EN.553.481/681 Numerical Analysis – Homework 5 Solutions

Problem 1. (a) Note that

$$-\frac{\mathrm{d}}{\mathrm{d}x}x^n e^{-x} = x^n e^{-x} - nx^{n-1}e^{-x} = e^{-x}\left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)x^n.$$

By noting that $\mathrm{d}/\mathrm{d}x$ and $1 - \mathrm{d}/\mathrm{d}x$ commute we can extend the above to

$$\left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^n x^n e^{-x} = e^{-x}\left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^n x^n.$$

With the binomial expansion we can write

$$\begin{aligned} p_n(x) &= e^x \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^n x^n e^{-x} \\ &= \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^n x^n \\ &= \sum_{k=0}^{n} \binom{n}{k} \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^k x^n \\ &= \sum_{k=0}^{n} \binom{n}{k} (-1)^k \frac{n!}{(n-k)!} x^{n-k}. \end{aligned}$$

(b) Without loss of generality assume $n > m$. Then:

$$\begin{aligned} \int_0^\infty p_n(x)p_m(x)e^{-x}\,\mathrm{d}x &= \int_0^\infty e^x \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^n \left\{x^n e^{-x}\right\} \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m e^{-x}\right\} \mathrm{d}x \\ &= \int_0^\infty \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^n \left\{x^n e^{-x}\right\} \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \mathrm{d}x \\ &= \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-1} \left\{x^n e^{-x}\right\} \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \Bigg|_0^\infty \\ &\qquad + \int_0^\infty \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-1} \left\{x^n e^{-x}\right\} \frac{\mathrm{d}}{\mathrm{d}x} \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \mathrm{d}x. \end{aligned}$$

In this last step we used integration by parts. Note that when expanded $(-\mathrm{d}/\mathrm{d}x)^{n-1}x^n e^{-x}$ has a polynomial term of degree at least 1 and a factor e^{-x} in each summand, implying it vanishes at $x = 0, \infty$. Thus:

$$\left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-1} \left\{x^n e^{-x}\right\} \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \Bigg|_0^\infty = 0.$$

We repeat integration by parts:

$$\begin{aligned} \int_0^\infty p_n(x)p_m(x)e^{-x}\,\mathrm{d}x &= \int_0^\infty \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-1} \left\{x^n e^{-x}\right\} \frac{\mathrm{d}}{\mathrm{d}x} \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \mathrm{d}x \\ &= \int_0^\infty \left(-\frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-2} \left\{x^n e^{-x}\right\} \left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^2 \left(1 - \frac{\mathrm{d}}{\mathrm{d}x}\right)^m \left\{x^m\right\} \mathrm{d}x, \end{aligned}$$

and continue until

$$\begin{aligned}\int_0^\infty p_n(x)p_m(x)e^{-x}\,\mathrm{d}x &= \int_0^\infty x^n e^{-x}\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^n\left(1-\frac{\mathrm{d}}{\mathrm{d}x}\right)^m x^m\,\mathrm{d}x\\ &= \int_0^\infty x^n e^{-x}\left(1-\frac{\mathrm{d}}{\mathrm{d}x}\right)^m\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^n x^m\,\mathrm{d}x\\ &= 0.\end{aligned}$$

Problem 2. (a) We have: $p_0(x) = 1$, $p_1(x) = x - 1$, $p_2(x) = x^2 - 4x + 2$, and $p_3(x) = x^3 - 9x^2 + 18x - 6$.

(b) The roots of p_2 are $2 \pm \sqrt{2}$.

