APL321 : Introduction to Computational Fluid Dynamics
Indian Institute of Technology Delhi
Spring 2024
Prof. Amitabh Bhattacharya

Lab 2

Aditya Agrawal (2021AM10198)

Consider the non-dimensional 1-D convection-diffusion equation for $\phi(x)$ in class:

$$Pe\frac{d\phi}{dx} = \frac{d^2\phi}{dx^2} \tag{1}$$

with boundary conditions $\phi(0) = 0$ and $\phi(1) = 1$. Here Pe is the Peclet number; choose $Pe = 50$ for all the parts in this assignment. The exact solution to $\phi(x)$ is given by $\phi_{exact}(x) = (exp(xPe) - 1)/(exp(Pe) - 1)$.

1. Discretize Eqn. (1) and solve it for $N = 11, 21, 41, 81$ points using central differencing scheme (CDS) for the diffusive term and
   a) First order upward differencing scheme (UDS) for the convective term
   b) Second order CDS for the convective terminate
   Clearly state A_E, A_W, A_P for each case and compare your solutions with the exact solution, on the same graph. You may solve the set of linear equations using a direct equation solver in MATLAB or using the TDMA solver taught in class. You should plot one graph each for parts (a) and (b).

   Solution 1:
   To solve the equation, we first define the total number of points N and the step size of each cell $h = \frac{1}{N-1}$ of the 1 dimensional mesh grid for each value of N.
   For part (a), we solve equation (1) using the central differencing scheme (CDS) for the diffusive term (RHS) and the upward differencing scheme (UDS) for the convective term (LHS). The first and second order approximation of the derivative at node i according to UDS and CDS respectively are given by:

$$\frac{d\phi}{dx}|_i \approx \frac{\phi_i - \phi_{i-1}}{h} \tag{2}$$

$$\frac{d^2\phi}{dx^2}|_i \approx \frac{\phi_{i+1} + \phi_{i-1} - 2\phi_i}{h^2} \tag{3}$$

   Note that we have assumed $u > 0$ for the UDS so that the first order derivative depends on the current and previous node values. Substituting these values in the original convection-diffusion equation (1) yields:

$$(1 + \frac{hPe}{L})\phi_{i-1} - (2 + \frac{hPe}{L})\phi_i + \phi_{i+1} = 0 \tag{4}$$

   Accordingly, the constants for the scheme are $A_E = 1, A_W = 1 + \frac{hPe}{L}$, and $A_P = -(2 + \frac{hPe}{L})$. Thus, we now have to solve a system of N equations with N unknowns

given by:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ A_W & A_P & A_E & 0 & \cdots & 0 \\ 0 & A_W & A_P & A_E & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & 0 & A_W & A_P & A_E \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_N \end{bmatrix} = \begin{bmatrix} \phi_0 \\ 0 \\ \vdots \\ \vdots \\ \phi_L \end{bmatrix}$$

where $\phi_0 = 0$ and $\phi_L = 1$ are the boundary conditions at $x = 0$ and $x = L$.
We now use the inbuilt system of equations solver *linsolve* to solve the above system to get the values of ϕ at all the nodes.

For part (b), we solve equation (1) using the central differencing scheme (CDS) for the diffusive term (RHS) and the convective term (LHS). According to CDS, the first and second order approximation of the derivative at node i are given by:

$$\frac{d\phi}{dx}|_i \approx \frac{\phi_{i+1} - \phi_{i-1}}{2h} \tag{5}$$

$$\frac{d^2\phi}{dx^2}|_i \approx \frac{\phi_{i+1} + \phi_{i-1} - 2\phi_i}{h^2} \tag{6}$$

Substituting these values in the original convection-diffusion equation (1) yields:

$$(1 - \frac{hPe}{2L})\phi_{i+1} + (1 + \frac{hPe}{2L})\phi_{i-1} - 2\phi_i = 0 \tag{7}$$

Accordingly, the constants for the scheme are $A_E = 1 - \frac{hPe}{2L}, A_W = 1 + \frac{hPe}{2L}$, and $A_P = -2$. We use the same system of equations as above with the new constants to get the values of ϕ at all nodes.

