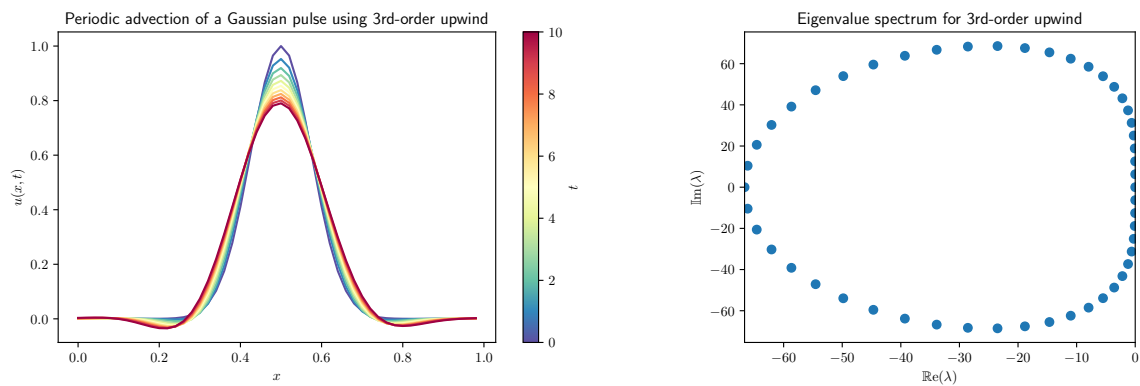


Figure 3: Unsteady solution (left) and spectrum of matrix A (right) for the 3rd order upwind scheme.

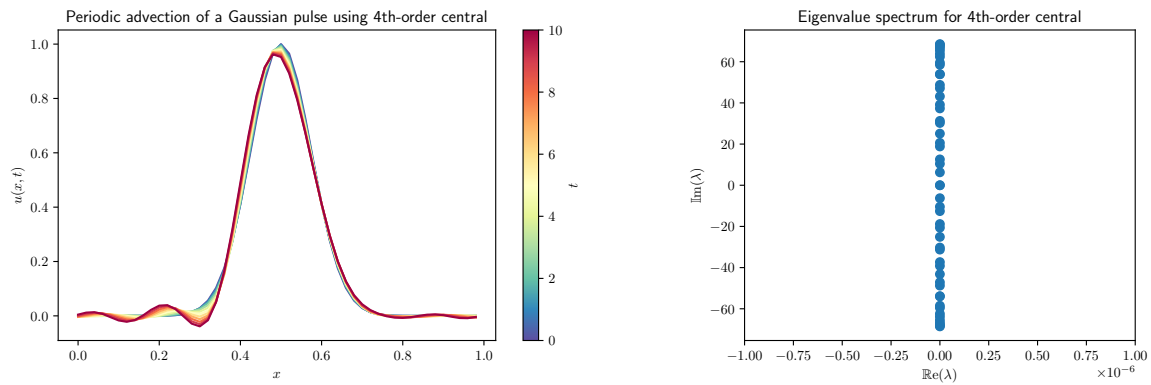


Figure 4: Unsteady solution (left) and spectrum of matrix \mathbf{A} (right) for the 4th order central scheme.

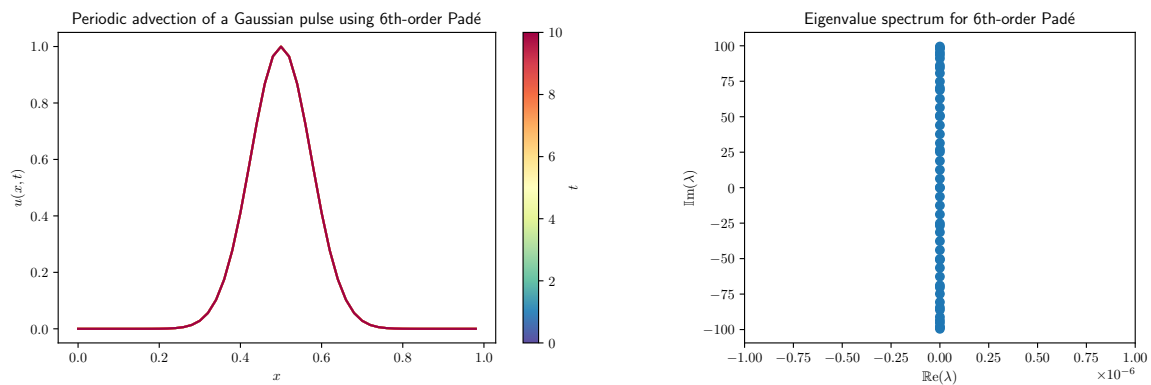


Figure 5: Unsteady solution (left) and spectrum of matrix \mathbf{A} (right) for the 6th order Padé scheme.

Problem 3 · For each of the schemes you have used in Problem 2:

- (1) Derive the analytical expression of the modified wavenumber κ^* using Fourier error analysis.
- (2) Plot the real and imaginary parts of $\kappa^* \Delta x$ as a function of $\kappa \Delta x$, for $\kappa \Delta x \in [0, \pi]$.
- (3) On the same plots as in (2), plot the discrete values of κ_n^* , $n \in \{0, \dots, N_x - 1\}$, obtained from the eigenvalues of the matrix \mathbf{A} .