(c) We have: $\ell_1(x) = -(x - x_2)/2\sqrt{2}$ and $\ell_2(x) = (x - x_1)/2\sqrt{2}$. Then

$$\begin{aligned}w_1 &= \int_0^\infty \ell_1(x)\exp(-x)\,\mathrm{d}x = -\frac{1-x_2}{2\sqrt{2}} = \frac{1}{2}\left(1+\frac{1}{\sqrt{2}}\right)\\ w_2 &= \int_0^\infty \ell_2(x)\exp(-x)\,\mathrm{d}x = \frac{1-x_1}{2\sqrt{2}} = \frac{1}{2}\left(1-\frac{1}{\sqrt{2}}\right).\end{aligned}$$

(d) We find that

$$\begin{aligned}w_1 + w_2 &= \int_0^\infty e^{-x}\,\mathrm{d}x = 1\\ w_1x_1 + w_2x_2 &= \int_0^\infty xe^{-x}\,\mathrm{d}x = 1.\end{aligned}$$

With $x_1 = 2 - \sqrt{2}$ and $x_2 = 2 + \sqrt{2}$ from (b) this system of equations lets us recover the same w_1, w_2 in (c).

(e) See Program 2. We find that $I_2(f) \approx 0.6634$. Compare with

$$\begin{aligned}I(f) &= \int_0^\infty e^{-x}e^{-x/2}\,\mathrm{d}x\\ &= \int_0^\infty e^{-\frac{3}{2}x}\,\mathrm{d}x\\ &= (-\frac{2}{3}e^{-\frac{3}{2}x})_{x=0}^\infty\\ &= \frac{2}{3}\end{aligned}$$

Problem 3. (a) See Program 3.1.

(b) See Program 3.2 for (i). We compute $I \approx 0.8591$. The Gauss–Legendre method made 66 calls, while Romberg made 257 calls. This reflects the exponential rate of convergence of Gaussian quadrature for C^∞ functions. However, in terms of wall clock time, the Gauss–Legendre method took approximately 4.8457 times the number of seconds it took the Romberg method to compute.

See Program 3.3 for (ii). We compute $I \approx 0.5300$. The Gauss–Legendre method made 3486 calls, while Romberg made 131073 calls. Both methods have slower convergence for this singular integrand, but the asymptotic order of convergence of Romberg is here only 1.9, while Gauss-Legendre has asymptotic convergence rate 2 times as large, or 3.8. Thus, the many function calls made by Romberg to achieve higher-order accuracy

are wasted here and Gauss-Legendre quadrature is much more efficient. However, the Gauss–Legendre method in wall-clock time took approximately 6.5728 times the number of seconds required by Romberg method!

There is a large time loss from having to perform, for each successive choice of n, an eigendecomposition of an n-dimensional tridiagonal matrix in the Gauss–Legendre method. This can be avoided by pre-calculating the weights and nodes needed in Gauss-Legendre quadrature for the various n values and storing them in a file, since these numbers are the same for all integrands. The weights and nodes can then be read in when computing the approximate integrals and not recomputed each time. This will greatly reduce the wall-clock time and make Gauss-Legendre much more efficient than Romberg.

Problem 4. (a) See Program 4.1. We find the Konrod weights $w_0' \approx 0.45092$, $w_1' \approx 0.26849$, $w_2' \approx 0.40140$, and $w_3' \approx 0.10466$.

(b) See Program 4.2.

(c) See Program 4.3 for (i). The exact integral is $I = 2^{11}/11 \approx 186.1818$. The G_3 integral is approximately 172.9664 with error -0.0710. The K_7 integral is approximately 186.1818 with error 2.2204×10^{-16}. The G_7 integral is approximately 186.1818 with error 2.2204×10^{-16}. Note that K_7 has precision 10 and and G_7 has precision 13, so that both of these methods integrate x^{10} exactly (in infinite-precision arithmetic). This explains their spectacular accuracy here.

See Program 4.4 for (ii). The exact integral is $I = e^1 - e^{-1} \approx 2.3504$. The G_3 integral is approximately 2.3503 with error -2.7850×10^{-5}. The K_7 integral is approximately 2.3504 with error 2.5291×10^{-13}. The G_7 integral is approximately 2.3504 with error -1.1102×10^{-15}.

See Program 4.5 for (iii). The exact integral is $I = 2C(1) \approx 1.5598$ where C is the cosine Fresnel integral. The G_3 integral is approximately 1.5420 with error -0.0114. The K_7 integral is approximately 1.5598 with error -3.0731×10^{-6}. The G_7 integral is approximately 1.5598 with error -3.6583×10^{-7}.