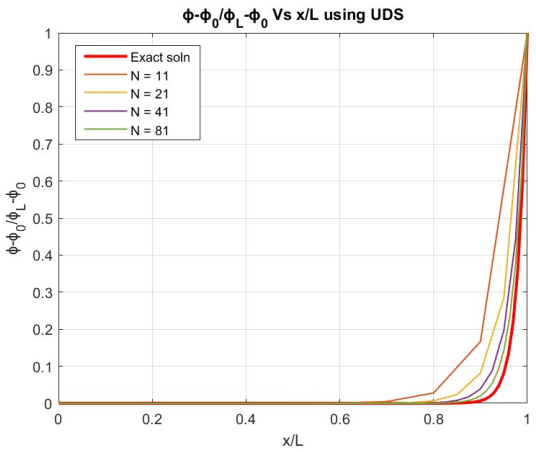
Finally, we plot the solution for ϕ against the position x obtained using the UDS and CDS schemes separately along with the exact solution provided in the problem statement itself. The plots and the MATLAB code implemented are given below:

```
% Question 1

Pe = 50;                    % Setting Peclet No. as 50
phi_0 = 0;                  % Boundary Condition at x = 0
phi_L = 1;                  % Boundary Condition at x = L
N = [11, 21, 41, 81];       % Set of total no. of grid points

% array of arrays to store solutions of UDS for
   convective term
uds = cell(1,size(N,2));
% array of arrays to store solutions of CDS for
   convective term
cds = cell(1,size(N,2));

for i = 1:size(N,2)
```

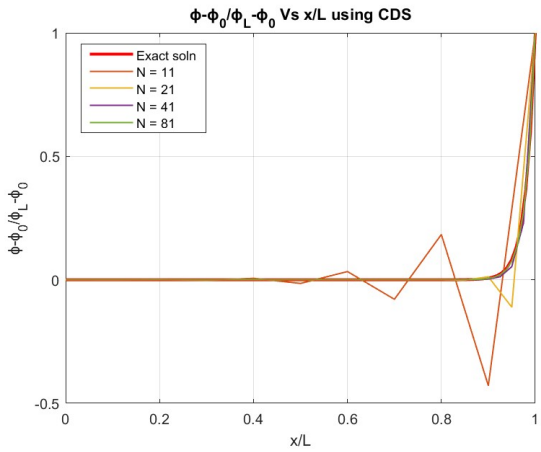
```
    n = N(1,i);                 % No. of grid points
    h = 1/(n-1);                % step size

    % UDS solution assuming u > 0
    % Setting the parameters of UDS
    Ae = 1;                     % coeff. of phi(i+1)
    Ap = -(2 + Pe*h);           % coeff. of phi(i)
    Aw = 1 + Pe*h;              % coeff. of phi(i-1)

    % Defining the required sparse matrix A of size nxn
    A = zeros(n,n);
    A(1,1) = 1; A(n,n) = 1;
    for j = 2:n-1
        A(j,j) = Ap;
        A(j,j-1) = Aw;
        A(j,j+1) = Ae;
    end

    % Defining the load vector b of size nx1
    b = zeros(n,1);
    b(1,1) = phi_0; b(n,1) = phi_L;

    % Using the inbuilt fnc linsolve to solve system of
       equations
    phi = linsolve(A, b);
    % Storing the solution in array of arrays
    uds{1,i} = phi;

    % CDS solution
    % Setting the parameters of CDS
    Ae = 1 - (Pe*h)/2;          % coeff. of phi(i+1)
    Ap = -2;                    % coeff. of phi(i)
    Aw = 1 + (Pe*h)/2;          % coeff. of phi(i-1)

    % Defining the required sparse matrix A of size nxn
    A = zeros(n,n);
    A(1,1) = 1; A(n,n) = 1;
    for j = 2:n-1
        A(j,j) = Ap;
        A(j,j-1) = Aw;
        A(j,j+1) = Ae;
    end

    % Defining the load vector b of size nx1
    b = zeros(n,1);
    b(1) = phi_0; b(n) = phi_L;

    % Using the inbuilt fnc linsolve to solve system of
```

```
        equations
    phi = linsolve(A, b);
    % Storing the solution in array of arrays
    cds{1,i} = phi;
end

x = linspace(0,1,100);
% exact solution of the convection-diffusion eqn
phi_exact = (exp(x*Pe)-1)/(exp(Pe)-1);

% Plotting exact and computed solutions using UDS
figure;
plot(x,phi_exact,'r-','LineWidth',2)
for i=1:size(N,2)
    hold on
    interval = linspace(0,1,N(1,i));
    plot(interval, uds{1,i}, 'LineWidth', 1);
end

title('phi-phi_0/phi_L-phi_0 Vs x/L using UDS');
xlabel('x/L');
ylabel('phi-phi_0/phi_L-phi_0');
legend('Exact soln','N =11','N =21','N =41','N =81');
grid on;
xlim([0.000 1.000]);
ylim([-0.00 1.00]);
legend("Position", [0.15656,0.68826,0.20763,0.20984]);

% Plotting exact and computed solutions using CDS
figure;
plot(x,phi_exact,'r-','LineWidth',2)
for i=1:size(N,2)
    hold on
    interval = linspace(0,1,N(1,i));
    plot(interval, cds{1,i},'LineWidth',1);
end

title('phi-phi_0/phi_L-phi_0 Vs x/L using CDS');
xlabel('x/L');
ylabel('phi-phi_0/phi_L-phi_0');
legend('Exact soln','N =11','N =21','N =41','N =81');
legend('show');
grid on;
xlim([0.000 1.000]);
ylim([-0.50 1.00]);
legend("Position", [0.15497,0.70505,0.20634,0.19689]);
```