Solution:

Question (1):

1st order upwind scheme

$$\begin{aligned}
 \left. \frac{\delta f}{\delta x} \right|_n &= \frac{f_n - f_{n-1}}{\Delta x} = \frac{e^{i\kappa x_n} - e^{i\kappa(x_n - \Delta x)}}{\Delta x} = e^{i\kappa x_n} \frac{1 - e^{-i\kappa \Delta x}}{\Delta x} \\
 &= e^{i\kappa x_n} \frac{1}{\Delta x} [i \sin(\kappa \Delta x) - \cos(\kappa \Delta x) + 1] \\
 &= i\kappa^*
 \end{aligned}$$

Thus we get the relation

$$\kappa^* \Delta x = \sin(\kappa \Delta x) + i (\cos(\kappa \Delta x) - 1) \quad (1)$$

2nd order central scheme

$$\begin{aligned} \left. \frac{\delta f}{\delta x} \right|_n &= \frac{f_{n+1} - f_{n-1}}{2\Delta x} = \frac{e^{i\kappa(x_n + \Delta x)} - e^{i\kappa(x_n - \Delta x)}}{2\Delta x} = e^{i\kappa x_n} \frac{e^{i\kappa \Delta x} - e^{-i\kappa \Delta x}}{2\Delta x} \\ &= e^{i\kappa x_n} \frac{i}{2\Delta x} \sin(\kappa \Delta x) \\ &= i\kappa^* \end{aligned}$$

Thus we get the relation

$$\kappa^* \Delta x = \sin(\kappa \Delta x)$$

3rd order upwind scheme

Similarly to the derivations above, we can get

$$\left. \frac{\delta f}{\delta x} \right|_n = e^{i\kappa x_n} \frac{1}{6\Delta x} [2e^{i\kappa \Delta x} + 3 - 6e^{-i\kappa \Delta x} + e^{-2i\kappa \Delta x}]$$

and thus

$$\kappa^* \Delta x = \left(\frac{4}{3} \sin(\kappa \Delta x) - \frac{1}{6} \sin(2\kappa \Delta x) \right) + i \left(\frac{2}{3} \cos(\kappa \Delta x) - \frac{1}{6} \cos(2\kappa \Delta x) - \frac{1}{2} \right)$$

4th order central scheme

$$\left. \frac{\delta f}{\delta x} \right|_n = e^{i\kappa x_n} \frac{1}{12\Delta x} [-e^{2i\kappa \Delta x} + 8e^{i\kappa \Delta x} - 8e^{-i\kappa \Delta x} + e^{-2i\kappa \Delta x}]$$

and thus

$$\kappa^* \Delta x = \frac{4}{3} \sin(\kappa \Delta x) - \frac{1}{6} \sin(2\kappa \Delta x)$$

6th order Padé

$$\kappa^* \Delta x = \frac{1}{36} \frac{2 \sin(2\kappa \Delta x) + 56 \sin(\kappa \Delta x)}{1 + \frac{2}{3} \cos(\kappa \Delta x)}$$

Questions (2) and (3):

As seen in Fig. 6, the modified wave number plots show the previously observed characteristic properties of the differentiation schemes in a more physically relevant framework. Following properties can be observed:

- The wavenumbers obtained from Fourier error analysis or by direct calculation of the eigenvalues of the circulant matrix operator are identical.
- Both central schemes have a zero imaginary part and thus are neutrally stable.
- Both upwinding schemes have non-zero imaginary part and thus are dissipative.
- The upwinding schemes have the identical real part as the central schemes with one order higher, as has been proven for the 1st order upwind scheme in Homework 1.

- Higher order schemes are able to capture higher wave numbers and thus higher frequency content.
- For low wave numbers, all schemes adequately represent the exact wave number.
- The 6th order Padé scheme shows significantly better capturing of high wave number content while maintaining zero dissipation.

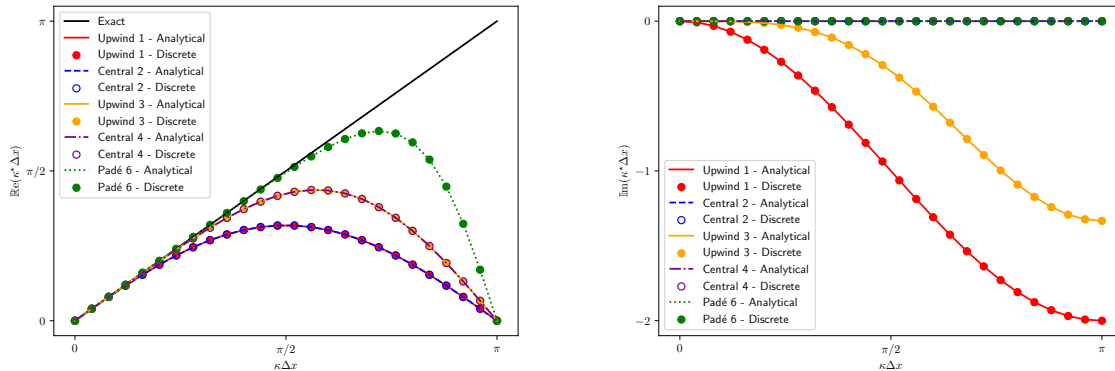


Figure 6: Modified wave number plots for both real (left) and imaginary (right) parts for varying differentiation schemes (symbols represent analytically derived distribution exploiting the circulant properties of the operators)

Solution: Full codes:

