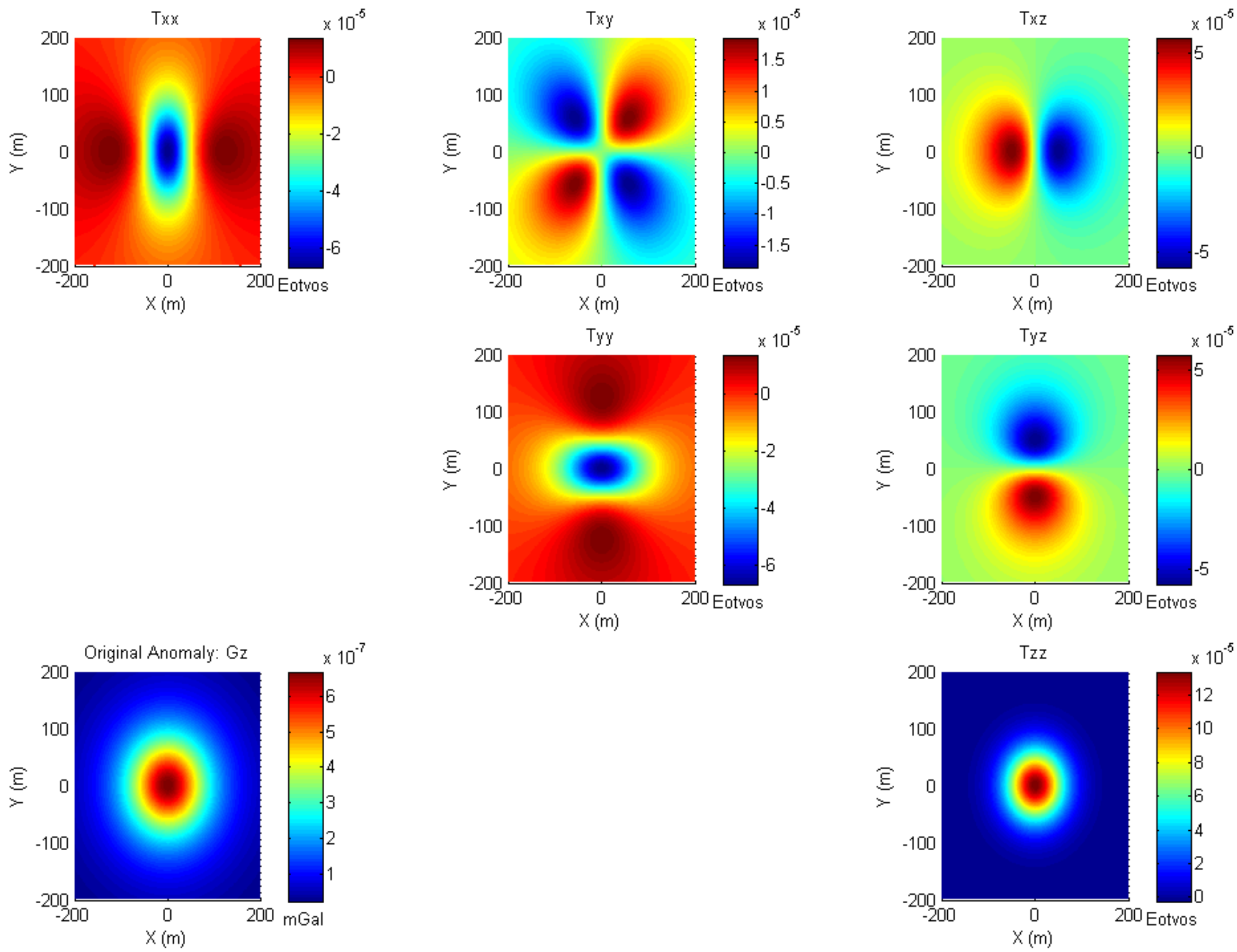
Tensor Matrix Graphs with the Original Anomaly



Command Used: Gravity_Tensor(1000, 100, 0, 0, 200, 1)

```
%Purpose:
%   The purpose of this function is to plot the tensor of the gravity response of a point
%   mass below the surface as a function of your distance from the point
%   mass using the point mass' mass, location, and depth.

