Problem 1 (graded by Dunzhu & Toby) X points

(a) - 10 points

Using the exact hessian, the result seems harder converge to global minimum, starting from some initial guess. For example, in the following code, we perturb original ti to ti_new, and starting from the original best guess, using approximate hessian and exact hessian we get

```
[Mapp, M] =
          315.420926236077          981.798748762528
            29.13082945887         -3197.23151318091
         -15.9074484210616          2983.63633307186
          15.8359059771525         -638.182236017816
          5.25225424962686         -89.3648967235656
```

The exact hessian converge to a local minimum, because the exact hessian is near singular at the initial guess, but the approximate hessian is much better. We can see the eigenvalue of those two hessian after the first iteration

```
[eigs(hess_app), eigs(hess)] =
          18.0176962395304          18.0254192403416
         0.098866195495413          0.1026429885873
        0.0037406745666659       0.00383013080700191
      3.13699101887875e-05      0.000138526270350115
      1.10844438320923e-05      6.51426797439541e-09
```

Note the exact hessian has an eigenvalue about 6E-9, which causes larger shift from the initial guess

```
[hess_app\grad, hess\grad] =
       -0.281987601375779         -666.651385634329
          1.23392027564989          3227.54405527842
         -1.29604880279318         -3000.78453513507
          0.116870531211363         654.168962989986
         0.0190736135015402         94.6348252902216
```

```
function hw2p1()
format long g
xi=[0 10 15 6 -7 3]';
yi=[0 0 6 13 10 7]';
zi=[0 0 0 0 0 0]';
ti=[322.418 321.031 321.228 323.093 324.415 322.706]';

% perturb the initial ti by a small amount
ti_new=[ 322.42173921685
    321.033451805079
    321.231385791978
    323.082219348377
    324.407698376676
    322.696836727301];

% use a very good initial guess
M0=[315.147372203449 30.3124981621377 -17.1481612164449 15.9867180612486
   5.26992727846599]';
```

```
[Mapp,errapp]=do_one_ti(xi,yi,zi,ti_new, M0,true, 100);
[M,err]      =do_one_ti(xi,yi,zi,ti_new, M0,false, 100);

% the exact one fail to converge to best result
[Mapp, M]


% if we check the hessian after just the first step
[M,err,hess,hess_app,grad]      =do_one_ti(xi,yi,zi,ti_new, M0,false,1);
% exact hessian has a eigenvalue that's very small, so the exact hessian is
% near singular, the approximate hessian is better
[eigs(hess_app), eigs(hess)]
% a near singular hessian gives large update, which moves the solution away
% from the best guess, and converge to a local minimum
[hess_app\grad, hess\grad]


function [M,err,hess,hess_app,grad]=do_one_ti(xi,yi,zi,ti,M0,use_app,maxstep
    )

M=M0;

for step = 1:maxstep
    grad=zeros(5,1);
    hess=zeros(5,5);      % exact hessian
    hess_app=zeros(5,5); % approximate hessian
    for i=1:length(xi)
        [tmp_grad, tmp_hess, tmp_hess_app]=get_grad_hessian(xi(i),yi(i),zi(i
            ),ti(i),M);
        grad = grad + tmp_grad;
        hess = hess + tmp_hess;
        hess_app = hess_app + tmp_hess_app;
    end
    err(step) = norm(predict(xi,yi,zi,M)-ti);
    if(step > 1 && err(step) > err(step-1))
        break;
    end
    if(use_app)
        M= M- pinv(hess_app)*grad;
    else
        M= M- pinv(hess)*grad;
    end
end

function [grad,hess,hess_approx] = get_grad_hessian(xi,yi,zi,ti,M)
ts=M(1); xs=M(2); ys=M(3); zs=M(4); v=M(5);
R=sqrt((xs-xi)^2 + (ys-yi)^2 + (zs-zi)^2);
nx=(xs-xi)/R;
ny=(ys-yi)/R;
nz=(zs-zi)/R;
```

```
e=(ts+R/v-ti);

grad = e*[1 nx/v ny/v nz/v -R/v^2]';
sen = [1 nx/v ny/v nz/v -R/v^2]';
hess_approx = sen*sen'; % approximate hess

hess=[1 nx/v ny/v nz/v -R/v^2
    0  1/v*(1/v*nx^2+e/R*(1-nx^2))  1/v*(1/v*nx*ny-e/R*nx*ny) 1/v*(1/v*nx*nz
       -e/R*nx*nz) -1*nx/v^2*(R/v+e)
    0  0  1/v*(1/v*ny^2+e/R*(1-ny^2))  1/v*(1/v*ny*nz-e/R*ny*nz) -1*ny/v^2*(
       R/v+e)
    0  0   0  1/v*(1/v*nz^2+e/R*(1-nz^2))  -1*nz/v^2*(R/v+e)
    0 0 0 0 1*R/v^3*(R/v+2*e)];
hess = hess + hess';
for k=1:5
    hess(k,k)=hess(k,k)/2;
end

function r=predict(xi,yi,zi,M)
ts=M(1); xs=M(2); ys=M(3); zs=M(4); v=M(5);
r=xi*0;
for i=1:length(xi)
    R=sqrt((xs-xi(i))^2 + (ys-yi(i))^2 + (zs-zi(i))^2);
    r(i)=(ts+R/v);
end
```

