PHY5340 Laboratory 4: Gaussian Quadrature

Table of Contents

Question 1 1
Question 2 2
Question 3 3
Question 4 3
Question 5 4

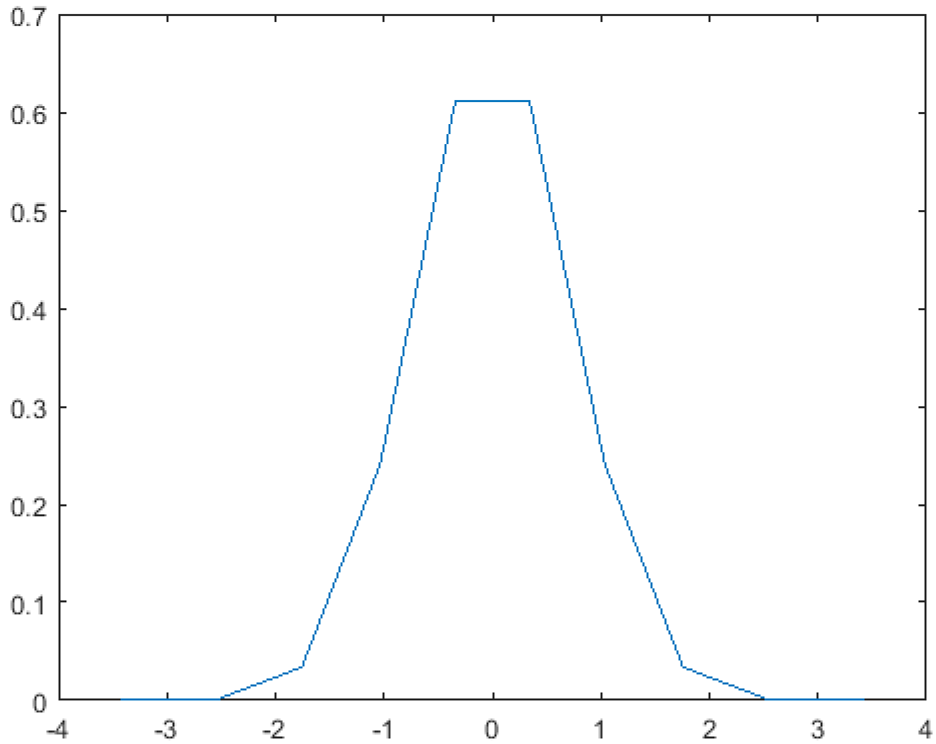
Jeremiah O'Neil, SN6498391

Question 1

```
type('gauss_hermite.m')
type('golubwelsch.m')
[x, w] = gauss_hermite(10);
plot(x, w)


function [x, w, a, b, q0] = gauss_hermite( N )
% Define recursion coefficients and obtain nodes and weights for order
 N
%  Gauss-Hermite quadrature.
a = sqrt((1:1:(N-1))/2);
b = zeros(1, N);
q0 = pi^(-1/4);
[x, w] = golubwelsch(a, b, q0);
end

function [ x, w ] = golubwelsch( a, b, q0 )
% Obtain nodes x and weights w for gaussian quadrature given
 polynomial
%  recursion coefficients a and b, and normalization q0.
% Using the Golub-Welsch algorithm; implementation adapted from C9_1.m
J = diag(a, -1) + diag(a, 1) + diag(b);
[V, D] = eig(J);
x = diag(D)';
w = V(1, :).^2/q0^2;
end
```



Under the change of variables x0 = sqrt(hbar / (m * w)), H = hbar * w / 2 * (x^2 / x0^2 + x0^2 * d^2/dx^2); and for units of length such that x0 = 1, H = hbar * w / 2 * (x^2 + d^2/dx^2).

Question 2

In units x0 = 1, the integrand is just c^2 times the Gauss-Hermite weight function; so, I'll just integrate f = 1 (vector of length N). I'll obtain the exact result with the minimum order N = 1, since f is a zeroeth order polynomial (ie., constant). The exact result is of course c = q0 = pi^(-1/4).

```
[x, w] = gauss_hermite(1);
f2 = @(x)ones(1, length(x));
type('gaussquad.m')
c = 1/sqrt(gaussquad(x, w, f))
```

```
function [ num ] = gaussquad( x, w, f )
% Integrate f by gaussian quadrature rule with nodes x and weights w.
num = f(x)*w.';
end

c =

    0.7511
```

Question 3

Removing the factor hbar * w, performing the derivatives, and pulling out the Gauss-Hermite weight exp(-x^2), the function under Gauss-Hermite integration is c^2 / 2 --- once again, a constant. Order N = 1 will give the exact answer.

```
f3 = @(x)f2(x) * c^2 / 2;
gaussquad(x, w, f3)
```

```
ans =

    0.5000
```

Question 4

Now the function isn't constant --- or even polynomial! I won't be able to get an exact result. I'll use a function to compute Gauss-Hermite quadrature iteratively to the desired relative tolerance, 1E-4.

```
type('gaussquad_hermite.m')
f4 = @(x)exp(-2 * x.^4);
targetF = @(num)1/sqrt(num);
[d, N] = gaussquad_hermite(f4, targetF, 1E-4)
```

```
function [ num, N, rel_tol ] = gaussquad_hermite( f, targetF, tol )
% Integrate f by Gauss-Hermite quadrature of increasing order until
%  specified relative tolerance is achieved in the function targetF.
N = 1; rel_tol = 1;
[x, w] = gauss_hermite(N);
num = targetF(gaussquad(x, w, f));
while rel_tol > tol,
    num_pre = num;
    N = N + 1;
    [x, w] = gauss_hermite(N);
    num = targetF(gaussquad(x, w, f));
    rel_tol = abs(num_pre/num - 1);
end
end

d =

    0.8990


N =

    20
```

Question 5

As above.

```
f5 = @(x)(16*x.^6 + 8*x.^4 - 10*x.^2 - 1) .* f4(x) * d^2 / 2;
targetF = @(num)num;
[num, N] = gaussquad_hermite(f5, targetF, 1E-4)
```

```
num =

   -0.7089


N =

    31
```

Published with MATLAB® R2016a