MATLAB code for the numerical solution of the 1D convection-diffusion equation



(a) Solution using UDS



(b) Solution using CDS

2. Denote $\phi(x)$ as the discrete solution for grid width $h = \frac{1}{N-1}$. We will denote magnitude of discretization error for a given h as $\epsilon_h = ||\phi_h - \phi_{exact}||$, where $||f|| = \sqrt{\sum_{i=1}^{N} \frac{f(x_i)^2}{N}}$ is a standard deviation norm of function $f(x)$. Plot ϵ_h vs h on a log-log graph for solutions ϕ_h from schemes in parts (a) and (b), for $N = 11, 21, 41, 81$. Also plot reference lines $y_1 = c_1 h$ and $y_2 = c_2 h^2$ on the same log-log axis. Choose c_1 and c_2 so that the reference lines are close to (but not co-inciding with) the ϵ_h vs h curves. Comment on the order of accuracy of both the schemes in part (a) and (b).
Solution 2:
We use the solution found in Question 1 for $N = 11, 21, 41, 81$ and find the standard deviation norm of the vector $\phi_h - \phi_{exact}$ using the inbuilt function *norm*. We do this for the solutions of both part (a) and (b) to get the discretization error ϵ_h values. Once we get these values, we plot these values against h for the respective N on a log-log graph using the inbuilt function *loglog*. On top of the graph, we also plot the lines $y_1 = c_1 h$ and $y_2 = c_2 h^2$ with the suitable values of c_1 and c_2 as reference to compare the rate of growth of the discretization errors of the solutions corresponding to the UDS and CDS schemes.
From the graph, it is evident that the discretization error grows faster for the CDS scheme as compared to the UDS scheme. This is due to the fact that the CDS curve is almost parallel to the quadratic curve y_2 which increases rapidly as compared to a linear curve y_1 which is what the UDS curve follows. Therefore, we can conclude that the discretization error in the CDS scheme is approximately of the order of $O(h^2)$ while that of the UDS scheme is of the order $O(h)$. Another key observation is that CDS outperforms UDS for finer grids and vice-verca for coarser grids.
The plots of ϵ_h vs h and the MATLAB code implemented is given below:

```
% Question 2

h_vals = zeros(size(N,2),1);
err_uds = zeros(size(N,2),1);
err_cds = zeros(size(N,2),1);

for i=1:size(N,2)
    n = N(1,i);              % No. of grid points
    h = 1/(n-1);             % step size
```

```
    h_vals(i) = h;
    x = linspace(0,1,n);
    x = transpose(x);
    phi_exact = (exp(x*Pe)-1)/(exp(Pe)-1);
    err_uds(i) = norm(uds{1,i}-phi_exact)/sqrt(n);
    err_cds(i) = norm(cds{1,i}-phi_exact)/sqrt(n);
end

c1 = 1.2;              % coeff. of linear reference line
c2 = 12;               % coeff. of quadratic reference curve

% Plotting the required graphs
figure;
loglog(h_vals, err_uds, 'LineWidth', 2);
hold on
loglog(h_vals,err_cds,"LineWidth", 2);
hold on
plot(h_vals,c1*h_vals,'--', "LineWidth", 0.5);
hold on
plot(h_vals,c2*h_vals.*h_vals, '--', "LineWidth",0.5);
xlabel('h');
ylabel('epsilon_h');
title('epsilon_h Vs h');
legend('UDS', 'CDS', 'y=c_1h', 'y=c_2h^2');
grid on;
legend("Position", [0.71467,0.15712,0.163,0.17487]);
```

