MA584: Numerical Solutions of PDEs – Finite Difference Methods

Zhilin Li

1 Introduction

1.1 The Problems

We want to solve ordinary/partial differential equations (ODE/PDE), especially linear second order ODE/PDEs, and first order systems, *numerically*.

What is a differential equation? It is an equation whose unknown is a function, say $u(x)$, or $u(x, y)$, or $u(t, x)$, or $u(t, x, y, z)$ etc. The equation involves the derivatives or partial derivatives of the unknown function.

Differential equations[1] have been used intensively to model many physical problems including fluid/solid mechanics, biology, material sciences, economics, ecology, sports and computer sciences, for example, the Navier Stokes equations in fluid dynamics, biharmonic equations for stress, the Maxwell equations in electro-magnetics, and many others.

Unfortunately, while differential equations can describe many physical problems, only very small portion of them can be solved exactly in terms of elementary functions (polynomials, $\sin x$, $\cos x$, $\log x$, e^x, a^x etc.) and their combinations (composite functions). Quite often, even if a differential equation can be solved analytically, great efforts and sound mathematical theories are needed. The closed form of the solution may be too complicated to be actually useful.

If the analytic solution is not available, we want to find an approximate solution to the differential equation. Typically, there are two approaches:

- Semi-analytic methods. Sometime we can approximate the solution using series, integral equations, perturbation techniques, asymptotic approximations etc. The solution is often expressed as some simpler function.

- Numerical approximate solutions in which some numbers are obtained. Nowadays, those numbers are obtained from computers. The development of modern computers

[1] There are other models as well, for example, statistical models.

make it possible to solve many problems that were impossible just a few decades, or years ago.

In this course, we will mainly use the second approach. In Fig.1, we show a flow chart of a problem solving process. In this class, we will focus on numerical solutions using computers, especially the finite difference methods for differential equations.

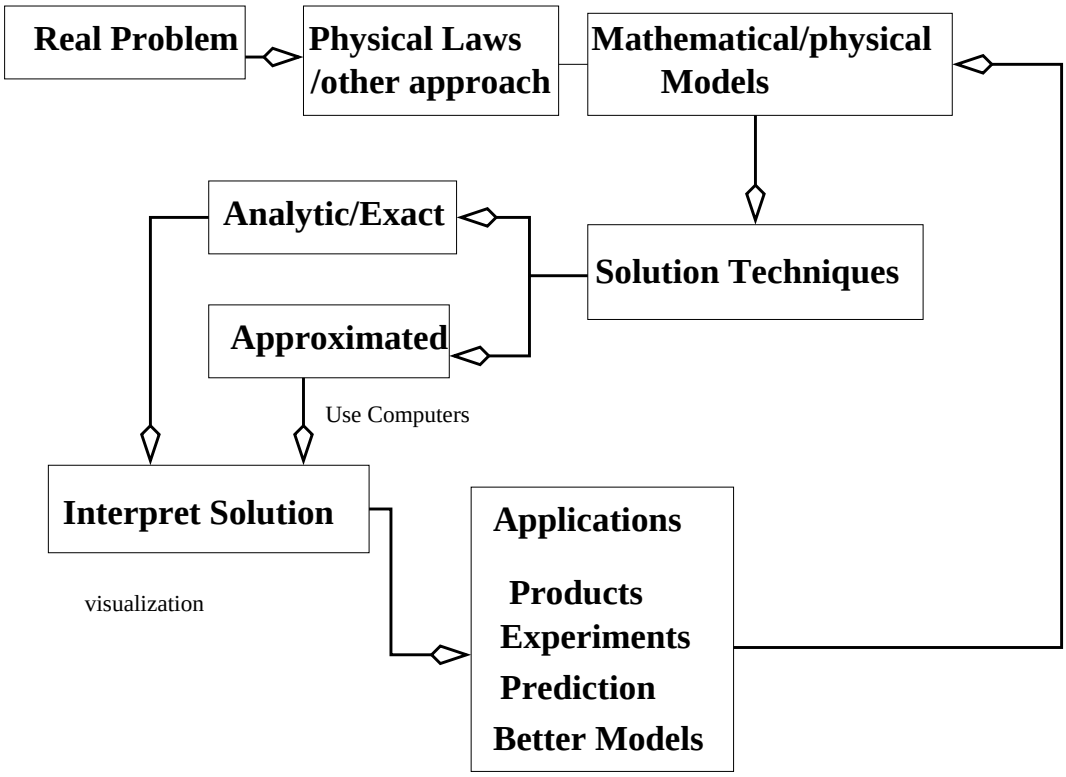

Figure 1: A flow chart of a problem solving process.

The linear differential equations can be classified as *elliptic*, *parabolic*, and *hyperbolic* equations. Below are some examples:

1. Initial value problems and ordinary differential equation (ODE) systems:

$$\begin{aligned} &u''(t) + a(t)u'(t) + b(t)u(t) = f(t), \\ &u(0) = u_0, \quad \boxed{u'(0) = v_0}. \end{aligned} \tag{1.1}$$

The general form can be written as

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{1.2}$$

The system of initial value problems can often be solved using the Runge-Kutta methods with adaptive time steps. In Matlab, it is the ode-suits which include ode45, ode23, ode23s, ode15s, $\cdots$.

2. Boundary value problems: One dimensional example,

$$\begin{aligned} &u''(x) + a(x)u'(x) + b(x)u(x) = f(x), \\ &u(0) = u_0, \quad \boxed{u(1) = u_1}. \end{aligned} \tag{1.3}$$

   Two dimensional example:

$$\begin{aligned} &-(u_{xx} + u_{yy}) = f(x,y), \quad (x,y) \in \Omega \\ &u(x,y) = u_0(x,y), \quad (x,y) \in \partial\Omega. \end{aligned} \tag{1.4}$$

3. Boundary and initial value problems

$$\begin{aligned} &u_t = au_{xx} + f(x,t) \\ &u(0,t) = g_1(t), \quad u(1,t) = g_2(t), \qquad \text{BC} \\ &u(x,0) = u_0(x), \qquad \text{IC} \end{aligned} \tag{1.5}$$

4. Eigenvalue problem:

$$\begin{aligned} &-u''(x) = \lambda u(x) \\ &u(0) = 0, \quad u(1) = 0. \end{aligned} \tag{1.6}$$

   Both $u(x)$ and scalar λ are unknowns.

   Generally, we will consider the second order *linear* partial differential equations of the form

$$a(x,y)u_{xx} + 2b(x,y)u_{xy} + c(x,y)u_{yy} + d(x,y)u_x + e(x,y)u_y + g(x,y)u(x,y) = f(x,y), \tag{1.7}$$

   in a domain Ω, where the coefficients are independent of $u(x,y)$. The equation is linear with respect to u and its partial derivatives. We can classify the equation above as

   - Elliptic if $b^2 - ac < 0$ for all $(x,y) \in \Omega$.
   - Parabolic if $b^2 - ac = 0$ for all $(x,y) \in \Omega$.
   - Hyperbolic if $b^2 - ac > 0$ for all $(x,y) \in \Omega$.

   Different type equation requires different method. Another type of problems is the first order system

$$\frac{\partial \mathbf{u}}{\partial t} = A(\mathbf{x})\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \tag{1.8}$$

   The classification then is determined from the eigenvalues of the matrix $A(\mathbf{x})$.

5. Diffusion and reaction equations:

$$\frac{\partial u}{\partial t} = \nabla \cdot \beta \nabla u + \mathbf{a} \cdot \nabla u + f(u) \tag{1.9}$$

   where $\mathbf{a}$ is a constant vector, $\nabla \cdot \beta \nabla u$ is called a diffusion term, $\mathbf{a} \cdot \nabla u$ is called an advection term, and $f(u)$ is called a reaction term.

6. System of PDEs and non-linear PDEs. A very important one is the incompressible Navier Stokes equations

$$\begin{aligned} &\rho\left(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = \nabla p + \mu\Delta\mathbf{u} + \mathbf{F} \\ &\nabla\cdot\mathbf{u} = 0 \end{aligned} \tag{1.10}$$

A finite difference approach is one of techniques to solve differential equations numerically. The other techniques include finite element methods, the collocation method, spectral method etc.

1.2 An example of a finite difference method for a model problem

Consider a model problem

$$u''(x) = f(x), \quad 0 \le x \le 1, \quad u(0) = u_a, \quad u(1) = u_b.$$

We will solve this problem using the finite difference method to see the general procedure.

The procedure a finite difference method:

1. Generate a grid. For example, we can use a uniform Cartesian grid

$$x_i = i\,h, \quad i = 0, 1, \cdots n, \quad h = \frac{1}{n}.$$

A grid is a finite set of points where we want to find an approximate solution to the differential equation.

