MAT 425: Homework 8 (03/30)

Ex 1. The code is given in page 5. 2pts

We plot in the figure 1 the solution for $\Delta x = 10^{-2}$. 1pt

To estimate the accuracy of the method, we estimate the solution for several $\Delta x = \frac{1}{N+1}$ (e.g. $N = [10, 30, 70, 200, 550, 1500]$). Then, we estimate the difference in L_∞ norm between the numerical solution $\widetilde{y}$ and the exact solution y_{ref}:

$$\text{error}(\Delta x) = \max_i |y_{exact}(x_i) - \widetilde{y}_i|.$$

We plot the log-log of the error in figure 2. A linear regression shows that the slope of the curve is $c = 1.9662 \approx 2$. Therefore, we deduce that the method is of order $O(\Delta x^2)$. 1pt 1pt

Ex 2. The code is given in page 6. 2pts

A solution of the BVP is given in figure 3 for $\Delta x = 10^{-2}$ after 5 iterations of the Newton's method. 1pt

To estimate the accuracy of the scheme, we first compute a 'reference' solution computed with $N = 4000$. Then, we estimate solutions for several $N < 4000$ and estimate the difference:

$$\text{error}(\Delta x) = \max_i |y_{ref}(x_i) - \widetilde{y}_i|.$$

The error in log-log plot is given in figure 4. The estimation of the slope yields to $c = 2.072$, thus the method is of accuracy $O(\Delta x^2)$. .5pt .5pt

Ex 3. The difficulty here is to compute the differential of $F(y) = y^3 - y \cdot y'$. Numerically, we have:

$$F_i = y_i^3 - y_i \cdot \frac{y_{i+1} - y_{i-1}}{2\Delta x}.$$

Denoting $\mathbf{y}$ and $\mathbf{dy}$ the numerical estimation of y and y' (using a central difference method), we deduce:

$$DF = \text{diag}(3\mathbf{y}^2 - \mathbf{dy}) + \text{diag}(-\mathbf{y}_{1:(N-1)}/(2\Delta x), 1) + \text{diag}(\mathbf{y}_{2:N}/(2\Delta x), -1).$$

See page 7. We plot in figure 5 the solution after 5 iterations of the Newton method along with the exact solution: $y(x) = \frac{1}{1+x}$. 1pt

We compare the accuracy and the computation time of the finite-difference method with the shooting method. With this aim, we compute the solution of the BVP for different values of N and estimate the error between the numerical solutions and the exact one (using L^∞ norm). As we observe in figure 6 (left), the shooting method is 4th order accurate whereas the FDM is only 2nd order accurate. In term of computation time, the FDM is faster than the shooting method for small N. But as N increases, the computation time for the FDM is increasing faster and it exceeds the shooting method for $N = 1500$, The reason is that the FDM requires to inverse the 'large' matrix DF. .5pt .5pt

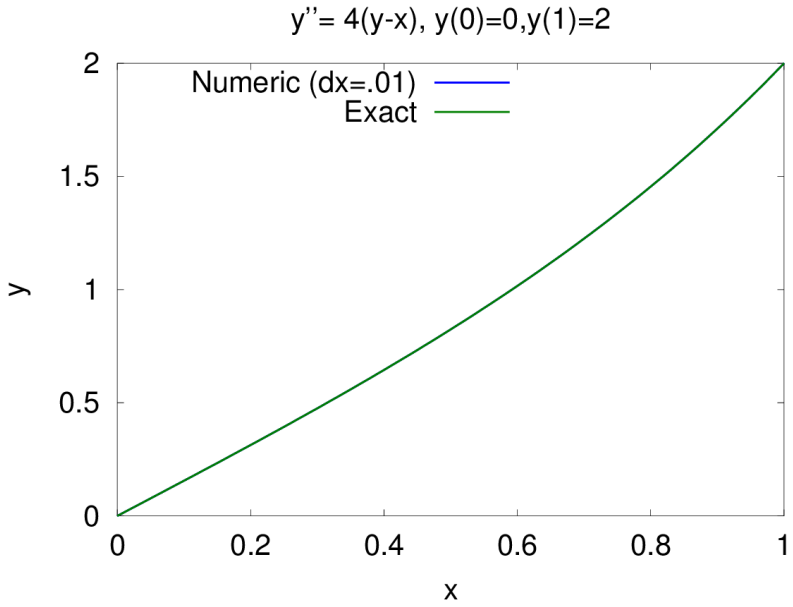

Figure 1: The solution of BVP given by the Finite-Difference-Method ($\Delta x = 10^{-2}$) and the exact solution. The two curves are in very good agreement.

