NAME:

MATH 353: Engineering Mathematics III

Spring'13 Spring 2013 final exam questions

1. Write the output of the following command lines in Matlab:

```
linspace(0,2,5)
```

```
2e3 + 1
```

```
10*2.^(1:5)
```

```
f = @(t) 1./t.^2+1;
sum( f(0:0.5:2) )
```

```
c=[0.1 0.4 0.3 -1 0.6];
perm=[3 2 1 5 4];
c(perm)
```

```
A=[2 1 3;1 5 3;1 1 1;2 1 4];
A(1,:)
A(:,2)
A(3,1)
```

2. Here's a piece of code that is almost finished

```
function [r,hist]=unknown(f,fp,x0,tol,itmax)

% [r,hist]=unknown(f,fp,x0,tol,itmax)
% Input:
%    Explanations
% Output:
%     Explanations

r = x0; hist = r;
for i=1:itmax
    d=f(r)/fp(r);
    r=r-d;
    hist=[hist r];
    if abs(d) < tol
        return
    end
end
display('Message')    % Line to be completed
return
```

Three questions on this piece of code:

(a) What method does this function correspond to? What is it that we are trying to solve?

(b) What (exactly) would happen if we type `help unknown`?

(c) The `'Message'` line is to be completed. What is the message you should write instead on the word `Message`. It has to make sense in the context that we are trying to use this.

3. Consider the points

$$(1,1), \qquad (2,2), \qquad (3,3), \qquad (2,2).$$

Write down the interpolating polynomial using Lagrange's formula.

4. Consider the points

$$(1,0), \qquad (0,-1), \qquad (2,3), \qquad (-1,0).$$

Compute the divided differences and write the interpolating polynomial in Newton's form.

What is the degree of this polynomial? What does this say about the original points?

5. Show that

$$\frac{f(x_0+h)-f(x_0-h)}{2h} = f'(x_0) + \mathcal{O}(h^2).$$

6. Find the degree of precision of the formula

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{2}{3}(f(-1)+f(0)+f(1)).$$

7. Apply the midpoint, trapezoid and Simpson rules (the simple ones) to approximate the following integral

$$\int_{0}^{1} (x^3 - x^2)\mathrm{d}x.$$

Is any of the three approximations the exact value of the integral? Why? (**Hint.** You don't need to compute the exact value of the integral to answer this question.)

8. Here's the code of the composite Simpson rule.

```
function integral=simpsonrule(f,interval,m)

% Shf=simpsonrule(f,[a b],m)
%
% Input:
%     f     : vectorized function of one variable
%     [a b] : vector with limits of integration interval
%     m     : number of subdivisions
% Output
%     Shf   : approx of \int_a^b f(x) dx with composite Simpson rule

x=linspace(interval(1),interval(2),m+1);
h=(interval(2)-interval(1))/m;
mdpt=0.5*(x(2:end)+x(1:end-1));
integral=(h/6)*(f(x(1))+2*sum(f(x(2:end-1)))+f(x(end))+4*sum(f(mdpt)));

return
```

   (a) Modify it to be the composite midpoint rule.

   (b) Modify it to be the composite trapezoid rule.

9. Apply one step of the midpoint method with time-step $h = 0.1$ to the differential equation

$$(y+t^2)y' + y^2 = 0, \qquad t \geq 0, \qquad y(0) = 1.$$

10. Here is the code of Heun's method.

```
function [w,t]=heun(f,tinit,tfinal,yinit,n)

% [w,t]=heun(f,tinit,tfinal,yinit,n)
% Input:
```

```
%     f      : function of two variables
%     tinit  : initial time
%     tfinal : final time
%     yinit  : initial value (at t=tinit)
%     n      : number of time-steps
% Output:
%     w      : vector with n+1 components w_i \approx y(t_i)
%              (Heun's method for y'=f(t,y), y(tinit)=yinit
%     t      : vector with n+1 components (times)

t=linspace(tinit,tfinal,n+1);
h=(tfinal-tinit)/n;
w=zeros(1,n+1);       % room to store the solution
w(1)=yinit;           % initial value
for i=1:n
    k1=f(t(i),w(i));
    k2=f(t(i)+h,w(i)+h*k1);
    w(i+1)=w(i)+h*0.5*k1+h*0.5*k2;
end
return
```

(a) Modify it for the midpoint method.