By inspection, G_7 is the most accurate. However, it must compute 7 new function values every time it is employed, while K_7 can reuse the 3 function values already computed for G_3. This makes the Kronrod pair (G_3, K_7) much more efficient for adaptive methods even though the pair of Gaussian rules (G_3, G_7) would achieve slightly higher accuracy.

Problem 5.* (a) Recall that

$$e^{xt} = \sum_{n=0}^{\infty} \frac{x^n t^n}{n!}, \qquad e^{-\frac{1}{2}t^2} = 1 + \sum_{k=1}^{\infty} (-1)^k \frac{t^{2k}}{2^k k!}.$$

Multiplying these series termwise, we see that the power x^n first appears multiplying t^n with coefficient $\frac{1}{n!}$ and thereafter appears with other coefficients multiplying t^{n+2k} for $k = 1, 2, 3, \ldots$. Therefore, if we combine all of the terms t^n for the same power n, then the coefficient multiplying t^n is $\frac{1}{n!}x^n$ plus linear combinations of x^{n-2}, x^{n-4},..., 1 (n odd) or 0 (n even). In other words,

$$e^{xt-\frac{1}{2}t^2} = \sum_{n=0}^{\infty} p_n(x) \frac{t^n}{n!}$$

where $p_n(x)$ is a monic polynomial of degree n, in fact containing only even powers of x for n even and only odd powers for n odd. This polynomial $p_n(x)$ is the so-called *Hermite polynomial*, usually denoted $H_n(x)$.
(b) Note that if you take $t = 0$ you immediately obtain $H_0(x) = 1$. From here we write

$$\frac{\mathrm{d}}{\mathrm{d}x} \exp\left(xt - \frac{1}{2}t^2\right) = t \exp\left(xt - \frac{1}{2}t^2\right) = \sum_{n=0}^{\infty} H_n(x) \frac{t^{n+1}}{n!} = \sum_{n=0}^{\infty} (n+1) H_n(x) \frac{t^{n+1}}{(n+1)!} = \sum_{n=1}^{\infty} n H_{n-1}(x) \frac{t^n}{n!}$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\sum_{n=0}^{\infty} H_n(x)\frac{t^n}{n!} = \sum_{n=0}^{\infty} H_n'(x)\frac{t^n}{n!} = \sum_{n=1}^{\infty} H_n'(x)\frac{t^n}{n!},$$

The two rows are equal. This gives us $H_n'(x) = nH_{n-1}(x)$. Next write

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\exp\left(xt - \frac{1}{2}t^2\right) &= (x-t)\exp\left(xt - \frac{1}{2}t^2\right) \\
&= \sum_{n=0}^{\infty} xH_n(x)\frac{t^n}{n!} - \sum_{n=0}^{\infty} H_n(x)\frac{t^{n+1}}{n!} \\
&= \sum_{n=0}^{\infty} xH_n(x)\frac{t^n}{n!} - \sum_{n=0}^{\infty} (n+1)H_n(x)\frac{t^{n+1}}{(n+1)!} \\
&= \sum_{n=0}^{\infty} xH_n(x)\frac{t^n}{n!} - \sum_{n=1}^{\infty} nH_{n-1}(x)\frac{t^n}{n!} \\
&= \sum_{n=0}^{\infty} xH_n(x)\frac{t^n}{n!} - \sum_{n=0}^{\infty} nH_{n-1}(x)\frac{t^n}{n!} \\
&= \sum_{n=0}^{\infty} \left[xH_n(x) - nH_{n-1}(x)\right]\frac{t^n}{n!}
\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{n=0}^{\infty} H_n(x)\frac{t^n}{n!} = \sum_{n=1}^{\infty} H_n(x)\frac{t^{n-1}}{(n-1)!} = \sum_{n=0}^{\infty} H_{n+1}(x)\frac{t^n}{n!}.$$

As before the two above expressions are equal, giving us $H_{n+1}(x) = xH_n(x) - nH_{n-1}(x)$.

