Homework 9 Solution
PHZ 5156, Computational Physics
November 9, 2005

Problem 1

(a) I will refer to integrals in Gradshteyn and Ryzhik using letters GR. The appropriate orthogonality integral for the Legendre polynomials is GR 7.221:

$$\int_{-1}^{1} dx\, P_n(x) P_m(x) = \frac{2}{2n+1}\delta_{nm}.$$

(a) Starting from

$$f(x) = \sum_{n=0}^{\infty} c_n P_n(x),$$

you multiply both sides by $P_m(x)$ and integrate from –1 to 1. This produces

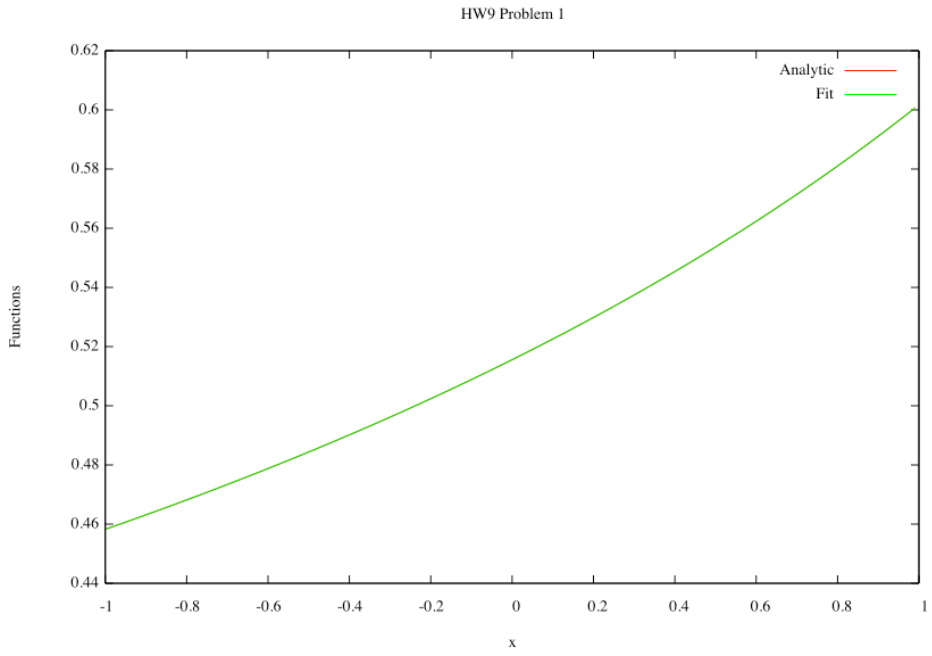
$\int_{-1}^{1} dx\, P_m(x) f(x) = \sum_{n=0}^{\infty} c_n \int_{-1}^{1} dx\, P_n(x) P_m(x) = \frac{2}{2m+1} c_m$. Thus

$$c_m = \frac{2m+1}{2}\int_{-1}^{1} dx\, P_m(x)\left(\cosh 2 - x\right)^{-1/2} = \frac{2\mathrm{m}+1}{2}\left\{\frac{2\sqrt{2}}{2m+1} e^{-(2m+1)}\right\} = \sqrt{2} e^{-(2m+1)}.$$

Here the last integral, given by the expression in braces, is GR 7.225(4) with p=1.

(a) Here is my python code and the resulting plot. The numeric and analytic results coincide.

```
#HW 9, PHZ 5156, October 25, 2005
from scipy import *
import Gnuplot,Gnuplot.funcutils

#Problem 1
x = arange(-1.,1.,0.01)
y = 1./sqrt(cosh(2.)-x)  # Original function
fit = zeros(len(x))
for n in arange(5):
        cn =sqrt(2.)*exp(-(2*n+1.))
        pnx = special.legendre(n)(x)
        fit = fit + cn*pnx
g=Gnuplot.Gnuplot(debug=0)
g.title('HW9 Problem 1')
g.xlabel('x')
g.ylabel('Functions')
g('set data style lines')
g1 = Gnuplot.Data(x,y,title='Analytic')
g2 = Gnuplot.Data(x,fit,title='Fit')
g.plot(g1,g2)
```


