MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
CAMBRIDGE, MASSACHUSETTS 02139

2.29 NUMERICAL FLUID MECHANICS— SPRING 2007

Solution of Problem Set 3

Totally 120 points
Posted 04/03/07, due Thursday 4 p.m. 04/19/07, Focused on Lecture 8 to 17

Problem 3.1 (15 points):

Consider the following system of equations:

$$Ax = b, \quad A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 8 & 0 \\ -1 & 0 & 4 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ 8 \\ 4 \end{bmatrix}$$

a) Cholesky factorize A (Note that A is positive definite).
b) Find an LU factorization form for A.
c) Use LU factorization of A to find x.
d) Compute the x by two iterations of successive over-relaxation scheme. Use relaxation parameter $\omega = 1.5$ and initial guess of zero.
e) Compute the solution by 4 iterations of conjugate gradient method.

Solution:

a) We need to find the lower triangular matrix L such that $A = LL^*$. However, since A is positive definite the "L" elements are real and we have $A = LL^* = LL^T$. So we need to solve the below equations:

$$Find \;\; L = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}, \; such \; that \; A = LL^T \; and \; l_{ii} > 0$$

$$\begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{22}^2 + l_{21}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{33}^2 + l_{32}^2 + l_{31}^2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 8 & 0 \\ -1 & 0 & 4 \end{bmatrix}$$

The above equation can be solved very easily:

$$l_{11} = 1, \quad l_{21} = 2, \quad l_{31} = -1$$
$$l_{22} = \sqrt{8 - l_{21}^2} = 2$$
$$l_{32} = -\frac{l_{21} l_{31} + 0}{l_{22}} = -\frac{2 \times -1}{2} = 1$$
$$l_{33} = \sqrt{4 - l_{32}^2 - l_{31}^2} = \sqrt{2}$$

So we have:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ -1 & 1 & \sqrt{2} \end{bmatrix}$$

Alternatively we could use the below formula

$$l_{i,j} = \frac{1}{l_{j,j}} \left(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k} \right), \qquad \text{for } i > j$$
$$l_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2}.$$

b) The Cholesky decomposition is already a "LU" factorization form so[1]:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ -1 & 1 & \sqrt{2} \end{bmatrix}, U = L^T = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 2 & 1 \\ 0 & 0 & \sqrt{2} \end{bmatrix}$$

c) We have to find y such that $Ly = b$ and then we have to find x such that $Ux = y$:

$$Ly = b \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ -1 & 1 & \sqrt{2} \end{bmatrix} y = \begin{bmatrix} 0 \\ 8 \\ 4 \end{bmatrix} \Rightarrow y = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix}$$
$$Ux = y \Rightarrow \begin{bmatrix} 1 & 2 & -1 \\ 0 & 2 & 1 \\ 0 & 0 & \sqrt{2} \end{bmatrix} x = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix} \Rightarrow x = \begin{bmatrix} -4 \\ 2 \\ 0 \end{bmatrix}$$

[1] Otherwise we can use the Gaussian Elimination and find L and U accordingly:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0.5 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 4 & 2 \\ 0 & 0 & 2 \end{bmatrix}$$

d) Since the matrix is positive definite, we do not need to impose diagonally dominant condition. As a result we can use the below format for Gauss-Seidel:

$$Ax = b \Rightarrow \begin{cases} x_1 = -2x_2 + x_3 \\ x_2 = -\dfrac{x_1}{4} + 1 \\ x_3 = \dfrac{x_1}{4} + 1 \end{cases}$$

Also for the relaxation we have:

$$x_i^{(n+1)} = (1-\omega)x_i^{(n)} + \omega\bar{x}_i^{(n+1)},\ \textit{where } \bar{x}_i^{n+1} \textit{ is the "}n+1\textit{" iterate on } x_i \textit{ computed by Gauss-Seidel from } x$$

So we have:

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\bar{x}_1^{(1)} = -2\times 0 + 0 = 0 \Rightarrow x_1^{(1)} = (1-1.5)\times 0 + 1.5\times 0 = 0$$

$$\bar{x}_2^{(1)} = -\frac{0}{4} + 1 = 1 \qquad \Rightarrow x_2^{(1)} = (1-1.5)\times 0 + 1.5\times 1 = 1.5$$

$$\bar{x}_3^{(1)} = +\frac{0}{4} + 1 = 1 \qquad \Rightarrow x_3^{(1)} = (1-1.5)\times 0 + 1.5\times 1 = 1.5$$

$$x^{(1)} = \begin{bmatrix} 0 \\ 1.5 \\ 1.5 \end{bmatrix}$$

$$\bar{x}_1^{(2)} = -2\times 1.5 + 1.5 = -1.5 \quad \Rightarrow x_1^{(2)} = (1-1.5)\times 0 + 1.5\times -1.5 \quad = -2.25$$

$$\bar{x}_2^{(2)} = -\frac{-2.25}{4} + 1 \quad = 1.5625 \Rightarrow x_2^{(2)} = (1-1.5)\times 1.5 + 1.5\times 1.5625 = 1.59375$$

$$\bar{x}_3^{(2)} = +\frac{-2.25}{4} + 1 \quad = 0.4375 \Rightarrow x_3^{(2)} = (1-1.5)\times 1.5 + 1.5\times 0.4375 = -0.09375$$

$$x^{(2)} = \begin{bmatrix} -2.25 \\ 1.59375 \\ -0.09375 \end{bmatrix}$$

e) Mathematically by at most "n=3" iterations we have to find the exact solution. However in practice there will be still errors due to numerical truncations. The good thing about conjugate gradient's method and other iterative methods is that they are selfcorrective and their repeated application decrease the accumulate errors due to numerical truncation (as seen in the program run).

$$v_0 = r_0 = b - Ax_0$$

do

$$\alpha_i = (v_i^\top r_i)/(v_i^\top A v_i)$$

$$x_{i+1} = x_i + \alpha_i v_i$$

$$r_{i+1} = r_i - \alpha_i A v_i$$

$$\beta_i = -(v_i^\top A r_{i+1})/(v_i^\top A v_i)$$

$$v_{i+1} = r_{i+1} + \beta_i v_i$$

until a stop criterion holds

The algorithm follows the above formula and it is implemented in the attached file "C2p29_PSET3_1.m". Here is the output:

```
x0'=0.000000 0.000000 0.000000
r0'=0.00000e+00  8.00000e+00  4.00000e+00
v0'=0.000000 8.000000 4.000000
alpha0'=0.138889
beta0'=0.084105

x1'=0.000000 1.111111 0.555556
r1'=-1.66667e+00  -8.88889e-01  1.77778e+00
v1'=-1.666667 -0.216049 2.114198
alpha1'=0.227941
beta1'=0.126296

x2'=-0.379901 1.061865 1.037467
r2'=-7.06361e-01  2.64885e-01  -5.29771e-01
v2'=-0.916853 0.237599 -0.262757
alpha2'=3.948394
beta2'=0.000000

x3'=-4.000000 2.000000 -0.000000
r3'=1.33227e-15  7.10543e-15  1.77636e-15
v3'=0.000000 0.000000 0.000000
alpha3'=0.122761
beta3'=0.023599

x4'=-4.000000 2.000000 -0.000000
r4'=-3.57755e-16  -1.99830e-16  1.06764e-15
v4'=-0.000000 -0.000000 0.000000
alpha4'=0.225271
beta4'=0.000345

x5'=-4.000000 2.000000 0.000000
r5'=-1.98078e-17  5.13313e-18  -5.67664e-18
v5'=-0.000000 0.000000 -0.000000
alpha5'=4.520056
beta5'=0.000000

x6'=-4.000000 2.000000 -0.000000
r6'=7.91942e-30  3.42415e-29  3.35266e-30
v6'=0.000000 0.000000 0.000000
alpha6'=0.118493
beta6'=0.006300
```

Problem 3.2 (10 points): Polynomial Interpolation

Consider the below (x,y) pairs:

$$x = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 2 \end{bmatrix}, \quad y = f(x) = \begin{bmatrix} 2 \\ 0 \\ 1 \\ -2 \end{bmatrix}$$

a) Find the Lagrange polynomial for above points.
b) Interpolate that polynomial at x=-1.
c) Find the ordered polynomial for above points with Newton's formula.
d) Interpolate the ordered polynomial at x=-1.
e) Find the 3rd order interpolating polynomial with forming a linear system of equations.
f) Interpolate the above polynomial at x=-1.

Solution:

a)

$$\begin{aligned} L(x) &= 2\times\frac{(x-0)(x-1)(x-2)}{(-2-0)(-2-1)(-2-2)} + 0\times\frac{(x-(-2))(x-1)(x-2)}{(0-(-2))(0-1)(0-2)} \\ &+1\times\frac{(x-(-2))(x-0)(x-2)}{(1-(-2))(1-0)(1-2)} - 2\times\frac{(x-(-2))(x-0)(x-1)}{(2-(-2))(2-0)(2-1)} \end{aligned}$$

$$L(x) = -\frac{x(x-1)(x-2)}{12} - \frac{(x+2)x(x-2)}{3} - \frac{(x+2)x(x-1)}{4}$$

$$L(x) = \frac{-2x^3+5x}{3}$$

b)

$$L(-1) = \frac{2-5}{3} = -1$$

c)

$$
\begin{array}{lllll}
x & f(x) & & & \\
-2 & 2 & & & \\
 & & \dfrac{0-2}{0-(-2)}=-1 & & \\
 & & & \dfrac{1-(-1)}{1-(-2)}=\dfrac{2}{3} & \\
0 & 0 & & & \dfrac{(-2)-\frac{2}{3}}{2-(-2)}=-\dfrac{2}{3} \\
 & & \dfrac{1-0}{1-0}=1 & & \\
1 & 1 & & \dfrac{-3-1}{2-0}=-2 & \\
 & & \dfrac{-2-1}{2-1}=-3 & & \\
2 & -2 & & &
\end{array}
$$

$$N(x)=2+(-1)\times(x-(-2))+\frac{2}{3}\times(x-(-2))x+(-\frac{2}{3})\times(x-(-2))x(x-1)$$

$$N(x)=2-1\times(x+2)+\frac{2}{3}\times(x+2)x-\frac{2}{3}\times(x+2)x(x-1)$$

$$N(x)=\frac{-2x^3+5x}{3}$$

d) Note that L(x)=N(x) and they pass from the same 4 pairs of points. Indeed both Newton's scheme and Lagrange' scheme refer to the same ordered polynomial and they are both two different method to find the same polynomial.

$$N(-1)=L(-1)=-1$$

e) We have to find the 3rd order polynomial $p(x)=a_3x^3+a_2x^2+a_1x+a_0$ such that it passes through all our points. This leads to:

$$x=\begin{bmatrix}-2\\0\\1\\2\end{bmatrix}, y=\begin{bmatrix}2\\0\\1\\-2\end{bmatrix}\Rightarrow \begin{bmatrix}(-2)^3 & (-2)^2 & -2 & 1\\ 0 & 0 & 0 & 1\\ 1 & 1 & 1 & 1\\ 2^3 & 2^2 & 2 & 1\end{bmatrix}\begin{bmatrix}a_3\\a_2\\a_1\\a_0\end{bmatrix}=\begin{bmatrix}2\\0\\1\\-2\end{bmatrix}$$

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \frac{-2}{3} \\ 0 \\ \frac{5}{3} \\ 0 \end{bmatrix}, \qquad p(x) = \frac{-2x^3 + 5x}{3} = N(x) = L(x)$$

f) Since $p(x) = N(x) = L(x)$ they will all have the same value. However, in general as the number of points becomes larger the Newton's method happens to be more efficient and it also provides better-localized information. Note that the coefficient matrix that is used in the linear equation is Vandermonde matrix which is basically a bad-conditioned matrix.

Problem 3.3 (35 points): Streamlines

For a uniform inviscid flow passing a sphere with radius "R", the potential field is given by:

$$\phi(r,\theta) = U(r + \frac{R^3}{2r^2})\cos(\theta)$$

Here U is the far field velocity and the far field pressure is zero.

a) Find the velocity field.
b) Find the tangential and normal acceleration of fluid particles.
c) Find the analytical form of the streamline that passes through arbitrary point of (r_0, θ_0). Simplify the relation for the case when $\theta_0 = \frac{\pi}{2}$.
d) Derive an analytical differential equation for the distance increment $ds = ds(r,\theta)$ traveled by a particle fluid at a given position from its velocity components. Note that "s" is the path length traveled by fluid particle.

Now do the following for $r_0 = 1.01R, 1.1R, 1.5R, 3R$:

e) Integrate the streamline differential equation as well as the path length differential equation. For the integration use the fixed step size $\Delta\theta = 0.05$ and continue as long as $\theta \le \pi - \Delta\theta$ *and* $r \le 10R$. Assume that $s(\frac{\pi}{2}) = 0$.
f) Plot the analytical form of streamlines as well as the numerical form obtained in previous part.
g) Plot both "r(θ)" and "s(θ)" for each streamline.
h) Fit a series of splines to your s(θ) discrete points computed at part "e". Then differentiate your fit two times to compute the tangential acceleration and compare it with the analytical value at the same θ.

