M368K Homework 9

§ 11.1 #2a[1], 9 § 11.2 #4a[2]

Hershal Bhave (hb6279)

April 12, 2013

1 § 11.3

1.1 § 2b[1]

The Boundary-value problem

$$y'' = y' + 2y + \cos x, \qquad 0 \leqslant x \leqslant \frac{\pi}{2}, \qquad y(0) = -0.3, \qquad y\left(\frac{\pi}{2}\right) = -0.1$$

has the solution $y(x) = -\frac{1}{10}(\sin x + 3\cos x)$. Use the Linear Finite-Difference method to approximate the solution, and explicitly write out the centered-difference equations. Solve and compare the results to the actual solution. Assume $h = \frac{\pi}{4}$.

Given a differential equation in the form

$$y''(x_i) = p(x_i)y'(x_i) + q(x_i)y(x_i) + r(x_i), \tag{1}$$

the third-order Taylor polynomial of $y(x_{i-1})$ and $y(x_{i-1})$ are

$$\begin{aligned} y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{3}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^+) + \dots \\ y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{3}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^+) - \dots \end{aligned} \tag{2}$$

Combining the equations in eq. (2) and solving for y'', we obtain the Centered-Difference Formula for y''.

$$y''(x_i) = \frac{1}{h^2}[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})] - \frac{h^2}{12}y^{(4)}(\xi_i) \tag{3}$$

A similar Central Difference Equation can be found for y'.

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

By ignoring the higher-order terms we can calculate an approximation to the solution $y(x_i)$. Using the Centered-Difference Formula in eq. (3) and the given equation for $y''(x_i)$ in the form of eq. (1) we obtain

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} = p(x_i)y'(x_i) + q(x_i)y(x_i) + r(x_i) \tag{5}$$

By applying eq. (4) to eq. (5), we obtain

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} = p(x_i)\frac{y(x_{i+1}) - y(x_{i-1})}{2h} + q(x_i)y(x_i) + r(x_i) \tag{6}$$

which will allow us to write the approximations in matrix-form and then solve the resulting matrix. Observe that we can simplify this into

$$-r(x_i) = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} + p(x_i)\left(\frac{y(x_{i+1} - y(x_{i-1})}{2h}\right) + q(x_i)y(x_i)$$

or alternatively

$$-h^2 r(x_i) = \left(-1 - \frac{h}{2}p(x_i)\right) y(x_{i-1}) + \left(2 + h^2 q(x_i)\right) y(x_i) - \left(-1 + \frac{h}{2}p(x_i)\right) y(x_{i+1}) \tag{7}$$

Then applying eq. (7) with $h = \frac{\pi}{4}$, we obtain

$$-\frac{\pi^2}{16}r(x_i) = \left(-1 - \frac{\pi}{8}p(x_i)\right) y(x_{i-1}) + \left(2 + \frac{\pi^2}{16}q(x_i)\right) y(x_i) - \left(-1 + \frac{\pi}{8}p(x_i)\right) y(x_{i+1}) \tag{8}$$

By invoking our boundary value conditions and bounds we have $i = 1$, $x_0 = 0$, $x_1 = \frac{\pi}{4}$, $x_2 = \frac{\pi}{2}$, $y(0) = -0.3$, and $y\left(\frac{\pi}{2}\right) = -0.1$. We can substitute the interior values into eq. (8) to obtain an equation which, when solved, describes $y(x_i)$ at the corresponding interior mesh points of x_i. In our case the interior mesh point is x_1.

$$\begin{aligned}
-\frac{\pi^2\sqrt{2}}{32} &= \left(-1 - \frac{\pi}{8}\right) y(x_0) + \left(2 + \frac{\pi^2}{8}\right) y(x_1) - \left(-1 + \frac{\pi}{8}\right) y(x_2) \\
\left(2 + \frac{\pi^2}{8}\right) y(x_1) &= -\frac{\pi^2\sqrt{2}}{32} + \left(-1 - \frac{\pi}{2}\right)(0.3) - \left(-1 + \frac{\pi}{2}\right)(0.1) \\
&= \left[-\frac{\pi^2\sqrt{2}}{32} + \left(-1 - \frac{\pi}{2}\right)(0.3) + \left(-1 + \frac{\pi}{2}\right)(0.1)\right]\left(2 + \frac{\pi^2}{8}\right)^{-1}
\end{aligned} \tag{9}$$