(b) - 10 points

(b.i)

Change variable $t = 1/\sigma$, then

$$\int_0^\infty \frac{1}{\sigma^N} \exp\left(-\frac{1}{2\sigma^2}A\right) d\sigma \quad = \quad \int_0^\infty t^{N-2} \exp\left(-\frac{A}{2}t^2\right) dt$$

Since

$$\int_0^\infty \exp\left(-\frac{A}{2}t^2\right) dt = \sqrt{\frac{\pi}{2}} A^{-1/2} \propto A^{-1/2}$$

$$\int_0^\infty t \exp\left(-\frac{A}{2}t^2\right) dt = A^{-1} \propto A^{-1}$$

Take derivative with respect to A, then

$$\int_0^\infty \frac{-t^2}{2} \exp\left(-\frac{A}{2}t^2\right) dt \propto A^{-1/2-1}$$

$$\int_0^\infty t\frac{-t^2}{2} \exp\left(-\frac{A}{2}t^2\right) dt \propto A^{-1-1}$$

Continue this derivative, we have

$$\int_0^\infty \left(\frac{-t^2}{2}\right)^{(N-2)/2} \exp\left(-\frac{A}{2}t^2\right) dt \propto A^{-1/2-(N-2)/2}$$

So

$$\int_0^\infty t^{N-2}\exp\left(-\frac{A}{2}t^2\right)dt \propto A^{-(N-1)/2}$$

(b.ii)

Assume uniform prior, then

$$\begin{aligned} P(x,y|d) &\propto P(d|x,y) = \exp -F(x,y) \\ &= \exp\left(-F(x_0,y_0) - gradF|_{x_0,y_0}\cdot(x-x_0,y-y_0)' - \frac{1}{2}(x-x_0,y-y_0)H|_{x_0,y_0}(x-x_0,y-y_0)'\right) \end{aligned}$$

Since at (x_0,y_0), $gradF = 0$, thus

$$P(x,y|d) \propto \exp\left(-\frac{1}{2}(x-x_0,y-y_0)H|_{x_0,y_0}(x-x_0,y-y_0)'\right)$$

write H as

$$H = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$$

thus the joint pdf will be

$$f(x,y) = K\exp\left(-\frac{1}{2}[A(x-x_0)^2 + 2B(x-x_0)(y-y_0) + C(y-y_0)^2]\right)$$