Solution:

a)

$$\vec{V} = \nabla\phi$$

$$V_r = \frac{\partial\phi}{\partial r} = U(1 - (\frac{R}{r})^3)\cos(\theta)$$

$$V_\theta = \frac{1}{r}\frac{\partial\phi}{\partial\theta} = -U(1 + \frac{1}{2}(\frac{R}{r})^3)\sin(\theta)$$

b) The acceleration can be computed by using the Navier Stokes equation in the spherical coordinate. Here is the symbolic derivation's from Maple (attached Maple file "C2p29_PSET3_3.mw"):

$$\Phi := U \cdot \left(r + \frac{R^3}{2r^2}\right) \cdot \cos(\theta)$$

$$U\left(r + \frac{1}{2}\frac{R^3}{r^2}\right)\cos(\theta)$$

$$V_r := \frac{d}{dr}\Phi$$

$$U\left(1 - \frac{R^3}{r^3}\right)\cos(\theta)$$

$$V_\theta := simplify\left(\frac{1}{r} \cdot \frac{d}{d\theta}\Phi\right)$$

$$-\frac{1}{2}\frac{U\,(2\,r^3 + R^3)\sin(\theta)}{r^3}$$

$$a_r := simplify\left(V_r \cdot \frac{d}{dr}V_r + \frac{V_\theta}{r}\frac{d}{d\theta}V_r - \frac{V_\theta^2}{r}\right)$$

$$\frac{3}{4}\frac{\left(-3\cos(\theta)^2 R^3 + 6\cos(\theta)^2 r^3 - 2\,r^3 - R^3\right)U^2 R^3}{r^7}$$

$$a_\theta := simplify\left(V_r \cdot \frac{d}{dr}V_\theta + \frac{V_\theta}{r}\frac{d}{d\theta}V_\theta + \frac{V_\theta \cdot V_r}{r}\right)$$

$$\frac{3}{4}\frac{U^2\cos(\theta)\sin(\theta)\,R^3\,(4\,r^3 - R^3)}{r^7}$$

By now we have computed the radial and angular components of acceleration. Now we can find the angular and radial components of acceleration by definition of tangential vector.

$$\vec{V} = V_r\hat{e}_r + V_\theta\hat{e}_\theta$$

$$\hat{e}_t = \frac{V_r\hat{e}_r + V_\theta\hat{e}_\theta}{|\vec{V}|},\ where\ |\vec{V}|^2 = V_r^2 + V_\theta^2$$

The normal vector is arbitrary defined such that $\hat{e}_t.\hat{e}_n = 0$.

$$\hat{e}_n = \frac{-V_\theta \hat{e}_r + V_r \hat{e}_\theta}{|\vec{V}|},\ where\ |\vec{V}|^2 = V_r^2 + V_\theta^2$$

$$\hat{e}_n = \frac{-V_\theta \hat{e}_r + V_r \hat{e}_\theta}{|\vec{V}|},\ where\ |\vec{V}|^2 = V_r^2 + V_\theta^2$$

$$\vec{a} = a_r \hat{e}_r + a_\theta \hat{e}_\theta = a_t \hat{e}_t + a_n \hat{e}_n$$
$$a_t = \vec{a}.\hat{e}_t$$
$$a_n = \vec{a}.\hat{e}_n$$

It is not important, to derive the symbolic forms of above values, but just for comparison:

$$a_r := simplify\left(V_r \cdot \frac{d}{dr} V_r + \frac{V_\theta}{r} \frac{d}{d\theta} V_r - \frac{V_\theta^2}{r}\right)$$

$$\frac{3}{4} \frac{\left(-3\cos(\theta)^2 R^3 + 6\cos(\theta)^2 r^3 - 2r^3 - R^3\right) U^2 R^3}{r^7}$$

$$a_\theta := simplify\left(V_r \cdot \frac{d}{dr} V_\theta + \frac{V_\theta}{r} \frac{d}{d\theta} V_\theta + \frac{V_\theta \cdot V_r}{r}\right)$$

$$\frac{3}{4} \frac{U^2 \cos(\theta) \sin(\theta) R^3 \left(4r^3 - R^3\right)}{r^7}$$

c) Note that for the fluid particle passing through point (r,θ), we have[2]:

$$\left.\begin{aligned} V_r &= \frac{dr}{dt} \\ V_\theta &= r\frac{d\theta}{dt} \end{aligned}\right\} \Rightarrow \frac{V_r}{V_\theta} = \frac{1}{r}\frac{dr}{d\theta} \Rightarrow \frac{dr}{d\theta} = r\frac{V_r}{V_\theta}$$

[2] Alternatively you can use streamline equations in spherical coordinate, but note that they are different than their corresponding version in cylindrical coordinate:

$$V_r = \frac{1}{r^2 \sin(\theta)} \frac{\partial \psi}{\partial \theta}$$
$$V_\theta = \frac{-1}{r \sin(\theta)} \frac{\partial \psi}{\partial r}$$

$$\frac{dr}{d\theta} = r\frac{V_r}{V_\theta} = -r\frac{(1-(\frac{R}{r})^3)\cos(\theta)}{(1+\frac{1}{2}(\frac{R}{r})^3)\sin(\theta)}$$

$$\frac{1}{\tan(\theta)}d\theta = \frac{r^3+\frac{1}{2}R^3}{-r(r^3-R^3)}dr$$

$$\frac{1}{\tan(\theta)}d\theta = -\frac{1}{2}(-\frac{1}{r}+\frac{3r^2}{(r^3-R^3)})dr$$

$$\int_{\theta_0}^{\theta} \Downarrow \int_{R_0}^{r}$$

$$\ln(\sin(\theta))\Big|_{\theta_0}^{\theta} = \frac{1}{2}(\ln(r)-\ln(r^3-R^3))\Big|_{r_0}^{r}$$

$$\ln(\sin(\theta))\Big|_{\theta_0}^{\theta} = \frac{1}{2}(\ln(\frac{r}{r^3-R^3}))\Big|_{r_0}^{r}$$

$$\ln(\frac{\sin(\theta)}{\sin(\theta_0)}) = \frac{1}{2}(\ln(\frac{r}{r_0}\frac{r_0^3-R^3}{r^3-R^3}))$$

$$(\frac{\sin(\theta)}{\sin(\theta_0)})^2 = \frac{r}{r_0}\frac{r_0^3-R^3}{r^3-R^3}$$

$$\theta_0 = \frac{\pi}{2} \Rightarrow \sin(\theta)^2 = \frac{r}{r_0}\frac{r_0^3-R^3}{r^3-R^3}$$

d)

$$\left.\begin{aligned} &\frac{ds}{dt} = V \\ &V_\theta = r\frac{d\theta}{dt} \Rightarrow \frac{dt}{d\theta} = \frac{r}{V_\theta} \end{aligned}\right\} \Rightarrow \frac{ds}{d\theta} = r\frac{V}{V_\theta}$$

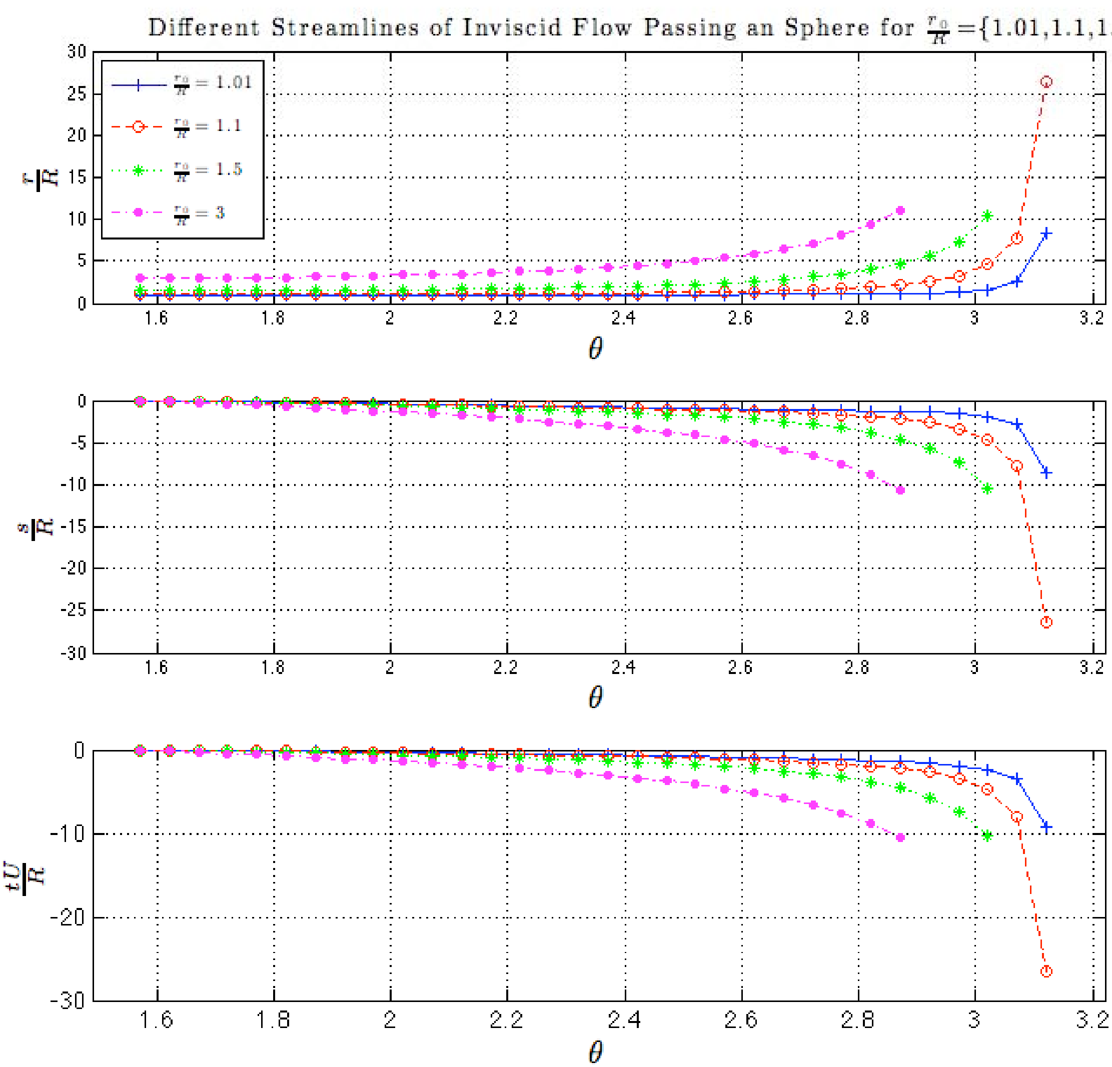
Note that while we have $V = \sqrt{V_\theta^{\,2} + V_r^{\,2}}$, it is not wise to write the relation as $\frac{ds}{d\theta} = r\sqrt{1+(\frac{V_r}{V_\theta})^2}$ or start from general algebra by $\frac{ds}{d\theta} = \sqrt{r^2+(\frac{dr}{d\theta})^2}$. That's because the new relations will introduce some errors whenever V_θ is negative (including our case when indeed $\frac{ds}{d\theta} \le 0$).

e) The below system of differential equations are integrated in the attached file "C2p29_PSET3_3.m", using a fixed step size $\Delta\theta = 0.05$ with Runge-Kutta method. Qunatities are normalized by "U" and "R" values. Apparently even $\Delta\theta = 0.5$ holds fairly accurate results. Plots are shown on the next page. Note that while $\frac{s}{R}$ *and* $\frac{tU}{R}$ graphs are very similar, careful examination shows that they are slightly different.

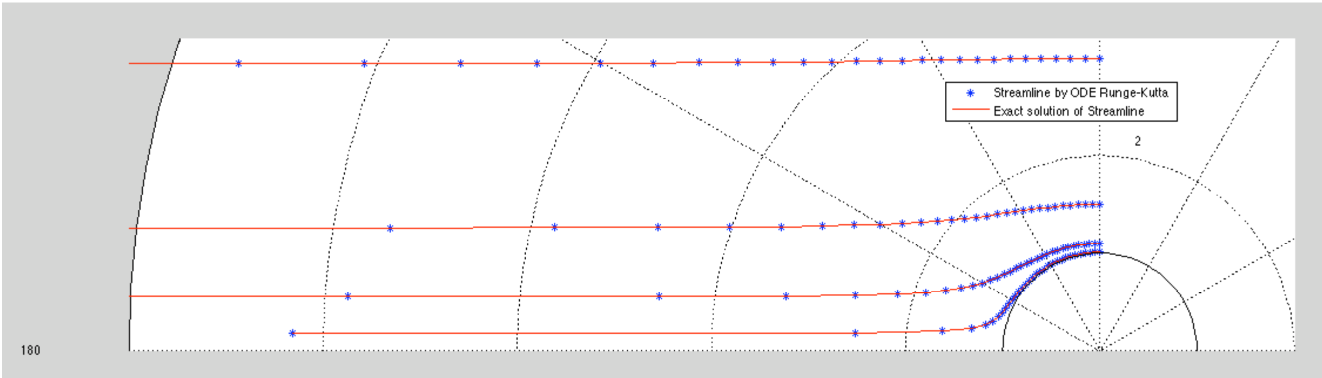
$$\frac{dr}{d\theta} = r\frac{V_r}{V_\theta}$$

$$\frac{ds}{d\theta} = r\frac{V}{V_\theta}$$

$$\frac{dt}{d\theta} = \frac{r}{V_\theta}$$



f) Done within the same file and apparently very accurate.



g) Previous plots.

h) For extra simplicity the time is also integrated, so we can directly compute s(t) and differentiate it to compute the tangential acceleration. Consequently, "s(t)" is fit into a series of splines and two boundary conditions are added for the fit from $\frac{ds}{dt} = V$ at "t" corresponding to $\theta = \frac{\pi}{2}$ *and* $\theta_{\max}$. The results for $a_t = \frac{d^2 s}{dt^2}$ are shown on the next page and they are reasonably accurate.