Python (0-based indexing)

```

1  # %% [markdown]
2  # Solver for the semi-discrete 1D advection problem:
3
4  # %%
5  import numpy as np
6  from scipy.linalg import circulant
7  from scipy.integrate import solve_ivp
8
9  # Operator from HW1
10 def D_operator_periodic(N,L,R,a):
11     first_row = np.zeros(N); first_row[0:L+R+1] = a; first_row = np.roll(first_row,-L)
12     return np.array(circulant(first_row)).transpose()
13
14 # Functor for the 1D advection equation
15 def LinearAdv1D(t,U,D):
16     # Initialize velocity
17     a = 1
18     # Return F(t,U)
19     return (-a*D)@U
20
21 # General integrator function
22 def Integrator(periodic,operator,problem,L,T,Nx,Nt,U0):
23     ##### Inputs of the function "Integrator" #####
24     #
25     # periodic : boolean flag to select periodicity (options: True of False)
26     # operator : string to select the spatial derivative operator
27     # problem : string to select the governing equations
28     # L : length of the physical domain, x runs from 0 to L
29     # T : length of the temporal domain, t runs from 0 to T
30     # Nx : number of points to use in x
31     # Nt : number of points to use in t (for reporting the solutions)

```

```

32 #         U0 : initial condition                                     #
33 #                                                                 #
34 ##### Outputs of the function "Integrator" #####
35 #                                                                 #
36 #         t : the discrete time levels (in a vector of size Nt)   #
37 #         U : the solutions (in a matrix of size Nt x Nx)         #
38 #                                                                 #
39 #####
40
41 # Initialize spatial domain
42 x = np.linspace(0, L, Nx, endpoint=(not periodic))
43 dx = x[1] - x[0]
44
45 # Initialize temporal domain
46 t = np.linspace(0, T, Nt, endpoint=True)
47
48 # Construct spatial matrix operator
49 match (operator, periodic):
50     case ('1st-order upwind', True): # Periodic 1st-order upwind
51         D = D_operator_periodic(Nx, 1, 0, [-1/dx, 1/dx])
52     case ('2nd-order central', True): # Periodic 2nd-order central differences
53         D = D_operator_periodic(Nx, 1, 1, [-1/(2*dx), 0, 1/(2*dx)])
54     case ('3rd-order upwind', True): # Periodic 3rd-order upwind
55         D = D_operator_periodic(Nx, 2, 1, [1/(6*dx), -1/dx, 1/(2*dx), 1/(3*dx)])
56     case ('4th-order central', True): # Periodic 4th-order central differences
57         D = D_operator_periodic(Nx, 2, 2, [1/(12*dx), -8/(12*dx), 0, 8/(12*dx), -1/(12*dx)])
58     case ('6th-order Padé', True): # Periodic 6th-order Padé
59         DR = D_operator_periodic(Nx, 2, 2, [-1/(36*dx), -28/(36*dx), 0, 28/(36*dx), 1/(36*dx)])
60         DL = D_operator_periodic(Nx, 1, 1, [1/3, 1, 1/3])
61         D = np.linalg.inv(DL)@DR
62     case _:
63         raise Exception("The %s operator '%s' is not yet implement!" % ('periodic' if
64                                     periodic else 'non-periodic', operator))
65
66 # Solve and return solutions!
67 match problem:
68     case 'LinearAdv1D':
69         # Solve initial value problem; see documentation at:
70         # https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.
71         # html
72         sol = solve_ivp(LinearAdv1D, [0, T], U0, args=(D,), t_eval=t, rtol=1.0e-6, atol=1.0e
73             -6)
74         # Transpose solution vector so U has the format (Nt x Nx)
75         U = sol.y.transpose()
76         # Return outputs
77         return t, U, D
78     case _:
79         raise Exception("The case '%s' is not yet implement!" % problem)
80
81 # %% [markdown]
82 # Problem 2:
83
84 # %%
85 import matplotlib as mpl
86 import matplotlib.pyplot as plt
87 from numpy import linalg as LA
88 plt.rcParams['text.usetex'] = True
89 plt.rcParams['figure.dpi'] = 300
90 plt.rcParams['savefig.dpi'] = 300
91 plt.rc('text.latex', preamble=r'\usepackage{amsmath} \usepackage{amssymb}')
92
93 # Initialize case parameters
94 L = 1; T = 10; Nx = 50; Nt = 11; a = 1
95 # Initialize solution at t=0
96 x = np.linspace(0, L, Nx, endpoint=False); dx = x[1] - x[0]
97 sigma = 3/40; U0 = np.exp(-(x-0.5)**2/(2*sigma**2))