HW9 Problem 1
Analytic
Fit
Functions
x
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Problem 2

(a) The appropriate orthonogonality integral for the Laguerre polynomials is GR 7.414(3) with α=0 [because $L_m^0(x)=L_m(x)$]. Using $\Gamma(n+1)=n!$, this gives:

$$\int_0^\infty dx\, e^{-x} L_n(x) L_m(x) = \delta_{nm}.$$

(b) Starting from

$$f(x) = \sum_{n=0}^{\infty} c_n L_n(x),$$

you multiply both sides by $e^{-x}L_m(x)$ and integrate from 0 to ∞. This produces

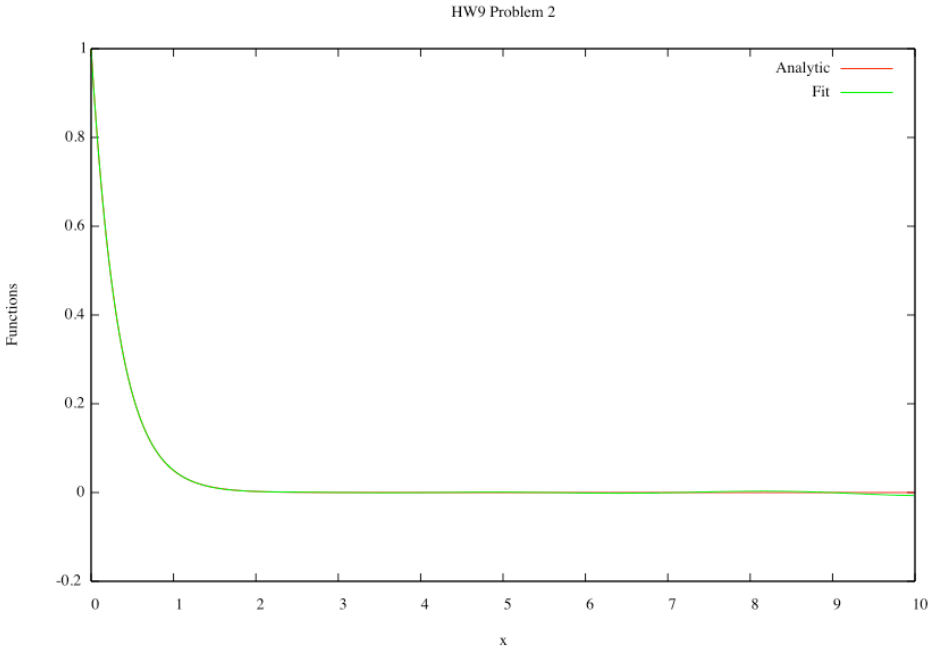
$$\int_0^\infty dx\, e^{-x} L_m(x) f(x) = \sum_{n=0}^{\infty} c_n \int_0^\infty dx\, e^{-x} L_n(x) L_m(x) = c_m. \text{ Thus}$$

$$c_m = \int_0^\infty dx\, e^{-4x} L_m(x) = 3^n 4^{-n-1}.$$

Here the last integral is GR 7.414(6) with b=4.

(c) Here is the relevant piece of my python code and the resulting plot.