where K is the constant that normalize the pdf, to get it, we have

$$\begin{aligned} \int_{-\infty}^\infty dy\int_{-\infty}^\infty dx f(x,y) &= \int_{-\infty}^\infty dy\int_{-\infty}^\infty dx K\exp\left(-\frac{1}{2}[A(x-x_0)^2 + 2B(x-x_0)(y-y_0) + C(y-y_0)^2]\right) \\ &= \int_{-\infty}^\infty dy\int_{-\infty}^\infty dx K\exp\left(-\frac{1}{2}[Ax^2 + 2Bxy + Cy^2]\right) \\ &= \int_{-\infty}^\infty dy\int_{-\infty}^\infty dx K\exp\left(-\frac{1}{2}[A(x+By/A)^2 + (C-B^2/A)y^2]\right) \\ &= \int_{-\infty}^\infty dy K\exp\left(-\frac{1}{2}(C-B^2/A)y^2\right)\int_{-\infty}^\infty dx\exp\left(-\frac{1}{2}[A(x+By/A)^2]\right) \\ &= \int_{-\infty}^\infty dy K\exp\left(-\frac{1}{2}(C-B^2/A)y^2\right)\int_{-\infty}^\infty dx\exp\left(-\frac{1}{2}Ax^2\right) \\ &= \int_{-\infty}^\infty dy K\exp\left(-\frac{1}{2}(C-B^2/A)y^2\right)\sqrt{2\pi/A} \\ &= K\sqrt{2\pi/(C-B^2/A)}\sqrt{2\pi/A} \\ &= 2\pi K/\sqrt{AC-B^2} \\ &= 1 \end{aligned}$$

Now we want to show $E[x] = x_0$. This is true because

$$\begin{aligned}
E[x - x_0] &= \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad (x - x_0) f(x, y) \\
&\propto \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad (x - x_0) \exp\left(-\frac{1}{2}[A(x - x_0)^2 + 2B(x - x_0)(y - y_0) + C(y - y_0)^2] \right) \\
&\propto \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad x \exp\left(-\frac{1}{2}[Ax^2 + 2Bx(y - y_0) + C(y - y_0)^2] \right) \\
&\propto \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad x \exp\left(-\frac{1}{2}[Ax^2 + 2Bxy + Cy^2] \right) \\
&\propto \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad x \exp\left(-\frac{1}{2}[A(x + By/A) + (C - B^2/A)y^2] \right) \\
&\propto \int_{-\infty}^{\infty} dy (By/A) \exp\left(-\frac{1}{2}[(C - B^2/A)y^2] \right) \\
&\propto \int_{-\infty}^{\infty} dy \quad y \exp\left(-\frac{1}{2}[(C - B^2/A)y^2] \right) \\
&= 0
\end{aligned}$$

So similarly $E[y] = y_0$.

Thus

$$\begin{aligned}
\sigma_x^2 &= E[(x - E[x])^2] \\
&= E[(x - x_0)^2] \\
&= \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad \frac{\partial f(x, y)}{\partial A} / (-1/2) \\
&= -2 \frac{\partial}{\partial A} \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad f(x, y) \\
&= -2 \frac{\partial}{\partial A} 2\pi K / \sqrt{AC - B^2} \\
&= \frac{2\pi K}{(A - B^2/C)\sqrt{AC - B^2}} \\
&= \frac{C}{AC - B^2}
\end{aligned}$$

Similarly

$$\sigma_y^2 = \frac{A}{AC - B^2}$$

$$\begin{aligned}
\sigma_{xy} &= E[(x - E[x])(y - E[y])] \\
&= E[(x - x_0)(y - y_0)] \\
&= \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad \frac{\partial f(x, y)}{\partial B} / (-1) \\
&= -\frac{\partial}{\partial B} \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \quad f(x, y) \\
&= \frac{-B}{AC - B^2}
\end{aligned}$$

Note that

$$\begin{pmatrix} A & B \\ B & C \end{pmatrix}^{-1} = \frac{1}{AC - B^2} \begin{pmatrix} C & -B \\ -B & A \end{pmatrix}$$

Thus we get what we want.

