Assignment 4: Iterative Methods
GPGN 409

Garrett Sickles

March 28, 2016

1. *Choose a set of parameters and formulate the objective function. Plot the value of the objective function for a range of parameter values. Choose a starting model, and plot the value of the objective function for your starting model.*

   The set of parameters I have chosen for this problem are

$$\mathbf{G} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \dots & \dots & \dots \\ 1 & x_N & x_N^2 \end{bmatrix}, \vec{d} = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_N \end{bmatrix}, \vec{m} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \vec{m}_o = \begin{bmatrix} 0 \\ 2 \\ -\frac{1}{2} \end{bmatrix}$$

   as stated in our first ballistic trajectory assignment for a deterministic inversion. We can combine these matrices and vectors with the objective function

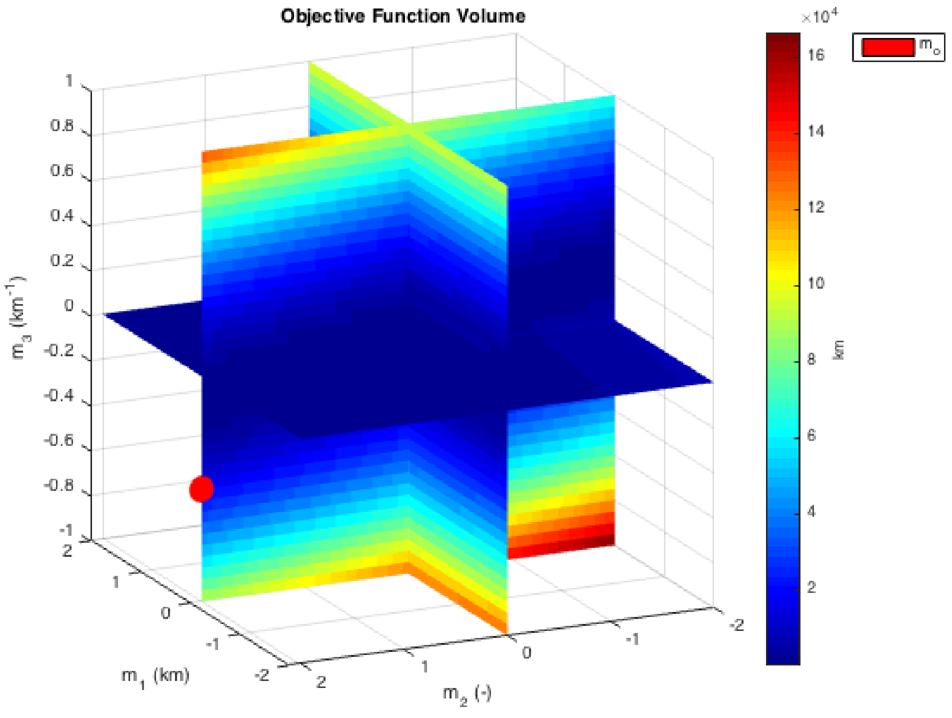
$$\mathbf{J} = \frac{1}{2}\,||\mathbf{G}\vec{m} - \vec{d}||^2 = \frac{1}{2}\vec{m}^\intercal \mathbf{A}\vec{m} - \vec{b}^\intercal \vec{m} + c$$

   such that the parameters $\mathbf{A}$, $\vec{b}$, and c are given in terms of $\mathbf{G}$ and $\vec{d}$ as seen below.

$$\begin{aligned} \mathbf{A} &= \mathbf{G}^\intercal \mathbf{G} \\ \vec{b} &= \mathbf{G}^\intercal \vec{d} \\ c &= \frac{1}{2}\vec{d}^\intercal \vec{d} \end{aligned}$$

   Using this function we can try to minimize the objective function using various methods. The two we will use in this assignment are the **Steepest Descent** and **Conjugate Gradient** methods.