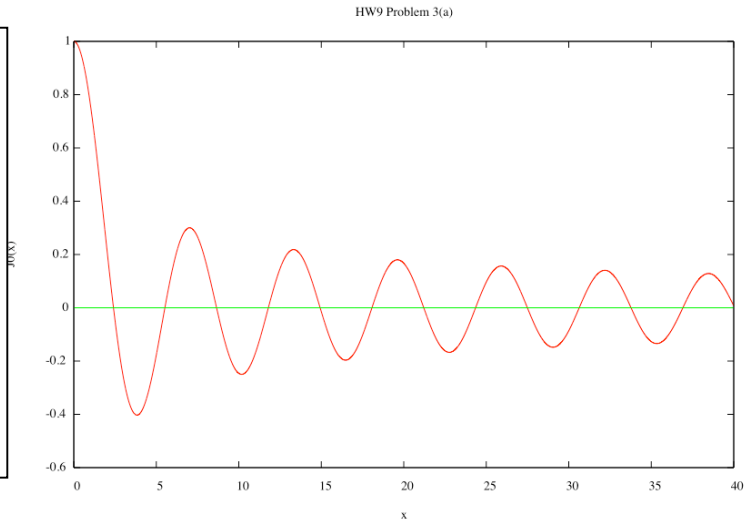
```
#Problem 2
x = arange(0.,10.,.01)
y = exp(-3*x)    # Original function
fit = zeros(len(x))
for n in arange(25):
        cn =0.25 * (0.75 ** n)
        lnx = special.laguerre(n)(x)
        fit = fit + cn*lnx
```



Problem 3

(a) Here is the code and plot of $J_0(x)$.

```
#Problem 3a
x = arange(0,40.,.01)
y = special.jn(0,x)
g=Gnuplot.Gnuplot(debug=0)
g.title('HW9 Problem 3(a)')
g.xlabel('x')
g.ylabel('J0(x)')
g('set data style lines')
g1 = Gnuplot.Data(x,y)
zeroline = 0.*x
g2 = Gnuplot.Data(x,zeroline)
print "about to plot 3a"
g.plot(g1,g2)
```



(b) Code and output:

```
#Problem 3b
s = special.jn_zeros(0,50)
print "Zeros of J0 are\n",s
```

```
Zeros of J0 are
[  2.40482556   5.52007811   8.65372791  11.79153444  14.93091771
  18.07106397  21.21163663  24.35247153  27.49347913  30.63460647
  33.77582021  36.91709835  40.05842576  43.19979171  46.34118837
  49.4826099   52.62405184  55.76551076  58.90698393  62.04846919
  65.1899648   68.33146933  71.4729816   74.61450064  77.75602563
  80.89755587  84.03909078  87.18062984  90.32217264  93.46371878
  96.60526795  99.74681986 102.88837425 106.02993092 109.17148965
 112.31305028 115.45461265 118.59617663 121.73774209 124.87930891
 128.02087701 131.16244628 134.30401664 137.44558802 140.58716035
 143.72873357 146.87030763 150.01188246 153.15345802 156.29503427]
```

(c) The appropriate orthonogonality integral for the 0th order Bessel functions with arguments $s_n x$ is GR 6.521(1) with $\nu=0$, $\alpha=s_n$ and $\beta=s_m$. This gives

$$\int_0^1 dx\, x J_0\left(s_n x\right) J_0\left(s_m x\right) = \delta_{nm} \frac{1}{2}\left[J_1\left(s_n\right)\right]^2 .$$

(d) Starting from

$$f(x) = \sum_{n=1}^{\infty} c_n J_0(s_n x),$$

you multiply both sides by $xJ_0(s_m x)$ and integrate from 0 to 1. This produces