2. Substitute the derivatives with some *finite difference* formulas at every grid points where the solution is unknown to get an algebraic system of equations. Notice that for a twice differentiable function $\phi(x)$, we have

$$\phi''(x) = \lim_{\Delta x\to 0} \frac{\phi(x-\Delta x) - 2\phi(x) + \phi(x+\Delta x)}{(\Delta x)^2}$$

Therefore we can approximate $u''(x)$ using nearby function values

$$\phi''(x) = \frac{\phi(x-\Delta x) - 2\phi(x) + \phi(x+\Delta x)}{(\Delta x)^2} + error$$

At each grid point x_i we approximate the differential equation by

$$\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} = f(x_i) + error.$$

We define the approximate solution of $u(x)$ at x_i as U_i as the solution (if exist) of the following linear system of equations

$$\begin{aligned}
\frac{u_a - 2U_1 + U_2}{h^2} &= f(x_1) \\
\frac{U_1 - 2U_2 + U_3}{h^2} &= f(x_2) \\
\frac{U_2 - 2U_3 + U_4}{h^2} &= f(x_3) \\
\cdots\cdots &= \cdots \\
\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} &= f(x_i) \\
\cdots\cdots &= \cdots \\
\frac{U_{n-3} - 2U_{n-2} + U_{n-1}}{h^2} &= f(x_{n-2}) \\
\frac{U_{n-2} - 2U_{n-1} + u_b}{h^2} &= f(x_{n-1}).
\end{aligned}$$

Note that at each grid, the finite difference approximation involve the solution at three grid points x_{i-1}, x_i, and x_{i+1}. The set of these three grid points is called the *finite difference stencil.*

This system of equations can be written as the matrix and vector form:

$$\begin{bmatrix}
-\frac{2}{h^2} & \frac{1}{h^2} & & & & \\
\frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & & \\
 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\
 & & & & \frac{1}{h^2} & -\frac{2}{h^2}
\end{bmatrix}
\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix}
=
\begin{bmatrix} f(x_1) - \frac{u_a}{h^2} \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \frac{u_b}{h^2} \end{bmatrix} \tag{1.11}$$

3. Solve the system of equations to get the approximate solution at each grid point.

4. Implement and debug the computer code. Run the program to get the output. Analyze the results (tables, plots etc.), see below:

5. Error analysis (consistency + stability $\Longrightarrow$ convergence). It is pointwise convergence, that is $\lim_{h\to 0} \|u(x_i) - U_i\|_\infty = 0$. The finite difference method needs the solution to have *second order derivative.*

1.3 The Matlab code for the model problem

The Matlab function for the model problem called two_point.m

```
function [x,U] = two_point(a,b,ua,ub,f,n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This matlab function two_point solvers the following two-point    %
%    boundary value problem: u''(x) = f(x) using center difference     %
%    scheme.                                                           %
%    Input:                                                            %
%     a, b: Two end points.                                            %
%     ua, ub: Dirichlet boundary conditions at a and b                 %
%     f: external function f(x).                                       %
%     n: number of grid points.                                        %
%    Output:                                                           %
%     x: x(1),x(2),...x(n-1) are grid points                           %
%     U: U(1),U(2),...U(n-1) are approximate solution at grid points   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = (b-a)/n; h1=h*h;

A = sparse(n-1,n-1);
F = zeros(n-1,1);

for i=1:n-2,
  A(i,i) = -2/h1; A(i+1,i) = 1/h1; A(i,i+1)= 1/h1;
end
  A(n-1,n-1) = -2/h1;

for i=1:n-1,
  x(i) = a+i*h;
  F(i) = feval(f,x(i));
end
  F(1) = F(1) - ua/h1;
  F(n-1) = F(n-1) - ub/h1;

U = A\F;

return
```

```
%%%%%--------  End of the program ------------------------------------
```

The Matlab function for the model problem called main.m

Let the interval be [0, 1], $f(x) = -\pi^2 \cos(\pi x)$, $u(0) = 0$ and $u(1) = -1$. We can use the following program to solve the problem numerically.

```
%%%%%%%% Clear all unwanted variable and graphs.

clear;  close all

%%%%%%% Input

a =0; b=1; n=40;
ua = 1; ub = -1;

%%%%%% Call solver: U is

[x,U] = two_point(a,b,ua,ub,'f',n);

%%%%%%%%%%%%%%%%%%% Plot and error analysis: %%%%%%%%%%%%%%%%%%%

plot(x,U,'o'); hold

u=zeros(n-1,1);
for i=1:n-1,
  u(i) = cos(pi*x(i));
end
plot(x,u)

%%%%%%% Plot error

figure(2); plot(x,U-u)

norm(U-u,inf)            %%% Print out the maximum error.
```

It is easy to check that the exact solution is $\cos(\pi x)$. If we plot the computed solution that is defined only at the grid points (use plot(x,u,'o'), and the exact solution, solid line in Fig. 2 (a), we see no difference with naked eyes. However, if we plot the difference of the computed solution and the exact solution, which we call the error, we see there is a difference which is about $O(10^{-3})$, see Fig. 2 (b). So we should be happy about our results.

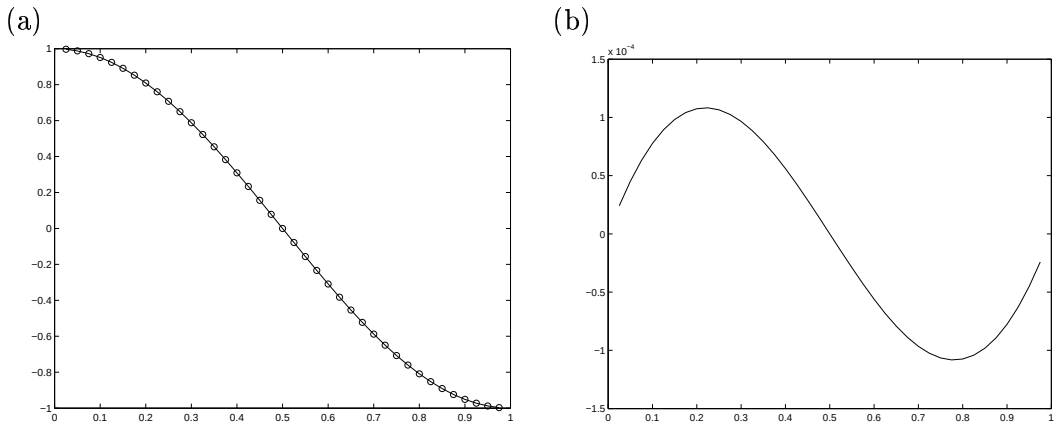

Figure 2: (a) The plot of the computed solution (in little 'o's), and the exact solution (in solid line). (b) The plot of the errors in the infinity norm.

1.4 Questions that we should ask from this example

- Are there other ways using finite difference to approximate derivatives? If so, are there general approaches?
- How do we know the method works or not? If it works, how accurate is it (that is how small is it the error)? This depends on the consistency and the stability.
- Do the computer errors, they are called round off errors, affect the computed solution? If so, how much?
- How do we deal with different boundary conditions such as derivative (Neumann) and mixed (Robin) boundary conditions?
- Do we need different finite difference methods for different problems? If the finite difference method is different, is there a similar procedure?
- How do we know that we are using the most efficient method? What are the criteria? How do we implement an method efficiently?

1.5 Advantages and disadvantages of finite difference and finite element methods

Finite difference methods:

- Simple to use and are easy to understand.
- Easy to implement for rectangular regions in two and three dimensions.
- It is pointwise.

- Many fast solvers and packages, e.g., FFT, for regular domains (rectangular and circular regions).
- Difficult for complicated geometries.
- Have strong regularity requirements (derivatives) for the solutions.

Finite element methods:

- Very successful for structural (elliptic type) problems.
- Natural approach for problems with complicated boundaries.
- Solid theoretical foundations at least for elliptic type problems.
- *Weaker requirement* for the solutions. For example, we do not need u'' for the 1D model problem since only the integral forms are used.
- Many commercial packages.
- Usually coupled with multigrid solvers.
- Need expert knowledge to generate the triangulation which is used to be the most difficult part. Now we can use many packages, for example, Matlab, Triangle, Pltmg, Fidap, Ansys etc.

Finite Difference methods, finite element methods, finite volume methods, spectral methods are all useful for solving PDEs/ODEs.

2 Fundamentals of Finite Difference Methods

The Taylor expansion is the most important tool in the analysis of finite difference methods. They can be written as two slightly different form

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \cdots + \frac{h^k}{k!}u^{(k)}(x) + \cdots \tag{2.1}$$

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \cdots + \frac{h^k}{k!}u^{(k)}(\xi), \tag{2.2}$$

if $|h|$ is small enough. The second one sometimes is called the extended mean value theorem.

