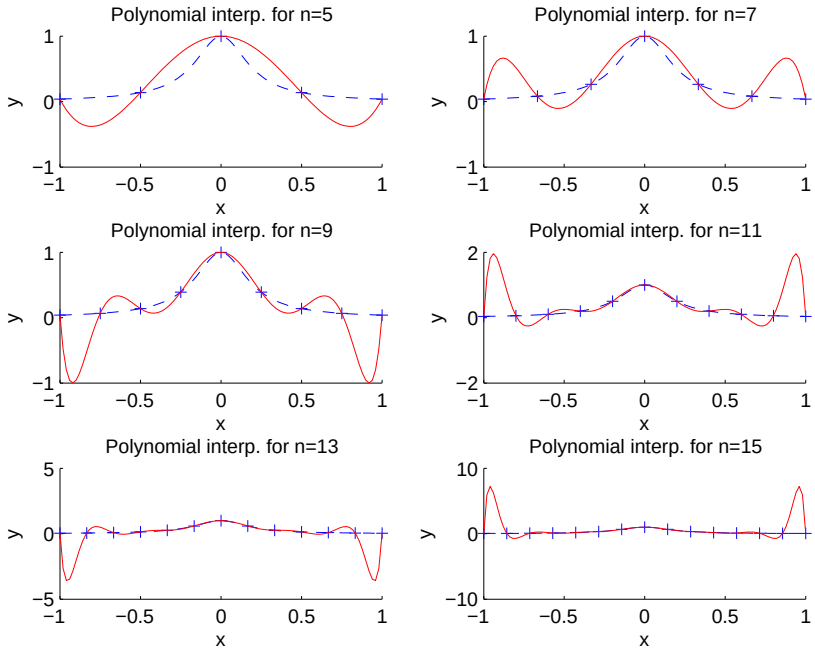
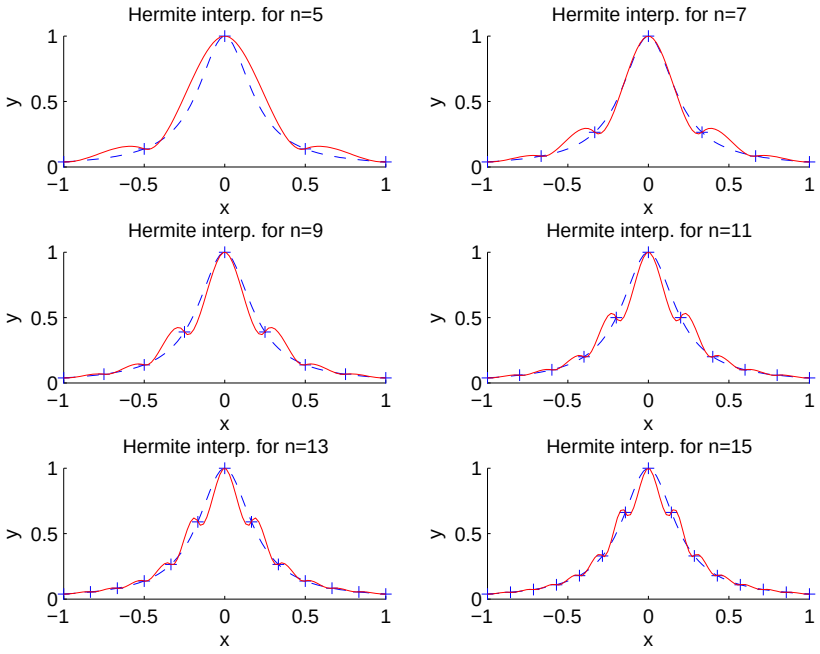
Peter Woodman
0127119
Amath 352 - B. Bale
Homework 8

1. (a)
```
>> ns = 5:2:15;
>> ps = ['r' 'b' 'g' 'c' 'm' 'y'];
>> for i = 1:6
subplot(3,2,i)
runge(ns(i),'r')
xlabel('x');ylabel('y');title(sprintf('Runge interp. for n=%i',ns(i)));
end

ans =

   0.43827287461341
   0.61640156864203
   1.04507821637813
   1.91543426968000
   3.61170159780476
   7.18929847207105
```