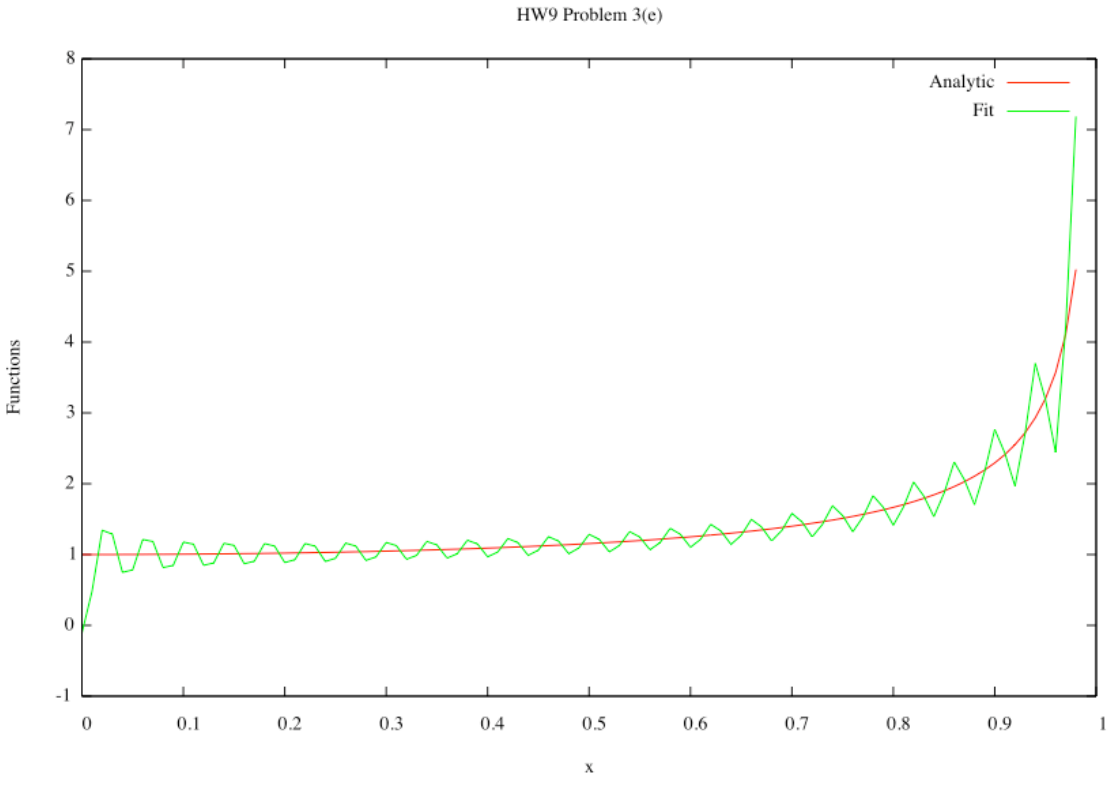
$$\int_0^1 dx\, xJ_0(s_m x) f(x) = \sum_{n=1}^{\infty} c_n \int_0^1 dx\, xJ_0(s_m x) J_0(s_n x) = c_m \frac{1}{2}\left[J_1(s_m)\right]^2 .$$ Thus

$$c_m = \frac{2}{\left[\mathrm{J}_1(s_m)\right]^2} \int_0^1 dx\, x\, J_0(s_m x)\left(1-x^2\right)^{-1/2} = \frac{2}{\left[\mathrm{J}_1(s_m)\right]^2} \frac{\sin s_m}{s_m}.$$

Here the last integral is GR 6.554(2).

(e) Here is the relevant piece of my python code and the resulting plot. The fit isn't superb – I couldn't get python to cooperate with high enough orders to get a smoother fit. (Doing so would require thinking about asymptotics for large n).

```
#Problem 3e
x = arange(0.,0.99,.01)
y = 1./sqrt(1.-x**2)        # Original function
fit = zeros(len(x))
for n in arange(len(s)):
        cn =( 2. / (special.jn(1,s[n])**2) ) * sin(s[n])/s[n]
        j0nx = special.jn(0,s[n]*x)
        fit = fit + cn*j0nx
```



Problem 4

Starting from

$$f(x) = \sum_{n=0}^{\infty} c_n \clubsuit_n(x),$$

you multiply both sides by $e^{-x*x}x^3\clubsuit_m(x)$ and integrate from -∞ to ∞. This produces

$\int_{-\infty}^{\infty} dx\, e^{-x^2} x^3 \clubsuit_m(x) f(x) = \sum_{n=0}^{\infty} c_n \int_0^{\infty} dx\, e^{-x^2} x^3 \clubsuit_n(x) \clubsuit_m(x) = c_m m!$. Thus

$$c_m = \frac{1}{m!} \int_0^{\infty} dx\, e^{-x^2} x^3 \clubsuit_m(x).$$

Problem 5

(a) Starting from $H\psi(x) = E\psi(x),$ with $\hat{H} = -\frac{d^2}{dx^2} + V(x),$ expand the unknown eigenfunction in the basis states: $\psi(x) = \sum_{n=0}^{\infty} c_n u_n(x)$:

$$\sum_{n=0}^{\infty} c_n \hat{H} u_n(x) = E \sum_{n=0}^{\infty} c_n u_n(x).$$

Multiply both sides with $u_m^*(x)$ and integrate from 0 to L. This gives

$$\sum_{n=0}^{\infty} H_{mn} c_n = E c_n \text{ where}$$

$$H_{mn} = \langle u_m | \hat{H} | u_n \rangle = \int_0^L dx\, u_m^*(x) \hat{H} u_n(x).$$