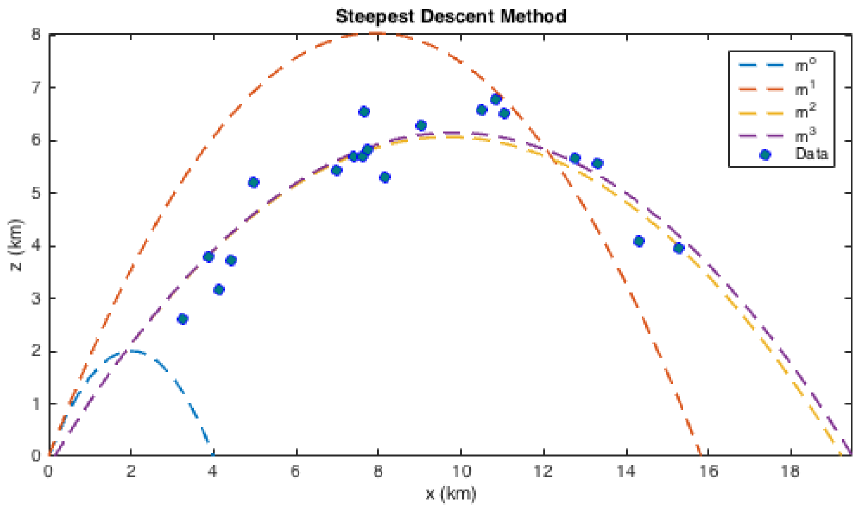
By defining a range for each model parameter as we did in class we can plot a three dimensional volume representing different values of the objective function (as seen below).



This figure shows the objective function and the initial model, m_o (red sphere), over a various range of each parameter. This starting model will be used in both the steepest descent and conjugate gradient methods.

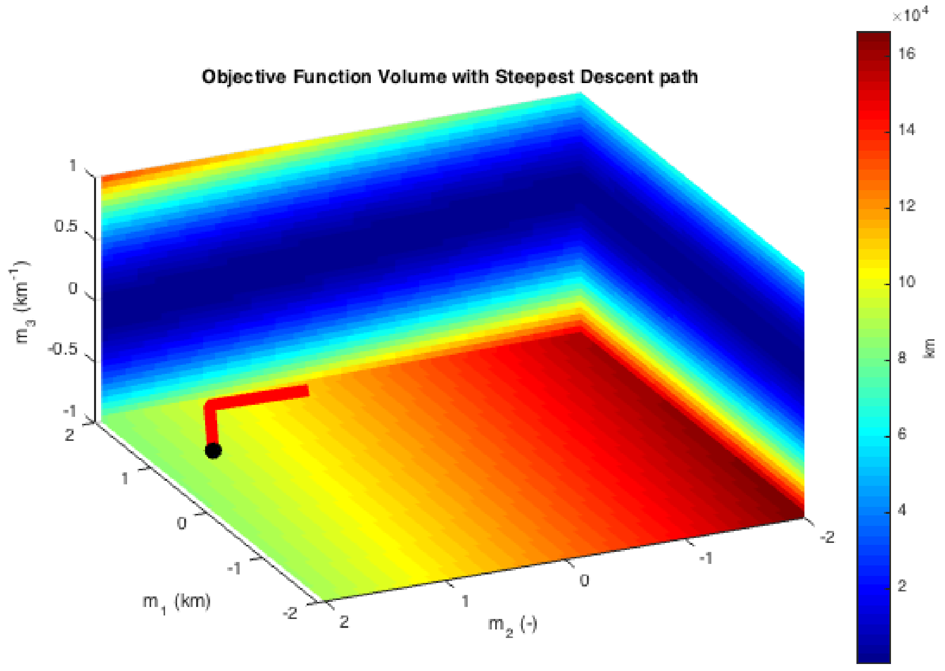
2. *Find the launch position x_o, launch angle θ, and launch speed v_o by solving the inverse problem using the* ***Steepest Descent*** *method. For the starting model and the updated model after the first three iterations, plot the trajectory that describes the motion of the object (four trajectories in total).*

   Using the steepest descent method I was able to calculate four different trajectories in the following figure and table.



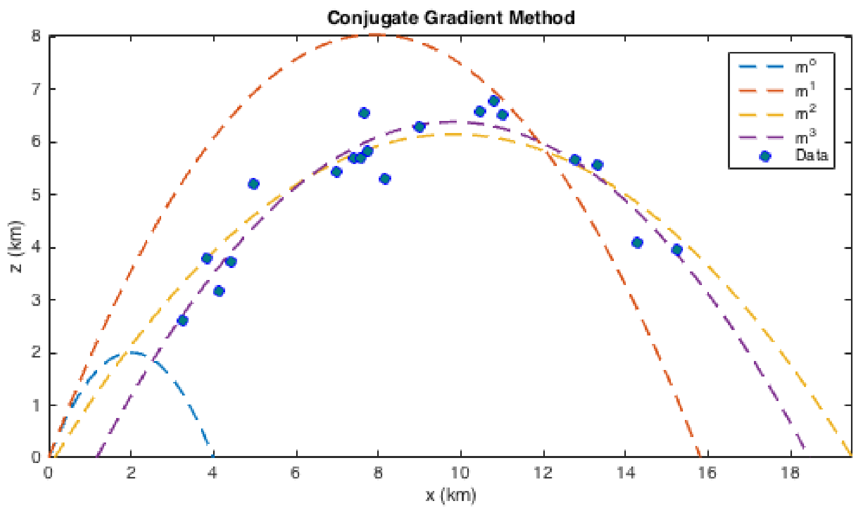
Color	x_0 (km)	$\|\vec{v}_0\|$ $\frac{km}{sec}$	θ (deg)	m^n
Blue	0.0	63.4349	0.2215	m^0
Orange	-0.0013	63.7820	0.4426	m^1
Yellow	0.1395	51.7647	0.4390	m^2
Purple	0.1395	51.7716	0.4420	m^3