Finally we obtain our solution:

$$\boxed{y\left(\frac{\pi}{4}\right) = -0.28287}$$

We can confirm this approximation is correct by running it through the code in listing 1 and approximating $y(x_i)$'s values at more mesh points. Doing this with $n = 3$ I obtained the data in table 1 on page 3, confirming that our solution is close to the actual solution, even with only one step.

x_i	w_i	$y(x_i)$	$\lvert w_i - y(x_i)\rvert$
0.00000	−0.30000	−0.30000	
0.39270	−0.31569	−0.31543	2.5320×10^{-4}
0.78540	−0.28291	−0.28284	6.3136×10^{-5}
1.17810	−0.20700	−0.20719	1.9735×10^{-4}
1.57080	−0.10000	−0.10000	

Table 1: Approximation of 2b with $n = 3$

1.2 9^2

Use Theorem 9.1 to prove Theorem 11.3.

Theorem 9.1 Let A be an $n \times n$ matrix and R_i denote the circle in the complex plane with center a_{ii} and radius $\sum_{j=1,j\neq i}^{n} |a_{ij}|$; that is,

$$R_i = \left\{ z \in \mathbb{C} \;\middle|\; |z - a_{ii}| \leqslant \sum_{j=1,j\neq i}^{n} |a_{ij}| \right\}$$

The eigenvalues of A are contained within the union of these circles, $R = \cup_{i=1}^{n} R_i$.

Theorem 11.3 Suppose that p, q, and r are continuous on $[a, b]$. If $q(x) \geqslant 0$ on $[a, b]$, then the tridiagonal linear system has a unique solution provided that $h < \frac{2}{L}$, where $L = \max_{a \leqslant x \leqslant b} |p(x)|$.

Proof:

By rearranging some of the inequality statements, Theorem 11.3 implies $|\frac{h}{2}p(x)| < 1$. Combining that knowledge with knowledge of the tridiagonal matrix which defines the solution to the linear BVP, we know that

$$\begin{aligned}\sum_{j=1,j\neq i}^{n} |a_{ij}| &= |-1 - \frac{h}{2}p(x_i)| + |-1 - \frac{h}{2}p(x_i)| \\ &= 1 + \frac{h}{2}p(x_i) + 1 - \frac{h}{2}p(x_i)\end{aligned}$$

Which implies

$$0 \leqslant \sum_{j=1,j\neq i}^{n} |a_{ij}| < 2$$

We know that the diagonal entries, a_{ii} are composed of $2 + h^2 q(x)$. Theorem 9.1 states that we must be able to find a radius z which satisfies

$$R_i = \left\{ z \in \mathbb{C} \;\middle|\; |z - 2 - h^2 q(x)| < 2 \right\}$$

Since we also know that $q(x)$ is greater than zero, then this radius must exist, meaning the matrix which defines the solutions to the BVP must have precisely k (counting multiplicities) of eigenvalues. This means that the tridiagonal matrix is non-singular, and thus has a unique solution.

■

2 § 11.4

2.1 4a[3]

The Boundary-Value Problem

$$y'' = y^3 - yy', \qquad 1 \leqslant x \leqslant 2, \qquad y(1) = \frac{1}{2}, \qquad y(2) = \frac{1}{3}$$

has the solution $y(x) = (x+1)^{-1}$. Use the Nonlinear Finite-Difference Algorithm with TOL $= 10^{-4}$ and $n = 3$ to approximate the solution. Explicitly write out the centered-difference equations and perform one Newton step with a straight guess of $y^{(0)}$. Solve and compare the results to the actual solution.

Given a differential equation in the form

$$y''(x_i) = f(x_i, y(x_i), y'(x_i)) \tag{10}$$

we can apply a similar method we used in eq. (6) to find the Centered-Difference equations in the nonlinear case.