%Setup:
%   Coordinate System:
%       Cartesian
%           X
%               Positive    ->  Right
%               Negative    ->  Left
%           Y
%               Positive    ->  Out of Page
%               Negative    ->  Into Page
%           Z
%               Positive    ->  Down
%               Negative    ->  Up

%Input:
%   mass    ->  Mass at the point
%                   Units: Kilograms
%   depth   ->  Magnitude of the depth of the point below the x-y plane
%                   Units: Meters
%   x_mass  ->  Displacement of the mass in x direction (Positive Right)
%                   Units: Meters
%   y_mass  ->  Displacement of the mass in y direction (Positive Out)
%                   Units: Meters
%   spread  ->  Distance of axis extension from the origin
%                   Units: Meters
%   res     ->  Distance between each gravity response measurement
%                   Units: Meters

function Gravity_Tensor(mass, depth, x_mass, y_mass, spread, res)

% Number of observations along a line of measurement
obs_total = 2 * spread / res;

% Setup arrays to store x and y coordinates and the corresponding gravity
% response
x_obs = zeros(obs_total, obs_total);
y_obs = zeros(obs_total, obs_total);
g_z = zeros(obs_total, obs_total);
T_xx = zeros(obs_total, obs_total);
T_yy = zeros(obs_total, obs_total);
T_zz = zeros(obs_total, obs_total);
T_xy = zeros(obs_total, obs_total);
T_yz = zeros(obs_total, obs_total);
T_xz = zeros(obs_total, obs_total);

% Iterate through every observation point on the observation grid and store
% data representing each measurement
for i = 1:obs_total
    for j = 1:obs_total
        % Calculate the x and y coordinates of each observation point
        x_obs(i, j) = -1.0 * spread + i * res;
        y_obs(i, j) = -1.0 * spread + j * res;

        % Calculate the gravity response at a given observation location,
        % (x,y)
        T = Gravity_Tensor_Matrix(mass, depth, x_mass, y_mass, x_obs(i, j), y_obs(i, j), 0);
        g = Gravity_Response_Vector(mass, depth, x_mass, y_mass, x_obs(i, j), y_obs(i, j), 0);
        g_z(i, j) = g(3);
        T_xx(i, j) = T(1, 1);
        T_yy(i, j) = T(2, 2);
        T_zz(i, j) = T(3, 3);
        T_xy(i, j) = T(1, 2);
        T_yz(i, j) = T(2, 3);
        T_xz(i, j) = T(1, 3);
    end;
end;

% Graph the data as a surface contour.

subplot(3, 3, 7);
surfc(x_obs, y_obs, g_z, 'EdgeColor', 'none');
title('Original Anomaly: Gz');
xlabel('X (m)');
ylabel('Y (m)');
c = colorbar('location','eastoutside');
xlabel(c, 'mGal');

subplot(3, 3, 1);
surfc(x_obs, y_obs, T_xx, 'EdgeColor', 'none');
title('Txx');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
```

```
xlabel(c, 'Eotvos');

subplot(3, 3, 5);
surfc(x_obs, y_obs, T_yy, 'EdgeColor', 'none');
title('Tyy');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
xlabel(c, 'Eotvos');

subplot(3, 3, 9);
surfc(x_obs, y_obs, T_zz, 'EdgeColor', 'none');
title('Tzz');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
xlabel(c, 'Eotvos');

subplot(3, 3, 2);
surfc(x_obs, y_obs, T_xy, 'EdgeColor', 'none');
title('Txy');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
xlabel(c, 'Eotvos');

subplot(3, 3, 6);
surfc(x_obs, y_obs, T_yz, 'EdgeColor', 'none');
title('Tyz');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
xlabel(c, 'Eotvos');

subplot(3, 3, 3);
surfc(x_obs, y_obs, T_xz, 'EdgeColor', 'none');
title('Txz');
xlabel('X (m)');
ylabel('Y (m)');
colorbar('location','eastoutside');
c = colorbar('location','eastoutside');
xlabel(c, 'Eotvos');


%Input:
%   mass    ->  Mass at the point