The steepest descent method can also be displayed in the volume of the objective function as seen in the figure below. This shows how the model is changing in the parameter space with each iteration.



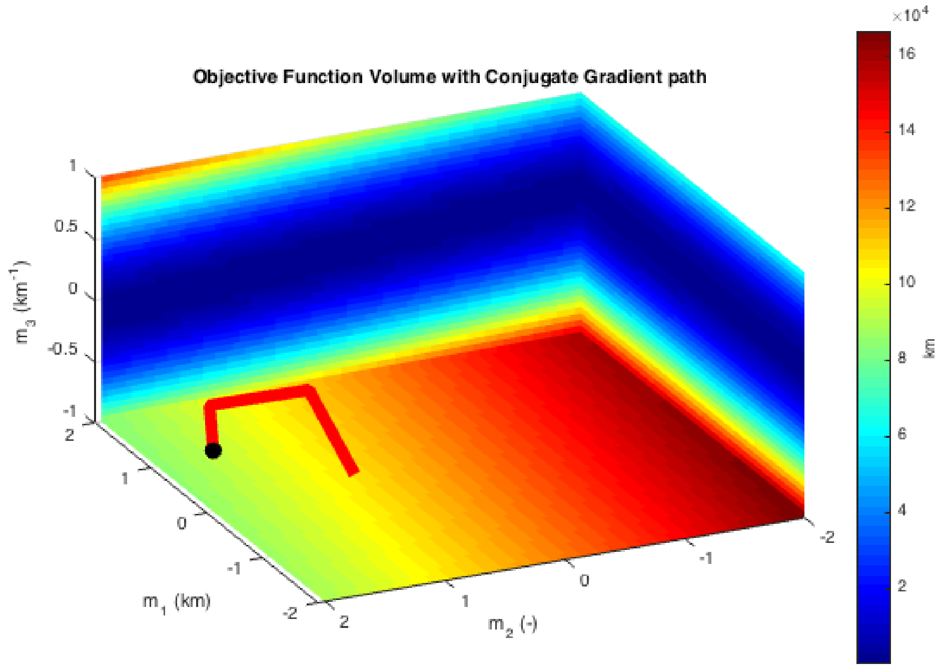
3. *Find the launch position x_o, launch angle θ, and launch speed v_o by solving the inverse problem using the* ***Conjugate Gradient*** *method. For the starting model and the updated model after the first three iterations, plot the trajectory that describes the motion of the object (four trajectories in total).*

   Using the conjugate gradient method I was able to calculate four different trajectories in the following figure and table.



Color	x_0 (km)	$\lVert\vec{v}_o\rVert$ $\frac{km}{sec}$	θ (deg)	m^n
Blue	0.0	63.4349	0.2215	m^0
Orange	-0.0013	63.7820	0.4426	m^1
Yellow	0.1400	51.7327	0.4421	m^2
Purple	1.1592	55.8294	0.4274	m^3

The steepest descent method can also be displayed in the volume of the objective function as seen in the figure below. This shows how the model is changing in the parameter space with each iteration.