```



```

95 # Initialize list of schemes that will be tested:
96 listofschemes = ['1st-order upwind', '2nd-order central', '3rd-order upwind', '4th-order
    central', '6th-order Padé']
97
98 # Plotting the eigenvalue spectra and numerical solutions for each scheme
99 A = []
100 for scheme in listofschemes:
101     # Run solver and plot solution
102     t, U, D = Integrator(True, scheme, 'LinearAdv1D', L, T, Nx, Nt, U0)
103     A.append(-a*D)
104     fig, ax = plt.subplots(1, 1, figsize=(7, 4))
105     ax.plot(x, U0, color=plt.cm.Spectral_r(0))
106     for j in range(1, Nt):
107         ax.plot(x, U[j], color=plt.cm.Spectral_r(t[j]/T))
108     plt.title('Periodic advection of a Gaussian pulse using %s' % scheme)
109     plt.xlabel(r'$x$'); plt.ylabel(r'$u(x,t)$');
110     fig.colorbar(mpl.cm.ScalarMappable(norm=mpl.colors.Normalize(0, T), cmap='Spectral_r'),
        ax=ax, orientation='vertical', label=r'$t$')
111     plt.savefig('solution-%s.pdf' % scheme)
112     plt.show()
113
114 # Plot eigenvalue spectrum
115 fig, ax = plt.subplots(1, 1, figsize=(5, 4))
116 eigenvalues, eigenvectors = LA.eig(-a*D)
117 relambda = [ele.real for ele in eigenvalues]
118 imlambda = [ele.imag for ele in eigenvalues]
119 ax.scatter(relambda, imlambda)
120 plt.xlim(min(-1e-6, np.min(relambda)), max(1e-6, np.max(relambda)))
121 plt.title('Eigenvalue spectrum for %s' % scheme)
122 plt.xlabel(r'$\mathbb{R}\mathrm{e}(\lambda)$'); plt.ylabel(r'$\mathbb{I}\mathrm{m}(\lambda)$');
123 plt.savefig('eigenvalues-%s.pdf' % scheme)
124 plt.show()
125
126 # %% [markdown]
127 # Problem 3:
128
129 # %%
130 import cmath
131 # We only plot the first N/2+1 eigenvalues, corresponding to k*dx in [0,pi]
132 halfNx = int(0.5*Nx+1)
133 kndx = np.zeros(halfNx)
134
135 # Initialize abscissae
136 for n in range(halfNx):
137     kndx[n] = 2*np.pi*n/Nx
138
139 # Exact modified wavenumbers
140 knmoddx_exact_upwind1 = np.sin(kndx) + (np.cos(kndx) - 1) * 1j
141 knmoddx_exact_central2 = np.sin(kndx)
142 knmoddx_exact_upwind3 = 4*np.sin(kndx)/3 - np.sin(2*kndx)/6 + (2*np.cos(kndx)/3 - np.cos
    (2*kndx)/6 - 1/2) * 1j
143 knmoddx_exact_central4 = 4*np.sin(kndx)/3 - np.sin(2*kndx)/6
144 knmoddx_exact_pade6 = (2*np.sin(2*kndx) + 56*np.sin(kndx))/(36*(1+2*np.cos(kndx)/3))
145
146 # Discrete modified wavenumbers
147 knmoddx_discrete_upwind1 = np.zeros(halfNx, dtype=complex)
148 knmoddx_discrete_central2 = np.zeros(halfNx, dtype=complex)
149 knmoddx_discrete_upwind3 = np.zeros(halfNx, dtype=complex)
150 knmoddx_discrete_central4 = np.zeros(halfNx, dtype=complex)
151 knmoddx_discrete_pade6 = np.zeros(halfNx, dtype=complex)
152
153 # First rows the matrix A
154 firstrow_upwind1 = A[0][0]
155 firstrow_central2 = A[1][0]
156 firstrow_upwind3 = A[2][0]

```

```

157 firstrow_central4 = A[3][0]
158 firstrow_pade6 = A[4][0]
159
160 # Loop over all schemes to compute discrete eigenvalues
161 for n in range(halfNx):
162     # n-th eigenvector of the circulant matrix
163     Vn = np.zeros(Nx, dtype=complex)
164     for k in range(Nx):
165         Vn[k] = np.exp(2*np.pi*k*n*1j/Nx)
166     # The n-th eigenvalue of A is the dot product of the n-th eigenvector with the first row
167     # of A (Eq 46 of notes 3)
168     knmoddx_discrete_upwind1[n] = dx*np.dot(firstrow_upwind1, Vn)*1j/a
169     knmoddx_discrete_central2[n] = dx*np.dot(firstrow_central2, Vn)*1j/a
170     knmoddx_discrete_upwind3[n] = dx*np.dot(firstrow_upwind3, Vn)*1j/a
171     knmoddx_discrete_central4[n] = dx*np.dot(firstrow_central4, Vn)*1j/a
172     knmoddx_discrete_pade6[n] = dx*np.dot(firstrow_pade6, Vn)*1j/a
173
174 # Plot real part
175 fig, ax = plt.subplots(1, 1, figsize=(7, 5))
176 ax.plot(kndx, knmoddx_exact_upwind1.real, color='red', label=r'Upwind 1 - Analytical')
177 ax.scatter(kndx, knmoddx_discrete_upwind1.real, color='red', label=r'Upwind 1 - Discrete')
178 ax.plot(kndx, knmoddx_exact_central2.real, color='blue', linestyle='dashed', label=r'
    Central 2 - Analytical')
179 ax.scatter(kndx, knmoddx_discrete_central2.real, facecolors='none', edgecolors='blue',
    label=r'Central 2 - Discrete')
180 ax.plot(kndx, knmoddx_exact_upwind3.real, color='orange', label=r'Upwind 3 - Analytical')
181 ax.scatter(kndx, knmoddx_discrete_upwind3.real, color='orange', label=r'Upwind 3 -
    Discrete')
182 ax.plot(kndx, knmoddx_exact_central4.real, color='purple', linestyle='dashdot', label=r'
    Central 4 - Analytical')
183 ax.scatter(kndx, knmoddx_discrete_central4.real, facecolors='none', edgecolors='purple',
    label=r'Central 4 - Discrete')
184 ax.plot(kndx, knmoddx_exact_pade6.real, color='green', linestyle='dotted', label=r'Padé 6
    - Analytical')
185 ax.scatter(kndx, knmoddx_discrete_pade6.real, color='green', label=r'Padé 6 - Discrete')
186 plt.ylabel(r'$\mathbb{R}\{\mathrm{e}\}(\kappa^*\Delta x)$'); plt.xlabel(r'$\kappa\Delta x$
    ')
187 plt.legend()
188 tick_labels = [r'$0$', r'$\pi/2$', r'$\pi$']
189 ticks = [0, np.pi/2, np.pi]
190 ax.set_xticks(ticks); ax.set_yticks(ticks)
191 ax.set_xticklabels(tick_labels); ax.set_yticklabels(tick_labels)
192 plt.savefig('modwavenumber_real.pdf')
193 plt.show()
194
195 # Plot imaginary part
196 fig, ax = plt.subplots(1, 1, figsize=(7, 5))
197 ax.plot(kndx, knmoddx_exact_upwind1.imag, color='red', label=r'Upwind 1 - Analytical')
198 ax.scatter(kndx, knmoddx_discrete_upwind1.imag, color='red', label=r'Upwind 1 - Discrete')
199 ax.plot(kndx, knmoddx_exact_central2.imag, color='blue', linestyle='dashed', label=r'
    Central 2 - Analytical')
200 ax.scatter(kndx, knmoddx_discrete_central2.imag, facecolors='none', edgecolors='blue',
    label=r'Central 2 - Discrete')
201 ax.plot(kndx, knmoddx_exact_upwind3.imag, color='orange', label=r'Upwind 3 - Analytical')
202 ax.scatter(kndx, knmoddx_discrete_upwind3.imag, color='orange', label=r'Upwind 3 -
    Discrete')
203 ax.plot(kndx, knmoddx_exact_central4.imag, color='purple', linestyle='dashdot', label=r'
    Central 4 - Analytical')
204 ax.scatter(kndx, knmoddx_discrete_central4.imag, facecolors='none', edgecolors='purple',
    label=r'Central 4 - Discrete')
205 ax.plot(kndx, knmoddx_exact_pade6.imag, color='green', linestyle='dotted', label=r'Padé 6
    - Analytical')
206 ax.scatter(kndx, knmoddx_discrete_pade6.imag, color='green', label=r'Padé 6 - Discrete')
207 plt.ylabel(r'$\mathbb{I}\{\mathrm{m}\}(\kappa^*\Delta x)$'); plt.xlabel(r'$\kappa\Delta x$
    ')

