一、二维板模型重磁异常

求如下的平面板 ABCD 对在 OX 轴上的 P 点的重磁异常，参数在下面给出，

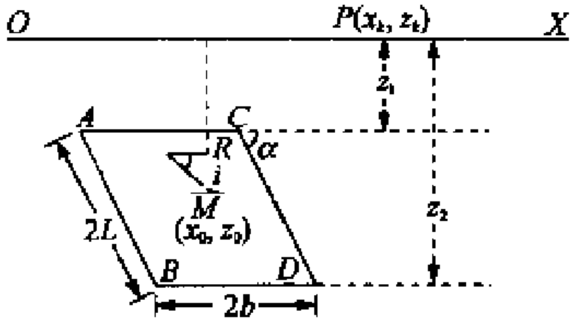


图 1.1　二维板模型

其中，$x_0 = 1000.0, z_0 = 1000.0, 2b = 400.0, 2l = 800.0, \alpha = 90(45), i = 90(45), M = 2000.0, \sigma = 2.67g \cdot cm^{-1},\ G = 6.67 \times 10^{-5}, x = 0,20,40 \ldots 2000$，测量点为 101 个。

（1）实验目的：

通过对二维板模型的重磁异常进行数值计算和分析，探究如下问题：

1.推导并实现求解平面板对指定点的重磁异常的算法。
2.理解重力异常和磁异常的计算原理，并将其应用于地球物理反演中的具体案例。
3.使用数值方法计算重力异常和磁异常，并考虑地质体的密度和磁性参数。
4.绘制重力异常和磁异常随测量点变化的曲线，并进行可视化分析。
5.考虑数值计算中的数值稳定性，采用奇异值分解等方法保证计算结果的准确性和稳定性。

（2）实验内容如下：

1.根据给定的平面板参数和测量点，计算板上各点到测量点的距离及与 X 轴正向的夹角。
2.使用几何关系和数值积分方法，计算重力异常和磁异常的值。
3.考虑地球物理反演中的倾角和偏角参数，计算总磁异常。
4.采用数值稳定性处理方法，如奇异值分解，保证计算结果的可靠性。
5.将计算结果记录到数组中，并进行可视化处理，绘制重力异常和磁异常随测量点变化的曲线图。
通过实验目的和内容的设计，可以深入理解地球物理反演中的二维板模型算法，并掌握其在实际问题中的应用。

解：

求出 ABCD 四个点分别对 P 点(x_k,z_k)的距离即r_1,r_2,r_3,r_4，根据几何关系有：

$$
\begin{aligned}
r_1^2 &= (x_k-x_0+b+l\cos\alpha)^2+(z_0-z_k-l\sin\alpha)^2\\
r_2^2 &= (x_k-x_0+b-l\cos\alpha)^2+(z_0-z_k+l\sin\alpha)^2\\
r_3^2 &= (x_k-x_0-b+l\cos\alpha)^2+(z_0-z_k-l\sin\alpha)^2\\
r_4^2 &= (x_k-x_0-b-l\cos\alpha)^2+(z_0-z_k+l\sin\alpha)^2
\end{aligned}
$$

并求出r_1,r_2,r_3,r_4与 X 轴正向的夹角，由 X 轴顺时针起算：

$$
\begin{aligned}
\varphi_1 &= \pi-\mathrm{tg}^{-1}\frac{z_0-z_k-l\sin\alpha}{x_k-x_0+b+l\cos\alpha}\\
\varphi_2 &= \pi-\mathrm{tg}^{-1}\frac{z_0-z_k+l\sin\alpha}{x_k-x_0+b-l\cos\alpha}\\
\varphi_3 &= \pi-\mathrm{tg}^{-1}\frac{z_0-z_k-l\sin\alpha}{x_k-x_0-b+l\cos\alpha}\\
\varphi_4 &= \pi-\mathrm{tg}^{-1}\frac{z_0-z_k+l\sin\alpha}{x_k-x_0-b-l\cos\alpha}
\end{aligned}
$$

（1） 对于重力异常，假设地质体的密度是均匀的，则有：

$$\Delta g = G\sigma\iint_Q \frac{z_Q-z_P}{R^3}\mathrm{d}V$$

考虑如图情况，则有：

$$
\begin{aligned}
\Delta g = 2G\sigma\{&[z_2(\varphi_2-\varphi_4)-z_1(\varphi_1-\varphi_3)]+x_k[\sin^2\alpha\ln\frac{r_2r_3}{r_1r_4}+\cos\alpha\sin\alpha(\varphi_1-\varphi_2-\varphi_3+\varphi_4)]\\
&+2b[\sin^2\alpha\ln\frac{r_4}{r_3}+\cos\alpha\sin\alpha(\varphi_3-\varphi_4)]\}
\end{aligned}
$$

（2） 对于磁异常，看成由许多体积微小的元磁体所组成，则有：

$$\mathrm{d}U=\frac{1}{4\pi R^3}(\vec{R}\cdot\vec{M}\mathrm{d}V)$$

考虑如图情况，得到磁异常的 X 和 Z 分量的值：

$$\Delta X=\frac{M}{2\pi}\sin\alpha[\ln\frac{r_2r_3}{r_1r_4}\cos(\alpha-i)-\sin(\alpha-i)(\varphi_1-\varphi_2-\varphi_3+\varphi_4)]$$

$$\Delta Z=\frac{M}{2\pi}\sin\alpha[\sin(\alpha-i)\ln\frac{r_2r_3}{r_1r_4}+\cos(\alpha-i)(\varphi_1-\varphi_2-\varphi_3+\varphi_4)]$$

其总磁异常为：

$$\Delta T=\Delta X\cos I\cos D+\Delta Y\cos I\sin D+\Delta Z\sin I$$

其中，I为地磁场倾角，D为地磁场偏角，有：

$$I=\arctan\frac{Y}{X}$$

$$D=\arctan\frac{Z}{\sqrt{X^2+Y^2}}$$

对于上式，只需将 P 点的坐标带入x_k,z_k即可得到对应的重力异常和磁异常，用数组记录下来，并绘图可视化显示，值得注意的是，数学上，$\arctan\frac{b}{a}$的主值应在$-\frac{\pi}{2}\sim\frac{\pi}{2}$区间内，但是这里所讨论的$\frac{b}{a}$实际为由坐标原点对多边形第$i$边的夹角，

这样它就有可能在$-\pi \sim \pi$之间变动，因此需要进行奇异值分解以保证数值稳定，即：

$$if\ |a| > 1.0^{-15},$$

$$a > 0.0, b > 0.0, \theta = \arctan\frac{b}{a},$$

$$a < 0.0, b < 0.0, \theta = -\pi + \arctan\frac{b}{a}$$

$$\frac{b}{a} < 0.0, \theta = -\pi + \arctan\frac{b}{a}$$

$$if\ |a| \leq 1.0^{-15},$$

$$b < 0.0, \theta = \frac{\pi}{2},$$

$$b \geq 0.0, \theta = \frac{\pi}{2}$$

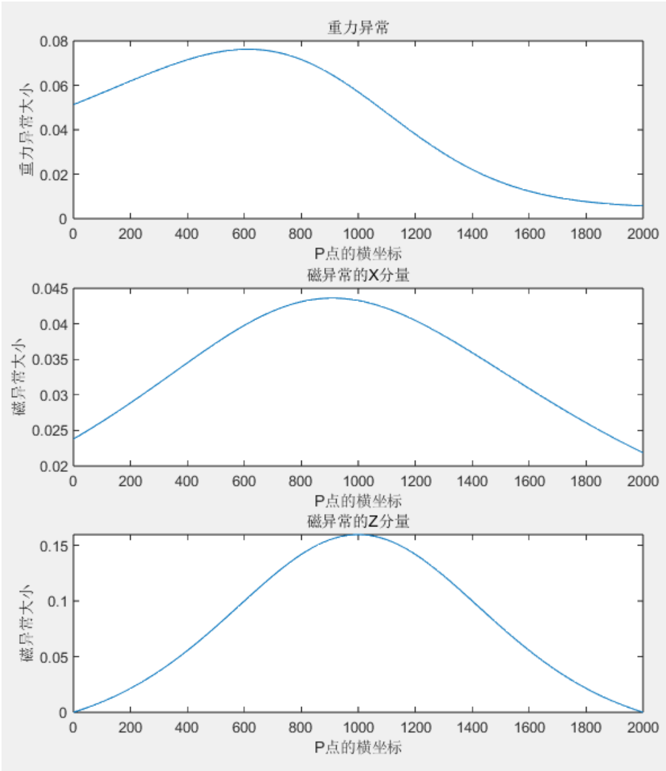
Code：

```
clear, clc, close all;
% 参数
x0 = 1000.0;
z0 = 1000.0;
b = 200;
l = 200;
angle_alpha = 90;
angle_i = 90;
M = 2000.0; % 磁化强度
xk = linspace(0, 2000, 101); % xk 即 P 点的横坐标
zk = 0; % P 点的纵坐标始终为 0
sigma = 2.67; % 剩余密度大小
G = 6.67e-5;

% 转化为弧度制
rad_alpha = deg2rad(angle_alpha);
rad_angle_i = deg2rad(angle_i);
z1 = z0 - l * sin(rad_angle_i);
z2 = z0 + l * sin(rad_angle_i);

% 创建不同位置 P 点所受磁异常的 X 分量和 Y 分量
DeltaX = zeros(size(xk));
DeltaY = zeros(size(xk));
DeltaZ = zeros(size(xk));
Deltag = zeros(size(xk));
DeltaT = zeros(size(xk));
```