Appendix I: The Code

```
function IterativeMethods()
    filename = 'data.txt';
    data = importdata(filename);
    x = data(:,1);
    z = data(:,2);

    d = [z(:)];
    G = [ones(length(x), 1) x(:) x(:).^2];

    a = -2:0.05:2;
    b = -2.0:0.05:2;
    c = -1.0:0.05:1.0;
    m_naught = [0 2.0 -0.5]';
    [mx, my, mz] = sphere;
    r = 0.06;

    SD = SteepestDescent(m_naught, d, G, 3);
    CG = ConjugateGradient(m_naught, d, G, 3);

    [J, X, Y, Z] = ObjectiveFunction(a, b, c, d, G);

    figure('Position',[400,400,600,450]);
    surf(mx*r+m_naught(1), my*2*r+m_naught(2), mz*r+m_naught(3),...
        'FaceColor',[1 0 0], 'EdgeAlpha', 0);
    hold on;
    set(slice(X,Y,Z,J,median(a),median(b),median(c)),'edgecolor','none');
    xlabel('m_{1} (km)');
    ylabel('m_{2} (-)');
    zlabel('m_{3} (km^{-1})');
    title('Objective Function Volume');
    colormap jet;
    axis tight;
    xlabel(colorbar, 'km');
    legend('m_{o}');
    view(-115, 17);
```

```
figure('Position',[400,400,600,450]);
set(slice(X,Y,Z,J,max(a),min(b),min(c)),'edgecolor','none');
xlabel('m_{1} (km)');
ylabel('m_{2} (-)');
zlabel('m_{3} (km^{-1})');
title('Objective Function Volume with Steepest Descent path');
colormap jet;
axis tight;
xlabel(colorbar, 'km');
hold on;
plot3(SD(:,1),SD(:,2),SD(:,3),'r','LineWidth',7); hold on;
hold on;
surf(mx*r+m_naught(1), my*r+m_naught(2), mz*r+m_naught(3));
axis equal;
view(-115, 22);

figure('Position',[400,400,600,450]);
set(slice(X,Y,Z,J,max(a),min(b),min(c)),'edgecolor','none');
xlabel('m_{1} (km)');
ylabel('m_{2} (-)');
zlabel('m_{3} (km^{-1})');
title('Objective Function Volume with Conjugate Gradient path');
colormap jet;
axis tight;
xlabel(colorbar, 'km');
hold on;
plot3(CG(:,1),CG(:,2),CG(:,3),'LineWidth',7); hold on;
surf(mx*r+m_naught(1), my*r+m_naught(2), mz*r+m_naught(3));
axis equal;
view(-115, 22);

figure('Position',[400,400,600,300]);
BallisticPlot(SD, [x(:) z(:)]);
xlabel('x (km)');
ylabel('z (km)');
title('Steepest Descent Method');
legend('m^{o}','m^{1}','m^{2}','m^{3}','Data');
```

```
    figure('Position',[400,400,600,300]);
    BallisticPlot(CG, [x(:) z(:)]);
    xlabel('x (km)');
    ylabel('z (km)');
    title('Conjugate Gradient Method');
    legend('m^{o}','m^{1}','m^{2}','m^{3}','Data');
end

function [J, X, Y, Z] = ObjectiveFunction(x, y, z, d, G)
    [X, Y, Z] = meshgrid(x, y, z);
    OJ = @(m) ((m'*(G'*G)*m)./2-((G'*d)'*m)+(d'*d)./2);
    J = zeros(length(y), length(x), length(z));
    for i=1:length(x)
        for j=1:length(y)
            for k=1:length(z)
                J(j,i,k) = OJ([x(i) y(j) z(k)]');
            end
        end
    end
end

function [mv] = SteepestDescent(m, d, G, n)
    mv = zeros(n+1,size(G,2));
    mv(1,:) = m';
    for i=1:n
        A = G'*G;
        b = G'*d;
        m = mv(i,:)';
        p = b-A*m;
        mv(i+1,:) = (m+p*((p'*p)/(p'*A*p)))';
    end
end

function [mv] = ConjugateGradient(m, d, G, n)
    mv = zeros(n+1,size(G,2));
    mv(1,:) = m';
    A = G'*G;
    b = G'*d;
    r_old = b-A*m;
    p = r_old;
    for i=1:n
        m = mv(i,:)';
        a = (r_old'*r_old)/(p'*A*p);
        mv(i+1,:) = (m+a*p)';
        r_new = r_old - a*A*p;
        p = r_new + ((r_new'*r_new)/(r_old'*r_old))*p;
        r_old = r_new;
    end
end

function BallisticPlot(m, d)
    for i=1:size(m,1)
        [x, y, p] = BallisticTrajectory(m(i,:));
        plot(x, y,...
            '--','LineWidth', 1.5);
        hold on;
    end
    scatter(d(:,1), d(:,2),...
        'MarkerEdgeColor','b',...
        'MarkerFaceColor',[0 .5 .5],...
        'LineWidth',1.0);
    axis tight;
    hold on;
end

function [x, y, p] = BallisticTrajectory(m)
    r = roots([m(3) m(2) m(1)]);
    x = min(r(:)):0.01:max(r(:));
    y = m(1).*ones(1, length(x)) + m(2).*x + m(3).*(x.^2);
    x0 = x(1);
    theta = atan(m(2)+(2.*m(3).*x0));
    v0 = ((-9.81*(10^(-3)))./((2.0).*m(3).*(cos(theta).^2))).^0.5;
    p = [x0 theta.*180./(3.1415926) v0];
end
```