Alternatively we could differentiate the fit at part "d":

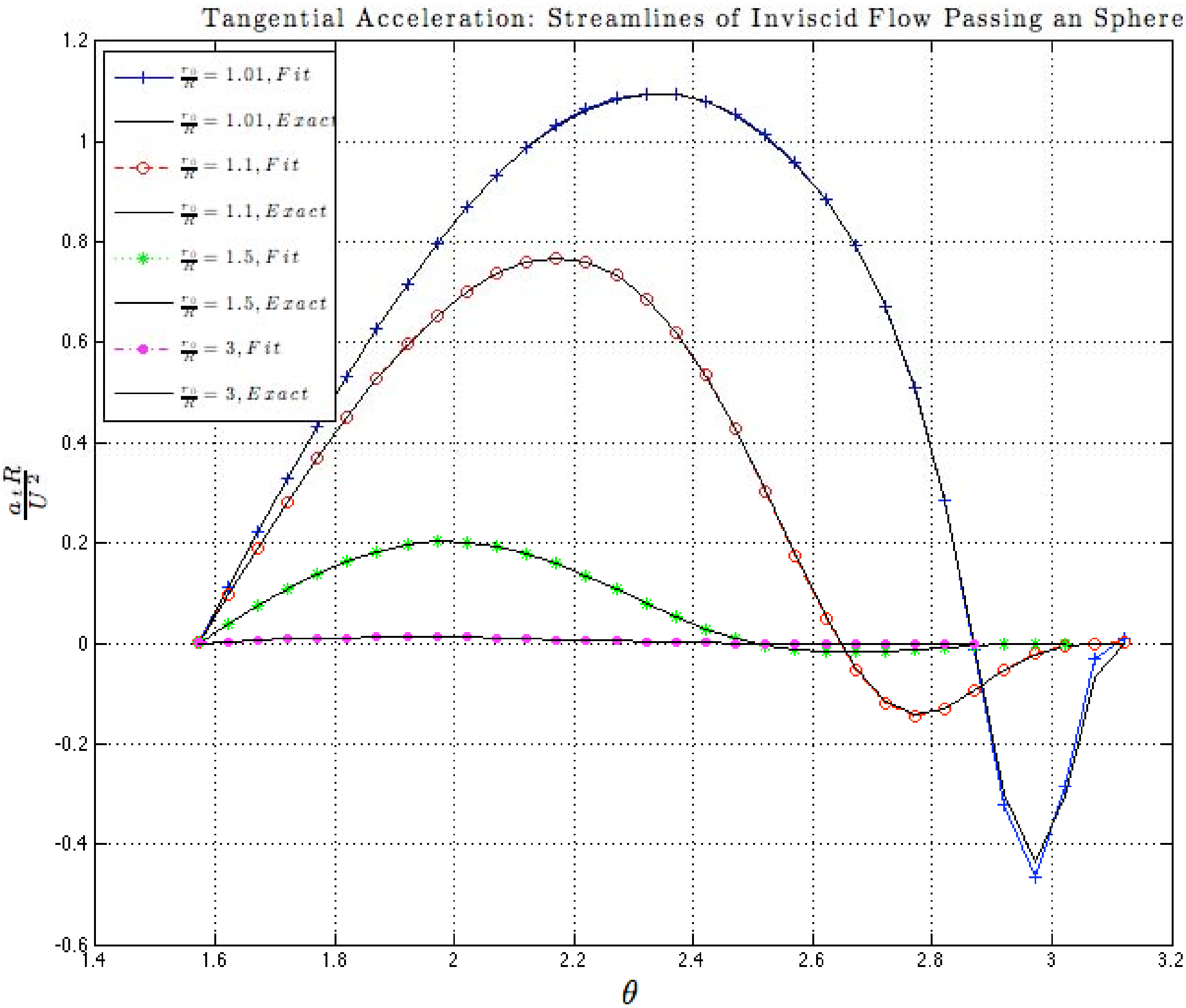
$$s = s(\theta)$$
$$\frac{ds}{dt} = \frac{ds}{d\theta}\frac{d\theta}{dt}$$
$$a_t = \frac{d^2 s}{dt^2} = \frac{d^2 s}{d\theta^2}\left(\frac{d\theta}{dt}\right)^2 + \frac{ds}{d\theta}\frac{d^2\theta}{dt^2}$$

The differentiations can be computed as below:

$$V_\theta = r\frac{d\theta}{dt} \Rightarrow \frac{d\theta}{dt} = \frac{V_\theta}{r} = -U\left(\frac{1}{r} + \frac{1}{2}\frac{R^3}{r^4}\right)\sin(\theta) = f(r,\theta)$$

$$\frac{d^2\theta}{dt^2} = \frac{\partial f}{\partial r}\frac{dr}{dt} + \frac{\partial f}{\partial \theta}\frac{d\theta}{dt} = \frac{\partial f}{\partial r}V_r + \frac{\partial f}{\partial \theta}\frac{V_\theta}{r}$$

$$\frac{\partial f}{\partial r} = U\left(\frac{1}{r^2} + 2\frac{R^3}{r^5}\right)\sin(\theta)$$
$$\frac{\partial f}{\partial \theta} = -U\left(\frac{1}{r} + \frac{1}{2}\frac{R^3}{r^4}\right)\cos(\theta)$$



Problem 3.4 (60 Points): Textbook problems

Solve the below problems from “Chapara and Canale” textbook. Note that you can use MATLAB functions whenever possible.

- 11.18, 11.20
- 13.8,13.9, 13.11
- EXTRA CREDIT: 13.19 (5 Points)
- 14.8, 14.12
- 17.12, 17.29
- 18.4
- EXTRA CREDIT: 18.9 (5 Points)
- 19.18
- 20.19

- 21.7
- 22.9 only part b, 22.13
- 23.19
- EXTRA CREDIT: 23.26 (5 Points)

Solutions:

Textbook problem 11.18

We find an upper bound for the error similar to the example 10.4 of the book. Note that we have to use the same type of norm for both "A" matrix and the "x" solution. So we expect that at least up to 2.7 significant digits are accurate (a relative error equal to 1.8×10^{-3}). However, for our choice of "b" the actual solution is accurate up to 4-5 significant digits (a relative error equal to 0.87×10^{-3}).

Furthermore note that MATLAB by default uses the double format, which has 52 bits for mantissa. This means that the coefficients of A has at most a relative error equal to the $\frac{2^{-52}}{2}$ (also recall from homework problem 1.2 that "2^-53=eps(0.5)").

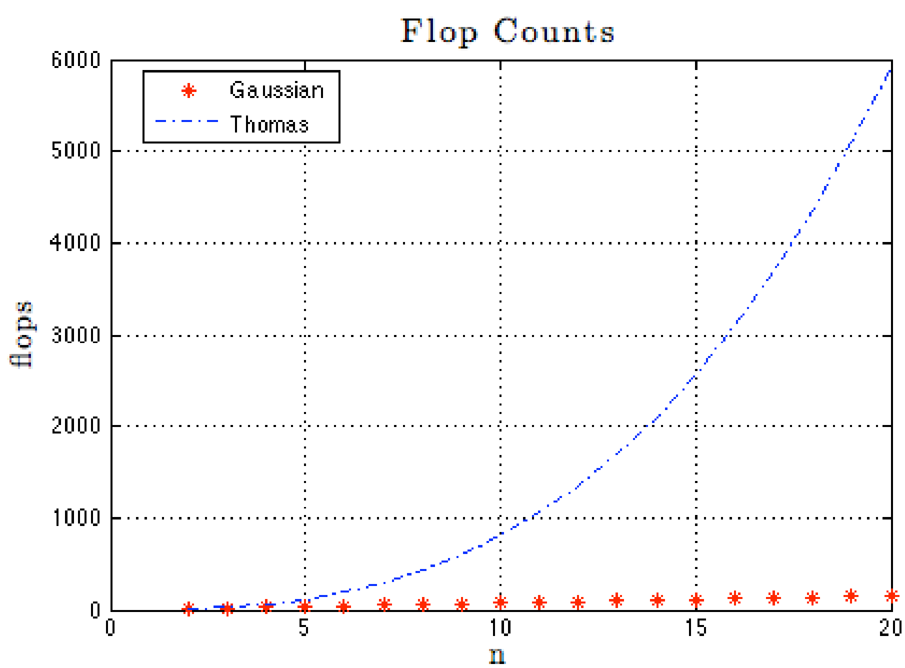
```
>> A=hilb(10)

A =

    1.0000    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000
    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909
    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833
    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769
    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714
    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667
    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625
    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588
    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588    0.0556
    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588    0.0556    0.0526

>> cnd=cond(A,2)

cnd =

   1.6025e+13

>> error_digits=log10(cnd)  % The number of missing precision

error_digits =

   13.2048

>> init_digits=log10(2^53) % Coefficient error using double with 52 bits for fraction

init_digits =

   15.9546

>> sol_digits=init_digits-error_digits % Expected accuracy

sol_digits =

    2.7498

>> sol_error=10^-sol_digits

sol_error =

    0.0018
```

```
>> b=sum(A,1); b=b'

b =

    2.9290
    2.0199
    1.6032
    1.3468
    1.1682
    1.0349
    0.9307
    0.8467
    0.7773
    0.7188

>> x=A\b

x =

    1.0000
    1.0000
    1.0000
    1.0000
    0.9999
    1.0003
    0.9995
    1.0005
    0.9997
    1.0001

>> sol_error_real=norm(x-1,2)

sol_error_real =

   8.7188e-04
```

Textbook problem 11.20

$$flops_{\text{Thomas}}(n) = \underbrace{3(n-1)}_{Decomposition} + \underbrace{2(n-1)}_{Forward} + \underbrace{1+3(n-1)}_{Backward} = 8n-7$$

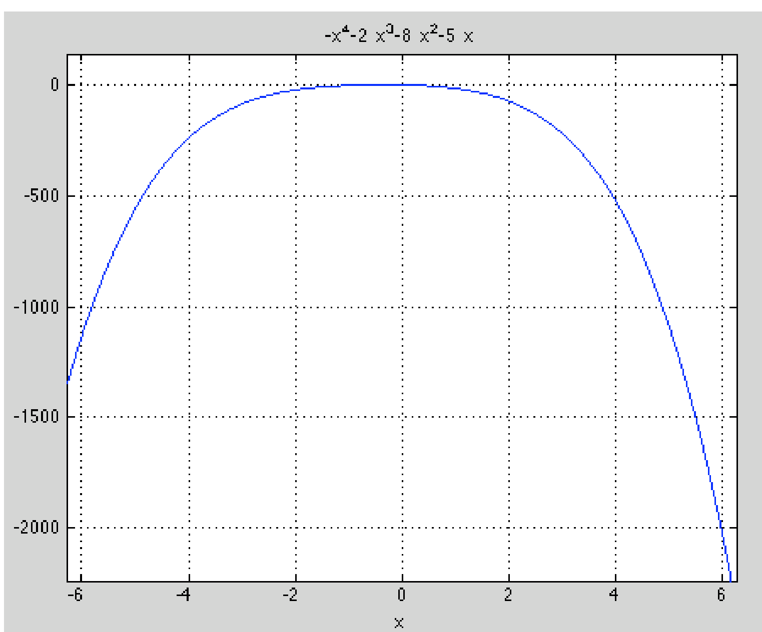
$$flops_{\text{Gauss}}(n) = \frac{2n^3}{3} + \frac{3n^2}{2} - \frac{7n}{6}$$



Textbook problem 13.8

Note that as $x \to \pm\infty \Rightarrow f \to -\infty$. So since the function is continuous it will have at least one global maximum.

```
>> syms x
>> f=-x^4-2*x^3-8*x^2-5*x;
>> ezplot(f),box on,grid on
>> df=diff(f)

df =

-4*x^3-6*x^2-16*x-5


>> d2f=diff(df)

d2f =

-12*x^2-12*x-16


>> roots_df=double(solve(df))

roots_df =

  -0.3473
  -0.5764 + 1.8076i
  -0.5764 - 1.8076i

>> x=roots_df(1); % The only real root and the only candidate as an extreme point
>> df_extreme=eval(d2f) % If it is smaller than zero, then it is a maximum

df_extreme =

  -13.2800

>>
```



Textbook problem 13.9

Part a,b are based on attached file “C2p29_PSET3_13p9.m”. Part c is based on the previous file “solver.m” from PSET#2 which is attached again. Clearly quadratic interpolation and Newton’s method have faster convergence rate.

Furthermore, note that during quadratic interpolation, while we have $x_0 \le x_1 \le x_2$, the x_3 will be either $x_3 \in [x_0, x_1]$ or $x_3 \in [x_1, x_2]$. So this condition should be checked prior to the selection of points for the next step.