(b) Modify it for Euler's method.

11. We have run three different methods for a differential equation in the interval $0 \leq t \leq 1$. We have computed the maximum error as a function of $h = 1/n$, where n is the number of time steps. We run experiments with $n = 1, 2, 4, 8, 16$. Each of the three columns contains all the experiments for a method and we are expecting

$$E_h = \max_{0 \leq i \leq n} |w_i - y(t_i)| = \mathcal{O}(h^p)$$

($y(t)$ is the exact solution, $t_i = 0 + i\,h$, and $w_i \approx y(t_i)$ is the approximated solution). What is p for each column/method?

```
0.569560557758917   0.000579323417547   0.140859085770477
0.267769111808837   0.000037013462701   0.035649264005780
0.129466548687958   0.000002326240851   0.008940076098471
0.063618163463359   0.000000145592846   0.002236763705256
0.031529636406385   0.000000009102726   0.000559300120949
```

12. Write the differential equation

$$y'' - 2yy' + \sin t = 0, \qquad y(0) = 1, \qquad y'(0) = 2,$$

as a system of first order differential equations.

13. We are given the following matrix decomposition PA = LU,

$$\mathrm{P} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \mathrm{A} = \begin{bmatrix} -2 & 2 & 0 \\ 0 & 6 & 3 \\ 2 & 1 & 1 \end{bmatrix}$$

$$\mathrm{L} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \qquad \mathrm{U} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 3 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Using it, solve the linear system

$$\begin{bmatrix} -2 & 2 & 0 \\ 0 & 6 & 3 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 3 \\ 5 \end{bmatrix}.$$

(**Hint.** The system $\mathrm{A}\mathbf{x} = \mathbf{b}$ is equivalent to the system $\mathrm{LU}\mathbf{x} = \mathrm{P}\mathbf{b}$.)

14. Write an algorithmic description (that is, write the iteration with mathematically meaningful formulas) of the following iterative process:

```
x=1;
for i=1:3e+2
    x=0.5*(x+4/x);
end
```

(I'm asking also how many iterations you get to do when you run this piece of code.)

15. What algorithm does this piece of code correspond to?

```
x=zeros(n,1);
for i=n:-1:1
    x(i)=(b(i)-R(i,i+1:n)*x(i+1:n))/R(i,i);
end
```

16. Working by hand, compute two iterations for Newton's method trying to approximate a root of $f(x) = x^6 - 2$ starting at $x_0 = 1$.

17. Two different methods, trying to compute a particular number r we know, produce iteratively $x_1, x_2, \ldots$. We compute and tabulate errors

$$e_i^{\mathrm{Meth}\#\ell} = |x_i^{\mathrm{Meth}\#\ell} - r| \qquad \ell = 1, 2$$

```
errMethod1 =
   0.020746217944410
   0.006076689632185
   0.001779827797216
   0.000521299658575
   0.000152685139140
```

```
errMethod2 =
   1.354248688935409
   0.229248688935409
   0.009139993283235
   0.000015733125699
   0.000000000046779
```

(a) Show that the first list corresponds to a method with linear convergence. What is the rate of convergence?

(b) Show that the second list of errors corresponds to a method with quadratic convergence.

18. Working by hand, apply nested evaluation of the polynomial

$$2 - 4(x-2) + 4(x-2)(x+5) + 6(x-2)(x+5)x,$$

at the points $x = 1$ and $x = 4$.

19. Consider the points

$$(1,1), \qquad (2,2), \qquad (3,1), \qquad (5,0).$$

(a) Write down the interpolating polynomial using Lagrange's formula. Do not simplify the result!

(b) Evaluate the previous formula at the point $x = 0$.

(c) Compute the divided differences and write the interpolating polynomial in Newton's form. Do not simplify the result!

(d) Evaluate the last polynomial you got in $x = 0$.

If you got everything right, the values at $x = 0$ of both formulas (for each problem) should be the same. Why?

20. Show that

$$\frac{-f(x_0+2h) + 4f(x_0+h) - 3f(x_0)}{2h} = f'(x_0) + \mathcal{O}(h^2)$$

21. Find the degree of precision of the formula

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx f(-1/\sqrt{3}) + f(1/\sqrt{3}).$$

22. Apply the midpoint, trapezoid and Simpson rules (the simple ones) to approximate the following integral

$$\int_0^1 (2x^3 + x^2 - 1)\mathrm{d}x.$$

Is any of the three approximations the exact value of the integral? Why? (**Hint.** You don't need to compute the exact value of the integral to answer this question.)

