AMATH 585 Homework 3

Cade Ballew

January 31, 2022

1 Problem 1

1.1 Part a

Considering an initial guess of

$$\theta(t) = 0.7 + t(t - 2\pi)^2$$

with a stepsize of $h = 1/2002$ and a stopping tolerance of $\|\delta^{[k]}\|_\infty < 10^{-12}$, we are able to produce yet another solution to the nonlinear problem in the text different from those illustrated in figures 2.4 and 2.5 which is displayed in the following plot.

1.2 Part b

To find a solution to this BVP with the same general behavior as figure 2.5 for a longer time interval, we choose our initial guess based on properties at our initial time interval $T = 2\pi$. Namely, we reuse the computed solution for a smaller T by using it as an initial guess (after scaling to the new time interval appropriately) for a larger time interval. The following plots display the solution that we obtain for $T = 20, 50, 100, 200$ by doing this recursively.

The following table lists the maximum value of θ obtained in each time interval. Clearly, they appear to be converging to π.

T	$\max_i \theta_i$
2π	2.8975061095462498
20	3.1413385828458429
50	3.1415926535120287
100	3.1415926535897931
200	3.1415926535897931

The following MATLAB code was used the obtain the results for this problem.

```
T=2*pi;
m=2001;
t=linspace(0,T,m+2)';
h = t(2)-t(1);
alpha=0.7; beta=0.7;

theta=0.7+t.*(t-2*pi).^2;


theta = theta(2:end-1); %remove boundary
tol=1e-12; itermax=1000;
[theta,~,~]=NewtonSolve(theta,alpha,beta,h,tol,itermax);
figure(1)
plot(t,theta)
plot(t,theta)
xlabel('t'); ylabel('\theta(t)')
title('New solution for given BCs')
saveas(gcf,'hw3p1a','epsc')

i=1;
fprintf('T           max(theta)\n')

theta=0.7+sin(t/2);
theta = theta(2:end-1); %remove boundary
[theta,~,~]=NewtonSolve(theta,alpha,beta,h,tol,itermax);
fprintf('%i      %16.16d\n',T,max(theta))
figure(2)
for T=[20,50,100,200]
    t=linspace(0,T,m+2)';
    h = t(2)-t(1);
    theta = theta(2:end-1); %remove boundary, reuse old theta as new guess
    [theta,~,~]=NewtonSolve(theta,alpha,beta,h,tol,itermax);
    subplot(2,2,i)
    plot(t,theta)
    title(['T=',num2str(T)])
    xlabel('t')
```

```
    ylabel('\theta(t)')
    fprintf('%i      %16.16d\n',T,max(theta))
    i=i+1;
end
saveas(gcf,'hw3p1b','epsc')

function G=buildG(theta,alpha,beta,h)

G=zeros(length(theta),1);
theta = [alpha; theta; beta]; %include BCs for computation
for i=1:length(theta)-2
    G(i)=(theta(i)-2*theta(i+1)+theta(i+2))/h^2+sin(theta(i+1));
end

end


function J=buildJ(theta,h)

maindiag=-2+cos(theta)*h^2;
J=spdiags([ones(length(theta),1),maindiag,ones(length(theta),1)],-1:1,...
    length(theta),length(theta));
J=J/h^2;

end


function [theta,iter,dnormvec]=NewtonSolve(initial_guess,alpha,beta,h,tol,itermax)

theta = initial_guess;
deltanorm=Inf;
iter=0;
while deltanorm>tol && iter<itermax
    G=buildG(theta,alpha,beta,h);
    J=buildJ(theta,h);
    delta=-J\G;
    theta=theta+delta;
    deltanorm=norm(delta,'inf');
    iter=iter+1;
    dnormvec(iter)=deltanorm;
end

theta=[alpha;theta;beta]; %add BCs

end
```

2 Problem 2

2.1 Part a

Consider the boundary value problem

$$-u_{xx} + (1+x^2)u = f, \quad 0 \le x \le 1,$$

$$u(0) = 0, \quad u(1) = 0$$

on a uniform grid with spacing $h = 1/(m+1)$, with the following set of difference equations that has local truncation error $O(h^2)$:

$$\frac{2u_i - u_{i+1} - u_{i-1}}{h^2} + (1+x_i^2)u_i = f(x_i), \quad i = 1, \ldots, m.$$

By Gerschgorin's theorem, we can sum the absolute values of the non-diagonal entries of the coefficient matrix A associated with this set of difference equations to get that the Gerschgorin disks are given by

$$B\left(\frac{2}{h^2} + 1 + x_1^2, \frac{1}{h^2}\right) = B\left(\frac{2}{h^2} + 1 + h^2, \frac{1}{h^2}\right)$$

for $i = 1$,

$$B\left(\frac{2}{h^2} + 1 + x_m^2, \frac{1}{h^2}\right) = B\left(\frac{2}{h^2} + 1 + h^2m^2, \frac{1}{h^2}\right)$$

for $i = m$, and

$$B\left(\frac{2}{h^2} + 1 + x_i^2, \frac{2}{h^2}\right) = B\left(\frac{2}{h^2} + 1 + h^2i^2, \frac{2}{h^2}\right)$$