```

```

208 plt.legend()
209 ytick_labels = [r'$0$',r'$-1$',r'$-2$']
210 yticks = [0,-1,-2]
211 ax.set_xticks(ticks); ax.set_yticks(yticks)
212 ax.set_xticklabels(tick_labels); ax.set_yticklabels(ytick_labels)
213 plt.savefig('modwavenumber_imag.pdf')
214 plt.show()

```

Matlab (1-based indexing)

```

1  clear all; clc; close all;
2
3  %% input parameter
4  T = 10;
5  Nt = 11;
6  L = 1;
7  Nx = 51;
8
9  %% compute unsteady solution and eigenvalues of differentiation operators
10 [t,u1,x,eigvals1] = integrator(L,Nx,T,Nt,'Upwind1FirstDeriv','LinearAdvection');
11 [t,u2,x,eigvals2] = integrator(L,Nx,T,Nt,'Central2FirstDeriv','LinearAdvection');
12 [t,u3,x,eigvals3] = integrator(L,Nx,T,Nt,'Upwind3FirstDeriv','LinearAdvection');
13 [t,u4,x,eigvals4] = integrator(L,Nx,T,Nt,'Central4FirstDeriv','LinearAdvection');
14 [t,u5,x,eigvals5] = integrator(L,Nx,T,Nt,'Pade6FirstDeriv','LinearAdvection');
15
16 %% plot unsteady solution
17 figure(1), clf; hold on
18 for i=1:length(t)
19     plot(x,u1(i,:), 'LineWidth',2, 'Color', [i/length(t),0,0])
20 end
21 xlim([0,1.4])
22 ylim([-0.3,1.4])
23 set(gca,'FontSize',18)
24 xlabel('x','Interpreter','Latex','FontSize',30)
25 ylabel('u','Interpreter','Latex','FontSize',30)
26 leg=legend('$t=0$ ($u_{exact}$)', '$t=1$', '$t=2$', '$t=3$', '$t=4$', '$t=5$', '$t=6$', ...
27           '$t=7$', '$t=8$', '$t=9$', '$t=10$');
28 set(leg,'Interpreter','Latex','FontSize',22)
29 set(gcf,'PaperPositionMode','Auto')
30 set(gcf,'Position',[100 400 850 400])
31 fname = 'HW3P2_1stup';
32 print(gcf,fname,'-djpeg90')
33
34 figure(2), clf; hold on
35 for i=1:length(t)
36     plot(x,u2(i,:), 'LineWidth',2, 'Color', [i/length(t),0,0])
37 end
38 xlim([0,1.4])
39 ylim([-0.3,1.4])
40 set(gca,'FontSize',18)
41 xlabel('x','Interpreter','Latex','FontSize',30)
42 ylabel('u','Interpreter','Latex','FontSize',30)
43 leg=legend('$t=0$ ($u_{exact}$)', '$t=1$', '$t=2$', '$t=3$', '$t=4$', '$t=5$', '$t=6$', ...
44           '$t=7$', '$t=8$', '$t=9$', '$t=10$');
45 set(leg,'Interpreter','Latex','FontSize',22)
46 set(gcf,'PaperPositionMode','Auto')
47 set(gcf,'Position',[100 400 850 400])
48 fname = 'HW3P2_2ndcentral';
49 print(gcf,fname,'-djpeg90')
50
51 figure(3), clf; hold on
52 for i=1:length(t)
53     plot(x,u3(i,:), 'LineWidth',2, 'Color', [i/length(t),0,0])
54 end