(c) Let $\mathbf{e}_n := (0, \ldots, -\sqrt{n}) \in \mathbf{R}^n$ and write

$$p_{n+1}(x) = \det\left(x\mathbf{I}_{n+1} - \mathbf{A}_{n+1}\right) = \det\begin{bmatrix} x\mathbf{I}_n - \mathbf{A}_n & \mathbf{e}_n \\ \mathbf{e}_n^\top & x \end{bmatrix}$$

Now expand in minors along the last row or last column to obtain

$$p_{n+1}(x) = x\det\left(x\mathbf{I}_n - \mathbf{A}_n\right) + \sqrt{n}\det\begin{bmatrix} x\mathbf{I}_{n-1} - \mathbf{A}_{n-1} & \mathbf{e}_{n-1} \\ \mathbf{0}_{n-1}^\top & -\sqrt{n} \end{bmatrix}$$

Next expand the final determinant along its bottom row to obtain

$$\begin{aligned}
p_{n+1}(x) &= x\det\left(x\mathbf{I}_n - \mathbf{A}_n\right) - n\det\left(x\mathbf{I}_{n-1} - \mathbf{A}_{n-1}\right) \\
&= xp_n(x) - np_{n-1}(x),
\end{aligned}$$

as desired. Note we immediately have $p_1(x) = x = H_1(x)$ and also $p_2(x) = xp_1(x) - 1$, consistent with $p_0(x) = 1 = H_0(x)$. Since the characteristic polynomials $p_n(x)$ satisfy the same recurrence relation as do the Hermite polynomials and also agree for $n = 0, 1$, it follows that $p_n(x) = H_n(x)$ for *all* $n \geq 0$. In particular, the eigenvalues of $\mathbf{A}_n$ are precisely the roots of the Hermite polynomial $H_n(x)$.

(d) See Program 5.1.

(e) See Program 5.2. See Figure 5 for the semi-log error plot. The approximate linear decrease is evidence of an exponential rate of convergence. Approximate values listed below with errors for each n.

n	Gauss–Hermite Approximation	Error
1	1.0000	-0.6487
3	1.6382	-0.0105
5	1.6487	-0.0000
7	1.6487	-0.0000
9	1.6487	-0.0000
11	1.6487	-0.0000
13	1.6487	-0.0000
15	1.6487	-0.0000