Problem 2 (graded by Dunzhu & Toby) 20 points

(b) (10 points)

Here are some topics people mentioned about the ABT book:

- Appendix A, background knowledge
- Number of data points vs number of model parameter, overdetermined, underdeterminined, mixed determinied
- MAP and Bayes' theorem
- L1 norm is better when existence of outlier
- difficulty in inverse: existance, uniqueness, and instability
- Fredholm integration equation of the first kind generalize many inverse problem
- Chapter 11 Section 3, general multivarate normal case with prior
- damped Newton's method, choose the step length instead of using full newton step
- ABT mentioned p-values, chi-square statistics
- in lecture, we maximize $P(d|m)$ to get the maximum likelihood, in ABT they defined $L(m|d) = P(d|m)$ first, then maximize it. ABT one is more clear, because $P(d|m)$ apprears to be a function of d, where in fact we want it to be a function of m.

(c) (10 points)

Here are some topics people mentioned:

- When estimating σ^2, devided by $N - 1$ instead of N
- Confused about $J(m)$ and $G(m)$, it's really the same thing, Jacobian
- class did a better job of describing prior
- Class and ABT does not cover the case when prior is a given range
- numerical calculation of derivative when it's hard to do it analytically
- any method for finding global minimum?

Problem 3 (graded by Toby) 35 points

(a) (5 points)

The data d_k are the measurements $u_k = (1.97, 1.81, 1.59, 1.44)^T \text{fts}^{-1}$. The model parameters are $\mathbf{m} = (u^*, z_0)^T$. We could have picked $\mathbf{m} = (\theta, z_0)^T$, but the former choice simplifies the calculations (see hint in problem statement). The model predictions are

$$g_k = \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right). \tag{1}$$

This solves problem part (a).

(b) (5 points)

The squared error misfit function is often denoted as χ^2 and is given by

$$\chi^2(\mathbf{m}) = \frac{1}{2}\sum_k \left(d_k - g_k(\mathbf{m})\right)^2. \tag{2}$$

This function was sometimes called $F(\mathbf{m})$ in the lecture notes. As derived in the lecture notes, for the Newton-method we need to calculate the $\mathbf{G}(\mathbf{m})$ matrix defined by the column vectors

$$G_{ku^*} = \frac{1}{\kappa}\log\left(\frac{z_k + z_0}{z_0}\right), \forall k = 1, \ldots, 4 \tag{3a}$$

$$G_{kz_0} = -\frac{u^*}{\kappa}\left(\frac{z_k}{z_0(z_k + z_0)}\right), \forall k = 1, \ldots, 4 \tag{3b}$$

with the simplified Hessian $\nabla\nabla\chi^2(\mathbf{m}) \approx \mathbf{G}(\mathbf{m})^T\mathbf{G}(\mathbf{m})$ so that

$$\mathbf{m}_{n+1} = \mathbf{m}_n - \nabla\nabla\chi^2(\mathbf{m})^{-1}\left(\mathbf{d} - \mathbf{g}(\mathbf{m})\right). \tag{4}$$

Iteration with an approproate initial condition yields the best-fit answer. The MATLAB code is shown at the end of the solution set. We could have used the exact Hessian to calculate the best-fit answer, but the approximate Hessian perfroms well for reasonable initial conditions. The best-fit solution is $u^* = 0.2025$ fts^{-1} **and** $z_0 = 0.2394$ ft.

(c) (5 points)

In this part of the problem we calculate the mean veclocity U by integrating the velocity profile in the vertical. We have

$$U = \frac{1}{H}\int_0^H u\,\mathrm{d}z = \frac{1}{H}\int_0^H \frac{u^*}{\kappa}\log\left(\frac{z_k + z_0}{z_0}\right)\mathrm{d}z \tag{5a}$$

$$= \frac{z_0}{H}\frac{u^*}{\kappa}\int_1^{\frac{H+z_0}{z_0}} \log\left(z'\right)\mathrm{d}z' \tag{5b}$$

$$= \frac{z_0}{H}\frac{u^*}{\kappa}\left[\frac{H + z_0}{z_0}\left[\log\left(\frac{H + z_0}{z_0}\right) - 1\right] + 1\right] \tag{5c}$$

$$= \frac{u^*}{\kappa}\left[\frac{H + z_0}{H}\log\left(\frac{H + z_0}{z_0}\right) - 1\right] \tag{5d}$$

Plugging in our best-fit parameter, we obtain an estimate as $U = 1.5251$ fts^{-1}.