In a finite difference method, we need to substitute the derivatives with finite difference scheme to get an linear/non-linear algebraic system. There are several different ways to substitute the derivatives with finite difference formulas.

2.1 Forward, backward, and central finite difference formulas for $u'(x)$

Let first exam the first derivative $u'(x)$ of $u(x)$ at $\bar{x}$ using the nearby function values $u(\bar{x}\pm h)$, where h is called the step size. There are three commonly used formulas:

$$\text{Forward finite difference:} \qquad \Delta_+ u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} \sim u'(\bar{x}), \tag{2.3}$$

$$\text{Backward finite difference:} \qquad \Delta_- u(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x}-h)}{h} \sim u'(\bar{x}), \tag{2.4}$$

$$\text{Backward finite difference:} \qquad \delta u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h} \sim u'(\bar{x}). \tag{2.5}$$

2.1.1 Derivation of finite difference formulas from geometric intuition and calculus.

From calculus, we know that

$$u'(x) = \lim_{h\to 0} \frac{u(x+h) - u(x)}{h}$$

Assume h is small and $u'(x)$ is continuous, then if h is small, we expect that $\frac{u(x+h)-u(x)}{h}$ is very close to $u'(x)$. We can use it to approximate the first derivative

$$\frac{u(x+h) - u(x)}{h} \sim u'(x). \tag{2.6}$$

Note that usually they are not the same, so this step brings an error! In practice, h is the step size which is the distance between two grid points, so $h > 0$, we call the following formula as the ***forward finite difference***

$$\Delta_+ u(x) = \frac{u(x+h) - u(x)}{h}. \tag{2.7}$$

Geometrically, it is the slope of the secant line that connects the two points $(x, u(x))$ and $(x+h, u(x+h))$. How close is this formula to the derivative? We can use the extend mean value theorem (truncated Taylor expansion) to find it out. Assume that $u(x)$ has second order continuous derivatives. if h is small, then we have

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(\xi)h^2. \tag{2.8}$$

Therefore we can get an error estimate

$$E_f(h) = \frac{u(x+h) - u(x)}{h} - u'(x) = \frac{1}{2}u''(\xi)h = Ch = O(h). \tag{2.9}$$

So the error, defined as the difference of the approximate value and the exact one, is proportional to h, such discretization is called *first order accurate.* Generally, if the error has the form

$$E(h) = Ch^p, \qquad p > 0, \tag{2.10}$$

the method is called p-th order accurate.

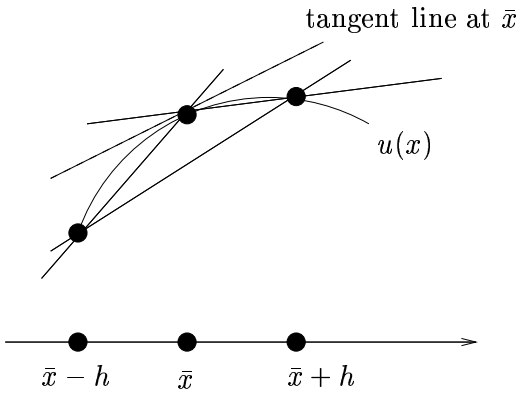

Figure 3: A geometric illustration of forward, backward, and central finite difference to approximate $u'(x)$.

Similarly, we can get and analyze the ***backward finite difference*** formula for approximating $u'(x)$ which is

$$\Delta_- u(x) = \frac{u(x) - u(x-h)}{h}, \quad h > 0. \tag{2.11}$$

It is easy to get

$$E_b(h) = \frac{u(x) - u(x-h)}{h} - u'(x) = -\frac{1}{2}u''(\xi)\, h = Ch \tag{2.12}$$

Therefore, the backward finite difference is also first order accurate.

Geometrically, we can see that the slope of the secant line that pass through $(x+h, u(x+h))$ and $(x-h, u(x-h))$ is a better approximation to the slope of the tangent line of $u(x)$. The formula

$$\delta u(x) = \frac{u(x+h) - u(x-h)}{2h}, \quad h > 0, \tag{2.13}$$

is called the *central finite difference* for the first order derivative.

In order to get a correct error estimate, we need to use more terms in the Taylor expansion

$$\begin{aligned} u(x+h) &= u(x) + hu'(x) + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + \frac{1}{24}u^{(4)}(x)h^4 + \cdots \\ u(x+h) &= u(x) - hu'(x) + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + \frac{1}{24}u^{(4)}(x)h^4 + \cdots \end{aligned}$$

Therefore, we have

$$E_c(h) = \frac{u(x+h) - u(x-h)}{2h} - u'(x) = \frac{1}{3}u'''(x)h^2 + h.o.t = Ch^2, \tag{2.14}$$

where $h.o.t$ stands for higher order terms. Note we have used a slightly different approach from that we used for forward and backward formulas. Therefore the central finite difference formula is second order accurate. It can be proved that the formula above can be written as

$$\frac{u(x+h) - u(x-h)}{2h} = \frac{1}{2}(\Delta_+ + \Delta_-)u(x)$$

A third order accurate finite difference formula for $u'(x)$ is

$$\delta_3 u(x) = \frac{2u(x+h) + 3u(x) - 6u(x-h) + u(x-2h)}{6h}. \tag{2.15}$$

2.1.2 Verification, and grid refinement analysis

Assume that we have learned or developed a numerical method and have had some analysis. We can write a compute code to implement the method. How do we know that our code is bug-free and our analysis is correct. One method is called the grid refinement analysis.

Typically, in the grid refinement analysis, we solve a problem in which we know the exact solution[2], we start with a fixed h, say $h = 0.1$, then decrease h by half to see how the error changes. For a first order method, the error should be decrease by factor two, and for a second order method, the error should be decrease by factor four. We can also plot the error versus h in $\log - \log$ scale. In the $\log - \log$ scale, the slope is the order of convergence.

[2]This is not always available. We will mention other techniques about how to validate a computed solution later.

Below we show the comparison of the forward, backward, and central finite difference formula in Matlab script file compare.m. We choose the test function as $u(x) = \sin x$ at $x = 1$, so the exact derivative is $\cos 1$. In Fig. 4, we plot the error versus h in $\log - \log$ scale in which the slopes do give correct indication of the convergence order.

```
% Compare truncation errors of the forward, backward, and central
% scheme for approximating u'(x). Plot the error and estimate the
% convergence order.
%    u(x) = sin(x) at x=1.   Exact u'(1) = cos(1)

clear; close all
  h = 0.1;
  for i=1:5,
    a(i,1) = h;
    a(i,2) = (sin(1+h)-sin(1))/h - cos(1);
    a(i,3) = (sin(1) - sin(1-h))/h - cos(1);
    a(i,4) = (sin(1+h)-sin(1-h))/(2*h)- cos(1);
    h = h/2;
  end

  format short e
  a % Display the result

  a = abs(a);    % Take absolute value of the matrix.
  h1 = a(:,1);    % Extract the first column which is h.
  e1 = a(:,2); e2 = a(:,3); e3 = a(:,4);

  loglog(h1,e1,h1,e2,h1,e3)
  axis('square')
  axis([1e-6 1e1 1e-6 1e1])
  gtext('Slope of FW and BW = 1')
  gtext('Slope of CD =2')

%%%%%%%%%%%%%%%% End Of Matlab Program %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Computed Results:

%      h          forward      backward     central
```

```
%   1.0000e-01  -4.2939e-02   4.1138e-02  -9.0005e-04
%   5.0000e-02  -2.1257e-02   2.0807e-02  -2.2510e-04
%   2.5000e-02  -1.0574e-02   1.0462e-02  -5.6280e-05
%   1.2500e-02  -5.2732e-03   5.2451e-03  -1.4070e-05
%   6.2500e-03  -2.6331e-03   2.6261e-03  -3.5176e-06
```

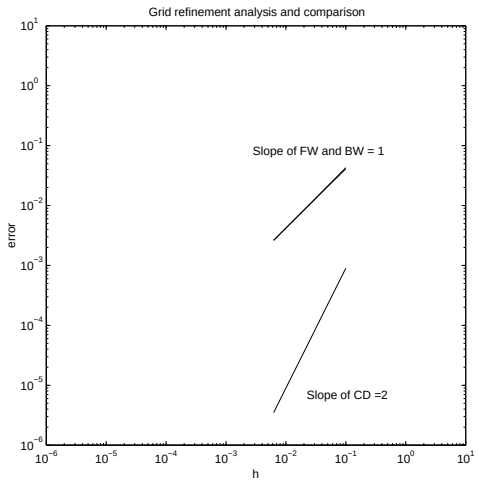

Figure 4: A plot of grid refinement analysis of the forward, backward, and central finite difference formula for $u'(x)$ using $log - log$ plot. The curves for forward and backward finite difference are almost identical and have slope one. The central formula is second order accurate and the slope of the plot is two.