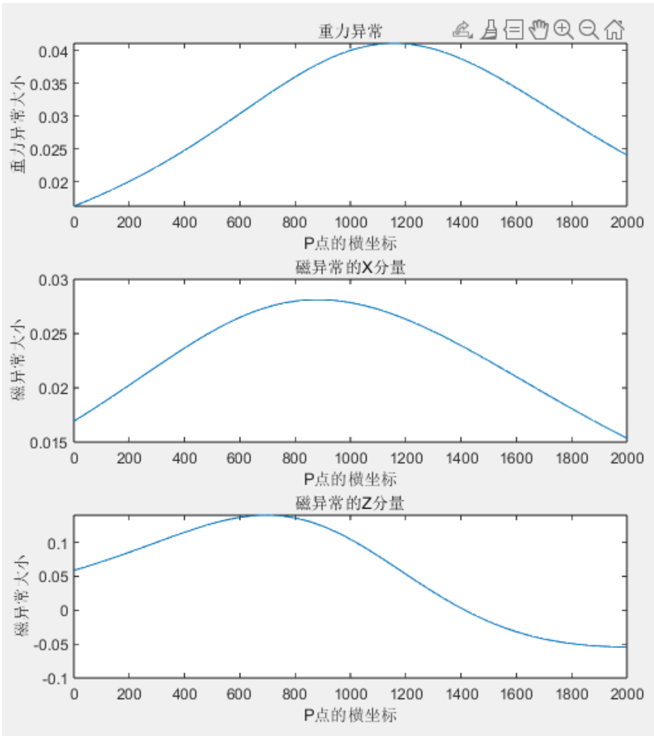
```matlab
% 奇异值特判


% 对 P 点位置进行枚举循环
for index = 1: length(xk)
    r1 = sqrt((xk(index) - x0 + b + l*cos(rad_alpha)).^2 + (z0 - zk
- l*sin(rad_alpha)).^2);
    r2 = sqrt((xk(index) - x0 + b - l*cos(rad_alpha)).^2 + (z0 - zk
+ l*sin(rad_alpha)).^2);
    r3 = sqrt((xk(index) - x0 - b + l*cos(rad_alpha)).^2 + (z0 - zk
- l*sin(rad_alpha)).^2);
    r4 = sqrt((xk(index) - x0 - b - l*cos(rad_alpha)).^2 + (z0 - zk
+ l*sin(rad_alpha)).^2);

    % 决定是否奇异
    % phi1 = pi - atan((z0 - zk - l*sin(rad_alpha)) / (xk(index) -
x0 + b + l*cos(rad_alpha)));
    % phi2 = pi - atan((z0 - zk + l*sin(rad_alpha)) / (xk(index) -
x0 + b - l*cos(rad_alpha)));
    % phi3 = pi - atan((z0 - zk - l*sin(rad_alpha)) / (xk(index) -
x0 - b + l*cos(rad_alpha)));
    % phi4 = pi - atan((z0 - zk + l*sin(rad_alpha)) / (xk(index) -
x0 - b - l*cos(rad_alpha)));
    phi1 = cal_phi(z0, zk, l, rad_alpha, xk(index), x0, b, 1);
    phi2 = cal_phi(z0, zk, l, rad_alpha, xk(index), x0, b, 2);
    phi3 = cal_phi(z0, zk, l, rad_alpha, xk(index), x0, b, 3);
    phi4 = cal_phi(z0, zk, l, rad_alpha, xk(index), x0, b, 4);

    res_X = (M / 2*pi) * sin(rad_alpha) * (log(r2 * r3) / (r1 * r4)
* cos(rad_alpha - rad_angle_i) - ...
        sin(rad_alpha - rad_angle_i) * (phi1 - phi2 - phi3 + phi4));
    res_Y = 0;
    res_Z = (M / 2*pi) * sin(rad_alpha) * (sin(rad_alpha -
rad_angle_i) * log(r2 * r3) / (r1 * r4)) + ...
        cos(rad_alpha - rad_angle_i) * (phi1 - phi2 - phi3 + phi4);

    % 重力异常
    left = z2*(phi2 - phi4) - z1*(phi1 - phi3);
    mid = xk(index) * (sin(rad_alpha).^2 * log((r2* r3) / (r1*r4)) +
cos(rad_alpha) * sin(rad_alpha) * (phi1 - phi2 - phi3 + phi4));
    right = 2 * b * (sin(rad_alpha).^2 * log(r4/r3) + cos(rad_alpha)
* sin(rad_alpha) * (phi3 - phi4));

    res_g = 2 * G * sigma * (left + mid + right);
```

```
    Deltag(index) = res_g;

    % 磁异常分量
    DeltaX(index) = res_X;
    DeltaY(index) = res_Y;
    DeltaZ(index) = res_Z;

    D = deg2rad(atan(res_Y / res_X));
    I = deg2rad(atan(res_Z / (res_X.^2 + res_Y)));

    res_T = res_X * cos(I) * cos(D) + res_Y * cos(I) * sin(D) +
res_Z * sin(I);
    DeltaT(index) = res_T;

end

% X 坐标是 xk
% 重力异常 g
subplot(3, 1, 1);
plot(xk, Deltag);

% X 坐标是 xk
% 磁异常 T
subplot(3, 1, 2);
plot(xk, DeltaX);

subplot(3, 1, 3);
plot(xk, DeltaZ);
```

Result:


重力异常
重力异常大小
P点的横坐标
磁异常的X分量
磁异常大小
P点的横坐标
磁异常的Z分量
磁异常大小
P点的横坐标

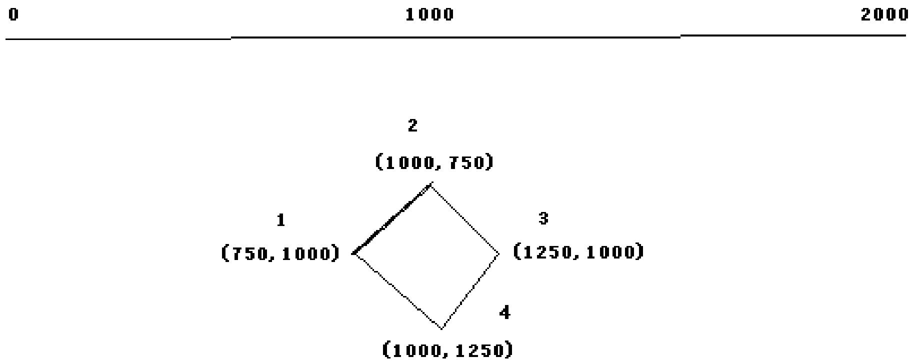
90 度


重力异常
重力异常大小
P点的横坐标
磁异常的X分量
磁异常大小
P点的横坐标
磁异常的Z分量
磁异常大小
P点的横坐标

45 度

二、截面为多边形的水平柱体的重磁异常

求如下的平面板 ABCD 对在 OX 轴上的 P 点的重磁异常，参数在下面给出，



其中，共有 4 个实际点，各点的坐标值在上图中显示，磁化强度$M = 2000.0$, 剩余密度$\sigma = 2.67g \cdot cm^{-1}$, $G = 6.67 \times 10^{-5}$,观测点$x_k = 0,20,40 \ldots 2000$，观测点一共有 101 个。