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} = f\left(x_i, y(x_i), \frac{y(x_{i+1}) - y(x_{i-1})}{2h}\right) \tag{11}$$

By invoking our boundary value conditions and bounds we have $i = 1$, $x_0 = 1$, $x_1 = \frac{3}{2}$, $x_2 = 2$, $y(1) = \frac{1}{2}$, and $y(2) = \frac{1}{3}$. Since $n = 3$, $h = \frac{1}{4}$.

$$\frac{y(2) - 2y(\frac{3}{2}) + y(1)}{\frac{1}{16}} - f\left(\frac{3}{2}, y\left(\frac{3}{2}\right), \frac{y(2) - y(1)}{\frac{1}{2}}\right) = 0 \tag{12}$$

Performing one Newton Iteration, I obtained the table of values listed in table 2.

x_i	w_i	$y(x_i)$	$\lvert w_i - y(x_i)\rvert$
1.0000	0.50000	0.50000	
1.2500	0.44419	0.44444	2.5317×10^{-4}
1.5000	0.39944	0.40000	5.5656×10^{-4}
1.7500	0.36275	0.36364	8.9081×10^{-4}
2.0000	0.33333	0.33333	

Table 2: Approximation of 4a with n=3 using one Newton Iteration

3 § 11.5

3.1 2^4

Use the Piece-wise Linear Algorithm to approximate the solution to the boundary-value problem

$$-\frac{\mathrm{d}}{\mathrm{d}x}(xy') + 4y = 4x^2 - 8x + 1, \qquad 0 \leqslant x \leqslant 1, \qquad y(0) = y(1) = 0$$

using $x_0 = 0$, $x_1 = 0.4$, $x_2 = 0.8$, $x_3 = 1$. Compare your results to the actual solution $y(x) = x^2 - x$. Explicitly write the finite-element equations. Solve and compare at nodes. Note $\int_0^1 \phi_1 f \mathrm{d}x = -0.5813$ and $\int_0^1 \phi_2 f \mathrm{d}x = -0.7960$.

The nonzero entries of the tridiagonal matrix A are defined by

$$a_{ij} = \int_0^1 [p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i'(x)\phi_j'(x)]\mathrm{d}x \tag{13}$$

and

$$b_i = \int_0^1 f(x)\phi_i(x)\mathrm{d}x \tag{14}$$

So then

$$\begin{aligned}
Q_{1,i} &= \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)(x - x_i)q(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n-1 \\
Q_{2,i} &= \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} (x - x_{i-1})^2 q(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n \\
Q_{3,i} &= \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)^2 q(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n \\
Q_{4,i} &= \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n+1 \\
Q_{5,i} &= \frac{1}{h_{i-1}} \int_{x_{i-1}}^{x_i} (x - x_{i-1})f(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n \\
Q_{6,i} &= \frac{1}{h_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - x)f(x)\mathrm{d}x, && \text{for each } i = 1, \dots, n
\end{aligned} \tag{15}$$

which expands the entries of the matrix to

$$\begin{aligned}
a_{i,i} &= Q_{4,i} + Q_{4,i+1} + Q_{2,i} + Q_{3,i}, && \text{for each } i = 1, \dots, n \\
a_{i,i+1} &= -Q_{4,i+1} + Q_{1,i}, && \text{for each } i = 1, \dots, n-1 \\
a_{i,i-1} &= -Q_{4,i} + Q_{1,i-1} && \text{for each } i = 2, \dots, n \\
b_i &= Q_{5,i} + Q_{6,i} && \text{for each } i = 1, \dots, n
\end{aligned} \tag{16}$$

Evaluating each integral for $i = 1, 2$ gives

$$A = \begin{pmatrix} 3.06667 & -1.23333 \\ -1.23333 & 6.8 \end{pmatrix} \tag{17}$$

and

$$b = \begin{pmatrix} -0.5813 \\ -0.7960 \end{pmatrix} \tag{18}$$

Finally, the solution is

$$c = \begin{pmatrix} -.25553 \\ -.16335 \end{pmatrix} \tag{19}$$

Comparing this to the actual solution, we obtain the table table 3.

x_i	c_i	$y(x_i)$	$\lvert c_i - y(x_i)\rvert$
0	0	0	
0.4	-.25553	-0.2400	0.0155300
0.8	-.16335	-0.1600	0.0033500
1	0	0	

Table 3: Approximation of 4a with n=3 using one Newton Iteration

4 Minilab

4.1 Part b

The maximum temperature occurs at approximately $(-0.12, 243.6)$. Less granular data is listed in table 4.

x_i	y_i
-2.00000	0.00000
-1.50000	88.54167
-1.00000	177.08333
-0.50000	265.62500
0.00000	291.66667
0.50000	242.18750
1.00000	161.45833
1.50000	80.72917
2.00000	0.00000

Table 4: Minilab data with $\gamma = 50$ and $\beta = 0$

4.2 Part c

The optimal laser intensity parameter γ I obtained was 486, and the optimal cooling air velocity parameter β I obtained was 37. This yielded the data in table 5.

