*(a) Use one of the methods described in class to derive a one-sided, second order accurate, finite difference expression(s) for the first derivative that could be used at the boundaries x = 0 and x = 1 of the domain.*

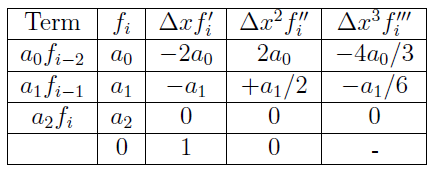
The first derivative can be approximated with a second order accurate one-sided truncated Taylor series as:

*df/dx = a2fi + a1fi-1 + a0fi-2 =*

*(a0+a1+a2)fi + (-2a0-a1)Δxfi’ + (2a0+a1/2)Δx²fi’’ + (-4a0/3-a1/6*) Δx³fi’’’ + Error *(1)*

Using a Taylor table to determine term coefficients:

Table 1: Determination of first derivative [1]



For the first derivative the following must be satisfied:

*a0+a1+a2=0*

*-2a0-a1=1*

*2a0+a1/2=0*

Solving yields: a0 = 0.5, a1 = -2, and a2 = 3/2. Substituting the coefficients back into Eqn. 1 and solving for df/dx yields:

*df/dx = (fi-2 – 4fi-1 + 3fi)/(2Δx) + O(Δx²) (2)*

This is usable at x=1, however a forward difference version has to be used at x=0:

*df/dx = (-fi+2 + 4fi+1 - 3fi)/(2Δx) + O(Δx²) (3)*

*(b) Write a pseudo code (a code layout/design) to solve the above PDE.*

Define number of nodes, length of bar, and step size

Create vector of nodal locations

Create “A” and “F” matrices

Populate A matrix with finite difference solution to second derivative

Populate F matrix with heat flux

Multiply A matrix by 1/Δx²

Set boundary conditions (T=0 @ x=0 and dT/dx=0 at x=1)

Solve matrices

Plot results

*(c) Implement your own version of a finite difference solver for the above equation using your pseudo code as a code outline. Set the temperature T(x) = 0 at x = 0 (Dirichlet boundary condition) and the first derivative of the temperature dT/dx = 0 at x = 1 (Neumann boundary).*

See Appendix A for program code in full.

For the case where f(x) = 1 and N=16, the following solution is obtained



Figure 1: Numerical solution (red) with analytical solution (green)

*(d) Derive the analytical solution to the above equation for these boundary conditions.*

Applying boundary conditions T=0 @ x=0 and dT/dx=0 @ x=1 yields:

A=-1, B=0

Therefore:

**T=0.5x²-x**

The analytical solution is plotted along side the numeric solution in Fig. 1.

*(e) Using your code from part c, determine the convergence rate of the maximum of the solution. Plot the error as a function of grid size, Δx (use an appropriate log plot to illustrate convergence).*

Neglecting computational errors (roundoff, quantization error from floating point math, etc) the inf-norm error for the case where f(x) = 1 is trivially zero. The higher order terms in the truncated portion of the Taylor series expansion contain differentiated terms of a higher order than the order of the analytical solution which is a polynomial. This means the truncated terms are therefore zero and the numerical solution is identical to the analytical one.

In order to fully investigate convergence of the numerical solution an infinitely differentiable forcing function could be used so that all of the truncated terms are not trivially zero. The case of f(x) = sin(x) was examined and is plotted in Fig. 2. The slope of the log-log plot is annotated over the line.

The magnitude of the slope is 2.0182; This is quite close to the expected value of 2, the exponent on the lowest order (and dominating) truncated term from the Taylor series used in the numerical solution.



Figure 2: Convergence of numerical solution and rate of convergence

*(f) Modify your finite difference solver by setting the first derivative of the temperature dT/dx = 0 at the both end points. What happens when you try to solve the problem? Why?*

Matlab gives the warning “Warning: Matrix is singular to working precision.”

This indicates that either the matrix is singular, or close enough to singular that due to working precision is indistinguishable. In this case it is actually singular, and is not a result of working precision.

A singular matrix is a square matrix whose determinant is zero. By setting a Dirichlet at both ends the matrix becomes symmetric about its main diagonal, the determinant is therefore zero, and thus the matrix is singular. A singular matrix does not have an inverse and therefore it is not possible to solve Ax=F.

The mathematical insolubility is a manifestation of a physical impossibility. A Dirichlet boundary condition indicates the end is perfectly insulated. If both ends are insulated, and the bar is being heated then the bar must be changing temperature with respect to time. However we have implicitly set dT/dt = 0 in the working equation (a special case, the stationary heat equation), so therefore we have a contradiction. The bar *must* change temperature with respect to time, but we are imposing the condition that it can’t.

From an analytical standpoint it is impossible to solve T=½x²+Ax+B with dT/dx = 0 at both x=0 and x=1 as we would require that 1+A=0+A, for which no A exists. If however we allow dT/dt to be non-zero the analytical solution is not of this form and is solvable.

**References:**

[1] <http://www.stanford.edu/~fringer/teaching/numerical_methods_02/handouts/lecture4.pdf>

**Appendix A: Matlab program**

%Finite difference method for heat transfer in a uniform bar

close all

clear all

%Preallocate

Size=16;

InfNormError = zeros(Size-1,1);

%Define non-changing geometry

LBar = 1; %Length of bar

Forcing = 1; %Heat flux on bar

Mode = 1; %1 for normal operation, 2 to examine error

for N=2:Size

if Mode==1

N=Size;

end

%Geometry

NNodes = N+1; %number of nodes

DeltaX = LBar/N; %size of step

X = 0:DeltaX:LBar; %Nodal locations

%Set up matrices

A = zeros(NNodes, NNodes);

F = zeros(NNodes,1);

%Populate with problem specific equations/values

for(i = 2:(NNodes-1))

A(i,i) = -2;

A(i,i-1) = 1;

A(i,i+1) = 1;

switch Mode

case 1

F(i) = 1;

case 2

F(i) = sin(X(i));

end

end

A = (1./DeltaX^2).\*A;

%Boundary conditions

%x=0

A(1,1) = 1;

F(1) = 0; %Set T=0

%x=LBar

A(NNodes, NNodes) = 3/(2\*DeltaX); %-

A(NNodes, NNodes-1) = -4/(2\*DeltaX); %Apply back difference for df/dx

A(NNodes, NNodes-2) = 1/(2\*DeltaX); %-

F(NNodes) = 0; %Set dT/dx=0

%Solve

Solution = A\F;

switch Mode

case 1

Exact = (0.5\*(X.^2))-X;

case 2

Exact = cos(1)\*X - sin(X);

end

Exact = Exact';

if Mode==1

break

end

%Examine inf-norm error

InfNormError(N-1)=max(abs(Exact-Solution));

end

%Plot results

figure(1)

plot(X, X\*0,'-o')

hold on

plot(X, Solution, '-\*r')

temp = plot(X, Exact, 'o');

set(temp,'Color','green')

%Plot error, find slope of loglog curve

if Mode==2

figure(2)

N = 2:Size;

N=N';

loglog(N,InfNormError, '-r')

loglogfit = polyfit(log(N),log(InfNormError),1);

text(10,0.001,num2str(loglogfit(1)))

end