**基础题：**

证明式（15）中，取 y=u4 是该问题的最优解，

- **寻找最小二乘解：**

$$\min_{\mathbf{y}} \|\mathbf{D}\mathbf{y}\|_2^2, \quad s.t. \|\mathbf{y}\| = 1$$

提示设置 y’ =$u_4$+v,其中 v 正交于 $u_4$，证明

$$y'^T D^T D y' = y'^T D^T D y$$

该式方法基于奇异值构造矩阵零空间。

**解：**

首先了解 SVD 分解：

$$D = U \sum V^T \qquad (1)$$

其中：$\sum$ 为对角阵，$diag(\sigma_1,\sigma_2,\sigma_3,\sigma_4)$ 而且 $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \sigma_4$

U 为（$u_1$, $u_2$, $u_3$, $u_4$），是单位正交集,而且 V 为（$v_1$, v, $v_3$, $v_4$），是单位正交集。

假设 A 是一个 N*M 的矩阵，那么得到的 U 是一个 N ∗ 4 的方阵（里面的向量是正交的，U 里面的向量称为左奇异向量），Σ 是一个 4 ∗ 4 的矩阵（除了对角线的元素都是 0，对角线上的元素称为奇异值），V’（V 的转置)是一个 N ∗ 4 的矩阵，里面的向量也是正交的，V 里面的向量称为右奇异向量）

所以:

$$D^T = V \sum U^T \qquad (2)$$

由公式 1 和 2 得：

$$D^T D = V \sum^T U^T U \sum V^T$$

$$= V \sum^T \sum V^T$$

$$= \left[v_1, v_2, v_3, v_4\right] diag\left(\sigma_1, \sigma_2, \sigma_3, \ \sigma_4\right)^T \ diag\left(\sigma_1, \sigma_2, \sigma_3, \ \sigma_4\right) \begin{bmatrix} {v_1}^T \\ {v_2}^T \\ {v_3}^T \\ {v_4}^T \end{bmatrix}$$

$$= \sum_{i=1}^{4} \sigma_i^2 v_i v_i^T$$

$$= \sum_{i=1}^{4} \sigma_i^2 \qquad (3)$$

所以可以的到时

因为 V 为（$v_1$, $v_2$, …, $v_n$），是正交阵，所以元素组成空间的一组基函数

令 $$y = \sum_{i=1}^{4} k_i v_i = (k_i, k_2, k_3, k_4) \qquad (4)$$

而且

$$\sum_{i=1}^{4} k_i^{\ 2} = 1 \qquad （5）$$

结合 4 得到如下：

$$\begin{aligned}
&\|Dy\|^2 \\
&= y^T D^T D y \\
&= (k_1, k_2, k_3, k_4)^T \left( \sum_{i=0}^{4} \sigma_i^2 \right) (k_1, k_2, k_3, k_4) \\
&= \sigma_1^2 k_1^{\ 2} + \sigma_2^2 k_2^{\ 2} + \sigma_3^2 k_3^{\ 2} + \sigma_4^2 k_4^{\ 2}
\end{aligned} \qquad （6）$$

**其中：** $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \sigma_4$

由公式 5 和 6 得出：

目标函数 $\min \|Dy\|^2$

容易最小化目标函数可得到：

$k_4=1$；$k_1=0$；$k_2=0$；$k_3=0$；

**结论：**

a. 把上面 $k_i$ 值带入公式（4）得到 $y=v_4$(推到结果和 ppt 一致，推到过程符号问题，其实 $y=u_4$)

b. 把优化变量带入目标函数，等于 $\sigma_4^{\ 2}$

提升题：

完成代码部分：

解：

a. 首次要读懂代码：

1. 生成姿态数据，n=10
2. 然后随机生成随机观测量（x, y, z）
3. 代码补全，实现三角化原理
4. 打印估计值和实际值，对算法验证

b. 补全代码

```cpp
    /// TODO::homework; 请完成三角化估计深度的代码
    // 遍历所有的观测数据，并三角化
    Eigen::Vector3d P_est; // 结果保存到这个变量
    P_est.setZero();
    /* your code begin */

    //主要思想：相机多帧观测对应空间在一点
    Eigen::Matrix4d D;
    Eigen::MatrixXd P(2 * (end_frame_id - start_frame_id), 4);

    for (int i = 3, j = 0; i < poseNums; i++)
    {
        Eigen::Isometry3d Tcw = Eigen::Isometry3d::Identity();
        Tcw.rotate(camera_pose[i].Rwc.transpose());
        Tcw.pretranslate(-camera_pose[i].Rwc.transpose() * camera_pose[i].twc);

        P.block(j, 0, 1, 4) = camera_pose[i].uv.x() * Tcw.matrix().row(2) - Tcw.matrix().row(0);
        P.block(j + 1, 0, 1, 4) = camera_pose[i].uv.y() * Tcw.matrix().row(2) - Tcw.matrix().row(
        j += 2;
    }

    D = P.transpose() * P;
    Eigen::JacobiSVD<Eigen::MatrixXd> svd(D, Eigen::ComputeThinU | Eigen::ComputeThinV);

    Eigen::MatrixXd U = svd.matrixU();
    Eigen::MatrixXd V = svd.matrixV();

    P_est << (U.rightCols(1) * U.rightCols(1).bottomRows(1).inverse()).topRows(3);

    /* your code end */
```

c. 运行结果：

```
vslam@vslam: ~/VIO_Tutorial/第7讲：视觉前端/course6_hw/build
vslam@vslam:~/VIO_Tutorial/第7讲：视觉前端/course6_hw/build$ ./estimate_depth
1
ground truth:
  -2.9477 -0.330799   8.43792
your result:
  -2.9477 -0.330799   8.43792
vslam@vslam:~/VIO_Tutorial/第7讲：视觉前端/course6_hw/build$
```