```

```

55 xlim([0,1.4])
56 ylim([-0.3,1.4])
57 set(gca,'FontSize',18)
58 xlabel('x','Interpreter','Latex','FontSize',30)
59 ylabel('u','Interpreter','Latex','FontSize',30)
60 leg=legend('$t=0$ ($u_{exact}$)',' $t=1$',' $t=2$',' $t=3$',' $t=4$',' $t=5$',' $t=6$',...
61           '$t=7$',' $t=8$',' $t=9$',' $t=10$');
62 set(leg,'Interpreter','Latex','FontSize',22)
63 set(gcf,'PaperPositionMode','Auto')
64 set(gcf,'Position',[100 400 850 400])
65 fname = 'HW3P2_3rdup';
66 print(gcf,fname,'-djpeg90')
67
68 figure(4), clf; hold on
69 for i=1:length(t)
70     plot(x,u4(i,:), 'LineWidth',2,'Color',[i/length(t),0,0])
71 end
72 xlim([0,1.4])
73 ylim([-0.3,1.4])
74 set(gca,'FontSize',18)
75 xlabel('x','Interpreter','Latex','FontSize',30)
76 ylabel('u','Interpreter','Latex','FontSize',30)
77 leg=legend('$t=0$ ($u_{exact}$)',' $t=1$',' $t=2$',' $t=3$',' $t=4$',' $t=5$',' $t=6$',...
78           '$t=7$',' $t=8$',' $t=9$',' $t=10$');
79 set(leg,'Interpreter','Latex','FontSize',22)
80 set(gcf,'PaperPositionMode','Auto')
81 set(gcf,'Position',[100 400 850 400])
82 fname = 'HW3P2_4thcentral';
83 print(gcf,fname,'-djpeg90')
84
85 figure(5), clf; hold on
86 for i=1:length(t)
87     plot(x,u5(i,:), 'LineWidth',2,'Color',[i/length(t),0,0])
88 end
89 xlim([0,1.4])
90 ylim([-0.3,1.4])
91 set(gca,'FontSize',18)
92 xlabel('x','Interpreter','Latex','FontSize',30)
93 ylabel('u','Interpreter','Latex','FontSize',30)
94 leg=legend('$t=0$ ($u_{exact}$)',' $t=1$',' $t=2$',' $t=3$',' $t=4$',' $t=5$',' $t=6$',...
95           '$t=7$',' $t=8$',' $t=9$',' $t=10$');
96 set(leg,'Interpreter','Latex','FontSize',22)
97 set(gcf,'PaperPositionMode','Auto')
98 set(gcf,'Position',[100 400 850 400])
99 fname = 'HW3P2_6thpade';
100 print(gcf,fname,'-djpeg90')
101
102 %% plot eigenspectra
103 figure(11), clf;
104 plot(real(eigvals1),imag(eigvals1),'k.')
105 set(gca,'FontSize',18)
106 xlabel('$\Re (\lambda)$','FontSize',30,'Interpreter','Latex')
107 ylabel('$\Im (\lambda)$','FontSize',30,'Interpreter','Latex')
108 xlim([-120,0])
109 ylim([-70,70])
110 leg=legend('$1^{\{st\}}$ order upwind');
111 set(leg,'Interpreter','Latex','FontSize',24)
112 set(gcf,'PaperPositionMode','Auto')
113 set(gcf,'Position',[100 400 450 400])
114 fname = 'HW3P2_1stup_spectrum';
115 print(gcf,fname,'-djpeg90')
116
117 figure(12), clf;
118 plot(real(eigvals2),imag(eigvals2),'k.')
119 set(gca,'FontSize',18)
120 xlabel('$\Re (\lambda)$','FontSize',30,'Interpreter','Latex')

```

```

121 ylabel('$\Im (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
122 xlim([-1, 1]*1e-10)
123 ylim([-70, 70])
124 leg=legend('$2^{\text{nd}}$ order central');
125 set(leg, 'Interpreter', 'Latex', 'FontSize', 24)
126 set(gcf, 'PaperPositionMode', 'Auto')
127 set(gcf, 'Position', [100 400 450 400])
128 fname = 'HW3P2_2ndcentral_spectrum';
129 print(gcf, fname, '-djpeg90')
130
131 figure(13), clf;
132 plot(real(eigvals3), imag(eigvals3), 'k.')
133 set(gca, 'FontSize', 18)
134 xlabel('$\Re (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
135 ylabel('$\Im (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
136 xlim([-80, 0])
137 ylim([-80, 80])
138 leg=legend('$3^{\text{rd}}$ order upwind');
139 set(leg, 'Interpreter', 'Latex', 'FontSize', 24)
140 set(gcf, 'PaperPositionMode', 'Auto')
141 set(gcf, 'Position', [100 400 450 400])
142 fname = 'HW3P2_3rdup_spectrum';
143 print(gcf, fname, '-djpeg90')
144
145 figure(14), clf;
146 plot(real(eigvals4), imag(eigvals4), 'k.')
147 set(gca, 'FontSize', 18)
148 xlabel('$\Re (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
149 ylabel('$\Im (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
150 xlim([-1, 1]*1e-10)
151 ylim([-90, 90])
152 leg=legend('$4^{\text{th}}$ order central');
153 set(leg, 'Interpreter', 'Latex', 'FontSize', 24)
154 set(gcf, 'PaperPositionMode', 'Auto')
155 set(gcf, 'Position', [100 400 450 400])
156 fname = 'HW3P2_4thcentral_spectrum';
157 print(gcf, fname, '-djpeg90')
158
159 figure(15), clf;
160 plot(real(eigvals5), imag(eigvals5), 'k.')
161 set(gca, 'FontSize', 18)
162 xlabel('$\Re (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
163 ylabel('$\Im (\lambda)$', 'FontSize', 30, 'Interpreter', 'Latex')
164 xlim([-1, 1]*1e-10)
165 ylim([-90, 90])
166 leg=legend('$6^{\text{th}}$ order Pade');
167 set(leg, 'Interpreter', 'Latex', 'FontSize', 24)
168 set(gcf, 'PaperPositionMode', 'Auto')
169 set(gcf, 'Position', [100 400 450 400])
170 fname = 'HW3P2_6thpade_spectrum';
171 print(gcf, fname, '-djpeg90')

1 clear all; close all; clc
2
3 %% parameters
4 L = 1;
5 nx = 51;
6 x = linspace(0, 1, nx+2);
7 x = x(1:end-1);
8 dx = L/nx;
9
10 %% analytical eigenvalues of FD schemes
11 kdx = linspace(0, pi, 100);
12 firstupwind = sin(kdx) + i*(cos(kdx)-1);
13 secondcentral = sin(kdx);
14 thirdupwind = -i/6*(3 - 4*cos(kdx) + cos(2*kdx) - 2*i*(-4 + cos(kdx)).*sin(kdx));