The first line above is equivalent to $Hc=Ec$ where H is the matrix made up of the matrix elements H_{mn}, and c is the column vector made up of the coefficients c_n.

(b) Python code and output:

```
#Homework 9 Problem 5
#Computational Physics PHZ 5156 October 2005
from scipy import *
from LinearAlgebra import *
import Gnuplot,Gnuplot.funcutils

#Set up the constants
L = 11.
w = L/10.
V0 = 50.
nmax = 5

#Calculate V(x)
h = 0.01
x = arange(0.,L+h,h)
Vhat = 0*x
for j in arange(len(x)):
        xj = x[j]
        if abs(xj-L/2)<w/2:
                Vhat[j] = V0 * ( (4./(w**2))*(xj-L/2)**2 - 1)

#Problem 5(b)
#Calculate the matrix elements of V by quadrature using the
#trapezoidal rule
V = zeros((nmax,nmax),Float)
for n in arange(nmax):
        for m in arange(n+1):
                un = sqrt(2./L)*sin((n+1)*pi*x/L)
                um = sqrt(2./L)*sin((m+1)*pi*x/L)
                Vnm = 0.5*un[0]*Vhat[0]*um[0]
                Vnm = Vnm + 0.5*un[-1]*Vhat[-1]*um[-1]
                for j in arange(1,len(x)-1):
                        Vnm = Vnm + un[j]*Vhat[j]*um[j]
                Vnm = h*Vnm
                V[n,m] = Vnm
                if (m<>n): V[m,n] = conjugate(Vnm)
#print "V=\n",round(V,5)

T = zeros((nmax,nmax),Float)
for n in arange(nmax):
        T[n,n] = (pi*(n+1)/L)**2
#print "T=\n",round(T,5)

H = T+V

print "H=\n",round(H,5)
```

Output:

```
H=
[[-6.55178 -0.       6.50365  0.      -6.24968]
 [-0.       0.19657  0.       0.25397 -0.     ]
 [ 6.50365  0.      -5.64528 -0.       6.13599]
 [ 0.       0.25397 -0.       0.80772  0.     ]
 [-6.24968 -0.       6.13599  0.      -3.87401]]
```

