智能传感与信号处理 Assignment2

Task1

> 推导高斯牛顿法中的 2.2 并展示推导过程，然后在代码的 `GN_Solver.m` 中完成 `dk` 的计算

推导过程

1. 线性化：

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + J_k(\mathbf{x} - \mathbf{x}_k)$$

2. 将线性化代入非线性最小二乘中，

   残差 $\mathbf{r}_k = f(\mathbf{x}_k) - \mathbf{z}$，问题变为：

$$\min_{\mathbf{d}_k} (J_k\mathbf{d}_k + \mathbf{r}_k)^T W^{-1} (J_k\mathbf{d}_k + \mathbf{r}_k)$$

3. 求解该线性最小二乘问题，展开目标函数：

$$(J_k\mathbf{d}_k + \mathbf{r}_k)^T W^{-1} (J_k\mathbf{d}_k + \mathbf{r}_k) = \mathbf{d}_k^T J_k^T \mathbf{W}^{-1} J_k \mathbf{d}_k + 2d_k^T J_k^T \mathbf{W}^{-1} \mathbf{r}_k + \mathbf{r}_k^T \mathbf{W}^{-1} \mathbf{r}_k$$

4. 对 $\mathbf{d}_k$ 求偏导并令导数为零：

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{d}_k} \left(\mathbf{d}_k^T J_k^T \mathbf{W}^{-1} J_k \mathbf{d}_k + 2d_k^T J_k^T \mathbf{W}^{-1} \mathbf{r}_k + \mathbf{r}_k^T \mathbf{W}^{-1} \mathbf{r}_k \right) \\
&= 2J_k^T \mathbf{W}^{-1} J_k \mathbf{d}_k + 2J_k^T \mathbf{W}^{-1} \mathbf{r}_k = 0
\end{aligned}$$

5. 求解得到：

$$\mathbf{d}_k = -(J_k^T \mathbf{W}^{-1} J_k)^{-1} J_k^T \mathbf{W}^{-1} \mathbf{r}_k$$

代码实现

在 `GN_Solver.m` 中实现这个解：

```
dk = -( J' / g.W * J) \ J' / g.W *r;
```

关于使用 `\`，而不是 `inv()`，参考 [MATLAB 官方文档]

> 反斜杠计算方法速度更快，而且残差减少了几个数量级。

Task2

> 推导雅可比矩阵中重要子矩阵 $\mathbf{E}_{ij}$，$\mathbf{G}_{ij}$ 并展示推导过程，然后在代码的 `compute_J.m` 中完成这些矩阵和残差的计算

推导过程

$$T_{i,j} = \frac{\|\mathbf{x}_i - \mathbf{s}_j\| - \|\mathbf{x}_1 - \mathbf{s}_j\|}{c} + \tau_i + \delta_i \Delta t_j$$

计算过程中用到的公式：

$$\|\mathbf{a}\| = (\mathbf{a}^T\mathbf{a})^{1/2}$$

$$\frac{\partial\|\mathbf{a}\|}{\partial\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}$$

对于 $\mathbf{E}_{ij}$

$$\begin{aligned}\mathbf{E}_{ij} &= \frac{\partial T_{i,j}}{\partial[x_i, \tau_{i,1}, \delta_{i,1}]^T} \\ &= \begin{bmatrix} \frac{\partial T_{i,j}}{\partial x_i} & \frac{\partial T_{i,j}}{\partial \tau_{i,1}} & \frac{\partial T_{i,j}}{\partial \delta_{i,1}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{c}\cdot\frac{\mathbf{x}_i-\mathbf{s}_j}{\|\mathbf{x}_i-\mathbf{s}_j\|} & 1 & \Delta t_j \end{bmatrix}\end{aligned}$$

对于 $\mathbf{G}_{ij}$：

$$\mathbf{G}_{ij} = \frac{\partial T_{i,j}}{\partial\mathbf{s}_j} = \frac{(\mathbf{s}_j-\mathbf{x}_i)}{c\|\mathbf{x}_i-\mathbf{s}_j\|} - \frac{(\mathbf{s}_j-\mathbf{x}_1)}{c\|\mathbf{x}_1-\mathbf{s}_j\|}$$

代码实现

`compute_J.m`

```
% Eij
Eij = [dx / norm(dx) / g.cc, 1, sum(g.dt(1:j))];
% Gij
Gij = (-dx / norm(dx) - s_loc / norm(s_loc)) / g.cc;
% r_tdoaij
r_tdoaij= (norm(dx)-norm(s_loc)) / g.cc + off + sum(g.dt(1:j)) * dri - g.tdoa(i-1,j)
% r_odoj
r_odoj = s_next - s_now - g.S(:,j+1) + g.S(:,j);
```

Task3

> 基于任务 1 和任务 2 完成的代码，在 `main.m` 实现同时麦克风阵列标定和声音事件位置估计

MATLAB 代码见附件。

实验结果分析

50 轮的平均麦克风位置估计误差

init_sigma	mean error (m)
0.5	0.050870
1.0	0.498612
2.0	1.930112

- init_sigma 为 0.5 时，平均麦克风位置估计误差较小，且结果稳定
- init_sigma 为 1.0 时，平均麦克风位置估计误差较大，且结果不稳定（运行500轮依然不稳定）
- init_sigma 为 2.0 时，平均麦克风位置估计误差很大，且结果不稳定（运行500轮依然不稳定）

问题解释

> 为什么传统非线性优化结果依赖初值?

1. 多局部极值
   非线性优化问题可能存在多个局部极值点，优化过程可能收敛于局部极值，无法得到
   全局最优的结果。
2. 算法性质
   例如牛顿-欧拉方法，依赖于当前点的梯度来决定搜索方向和步长，若初始点离全局最优较远，可能出现 **收敛于局部最优** 或 **在限定时间内无法完成计算** 的问题。