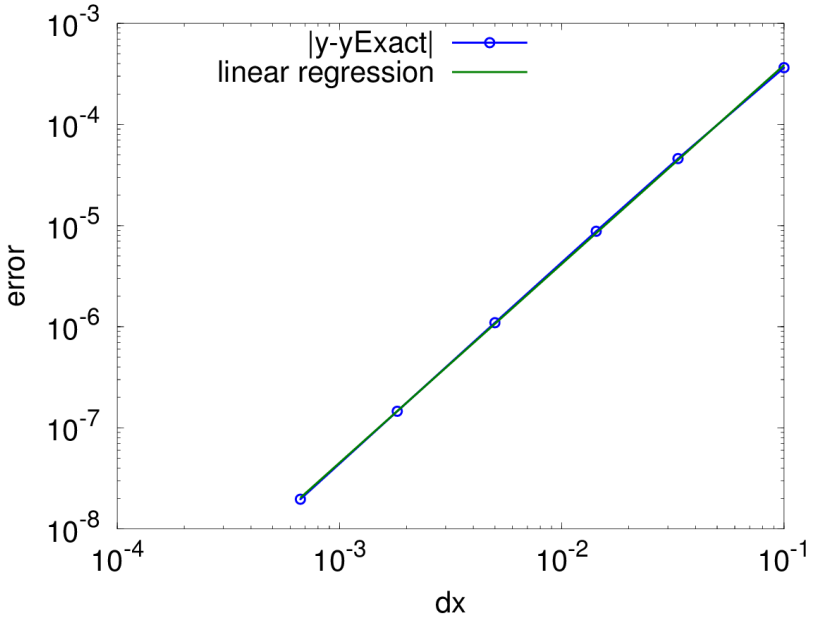

Figure 2: Error of the FDM method estimated for several Δx in log-log plot. We deduce that the method is of order 2.

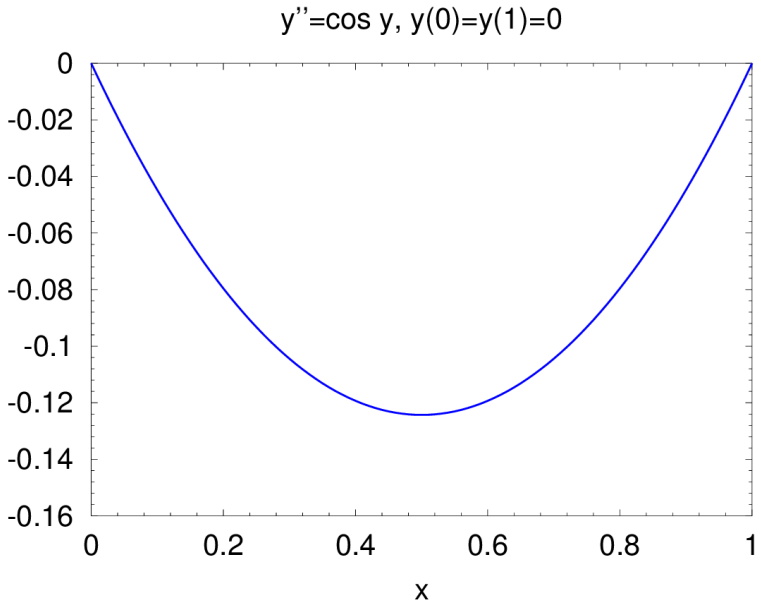

Figure 3: Solution of the non-linear BVP after 5 iterations of the Newton's method. Parameters: $\Delta x = 10^{-2}$.