MATLAB code for the variation of discretization error ϵ_h with h

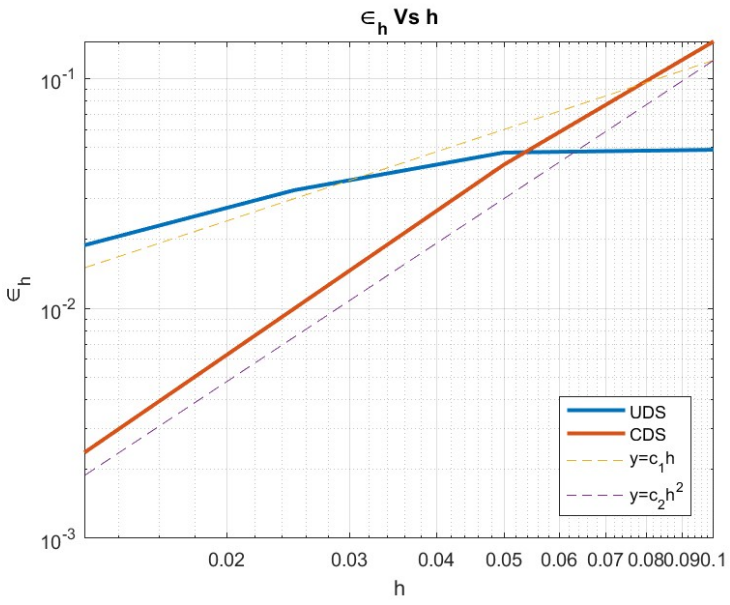


Figure 2: Plot of discretization error (ϵ_h) Vs h on log-log scale

3. In case ϕ_{exact} is not available, the order of accuracy m_h at any can be calculated from $\phi_h, \phi_{2h}, \phi_{4h}$ using the formula:

$$m_h = \frac{1}{ln2} \ln\left(\frac{||\phi_{2h} - \phi_{4h}||}{||\phi_h - \phi_{2h}||}\right) \tag{8}$$

Plot m_h vs h on a graph, for both cases (a) and (b), with h corresponding to $N = 41, 81, 161, 321$. Comment on the trends of m_h with respect to h.

Solution 3:
We use the code of Question 1 to find the solution for $N = 11, 21, 41, 81, 161$ and 321. To calculate m_h, we use the inbuilt *norm* function for the vectors $\phi_h - \phi_{2h}$ and $\phi_{2h} - \phi_{4h}$ and plug them into the given formula for m_h. Note that we use alternate values of the vector with greater no. of components to make the norm operation possible. Finally, we plot the m_h values for both the UDS and CDS solutions against h.
Based on the graph, we can observe that the UDS scheme's m_h curve reduces as cell width increases, which is in line with our expectations. However, the curve for the CDS system reaches a maximum around $h \approx 0.0125$, which indicates a somewhat unusual behavior.

```
% Question 3

% Computing the solutions for N = 161 and 321 as well
N = [11, 21, 41, 81, 161, 321];

uds = cell(1,size(N,2));
cds = cell(1,size(N,2));

for i = 1:size(N,2)

    n = N(1,i);                 % No. of grid points
    h = 1/(n-1);                % step size

    % UDS solution assuming u > 0
    % Setting the parameters of UDS
    Ae = 1;                     % coeff. of phi(i+1)
    Ap = -(2 + Pe*h);           % coeff. of phi(i)
    Aw = 1 + Pe*h;              % coeff. of phi(i-1)

    % Defining the required sparse matrix A of size nxn
    A = zeros(n,n);
    A(1,1) = 1; A(n,n) = 1;
    for j = 2:n-1
        A(j,j) = Ap;
        A(j,j-1) = Aw;
        A(j,j+1) = Ae;
    end