Running C2p29_PSET3_13p9.m:

Golden-Section Search for Maximum:

i	x_l	x_2	x_1	x_u	f_l	f_2	f_1	f_u	max(x_rel_error)
1	-2.00000	-0.85410	-0.14590	+1.00000	-2.20000e+01	-8.51449e-01	+5.64958e-01	-1.60000e+01	+7.85410e+02%
2	-0.85410	-0.14590	+0.29180	+1.00000	-8.51449e-01	+5.64958e-01	-2.19708e+00	-1.60000e+01	+4.85410e+02%
3	-0.85410	-0.41641	-0.14590	+0.29180	-8.51449e-01	+8.09216e-01	+5.64958e-01	-2.19708e+00	+1.05112e+02%
4	-0.85410	-0.58359	-0.41641	-0.14590	-8.51449e-01	+4.74847e-01	+8.09216e-01	+5.64958e-01	+6.49627e+01%
5	-0.58359	-0.41641	-0.31308	-0.14590	+4.74847e-01	+8.09216e-01	+8.33016e-01	+5.64958e-01	+5.33995e+01%
6	-0.41641	-0.31308	-0.24922	-0.14590	+8.09216e-01	+8.33016e-01	+7.76321e-01	+5.64958e-01	+3.30027e+01%
7	-0.41641	-0.35255	-0.31308	-0.24922	+8.09216e-01	+8.40608e-01	+8.33016e-01	+7.76321e-01	+1.81134e+01%
8	-0.41641	-0.37694	-0.35255	-0.31308	+8.09216e-01	+8.34956e-01	+8.40608e-01	+8.33016e-01	+1.11947e+01%
9	-0.37694	-0.35255	-0.33747	-0.31308	+8.34956e-01	+8.40608e-01	+8.40159e-01	+8.33016e-01	+6.91871e+00%
10	-0.37694	-0.36187	-0.35255	-0.33747	+8.34956e-01	+8.39377e-01	+8.40608e-01	+8.40159e-01	+4.27600e+00%
11	-0.36187	-0.35255	-0.34679	-0.33747	+8.39377e-01	+8.40608e-01	+8.40793e-01	+8.40159e-01	+2.68659e+00%
12	-0.35255	-0.34679	-0.34323	-0.33747	+8.40608e-01	+8.40793e-01	+8.40687e-01	+8.40159e-01	+1.66041e+00%
13	-0.35255	-0.34899	-0.34679	-0.34323	+8.40608e-01	+8.40774e-01	+8.40793e-01	+8.40687e-01	+1.02619e+00%
14	-0.34899	-0.34679	-0.34543	-0.34323	+8.40774e-01	+8.40793e-01	+8.40772e-01	+8.40687e-01	+3.91969e-01%

Final Solution: f(x= -0.3467910200656146)=+8.407925e-01 with 14 iterations

Quadratic-Interpolation Search for Maximum:

i	x_0	x_1	x_2	x_3	f_0	f_1	f_2	f_3
1	-2.00000	-1.00000	+1.00000	-0.38889	-2.20000e+01	-2.00000e+00	-1.60000e+01	+8.29323e-01
2	-1.00000	-0.38889	+1.00000	-0.41799	-2.00000e+00	+8.29323e-01	-1.60000e+01	+8.07760e-01
3	-0.41799	-0.38889	+1.00000	-0.36258	+8.07760e-01	+8.29323e-01	-1.60000e+01	+8.39236e-01
4	-0.38889	-0.36258	+1.00000	-0.35519	+8.29323e-01	+8.39236e-01	-1.60000e+01	+8.40376e-01

Final Solution: f(x= -0.3551882152295684)=+8.403759e-01 with 5 iterations

Calling solver.m:

```
>> syms x
>> f=-x^4-2*x^3-8*x^2-5*x;
>> df=diff(f)

df =

-4*x^3-6*x^2-16*x-5


>> d2f=diff(df)

d2f =

-12*x^2-12*x-16


>> df=@(x) -4*x^3-6*x^2-16*x-5;
>> d2f=@(x) -12*x^2-12*x-16;
>> solver(df,-1,'method','Newton','f_derivative',d2f)
```

k	x_o	x_n	f_o	f_n	df_o	x_rel_error
0	-1.00000000	-0.43750000	+9.00000e+00	+1.18652e+00	-1.60000e+01	+1.28571e+02%
1	-0.43750000	-0.34655689	+1.18652e+00	-9.21162e-03	-1.30469e+01	+2.62419e+01%
2	-0.34655689	-0.34725040	-9.21162e-03	-8.84268e-07	-1.32825e+01	+1.99716e-01%
3	-0.34725040	-0.34725047	-8.84268e-07	-7.99361e-15	-1.32800e+01	+1.91754e-05%
4	-0.34725047	-0.34725047	-7.99361e-15	+0.00000e+00	-1.32800e+01	+1.75845e-13%

```
        x= -0.3472504665430884 is the exact solution.

 Final Solution: f(x= -0.3472504665430884)=+0.000000e+00 with NEWTON method and 4 iterations

ans =
```

Textbook problem 13.11

The previous "solver.m" from PSET#2 is used again:

```
>> syms x
>> f=3+6*x+5*x^2+3*x^3+4*x^4;
>> df=diff(f)

df =

6+10*x+9*x^2+16*x^3


>> d2f=diff(df)

d2f =

10+18*x+48*x^2


>> x=solver('6+10*x+9*x^2+16*x^3',[-2 1],'method','Bi-section','rel_tolerance',1e-3)

 k       x_l            x_u            x_n            f_l            f_u            f_n            x_rel_error
------------------------------------------------------------------------------------------------------------------
 0       -2.00000000    +1.00000000    -0.50000000    -1.06000e+02   +4.10000e+01   +1.25000e+00   +3.00000e+02%
 1       -2.00000000    -0.50000000    -1.25000000    -1.06000e+02   +1.25000e+00   -2.36875e+01   +6.00000e+01%
 2       -1.25000000    -0.50000000    -0.87500000    -2.36875e+01   +1.25000e+00   -6.57812e+00   +4.28571e+01%
 3       -0.87500000    -0.50000000    -0.68750000    -6.57812e+00   +1.25000e+00   -1.82031e+00   +2.72727e+01%
 4       -0.68750000    -0.50000000    -0.59375000    -1.82031e+00   +1.25000e+00   -1.13770e-01   +1.57895e+01%
 5       -0.59375000    -0.50000000    -0.54687500    -1.13770e-01   +1.25000e+00   +6.06018e-01   +8.57143e+00%
 6       -0.59375000    -0.54687500    -0.57031250    -1.13770e-01   +6.06018e-01   +2.56218e-01   +4.10959e+00%
 7       -0.59375000    -0.57031250    -0.58203125    -1.13770e-01   +2.56218e-01   +7.38249e-02   +2.01342e+00%
 8       -0.59375000    -0.58203125    -0.58789062    -1.13770e-01   +7.38249e-02   -1.93125e-02   +9.96678e-01%
 9       -0.58789062    -0.58203125    -0.58496094    -1.93125e-02   +7.38249e-02   +2.74199e-02   +5.00835e-01%
 10      -0.58789062    -0.58496094    -0.58642578    -1.93125e-02   +2.74199e-02   +4.09481e-03   +2.49792e-01%
 11      -0.58789062    -0.58642578    -0.58715820    -1.93125e-02   +4.09481e-03   -7.59856e-03   +1.24740e-01%
 12      -0.58715820    -0.58642578    -0.58679199    -7.59856e-03   +4.09481e-03   -1.74930e-03   +6.24090e-02%

 Final Solution: f(x= -0.5867919921875000)=-1.749304e-03 with BI-SECTION method and 12 iterations

x =

   -0.5868
 >> d2f_xroot=eval(d2f) % Should be larger than zero to be a minimum

 d2f_xroot =

    15.9653

 >>
```

(EXTRA CREDIT) Textbook problem 13.19

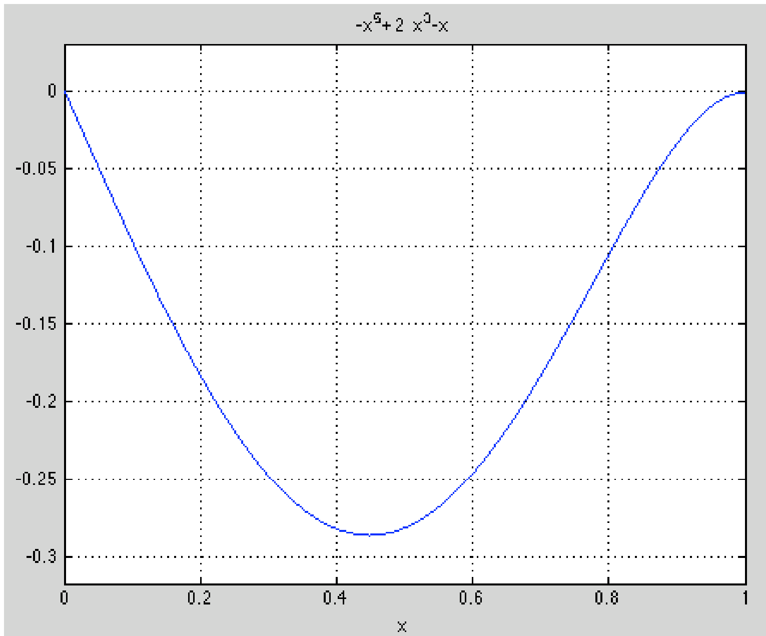
The problem is equivalent to the simpler problem of $\max\limits_{0\le s\le 1}(-s^5+2s^3-s) \overset{find\ s^*}{\Rightarrow} x^* = s^*L$ (because $y(x)=K\left(-\left(\frac{x}{L}\right)^5+2\left(\frac{x}{L}\right)^3-\left(\frac{x}{L}\right)\right),\quad 0\le x\le L$)[3]. Here the previous file

[3] Here we can find the analytical solution:

$$\max_{0\le s\le 1}(f(s)=-s^5+2s^3-s) \overset{find\ s^*}{\Rightarrow} f'(s^*)=0$$
$$-5s^{*4}+6s^{*2}-1=(-5s^{*2}+1)(s^{*2}-1)\Rightarrow s^*=\frac{1}{\sqrt{5}}\cong 0.4472$$

"C2p29_PSET3_13p9.m" is slightly modified and used again. Note that we are looking for minimum of "y" and it is equivalent to the maximum of "-y".

The maximum absolute value of deflection happens at $s^* = 0.448$ and hence at $x^* = s^* L = 0.448 \times 600cm = 268.8cm$.



Golden-Section Search for Maximum:

i	x_l	x_2	x_1	x_u	f_l	f_2	f_1	f_u	max(x_rel_error)
1	+0.00000	+0.38197	+0.61803	+1.00000	+0.00000e+00	+2.78640e-01	+2.36068e-01	+0.00000e+00	+1.00000e+02%
2	+0.00000	+0.23607	+0.38197	+0.61803	+0.00000e+00	+2.10490e-01	+2.78640e-01	+2.36068e-01	+6.18034e+01%
3	+0.23607	+0.38197	+0.47214	+0.61803	+2.10490e-01	+2.78640e-01	+2.85106e-01	+2.36068e-01	+3.09017e+01%
4	+0.38197	+0.47214	+0.52786	+0.61803	+2.78640e-01	+2.85106e-01	+2.74679e-01	+2.36068e-01	+1.90983e+01%
5	+0.38197	+0.43769	+0.47214	+0.52786	+2.78640e-01	+2.86055e-01	+2.85106e-01	+2.74679e-01	+1.27322e+01%
6	+0.38197	+0.41641	+0.43769	+0.47214	+2.78640e-01	+2.84521e-01	+2.86055e-01	+2.85106e-01	+7.86893e+00%
7	+0.41641	+0.43769	+0.45085	+0.47214	+2.84521e-01	+2.86055e-01	+2.86193e-01	+2.85106e-01	+4.72136e+00%
8	+0.43769	+0.45085	+0.45898	+0.47214	+2.86055e-01	+2.86193e-01	+2.85969e-01	+2.85106e-01	+2.91796e+00%
9	+0.43769	+0.44582	+0.45085	+0.45898	+2.86055e-01	+2.86213e-01	+2.86193e-01	+2.85969e-01	+1.82373e+00%
10	+0.43769	+0.44272	+0.44582	+0.45085	+2.86055e-01	+2.86181e-01	+2.86213e-01	+2.86193e-01	+1.12712e+00%
11	+0.44272	+0.44582	+0.44774	+0.45085	+2.86181e-01	+2.86213e-01	+2.86216e-01	+2.86193e-01	+4.28678e-01%

Final Solution: f(x= +0.4477440987322294)=+2.862162e-01 with 11 iterations

Textbook problem 14.8

```
>> syms x y h
>> f=-8*x+x^2+12*y+4*y^2-2*x*y;
>> Df=[diff(f,x), diff(f,y)]

Df =

[ -8+2*x-2*y, 12+8*y-2*x]


>> x=0;y=0; Df=double(eval(Df))

Df =

    -8    12

>> f_h=simple(subs(f,{'x','y'},{'0-8*h','0+12*h'}))

f_h =

208*h+832*h^2
```

We have to find the roots of $\frac{df(h)}{dh}$, and compare the value of "f" at h=0 (current point) and new point h_{opt}, $s.t.$ $\frac{df(h_{opt})}{dh}=0$. The new point x=1,y=-1.5 corresponding to $h_{opt}=-\frac{1}{8}$ has a smaller value of "f" and the optimal gradient steepest descent method has been successful (changing "f" from 0 to -13). The next step comes to x=2.5, y=-0.5 and decreases f to the "-16.25".

```
>> Df_h=diff(f_h)
Df_h =
208+1664*h

>> h_opt=solve(Df_h)
h_opt =
-1/8

>> h=h_opt; x=0-8*h,y=0+12*h
x =
1


y =
-3/2
>> double(eval(f_h))
ans =
   -13
>> h=0;double(eval(f_h))
ans =
     0
>> Df=[diff(f,'x'), diff(f,'y')]
Df =
[ -8+2*x-2*y, 12+8*y-2*x]

>> x=1;y=-1.5; Df=double(eval(Df))  % 2nd Iteration
Df =
    -3    -2
>> f_h=simple(subs(f,{'x','y'},{'1-3*h','-1.5-2*h'}))
f_h =
-13.+13.*h+13.*h^2

>> Df_h=diff(f_h,'h')
Df_h =
13.+26.*h
>> h_opt=solve(Df_h); h=h_opt, x=1-3*h,y=-1.5-2*h
h =
-.50000000000000000000000000000000


x =
2.5000000000000000000000000000000


y =
-.50000000000000000000000000000000
```

```
>> double(eval(f_h))

ans =

  -16.2500

>> h=0;double(eval(f_h))

ans =

   -13
```