`MATLAB` Plots

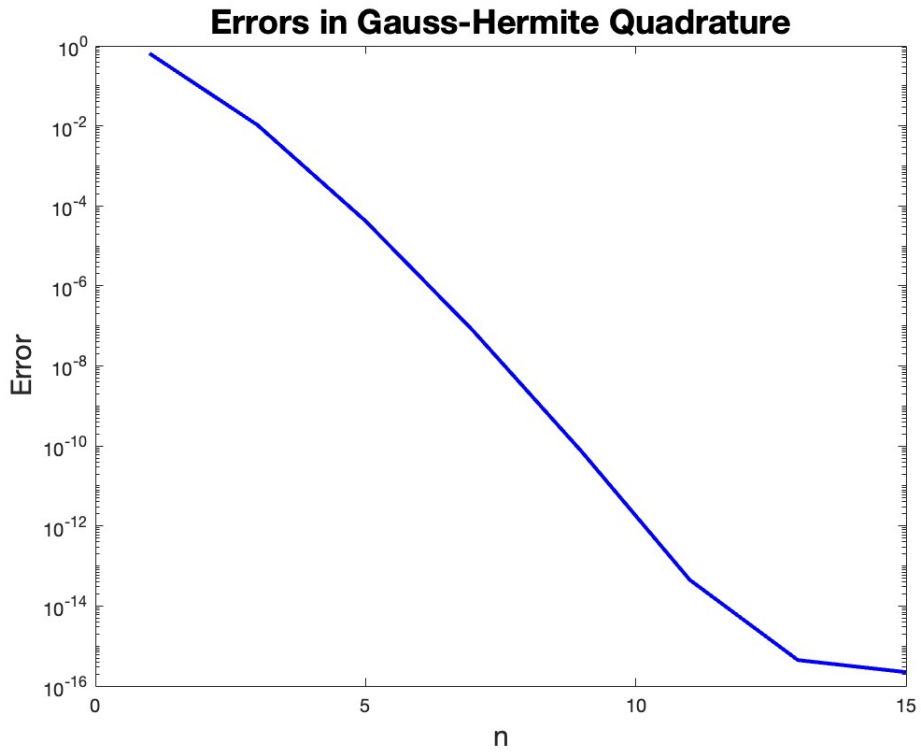

Figure 5. Semi-log error in Gauss–Hermite quadrature.

MATLAB Code

Program 2. Gauss–Laguerre quadrature.

```
x=2*[1 1]+sqrt(2)*[-1 1];

w=[1 1]+[1 -1]/sqrt(2);
w=w/2;

f=@(x) exp(-x/2);

I=sum(w.*f(x))
```

Program 3.1. Implementation of gaussleg2.m.

```
function [I, fcnt]=gaussleg2(f,a,b,tol)

itmax=400;

n=1;
I=(b-a)*f((a+b)/2);
fcnt=1;
E=1;

while E>tol
     n=n+1;
     if n>itmax
       break
     end

Iold=I;

% calculate zeros & weights by the Gollub-Welsch algorithm

A=zeros(n,n);
for i=2:n
   A(i-1,i)=(i-1)/sqrt((2*i-1)*(2*i-3));
end
A=A+A';

[V,D]=eig(A);

x=diag(D);

for j=1:n,
    v=V(:,j);
    w(j,1)=2*v(1)^2;
end

% Gauss-Legendre approximation of the integral

x=((b-a).*x+(a+b))./2;

I=sum(w.*f(x))*(b-a)/2;
fcnt=fcnt+n;
E=abs(I-Iold)/abs(I);

end
```

Program 3.2. Gauss–Legendre versus Romberg for $f(x) = x\exp(x^2)$.

```
f=@(x) x.*exp(x.^2);
a=0; b=1; tol=1e-15;
I=(exp(1)-1)/2

timeGL=0; timeR=0;
for ii=1:1000

    tic
    [IGL,fcntGL]=gaussleg2(f,a,b,tol);
    timeGL=timeGL+toc;

    tic
    [IR,fcntR]=romberg(f,a,b,tol);
    timeR=timeR+toc;
end

IGL=IGL, fcntGL=fcntGL
IR=IR, fcntR=fcntR

tratio=timeGL/timeR
```

Program 3.3. Gauss–Legendre versus Romberg for $f(x) = (x(1-x))^{1/3}$.

```
f=@(x) (x.*(1-x)).^(1/3);
a=0; b=1; tol=1e-7;
I=beta(4/3,4/3)


timeGL=0; timeR=0;
for ii=1:1000

    tic
    [IGL,fcntGL]=gaussleg2(f,a,b,tol);
    timeGL=timeGL+toc;

    tic
    [IR,fcntR]=romberg(f,a,b,tol);
    timeR=timeR+toc;
end

IGL=IGL, fcntGL=fcntGL
IR=IR, fcntR=fcntR

tratio=timeGL/timeR
```

Program 4.1. Finding the Kronrod weights by finding the linear system given by the constraint that K_7 exactly integrates x^{2i} for $i = 0, \ldots, 3$.

```
xg=[-sqrt(.6),0,sqrt(.6)]; xg1=xg(3);
wg=[5,8,5]/9;

% use the quadratic formula
```

```
b=-10/9; c=155/891; d=sqrt(b^2-4*c);
xk1=sqrt((-b-d)/2); xk2=sqrt((-b+d)/2);

% half of the integral of x^{2k} is 1/(2k+1)
A=[1/2 1 1 1; 0 xg1^2 xk1^2 xk2^2; 0 xg1^4 xk1^4 xk2^4; 0 xg1^6 xk1^6 xk2^6];
b=[1; 1/3; 1/5; 1/7];
wk=A\b
```