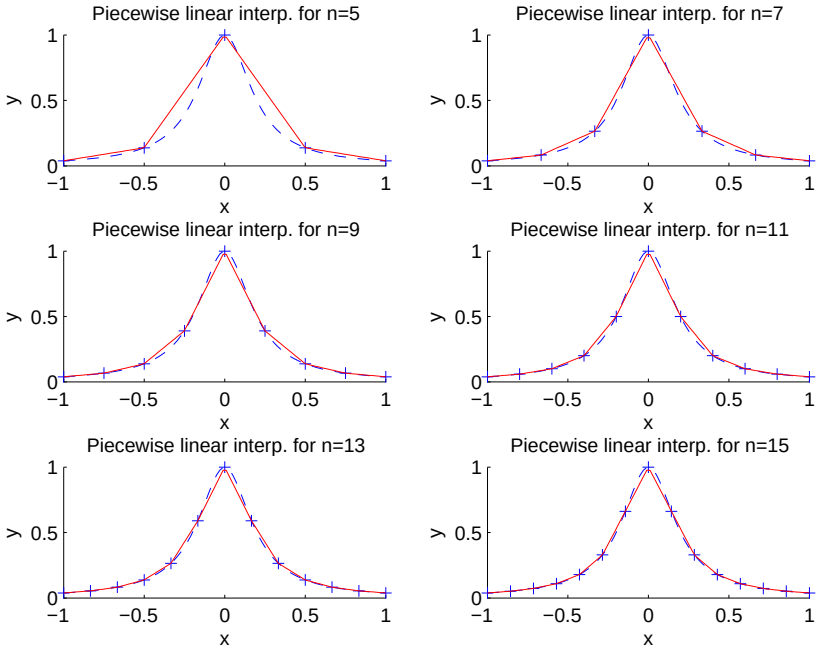


(b) i.
```
ans =

   0.17427359657640
   0.10218389932811
   0.12183153770087
   0.13638365919522
   0.13939853713347
   0.13096004051298
```



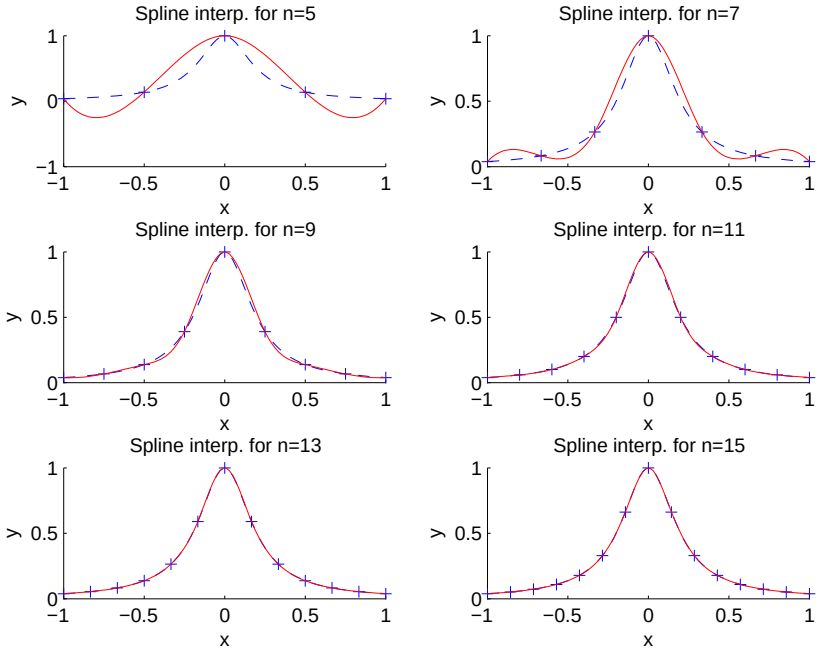
ii. `ans =`

```
0.18006958160436
0.06187301559285
0.06323676188648
0.06631633813679
0.06424645901773
0.05949133131178
```



iii. `ans =`

```
0.31637838556228
0.13240449263880
0.05565201556895
0.02175679160524
0.00679052527923
0.00247620491592
```