x_i	y_i
-2.00000	0
-1.50000	24.03177
-1.00000	-52.40827
-0.50000	90.25972
0.00000	500.74233
0.50000	98.48364
1.00000	-50.29037
1.50000	21.14197
2.00000	0

Table 5: Minilab data with $\gamma = 486$ and $\beta = 37$

5 Code

```
#!/usr/bin/octave
# Created by Hershal Bhave on 04/11/13
# For M368K HW10,  11.3 Number 2a
# Written in GNU Octave
#
# Description: Uses the Linear Finite-Difference method to approximate
# the solution within the interval of the Boundary Value Problem in
# the form of -y''+p(x)y'+q(x)y+r(x)=0, given p(x), q(x), r(x),
# endpoints a, b, boundary conditions alpha, beta and subintervals n.
#

function [x,w] = linfindiff(p, q, r, a, b, alpha, beta, n)

  ai = bi = ci = di = zeros(n, 1);

  h = (b - a)/(n+1);
  x = a + h;
  ai(1) = 2 + (h^2)*q(x);
  bi(1) = -1 + (h/2)*p(x);
  di(1) = -h^2*r(x) + (1 + (h/2)*p(x))*alpha;

  for i=2:n-1
    x = a + i*h;
    ai(i) = 2 + (h^2)*q(x);
    bi(i) = -1 + (h/2)*p(x);
    ci(i) = -1 - (h/2)*p(x);
    di(i) = -(h^2)*r(x);
  endfor

  x = b - h;
  ai(n) = 2 + (h^2)*q(x);
  ci(n) = -1 - (h/2)*p(x);
  di(n) = -h^2*r(x) + (1 - (h/2)*p(x))*beta;

  l(1) = ai(1);
  u(1) = bi(1)/ai(1);
  z(1) = di(1)/l(1);

  for i=2:n-1
      l(i) = ai(i) - ci(i)*u(i-1);
      u(i) = bi(i)/l(i);
      z(i) = (di(i) - ci(i)*z(i-1))/l(i);
  endfor

  l(n) = ai(n) - ci(n)*u(n-1);
  z(n) = (di(n) - ci(n)*z(n-1))/l(n);

  w(n+1) = beta;
  w(n) = z(n);

  for i=n-1:-1:1
      w(i) = z(i)-u(i)*w(i+1);
  endfor

  x = a:h:b;
  w = [alpha w];

endfunction
```

Listing 1: `linfindiff.m`

```
#!/usr/bin/octave
# Created by Hershal Bhave on 04/11/13
# For M368K HW10,  11.4 Number 4a
# Written in GNU Octave
#
# Description: Uses the Nonlinear Finite-Difference method to approximate
# the solution at values within the interval of the Boundary Value
# Problem in the form of -y''+p(x)y'+q(x)y+r(x)=0, given p(x), q(x),
# r(x), endpoints a, b, boundary conditions alpha, beta, the number
# of subintervals n, and maximum iterations M.
#

function [x,w] = nonlinfindiff(f, fy, fyp, a, b, alpha, beta, n, M)

  if n<2
     error("n must be >=2\n");
  endif

  tol = 10^-6;
  w = zeros(1, n+1);
  ai = bi = ci = di = zeros(1,n);
  h = (b - a)/(n+1);
  w(n+1) = beta;

  for i=1:n
    w(i) = alpha + i*((beta-alpha)/(b-a))*h;
  endfor

  k=1;
  do
    x = a + h;
    t=(w(2)-alpha)/(2*h);
    ai(1) = 2 + h^2*fy(x,w(1),t);
    bi(1) = -1 + (h/2)*fyp(x,w(1),t);
    di(1) = -(2*w(1) - w(2) - alpha + h^2*f(x,w(1),t));

    for i=2:n-1
      x = a + i*h;
      t = (w(i+1)-w(i-1))/(2*h);
      ai(i) = 2 + h^2*fy(x,w(i),t);
      bi(i) = -1 + (h/2)*fyp(x,w(i),t);
      ci(i) = -1 - (h/2)*fyp(x,w(i),t);
      di(i) = -(2*w(i) - w(i+1) - w(i-1) + h^2*f(x,w(i),t));
    endfor

    x = b - h;
    t = (beta - w(n-1))/(2*h);
    ai(n) = 2 + h^2*fy(x,w(n),t);
    ci(n) = -1 - (h/2)*fyp(x,w(n),t);
    di(n) = -(2*w(n) - w(n-1) - beta + h^2*f(x,w(i),t));

    l(1) = ai(1);
    u(1) = bi(1)/ai(1);
    z(1) = di(1)/l(1);

    for i=2:n-1
      l(i) = ai(i) - ci(i)*u(i-1);
      u(i) = bi(i)/l(i);
      z(i) = (di(i) - ci(i)*z(i-1))/l(i);
    endfor

    l(n) = ai(n) - ci(n)*u(n-1);
    z(n) = (di(n) - ci(n)*z(n-1))/l(n);
```

```
    v(n) = z(n);
    w(n) += v(n);

    for i=n-1:-1:1
      v(i) = z(i)-u(i)*v(i+1);
      w(i) = w(i)+v(i);
    endfor

    k++;
    until k>M || norm(v) < tol

  if(k>M)
    printf("max iteration exceeded\n");
  endif

  x = a:h:b;
  w = [alpha w];

endfunction
```