（1）实验目的：

1.通过对截面为多边形的水平柱体的重磁异常进行数值计算和分析，旨在：
2.推导并实现求解多边形板对指定观测点的重磁异常的算法。理解多边形板对重力异常和磁异常的计算原理，并将其应用于地球物理反演中的具体案例。
3.使用数值方法计算重力异常和磁异常，并考虑地质体的密度和磁性参数。
4.考虑地磁场倾角和偏角参数，计算总磁异常。
5.对计算结果进行可视化处理，绘制重力异常和磁异常随观测点变化的曲线，并进行结果分析。

（2）实验内容：

1.根据给定的多边形板参数和观测点坐标，计算实际点与观测点的相对位置。
2.利用重力异常公式和磁异常公式，对每个观测点进行重力异常和磁异常的计算。
3.考虑到多边形板的复杂形状，采用双层循环对实际点和观测点进行遍历计算。
4.实现数值稳定性处理方法，如奇异值分解，保证计算结果的准确性和稳定性。
5.将计算得到的重力异常和磁异常分量存储到相应的数组中，并绘制曲线图进行可视化展示。

解：

由于在公式中将每一个点均视为坐标原点考虑，因此在计算中先要作：

$$x_i = x_i' - x_k$$
$$z_i = z_i' - z_k$$

其中，(x_k, z_k)表示观测点的坐标，(x_i', z_i')表示第i个点的实际坐标,(x_i, z_i)为计算公式中用到的值，带入重力异常公式中则有：

$$\Delta g = 2G\sigma \sum_{i=1}^{n} \left\{ \frac{1}{2} \cdot \frac{(z_{i+1} - z_i)(x_i z_{i+1} - x_{i+1} z_i)}{(z_{i+1} - z_i)^2 + (x_{i+1} - x_i)^2} \cdot \ln \frac{x_{i+1}^2 + z_{i+1}^2}{x_i^2 + z_i^2} + \frac{(x_{i+1} - x_i)(x_i z_{i+1} - x_{i+1} z_i)}{(z_{i+1} - z_i)^2 + (x_{i+1} - x_i)^2} \cdot \left[\lg^{-1} \frac{z_i}{x_i} - \lg^{-1} \frac{z_{i+1}}{x_{i+1}} \right] \right.$$

通过对观测点和实际点的双层循环，计算可得磁异常的 P 分量和 Q 分量：

$$Q = \sum_{i=1} \frac{(z_{i+1} - z_i)(x_i - x_{i+1})}{(z_{i+1} - z_i)^2 + (x_i - x_{i+1})} \cdot \left[tg^{-1} \frac{z_i}{x_i} - tg^{-1} \frac{z_{i+1}}{x_{i+1}} - \frac{1}{2} \cdot \frac{(z_{i+1} - z_i)^2}{(z_{i+1} - z_i)^2 + (x_i - x_{i+1})^2} \right] \cdot \ln \frac{x_{i+1}^2 + z_{i+1}^2}{x_i^2 + z_i^2}$$

$$P = \sum_{n=1}^{n} \left[\frac{(z_{i+1} - z_i)^2}{(z_{i+1} - z_i)^2 + (x_i - x_{i+1})^2} \right] \cdot \left[tg^{-1} \frac{z_i}{x_i} - tg^{-1} \frac{z_{i+1}}{x_{i+1}} \right]$$

将其累加并存入磁异常分量的 X 和 Z 分量数组中：

$$\Delta X = \frac{1}{2\pi}(M_x P + M_z Q)$$

$$\Delta Z = \frac{1}{2\pi}(M_x Q - M_z P)$$

至此即可绘制出对于每个观测点对应的重力异常和磁异常分量。值得注意的是的，也需要进行tg^{-1}的奇异值进行分解。

Code:

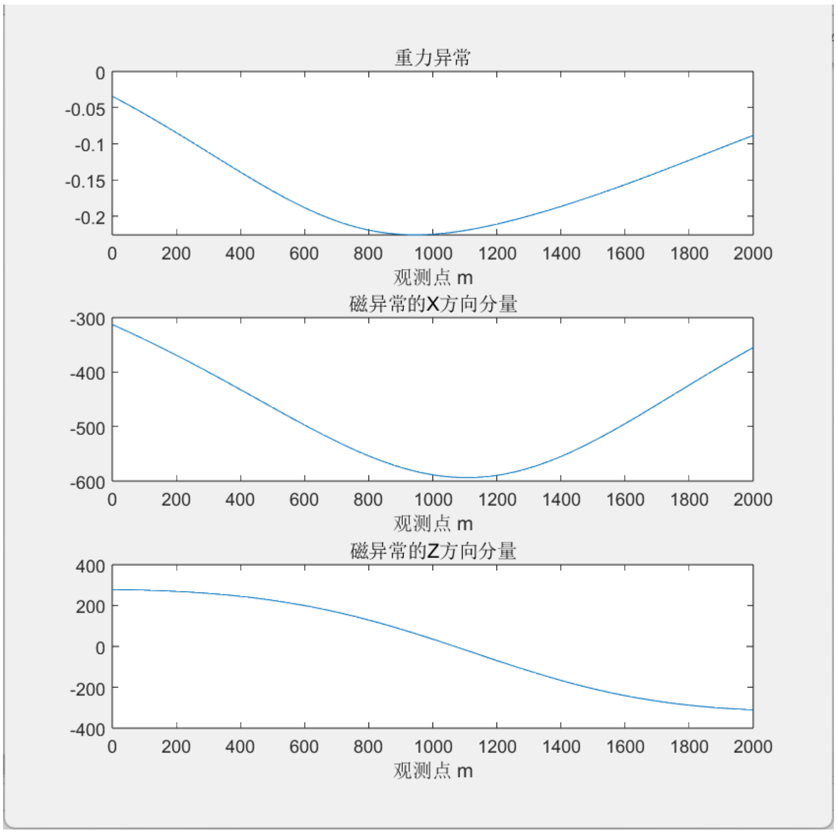
```
% 截面为多边形的水平柱体
clear, clc, close all;

% 参数
M = 2000.0; % 磁化强度
Mx = 1800.0;
Mz = 1600.0;
sigma = 2.67; % 剩余密度
G = 6.67e-5;

n = 4;
Xi = [750, 1000, 1250, 1000];
Zi = [1000, 750, 1000, 1250];

% 观测点
xk = linspace(0, 2000, 101);
zk = zeros(1, length(xk));

Deltag = zeros(1, length(xk));
```

```
DeltaX = zeros(1, length(xk));
DeltaZ = zeros(1, length(xk));

% 核心计算
% 对观测点进行循环
for index_i = 1: length(xk)
    % 实际点 - 观测点
    Xtemp = zeros(1, n);
    Ztemp = zeros(1, n);
    sum_of_g = 0;
    sum_of_q = 0;
    sum_of_p = 0;

    % 对实际点进行循环
    for index_j = 1: n
        Xtemp(index_j) =  Xi(index_j) - xk(index_i);
        Ztemp(index_j) = Zi(index_j) - zk(index_i);
    end


    for index_j = 1: n-1
        % 重力异常的计算
        % 左半部分
        leftup = 1/2 * (Ztemp(index_j+1) - Ztemp(index_j) *
(Xtemp(index_j) * Ztemp(index_j+1) - Xtemp(index_j+1) *
Ztemp(index_j)));
        leftdown = (Ztemp(index_j+1) - Ztemp(index_j)).^2 +
(Xtemp(index_j+1) - Xtemp(index_j)).^2;
        left_right = log((Xtemp(index_j+1).^2 + Ztemp(index_j+1).^2 )
/ (Xtemp(index_j).^2 + Ztemp(index_j).^2) );

        % 右半部分
        rightup = (Xtemp(index_j+1) - Xtemp(index_j)) *
(Xtemp(index_j) * Ztemp(index_j+1) - Xtemp(index_j+1) *
Ztemp(index_j));
        rightdown = (Ztemp(index_j+1) - Ztemp(index_j) ).^2 +
(Xtemp(index_j+1) - Xtemp(index_j)).^2;
        right_right = cal_atan(Ztemp(index_j), Xtemp(index_j)) -
cal_atan(Ztemp(index_j+1), Xtemp(index_j+1));

        % 汇总
        sum_of_g = sum_of_g + ((leftup * left_right) / leftdown +
(rightup * right_right) / rightdown);
```

```
        % 磁异常分量的计算
        Q_left_up = (Ztemp(index_j+1) - Ztemp(index_j)) *
(Xtemp(index_j) - Xtemp(index_j+1));
        Q_left_down = (Ztemp(index_j+1) - Ztemp(index_j)).^2 +
(Xtemp(index_j) - Xtemp(index_j+1)).^2;
        Q_left_right = cal_atan(Ztemp(index_j), Xtemp(index_j)) -
cal_atan(Ztemp(index_j), Xtemp(index_j));

        Q_right_up = -1/2 * (Ztemp(index_j+1) - Ztemp(index_j)).^2;
        Q_right_down = (Ztemp(index_j+1) - Ztemp(index_j)).^2 +
(Xtemp(index_j) - Xtemp(index_j+1)).^2;
        Q_right_right = log( (Xtemp(index_j+1).^2 +
Ztemp(index_j+1).^2) / (Xtemp(index_j).^2 + Ztemp(index_j).^2));

        % 汇总
        sum_of_q = ((Q_left_up * Q_left_right / Q_left_down) +
(Q_right_up * Q_right_right / Q_right_down));
        % fprintf("sum_of_q : %f\n", sum_of_q);
        P_left_up = (Ztemp(index_j+1) - Ztemp(index_j)).^2;
        P_left_down = (Ztemp(index_j+1) - Ztemp(index_j)).^2 +
(Xtemp(index_j) - Xtemp(index_j+1)).^2;
        P_left_right = cal_atan(Ztemp(index_j), Xtemp(index_j)) -
cal_atan(Ztemp(index_j+1), Xtemp(index_j+1));

        % 汇总
        sum_of_p = (P_left_up * P_left_right / P_left_down);
        % fprintf("sum_of_p :%f\n", sum_of_p);
    end

    Deltag(index_i) = 2 * G * sigma * sum_of_g;
    DeltaX(index_i) = (1/2*pi) * (Mx * sum_of_p + Mz * sum_of_q);
    DeltaZ(index_i) = (1/2*pi) * (Mx * sum_of_q - Mz * sum_of_p);
end


subplot(3, 1, 1);
plot(xk, Deltag);
xlabel("观测点 m");
title("重力异常");

subplot(3, 1, 2);
plot(xk, DeltaX);
xlabel("观测点 m");
title("磁异常的 X 方向分量");
```

```
subplot(3, 1, 3);
plot(xk, DeltaZ);
xlabel("观测点 m");
title("磁异常的 Z 方向分量");
```