(d) (5 points)

Because the model errors are assumed to be Gaussian, we obtain a Gaussian likelihood for the data $\mathbf{d} = (d_1, \ldots, d_N)^T$

$$p(\mathbf{d}|\mathbf{m}, \sigma) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{\sum_{k=1}^{N} (d_k - \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right))^2}{2\sigma^2} \right), \tag{6}$$

where $N = 4$ is the number of data points. According to the lecture notes, marginalization with respect to σ in the case of uniform priors for σ yields a marginalized likelihood function of the form

$$p(\mathbf{d}|\mathbf{m}) \propto \left(\frac{1}{2} \sum_{k=1}^{N} (d_k - \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right))^2 \right)^{-\frac{N-1}{2}} = \exp(-F(\mathbf{m})), \tag{7}$$

which defines the function $F(\mathbf{m})$ as

$$F(\mathbf{m}) = \frac{N-1}{2} \log\left[\frac{1}{2} \sum_{k=1}^{N} \left(d_k - \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right) \right)^2 \right] \equiv \frac{N-1}{2} \log\left[\chi^2(\mathbf{m})\right], \tag{8}$$

where $\chi^2(\mathbf{m})$ stands for squared error misfit. Assuming uniform priors for $\mathbf{m}$ as well and using Bayes' theorem, we arrive at the posterior probability density function

$$p(\mathbf{m}|\mathbf{d}) = \frac{e^{-F(\mathbf{m})}}{\int \mathrm{d}\mathbf{m} e^{-F(\mathbf{m})}} = \frac{\left(\sum_{k=1}^{N} (d_k - \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right))^2 \right)^{-\frac{N-1}{2}}}{\int \left(\sum_{k=1}^{N} (d_k - \frac{u^*}{\kappa} \log\left(\frac{z_k + z_0}{z_0}\right))^2 \right)^{-\frac{N-1}{2}} \mathrm{d}\mathbf{m}}, \tag{9}$$

where we also took into account the normalization factor. As we have shown in class and on the homework sets before, because of the chain rule, the maximum a posteriori solution is given by the minimum of $F(\mathbf{m})$ and thus is equivalent to the least squares solution.

$$\nabla_{\mathbf{m}} p(\mathbf{m}|\mathbf{d}) \propto \nabla_{\mathbf{m}} F(\mathbf{m}) \propto \nabla_{\mathbf{m}} \chi^2(\mathbf{m}) = 0. \tag{10}$$

This is always the case if $F(\mathbf{m})$ is a strictly monotonic function of $\chi^2(\mathbf{m})$.

(e) (5 points)

For the sample variance estimate S^2 at the maximum a posteriori parameter values $\mathbf{m}_0$ from the lecture notes and homework sets, we have

$$S^2 \approx \frac{2\chi^2(\mathbf{m}_0)}{N-1} = \frac{\sum_{k=1}^{N} (d_k - \frac{u_0^*}{\kappa} \log\left(\frac{z_k + z_{0_0}}{z_{0_0}}\right))^2}{N-1} \approx 1.958 \cdot 10^{-3} \text{ ft}^2\text{s}^{-2}, \tag{11}$$

which similarly to previous homework sets gives an estimate for the data variance

$$\sigma^2 = \frac{S}{N} \approx 4.8950 \cdot 10^{-4} \text{ ft}^2\text{s}^{-2}. \tag{12}$$

Both of them are counted as appropriate answers to this question.