Listing 2: nonlinfindiff.m

```
/************************************************************
Program 10. Uses the piecewise linear finite-element method
to find an approximate solution of a two-point BVP of the
form
              -[p(x) y']' + q(x) y = f(x), a<=x<=b
               y(a)=alpha, y(b)=beta

Inputs:
  BVPeval Function to evaluate p,q,f and g,g' (BC func)
  a,b Interval params
  alpha,beta Boundary value params
  N Number of interior grid pts (N+2 total pts)
  x Grid points: x(j), j=0...N+1

Outputs:
  x Grid points: x(j), j=0...N+1
  y Approx soln: y(j), j=0...N+1


Note 1: For any given problem, the function BVPeval must
be changed. This function computes p,q,f and g,g' for
any given x. The function g is the BC function
defined by

        g(x) = alpha + (x-a)[(beta-alpha)/(b-a)].
        g'(x) = (beta-alpha)/(b-a).

Note 2: For any given problem, the values of N and x(j),
j=0...N+1 must be specified. The default choice for the
grid points is x(j) = a + jh, where h=(b-a)/(N+1).

Note 3: The midpoint quadrature rule is used to approximate
the FE integrals. Gauss elimination is used to solve the
FE equations.

Note 4: To compile this program use the command (all on
one line)

  c++ -o program10 matrix.cpp gauss_elim.cpp
                        linearfem.cpp program10.cpp

Note 5: The program output is written to a file.
```

```
************************************************************/
#include <iostream>
#include <iomanip>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include "matrix.h"
using namespace std;

/*** Define output file ***/
const char myfile[20]="program10.out" ;
ofstream prt(myfile) ;

/*** Declare external function ***/
int linearfem(int, vector&, vector&) ;

/*** Define p(x), q(x), f(x), g(x), g'(x) ***/
void BVPeval(const double& x, double& p, double& q,
                         double& f, double& g, double& dg){

  double pi=4.0*atan(1.0) ;
  double a=-2, b=2, alphaBC=0, betaBC=0 ;

  g = alphaBC + (x-a)*((betaBC-alphaBC)/(b-a)) ;
  dg = (betaBC-alphaBC)/(b-a) ;

  double gamma = 486;
  double beta = 37;
  p = x<0 ? 0.1 : 0.2;
  q = fabs(x)<=0.5 ? 0 : beta;
  f = fabs(x)<=0.4 ? gamma : 0;
}

int main() {
  /*** Define problem parameters ***/
  int N=7, success_flag ;
  vector x(N+2), y(N+2) ;
  double a=-2, b=2, h=(b-a)/(N+1) ;

  /*** Define FE grid pts ***/
  for(int j=0; j<=N+1; j++){
    x(j) = a + j*h ; //default
  }

  /*** Call linear FE method ***/
  success_flag=linearfem(N,x,y) ;

  /*** Print results to output file ***/
  prt.setf(ios::fixed) ;
  prt << setprecision(5) ;
  cout << "Linear-FE: output written to " << myfile << endl ;
  prt << "Linear-FE results" << endl ;
  prt << "Number of interior grid pts: N = " << N << endl ;
  prt << "Approximate solution: x_j, y_j" << endl ;
  for(int j=0; j<=N+1; j++){
    prt << setw(8) << x(j) ;
    prt << " " ;
    prt << setw(8) << y(j) ;
    prt << " " ;
    prt << endl;
  }
  return 0 ; //terminate main program
}
```

Listing 3: program10.cpp