%                   Units: Kilograms
%   depth   ->  Magnitude of the depth of the point below the x-y plane
%                   Units: Meters
%   x_mass  ->  Displacement of the mass in the x direction
%                   Units: Meters
%   y_mass  ->  Displacement of the mass in the y direction
%                   Units: Meters
%   x_obs   ->  Displacement of the observation point in the x direction
%                   Units: Meters
%   y_obs   ->  Displacement of the observation point in the y direction
%                   Units: Meters
%   z_obs   ->  Displacement of the observation point in the z direction
%                   Units: Meters

%Output:
%   T:  Tensor Matrix for cartesian coordinate system
%
%       |T_xx,T_xy,T_xz|
%   T = |T_yx,T_yy,T_yz|
%       |T_zx,T_zy,T_zz|
%
%       Units:  Eotvos

function T = Gravity_Tensor_Matrix(mass, depth, x_mass, y_mass, x_obs, y_obs, z_obs)

%Constants
%   big_g   ->  The Universal Gravitational Constant
%                   Units: (Meters)^(3) * (Kilograms)^(-1) * (Seconds)^(-2)
%   g_conv  ->  Conversion ratio from (Meter) * (Seconds)^(-2) to milliGals
%                   Units: None
%   r_sqrd  ->  Radius from the observation point squared
%                   Units: (Meters)^2

big_g   = 6.67384E-11;
g_conv  = 1000000000;
r_sqrd = ((x_obs - x_mass)^2 + (y_obs - y_mass)^2 + (z_obs + abs(depth))^2);

coefficient = (g_conv * big_g * abs(mass));

%   Return Tensor Matrix
T(1, 1) = coefficient * ((3) * (x_obs - x_mass)^(2) * r_sqrd^(-5/2) - r_sqrd^(-3/2));
T(1, 2) = coefficient * ((3) * (y_obs - y_mass) * (x_obs - x_mass) * r_sqrd^(-5/2));
T(1, 3) = coefficient * ((-3) * (z_obs + abs(depth)) * (x_obs - x_mass) * r_sqrd^(-5/2));
T(2, 2) = coefficient * ((3) * (y_obs - y_mass)^(2) * r_sqrd^(-5/2) - r_sqrd^(-3/2));
```

```
T(2, 3) = coefficient * ((-3) * (z_obs + abs(depth)) * (y_obs - y_mass) * r_sqrd^(-5/2));
T(3, 3) = coefficient * ((3) * (z_obs + abs(depth))^(2) * r_sqrd^(-5/2) - r_sqrd^(-3/2));


%Input:
%   mass    ->  Mass at the point
%                   Units: Kilograms
%   depth   ->  Magnitude of the depth of the point below the x-y plane
%                   Units: Meters
%   x_mass  ->  Displacement of the mass in the x direction
%                   Units: Meters
%   y_mass  ->  Displacement of the mass in the y direction
%                   Units: Meters
%   x_obs   ->  Displacement of the observation point in the x direction
%                   Units: Meters
%   y_obs   ->  Displacement of the observation point in the y direction
%                   Units: Meters
%   z_obs   ->  Displacement of the observation point in the z direction
%                   Units: Meters
%
%Output:
%   g       ->  Three component gravity response vector <x, y, z>
%                   Units: mGal


function g = Gravity_Response_Vector(mass, depth, x_mass, y_mass, x_obs, y_obs, z_obs)

%Constants
%   big_g   ->  The Universal Gravitational Constant
%                   Units: (Meters)^(3) * (Kilograms)^(-1) * (Seconds)^(-2)
%   g_conv  ->  Conversion ratio from (Meter) * (Seconds)^(-2) to milliGals
%                   Units: None
%   r_sqrd  ->  Radius from the observation point squared
%                   Units: (Meters)^2

big_g   = 6.67384E-11;
g_conv  = 100000;
r_sqrd = ((x_obs - x_mass)^2 + (y_obs - y_mass)^2 + (z_obs + abs(depth))^2);

coefficient = (g_conv * big_g * abs(mass)) / (r_sqrd ^ (3/2));

%   Return Gravity Response Vector
g(1) = coefficient * (x_obs - x_mass);
g(2) = coefficient * (y_obs - y_mass);
g(3) = coefficient * (z_obs + abs(depth));
```