Textbook problem 14.12

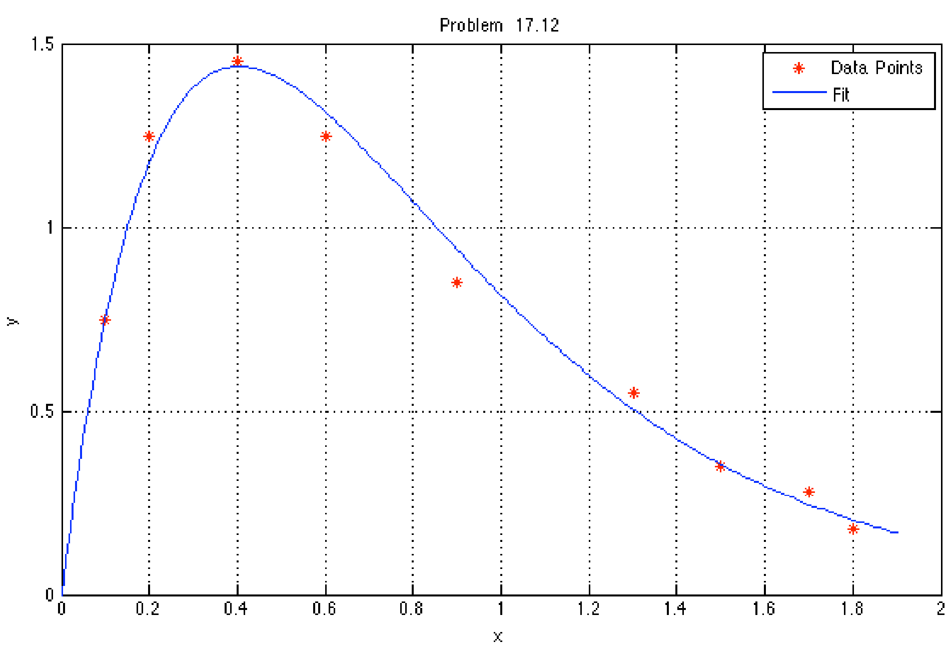
```
>> syms x y h
>> f=2*x^3*y^2-7*x*y+x^2+3*y;
>> Df=[diff(f,x), diff(f,y)]

Df =

[ 6*x^2*y^2-7*y+2*x,     4*x^3*y-7*x+3]

>> x=1;y=1;eval(Df)

ans =

     1     0

>> f_h=simple(subs(f,{'x','y'},{'1+1*h','1+0*h'}))

f_h =

-1+h+7*h^2+2*h^3
```

Textbook problem 17.12

The equation is linearized by below formula and then the MATLAB function "polyfit" is used to find the coefficients $\ln(\alpha_4), \beta_4$:

$$y = \alpha_4 x e^{\beta_4 x}$$
$$\frac{y}{x} = \alpha_4 e^{\beta_4 x} \Rightarrow \ln(\frac{y}{x}) = \ln(\alpha_4) + \beta_4 x$$

```
>> x=[ 0.1 0.2 0.4 0.6 0.9 1.3 1.5 1.7 1.8];
>> y=[0.75 1.25 1.45 1.25 0.85 0.55 0.35 0.28 0.18];
>> Y=log(y./x)

Y =

    2.0149    1.8326    1.2879    0.7340   -0.0572   -0.8602   -1.4553   -1.8036   -2.3026

>> p=polyfit(x,Y,1)

p =

   -2.4733    2.2682

>> beta_4=p(1)

beta_4 =

   -2.4733

>> alpha_4=exp(p(2))

alpha_4 =

    9.6618

>> x_fit=[0:.01:1.9];
>> y_fit=alpha_4.*x_fit.*exp(beta_4*x_fit);
>> plot(x,y,'r*',x_fit,y_fit,'b')
>> grid on, box on, xlabel('x'),ylabel('y')
>> legend('Data Points','Fit')
>> set(gcf,'Position',[50 50 [25 20]*30],'color','w')
>> title('Problem  17.12')
```



Textbook problem 17.29

There are a couple of techniques:

- Using MATLAB curve fitting tool by an optional fit (next page):
  This provides a confidence level as well. The default search technique is "trust region" and we found out that $\beta_4 = -2.532,\ \alpha_4 = 9.897$. This is comparable to the previous problem where we used a linearized model and found out that $\beta_4 = -2.473,\ \alpha_4 = 9.662$.

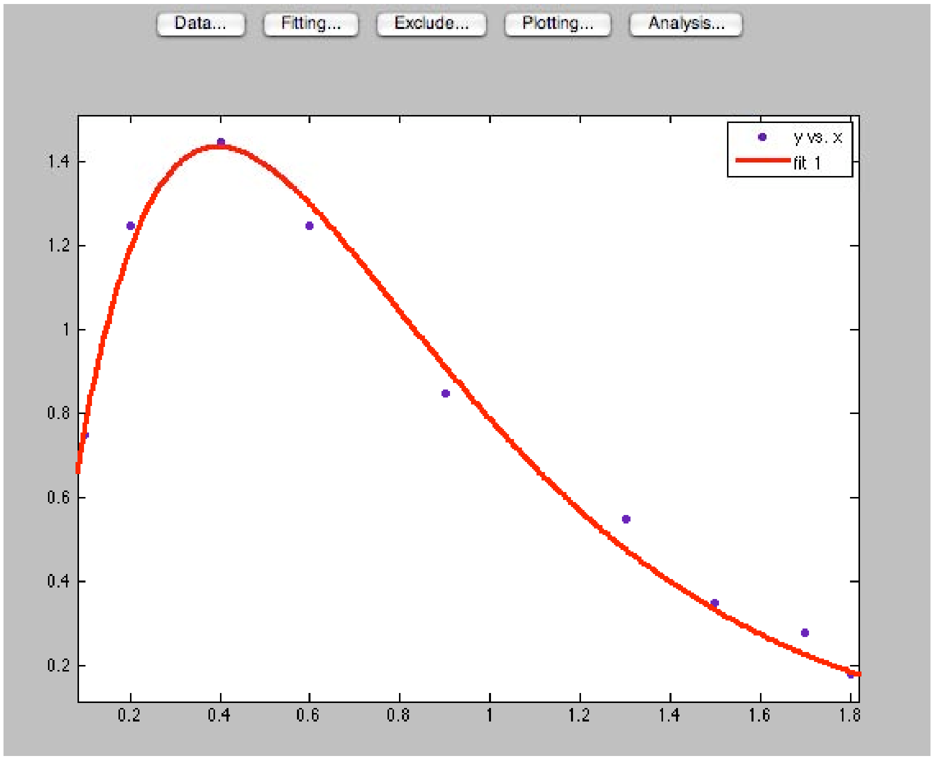
- Use nonlinear regression method described in the section 17.5 (implemented in "C2p29_PSET3_17p29.m"):

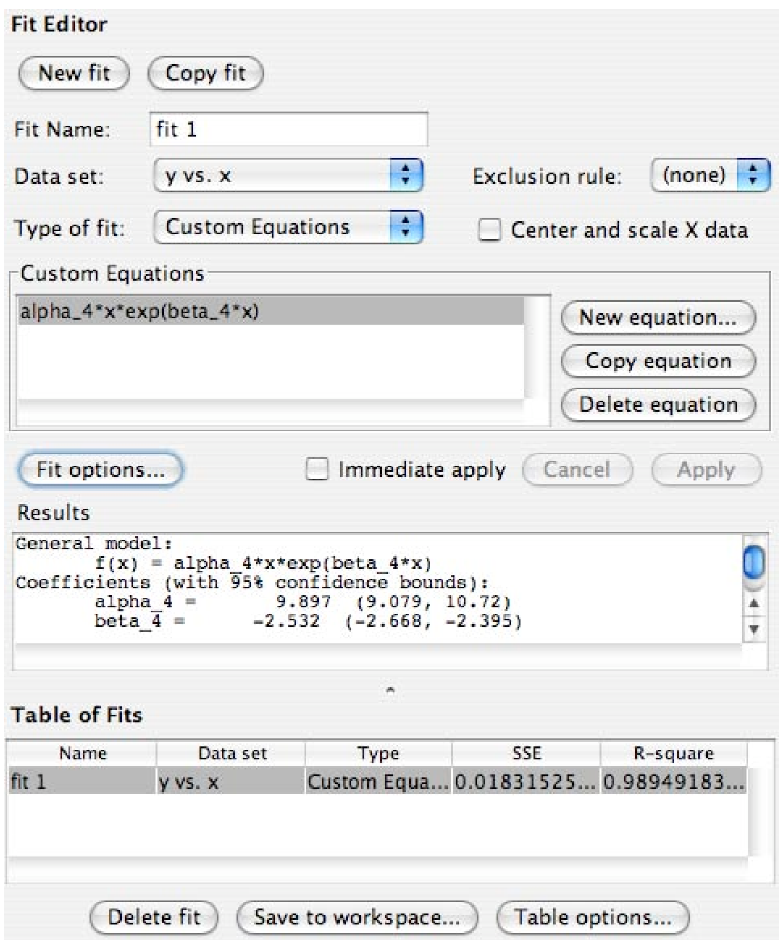
$$y = \alpha_4 x e^{\beta_4 x}$$
$$\frac{\partial y}{\partial \alpha_4} = x e^{\beta_4 x}$$
$$\frac{\partial y}{\partial \beta_4} = \alpha_4 x^2 e^{\beta_4 x}$$

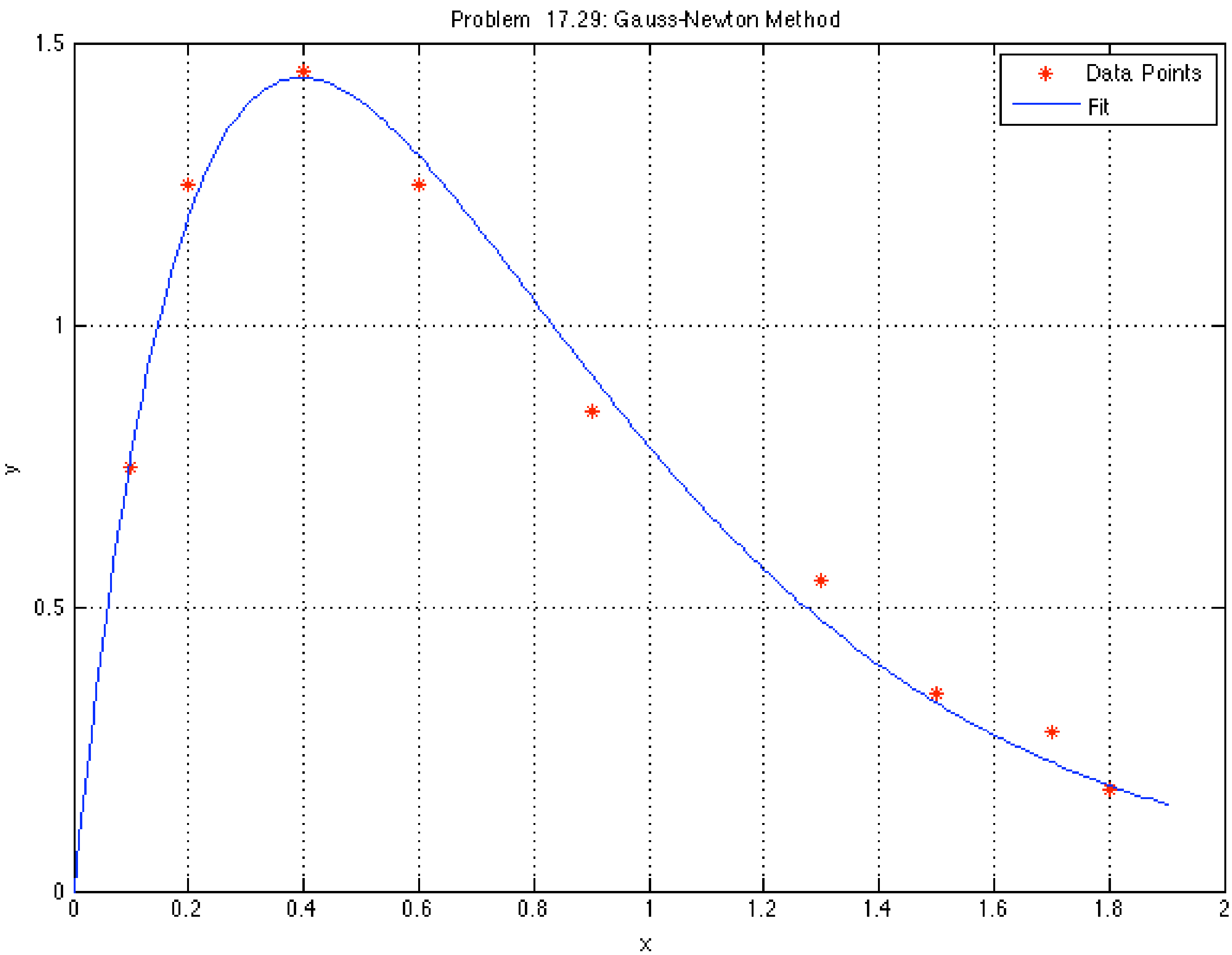
The initial guess is set to $\beta_4 = 0,\ \alpha_4 = 1$. By 8 iterations we found out that $\beta_4 = -2.532,\ \alpha_4 = 9.897$. The result is equal to the previous fit and it is not sensitive to the initial guess. Here is the program output and the plot is shown on page 25:

```
        Gauss-Newton Method Fit

 i      alpha_4              beta_4
-------------------------------------------------------
 1      +1.00000        +0.00000
 2      +2.89844        -1.65718
 3      +8.44690        -2.99646
 4      +9.66138        -2.29792
 5      +9.74769        -2.48887
 6      +9.88407        -2.52918
 7      +9.89672        -2.53174
 8      +9.89733        -2.53186
 9      +9.89736        -2.53187
 10     +9.89736        -2.53187
>>
```

- Use residual techniques with nonlinear fit (similar to MATLAB fit).


Data...
Fitting...
Exclude...
Plotting...
Analysis...
y vs. x
fit 1
1.4
1.2
1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8


Fit Editor
New fit
Copy fit
Fit Name: fit 1
Data set: y vs. x
Exclusion rule: (none)
Type of fit: Custom Equations
Center and scale X data
Custom Equations
alpha_4*x*exp(beta_4*x)
New equation...
Copy equation
Delete equation
Fit options...
Immediate apply
Cancel
Apply
Results
General model:
f(x) = alpha_4*x*exp(beta_4*x)
Coefficients (with 95% confidence bounds):
alpha_4 = 9.897 (9.079, 10.72)
beta_4 = -2.532 (-2.668, -2.395)
Table of Fits
Name
Data set
Type
SSE
R-square
fit 1
y vs. x
Custom Equa...
0.01831525...
0.98949183...
Delete fit
Save to workspace...
Table options...