Program 4.2. Implementation of `G3K7.m`.

```
function [Ig,Ik] = G3K7(f)

xg=[-sqrt(.6),0,sqrt(.6)]; xg1=xg(3);
wg=[5,8,5]/9;

b=-10/9; c=155/891; d=sqrt(b^2-4*c);
xk2=sqrt((-b-d)/2); xk3=sqrt((-b+d)/2);

A=[1/2 1 1 1; 0 xg1^2 xk2^2 xk3^2; 0 xg1^4 xk2^4 xk3^4; 0 xg1^6 xk2^6 xk3^6];
b=[1; 1/3; 1/5; 1/7];
wk=A\b;

wkg=[wk(2) wk(1) wk(2)];
xkp=[-xk3 -xk2 xk2 xk3];
wkp=[wk(4) wk(3) wk(3) wk(4)];

fg=f(xg);
Ig=sum(wg.*fg);
Ik=sum(wkg.*fg)+sum(wkp.*f(xkp));

end
```

Program 4.3. Comparing G_3, K_7, and G_7 for $f(x) = (x+1)^{10}$.

```
xw=glquad(7)
xg7=xw(:,1);
wg7=xw(:,2);

f=@(x) (x+1).^10
I=2^11/11

[Ig,Ik]=G3K7(f);
Ig=Ig
Eg=Ig/I-1

Ik=Ik
Ek=Ik/I-1

Ig7=sum(wg7.*f(xg7))
Eg7=Ig7/I-1
```

Program 4.4. Comparing G_3, K_7, and G_7 for $f(x) = \cosh(x)$.

```
xw=glquad(7)
xg7=xw(:,1);
wg7=xw(:,2);
```

```

f=@(x) cosh(x)
I=exp(1)-exp(-1)

[Ig,Ik]=G3K7(f);
Ig=Ig
Eg=Ig/I-1

Ik=Ik
Ek=Ik/I-1

Ig7=sum(wg7.*f(xg7))
Eg7=Ig7/I-1
```

Program 4.5. Comparing G_3, K_7, and G_7 for $f(x) = \cos(\frac{\pi}{2}x^2)$.

```
xw=glquad(7)
xg7=xw(:,1);
wg7=xw(:,2);

f=@(x) cos(pi*x.^2/2)
I=2*fresnelc(1)

[Ig,Ik]=G3K7(f);
Ig=Ig
Eg=Ig/I-1

Ik=Ik
Ek=Ik/I-1

Ig7=sum(wg7.*f(xg7))
Eg7=Ig7/I-1
```

Program 5.1. Implementation of `ghquad.m`.

```
function I=ghquad(f,n)

% calculate improper integral by Gaussian quadrature with weight function
% w(x)=exp(-x^2/2)/sqrt(2*pi) with nodes & weights by the Gollub-Welsch algorithm

A=zeros(n,n);
for i=2:n
   A(i-1,i)=sqrt(i-1);
end
A=A+A';

[V,D]=eig(A);

[x,in]=sort(diag(D));

for j=1:n,
    v=V(:,j);
    w(j)=v(1)^2;
end

w=w(in)';
```

```
22
23 I=sum(f(x).*w);
24
25 return
```

Program 5.2. Approximation of $I(f)$ by Gauss–Hermite quadrature.

```
1  f=@(x) cosh(x);
2  I=exp(1/2);
3
4  for i=1:8
5      N(i)=2*i-1;
6      IH(i)=ghquad(f,N(i));
7      EH(i)=IH(i)-I;
8  end
9
10 [N',IH.',EH.']
11
12 semilogy(N,abs(EH),'-b','LineWidth',2)
13 xlabel('n','FontSize',15)
14 ylabel('Error','FontSize',15)
15 title('Errors in Gauss-Hermite Quadrature','FontSize',18)
```