Result：



结果分析：在靠近水平柱体时，重力异常的值突然增大，并且光滑趋近，对于磁异常的 X 分量也是如此；但是对于磁异常的 Z 分量，由于 4 个点之间的差异并不明显，故增长降落趋势较为缓慢。

三、求空间中长方体的重磁异常

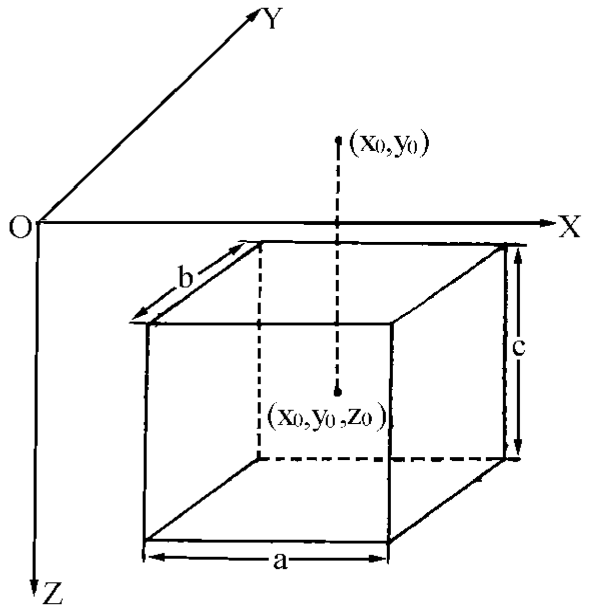


图1. 3

（1）实验目的：
1. 通过对正方体的重力异常和磁异常进行数值计算和分析，旨在：
2. 推导并实现求解正方体对指定观测点的重力异常和磁异常的算法。理解正方体对重力异常和磁异常的计算原理，并将其应用于地球物理反演中的具体案例。
3. 使用数值方法计算重力异常和磁异常，并考虑地质体的密度和磁性参数。
4. 考虑地磁场倾角和偏角参数，计算总磁异常。
5. 对计算结果进行可视化处理，绘制重力异常和磁异常随观测点变化的曲线，并进行结果分析。

（2）实验内容：
1. 根据给定的正方体参数和观测点坐标，计算实际点与观测点的相对位置。
2. 利用重力异常公式和磁异常公式，对每个观测点进行重力异常和磁异常的计算。
3. 考虑到正方体的复杂形状，采用双层循环对实际点和观测点进行遍历计算。
4. 实现数值稳定性处理方法，如奇异值分解，保证计算结果的准确性和稳定性。
5. 将计算得到的重力异常和磁异常分量存储到相应的数组中，并绘制曲线图进行可视化展示。

解：
对空间中的长方体重力异常有：

$$
\begin{aligned}
\Delta g = -G\sigma\Big\{&(x-x_k)\ln\left[(y-y_k)+R\right] \\
&+(y-y_k)\ln\left[(x-x_k)+R\right] \\
&+(z-z_k)\cdot tg^{-1}\frac{(Z-Z_k)R}{(x-x_k)(y-y_k)} \\
&\Bigg|_{x_0-a/2}^{x_0+a/2}\Bigg|_{y_0-b/2}^{y_0+b/2}\Bigg|_{z_0-c/2}^{z_0+c/2}
\end{aligned}
$$

对空间中的磁异常分量 XYZ 有：

$$
\begin{aligned}
\Delta X = \frac{1}{4\pi}\Big\{&-M_x tg^{-1}\frac{(y-y_k(z-z_k)}{(x-x_k)R}+M_y\ln\left[R+(z-z_k)\right] \\
&M_z\ln\left[R+(y-y_k)\right]\Big\}\Bigg|_{x_0-a/2}^{x_0+a/2}\Bigg|_{y_0-b/2}^{y_0+b/2}\Bigg|_{z_0-c/2}^{z_0+c/2}
\end{aligned}
$$

$$
\begin{aligned}
\Delta Y = \frac{1}{4\pi}\Big\{&M_x\ln\left[R+(z-z_k)\right]-M_y tg^{-1}\frac{(x-x_k)(z-z_k)}{(y-y_k)R} \\
&+M_z\ln\left[R+(x-x_k)\right]\Big\}\Bigg|_{y_0-a/2}^{x_0+a/2}\Bigg|_{y_0-b/2}^{y_0+b/2}\Bigg|_{z_0-c/2}^{z_0+c/2}
\end{aligned}
$$

$$
\begin{aligned}
\Delta Z = \frac{1}{4\pi}\Big\{&M_x\ln\left[R+(y-y_k)\right]-M_y\ln\left[R+(x-x_k)\right] \\
&-M_z tg^{-1}\frac{(x-x_k)}{(z-z_k)}\frac{(y-y_k)}{R}\Big\}\Bigg|_{x_0-a/2}^{x_0+a/2}\Bigg|_{y_0-b/2}^{y_0+b/2}\Bigg|_{z_0-c/2}^{z_0+c/2}
\end{aligned}
$$

其中 R 为观测点到长方体上顶点的距离为：

$$R=\left[(x-x_k)^2+(y-y_k)^2+(z-z_k)^2\right]^{1/2}$$

在计算过程中，需要代入并分解，

$$
\begin{gathered}
f(x)|_a^b = f(b)-f(a) \\
f(x,y,z)|_{x_1}^{x_2}|_{y_1}^{y_2}|_{z_1}^{z_2} \\
=[f(x_2,y,z)-f(x_1,y,z)]|_{y_1}^{y_2}|_{z_1}^{z_2}
\end{gathered}
$$

可以用三层循环枚举表示：

for(i=1;i<=2;i++) $x_1, x_2 \rightarrow x_i$

for(j=1;j<=2;j++) $y_1, y_2 \rightarrow y_i$

for(m=1;m<=2;m++) $z_1, z_2 \rightarrow z_i$

{

$$sign=\begin{cases}+1; & if((i+j+m)\%2==0) \\ -1; & if((i+j+m)\%2==1)\end{cases}$$