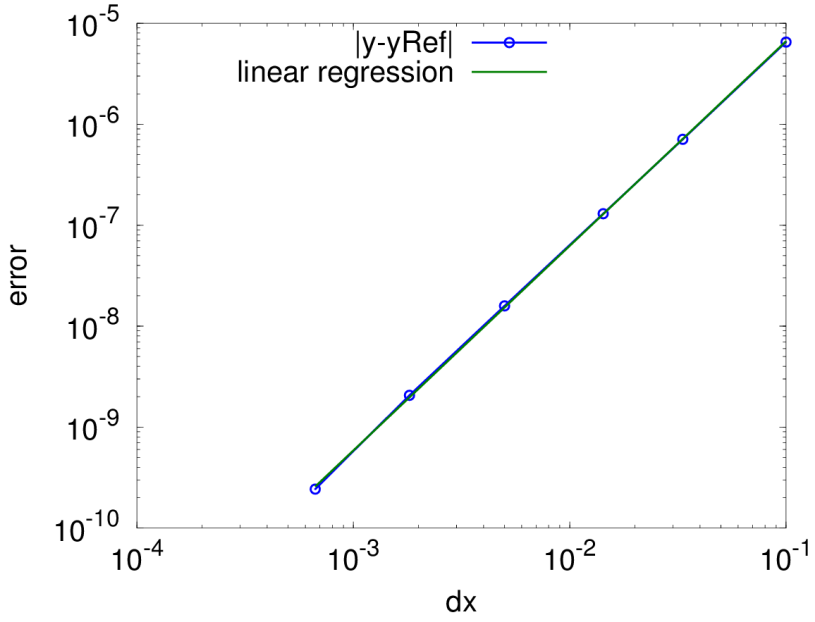

Figure 4: Error of the non-linear FDM method estimated for several Δx in log-log plot. The error is estimated using a 'reference' solution estimated with $N = 4000$ (i.e. $\Delta x = 2.5 \cdot 10^{-4}$). We find that the method is of order 2.

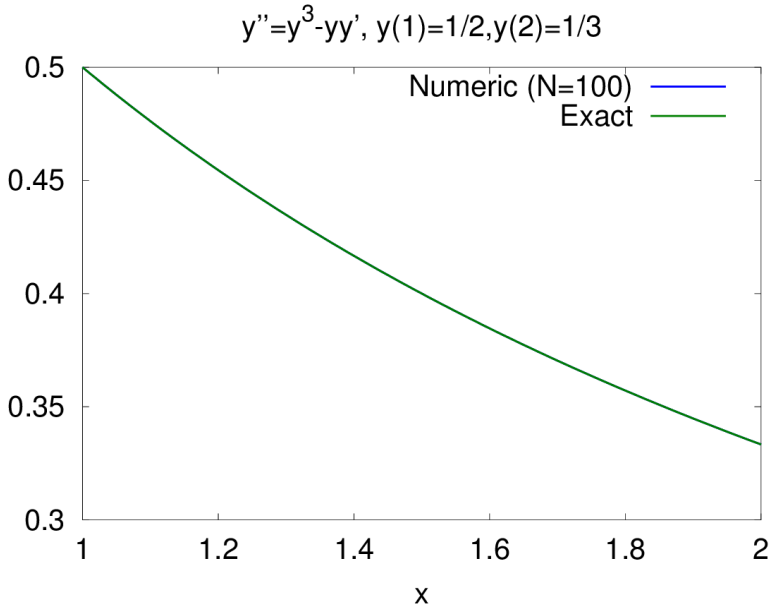

Figure 5: The solution of the non-linear BVP after 5 iterations of the Newton's method.

