手写VIO-第三章作业分享

主讲人 啦啦啦

题目

作业

1 样例代码给出了使用 LM 算法来估计曲线 $y = \exp(ax^2 + bx + c)$ 参数 a, b, c 的完整过程。

1. 请绘制样例代码中 LM 阻尼因子 μ 随着迭代变化的曲线图
2. 将曲线函数改成 $y = ax^2 + bx + c$，请修改样例代码中残差计算，雅克比计算等函数，完成曲线参数估计。
3. 实现其他更优秀的阻尼因子策略，并给出实验对比（选做，评优秀），策略可参考论文[a] 4.1.1 节。

2 公式推导，根据课程知识，完成 $\mathbf{F}, \mathbf{G}$ 中如下两项的推导过程：

$$\mathbf{f}_{15} = \frac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \delta \mathbf{b}_k^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)]_\times \delta t^2)(-\delta t)$$

$$\mathbf{g}_{12} = \frac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \mathbf{n}_k^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)]_\times \delta t^2)(\frac{1}{2}\delta t)$$

3 证明式(9)。

[a] Henri Gavin. "The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems". In: *Department of Civil and Environmental Engineering, Duke University* (2011), pp. 1–15.

第一题

- **T1.1**：找到problem.cpp中的Problem::Solve()函数， 相应代码即在该函数内。可以输出两种Lambda (也即

这里展示的使误差下降的Lambad；

另一种放到**while (!oneStepSuccess)**中，表示所有的Lambda。

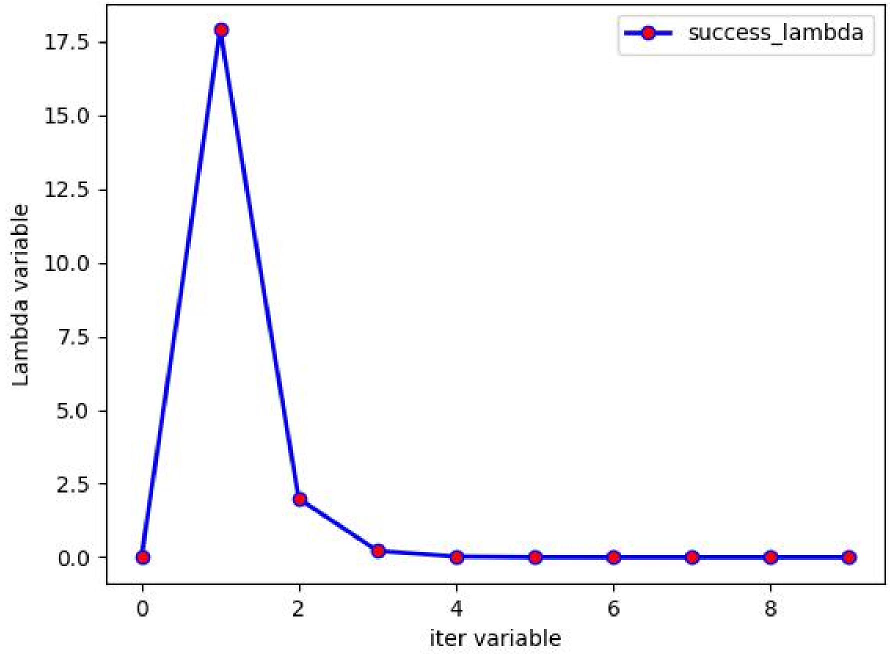
导出两种Lambda，可以进行分析对比。

```
//---------------------------------------------
//存csv文件
ostringstream csvfilename; // 文件名
std::ofstream Lambda_data;
csvfilename << "Lambda.csv"; // 名称+格式
Lambda_data.open(csvfilename.str(), ios_base::out); // 输出到对应文件格式中
Lambda_data << "iter"  << "," << "chi"  << ","<< "Lambda"<<endl;

while (!stop && (iter < iterations)) {
    std::cout << "iter: " << iter << " , chi= " << currentChi_ << " , Lambda= " << currentLambda_
              << std::endl;
    //---------------------------------------------
    Lambda_data <<iter << "," << currentChi_ << " , " << currentLambda_<<endl;
    //---------------------------------------------
```

第一题

●绘制曲线图，使用python程序

```python
# #!/usr/bin/env python
# # _*_ coding:utf-8 _*_
import matplotlib.pyplot as plt
import csv

file = "../bin/Method1_Lambda.csv"

iter,Lambda = [],[]

# 用reder读取csv文件
with open(file,'r') as csvFile:
  reader = csv.reader(csvFile)
  for row,col in enumerate(reader):
    if  row != 0:
      numbers_float = map(float, col) #转化为浮点数
      numbers_float = list(numbers_float)
      iter.append(numbers_float[0:1])
      Lambda.append( numbers_float[2:3])

plt.plot(iter, Lambda, ls='-', lw=2, label='success_lambda', color='blue', markerfacecolor='red',marker='o')
plt.legend()# 显示图例
plt.xlabel('iter variable')
plt.ylabel('Lambda variable')
plt.show()
```