```

```

15 fourthcentral = 4/3*sin(kdx)-1/6*sin(2*kdx);
16 sixthpade = 1/36*( 2*sin(2*kdx)+56*sin(kdx) )./( 1+2/3*cos(kdx) );
17
18 %% eigenvalues of circulant matrix for 1st order upwind
19 D = D1_operator( nx,dx,0,1,[-1,1] ); % first upwind
20 [ k_tilde_1up,dxkm_1up ] = circD_wavenum( D );
21 D = D1_operator( nx,dx,1,1,[-1,0,1]/2 ); % second central
22 [ k_tilde_2cfd,dxkm_2cfd ] = circD_wavenum( D );
23 D = D1_operator( nx,dx,1,2,[1,-6,3,2]/6 ); % third upwind
24 [ k_tilde_3up,dxkm_3up ] = circD_wavenum( D );
25 D = D1_operator( nx,dx,2,2,[1,-8,0,8,-1]/12 ); % fourth central
26 [ k_tilde_4cfd,dxkm_4cfd ] = circD_wavenum( D );
27 D = D1_operatorPade( nx,dx ); % sixth Pade
28 [ k_tilde_6pade,dxkm_6pade ] = circD_wavenum( D );
29
30 %% plot
31 line = 3;
32 labelsz = 24;
33 legendsz = 20;
34
35 figure(1), clf;
36 a(1) = subplot(1,2,1); hold on
37 h(1) = plot(kdx,real(firstupwind),'k-','LineWidth',line,'LineStyle','--');
38 h(2) = plot(kdx,real(secondcentral),'b','LineWidth',line,'LineStyle','--');
39 h(3) = plot(kdx,real(thirdupwind),'m','LineWidth',line);
40 h(4) = plot(kdx,real(fourthcentral),'c','LineStyle','--','LineWidth',line);
41 h(5) = plot(kdx,real(sixthpade),'g','LineStyle','--','LineWidth',line);
42 plot(dxkm_1up,real(k_tilde_1up*dx),'k','LineStyle','none','Marker','d')
43 plot(dxkm_2cfd,real(k_tilde_2cfd*dx),'b','LineStyle','none','Marker','x')
44 plot(dxkm_3up,real(k_tilde_3up*dx),'m','LineStyle','none','Marker','>')
45 plot(dxkm_4cfd,real(k_tilde_4cfd*dx),'c','LineStyle','none','Marker','p')
46 plot(dxkm_6pade,real(k_tilde_6pade*dx),'g','LineStyle','none','Marker','p')
47 h(6) = plot(kdx,kdx,'k','LineWidth',line);
48 xlim([0,pi]); ylim([0,pi])
49 set(gca,'FontSize',20)
50 xlabel('$k_m \Delta x$', 'Interpreter','Latex','FontSize',labelsz)
51 ylabel('$Re(\tilde{k}_m \Delta x)$', 'Interpreter','Latex','FontSize',labelsz)
52 leg=legend(h, '$1^{st}$ order upwind', '$2^{nd}$ order central',...
53 '$3^{rd}$ order upwind', '$4^{th}$ order central', '$6^{th}$ order Pad\'e',
54 'exact');
55 set(leg,'Interpreter','Latex','FontSize',legendsz,'Location','NorthWest')
56
57 a(2) = subplot(1,2,2); hold on
58 plot(kdx,imag(firstupwind),'k','LineWidth',line,'LineStyle','--')
59 plot(kdx,imag(secondcentral),'b','LineWidth',line,'LineStyle','--')
60 plot(kdx,imag(thirdupwind),'m','LineWidth',line)
61 plot(kdx,imag(fourthcentral),'c','LineStyle','-.','LineWidth',line)
62 plot(kdx,imag(sixthpade),'g','LineStyle','-.','LineWidth',line)
63 plot(dxkm_1up,imag(k_tilde_1up*dx),'k','LineStyle','none','Marker','d')
64 plot(dxkm_2cfd,imag(k_tilde_2cfd*dx),'b','LineStyle','none','Marker','x')
65 plot(dxkm_3up,imag(k_tilde_3up*dx),'m','LineStyle','none','Marker','>')
66 plot(dxkm_4cfd,imag(k_tilde_4cfd*dx),'c','LineStyle','none','Marker','p')
67 plot(dxkm_6pade,imag(k_tilde_6pade*dx),'c','LineStyle','none','Marker','p')
68 xlim([0,pi])
69 set(gca,'FontSize',20)
70 xlabel('$k_m \Delta x$', 'Interpreter','Latex','FontSize',labelsz)
71 ylabel('$Im(\tilde{k}_m \Delta x)$', 'Interpreter','Latex','FontSize',labelsz)
72
73 set(gcf,'PaperPositionMode','Auto')
74 set(gcf,'Position',[100 400 1200 400])
75 fname = 'HW3P3';
76 print(gcf,fname,'-djpeg90')
77
78 function [t,u,x,eigvals] = integrator(L,Nx,T,Nt,spacedef,casedef)
79 %
80 % Inputs