Problem 17.29: Gauss-Newton Method
Data Points
Fit
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
0.5
1
1.5

Textbook problem 18.4

For the best estimate, we have to reorder the numbers so that they are as close to x=2.8 as possible. The error in each order is approximated by the term from the next order. The below graph shows the schematic evaluation, while indeed MATALB functions "diff, prod, sum" can be used to conveniently compute different orders, as shown on the next page.

x	$f(x)$					
2.5	14					
		0.7	1.429			
3.2	15			−8.809		
		−1.2	5.833		1.012	
2	8			−7.292		1.848
		2	0		−0.651	
4	8			−6.25		
		−2.4	2.5			
1.6	2					

$$p_1(x) = 14 + (1.429)\times(x-2.5)$$
$$p_2(x) = 14 + (1.429)\times(x-2.5) - 8.809\times(x-2.5)(x-3.2)$$
$$p_3(x) = 14 + (1.429)\times(x-2.5) - 8.809\times(x-2.5)(x-3.2) + -1.012\times(x-2.5)(x-3.2)(x-4)$$

$$e_1(x) \cong -8.809\times(x-2.5)(x-3.2)$$
$$e_2(x) \cong -1.012\times(x-2.5)(x-3.2)(x-4)$$
$$e_3(x) \cong +1.848\times(x-2.5)(x-3.2)(x-4)(x-1.6)$$

```
>> x=[2.5 3.2 2 4 1.6];
>> y=[14 15 8 8 2];
>> b0=y(1)

b0 =

    14

>> f10=diff(y)./diff(x)

f10 =

    1.4286    5.8333         0    2.5000

>> b1=f10(1);
>> f210=diff(f10)./(x(3:end)-x(1:end-2))

f210 =

   -8.8095   -7.2917   -6.2500

>> b2=f210(1)

b2 =

   -8.8095

>> f3210=diff(f210)./(x(4:end)-x(1:end-3))

f3210 =

    1.0119   -0.6510

>> b3=f3210(1)

b3 =

    1.0119

>> f43210=diff(f3210)./(x(5:end)-x(1:end-4))

f43210 =

    1.8477
>> b4=f43210(1)

b4 =

    1.8477

>> x_diff=2.8-x;
>> Terms=[b0 b1 b2 b3 b4].*[1 prod(x_diff(1)) prod(x_diff(1:2)) prod(x_diff(1:3)) prod(x_diff(1:4))]

Terms =

   14.0000    0.4286    1.0571   -0.0971    0.2129

>> Poly_orders=[ sum(Terms(1:2)) sum(Terms(1:3)) sum(Terms(1:4))] % Newton Polynomial Interpolation from Order 1 to 3

Poly_orders =

   14.4286   15.4857   15.3886

>> Poly_errors=Terms(3:end)

Poly_errors =

    1.0571   -0.0971    0.2129
```

(EXTRA CREDIT) Textbook problem 18.9

Note that there has been a typo in the book. Indeed the actual function is $f(x) = \dfrac{x^2}{1+x^2}$.

```
>> x=[0:5];
>> fx=[0 0.5 0.8 0.9 0.941176 0.961538];
>> % Part a
>> x_correct=solve('x^2/(1+x^2)-0.85')

x_correct =

  2.3804761428476166659997999371224
 -2.3804761428476166659997999371224


>> x_correct=x_correct(1); % Only the positive solution is acceptable
>> % Part b
>> p3=polyfit(fx(2:5),x(2:5),3) % Use x(2) to x(5) for 3rd order fit

p3 =

  191.5927 -404.8373  282.4672  -62.9734

>> x_b=polyval(p3,0.85)

x_b =

    2.2907

>> x_b_rel_error_percent=abs((x_b-x_correct)/x_correct)*100

x_b_rel_error_percent =

3.7717851422500253499036263571628


>> % Part c
>> p2_inv=polyfit(x(3:5),fx(3:5),2) % Use x(3) to x(5) for 2nd order inverse fit

p2_inv =

   -0.0294    0.2471    0.4235

>> g2=@(x) p2_inv(1)*x^2+p2_inv(2)*x+p2_inv(3)-0.85 % Quadratic Interpolation Function

g2 =

    @(x)p2_inv(1)*x^2+p2_inv(2)*x+p2_inv(3)-0.85

>> x_c=fzero(g2,2)

x_c =

    2.4280


>> x_c_rel_error_percent=abs((x_c-x_correct)/x_correct)*100

x_c_rel_error_percent =

1.9961884515965268563767646105122


>> % Part d
>> p3_inv=polyfit(x(2:5),fx(2:5),3) % Use x(2) to x(5) for 3rd order inverse fit

p3_inv =

    0.0235   -0.2412    0.8588   -0.1412

>> g3=@(x) p3_inv(1)*x^3+p3_inv(2)*x^2+p3_inv(3)*x+p3_inv(4)-0.85 % Cubic Interpolation Function

g3 =

    @(x)p3_inv(1)*x^3+p3_inv(2)*x^2+p3_inv(3)*x+p3_inv(4)-0.85
```

```
>> x_d=solver(g3,[2 3],'method','Bi-section')

 k       x_l            x_u            x_n            f_l            f_u            f_n            x_rel_error
----------------------------------------------------------------------------------------------------------------------
 0       +2.00000000    +3.00000000    +2.50000000    -5.00000e-02   +5.00000e-02   +1.61765e-02   +2.00000e+01%
 1       +2.00000000    +2.50000000    +2.25000000    -5.00000e-02   +1.61765e-02   -1.17647e-02   +1.11111e+01%
 2       +2.25000000    +2.50000000    +2.37500000    -1.17647e-02   +1.61765e-02   +3.35480e-03   +5.26316e+00%
 3       +2.25000000    +2.37500000    +2.31250000    -1.17647e-02   +3.35480e-03   -3.90048e-03   +2.70270e+00%
 4       +2.31250000    +2.37500000    +2.34375000    -3.90048e-03   +3.35480e-03   -1.98879e-04   +1.33333e+00%
 5       +2.34375000    +2.37500000    +2.35937500    -1.98879e-04   +3.35480e-03   +1.59618e-03   +6.62252e-01%
 6       +2.34375000    +2.35937500    +2.35156250    -1.98879e-04   +1.59618e-03   +7.03241e-04   +3.32226e-01%
 7       +2.34375000    +2.35156250    +2.34765625    -1.98879e-04   +7.03241e-04   +2.53332e-04   +1.66389e-01%
 8       +2.34375000    +2.34765625    +2.34570312    -1.98879e-04   +2.53332e-04   +2.75151e-05   +8.32639e-02%
 9       +2.34375000    +2.34570312    +2.34472656    -1.98879e-04   +2.75151e-05   -8.56098e-05   +4.16493e-02%
 10      +2.34472656    +2.34570312    +2.34521484    -8.56098e-05   +2.75151e-05   -2.90293e-05   +2.08203e-02%
 11      +2.34521484    +2.34570312    +2.34545898    -2.90293e-05   +2.75151e-05   -7.52591e-07   +1.04091e-02%
 12      +2.34545898    +2.34570312    +2.34558105    -7.52591e-07   +2.75151e-05   +1.33824e-05   +5.20427e-03%
 13      +2.34545898    +2.34558105    +2.34552002    -7.52591e-07   +1.33824e-05   +6.31518e-06   +2.60220e-03%
 14      +2.34545898    +2.34552002    +2.34548950    -7.52591e-07   +6.31518e-06   +2.78137e-06   +1.30112e-03%
 15      +2.34545898    +2.34548950    +2.34547424    -7.52591e-07   +2.78137e-06   +1.01440e-06   +6.50563e-04%
 16      +2.34545898    +2.34547424    +2.34546661    -7.52591e-07   +1.01440e-06   +1.30911e-07   +3.25283e-04%
 17      +2.34545898    +2.34546661    +2.34546280    -7.52591e-07   +1.30911e-07   -3.10839e-07   +1.62642e-04%
 18      +2.34546280    +2.34546661    +2.34546471    -3.10839e-07   +1.30911e-07   -8.99634e-08   +8.13207e-05%
 19      +2.34546471    +2.34546661    +2.34546566    -8.99634e-08   +1.30911e-07   +2.04740e-08   +4.06603e-05%
 20      +2.34546471    +2.34546566    +2.34546518    -8.99634e-08   +2.04740e-08   -3.47447e-08   +2.03302e-05%

 Final Solution: f(x= +2.3454651832580566)=-3.474471e-08 with BI-SECTION method and 20 iterations

x_d =

    2.3455

>> x_d_rel_error_percent=abs((x_d-x_correct)/x_correct)*100

x_d_rel_error_percent =

1.4707544830791110386302785482009
```

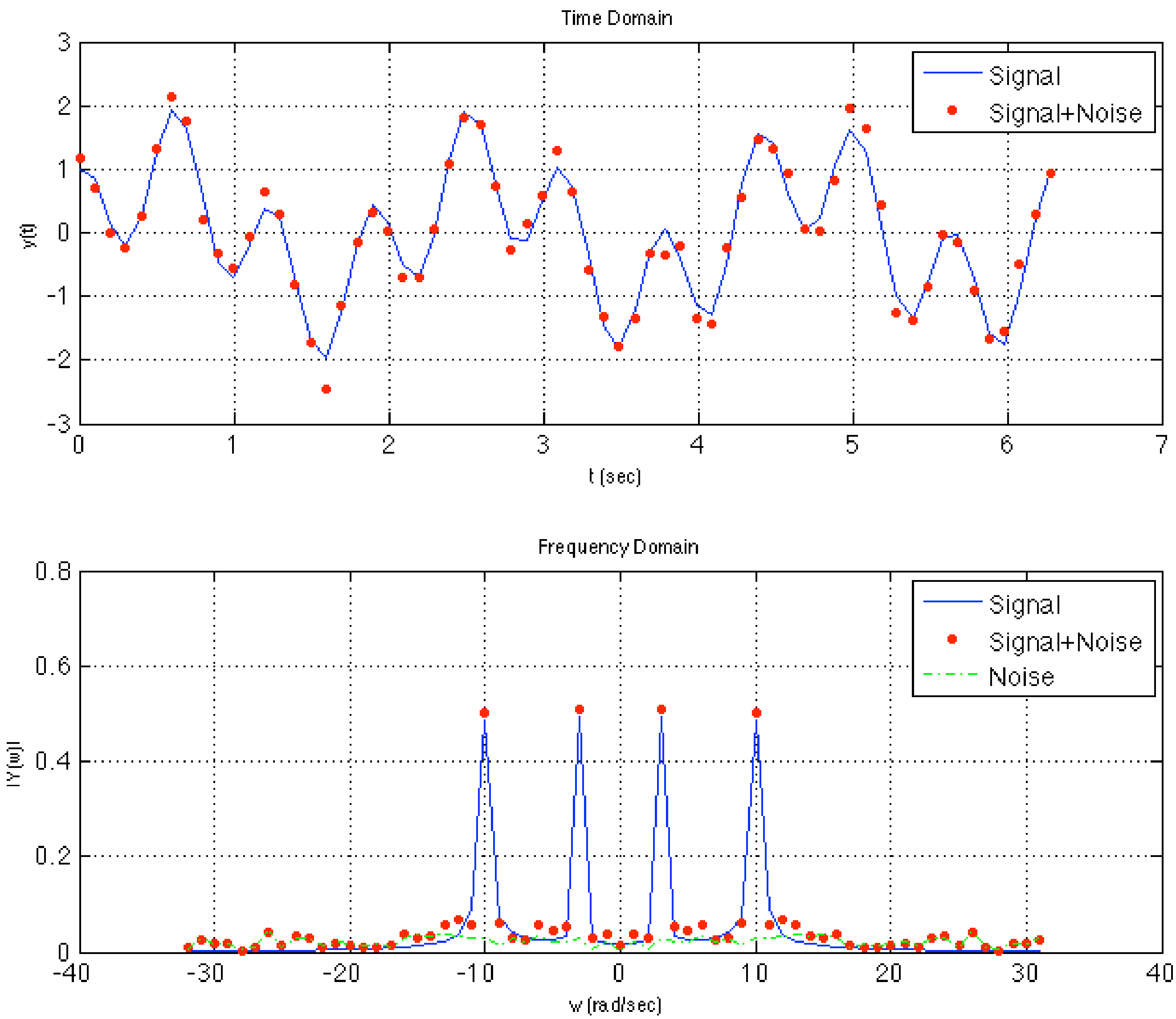
Textbook problem 19.18

Note that the sampling frequency is equal to $f_s = \frac{64}{2\pi}\, Hz$. So $\omega_s = 64\ \frac{rad}{s}$ and consequently the angular frequencies that correspond to FFT will be limited by $\omega_{\max} = \frac{\omega_s}{2} = 32\ \frac{rad}{s}$ (using Nyquist theorem). On the other hand since we are using 64 points for FFT, then the FFT will produce N=64 components of ω, which will be $\omega = -32, -31, \ldots, -1, 0, 1, \ldots, 30, 31\ \frac{rad}{s}$. Note that since our signal is real, the FFT values at negative frequency is equal to positive ones, but with negative phase. Here FFT magnitude (or signal power) is plotted which is equal at both negative and positive frequencies. The noise has had an amplitude equal to "0.2".