```

```
29      % Defining the load vector b of size nx1
30      b = zeros(n,1);
31      b(1,1) = phi_0; b(n,1) = phi_L;
32
33      % Using the inbuilt fnc linsolve to solve system of
           equations
34      phi = linsolve(A, b);
35      % Storing the solution in array of arrays
36      uds{1,i} = phi;
37
38
39      % CDS solution
40      % Setting the parameters of CDS
41      Ae = 1 - (Pe*h)/2;          % coeff. of phi(i+1)
42      Ap = -2;                    % coeff. of phi(i)
43      Aw = 1 + (Pe*h)/2;          % coeff. of phi(i-1)
44
45      % Defining the required sparse matrix A of size nxn
46      A = zeros(n,n);
47      A(1,1) = 1; A(n,n) = 1;
48      for j = 2:n-1
49          A(j,j) = Ap;
50          A(j,j-1) = Aw;
51          A(j,j+1) = Ae;
52      end
53
54      % Defining the load vector b of size nx1
55      b = zeros(n,1);
56      b(1) = phi_0; b(n) = phi_L;
57
58      % Using the inbuilt fnc linsolve to solve system of
           equations
59      phi = linsolve(A, b);
60      % Storing the solution in array of arrays
61      cds{1,i} = phi;
62  end
63
64  m_uds = zeros(4,1);
65  m_cds = zeros(4,1);
66  h_vals = zeros(4,1);
67
68  for i = 3:6
69      n1 = N(1,i);                % N corresponding to h
70      n2 = N(1,i-1);              % N corresponding to 2h
71      n4 = N(1,i-2);              % N corresponding to 4h
72
73      h = 1/(n1-1);               % step size
74      h_vals(i-2,1) = h;
```

```
75
76      % computing the order of accuracy m_h for uds
77      num=(norm(uds{1,i-1}(1:2:end)-uds{1,i-2}))/sqrt(n4);
78      denom=(norm(uds{1,i}(1:2:end)-uds{1,i-1}))/sqrt(n2);
79      m_uds(i-2,1) = log(num/denom)/log(2);
80
81      % computing the order of accuracy m_h for cds
82      num=(norm(cds{1,i-1}(1:2:end)-cds{1,i-2}))/sqrt(n4);
83      denom=(norm(cds{1,i}(1:2:end)-cds{1,i-1}))/sqrt(n2);
84      m_cds(i-2,1) = log(num/denom)/log(2);
85  end
86
87  % Plotting the m_h vs h curves for UDS and CDS
88  figure;
89  plot(h_vals,m_uds,'LineWidth',1);
90  hold on
91  plot(h_vals,m_cds,'LineWidth',1);
92  title('Order of Accuracy (m_h) Vs h');
93  legend('UDS','CDS');
94  xlabel('h');
95  ylabel('m_h');
96  grid on;
```

MATLAB code for the computation of order of accuracy m_h with h

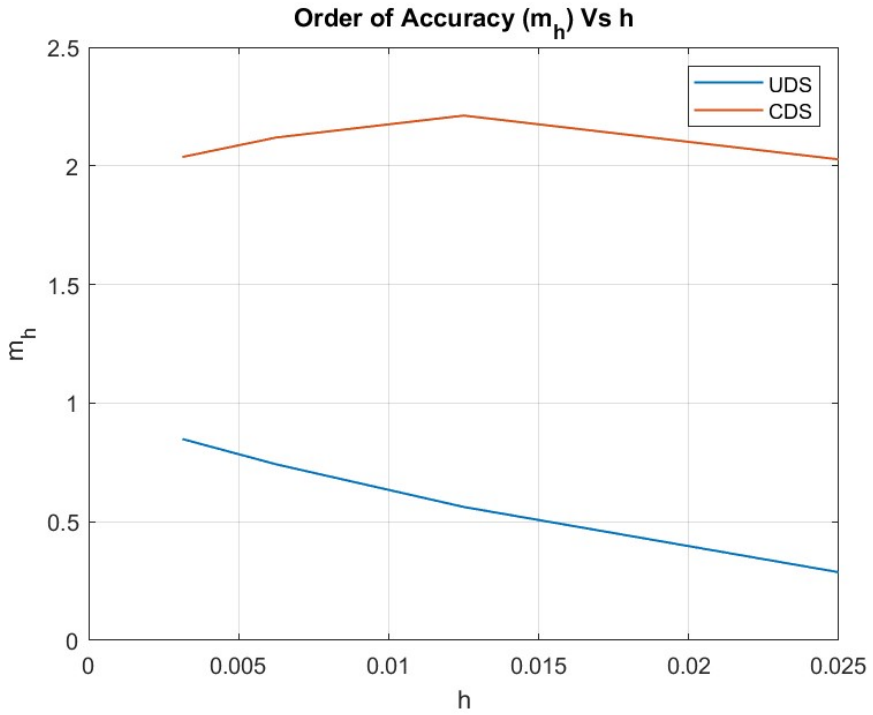


Figure 3: Plot of order of accuracy (m_h) Vs h