$gravity \mathrel{+}= sign * f(x_i, y_j, z_m)$

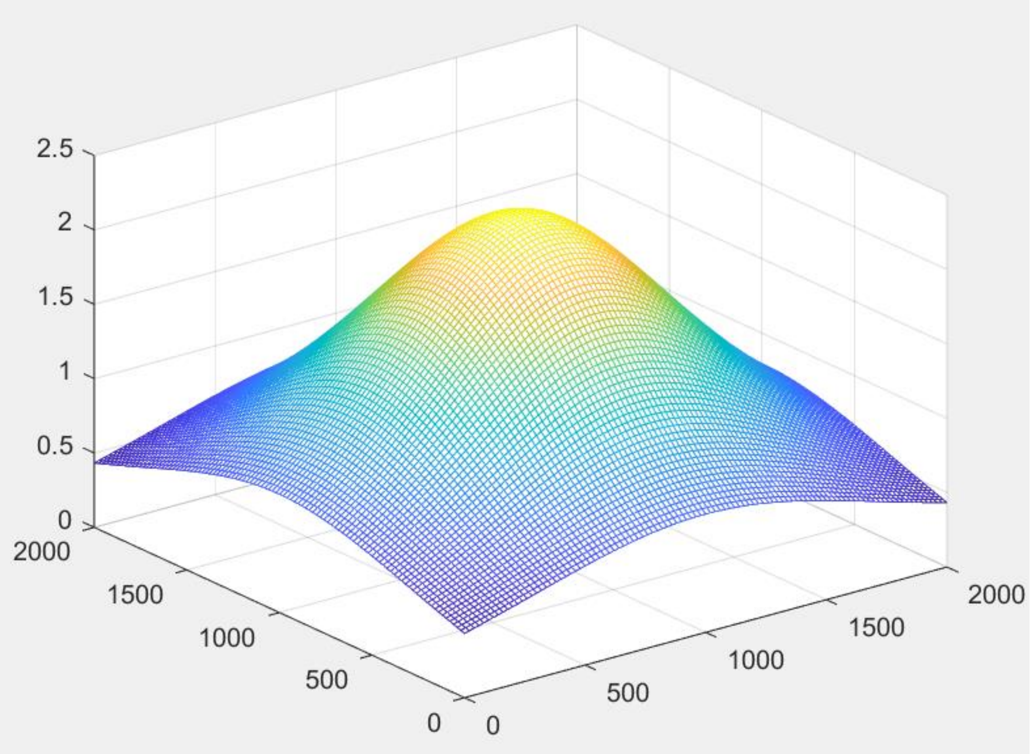
至此即可绘制出对于每个观测点对应的重力异常和磁异常分量。值得注意的是的，也需要进行tg^{-1}的奇异值进行分解。

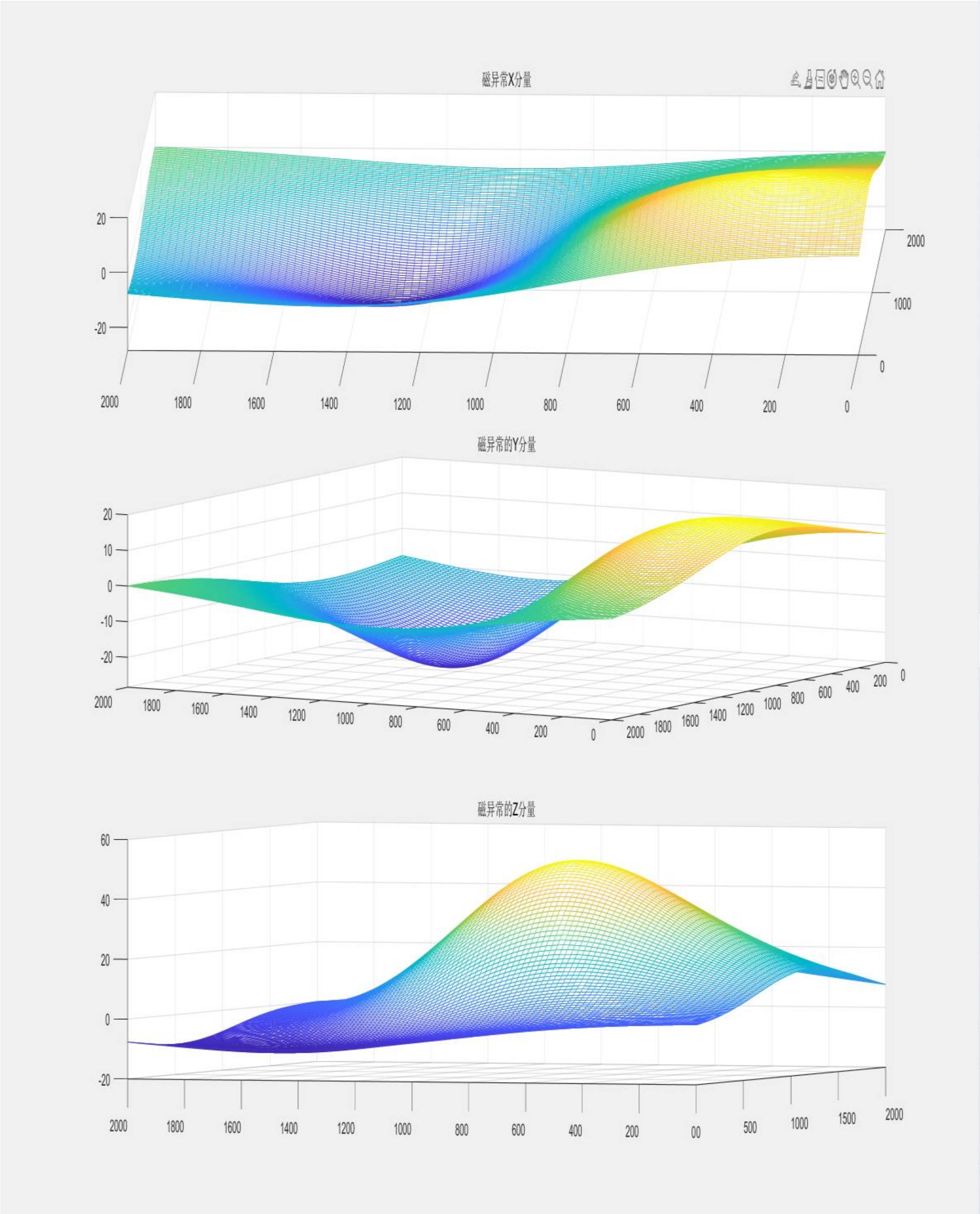
Code:
```
clear, clc, close all;

% 参数
x0 = 1000;
y0 = 1000;
z0 = 1000;
a = 500;
b = 500;
c = 500;

% 位移序列
xk = linspace(0, 2000, 101);
yk = linspace(0, 2000, 101);
zk = zeros(1, length(xk)); % z 方向始终为 0
G = 6.67e-5; % 引力常数
sigma = 2.67; % 剩余密度

x = [x0 + a/2, x0 - a/2];
y = [y0 + b/2, y0 - b/2];
z = [z0 + c/2, z0 - c/2];

%% (1) 重力异常
Gravity2D = zeros(length(xk), length(yk));

for i = 1: length(xk)
    for j = 1: length(yk)
        sum_of_g = 0;
        for k = 1: length(zk)
            for index_i = 1: 2
                tempX = x(index_i);
                for index_j = 1: 2
                    tempY = y(index_j);
                    for index_k = 1: 2
                        tempZ = z(index_k);

                        if mod(index_i + index_j + index_k, 2) == 0
                            sign = 1;
                        else
                            sign = -1;
                        end
```

```
                        % 计算观测点对边界点的单个力
                        sum_of_g = sum_of_g + ...
                               sign * cal_g(tempX, tempY, tempZ,
xk(i), yk(j), zk(k));
                    end
                end
            end
        end
        Gravity2D(i, j) = G * sigma * sum_of_g;
    end
end

figure;
[X, Y] = meshgrid(xk, yk);
mesh(X, Y, Gravity2D);

%% (2)磁异常

XMag2D = zeros(length(xk), length(zk));
YMag2D = zeros(length(xk), length(zk));
ZMag2D = zeros(length(xk), length(zk));

for i = 1:length(xk)
    for j = 1:length(yk)
        for k = 1:length(zk)
            sum_of_X = 0;
            sum_of_Y = 0;
            sum_of_Z = 0;
            for index_i = 1:2
                for index_j = 1:2
                    for index_k = 1:2
                        if mod(index_i + index_j + index_k, 2) == 0
                            sign = 1;
                        else
                            sign = -1;
                        end
                        % 计算观测点对边界点的单个力
                        sum_of_X = sum_of_X + sign *
calculate_X(index_i, index_j, index_k, xk(i), yk(j), zk(k));
                        sum_of_Y = sum_of_Y + sign *
calculate_Y(index_i, index_j, index_k, xk(i), yk(j), zk(k));
                        sum_of_Z = sum_of_Z + sign *
calculate_Z(index_i, index_j, index_k, xk(i), yk(j), zk(k));
                    end
```

```
                end
            end
            XMag2D(i, j) = sum_of_X;
            YMag2D(i, j) = sum_of_Y;
            ZMag2D(i, j) = sum_of_Z;
        end
    end
end

figure;

subplot(3, 1, 1);
mesh(X, Y, XMag2D);
title('磁异常 X 分量');

subplot(3, 1, 2);
mesh(X, Y, YMag2D);
title('磁异常的 Y 分量');

subplot(3, 1, 3);
mesh(X, Y, ZMag2D);
title("磁异常的 Z 分量");
```