2.2 Using the method of un-determined coefficients method to determine finite difference formula

Suppose we want to use one-sided finite difference to approximate $u'(x)$ at $\bar{x} = b$ which is a boundary using $u(\bar{x})$, $u(\bar{x}-h)$, $u(\bar{x}-2h)$ to second order accuracy. We can use the method of un-determined coefficients method. Let

$$u'\bar{x} \sim \gamma_1 u(\bar{x}) + \gamma_2 u(\bar{x}-h) + \gamma_3 u(\bar{x}-2h).$$

We can use the Taylor expansion at $\bar{x}$ to get a system of equations for the coefficient

$$\begin{aligned}
&\gamma_1 u(\bar{x}) + \gamma_2 u(\bar{x}-h) + \gamma_3 u(\bar{x}-2h) = \\
&\quad \gamma_1 u(\bar{x}) + \gamma_2 \left((u(\bar{x}) - hu'(\bar{x}) + \frac{h^2}{2}u''(\bar{x}) - \frac{h^3}{6}u'''(\bar{x}) \right) + \\
&\quad \gamma_3 \left((u(\bar{x}) - 2hu'(\bar{x}) + \frac{4h^2}{2}u''(\bar{x}) - \frac{8h^3}{6}u'''(\bar{x}) \right) + O(\max|\gamma_k| h^4)
\end{aligned}$$

The linear combination should approximate $u'(\bar{x})$. So we should set

$$\begin{aligned} \gamma_1 + \gamma_2 + \gamma_3 &= 0 \\ -h\gamma_2 - 2h\gamma_3 &= 1 \\ h^2\gamma_2 + 2h^2\gamma_3 &= 0 \end{aligned}$$

It is easy to check the solution is

$$\gamma_1 = -\frac{3}{2h}, \qquad \gamma_2 = \frac{2}{h}, \qquad \gamma_3 = -\frac{1}{2h}.$$

Therefore we get a one-sided finite difference scheme

$$u'(\bar{x}) = \frac{3}{2h}\,u(\bar{x}) - \frac{2}{h}\,u(\bar{x}-h) + \frac{1}{2h}\,u(\bar{x}-2h) + O(h^2). \tag{2.16}$$

Similarly we can get another one-sided finite difference formula by setting $h = -h$ in the formula above

$$u'(\bar{x}) = -\frac{3}{2h}\,u(\bar{x}) + \frac{2}{h}\,u(\bar{x}+h) - \frac{1}{2h}\,u(\bar{x}+2h) + O(h^2). \tag{2.17}$$

One can also differentiate a polynomials interpolation to get finite difference scheme. For example, given a sequence points $(x_i, u(x_i))$, $i = 0, 1, 2, \cdots, n$. Let the Lagrange interpolation polynomial be

$$p_n(x) = \sum_{j=0}^{n} l_j(x)u(x_j), \quad \text{where} \quad l_j(x) = \prod_{i\neq j}^{n} \frac{(x - x_j)}{(x_i - x_j)}.$$

Then $u'(\bar{x})$ can be approximated by

$$u'(\bar{x}) \sim p_n'(\bar{x}) = \sum_{j=0}^{n} l_j'(\bar{x})u(x_j).$$

2.3 Finite difference formula for second order and higher derivatives

We can apply the finite difference operator for the first order derivative twice to get finite difference formula for second order derivative $u''(\bar{x})$. For example, the central finite difference formula for second order derivative can be obtained from

$$\begin{aligned} \Delta_+\Delta_- u(\bar{x}) &= \Delta_+ \frac{u(x) - u(x-h)}{h} \\ &= \frac{1}{h}\left(\frac{u(x+h) - u(x)}{h} - \frac{u(x) - u(x-h)}{h}\right) \\ &= \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = u''(\bar{x}) + O(h^2) \\ &= \Delta_-\Delta_+ u(\bar{x}) = \delta^2 u(\bar{x}) = \delta_{xx}^2 u(\bar{x}). \end{aligned} \tag{2.18}$$

We will either use δ^2 if there is no confusion occurs or δ^2_{xx} is we want to specify in the x direction for the central finite difference formula for $u''(\bar{x})$. The central formula used very often for second order differential equations. If we use the same finite difference operator twice, we get a one-sided finite difference formula for second order derivative

$$\begin{aligned} \Delta_+\Delta_+u(\bar{x}) &= (\Delta_+)^2u(\bar{x}) = \Delta_+\frac{u(x+h)-u(x)}{h} \\ &= \frac{1}{h}\left(\frac{u(x+2h)-u(x+h)}{h} - \frac{u(x+h)-u(x)}{h}\right) \\ &= \frac{u(x)-2u(x+h)+u(x+2h)}{h^2} = u''(\bar{x}) + O(h) \end{aligned} \tag{2.19}$$

which is only first order accurate. Below is an example of a finite difference formula for a cross derivative,

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{u(x+h,y+h)+u(x-h,y-h)-u(x+h,y-h)-u(x-h,y+h)}{4h^2} \tag{2.20}$$

with a uniform step size in both directions.

2.3.1 Finite difference formula for third order derivatives

We can either apply the lower order finite difference formulas or use the method of undetermined coefficients method to obtain finite difference formulas for third order derivatives. For example, we have

$$\begin{aligned} \Delta_+\delta^2u(\bar{x}) &= \Delta_+\frac{u(\bar{x}-h)-2u(\bar{x})+u(\bar{x}+h)}{h^2} \\ &= \cdots \\ &= \frac{u(\bar{x}-h)+3u(\bar{x})-3u(\bar{x}+h)+u(\bar{x}+2h)}{h^3} \\ &= u'''(\bar{x}) + \frac{h}{2}u^{(4)}(\bar{x}) + \cdots \end{aligned}$$

So the finite difference formula is first order accurate. If we use the central formula below

$$\frac{-u(\bar{x}-2h)+2u(\bar{x}-h)-2u(\bar{x}+h)+u(\bar{x}+2h)}{2h^3} = u'''(\bar{x}) + \frac{h^2}{4}u^{(5)}(\bar{x}) + \cdots \tag{2.21}$$

we can have second order accuracy.

In practice, we seldom need more than fourth order derivatives. For high order differential equations, we can transfer them to first order systems.

3 Consistency, stability, convergence, and error estimates of finite difference methods

When we use a finite difference method to solve a differential equation, we need to know how accurate the approximate solution compared to the exact one.

3.1 Definition of the global error

Let $\mathbf{U} = [U_1, U_2, \cdots U_n]^T$ be the solution of the finite difference scheme (assume no round-off errors), and $\mathbf{u} = [u(x_1), u(x_2), \cdots, u(x_n)]$ be the exact solution at grid points. The global error vector is defined as $\mathbf{E} = \mathbf{U} - \mathbf{u}$. Naturally, we wish to give a smallest upper bound for the error vector. Usually we can use different norms

- The maximum norm or the infinity norm $\|\mathbf{E}\|_\infty = \max_i |e_i|$. Usually this is regarded as the strongest measurement. If error is large at one grid point, then the maximum norm is also large.
- The 1-norm is one of the average norms. It is defined as $\|\mathbf{E}\|_1 = \sum_i h_i|e_i|$ which is similar to the continuous space $\int |e(x)|dx$.
- The 2-norm is another average norm. It is defined as $\|\mathbf{E}\|_2 = (\sum_i h_i|e_i|^2)^{1/2}$ which is similar to the continuous space $(\int |e(x)|^2 dx)^{1/2}$.

Note they are slightly different from, and more appropriate than, the definition of vector norms in the one and two norms.

If $\|E\| \leq Ch^p$, $p > 0$, we can the finite difference method is ***p-th order accurate***. Naturally, we wish to have reasonably high order method while keep the computational cost low.

Definition 3.1 *A finite difference method is called* ***convergent*** *if* $\lim_{h\to 0} \|\mathbf{E}\| = 0$.

3.2 Definition of the local truncation error

Let $P(\frac{d}{dx})$ be the differential operator (take off u from the linear differential equation). Examples:

- Given $u''(x) = f(x)$, then $P(\dfrac{d}{dx}) = \dfrac{d^2}{dx^2}$, and $Pu = f$.
- If $P(\frac{d}{dx}) = \frac{d^3}{dx^3} + a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x)$, then $Pu = f$ stands for the differential equation: $u''' + au'' + bu' + cu = f(x)$,

Let P_h be the finite difference operator. For example, for the second order differential equation $u''(x) = f(x)$, one of the finite difference operator is

$$P_h u(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}.$$

More examples will be given later.