```

```

4 %
5 % L :: length of physical domain, x runs from 0 to L
6 % Nx :: number of points to use in x
7 % T :: length of temporal domain, t runs from 0 to T
8 % Nt :: number of points to use in t (for reporting solution, ode45 chooses dt internally)
9 % spacedef :: string to select spatial derivative operator
10 % casedef :: string to select governing equation
11 %
12 % Output
13 %
14 % t :: time vector
15 % u :: solution vector
16 % spatial domain
17 dx = L/(Nx+1);
18 x = dx*linspace(0,Nx,Nx+1).';
19 % temporal domain
20 dt = T/(Nt-1);
21 t = dt*linspace(0,Nt-1,Nt);
22 % spatial operator
23 if (strcmp(spacedef,'Central2FirstDeriv'))
24     D1 = (D1_operator(Nx,dx,1,1,[-0.5 0 0.5]));
25 elseif (strcmp(spacedef,'Upwind1FirstDeriv'))
26     D1 = (D1_operator(Nx,dx,0,1,[-1,1]));
27 elseif (strcmp(spacedef,'Upwind3FirstDeriv'))
28     D1 = (D1_operator(Nx,dx,1,2,[1,-6,3,2]/6));
29 elseif (strcmp(spacedef,'Central4FirstDeriv'))
30     D1 = (D1_operator(Nx,dx,2,2,[1,-8,0,8,-1]/12));
31 elseif (strcmp(spacedef,'Pade6FirstDeriv'))
32     D1 = (D1_operatorPade(Nx,dx));
33 else
34     error(sprintf('Unknown spacedef = %s',spacedef));
35 end
36 [~,eigvals] = eig(-D1);
37 eigvals = diag(eigvals);
38 % initial condition
39 u0 = exp(-(x-0.5).^2/(2*(3/40)^2));
40 % run
41 if (casedef == 'LinearAdvection')
42     [t,u] = ode45(@(t,y) LinearAdvection(t,y,D1), t, u0);
43 else
44     error(sprintf('Unknown casedef = %s',casedef));
45 end

1 function udot = LinearAdvection(t,u,D)
2     udot = -D*u;
3     return

1 function [ D ] = D1_operator( n,dx,R,L,a )
2 %computes a finite difference operator
3 %   n = number of grid points
4 %   dx = step size
5 %   [R,L] = right/left bound of stencil
6 %   a = stencils
7
8 % initialize
9 D = zeros(n+1);
10 % fill interior domain
11 for i=1+L:n+1-R
12     D(i,i-L:i+R) = a;
13 end
14 % fill left boundary
15 for i=1:L
16     D(i,1:i+R) = a(L-i+2:end);
17     D(i,end-L+i:end) = a(1:L-i+1);
18 end
19 % fill right boundary

```

```

20 for i=n+2-R:n+1
21     D(i,end-L-1+(i-n):end) = a(1:L+(n+2-i));
22     D(i,1:(i-n+R-1))      = a(L+(n+3-i):end);
23 end
24 D = 1/dx*D;
25 end

1 function [ D ] = D1_operatorPade( n,dx)
2 %computes th order Pade scheme operator for first derivative
3 % n = number of grid points
4 % dx = step size
5 % [R,L] = right/left bound of stencil
6 % a = stencils
7
8 a = [-1,-28,0,28,1]/36;
9 R = 2;
10 L = 2;
11
12 % initialize
13 RHS = zeros(n+1);
14
15 % fill interior domain
16 for i=1+L:n+1-R
17     RHS(i,i-L:i+R) = a;
18 end
19 % fill left boundary
20 for i=1:L
21     RHS(i,1:i+R)      = a(L-i+2:end);
22     RHS(i,end-L+i:end) = a(1:L-i+1);
23 end
24 % fill right boundary
25 for i=n+2-R:n+1
26     RHS(i,end-L-1+(i-n):end) = a(1:L+(n+2-i));
27     RHS(i,1:(i-n+R-1))      = a(L+(n+3-i):end);
28 end
29 RHS = RHS/dx;
30
31 % assemble left matrix
32 a2 = [1/3,1,1/3];
33 R = 1;
34 L = 1;
35 % fill interior domain
36 for i=1+L:n+1-R
37     LHS(i,i-L:i+R) = a2;
38 end
39
40 % initialize
41 LHS = zeros(n+1);
42 R = 1;
43 L = 1;
44
45 % fill interior domain
46 for i=1+L:n+1-R
47     LHS(i,i-L:i+R) = a2;
48 end
49
50 % fill left boundary
51 for i=1:L
52     LHS(i,1:i+R)      = a2(L-i+2:end);
53     LHS(i,end-L+i:end) = a2(1:L-i+1);
54 end
55 % fill right boundary
56 for i=n+2-R:n+1
57     LHS(i,end-L-1+(i-n):end) = a2(1:L+(n+2-i));
58     LHS(i,1:(i-n+R-1))      = a2(L+(n+3-i):end);
59 end

```



```
60
61 D = inv(LHS)*RHS;
62
63 end

1 function [ k_tilde,dxkm ] = circD_wavenum( D )
2 %computes the wavenumber and the modified wavenumber for given
3 %circulant matrix D
4
5 d = D(1,:);
6 n=length(D);
7 for k=1:n
8     sum = 0;
9     for j=1:n
10        sum = sum + d(j)*exp(i*2*pi*(j-1)*(k-1)/n);
11    end
12    lambda(k) = sum;
13    dxkm(k) = 2*pi*(k-1)/(n-1);
14 end
15
16 k_tilde = -i*lambda;
17
18 end
```

Submission guidelines · Instructions on how to prepare and submit your report are available on the course's Canvas page at <https://canvas.illinois.edu/courses/43781/assignments/syllabus>