Result:




磁异常X分量
20
0
-20
2000
1800
1600
1400
1200
1000
800
600
400
200
0
2000
1000
0
磁异常的Y分量
20
10
0
-10
-20
2000
1800
1600
1400
1200
1000
800
600
400
200
0
磁异常的Z分量
60
40
20
0
-20
2000
1800
1600
1400
1200
1000
800
600
400
200
0
500
1000
1500
2000

四、二维阻尼板最小二乘法反演程序

（1） 实验目的

1. 熟悉二维板正演流程
2. 熟悉二维板反演流程
3. 熟悉正演到反演的转换

（2） 实验内容

解：

<1> 计算理论函数f_k:当给定参数值和测点坐标，计算出各点的f_k值，它由所选的模型体而决定的一套正演计算公式所构成，具体参考实验一中的：

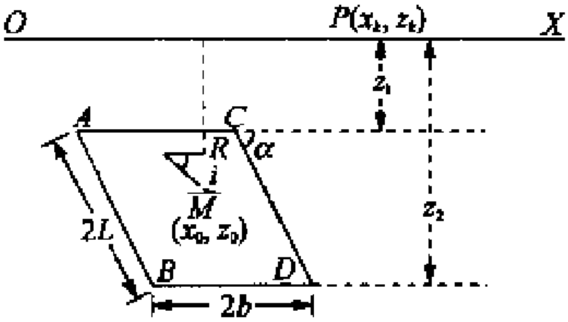


图 1.1　二维板模型

求出 ABCD 四个点分别对 P 点(x_k, z_k)的距离即r_1, r_2, r_3, r_4，根据几何关系有：

$$\begin{aligned}
r_1^2 &= (x_k - x_0 + b + l\cos\alpha)^2 + (z_0 - z_k - l\sin\alpha)^2 \\
r_2^2 &= (x_k - x_0 + b - l\cos\alpha)^2 + (z_0 - z_k + l\sin\alpha)^2 \\
r_3^2 &= (x_k - x_0 - b + l\cos\alpha)^2 + (z_0 - z_k - l\sin\alpha)^2 \\
r_4^2 &= (x_k - x_0 - b - l\cos\alpha)^2 + (z_0 - z_k + l\sin\alpha)^2
\end{aligned}$$

并求出r_1, r_2, r_3, r_4与 X 轴正向的夹角，由 X 轴顺时针起算：

$$\varphi_1 = \pi - \mathrm{tg}^{-1}\frac{z_0 - z_k - l\sin\alpha}{x_k - x_0 + b + l\cos\alpha}$$

$$\varphi_2 = \pi - \mathrm{tg}^{-1}\frac{z_0 - z_k + l\sin\alpha}{x_k - x_0 + b - l\cos\alpha}$$

$$\varphi_3 = \pi - \mathrm{tg}^{-1}\frac{z_0 - z_k - l\sin\alpha}{x_k - x_0 - b + l\cos\alpha}$$

$$\varphi_4 = \pi - \mathrm{tg}^{-1}\frac{z_0 - z_k + l\sin\alpha}{x_k - x_0 - b - l\cos\alpha}$$

根据以上参量可计算出磁异常分量：

$$\Delta X = \frac{M}{2\pi}\sin\ \alpha[\ln\frac{r_2 r_3}{r_1 r_4}\cos(\alpha - i) - \sin(\alpha - i)(\varphi_1 - \varphi_2 - \varphi_3 + \varphi_4)]$$

$$\Delta Z = \frac{M}{2\pi}\sin\alpha[\sin(\alpha - i)\ln\frac{r_2 r_3}{r_1 r_4} + \cos(\alpha - i)(\varphi_1 - \varphi_2 - \varphi_3 + \varphi_4)]$$

即各点的f_k值，

<2>计算一阶导数并形成雅可比矩阵，使用数值微分中的差商法，用x_0和$x_0 + \Delta x$两次来调用计算理论函数f_k的子程序计算可得，

$$\frac{\partial f_1}{\partial x_1} = \lim \frac{f_1(x_1 + \Delta x_1) - f_1(x_1)}{\Delta x_1}$$

程序中取$\Delta x = 10^{-5}$，

<3>计算目标函数ϕ，直接按ϕ的表达式计算可得，即L_2范数

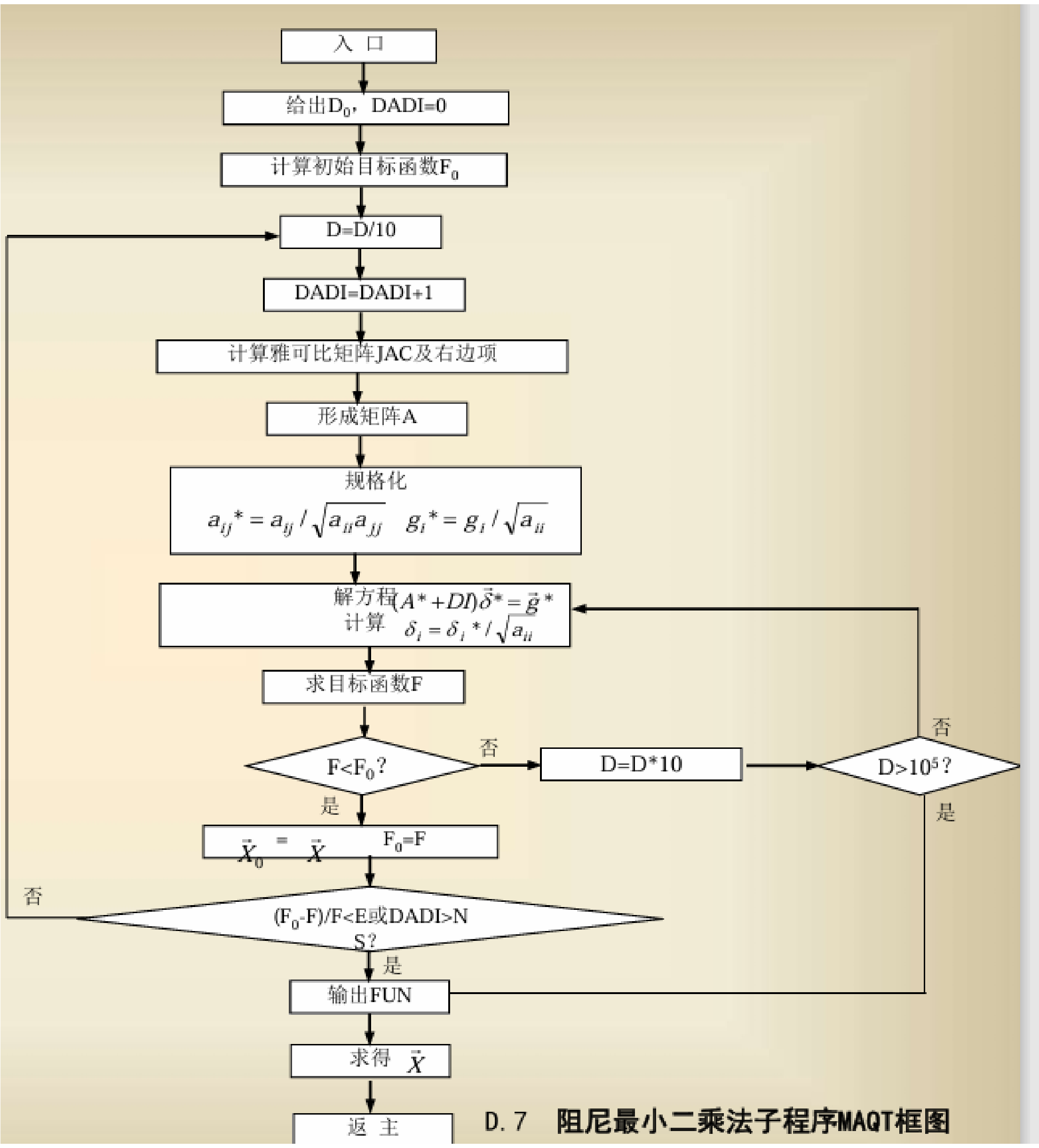
$$loss = \sum_{i=1}^{n} \left(y_{pred} - y_i - b\right)^2$$