The local truncation error is defined as

$$T(x) = Pu - P_h u \tag{3.1}$$

Now that we use the exact solution in the definition above. For the differential equation $u''(x) = f(x)$ and the three-point central difference scheme, the local truncation error is

$$T(x) = Pu - P_h u = u''(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \tag{3.2}$$

$$= f(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}. \tag{3.3}$$

Note that the local truncation error only depends on the solution in the finite difference stencil (it is three-point in the example) but do not depends on the solution globally (far away). That is one of reasons that is called the *local* truncation error. The local truncation errors measure how well the finite difference discretization approximates the differential equation.

Definition 3.2 *A finite difference scheme is called* ***consistent*** *if*

$$\lim_{h\to 0} T(x) = \lim_{h\to 0}(Pu - P_h u) = 0 \tag{3.4}$$

Usually, we should use a consistent finite difference scheme at least for most of grid points.

If $T(x) = Ch^p$, $p > 0$, then we say the discretization is p-th order accurate, where C is a constant independent of h but may depends on the solution $u(x)$. To check whether a finite difference scheme is consistent or not, we usually use Taylor expansion. For example, for the three point central finite difference scheme for $u''(x) = f(x)$, we have

$$T(x) = u''(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = -\frac{h^2}{12}u^{(4)}(x) + h.o.t. = Ch^2$$

where $C = \frac{1}{12}u^{(4)}(x)$. Therefore the finite difference scheme is consistent and the discritization is second order accurate.

Now let us exam another finite difference scheme for $u''(x) = f(x)$. The finite difference scheme is

$$\frac{u(x_i) - 2u(x_{i+1}) + u(x_{i+2})}{h^2} = f(x_i), \quad i = 1, 2, \cdots, n-2,$$

$$\frac{u(x_{n-2}) - 2u(x_{n-1}) + u(b)}{h^2} = f(x_{n-1})$$

The local truncation error at x_{n-1} satisfies $T(x_{n-1}) = Ch^2$. The local truncation error at other grid points

$$T(x_i) = u''(x) - \frac{u(x_i) - 2u(x_{i+1}) + u(x_{i+2})}{h^2} = Ch$$

So the finite difference scheme is consistent. But if we implement the finite difference scheme, you may get weird results.

So the consistence can not guarantee a finite difference work. We need another condition to determine whether a finite difference method converge or not. Such a condition is called the *stability* of a finite difference method.

For the model problem, we have

$$A\mathbf{u} = \mathbf{F} + \mathbf{T}, \qquad A\mathbf{U} = \mathbf{F}, \qquad A(\mathbf{u} - \mathbf{U}) = \mathbf{T} = -A\mathbf{E}, \tag{3.5}$$

where A is the coefficient matrix of the finite difference equations, $\mathbf{F}$ is the modified source term by the boundary condition, and $\mathbf{T}$ is the vector of the local truncation error at grid points.

If A is non-singular, then $\|\mathbf{E}\| = \|A^{-1}\mathbf{T}\| \leq \|A^{-1}\|\|\mathbf{T}\|$. If A is singular, then $\|E\|$ may be arbitrary large like the example above. The finite difference method does not converge. For the central finite difference scheme, we have $\|\mathbf{E}\| \leq \|A^{-1}\|h^2$. So the global error depends on both the local truncation error and $\|A^{-1}\|$.

Definition 3.3 *A finite difference method for the elliptic differential equation is stable if A is invertible and*

$$\|A^{-1}\| \leq C, \quad \textit{for all } h < h_0, \tag{3.6}$$

where C and h_0 are constants.

Theorem 3.1 *A consistent and stable finite difference method is convergent.*

Usually it is relatively easier to prove the consistency, but it is more difficult, sometime impossible, to prove the stability.

Now we are ready to prove the convergence of the central finite difference scheme for $u''(x) = f(x)$. We need the following lemma.

Lemma 3.1 *Given a symmetric tridiagonal matrix $A \in R^{n\times n}$ whose main diagonals and off-diagonals are two constant, d and α. Then the eigenvalues of A are*

$$\lambda_j = d + 2\alpha\cos\left(\frac{\pi j}{n+1}\right), \quad j = 1, 2, \cdots n, \tag{3.7}$$

each with corresponding orthogonal eigenvector

$$x_k^j = \sin\left(\frac{\pi kj}{n+1}\right), \quad k = 1, 2, \cdots n, \tag{3.8}$$

The Lemma can be proved by direct verification $A\mathbf{x}^j = \lambda_j\mathbf{x}^j$.

Theorem 3.2 *The central finite difference method for $u''(x) = f(x)$ with Dirichlet boundary condition is second order accurate.*

Proof: Assume the matrix is $A \in R^{(n-1)\times(n-1)}$, then we have $d = -2/h^2$, $\alpha = 1/h^2$, the eigenvalues of A is

$$\lambda_j = -\frac{2}{h^2} + \frac{1}{h^2}\cos\left(\frac{\pi j}{n}\right) = \frac{2}{h^2}\left(\cos(\pi j h) - 1\right)$$

Note that the eigenvalues of A^{-1} are $1/\lambda_j$ and A^{-1} is also symmetric, we have

$$\|A^{-1}\|_2 = \frac{1}{\min|\lambda_j|} = \frac{h^2}{2(1-\cos(\pi h))} = \frac{h^2}{2(1-(1-(\pi h)^2/2 + (\pi h)^4/4! + \cdots)} < \frac{1}{\pi^2}.$$

Therefore, we have

$$\|\mathbf{E}\|_2^{vec} \le \|A^{-1}\|_2 \|\mathbf{T}\|_2 \le \frac{1}{\pi^2}\sqrt{n-1}Ch^2 \le \bar{C}h^{3/2}$$

in the vector 2-norm. In the average 2-norm, we have $\|E\|_2 = \sqrt{h}\|E\|_2^{vec} \le \bar{C}h^2$. We can also prove that the infinity norm is also proportional to h^2 using the maximum principal.

Remark 3.1 *The eigenvectors and eigenvalues of the coefficient matrix in (1.11) can also be obtained by considering the eigenvalue problem of the Sturm-Luiville problem*

$$u''(x) + \lambda u = 0, \quad u(0) = u(1) = 0. \tag{3.9}$$

It is easy to check that the eigenvalues are

$$\lambda_k = (k\pi)^2, \quad k = 1, 2, \cdots. \tag{3.10}$$

The corresponding eigenvectors are

$$u_k(x) = \sin(k\pi x), \tag{3.11}$$

Its discrete form at grid point is

$$u_k(x_i) = \sin(k\pi i h), \quad i = 1, 2, \cdots, n-1. \tag{3.12}$$

This is one of the eigenvectors of the coefficient matrix in (1.11). The corresponding eigenvalue can be found using the definition $A\mathbf{x} = \lambda\mathbf{x}$.

3.3 The effect of round-off errors and the choice of machine precision.

From the knowledge of numerical linear algebra, we know that

- $\|A\|_2 = \max|\lambda_j| = \frac{2}{h^2}\left(1 - \cos(\pi(n-1)h)\right) \sim \frac{4}{h^2} = 4n^2$. Therefore $\kappa(A) = \|A\|_2\|A^{-1}\|_2 \sim n^2$.

- The relative error of the computed solution $\mathbf{U}$ for a stable scheme satisfies

$$\begin{aligned}\frac{\|\mathbf{U}-\mathbf{u}\|}{\|\mathbf{u}\|} &\leq \text{local truncation error + round-off error}\\ &\leq \|A^{-1}\|\|\mathbf{T}\| + \bar{C}g(n)\|A\|\|A^{-1}\|\,\epsilon\\ &\leq Ch^2 + \bar{C}g(n)\frac{1}{h^2}\,\epsilon\end{aligned}$$

where $g(n) \sim n$ is the growth factor of the algorithm for solving the linear system of equations, ϵ is the machine precision. For several type of computers, $\epsilon \sim 10^{-8}$ when we use the single precision, and $\epsilon \sim 10^{-16}$ for the double precision. To make the discussion simple, assume that $C = \bar{C} = g(n) = 1$, the smallest error that we can expect is when the truncation error is roughly the same as the round-off error

$$h^2 \sim \frac{1}{h^2}\epsilon, \quad \Longrightarrow \quad n \sim \frac{1}{h} = \frac{1}{\epsilon^{1/4}}$$

which is about 100 for the single precision, and $10,000$ for the double precision. Therefore, if we use single precision, there is no point to take more than 100 grid points, and we can at best get roughly four significant digits. That is why we usually use double precision to solve ODE/PDEs. Note that Matlab is using double precision.