(f) (5 points)

For this part of the problem, we are first going to derive the covariance matrix, by approximating the posterior probability distribution as a Gaussian. We can do that by expanding the exponent to second order around the maximum a posteriori parameter values $\mathbf{m}_0$. We then have that

$$p(\mathbf{m}|\mathbf{d}) = \frac{e^{-F(\mathbf{m})}}{\int \mathrm{d}\mathbf{m} e^{-F(\mathbf{m})}} \approx \frac{e^{-F(\mathbf{m_0})-\frac{1}{2}(\mathbf{m}-\mathbf{m}_0)^T\mathbf{H}(\mathbf{m_0})(\mathbf{m}-\mathbf{m}_0)}}{\int \mathrm{d}\mathbf{m} e^{-F(\mathbf{m_0})-\frac{1}{2}\mathbf{H}(\mathbf{m_0})(\mathbf{m}-\mathbf{m}_0)^2}} = \frac{e^{-\frac{1}{2}(\mathbf{m}-\mathbf{m}_0)^T\mathbf{H}(\mathbf{m_0})(\mathbf{m}-\mathbf{m}_0)}}{2\pi\sqrt{\det(\mathbf{H}(\mathbf{m_0})^{-1})}} \tag{13}$$

As we have seen in class, the covariance matrix is given by the inverse of the Hessian of $F(\mathbf{m})$. In this case, we have

$$\mathbf{H}(\mathbf{m_0}) = \frac{N-1}{2\chi^2(\mathbf{m}_0)} \nabla\nabla\chi^2(\mathbf{m}_0) \tag{14}$$

, where $\nabla\nabla$ stands for the Hessian with respect to a function. From problem part (b), we already know that the Hessian of χ^2 can be approximated as $\nabla\nabla\chi^2 = \mathbf{G}^T\mathbf{G}$ at the maximum a posteriori parameter values. Thus we arrive at the covariance matrix

$$\Sigma = \mathbf{H}(\mathbf{m_0})^{-1} = \frac{2\chi^2(\mathbf{m}_0)}{N-1}\left(\mathbf{G}(\mathbf{m}_0)^T\mathbf{G}(\mathbf{m}_0)\right)^{-1} = S^2\left(\mathbf{G}(\mathbf{m}_0)^T\mathbf{G}(\mathbf{m}_0)\right)^{-1}. \tag{15}$$

The values are calculated in the MATLAB script and are given by

$$\Sigma = \begin{pmatrix} 0.01161 & 0.00278 \\ 0.00278 & 0.00067 \end{pmatrix} \tag{16}$$

in the appropriate units. This can be used to calculate an uncertainty estimate for the mean velocity U. We can perform a Taylor expansion of U around the maximum a posteriori parameter values $\mathbf{m}_0$

$$U(\mathbf{m}) = U_0 + \nabla U_{\mathbf{m_0}} \cdot (\mathbf{m}-\mathbf{m}_0) + \frac{1}{2}(\mathbf{m}-\mathbf{m}_0)^T\nabla\nabla U_{\mathbf{m_0}}(\mathbf{m}-\mathbf{m}_0) + \mathcal{O}((\mathbf{m}-\mathbf{m}_0)^3) \tag{17a}$$

$$U(\mathbf{m})^2 = U_0^2 + 2U_0\nabla U_{\mathbf{m_0}} \cdot (\mathbf{m}-\mathbf{m}_0) + U_0(\mathbf{m}-\mathbf{m}_0)^T\nabla\nabla U_{\mathbf{m_0}}(\mathbf{m}-\mathbf{m}_0) \tag{17b}$$

$$+ (\mathbf{m}-\mathbf{m}_0)^T\nabla U_{\mathbf{m_0}}\nabla U_{\mathbf{m_0}}^T \cdot (\mathbf{m}-\mathbf{m}_0) + \mathcal{O}((\mathbf{m}-\mathbf{m}_0)^3). \tag{17c}$$

Thus using the Gaussian approximation for the posterior probability distribution function the first moment of U is given by

$$\mathbb{E}[U] = \int \mathrm{d}\mathbf{m} U(\mathbf{m}) p(\mathbf{m}|\mathbf{d}) = U_0 + U_1 + \mathcal{O}((\mathbf{m}-\mathbf{m}_0)^4) \tag{18}$$