第一题

T1.2： y函数改变 -> 对应改变残差函数ComputeResidual() -> 改变雅克比Jacobian于ComputeJacobians()函数中。

$$y = ax_i^2 + bx_i + c$$

$$err_i = ax_i^2 + bx_i + c$$

$$Jacobians_i = [x_i^2, x_i, 1]$$

```
//**********************NO.1**********************
  virtual void ComputeResidual() override
  {
    Vec3 abc = verticies_[0]->Parameters(); // 估计的参数
    // residual_(0) = std::exp( abc(0)*x_*x_ + abc(1)*x_ + abc(2) ) - y_; // 构建残差
    residual_(0) = abc(0) * x_ * x_ + abc(1) * x_ + abc(2) - y_; // 构建残差
  }
//**********************NO.2**********************
  // 计算残差对变量的雅克比
  virtual void ComputeJacobians() override
  {
    // Vec3 abc = verticies_[0]->Parameters();
    // double exp_y = std::exp( abc(0)*x_*x_ + abc(1)*x_ + abc(2) );

    Eigen::Matrix<double, 1, 3> jaco_abc; // 误差为1维，状态量 3 个，所以是 1x3 的雅克比矩阵
    // jaco_abc << x_ * x_ * exp_y, x_ * exp_y , 1 * exp_y;
    jaco_abc << x_ * x_ , x_ , 1 ;
    jacobians_[0] = jaco_abc;
  }
//**********************NO.3**********************
  // double y = std::exp( a*x*x + b*x + c ) + n; //加入噪声数据
  double y = (a * x * x + b * x + c);//TODO: new-作业
```

第一题-分析

1. $\lambda_0 = \lambda_o$; λ_o is user-specified [5].
   use eq'n (13) for $\mathbf{h}_{\text{lm}}$ and eq'n (16) for ρ
   if $\rho_i(\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{h}$; $\lambda_{i+1} = \max[\lambda_i / L_\downarrow, 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \min[\lambda_i L_\uparrow, 10^7]$;

2. $\lambda_0 = \lambda_o \max\left[\text{diag}[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J}]\right]$; λ_o is user-specified.
   use eq'n (12) for $\mathbf{h}_{\text{lm}}$ and eq'n (15) for ρ
   $\alpha = \left(\left(\mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))\right)^\mathsf{T}\mathbf{h}\right) / \left(\left(\chi^2(\mathbf{p}+\mathbf{h}) - \chi^2(\mathbf{p})\right)/2 + 2\left(\mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))\right)^\mathsf{T}\mathbf{h}\right)$;
   if $\rho_i(\alpha\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p} + \alpha\mathbf{h}$; $\lambda_{i+1} = \max[\lambda_i/(1+\alpha), 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \lambda_i + |\chi^2(\mathbf{p}+\alpha\mathbf{h}) - \chi^2(\mathbf{p})|/(2\alpha)$;

3. $\lambda_0 = \lambda_o \max\left[\text{diag}[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J}]\right]$; λ_o is user-specified [6].
   use eq'n (12) for $\mathbf{h}_{\text{lm}}$ and eq'n (15) for ρ
   if $\rho_i(\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{h}$; $\lambda_{i+1} = \lambda_i \max\left[1/3, 1 - (2\rho_i - 1)^3\right]$; $\nu_i = 2$;
   otherwise: $\lambda_{i+1} = \lambda_i \nu_i$; $\quad \nu_{i+1} = 2\nu_i$;

第一题-分析

深蓝学院 shenlanxueyuan.com

$$\left[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J} + \boxed{\lambda\mathbf{I}}\right]\mathbf{h}_{\mathsf{lm}} = \mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}) \;, \qquad \boxed{(12)} \longrightarrow \text{策略2\&3}$$

$$\left[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J} + \boxed{\lambda\,\mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J})}\right]\mathbf{h}_{\mathsf{lm}} = \mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}) \;. \qquad \boxed{(13)}$$