4 Finite difference methods for 1-D elliptic equations.

4.1 Finite difference methods for 1-D self adjoint elliptic equations.

Consider the one dimensional self adjoint elliptic equations of the form

$$\big(p(x)u'(x)\big)' - q(x)u(x) = f(x), \quad a < x < b, \tag{4.1}$$

$$u(a) = u_a, \quad u(b) = u_b, \quad \text{or other BC.} \tag{4.2}$$

The existence and uniqueness of the solution is given in the following theorem.

Theorem 4.1 *If $p(x)$ and $q(x)$ are continuous, and $f(x) \in L^(a,b)$, that is $\int_a^b f^2 dx < \infty$, $q(x) \geq 0$, and there is a positive constant such that $p(x) \geq p_0 > 0$, then there is unique solution $u(x) \in C^2(a,b)$.*

The proof is usually given in a course of differential equations. Now let us discuss the finite difference method.

Step 1: Generate a grid. For simplicity, we use a uniform Cartesian grid here

$$x_i = a + ih, \quad h = \frac{b-a}{n}, i = 0, 1, \cdots n.$$

Note that $x_0 = a$, $x_n = b$. Sometimes an adaptive grid may be preferred, however, we can not use the central finite difference scheme for adaptive grids.

Step 2: Substitute derivatives with finite difference at each grid point that the solution is unknown. This is step is also called discretization. Define $x_{i+\frac{1}{2}} = x_i + h/2$. We have $x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} = h$. Use the central finite difference formula at a typical grid point x_i with half grid size, we can get

$$\frac{p_{i+\frac{1}{2}}u'(x_{i+\frac{1}{2}}) - p_{i-\frac{1}{2}}u'(x_{i-\frac{1}{2}})}{h} - q_i u(x_i) = f(x_i) + E_i^1$$

where $p_{i+\frac{1}{2}} = p(x_{i+\frac{1}{2}})$, $q_i = q(x_i)$, $f_i = f(x_i)$, and $E_i^1 = Ch^2$. Apply the central finite difference scheme for the first order derivative further, we get

$$\frac{p_{i+\frac{1}{2}}\frac{u(x_{i+1})-u(x_i)}{h} - p_{i-\frac{1}{2}}\frac{u(x_i)-u(x_{i-1})}{h}}{h} - q_i u(x_i) = f(x_i) + E_i^1 + E_i^2, \quad i = 1, 2, \cdots n-1.$$

If we ignore the local truncation error and substitute $u(x_i)$ with U_i, we get a linear system of equations of U_i which is a finite difference approximation to the exact solution $u(x_i)$:

$$\frac{p_{i+\frac{1}{2}}U_{i+1} - (p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}})U_i + p_{i-\frac{1}{2}}U_{i-1}}{h^2} - q_i U_i = f_i, \quad i = 1, 2, \cdots n-1. \tag{4.3}$$

In matrix-vector form, the linear system above can be written as $A\mathbf{U} = \mathbf{F}$, where

$$A = \begin{bmatrix} -\frac{p_{1/2}+p_{3/2}}{h^2} - q_1 & \frac{p_{3/2}}{h^2} & & & \\ \frac{p_{3/2}}{h^2} & -\frac{p_{3/2}+p_{5/2}}{h^2} - q_2 & \frac{p_{5/2}}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & & \frac{p_{n-3/2}}{h^2} & -\frac{p_{n-3/2}+p_{n-1/2}}{h^2} - q_{n-1} \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} f(x_1) - \frac{p_{1/2}u_a}{h^2} \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \frac{p_{n-1/2}u_b}{h^2} \end{bmatrix}$$

It is important to note that A is symmetric, and negative definite. A is also weakly weakly diagonally dominant and it is an M-matrix. Those properties guarantee that A is non-singular.

Question: The differential equation can be written as

$$p(x)u'' + p'(x)u' - qu = f(x).$$

This is called a non-conservative form. We can directly apply second order finite difference formulas to the equation above. What are advantages and dis-advantages of this approach?

- We need to find the derivative, or its approximation, of $p(x)$.
- The coefficient matrix of the finite difference equations is not symmetric, or negative positive definite, or diagonally dominant anymore.

Therefore we should avoid such discretization if possible.

The local truncation error. The local truncation error is

$$T_i = \frac{p_{i+\frac{1}{2}}u(x_{i+1}) - (p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}})u(x_i) + p_{i-\frac{1}{2}}u(x_{i-1})}{h^2} - q_i u(x_i) - f_i, \tag{4.4}$$

Note that $f_i = (pu')' - qu$ is the differential operator. It is easy to show that $T_i = Ch^2$. It is more difficult to show that $\|A^{-1}\| \leq C$. However, we can use the maximum principal to prove second order convergence of the finite difference scheme.

4.2 Finite difference methods for general 1D elliptic equations.

Consider the differential equation

$$p(x)u''(x) + r(x)u'(x) - q(x)u(x) = f(x), \quad a < x < b, \tag{4.5}$$

$$u(a) = u_a, \quad u(b) = u_b, \quad \text{or other BC.} \tag{4.6}$$

If $p(x) = 1$, $r(x) = 0$, and $q(x) \leq 0$, such an equation is called Helmholtz equation which is difficult to solve if $q(x) \leq 0$ and $|q(x)|$ is large, say $q(x) \sim 1/h^2$.

There are two different discretization technique that we can use.

- Central finite difference scheme for all the derivatives. The finite difference scheme is

$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_{i+1} - U_{i-1}}{2h} - q_i U_i = f_i, \quad i = 1, 2, \cdots n-1. \tag{4.7}$$

  The advantage of this discretization is that the method is second order accurate. The disadvantage is that the coefficient matrix may not be diagonally dominant even if $q(x) > 0$, $p(x) > 0$. The term $r(x)u'(x)$ sometimes is called the advection term if u is the velocity. When the advection is strong, that is $|r(x)|$ is large, the equation behaves like a wave equation.

- A mixed scheme: the central finite difference scheme for diffusion term and the *upwind* scheme for advection term.

$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_{i+1} - U_i}{h} - q_i U_i = f_i, \text{ if } r_i \geq 0,$$
$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_i - U_{i-1}}{h} - q_i U_i = f_i, \text{ if } r_i < 0.$$

The purpose is to increase the diagonally dominance. It is first order accurate. It may be recommended if $|r(x)|$ is very large, say $|r(x)| \sim 1/h$. It is easier to solve the linear system of equations with either direct method or iterative methods.

4.3 Ghost point method for boundary condition involve the derivatives.

In this sub-section, we discuss the treatment of the Neumann and Robin, or mixed, boundary conditions. First we consider the differential equation

$$u''(x) = f(x), \quad a < x < b,$$
$$u'(a) = uxa, \qquad u(b) = ub$$

Note that the solution at $x = a$ is unknown. If we use a uniform Cartesian grid $x_i = a + ih$, U_0 is one of component of the solution. We still can use

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = f_i, \quad i = 1, 2, \cdots n_1.$$

But we need an additional equation at $x_0 = a$. Certainly we should use the Neumann boundary condition at $x = a$. One intuitive approach is

$$\frac{U_1 - U_0}{h} = uxa, \quad \text{or} \quad \frac{-U_0 + U_1}{h^2} = \frac{uxa}{h}. \tag{4.8}$$

It works and the linear system of equations is still tri-diagonal and symmetric positive definite:

$$\begin{bmatrix} -\frac{1}{h^2} & \frac{1}{h^2} & & & & \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & & \\ & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ & & & & \frac{1}{h^2} & -\frac{2}{h^2} \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{uxa}{h} \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \frac{u_b}{h^2} \end{bmatrix} \tag{4.9}$$

But the global error is only first order accurate even though some people do not believe it.

To keep the second order accuracy, we use the ghost point method by adding a ghost grid point $x_{-1} = x_0 - h = a - h$ and assume that the solution is extended to the interval $[a-h, a]$ as well. Now we can use the central finite difference scheme for the differential equation at all grid points that the solution is unknown, that is, $i = 0, 1, \cdots n$. We have n equations and $n+1$ unknowns. The extra one is U_1. We need one more equation to close the system. The additional equation is the central finite difference equation for the Neumann boundary condition:

$$\frac{U_1 - U_{-1}}{2h} = uxa. \tag{4.10}$$

From the equation above, we can get $U_{-1} = U_1 - 2huxa$. Plug this into the finite difference equation for the differential equation at $x = a$, we have

$$\begin{aligned}
\frac{U_{-1} - 2U_0 + U_1}{h^2} &= f_0 \\
\frac{U_1 - 2huxa - 2U_0 + U_1}{h^2} &= f_0 \\
\frac{-U_0 + U_1}{h^2} &= \frac{f_0}{2} + \frac{uxa}{h}
\end{aligned}$$

The coefficient matrix of the finite difference equation is the exact the same as (4.9). The only difference is the first component in the right hand side vector. Now the component is $f_0/2 + uxa/2$ compared with $uxa/2$ of the first order method. But the results obtained from the ghost point method is much better, second order method versus first order method.