Furthermore note that you need to perform two things in MATLAB:

1) Use "fftshift" function to reorder the FFT values. (Otherwise the 0 frequency is not at the center).
2) Divide the FFT by the number of points to get the correct power (or FFT magnitude) and also in other circumstances by other related coefficients. This is done here and we can see that the original signal has an amplitude equal to 0.5 at $\omega = -10, 10, -3, 3\ \frac{rad}{s}$ (because $\cos(wt) = \frac{1}{2}e^{jwt} + \frac{1}{2}e^{-jwt}$).

```
>> n=64;
>> t=linspace(0,2*pi,n);
>> f=cos(10*t)+sin(3*t);
>> f_noisy=f+.2*randn(1,64);
>> subplot(2,1,1)
>> plot(t,f,'b',t,f_noisy,'r.')
>> xlabel('t (sec)'), ylabel('y(t)'), title('Time Domain'), legend('Signal','Signal+Noise'), box on, grid on
>> subplot(2,1,2)
>> F=fftshift(fft(f))/n;
>> F_noisy=fftshift(fft(f_noisy))/n;
>> w_fft=[-32:31];
>> plot(w_fft,abs(F),'b',w_fft,abs(F_noisy),'r.')
>> xlabel('w (rad/sec)'), ylabel('|Y(w)|'), title('Frequency Domain'), legend('Signal','Signal+Noise'), box on, grid on
>> hold on
>> plot(w_fft,abs(F_noisy-F),'g-.')
>>  legend('Signal','Signal+Noise','Noise')
>> set(gcf,'Position',[50 50 [25 20]*30],'color','w')
```



Textbook problem 20.19

Note that linear regressions are based on the below formula (adopted from section 17.1.2)

$$n \text{ points } \{x_i, y_i\} \xrightarrow{\text{fit into } y=mx+b} \begin{cases} m = \dfrac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2} \\ b = \dfrac{1}{n}(\sum y - m(\sum x)) \end{cases}$$

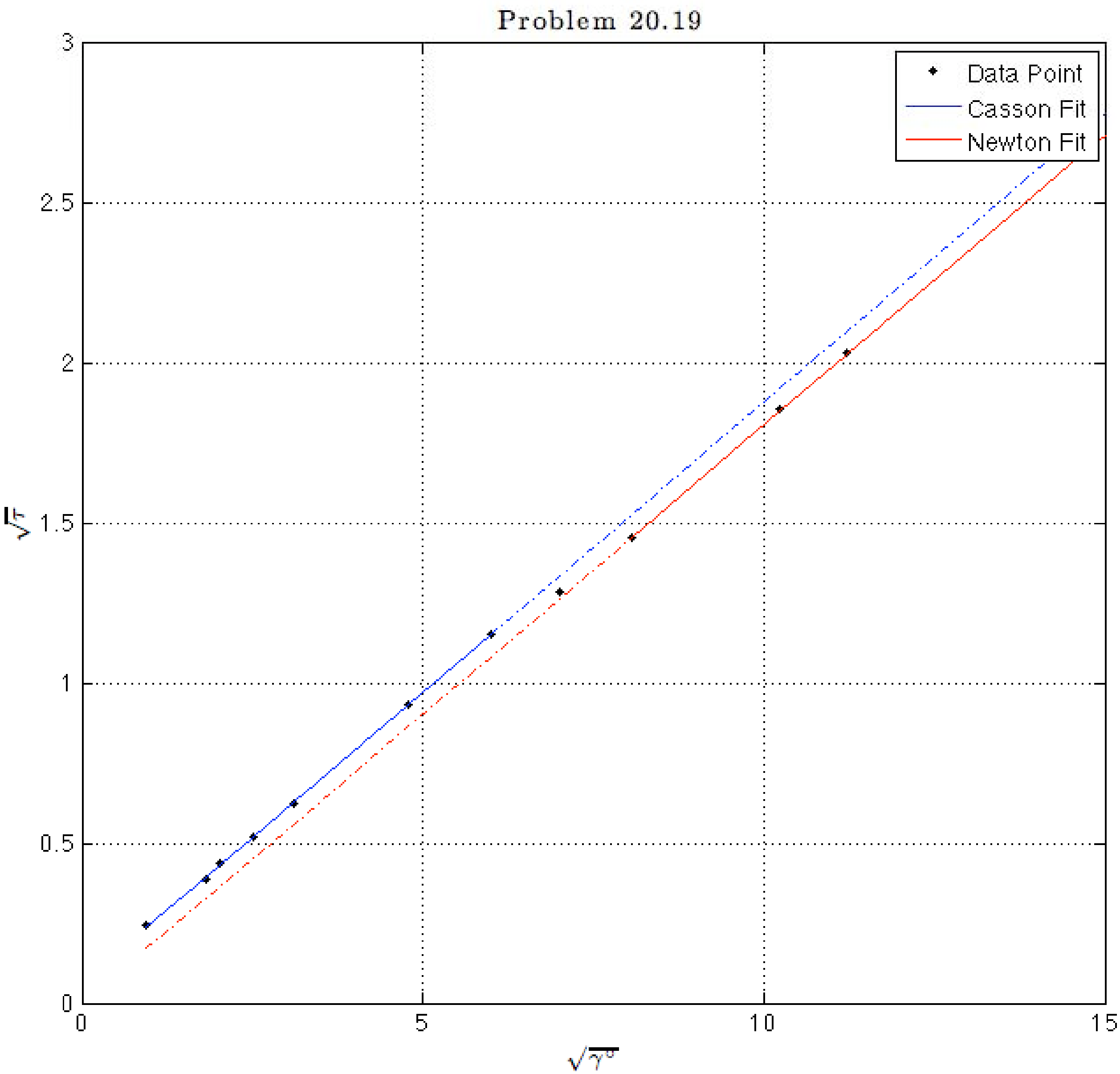
$$n \text{ points } \{x_i, y_i\} \xrightarrow{\text{fit into } y=mx} m = \frac{\sum xy}{\sum x^2}$$

Consequently for a fit like "y=mx", we cannot fit first to the "y=mx+b" and then omit the "b" term. However in both cases, the "r" is computed according to the below formula:

$$r = \sqrt{\frac{S_t - S_r}{S_t}}$$

The results are displayed below and detailed computation can be followed from attached file "C2p29_PSET3_20p19.m". Note that both fits are implemented on $(\sqrt{\dot{\gamma}}, \sqrt{\tau})$ domain. Here are the results:

```
        Casson Region
tau_y==0.00433 (N/m^2)                Kc=0.18092 (N.sec/m^2)^0.5                r=0.999925
        Newton Region
mu==0.03250 (N.sec/m^2)                     r=0.999953
```



Textbook problem 21.7

$$14^{2x} = e^{\ln(14^{2x})} = e^{2x\ln(14)} = e^{2\ln(14)x}$$

$$\int_{0.5}^{1.5} 14^{2x} dx = \int_{0.5}^{1.5} e^{2\ln(14)x} dx = \frac{1}{2\ln(14)} e^{2\ln(14)x} \Big|_{x=0.5}^{x=1.5} = \frac{1}{2\ln(14)} (e^{3\ln(14)} - e^{\ln(14)})$$

$$\int_{0.5}^{1.5} 14^{2x} dx = \frac{1}{2\ln(14)} (14^3 - 14) = \frac{1365}{\ln(14)}$$

The results are displayed on the below. The error follows our expected trend and clearly Boole's rule has the best performance. Furthermore note that due to sharp exponential rise of the function, all the closed formula have overestimated the integral, while the open ones, have all underestimated the integral.

```
>> syms x
>> a=0.5;b=1.5;
>> int_exact=int(14^(2*x),a,b)

int_exact =

1365/(log(2)+log(7))


>> int_exact=double(int_exact)

int_exact =

  517.2301

>> f=@(x) 14^(2*x);
>> int_a=(b-a)*(f(a)+f(b))/2 % Part a

int_a =

        1379

>> int_b=(b-a)*(f(a)+4*f((b+a)/2)+f(b))/6 % Part b

int_b =

  590.3333

>> int_c=(b-a)*(f(a)+3*f((b+2*a)/3)+3*f((2*b+a)/3)+f(b))/8 % Part c

int_c =

  552.3916

>> int_d=(b-a)*(7*f(a)+32*f((b+3*a)/4)+12*f((2*b+2*a)/4)+32*f((3*b+a)/4)+7*f(b))/90 % Part d

int_d =

  520.0215

>> int_e=(b-a)*f((a+b)/2) % Part e

int_e =

   196

>> int_f=(b-a)*(f(a+(b-a)/3)+f(a+2*(b-a)/3))/2 % Part f

int_f =

  276.8554

>> int_g=(b-a)*(2*f(a+(b-a)/4)-f(a+2*(b-a)/4)+2*f(a+3*(b-a)/4))/3 % Part f

int_g =

  458.4987

>> int_percent_error=100*([int_a int_b int_c int_d int_e int_f int_g]/int_exact-1)

int_percent_error =

  166.6125   14.1336    6.7980    0.5397  -62.1058  -46.4735  -11.3550
```

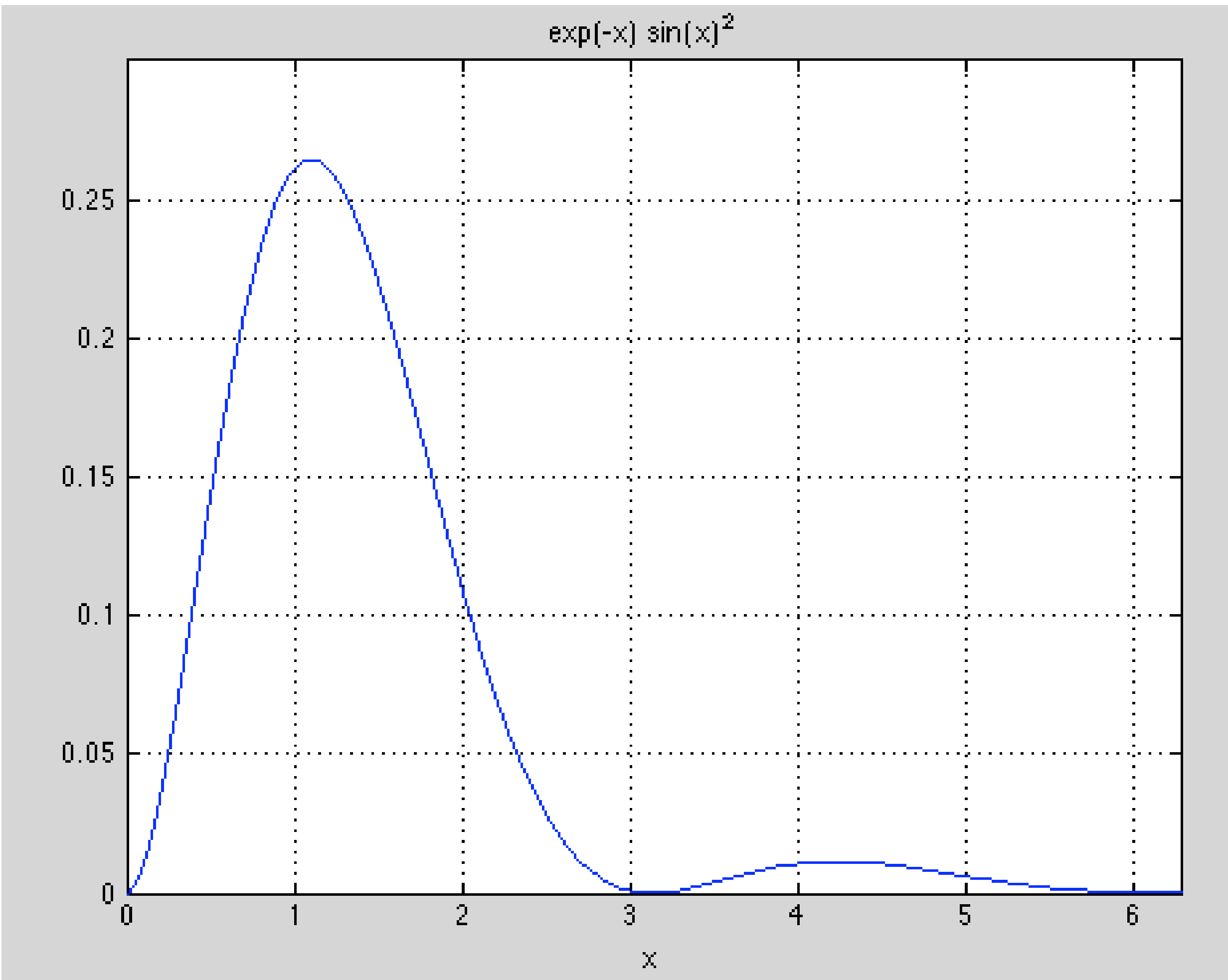
Textbook problem 22.9, Part b

$$\int_0^\infty e^{-y}\sin^2(y)dy = \int_0^\infty e^{-y}\frac{1-\cos(2y)}{2}dy$$

$$\int_0^{\infty} e^{-y}\sin^2(y)dy = \frac{1}{2}\left(\int_0^{\infty} e^{-y}dy - \int_0^{\infty} e^{-y}\cos(2y)dy\right)$$

$$\int_0^{\infty} e^{-y}\sin^2(y)dy = \frac{1}{2}\left\{-e^{-y} - \frac{e^{-y}}{5}(-\cos(2y)+2\sin(2y))\right\}\Bigg|_{y=0}^{y=\infty}$$

$$\int_0^{\infty} e^{-y}\sin^2(y)dy = -\frac{1}{2}(-1+\frac{1}{5}) = \frac{2}{5}$$



We have evaluated the analytical value for the comparison. As the above graph shows, the function strongly decays after $x = 2\pi$. Indeed as shown below, by $x = 2\pi$ the integral has already attained 99.825% of its value.