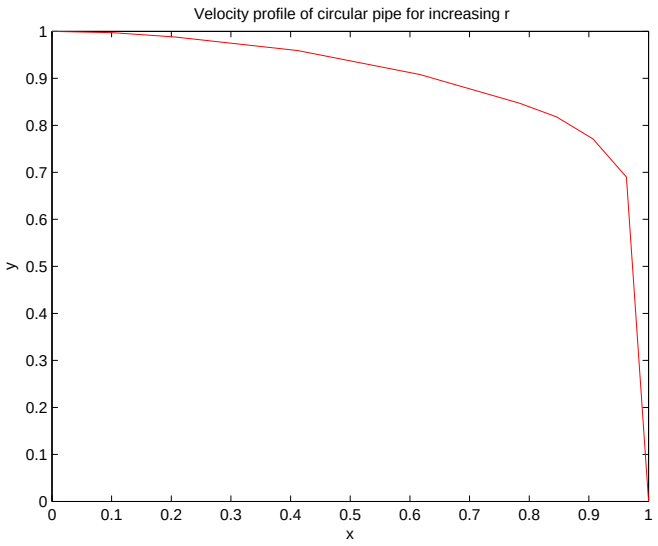



```
>> one = temp(1);two = temp(2);three = temp(3);four=temp(4);
>> [one two three four]

ans =

   0.43827287461341   0.17427359657640   0.18006958160436   0.31637838556228
   0.61640156864203   0.10218389932811   0.06187301559285   0.13240449263880
   1.04507821637813   0.12183153770087   0.06323676188648   0.05565201556895
   1.91543426968000   0.13638365919522   0.06631633813679   0.02175679160524
   3.61170159780476   0.13939853713347   0.06424645901773   0.00679052527923
   7.18929847207105   0.13096004051298   0.05949133131178   0.00247620491592
```

For $n = 5$, Hermite interpolation is best.
For $n = 7$, linear is best.
For all the rest, cubic spline interpolation is best.

2. (a) When $r = R$, $u/u_c = 0$.

```
>> r = [vprofile(:,1)' 1]'; u = [vprofile(:,2)' 0]';
>> plot(r,u);
```



(b)

$$\bar{U} = \frac{1}{\pi R^2} \int_0^R u 2\pi r dr$$
$$\bar{U} = \int_0^R \frac{u 2\pi r}{\pi R^2} dr$$
$$\bar{U} = \int_0^1 u 2n dn$$
$$\frac{\bar{U}}{u_c} = 2 \int_0^1 \frac{u}{u_c} n dn$$

(c)
```
function [avg] = trap(x,y,uc)
u_c = 30.5;
total  = 0;
for k = 1:9
        total = total + (x(k+1)-x(k))*(y(k)*x(k)+y(k+1)*x(k+1));
end;
avg = total * u_c;
```

```
>> trap(r,u,30.5)

ans =

  25.48701142950000
```

3. For the composite trapezoidal method, error is calculated as follows:

$$\epsilon = -\frac{(b-a)f''(c)}{12} h^2$$

where

$$h = \frac{b-a}{M}$$

Combining these two equations, we get:

$$\epsilon = -\frac{(b-a)^3 f''(c)}{12M^2}$$

Defining $\mathsf{max}(f''(c))$ as the largest possible value for $f''(c)$ on the interval $a \leq c \leq b$, we can rewrite the equation as follows:

$$M^2 > \frac{(b-a)^3|\mathsf{max}(f''(c))|}{12\epsilon}$$

The minimum number of subintervals to guarantee a given degree of error $< \epsilon$ is as follows:

$$M > \sqrt{\frac{(b-a)^3|\mathsf{max}(f''(c))|}{12\epsilon}}$$

(a) As the range of $f''(x) = -cos(x)$ is $[-1, 1]$, our max is 1.

$$M > \sqrt{(\pi/6 + \pi/6)^3 \cdot 1} 12 \cdot 5 \cdot 10^{-9} = 4374.89$$

$$h = \frac{b-a}{M} = 2.394 \cdot 10^{-4}$$

Minimum M is 4375, $h = 2.394 \cdot 10^{-4}$.