<4>阻尼最小二乘法的迭代计算，即解出下列方程：

$$(A^* + \lambda l)\ \sigma^* = g^*$$

其中A^*等是规格化后的结果，为了减小计算误差，改进计算结果

$$a_{ij}^* = a_{ij} / \sqrt{a_{ii} * a_{jj}}$$



D. 7 阻尼最小二乘法子程序MAQT框图

源代码 main.h：

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

const double pi = 3.1415926;
const double pi2rad = pi / 180.0;
const double eps = 1e-15;
const double Delta_b = 1e-5;

const int N = 100; // 参加计算的剖面上的测点数 = PPT 里面的 m1
const int M = 7; // 模型体参数的个数 = PPT 里面的 m2
// PPT 里面的 m3 = M+1
const int NS = 1000; // 允许的最多迭代次数: 待给
const double bo = 1; // 正常场改正值: 待给
// GE: 存放参数值, 前 3 个数为: DX, DC, MC; 后 7 个数为(XO, ZO, 2b, 2l, alpha, i,
M)
// const double ge[10] = {20, 0, 1, 1000, 1000, 400, 800, 90, 90, 2000};
const double ge[10] = {20, 0, 1, 1000, 1000, 400, 800, 90, 90, 2000};

double calculate_atan(double up, double down);

// 给定参量值和测点坐标, 计算出各点 f_k 值
// 由所选的模型体而决定的一套正演计算公式
void calculate_model_forward(double *x, double *fun, double *xk,
                             double *zk, int n);
// 计算一阶导数, 形成雅可比矩阵
void calculate_1D_Jacobi(double *x, double jac[][8], double *fun,
                         double *funo, double *xk, double *zk, int n, int m);
// 计算目标函数
double calculate_target_F(double *g, double *dg, double b, double *fun, int n);

// 阻尼最小二乘法迭代过程并输出迭代结果
void damp_LM(double d0, double e, int ns, double bo, double *x0,
        double jac[][M+1], double *g, double *dg, double *fun,
        double *funo, double *xk, double *zk, int n, int m);
```

源代码 main.c:

```
#include "main.h"

// 计算:atan(b/a) 反正切值, 并且去除奇异值
double calculate_atan(double up, double down)
{
   // up: 分子
```

```
    // down: 分母
    double theta = 0;
    if (fabs(down) > eps)
    {
        if (up / down > eps)
        {
            if (down > eps && up > eps)
            {
                theta = atan(up / down);
            }
            else if (down < eps && up < eps)
            {
                theta = -pi + atan(up / down);
            }
        }
        else if (up / down < eps)
        {
            theta = -pi + atan(up / down);
        }
    }
    else
    {
        if (up < eps)
        {
            theta = -pi / 2.0;
        }
        else
        {
            theta = pi / 2.0;
        }
    }
    return theta;
}

// 给定参量值和测点坐标, 计算出各点 f_k 值
// 由所选的模型体而决定的一套正演计算公式
void calculate_model_forward(double *x, double *fun, double *xk,
                             double *zk, int n)
{
    // x: 模型实验时, 理论模型参量值
    // x: [x0, z0, 2b, 2l, alpha, i, M]
    // xk: 各测点的横坐标
    // zk: 各测点的纵坐标
    // fun: 理论值 f_k, n: 测线长度
```

```
    double piz = 1.0 / (2 * pi);

    // 角度制转为弧度制
    double aa = sin(x[4] * pi2rad); // sin(alpha / (pi / 180.0))
    double bb = cos(x[4] * pi2rad);
    double cc = sin(x[5] * pi2rad);
    double dd = cos(x[5] * pi2rad);

    for (int k = 0; k < n; k++)
    {
        // z0 - zk[k] + l*sin(alpha)
        double a = x[1] - zk[k] + x[3] * aa / 2.0;
        double b = x[1] - zk[k] - x[3] * aa / 2.0;

        // xk[k] - x0 + l*cos(alpha)
        double c = xk[k] - x[0] + x[3] * bb / 2.0;
        double d = xk[k] - x[0] - x[3] * bb / 2.0;

        // -+ b
        double e = c + x[2] / 2.0;
        double f = c - x[2] / 2.0;
        double g = d + x[2] / 2.0;
        double h = d - x[2] / 2.0;

        // r2 * r3 / r1 / r4
        double u = (b * b + f * f) * (a * a + g * g) /
                   (b * b + e * e) / (a * a + h * h);

        double a2 = b * b + c * c - x[2] * x[2] / 4.0;
        double b2 = x[2] * b;
        double v = calculate_atan(b2, a2);

        double a1 = a * a + d * d - x[2] * x[2] / 4.0;
        double b1 = x[2] * a;
        double w = calculate_atan(b1, a1);

        u = log(u);

        fun[k] = piz * x[6] * aa * ((dd * aa - cc * bb) / 2.0 * u + (dd * bb + cc * aa) * (v - 
w));
    }
}

// 计算一阶导数, 形成雅可比矩阵
```

```
void calculate_1D_Jacobi(double *x, double jac[][8], double *fun,
                         double *funo, double *xk, double *zk, int n, int m)
{

    for (int i = 0; i < m; i++)
    {
        x[i] = x[i] + Delta_b;
        calculate_model_forward(x, funo, xk, zk, n); // 计算模型的理论值

        for (int k = 0; k < n; k++)
        {
            jac[k][i] = (funo[k] - fun[k]) / Delta_b;
        }
        x[i] = x[i] - Delta_b;
    }
}

// 计算目标函数
double calculate_target_F(double *g, double *dg, double b, double *fun, int n)
{
    double f = 0;
    for (int k = 0; k < n; k++)
    {
        dg[k] = g[k] - fun[k] - b;
        f += dg[k] * dg[k];
    }
    return f;
}

void damp_LM(double d0, double e, int ns, double bo, double *x0,
             double jac[][M + 1], double *g, double *dg, double *fun,
             double *funo, double *xk, double *zk, int n, int m)
{
    // d0: 精度 esp1
    // e: 精度 esp2
    // ns: 迭代次数的上限
    // bo: 正常场改正值
    // x0:模型的初值/迭代结果
    // jac 雅可比矩阵, 最后一列存放(Z - f)向量
    // g: 实测磁场值
    // dg: 存放(Z - f)向量
    // fun: 迭代中初值/结果
    // funo: 目标函数值
```

```
// 寻找模型体磁场值的最小/最大值
double gmax, gmin;
gmax = gmin = g[0];

for (int i = 0; i < n; i++)
{
    if (g[i] > gmax)
        gmax = g[i];
    if (g[i] < gmin)
        gmin = g[i];
}

// 计算正演模型理论值
calculate_model_forward(x0, fun, xk, zk, n);
// 计算目标函数
double f0 = calculate_target_F(g, dg, bo, fun, n);

// 初值状态
double d = d0;
int index = 0;
double a[M + 1][M + 1];
double x[M];
double r[M];
double q[M];
double b[M];

while (1)
{
    d = d / 10;
    index++;

    // 计算雅可比矩阵 jac 及右边项
    calculate_1D_Jacobi(x0, jac, fun, funo, xk, zk, n, m);

    for (int i = 0; i < n; i++)
    {
        // 右端项放于 M+1 列
        jac[i][m] = dg[i];
    }