, where U_1 is just a place holder function. The second moment of U is

$$\mathbb{E}\left[U^2\right] = \int \mathrm{d}\mathbf{m} U^2(\mathbf{m}) p(\mathbf{m}|\mathbf{d}) = U_0^2 + 2U_0U_1 \tag{19a}$$

$$+ (\partial_{m_1}U)^2\Sigma_{m_1m_1} + (\partial_{m_2}U)^2\Sigma_{m_2m_2} + 2(\partial_{m_1}U\partial_{m_2}U)\Sigma_{m_1m_2} + \mathcal{O}((\mathbf{m}-\mathbf{m}_0)^4). \tag{19b}$$

Here we just carried out Gaussian integrations that have been treated on previous homework sets or in class. The approximations are valid, if the posterior probability distribution function is well approximated by a Gaussian (i.e, when N is large) and when higher order terms in the Taylor expansions for U and U^2 can be neglected. The variance of U can then be approximated as

$$\sigma_U^2 = \mathrm{Var}(U) = \mathbb{E}\left[U^2\right] - \mathbb{E}[U]^2 = (\partial_{m_1}U)^2\Sigma_{m_1m_1} + (\partial_{m_2}U)^2\Sigma_{m_2m_2} + 2(\partial_{m_1}U\partial_{m_2}U)\Sigma_{m_1m_2} \tag{20a}$$

$$+ \text{higher order terms in 1/N.} \tag{20b}$$

This is called the error propagation formalism (cf. Wikipedia). The value for σ_U using the maximum a posteriori parameter values is

$$\sigma_U \approx 0.8082. \tag{21}$$

Uncertainties can also be estimated using Monte Carlo simulations (as done on homework set 2), but the details of those simulations need to be specified and justified. The same is true for the grid search approach. Both of those alternative methods should yield uncertainties that are at most lower than the number above, as both of those methods solve the problem exactly.

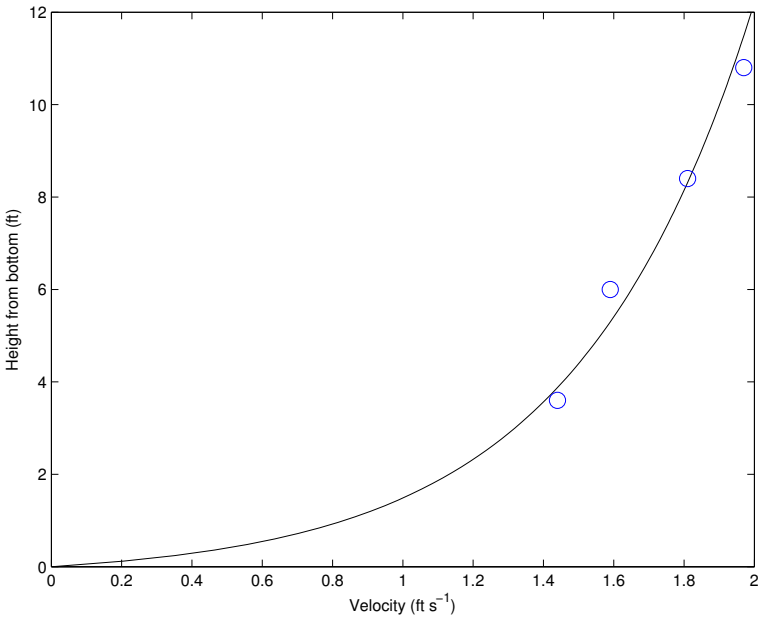

Figure 1: (b): Prediction in black and data in blue for the flow profile.

