APL321 : Introduction to Computational Fluid Dynamics

Indian Institute of Technology Delhi
Spring 2024
Prof. Amitabh Bhattacharya

Lab 3

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 $\phi_{exact}(x) = (exp(xPe) - 1)/(exp(Pe) - 1)$. Discretize Eqn. (1) using a Finite Difference CDS for convective term and diffusion term at N points $x_1, x_2, ...x_N$ on a uniform grid with cell width $h = \frac{1}{N-1}$.

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

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

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

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

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

Accordingly, the constants for the scheme are $A_E = 1 - \frac{hPe}{2L}$, $A_W = 1 + \frac{hPe}{2L}$, and $A_P = -2$. 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 and store this solution for questions ahead.

1. The discretized equation has a form $[L]\vec{\phi}_{exact} + \vec{\epsilon}_\tau = 0$ where L is an $N \times N$ matrix, $\vec{\phi}_{exact} = [\phi_{exact}(x_1)\phi_{exact}(x_2)...\phi_{exact}(x_N)]^T$, and $\vec{\epsilon}_\tau = [\epsilon_\tau(x_1)\epsilon_\tau(x_2)...\epsilon_\tau(x_N)]^T$ is the vector representing truncation error at $x_1, x_2, ...x_N$. The matrix $[L]$ has tridiagonal form, with entries A_W, A_P and A_E at each x_i. Derive and state the values for A_W, A_P and A_E in terms of grid width h and Peclet number for both interior and boundary nodes. Use Taylor series expansion to derive and state the expression $\epsilon_\tau(x_i)$ for only upto second order in h in terms of derivatives of ϕ_{exact} at x_i.

Solution 1:
We begin by using the Taylor Series expansion to find out the values of ϕ at the points x_{i-1} and x_{i+1} for $i = 2, 3, ...N-1$. The series expansion is given by:

$$\phi(x_{i-1}) = \phi(x_i) - h\phi'(x_i) + \frac{h^2}{2!}\phi''(x_i) - \frac{h^3}{3!}\phi'''(x_i) + \frac{h^4}{4!}\phi''''(x_i) + O(h^5) \tag{5}$$

$$\phi(x_{i+1}) = \phi(x_i) + h\phi'(x_i) + \frac{h^2}{2!}\phi''(x_i) + \frac{h^3}{3!}\phi'''(x_i) + \frac{h^4}{4!}\phi''''(x_i) + O(h^5) \tag{6}$$

Adding equations (2) and (3) and solving for $\phi"(x_i)$ gives:

$$\phi''(x_i) = \frac{1}{h^2}[\phi(x_{i+1}) - 2\phi(x_i) + \phi(x_{i-1})] - \frac{h^2}{12}\phi''''(x_i) + O(h^4) \tag{7}$$

Subtracting equations (2) and (3) and solving for $\phi'(x_i)$ gives:

$$\phi'(x_i) = \frac{1}{2h}[\phi(x_{i+1}) - \phi(x_{i-1})] - \frac{h^2}{6}\phi'''(x_i) + O(h^4) \tag{8}$$

Substituting these expressions into the original convection-diffusion equation (1) and ignoring the higher order terms ($O(h^4)$) yields:

$$Pe\left[\frac{1}{2h}\{\phi(x_{i+1}) - \phi(x_{i-1})\} - \frac{h^2}{6}\phi'''(x_i)\right] = \frac{1}{h^2}[\phi(x_{i+1}) - 2\phi(x_i) + \phi(x_{i-1})] - \frac{h^2}{12}\phi''''(x_i) \tag{9}$$

$$\implies \left[\frac{1}{h^2} - \frac{Pe}{2h}\right]\phi(x_{i+1}) - \frac{2}{h^2}\phi(x_i) + \left[\frac{1}{h^2} + \frac{Pe}{2h}\right]\phi(x_{i-1}) + \left[\frac{h^2 Pe}{6}\phi'''(x_i) - \frac{h^2}{12}\phi''''(x_i)\right] = 0 \tag{10}$$