23. We have used two different methods to compute the solution of an initial value problem:

$$y' = f(t,y), \quad a \le t \le b, \qquad y(a) = y_a.$$

We have done this for $n = 10, 20, 40, 80$ and 160 time steps. Then we have computed the errors:

$$E_h^{\text{Meth\#1}} = \max_{0\le j\le n} |w_j^{\text{Meth\#1}} - y(t_j)| \qquad \text{and} \qquad E_h^{\text{Meth\#2}} = \max_{0\le j\le n} |w_j^{\text{Meth\#2}} - y(t_j)|.$$

These are the errors.

```
Errors Method 1     Errors Method 2
0.015207614655732   0.121784073781251
0.004037913819598   0.059953565850001
0.001037371962928   0.028020725164368
0.000262757612076   0.013566946929654
0.000066134187187   0.006677021808841
```

Here is what you have to explain: one of the methods is Euler method and the other one is the midpoint method. Which is which and why?

24. Apply two steps of the midpoint method with time-step $h = 0.1$ to the differential equation

$$(y^2 + t^2)y' + 2y = 0, \qquad t \geq 0, \qquad y(0) = 1.$$

25. We want to use the code `rk4` to approximate the solution of

$$u\,u'' + u^2\,(t+1) - u' = 0, \qquad 0 \leq t \leq 2, \qquad u(0) = 1, \quad u'(0) = 0.$$

How do you have to write the problem so that you can apply the method? Use the Matlab function `rk4` (provided in the website) with 100 timesteps. Copy all the instructions you have needed to use and a sketch of he plot of the approximate solution $(y, u(t))$.

26. Knowing that this code

```
x=zeros(n,1);
for i=n:-1:1
    x(i)=(b(i)-R(i,i+1:n)*x(i+1:n))/R(i,i);
end
```

solves an upper triangular system with back substitution, write the code to solve a lower triangular system with (forward) susbtitution.

27. What does the following piece of code do?

```
N=4;
e=ones(N,1);
A=spdiags([2*e -e e],[0 -1 1],N,N);
A=full(A)
```

Using the command `spdiags`, write the instructions needed to construct the $N \times N$ tridiagonal matrix

$$\begin{bmatrix} 4 & 1 & & & \\ -1 & 4 & 1 & & \\ & -1 & 4 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 4 \end{bmatrix}$$

for any given N, as a sparse matrix. In the case $N = 4$, what exactly is stored in Matlab?

28. You are given a function and a collection of points

```
r = @(x) x.^2+2;
x = (1:100)./101;
```

Write what the values of the points x_i are, and construct, in only one line of Matlab, a sparse diagonal matrix with the values

$$\begin{bmatrix} r(x_1) & & & \\ & r(x_2) & & \\ & & \ddots & \\ & & & r(x_{100}) \end{bmatrix}$$

(Hint. The instructions to create sparse matrices are given in the previous problem.)

29. The following code corresponds to the forward Finite Difference method applied to solve the problem

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \qquad a < x < b, \quad 0 < t < T,$$

with initial condition $u(x,0) = u_0(x)$ for all $a \leq x \leq b$ and boundary conditions $u(a,t) = l(t)$ and $u(b,t) = r(t)$ for $0 < t \leq T$. Write the modifications you need to carry out to use the **backward Finite Difference method** for a slightly modified differential equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - R\,u.$$

```
function [U,x,t]=heatForwardFD(D,interval,T,u0,left,right,N,M)

% Help lines not shown

h=(interval(2)-interval(1))/(N+1);
x=interval(1)+h*(0:N+1);
k=T/M;
t=k*(0:M);

U=zeros(N+2,M+1);
U(:,1)=u0(x);

A=[-ones(N,1) 2*ones(N,1) -ones(N,1)];
A=k*D/h^2*spdiags(A,[-1 0 1],N,N);
int=2:N+1;
BC = @(t) k*D/h^2*[left(t);zeros(N-2,1);right(t)];

for n=1:M
    U(1,n+1)  =left((n+1)*k);
```

```
    U(N+2,n+1)=right((n+1)*k);
    U(int,n+1)=U(int,n)-A*U(int,n)+BC(n*k);
end
```