(g) (5 points)

From a physcal point of view, both z_0 and u^* need to be positive in this setup. The river flows to the right and z_0 is the roughness legnth, which is a positive quantitiy that also ensures that the argument of the logarithm does not go to zero. The roughness length is a scaling parameter and usually small. For scaling parameters, we can use the prior $p(z_0) = 1/z_0$. Because it is unlikely that our model holds for waterfalls, we need to empahsize small angles θ. For small angles $\sin(\theta) \approx \theta$ and thus $u^* \approx \sqrt{gH\theta}$. We could therefore pick a prior that emphasizes small u^*, for example $p(u^*) = 1/u^*$. For the posterior probability we find

$$p(\mathbf{m}|\mathbf{d}) = \frac{\left(\sum_{k=1}^{N}(d_k - \frac{u^*}{\kappa}\log\left(\frac{z_k+z_0}{z_0}\right))^2\right)^{-\frac{N-1}{2}} \frac{1}{z_0}\frac{1}{u^*}}{\int\left(\sum_{k=1}^{N}(d_k - \frac{u^*}{\kappa}\log\left(\frac{z_k+z_0}{z_0}\right))^2\right)^{-\frac{N-1}{2}} \frac{1}{z_0}\frac{1}{u^*}\,\mathrm{d}u^*\,\mathrm{d}z_0}. \tag{22}$$

Then, a modified $F(\mathbf{m})$ is given by

$$F(\mathbf{m}) = \frac{N-1}{2}\log\left(\sum_{k=1}^{N}(d_k - \frac{u^*}{\kappa}\log\left(\frac{z_k+z_0}{z_0}\right))^2\right) + \log(z_0) + \log(u^*). \tag{23}$$

Notice that optimizing this $F(\mathbf{m})$ is no longer equivalent to the least squares problem, because of the additional logarithmic terms. These two terms force the MAP solution to emphasize small positive values of u^* and z_0 and can be thought of as regularization terms.

Here is a plot of the profile:

MATLAB code for this problem:

```
clc; clear all; close all;
%This script finds the best-fit model parameters and returns them and the
%covariance matrix. It also plots the date vs. the model prediction

%Problem part (b)
%Measurements / data
height = 12; %depth of the river
z = [0.9 0.7 0.5 0.3]'*height; %depths
v = [1.97 1.81 1.59 1.44]'; %velocity measurements

%Initialize the least squares method
z0 = 0.2; %Roughness length
u0 = 0.1; %Velocity amplitude
tol = 1e-15; %error tolerance for iteration close to machine accuracy

%Physical parameters that define the problem
kappa = 0.4; %von Karman constant

%Set up the parameter vector and initialize it
m = [z0 u0]';

%This is the iteration loop for the Newton algorithm
err = 0.2; %Initialize error. Break loop when error is small enough
while err > tol
    %Store old values
    m_old = m;

    %Calculate the G-matrix
    u = m(2)/kappa*log((z+m(1))/m(1));
    G = [-m(2)/kappa*z./(m(1)+z)/m(1), 1/kappa*log((z+m(1))/m(1))];

    %Calculate gradient and approximate Hessian for the Newton step
    H = G'*G;
    grad = -G'*(v-u);

    %Update model parameters using the Newton step
    m = m_old - inv(H)*grad;
    err = norm(m-m_old);
end

%Problem part (c)
%Calculate the  average velocity
U = m(2)/kappa*((height+m(1))/height*log((height+m(1))/m(1))-1);

%Problem part (e)
%Calculate variance and covariance matrix
N = length(z); %Number of data points
S = 1/(N-1)*sum((v-u).^2); %Sample variance
sigma2 = S/N; %Data variance estimate
```

```
%Problem part (f)
cov = S*inv(H); %Covariance matrix
sigmaU = sqrt((1/kappa*((height+m(1))/height*log((height+m(1))/m(1))-1))^2*cov(1,1) ...
          + (m(2)/kappa*(log((height+m(1))/m(1))-1/m(1)))^2*cov(2,2) ...
          + 2/kappa*((height+m(1))/height*log((height+m(1))/m(1))-1) ...
          * m(2)/kappa*(log((height+m(1))/m(1))-1/m(1))*cov(1,2));

%Plot the predictions and the data
z_pred = (0:0.01:1)*height;
v_pred = m(2)/kappa*log((z_pred+m(1))/m(1));
plot(v_pred, z_pred, 'k-', v, z, 'bo', 'MarkerSize', 10);
xlabel('Velocity (ft s^{-1})');
ylabel('Height from bottom (ft)');
```