The above equation takes the form

$$[L]\vec{\phi}_{exact} + \vec{\epsilon}_\tau = 0 \tag{11}$$

where the matrix $[L]$ is given by:

$$[L] = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ A_W & A_P & A_E & 0 & \cdots & 0 \\ 0 & A_W & A_P & A_E & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & 0 & A_W & A_P & A_E \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$A_E = \frac{1}{h^2} - \frac{Pe}{2h}, A_P = -\frac{2}{h^2}, A_W = \frac{1}{h^2} + \frac{Pe}{2h}$

Comparing equation (10) with (11) gives the form of ϵ_τ as:

$$\vec{\epsilon_\tau} = \begin{bmatrix} \epsilon_\tau(x_1) \\ \epsilon_\tau(x_2) \\ \vdots \\ \epsilon_\tau(x_N) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{h^2 Pe}{6}\phi'''(x_2) - \frac{h^2}{12}\phi''''(x_2) \\ \vdots \\ 0 \end{bmatrix} \tag{12}$$

where ϕ''' and ϕ'''' are third and fourth order derivatives of the exact solution ϕ_{exact}. Therefore:

$$\phi'''(x_i) = \frac{Pe^3 \exp(x_i Pe)}{\exp(Pe) - 1} \tag{13}$$

$$\phi''''(x_i) = \frac{Pe^4 \exp(x_i Pe)}{\exp(Pe) - 1} \tag{14}$$

Note that $\epsilon_\tau(x_1)$ and $\epsilon_\tau(x_N)$ are 0 since the ϕ values at the boundary conditions are known with full accuracy.

2. Calculate and plot the approximate truncation error $\tilde{\epsilon}_\tau(x_i)$ at for points using only upto second order in in terms of derivatives of $x_1, x_2, ...x_N$ at $N = 41$ points using only terms upto second order in h^2 in terms of derivatives of ϕ_{exact} at x_i. Comment on the trends observed in this plot.
 Solution 2:
We use equations (12), (13) and (14) to calculate the truncation error $\tilde{\epsilon}_\tau(x_i)$ at all x_i. From the plots, we observe that the truncation error increases sharply as x tends to 1. This is because of the exponential solution ϕ_{exact} and its derivatives which means that its values increase exponentially with x. This makes sense because as we approach L, the significance of the higher order terms in the Taylor Series increases and hence accounts for the greater loss in accuracy of the error.
The plots of $\tilde{\epsilon}_\tau(x_i)$ vs h and the MATLAB code implemented are given below:

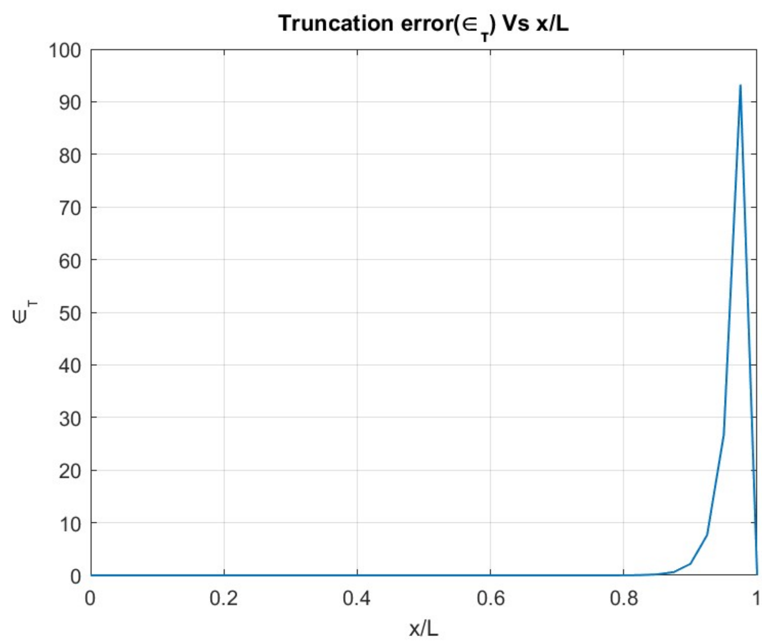

Figure 1: Plot of truncation error $\tilde{\epsilon}_\tau$ Vs x_i

```
% Question 2

Pe = 50;                    % Peclet No. = 50
N = 41;                     % Grid Points = 41
h = 1/(N-1);                % Grid Width = 1/(N-1)
err = zeros(N,1);           % vector to store truncation error
x = linspace(0,1,N);        % x_i values
% third order derivative values of phi_exact
phi3 = (Pe^3*exp(x*Pe))/(exp(Pe)-1);
% fourth order derivative values of phi_exact
phi4 = (Pe^4*exp(x*Pe))/(exp(Pe)-1);

for i = 2:N-1
    % substituting the value of err(i)
    err(i) = (Pe*h*h/6)*phi3(i) - (h*h/12)*phi4(i);
end

% Plot of truncation error vs x_i
figure;
plot(x, err, 'LineWidth',1);
title('Truncation error(epsilon_tau) Vs x/L')
xlabel('x/L');
ylabel('epsilon_tau');
grid on;
```

MATLAB code for the variation of truncation error $\tilde{\epsilon}_\tau$ with x_i

3. The discrete solution $\vec{\phi}_{exact}$ satisfies $[L]\vec{\phi}_h = 0$, where $\vec{\phi}_h = [\phi_h(x_1)\phi_h(x_2)...\phi_h(x_N)]^T$. The discretization error is defined as $\vec{\epsilon_h} = \vec{\phi}_{exact} - \vec{\phi}_h$. Estimate the approximate value of $\vec{\epsilon_h}$, denoted as $\vec{\tilde{\epsilon}}_h$, from the identity $\vec{\tilde{\epsilon}}_h = -[L]^{-1}\vec{\tilde{\epsilon}}_\tau$. Here $\vec{\tilde{\epsilon}}_\tau$ has been calculated from $O(h^2)$ terms in part 2, for $N = 41$ points. Compare $\vec{\tilde{\epsilon}}_h$ and $\vec{\epsilon_h}$, on the same graph, in which $\vec{\epsilon_h}$ and $\vec{\tilde{\epsilon}}_h$, have been plotted with respect to h . Comment on the difference between $\vec{\epsilon_h}$-vs-h and $\vec{\tilde{\epsilon}}_h$-vs-h curves.

Solution 3:
We use the relations given in the question to compute the exact and absolute discretization errors. We can clearly observe from the graph that the exact error is slightly greater than the approximated error especially when x tends to 1. This is due to the fact that we ignored the higher order terms ($O(h^4)$) while computing the truncation error in question 2. While this approximation holds good for lower values of x, it tends to deviate significantly as x increases.
The plots of $\tilde{\epsilon}_h$ and ϵ_h vs h and the MATLAB code implemented are given below:

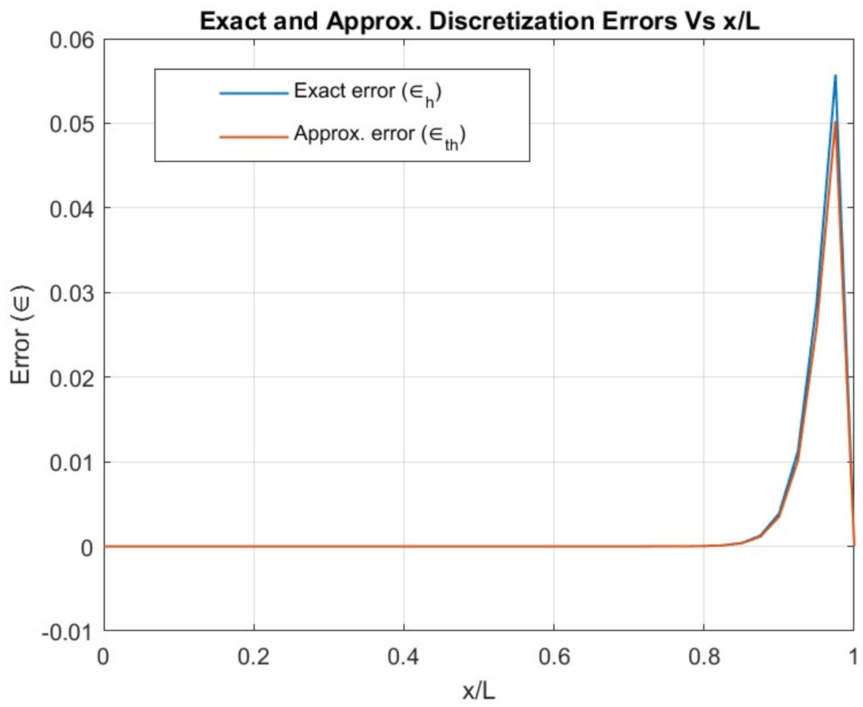

Figure 2: Plot of Exact and Approx. Discretization Errors Vs x/L

```
% Question 3

err_h = phi_exact - phi_h;  % discretization error

L = zeros(N,N);
L(1,1) = 1; L(N,N) = 1;
Aw = 1/h^2 + Pe/(2*h);
Ae = 1/h^2 - Pe/(2*h);
Ap = -2/h^2;

for j = 2:N-1
    L(j,j) = Ap;
    L(j,j-1) = Aw;
    L(j,j+1) = Ae;
end

err_th = -inv(L)*err;   % approx. discretization error

% Plotting exact and approx. discretization errors vs h
figure;
plot(x,err_h,'LineWidth',1);
hold on
plot(x, err_th,'LineWidth',1);
title('Exact and Approx. Discretization Errors Vs x/L')
xlabel('x/L');
ylabel('Error (epsilon)');
legend('Exact error (epsilon_h)','Approx. error (epsilon_
```

```
    {th})');
xlim([0.00 1.00])
ylim([-0.0100 0.0600])
legend("Position", [0.18496,0.7564,0.3824,0.12664]);
grid on;
```

MATLAB code for the computation of $\vec{\tilde{\epsilon}}_h$ and $\vec{\epsilon_h}$

4. Plot $||\vec{\tilde{\epsilon}}_h||$ and $||\vec{\epsilon_h}||$ with respect to h on the same log-log graph, using $N = 41, 81, 161, 321$ points. Show a reference line to indicate the order of accuracy. Here $||.||$ is the standard deviation norm. Comment on the differences between $||\vec{\tilde{\epsilon}}_h||$-vs-$h$ and $||\vec{\epsilon_h}||$-vs-h.

   Solution 4:
   We first find the vectors $\vec{\tilde{\epsilon}}_h$ and $\vec{\epsilon_h}$ using the same methods as in the previous questions. Next, we calculate their standard deviation norms using the inbuilt *norm* function. Finally, we plot these values against h on a log-log scale using the inbuilt *loglog* function.
   From the graph, it is evident that both the curves follow a second order of accuracy since they are almost parallel to the quadratic curve $y = ch^2$. This means that both the exact and approximate discretization errors increase quadratically with grid size h. This obeys our theory since we approximated the errors till the $O(h^2)$ terms. Furthermore, both the norms follow almost the same curve indicating that a second order approximation of the error is highly accurate with the exact error beginning to deviate slightly as h increases on account of the increasing significance of the higher order terms with increasing h.
   The plots of $||\vec{\tilde{\epsilon}}_h||$ and $||\vec{\epsilon_h}||$ vs h and the MATLAB code implemented are given below:

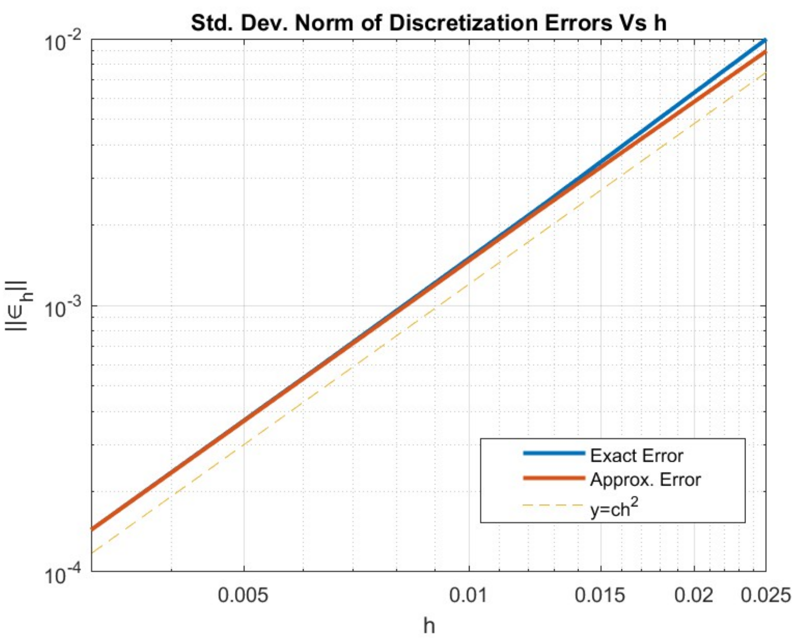

Figure 3: Plot of $||\vec{\tilde{\epsilon}}_h||$ and $||\vec{\epsilon_h}||$ vs h

```
% Question 4

% Solving for phi_exact
N = [41; 81; 161; 321];
phi_exact = cell(size(N,1),1);

for i=1:size(N,1)
    n = N(i);
    x = transpose(linspace(0,1,n));
    phi = (exp(x*Pe)-1)/(exp(Pe)-1);
    phi_exact{i,1} = phi;
end

% Solving for phi_h
phi_h = cell(size(N,1),1);

for i=1:size(N,1)
    n = N(i);                    % Grid Points = 41
    h = 1/(n-1);                 % Grid Width = 1/(N-1)

    % 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
    soln = linsolve(A, b);
    phi_h{i,1} = soln;
end

% Solving for err_h

err_h = cell(size(N,1),1);
```

```

for i = 1:size(N,1)
    err_h{i,1} = phi_exact{i,1} - phi_h{i,1};
end

% Solving for the std. dev. norm of err_h
err_h_norm = zeros(size(N,1),1);

for i = 1:size(N,1)
    val = norm(err_h{i,1})/sqrt(N(i));
    err_h_norm(i) = val;
end

% Solving for truncation_error

trunc_error = cell(size(N,1),1);

for i = 1:size(N,1)
    n = N(i);                   % Grid Points = 41
    h = 1/(n-1);                % Grid Width = 1/(N-1)
    x = transpose(linspace(0,1,n));
    % third order derivative values of phi_exact
    phi3 = (Pe^3*exp(x*Pe))/(exp(Pe)-1);
    % fourth order derivative values of phi_exact
    phi4 = (Pe^4*exp(x*Pe))/(exp(Pe)-1);
    vec = zeros(n,1);
    for j = 2:n-1
        % substituting the value of err(i)
        vec(j) = (Pe*h*h/6)*phi3(j) - (h*h/12)*phi4(j);
    end
    trunc_error{i,1} = vec;
end

% Solving for err_th

err_th = cell(size(N,1),1);
h_vals = zeros(size(N,1),1);

for i = 1:size(N,1)
    n = N(i);                   % Grid Points = 41
    h = 1/(n-1);                % Grid Width = 1/(N-1)
    h_vals(i) = h;
    L = zeros(n,n);
    L(1,1) = 1; L(n,n) = 1;
    Aw = 1/h^2 + Pe/(2*h);
    Ae = 1/h^2 - Pe/(2*h);
    Ap = -2/h^2;

```

```
    for j = 2:n-1
        L(j,j) = Ap;
        L(j,j-1) = Aw;
        L(j,j+1) = Ae;
    end
    % approx. discretization error
    err_th{i,1} = -inv(L)*trunc_error{i,1};
end

% Solving for the std. dev. norm of err_th
err_th_norm = zeros(size(N,1),1);

for i = 1:size(N,1)
    val = norm(err_th{i,1})/sqrt(N(i));
    err_th_norm(i) = val;
end

% Plotting err_h_norm and err_th_norm vs x_i
c = 12;

figure;
loglog(h_vals, err_h_norm, 'Linewidth', 2);
hold on
loglog(h_vals, err_th_norm, 'Linewidth', 2);
hold on
plot(h_vals,c*h_vals.*h_vals, '--', "LineWidth",0.5);
xlabel('h');
ylabel('||epsilon_h||');
title('Std. Dev. Norm of Discretization Errors Vs h');
legend('Exact Error', 'Approx. Error', 'y=ch^2');
grid on;
legend("Position", [0.57785,0.16243,0.30196,0.17396])
```

MATLAB code for the computation of std. dev. norm of $||\vec{\tilde{\epsilon}}_h||$ and $||\vec{\epsilon_h}||$