(b)

$$f''(x) = \frac{-2x}{(5-x)^3} + \frac{1}{(5-x)^2}$$

Max is 0.5.

$$M > \sqrt{\frac{(3-2)^3 \cdot 0.5}{12 \cdot 5 \cdot 10{-}9}} = 2886.75$$

$$h = \frac{b-a}{M} = \frac{3-2}{2887} = 3.464 \cdot 10^{-4}$$

(c)

$$f''(x) = -2e^{-x} + xe^{-x}$$

Max here is 2.

$$M > \sqrt{\frac{(2-0)^3 \cdot 2}{12 \cdot 5 \cdot 10^{-9}}} = 8164.96$$

$$h = \frac{2-0}{8165} = 2.449 \cdot 10^{-4}$$

4. (a)

$$v(x_n) = x^2 + \frac{1}{60}(x^2 v_0 + (4x^2 + 2)v_1 + (x^2 + 1)v_2)$$

(b)

$$\begin{aligned} x_0 = 0 \qquad & v_0 - -\frac{1}{30}v_1 + \frac{1}{60}v_2 = 0 \\ x_1 = 1 \qquad & \frac{241}{240}v_0 - \frac{3}{60}v_1 - \frac{5}{240}v_2 = \frac{-1}{4} \\ x_2 = 2 \qquad & \frac{61}{60}v_0 - \frac{1}{10}v_1 - \frac{1}{30}v_2 = \frac{1}{4} \end{aligned}$$

The particular x values are because we have already defined definitions of $v(x)$ at these points.

(c)
```
>> V = [1 -2/60 -1/60; -1/240 57/60 -5/240; -1/60 -1/10 29/30];
>> y = [0 1/4 1]';
>> V \ y

ans =

   0.02729754322111
   0.28662420382166
```

```
1.06460418562329
```

$$v(x_n) = x^2 + \frac{1}{60}(x^2(v_0 + 4v_1 + v_2) + 2v_1 + v_2)$$
$$v(x_n) = 1.03730664240218 \cdot x_2 + 0.02729754322111$$

(d) Substituting our approximated solution back into the original equation, we can intergrate and simplif as follows-

$$v(x) = x^2 + \frac{1}{10}\int_0^1 (x^2 + 1)v(t)dt$$
$$v(x) = x^2 + \frac{1}{10}\int_0^1 (x^2 + 1)(1.03730664240218x_2 + 0.02729754322111)dt$$
$$v(x) = 1.03730664240218x^2 + 0.03730664240218$$

This is very similar to the Simpson's rule solution, showing that it's somewhat stable. The constant is a bit higher, but it's most likely within error parameters for Simpson's rule (directly calculating the expected error of this function is not something that I feel is particularly worth doing at this juncture).

CODE: - *runge.m* -

```
function ret = runge(n,s)

% ret = 1/(1+25.*x.^2);

x = linspace(-1,1,n);
r = 1./(1+25.*x.^2);
rp = (50.*x)./(1+25.*x.^2).^2;
p = polyfit(x,r,length(x)-1);
xh = linspace(-1,1,100);
rh = 1./(1+25.*xh.^2);
if (s == 1)
    r_poly = newtint(x,r,xh);
%   xlabel('x');ylabel('y');title(sprintf('Polynomial interp. for n=%i',n));
elseif (s == 2)
    r_poly = hermint(x,r,rp,xh)';
%   xlabel('x');ylabel('y');title(sprintf('Hermite interp. for n=%i',n));
elseif (s == 3)
    r_poly = interp1(x,r,xh,'linear');
%   xlabel('x');ylabel('y');title(sprintf('Linear interp. for n=%i',n));
else
    r_poly = interp1(x,r,xh,'spline');
%   xlabel('x');ylabel('y');title(sprintf('Spline interp. for n=%i',n));
end
hold on;
plot(x, r, '+');
plot(xh, rh, 'b--');
plot( xh, r_poly, 'r');
ret = norm(rh - r_poly, inf);
```