For numerical evaluation the integral has been divided to two parts (following the logics of section 22.4):

$$\int_0^{\infty} e^{-y}\sin^2(y)dy = \int_0^{2\pi} e^{-y}\sin^2(y)dy + \int_{2\pi}^{\infty} e^{-y}\sin^2(y)dy$$

$$\underbrace{\int_0^{\infty} e^{-y}\sin^2(y)\,dy}_{I} = \underbrace{\int_0^{2\pi} e^{-y}\sin^2(y)\,dy}_{I_1} + \underbrace{\int_0^{\frac{1}{2\pi}} e^{-\frac{1}{t}}\frac{\sin^2(\frac{1}{t})}{t^2}\,dt}_{I_2}$$

Since $I = I_1 + I_2$ and $I_1 >> I_2$, the error in "I" is dominated by the error in I_1. So we can very easily just approximate the integral with its truncated version (thanks to exponential decay of the function!). A take home message could be that for fast decaying function we can approximate the improper integral by truncated version.

Now regardless of this fact, we try to evaluate each part numerically. Both parts are evaluated by 5 points evaluation. The first part uses Boole's rule as closed integration, while the 2nd part is based on open integration (because $\lim_{t\to 0}\frac{1}{t^2}e^{-\frac{1}{t}}\sin^2(\frac{1}{t})$ does not exist). Detailed calculations are shown below. We can see that the error is dominated by I_1 and is about 21%. If we implement two times application of Boole's rule, the error will be dropped to 3.1%.

```
>> syms x
>> ezplot(exp(-x)*sin(x)^2,[0:.01:2*pi]), grid on, box on
>> axis([0 2*pi 0 .3])
>> int_exact=int(exp(-x)*sin(x)^2,0,inf)

int_exact =

2/5


>> I=double(int_exact);
>> int(exp(-x)*cos(2*x))

ans =

-1/5*exp(-x)*cos(2*x)+2/5*exp(-x)*sin(2*x)


>> I_1=double(int(exp(-x)*sin(x)^2,0,2*pi))

I_1 =

    0.3993

>> I_2=I-I_1

I_2 =

   7.4698e-04

>> syms t
>> F_tzero=limit(1/t^2*exp(-1/t)*sin(1/t)^2)

F_tzero =

NaN


>> f=@(x) exp(-x)*sin(x)^2;
>> I_1_num=2*pi*(7*f(0)+32*f(pi/2)+12*f(pi)+32*f(3/2*pi)+7*f(2*pi))/90 % I1

I_1_num =

    0.4845

>> f=@(x) exp(-x)*sin(x)^2;
>> I_1_num=2*pi*(7*f(0)+32*f(pi/2)+12*f(pi)+32*f(3/2*pi)+7*f(2*pi))/90 % I1

I_1_num =

    0.4845

>> F=@(t) 1/t^2*exp(-1/t)*sin(1/t)^2;
>> a=1/(2*pi);
>> I_2_num=a*(11*F(a/6)-14*F(a*2/6)+26*F(a*3/6)-14*F(a*4/6)+11*F(a*5/6))/20 % I2

I_2_num =

    0.0024
```

```
>> I2_error_percent=100*(I_2_num/I_2-1)

I2_error_percent =

  220.2633

>> I1_error_percent=100*(I_1_num/I_1-1)

I1_error_percent =

   21.3457

>> I_error_percent=100*((I_1_num+I_2_num)/I-1)

I_error_percent =

   21.7171

>> I_1_num=pi*(7*f(0)+32*f(pi/2/2)+12*f(pi/2)+32*f(3/2*pi/2)+7*f(2*pi/2))/90+... % I1
pi*(7*f(pi+0)+32*f(pi+pi/2/2)+12*f(pi+pi/2)+32*f(pi+3/2*pi/2)+7*f(pi+2*pi/2))/90

I_1_num =

    0.4117

>> I1_error_percent=100*(I_1_num/I_1-1)

I1_error_percent =

    3.1201
```

Textbook problem 22.13

According to the logics of section 22.3.2, the integration can be approximate by the below formula (for two points Gauss quadrature):

$$\int_a^b f(x)dx \cong \frac{(b-a)}{2}\left\{ f\left(\frac{b+a}{2} - \frac{1}{\sqrt{3}}\frac{(b-a)}{2}\right) + f\left(\frac{b+a}{2} + \frac{1}{\sqrt{3}}\frac{(b-a)}{2}\right)\right\}$$

Note that this formula indeed refers to below linear transformation and it can be generalized as shown:

$$x = \frac{b+a}{2} + \xi\frac{(b-a)}{2} \Rightarrow (x = a, \xi = -1),\ (x = b, \xi = 1)$$

$$F(\xi) = f\left(\frac{b+a}{2} + \xi\frac{(b-a)}{2}\right) = f(x(\xi))$$

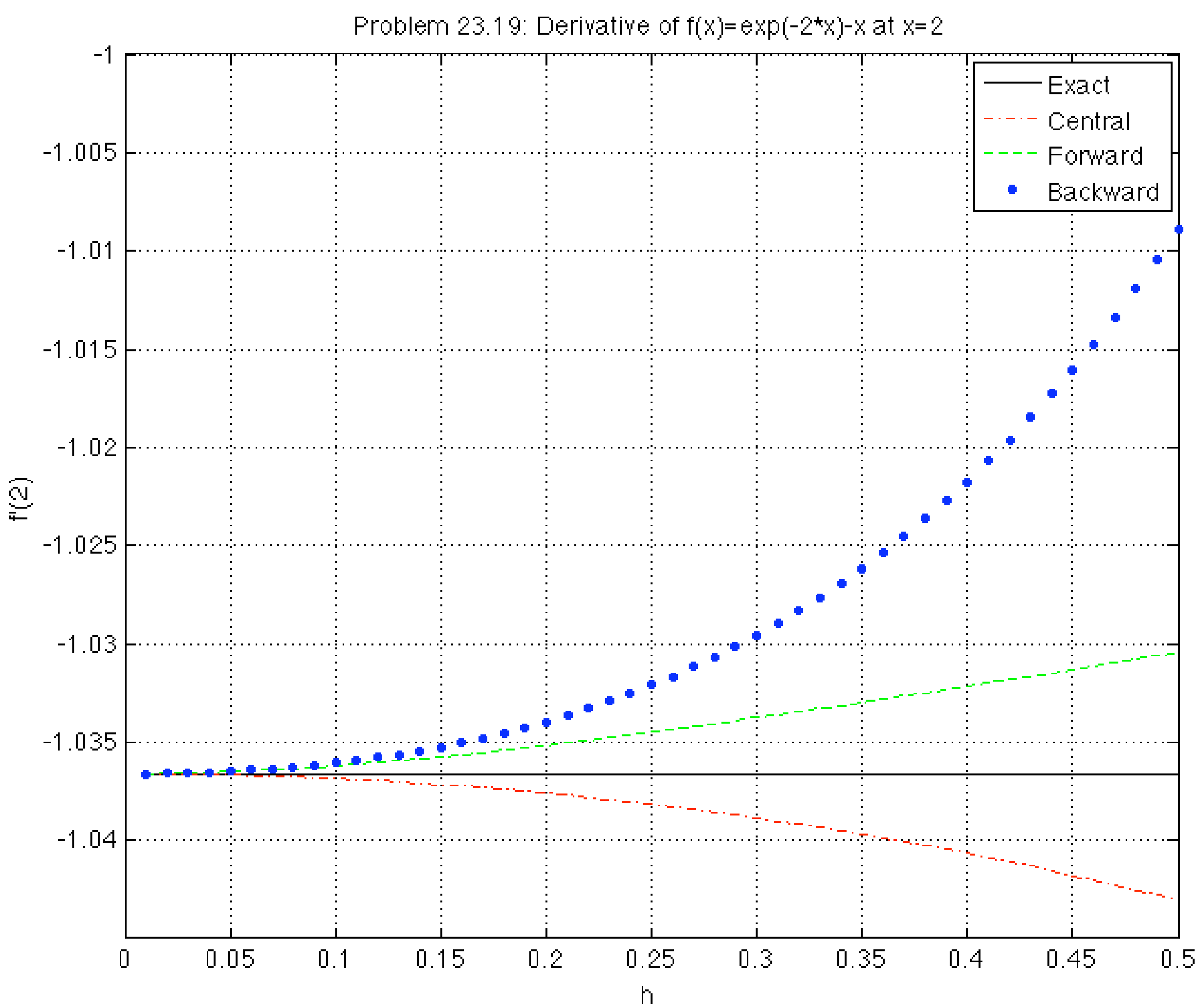
$$\int_a^b f(x)dx \cong \frac{(b-a)}{2}\sum_{i=1}^{n} c_i F(\xi_i),\ \text{where } (c_i, \xi_i) \text{ are Gauess-Quadrature Weights and Points:} \left|\xi_i\right| \le 1$$

The result is shown on the next page and is implemented as shown above. Note that as usual, the gauss-quadrature integration accuracy is absolutely excellent (0.83% error by evaluating at only two points). That's why the Gauss-quadrature method, is the preferred method for lots of applications (especially within finite element realm, where we know that our shape function is a polynomial).

```
>> I=erf(1.5) % Exact Value

I =

    0.9661

>> f=@(x) 2/pi^0.5*exp(-x^2); % Original Function
>> A=0;B=1.5; %Integral Bounds
>> T=@(z) (A+B)/2+z*(B-A)/2; % Linear Transformation
>> F=@(z) f(T(z)); % Transformated Function
>> I_num=(B-A)/2*(F(-1/3^.5)+F(+1/3^.5)) % Integral in Transformated Coordinates

I_num =

    0.9742

>> I_percent_error=100*(I_num/I-1)

I_percent_error =

    0.8351
```

Textbook problem 23.19

The graph clearly shows superior performance of central finite difference:



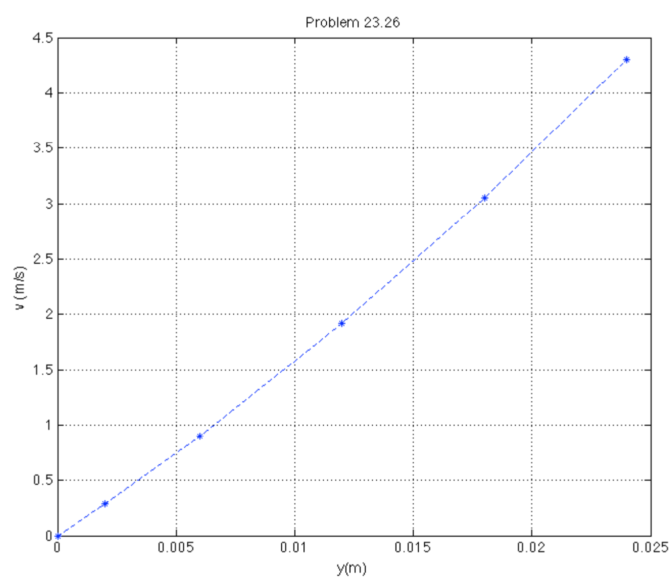
```
>> syms x
>> f=exp(-2*x)-x;
>> df=diff(f)

df =

-2*exp(-2*x)-1


>> x=2;
>> df_exact=double(eval(df)) % Part a

df_exact =

   -1.0366

>> h=[.01:.01:.5]; % Part b
>> f=@(x) exp(-2*x)-x;
>> df_central=(f(x+h)-f(x-h))./(2*h);
>> df_forward=(-f(x+2*h)+4*f(x+h)-3*f(x))./(2*h); % Part c
>> df_backward=(f(x-2*h)-4*f(x-h)+3*f(x))./(2*h);
>> % Part d
>> plot(h,df_exact*ones(size(h)),'-k',h,df_central,'r-.',h,df_forward,'g--',h,df_backward,'b.')
>> set(gcf,'Position',[50 50 [25 20]*30],'color','w')
>> set(gca,'fontsize',14)
>> xlabel('h'), ylabel('f''(2)'), title('Problem 23.19: Derivative of f(x)=exp(-2*x)-x at x=2'), legend('Exact','Central','Forward','Backward')
>>
```

(EXTRA CREDIT) Textbook problem 23.26

$$\tau\Big|_{y=0} = \mu \frac{dv}{dy}\Big|_{y=0}$$

We want to compute our derivative with at least $o(h^2)$ accuracy. However, we cannot use the central finite difference scheme and we have to rely on forward derivative approximations. Since the points are not equally spaced we will fit a polynomial to it and then we will compute the derivative of polynomial at y=0 (which is the coefficient of "y" at our fitted polynomial).

Initially we tried to use all 6 data point by fitting the data to a polynomial of order 5. This is shown on the next page. However, as the plot shows the "v(y)" graph is almost linear and the polynomial is badly conditioned. Nevertheless the linear term is rather accurate and even if we use just 3 point (2nd order polynomial fit) , we will get the same shear value with up to 0.2% accuracy.



```
>> mu=1.8e-5; % N.sec/m^2
>> y=[0 .002 .006 0.012 0.018 0.024];
>> v=[0 .287 .899 1.915 3.048 4.299];
>> p=polyfit(y,v,5)
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
> In polyfit at 80

p =

   1.0e+07 *

    1.3298   -0.0766    0.0016    0.0001    0.0000   -0.0000

>> dv_dy=p(end-1) % Derivative at y=0

dv_dy =

  140.4553

>> tau=mu*dv_dy

tau =

    0.0025

>> % Evaluated at units of N/m^2
>> p=polyfit(y(1:3),v(1:3),2)  % Use only 3 points for parabolic fit

p =

   1.0e+03 *

    1.5833    0.1403    0.0000

>> dv_dy=p(end-1) % Derivative at y=0

dv_dy =

  140.3333

>> tau=mu*dv_dy

tau =

    0.0025

>> plot(y,v,'*--')
>> xlabel('y(m)'), ylabel('v (m/s)'), title('Problem 23.26'), box on, grid on
>> set(gca,'fontsize',14)
>> set(gcf,'Position',[50 50 [25 20]*30],'color','w')
```