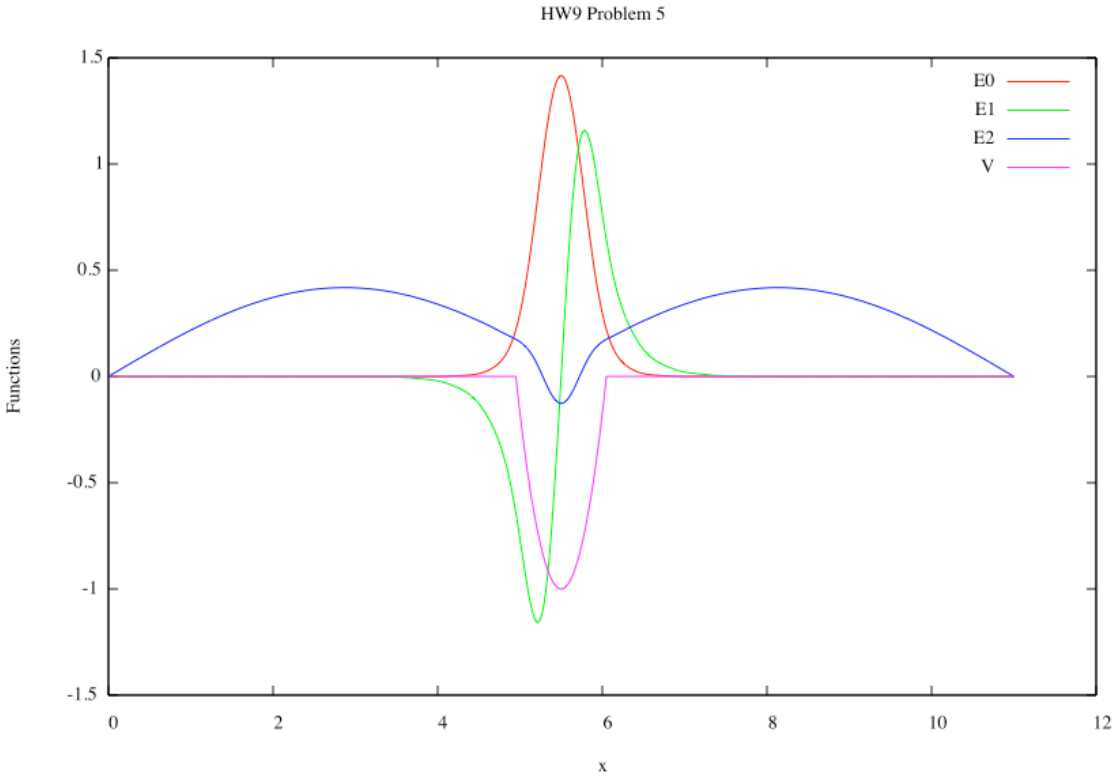
(c) I changed nmax to 50, deleted the statement to print H, and added the following to the end of the code.

```
evals,evecs = eigenvectors(H)

print "evals=\n",evals

psi = zeros( (3,len(x)), Float)
location = array( (7,26,34) )
for i in (0,1,2):
        ii = location[i]
        print "Energy",i,"is",evals[ii]
        for n in arange(nmax):
                cn =evecs[ii][n]
                un = sqrt(2./L)*sin((n+1)*pi*x/L)
                psi[i] = psi[i] + cn*un

g=Gnuplot.Gnuplot(debug=0)
g.title('HW9 Problem 5')
g.xlabel('x')
g.ylabel('Functions')
g('set data style lines')
g1 = Gnuplot.Data(x,psi[0],title='E0')
g2 = Gnuplot.Data(x,psi[1],title='E1')
g3 = Gnuplot.Data(x,psi[2],title='E2')
g4 = Gnuplot.Data(x,Vhat/V0,title='V')
g.plot(g1,g2,g3,g4)
```

This was kind of lazy – I had to run twice, once to identify (by hand) the three lowest eigenvalues; and the second time to plot them. The output and plot are below.

```
evals=
[ 193.59300767  201.69979497  177.65610196  185.46290043  162.49165543
      148.04124542  169.97836182  -37.23044219  134.2859383   155.18122355
      121.21267735  141.05306999  108.80860922  127.58961135   97.06139984
      114.79360356   85.96152542  102.67146781   75.50447904   91.23063719
       65.69163176   80.4770561    56.52952316   70.41279279   48.02817675
       61.03413815  -12.7601928    40.19922876   52.33110342   33.05437611
       44.28941408   26.60427319   36.8951018    20.85775941    0.30027911
        1.21216147    2.76207475   15.82119861    4.97901519    7.88600672
       11.49769768   30.13981887    0.39518938    1.57445871    3.52145609
        6.21495899    9.63282393   13.75359177   18.5568872    24.02404545]
Energy 0 is -37.2304421925
Energy 1 is -12.7601928046
Energy 2 is 0.300279112337
```

It would have been rather more elegant to sort the eigenvalues and pick out the eigenfunctions corresponding to the lowest three eigenvalues (with, say, python's function argsort).



Notice that the two negative-energy states are bound – that is, the eigenfunctions die quickly to zero outside the well. The most negative energy state is bound the tightest. The third state has positive energy, and is unbound – that is, its eigenfunction spread throughout the system. Also notice that the lowest-energy state has no nodes, the next state has one node, and the next two.