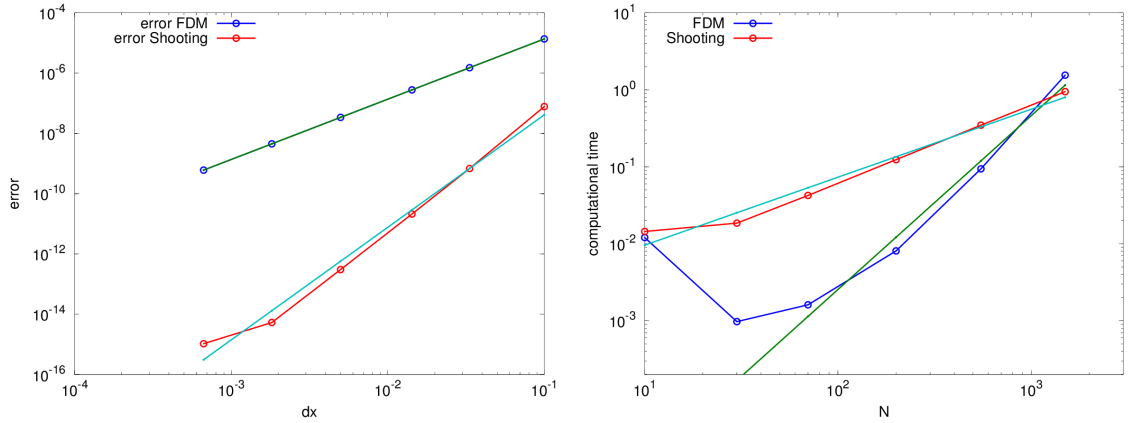

Figure 6: **Left:** accuracy of the FDM and shooting method for different Δx. **Right:** computational time for each method.

```
%% Solve the BVP with the finite difference method
%%   y'' = p(x) y' + q(x) y + r(x)
%%   y(0)= alpha, y(b)= beta
%% Boundary value problem
a     = 0;
b     = 1;
alpha = 0;
beta  = 2;
p = @(x) 0*x;
q = @(x) 4 + 0*x;
r = @(x) -4*x;
%% Parameters
N  = 100-1;
dx = 1/(N+1);
x  = a+dx*(1:N);
%% Initialization
A = diag(2+dx^2*q(x)) + diag(-1-dx/2*p(x(2:N)),-1) + diag(-1+dx/2*p(x(1:(N-1))),1);
vecB = -dx^2*r(x');
%% boundary condition
vecB(1) = vecB(1) + (1+dx/2*p(x(1)))*alpha;
vecB(N) = vecB(N) + (1-dx/2*p(x(N)))*beta;
%%----------------------------%%
%%-----       solve        ---%%
%%----------------------------%%
Y=A\vecB;
%% boundary condition
y_sol = [alpha; Y; beta];

%% ----- plot
yExact = @(x) exp(2)/(exp(4)-1)*(exp(2*x)-exp(-2*x))+x;
plot([a x b],y_sol', [a x b],yExact([a x b]))
xlabel('x')
ylabel('y')
legend('Numeric (dx=.01)', 'Exact','location','northwest')
title('y'''= 4(y-x), y(0)=0,y(1)=2')
```

```
%% Solve the BVP with the finite difference method
%%   y'' = cos y
%%   y(0)=y(1)= 0
F  = @(y) cos(y);
DF = @(y) -diag(sin(y));
%% Parameters
N  = 100-1;
dx = 1/(N+1);
intX = linspace(0,1,N+2);
%% iteration Newton method
nbIter = 5;
%% Initialization
L = 2*diag(ones(1,N)) - diag(ones(1,N-1),1) - diag(ones(1,N-1),-1);
Y = zeros(N,1);

%%---------------------------%%
%%-----       loop        ---%%
%%---------------------------%%
for k=1:nbIter
  %% Newton method
  DJ = L + dx^2*DF(Y);
  Y  = Y - DJ\ (L*Y + dx^2*F(Y));
end
%% plot
plot(intX,[0 Y' 0])
xlabel('x')
title('y''''=cos y, y(0)=y(1)=0')
```

```
%% Solve the BVP with the finite difference method
%%   y'' = y^3-yy'
%%   y(1)= 1/2  ,  y(2)=1/3.
F  = @(y,dy)    y.^3 - y.*dy;
DF = @(y,dy,dx) diag(3*y.^2 - dy)  +  ...
     diag(-y(1:(end-1))/(2*dx),1)  +  ...
     diag(y(2:end)/(2*dx),-1);
a = 1;
b = 2;
alpha = 1/2;
beta  = 1/3;
%% Numerical parameters
N  = 100-1;
nbIter = 5;
%% Initialization
dx = 1/(N+1);
intX = linspace(a,b,N+2);
L = 2*diag(ones(1,N)) - diag(ones(1,N-1),1) - diag(ones(1,N-1),-1);
Y = alpha + (1:N)'*dx*(beta-alpha);

%%---------------------------%%
%%-----       loop        ---%%
%%---------------------------%%
tic
for k=1:nbIter
  %%---- Newton method
  %% y'
  dY = (Y(3:end)-Y(1:(end-2)))/(2*dx);
  dY = [(Y(2)-alpha)/(2*dx); dY; (beta-Y(end-1))/(2*dx)];
  %% new y
  DJ = L + dx^2*DF(Y,dY,dx);
  Y  = Y - DJ\ (L*Y + dx^2*F(Y,dY) - [alpha; zeros(N-2,1); beta]);
end
cTime = toc

%% plot
yExact = @(x) 1./(1+x);
plot(intX,[alpha Y' beta], intX,yExact(intX))
xlabel('x')
legend('Numeric (N=100)','Exact')
title('y''''=y^3-yy'', y(1)=1/2,y(2)=1/3')
```