for $i = 2, \ldots, m-1$. We know that the eigenvalues of A must be contained in the union of these disks, so observing that A is symmetric so all eigenvalues are real,

$$\min_{i\in\{2,\ldots,m-1\}} \left\{\frac{1}{h^2} + 1 + h^2, \frac{1}{h^2} + 1 + h^2m^2, 1 + h^2i^2\right\} \le \lambda$$
$$\le \max_{i\in\{2,\ldots,m-1\}} \left\{\frac{3}{h^2} + 1 + h^2, \frac{3}{h^2} + 1 + h^2m^2, \frac{4}{h^2} + 1 + h^2i^2\right\}$$

if λ is an eigenvalue of A. Making the assumption that $m \ge 1$ so that $0 < h \le \frac{1}{2}$, we can easily reduce this to

$$\min\left\{\frac{1}{h^2} + 1 + h^2, 1 + h^2 2^2\right\} \le \lambda \le \max\left\{\frac{3}{h^2} + 1 + h^2m^2, \frac{4}{h^2} + 1 + h^2(m-1)^2\right\}.$$

Now, we note that with our restrictions on m and h, $3h^4 \le 3(1/2)^4 = 3/8 < 1$, so $1 + 4h^2 < \frac{1}{h^2} + 1 + h^2$. Similarly, we can also conclude that $\frac{3}{h^2} + 1 + h^2m^2 < \frac{4}{h^2} + 1 + h^2(m-1)^2$, so we can bound the eigenvalues of A by

$$1 + 4h^2 \le \lambda \le \frac{4}{h^2} + 1 + h^2(m-1)^2.$$

2.2 Part b

To see that the L_2-norm of the global error is the same order as the local truncation error, we borrow the book's notation by letting E be the global error and τ be the local truncation error and use the inequalities on page 19 to write

$$\|E\|_{L_2} \leq \|A^{-1}\|_{L_2}\|\tau\| = \left(\min_i |\lambda_i|\right)^{-1} \|\tau\| \leq \|\tau\| = \mathrm{O}(h^2).$$

The reason for this is that we know that 1 is a lower bound on the eigenvalues of A, because the lower bound we found in part a is clearly greater than 1 when $h > 0$.

3 Problem 3

Using the code from homework 2, we use the following to obtain approximate solutions at $h = .1$ and $h = .05$ compute a Richardson extrapolation

$$\phi_1(x_i; h) = \frac{4\tilde{u}(x_i; h/2) - \tilde{u}(x_i; h)}{3}$$

in the standard way where x_i denotes each gridpoint on the course grid.

```
m=9;
[u_coarse,h,errvec]=rodFD(c,utrue,ftrue,m);
errinf=norm(errvec,'inf');
fprintf('%.e       %.16e\n',h,errinf)

m=19;
[u_fine,h,errvec]=rodFD(c,utrue,ftrue,m);
errinf=norm(errvec,'inf');
fprintf('%.e       %.16e\n',h,errinf)

u_rich=(4*u_fine(1:2:end)-u_coarse)/3;
u_true=utrue(linspace(0,1,length(u_coarse)))';
errinf=norm(u_true-u_rich,'inf');
fprintf('Richardson       %.16e\n',errinf)
```

The following table prints error results which appear to give higher order accuracy as one would expect.

Approximation	Error in infinity norm
$\tilde{u}(x; .1)$	2.1832687983814104e-03
$\tilde{u}(x; .05)$	5.4785209963363103e-04
$\varphi_1(x; .1)$	2.7132000510379090e-06

Now, if we instead assume that the coarse grid approximation is piecewise linear, so that the approximation at the midpoint of each subinterval is the

average of the values at the two endpoints, one cannot use Richardson extrapolation with the fine grid approximation and these interpolated values on the coarse grid to obtain a more accurate approximation at these points in general. While we can Taylor expand

$$\frac{u(x+h)+u(x)}{2} - u(x+h/2) = \frac{hu'(x)+\mathrm{O}(h^2)}{2} - (h/2)u'(x) + \mathrm{O}(h^2) = \mathrm{O}(h^2)$$

to see that linear interpolation is also second order accurate, the issue lies in that we are using two different types of approximation. To see this more explicitly, say that the coefficient on the h^2 term in the error of our FD scheme is denoted $e(x)$ (with x dependence to account for the fact that these terms typically include some derivative of u evaluated at x), i.e. that

$$\tilde{u}(x;h) = u(x) + e(x)h^2 + \mathrm{O}(h^3).$$

Then, for our fine grid,

$$\tilde{u}(x;h/2) = u(x) + e(x)\frac{h^2}{4} + \mathrm{O}(h^3).$$

However, for the interpolation on our coarse grid, the approximation at a midpoint x is

$$\begin{aligned}
\frac{\tilde{u}(x+h/2;h)+\tilde{u}(x-h/2;h)}{2} &= \frac{u(x+h/2)+e(x+h/2)h^2+u(x-h/2)+e(x-h/2)h^2+\mathrm{O}(h^3)}{2} \\
&= \frac{u(x+h/2)+u(x-h/2)}{2} + \frac{e(x+h/2)h^2+e(x-h/2)h^2}{2} + \mathrm{O}(h^3) \\
&= u(x) + \frac{h^2}{4}u''(x) + \frac{e(x+h/2)h^2+e(x-h/2)h^2}{2} + \mathrm{O}(h^3) \\
&= u(x) + \frac{h^2}{4}u''(x) + \frac{h^2}{2}(2e(x)+\mathrm{O}(h^2)) + \mathrm{O}(h^3) \\
&= u(x) + (u''(x)/4 + e(x))h^2 + \mathrm{O}(h^3)
\end{aligned}$$