In order to discuss the stability, we need to find the eigenvalues of the coefficient matrix. Again, we can associate the eigenvector to the continuous problem:

$$u'' + \lambda u = 0, \quad u'(0) = 0, \quad u(1) = 0. \tag{4.11}$$

It is easy to show that the eigenvectors are

$$u_k(x) = \cos\left(\frac{\pi x}{2} + k\pi x\right) \tag{4.12}$$

corresponding to the eigenvalue $\lambda_k = (\pi/2 + k\pi)^2$.

We compare the two methods in Fig. 5. The differential equation is $u''(x) = f(x)$ with a Dirichlet boundary condition at $x = 0$, and a Neumann boundary condition at $x = 0.5$. The exact solution is $u(x) = \cos(\pi x)$. Fig. 5 (a) shows the grid refinement analysis. The error in the second order method is much smaller that that of the first order method. In Fig. 5 (b), we show the error plot of the ghost point method. Note that the error at $x = b$ is not zero anymore.

4.4 A Matlab code of the ghost point method.

```
function [x,U] = two_pointa(a,b,ua,uxb,f,n)
```

(a) (b)

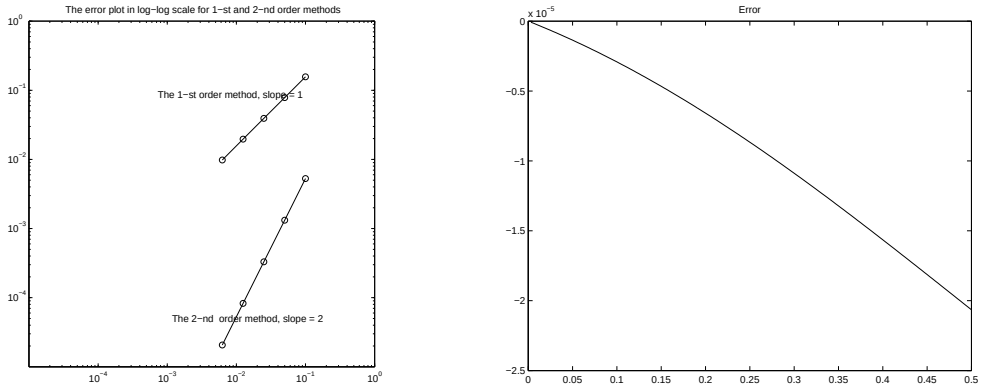

Figure 5: (a) The grid refinement analysis of the ghost point method and the first order method. The slopes of the curves are the order of convergence. (b) The error plot of the computed solution from the ghost point method.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This matlab function two_point solvers the following two-point    %
%    boundary value problem: u''(x) = f(x) using center difference     %
%    scheme.                                                           %
%    Input:                                                            %
%     a, b: Two end points.                                            %
%     ua, uxb: Dirichlet and Neumann boundary conditions at a and b    %
%     f: external function f(x).                                       %
%     n: number of grid points.                                        %
%    Output:                                                           %
%     x: x(1),x(2),...x(n-1) are grid points                           %
%     U: U(1),U(2),...U(n) are approximate solution at grid points.    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = (b-a)/n; h1=h*h;

A = sparse(n,n);
F = zeros(n,1);

for i=1:n-1,
  A(i,i) = -2/h1; A(i+1,i) = 1/h1; A(i,i+1)= 1/h1;
end
  A(n,n) = -2/h1;
```

```
  A(n,n-1) = 2/h1;

for i=1:n,
  x(i) = a+i*h;
  F(i) = feval(f,x(i));
end
  F(1) = F(1) - ua/h1;
  F(n) = F(n) - 2*uxb/h;

U = A\F;

return
```

4.5 The Matlab drive program

```
%%%%%%% Clear all unwanted variable and graphs.

clear;  close all

%%%%%%% Input

a =0; b=1/2;
ua = 1; uxb = -pi;

%%%%%% Call solver: U is

n=10;
k=1;

for k=1:5
    [x,U] = two_pointa(a,b,ua,uxb,'f',n);  %ghost-point method.
%    [x,U] = two_pointb(a,b,ua,uxb,'f',n);  %Backward Difference
  u=zeros(n,1);
  for i=1:n,
    u(i) = cos(pi*x(i));
  end