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j <= m; j++)
        {
```

```
            // 矩阵清零
            a[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                a[i][j] = a[i][j] + jac[k][i] * jac[k][j];
            }
        }

        r[i] = sqrt(a[i][i]);
        if (r[i] < eps) // eps = 1e-15
        {
            r[i] = r[i] + 1e-7;
        }
    }

    for (int i = 0; i < m; i++)
    {
        for (int j = i; j < m; j++)
        {
            // 规格化求 A*
            // a_{ij} = a_{ij} / sqrt(a_{ii} a_{jj})
            a[i][j] = a[i][j] / r[i] / r[j];
        }
        a[i][m] = a[i][m] / r[i];
    }

    while (1)
    {
        // printf("d = %lf\n", d);

        for (int i = 0; i < m; i++)
        {
            x[i] = d + 1.0;
            for (int k = 0; k < i - 1; k++)
            {
                // 分解: P17
                b[k] = x[k] * a[i][k];
                x[i] = x[i] - b[k] * a[i][k];
            }

            for (int j = i + 1; j <= m; j++)
            {
                a[j][i] = a[i][j];
                for (int k = 0; k <= i - 1; k++)
```

```
        {
            a[j][i] = a[j][i] - b[k] * a[j][k];
        }
        // 回代求解
        a[j][i] = a[j][i] / x[i];
    }
}

// Delta_i = Delta_i / sqrt(a_{ii})
for (int i = m - 1; i >= 0; i--)
{
    for (int j = i + 1; j < m; j++)
    {
        a[m][i] = a[m][i] - a[j][i] * a[m][j];
    }
    q[i] = a[m][i] / r[i];
    x[i] = x0[i] + q[i];
}

// 正演, 但是 x 已经改变
calculate_model_forward(x, fun, xk, zk, n);
double f = calculate_target_F(g, dg, bo, fun, n);

// 记录 迭代次数 index / f / f0 / d

if (f < f0)
{
    double s = (f0 - f) / f;
    f0 = f;

    // X_0 = X
    for (int i = 0; i < m; i++)
    {
        x0[i] = x[i];
    }

    if (index > ns || s < e)
    {
        // 输出结果啦
        printf("save the inverse model parameters");
        FILE *file = fopen("real_model_parameters.txt", "w");

        if (file == NULL)
        {
```

```
            printf("fail to open real_model_parameters.txt");
            return;
        }

        for (int i = 0; i < m; i++)
        {
            fprintf(file, "%lf\n", x[i]);
            // printf("第%d 个反演的模型参数为: %lf\n", i, x[i]);
        }

        fclose(file);

        return; // 跳出函数
    }
    else
    {
        // 回到循环开始的 d = d / 10
        // 这里可以不作处理, 直接 break
        break; // 跳出这层 while
    }
}
else
{
    d = d * 10.0;
    if (d > 1e5 || d <= 1e-10)
    {
        // 输出结果啦
        printf("save the inverse model parameters");
        FILE *file = fopen("real_model_parameters.txt", "w");

        if (file == NULL)
        {
            printf("fail to open real_model_parameters.txt");
            return;
        }

        for (int i = 0; i < m; i++)
        {
            // fprintf(file, "%lf\n", x[i]);
            printf("第%d 个反演的模型参数为: %lf\n", i, x[i]);
        }

        fclose(file);
```

```
                return; // 跳出函数
            }
            else
            {
                // 还要继续在第二个循环迭代
                // pass
            }
        }
    }
  }
}

int main()
{
    // 初始化变量和数组
    // 测点总数: N, 各测点的横坐标 xk, 各测点的纵坐标 zk, 实测/理论模型体的磁场值 G
    // 存放(Z - f)向量的 DG, 迭代过程中初值和迭代结果 fun/funo,
    double xk[N], zk[N], g[N], dg[N], fun[N], funo[N];

    // 模型体参数: M, 模型体参量值 xm, 模型体的初值/迭代结果 xo
    double xm[M], xo[M];

    // 正定方程的增广系数矩阵 A(M+1,M+1), 雅可比矩阵 JAC(N,M+1):最后一列存放(Z -f)向量
    double A[M + 1][M + 1], jac[N][M + 1];

    // 输入(N, M, NS, BO) 已经在 main.h 全局定义

    // ge 的参数值为实验一正演中的参数值
    double dx = ge[0], dc = ge[1], mc = ge[2];
    for (int i = 0; i < 10; i++)
    {
        printf("ge 的第%d 个参数为: %lf\n", i, ge[i]);
    }

    // 得到观测点横坐标的序列 xk:(0, 20, 40,...2000)共 101 个
    for (int k = 0; k < N; k++)
    {
        xk[k] = (double)k * dx;
        // printf("第%d 个观测点的横坐标为: %lf\n", k, xk[k]);
    }

    // 地形改正, 设定为模型一的均为平坦类型
```

```
if (dc > 0)
{
    // 打开 DATAZK 文件
    // for (int k = 0; k < n; k ++)
    // {

    // }
}
else // 实际上只走这里
{
    printf("theory model zk is zeros array\n");
    for (int k = 0; k < N; k++)
    {
        zk[k] = 0;
    }
}

// 实际上就是把 ge 排除前 3 个的量, 把模型体的值给了 xo 数组
for (int k = 0; k < M; k++)
{
    xo[k] = ge[k + 3];
    xm[k] = ge[k + 3];
}

// mc == 0: 正演模拟
// mc != 0: 反演
if (mc == 0)
{
    printf("Input theory model parameters for xm array\n");
    printf("begin theory model experiments\n");

    // 打开理论模型参数文件
    // for (int k = 0; k < M; k ++)
    // {
    //     fscanf(fp8, "%lf", &xm[k]);
    // }

    // xm = [x0, z0, 2b, 2l, alpha, i, M] 已经在上面给出

    // 用实验一的参数正演模拟得到模型体的磁异常值 g
    calculate_model_forward(xm, g, xk, zk, N);

    FILE *pfile = fopen("theory_model_result.txt", "w");
    if (pfile == NULL)
```

```
    {
        printf("fail to open theory_model_result.txt");
        return 1;
    }

    for (int i = 0; i < N; i++)
    {
        // printf("第%d 个测点的磁异常值为: %lf\n", i, g[i]);
        fprintf(pfile, "%lf\n", g[i]);
    }

    fclose(pfile);
}
else
{
    calculate_model_forward(xm, g, xk, zk, N);

    FILE *pfile = fopen("theory_model_result.txt", "w");
    if (pfile == NULL)
    {
        printf("fail to open theory_model_result.txt");
        return 1;
    }

    for (int i = 0; i < N; i++)
    {
        // printf("第%d 个测点的磁异常值为: %lf\n", i, g[i]);
        fprintf(pfile, "%lf\n", g[i]);
    }

    fclose(pfile);

    printf("begin real data process\n");

    FILE *pfile1 = fopen("theory_model_result.txt", "r");
    if (pfile == NULL)
    {
        printf("fail to open theory_model_result.txt");
        return 1;
    }

    int index = 0;
    while (fscanf(pfile1, "%lf", &g[index]) != EOF)
    {
```

```
        index++;
    }
    for (int i = 0; i < N; i++)
    {
        // printf("第%d个测点的磁异常值为: %lf\n", i, g[i]);
    }
    fclose(pfile1);
    // 精度定义
    double eps1 = 0.1, eps2 = 0.001;
    damp_LM(eps1, eps2, NS, bo, xo, jac, g, dg,
            fun, funo, xk, zk, N, M);
    FILE *file2 = fopen("real_model_result.txt", "w");
    if (file2 == NULL)
    {
        printf("fail to open real_model_result.txt");
        return 1;
    }
    for (int i = 0; i < N; i++)
    {
        // printf("%lf, %lf\n", fun[i], g[i]);
        fprintf(file2, "%lf %lf\n", fun[i], g[i]);
    }
    fclose(file2);
  }
}
```

绘图源代码 plot.py:

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('default')


# Read data from the text file
data = np.loadtxt('real_model_result.txt')

# Separate the data into two arrays for plotting
x = np.linspace(0, 2000, 100)
y1 = data[:, 0]
y2 = data[:, 1]

# Plot the curves
plt.plot(x, y1, label='real', linestyle='-', color='b')  # Blue solid line
plt.plot(x, y2, label='theory', linestyle=':', color='r') # Red dashed line
```

```
# Add legend
plt.legend()

# Add labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('real vs theory, bo:1, iterations:1000')

# Show the plot
plt.show()
```

结果：