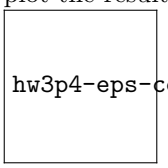
if we assume that e permits a Taylor expansion at x. In order for our standard Richardson extrapolation technique to work, we need to be able to eliminate the h^2 term by scaling the approximation on the fine grid by $2^2 = 4$. However, this will only work here if $e(x) = u''(x)/4 + e(x)$, i.e. that $u''(x) = 0$ which is not necessarily true in general. It would be difficult to even modify Richardson extrapolation to work here since $u''(x)$ is not necessarily known.

4 Problem 4

Differentiating the nonlinear difference equations (2.106) to get a Jacobian, we can see that

$$J_{i,j} = \begin{cases} -\frac{2\epsilon}{h^2} + \frac{u_{i+1}-u_{i-1}}{2h} - 1, & j = i \\ \frac{\epsilon}{h^2} + \frac{u_i}{2h}, & j = i+1 \\ \frac{\epsilon}{h^2} - \frac{u_i}{2h}, & j = i-1 \\ 0, & \text{otherwise,} \end{cases}$$

for $i, j = 1, \ldots m$ noting that $u_0 = \alpha$, $u_{m+1} = \beta$. With this, we apply Newton's method to solve (2.106) on the standard grid $a = x_0 < x_1 < \ldots < x_m < x_{m+1} = b$ with the standard spacing $h = x_{i+1} - x_i$ for $a = 0$, $b = 1$, $\alpha = -1$, $\beta = 1.5$, and $\epsilon = 0.01$. We do so for $h = 1/20$, $h = 1/40$, $h = 1/80$, and $h = 1/160$ and plot the results on top of each other as follows.



The following MATLAB code was used to solve the equations and create the above plot. Newton's method is used with a stopping tolerance of $\|\delta^{[k]}\|_\infty < 10^{-12}$ and an initial guess of the function defined in (2.105) with the values of w_0 and $\bar{x}$ as given in (2.103) and (2.104), respectively is built in the function "buildinitialguess."

```
figure(1)
for m=[19 39 79 159]
    a=0; b=1; alpha=-1; beta=1.5; epsilon=0.01;
    x=linspace(a,b,m+2)';
    h = x(2)-x(1);
    tol=1e-12; itermax=100;

    initial_guess=buildinitialguess(x(2:end-1),a,b,alpha,beta,epsilon);
    [u,iter,dnormvec]=NewtonSolve(initial_guess,alpha,beta,epsilon,h,tol,itermax);
    plot(x,u)
    hold on
end
xlabel('x')
ylabel('u(x)')
legend('h=1/20','h=1/40','h=1/80','h=1/160')
title('Solution of difference equations for different h')
hold off
saveas(gcf,'hw3p4','epsc')

function u=buildinitialguess(x,a,b,alpha,beta,epsilon) %function from 2.105

w0=(a-b+beta-alpha)/2;
xbar=(a+b-alpha-beta)/2;
u=x-xbar+w0*tanh(w0*(x-xbar)/(2*epsilon));

end



function G=buildG(u,alpha,beta,h,epsilon)
```

```
G=zeros(length(u),1);
u = [alpha; u; beta];
for i=1:length(u)-2
    G(i)=epsilon*(u(i)-2*u(i+1)+u(i+2))/h^2+u(i+1)*((u(i+2)-u(i))/(2*h)-1);
end

end



function J=buildJ(u,alpha,beta,h,epsilon)

u = [alpha; u; beta];

for i=1:length(u)-2
    Jdiag(i)=-2*epsilon/h^2+(u(i+2)-u(i))/(2*h)-1;
end
Jsuperdiag=epsilon/h^2+u(1:end-2)/(2*h);
Jsubdiag=epsilon/h^2-u(3:end)/(2*h);

J=spdiags([Jsubdiag,Jdiag',Jsuperdiag],-1:1,length(u)-2,length(u)-2);

end


function [u,iter,dnormvec]=NewtonSolve(initial_guess,alpha,beta,epsilon,h,tol,itermax)

u = initial_guess;
deltanorm=Inf;
iter=0;
while deltanorm>tol && iter<itermax
    G=buildG(u,alpha,beta,h,epsilon);
    J=buildJ(u,alpha,beta,h,epsilon);
    delta=-J\G;
    u=u+delta;
    deltanorm=norm(delta,'inf');
    iter=iter+1;
    dnormvec(iter)=deltanorm;
end

u=[alpha;u;beta]; %add BCs

end
```