  h(k) = 1/n;
  e(k) = norm(U-u,inf); %%% Print out the maximum error.
```

```
  k = k+1; n=2*n;
end

loglog(h,e,h,e,'o'); axis('equal'); axis('square'),
%title('The error plot in log-log scale, the slope = 2'); % Ghost point
title('The error plot in log-log scale, the slope = 1'); % BW
figure(2); plot(x,U-u); title('Error')
```

4.6 Mixed boundary condition.

The ghost point method can be used for the mixed boundary condition, also called Robin boundary condition in some literature. Assume that at $x = a$, we have $\alpha u'(a) + \beta u(b) = \gamma$, where $\alpha \neq 0$. Then we can use the ghost point method to discretize the boundary condition

$$\alpha \frac{U_1 - U_{-1}}{2h} + \beta U_0 = \gamma$$

$$\text{or} \qquad U_{-1} = U_1 + \frac{2\beta h}{\alpha} U_0 - \frac{2h\gamma}{\alpha}$$

Plugging this into the central finite difference equation at $x = x_0$ we get

$$\left(-\frac{2}{h^2} + \frac{2\beta}{\alpha h}\right) U_0 + \frac{2}{h^2} U_1 = f_0 + \frac{2\gamma}{\alpha h}, \tag{4.13}$$

or

$$\left(-\frac{1}{h^2} + \frac{\beta}{\alpha h}\right) U_0 + \frac{1}{h^2} U_1 = \frac{f_0}{2} + \frac{\gamma}{\alpha h}, \tag{4.14}$$

to get a symmetric coefficient matrix.

5 An example of a non-linear boundary value ODE.

By substituting the derivatives in a non-linear differential equation, we will have a non-linear algebraic system of equations. If we can solve the non-linear system, then we can get an approximate solution to the non-linear differential equation. We present an example in this section to illustrate the procedure.

Consider the motion of a pendulum with mass M at the end of a rigid (but massless) bar of length L. Let $\theta(t)$ be the angle of the pendulum from vertical at time t, as illustrated in Fig. 6. Ignoring the mass of the bar and forces of the friction and air resistance, the differential equation for the pendulum motion can be well approximated by

$$M\mathbf{a} = -Mg \sin\theta,$$

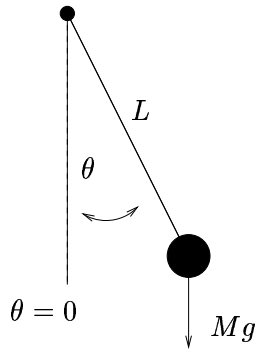

Figure 6: A diagram of a pendulum.

where g is the gravitation constant. Since $S = L\theta$, where S is the arc, so we have $\mathbf{v} = L\dot{\theta}$, and $\mathbf{a} = L\ddot{\theta}$, where $\mathbf{v}$ is the velocity, $\mathbf{a}$ is the acceleration. Therefore we have the following non-linear boundary value:

$$\ddot{\theta} = -\frac{g}{L}\sin\theta = K\sin\theta, \quad K = \frac{g}{L}. \tag{5.1}$$

with two reasonable boundary conditions $\theta(0) = \alpha$, say $\alpha = \pi/4$, and the observed angle at some time later $\theta(T) = \beta$. If θ is very small, then we can use the approximation $\sin\theta = \theta + O(\theta^3)$ to get a linear differential equation. However, we can apply the central finite difference scheme directly to get a system of non-linear equations.

$$\frac{\theta_{i-1} - 2\theta_i + \theta_{i+1}}{h^2} + K\sin\theta_i = 0, \quad i = 1, 2, \cdots, n-1, \tag{5.2}$$

where $h = T/n$ is the step size.

There are several ways to solve the non-linear problems. We explain the ideas briefly for the pendulum problem.

- Use a linear model to approximate the non-linear problem. For example, we can approximate $\theta'' + K\sin\theta = 0$ using the linear equation $\theta'' + K\theta = 0$. This is valid if θ is small.

- Use a substitution method in which we approximate the non-linear term with the previous approximation. Given an initial guess $\theta^{(0)}(x)$, we form the iteration

$$(\theta^{k+1})'' + K\sin\theta^k = 0, \quad k = 0, 1, \cdots \tag{5.3}$$

  The main concern is when the iterative method convergence or not, and the rate of the convergence.

- Solve the non-linear problem directly which is explained below.

Usually we will get a non-linear system of equations if we use finite difference method to discretize a non-linear ODE/PDE

$$\begin{cases} F_1(U_1, U_2, \cdots, U_n) = 0, \\ F_2(U_1, U_2, \cdots, U_n) = 0, \\ \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ F_n(U_1, U_2, \cdots, U_n) = 0. \end{cases} \tag{5.4}$$

A commonly used method is Newton's method. Given an initial guess $\mathbf{U}^{(0)}$, we form the Newton's iteration

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} - (J(\mathbf{U}^{(k)}))^{-1}\mathbf{F}(\mathbf{U}^{(k)}) \tag{5.5}$$

or

$$\begin{cases} J(\mathbf{U}^{(k)})\Delta\mathbf{U}^{(k)} = -\mathbf{F}(\mathbf{U}^{(k)}) \\ \mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \Delta\mathbf{U}^{(k)}, \end{cases} \quad k = 0, 1, \cdots$$

where $J(\mathbf{U}^{(k)})$ is the Jacobi matrix defined as

$$\begin{bmatrix} \frac{\partial F_1}{\partial U_1} & \frac{\partial F_1}{\partial U_2} & \cdots & \frac{\partial F_1}{\partial U_n} \\ \frac{\partial F_2}{\partial U_1} & \frac{\partial F_2}{\partial U_2} & \cdots & \frac{\partial F_2}{\partial U_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial U_1} & \frac{\partial F_n}{\partial U_2} & \cdots & \frac{\partial F_n}{\partial U_n} \end{bmatrix}$$

For the pendulum problem, we have

$$J(\theta) = \frac{1}{h^2}\begin{bmatrix} -2 + h^2K\cos\theta_1 & 1 & & & \\ 1 & -2 + h^2K\cos\theta_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 + h^2K\cos\theta_{n-1} & \end{bmatrix}$$

It is not always easy to find $J(\mathbf{U})$ and it can be very computationally expensive. Further more, Newton's method is called local convergent method because it requires that the initial guess is close to the solution to guarantee the convergence. Many new methods, for example, quasi-Newton type method, such as Broyden and BFGS rank-one and rank-two update, conjugate gradient methods, trust region methods, etc. The main issues include global convergence, the convergence order (Newton's method is quadratically convergent locally), storage etc. A well known software package is MINPACK which is available through netlib, see [**?**] for more complete discussions.

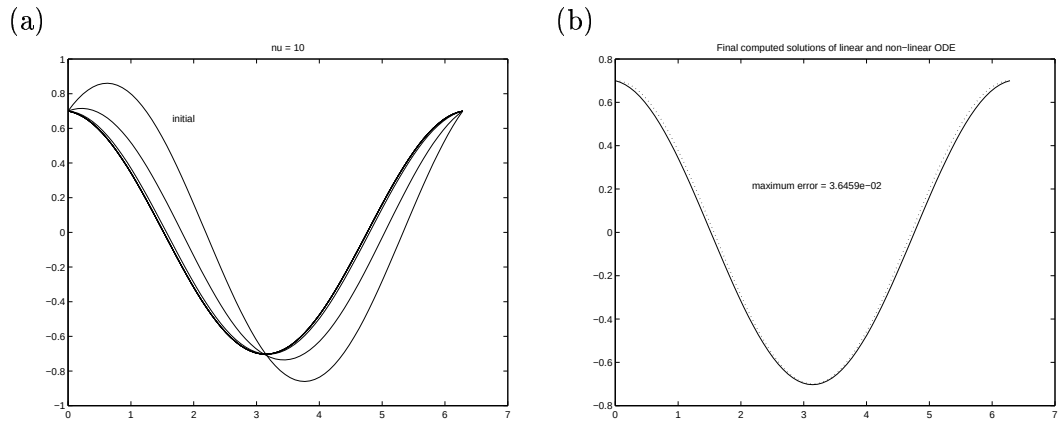

Figure 7: (a) Plot of some intermediate solution to the non-linear system of equations. (b) Comparison of the solution of the linear and non-linear solution to the pendulum motion.

5.1 The grid refinement analysis technique.

After we have learned or developed a numerical method, which include convergence analysis such as consistence, stability, and order of convergence, computational complexity (operation counts, storage and other issues) etc. We need to validate the method and numerically confirm the analysis. We can also find out the algorithm behavior through numerically results. There are same ways to serve these purposes.

- Analyze of the output. Examine the boundary conditions, maximum/minimum, etc.
- Compare with experiential data with sufficient variation of parameters.
- Do the grid refinement analysis with or without exact solutions.

Now we explain the grid refinement analysis with exact solutions. Assume a method is p-th order accurate, meaning that $\|E_h\| \sim Ch^p$ is h is small enough. We can divide h by half to get $\|E_{h/2}\|$. Then we have the following relations

$$\text{ratio} = \frac{\|E_h\|}{\|E_{h/2}\|} = \frac{Ch^p}{C(h/2)^p} = 2^p, \tag{5.6}$$

$$\log \|E_h\| = \log C + p \log h, \quad \text{and} \quad p = \frac{\log \left(\|E_h\| / \|E_{h/2}\| \right)}{\log 2}. \tag{5.7}$$

For a first order method ($p = 1$), the ratio approaches number two, and p approach number one. For a second order method ($p = 2$), the ratio approaches number four, and p approach number two, and so on. If p is some number between one and two, the method is called super-linear convergent.

For simple problems, we may come up with exact solution easily. For example, for all the linear single ODE/PDEs, we can simply set an exact solution $u_e(\mathbf{x})$, and from the

exact solution, we can determine other functions and parameters such as the source term $f(\mathbf{x}$, boundary and initial conditions etc. For more complicated system of ODE/PDEs, the exact solution sometimes is hard to construct. We can search for the literature to see whether there are similar examples. Some examples in the literature will become bench mark problems. New methods may have to compare the results of the bench mark problems before they can be published.

For one dimensional problems, we can take $n = 10, 20, 40, \cdots, 640$, depending on the size of the problem and the speed of the computers, to do the grid refinement analysis. For two dimensional problems, we can take $(10, 10)$, $(20, 20)$, $\cdots$, $(320, 320)$ to do the grid refinement analysis, for three dimensional problems, we can take $(10, 10, 10)$, $(20, 20, 20)$, $\cdots$, $(160, 160, 160)$ to do the grid refinement analysis if we have enough memory.

To present the grid refinement analysis, we can tabulate the grid size n, the ratio and/or order so that we can see the results at the first glance. The other way is to plot the error versus the step size h in $\log - \log$ scale, with the same scale on both x- and y- axis, the slope of the approximate line is the order of convergence.

6 Finite difference methods for two dimensional second order elliptic problems

There are many important applications of elliptic partial differential equations. Below are some examples:

- Laplace equation.

$$u_{xx} + u_{yy} = 0. \tag{6.1}$$

- Poisson equation.

$$u_{xx} + u_{yy} = f. \tag{6.2}$$

  The solution to a Laplace or Poisson equation sometimes is called a potential equation. This is because for a vector field $\mathbf{v}$ that is conservative meaning that $curl(\mathbf{v}) = \mathbf{0}$, its potential function u satisfies $u_{xx} + u_{yy} = \nabla \cdot \mathbf{v}$, where $\nabla = [\, \frac{\partial}{\partial x},\ \frac{\partial}{\partial y}\,]^T$ is the gradient operator.

- Helmholtz equation.

$$u_{xx} + u_{yy} + \lambda u = f. \tag{6.3}$$

  If $\lambda < 0$, it is called generalized Helmholtz equation and it is easy to solve. If $\lambda > 0$ is big, then the problem is hard to solve.

- General self-adjoint elliptic PDEs

$$\nabla \cdot a(x,y)\nabla u(x,y) + \lambda(x,y)u = f(x,y), \tag{6.4}$$

$$\text{or} \quad (au_x)_x + (au_y)_y + \lambda(x,y)u = f(x,y), \tag{6.5}$$

  We should assume that $a(x,y)$ does not change the sign in the solution domain.

- General elliptic PDEs (diffusion and advection equations)

  $a(x,y)u_{xx} + 2b(x,y)u_{xy} + c(x,y)u_{yy} + d(x,y)u_x + e(x,y)u_y + g(x,y)u(x,y) = f(x,y)$

  It can be re-written as

$$\nabla \cdot a(x,y)\nabla u(x,y) + \mathbf{w}(x,y) \cdot \nabla u + c(x,y)u = f(x,y) \tag{6.6}$$

  after some transformation, where $\mathbf{w}(x,y)$ is a vector.

Below are some examples of non-linear elliptic PDEs.