策略1

$$\begin{aligned}
\rho_i(\mathbf{h}_{\mathsf{lm}}) &= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{(\mathbf{y}-\hat{\mathbf{y}})^\mathsf{T}(\mathbf{y}-\hat{\mathbf{y}}) - (\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathsf{lm}})^\mathsf{T}(\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathsf{lm}})} && (14)\\
&= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{\mathbf{h}_{\mathsf{lm}}^\mathsf{T}\left(\lambda_i\mathbf{h}_{\mathsf{lm}} + \mathbf{J}^\mathsf{T}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} && \boxed{\text{if using eq'n (12) for } \mathbf{h}_{\mathsf{lm}} \quad (15)}\\
&= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{\mathbf{h}_{\mathsf{lm}}^\mathsf{T}\left(\lambda_i\mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J})\mathbf{h}_{\mathsf{lm}} + \mathbf{J}^\mathsf{T}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} && \boxed{\text{if using eq'n (13) for } \mathbf{h}_{\mathsf{lm}} \quad (16)}
\end{aligned}$$

第一题-分析

在代码中明确需要修改的部分：
1.根据论文中公式12 13，定位代码 **AddLambdatoHessianLM()**、**RemoveLambdaHessianLM().**
2.根据Lambda更新策略，定位 **IsGoodStepInLM()**

接下来，我们需要在相应位置加入代码。

```
while (!oneStepSuccess) // 不断尝试 Lambda, 直到成功迭代一步
    {

        //
        AddLambdatoHessianLM();
        // 第四步，解线性方程 H X = B || 直接求解 delta_x_
        SolveLinearSystem();
        //
        RemoveLambdaHessianLM();

        // // 优化退出条件1： delta_x_ 很小则退出
        // // TODO:: 退出条件还是有问题, 好多次误差都没变化了，还在迭代计算，应该搞一个误差不变了就中止
        if (delta_x_.squaredNorm() <= 1e-6 || false_cnt > 10) {
            stop = true;
            printf("failure:1\n ");
            break;
        }

        // 【new】 ------------------------------------------
        Lambda_data2 <<iter <<","<< currentChi_<<","<<currentLambda_ << endl;
        // 【end】 ------------------------------------------

        // 更新状态量 X = X+ delta_x
        UpdateStates();
        // 判断当前步是否可行以及 LM 的 lambda 怎么更新 || 判断当前误差是否在下降
        oneStepSuccess = IsGoodStepInLM();// TODO: 阻尼因子更新
        // 后续处理，
        if (oneStepSuccess) {
```

$$\left[J^T W J + \lambda\, \mathrm{diag}(J^T W J)\right] h_{lm} = J^T W (y - \hat{y})$$

与上面相反

第一题-代码更改

T1.3：策略1

```cpp
void Problem::AddLambdatoHessianLM() {
#if Lambda_update_math == 1
  ulong size = Hessian_.cols();
  assert(Hessian_.rows() == Hessian_.cols() && "Hessian is not square");
  for (ulong i = 0; i < size; ++i) {
    Hessian_(i, i) *= (1. + currentLambda_); // Hseeian = lambda
  }
#elif Lambda_update_math == 2 || Lambda_update_math == 3
  ulong size = Hessian_.cols();
  assert(Hessian_.rows() == Hessian_.cols() && "Hessian is not square");
  for (ulong i = 0; i < size; ++i) {
    Hessian_(i, i) += currentLambda_; // Hseeian = lambda
  }
#endif
}
```

策略1

策略2&3

$$\left[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J} + \boxed{\lambda\mathbf{I}}\right]\mathbf{h}_{\mathrm{lm}} = \mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y}-\hat{\mathbf{y}}) \,. \qquad \boxed{(12)}$$

$$\left[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J} + \boxed{\lambda\,\mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J})}\right]\mathbf{h}_{\mathrm{lm}} = \mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y}-\hat{\mathbf{y}}) \,. \qquad \boxed{(13)}$$

RemoveLambdaHessianLM()部分与AddLambdatoHessianLM()相反。(乘->除，加->减)

根据公式，我们只需要更新对角线上的元素。所以：

策略2&3： H += Lambda

策略1 ： H *= (1 + Lambda)

结合***SolveLinearSystem()***部分，h_{lm} (delta_x_)更新完成。

第一题-代码更改

T1.3：策略1

$$\rho_i(\mathbf{h}_{\mathrm{lm}}) = \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathrm{lm}})}{(\mathbf{y}-\hat{\mathbf{y}})^{\mathsf{T}}(\mathbf{y}-\hat{\mathbf{y}}) - (\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathrm{lm}})^{\mathsf{T}}(\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathrm{lm}})} \quad (14)$$

$$= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathrm{lm}})}{\mathbf{h}_{\mathrm{lm}}^{\mathsf{T}}\left(\lambda_i \mathbf{h}_{\mathrm{lm}} + \mathbf{J}^{\mathsf{T}}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} \quad \boxed{\text{if using eq'n (12) for } \mathbf{h}_{\mathrm{lm}} \quad (15)}$$

$$= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathrm{lm}})}{\mathbf{h}_{\mathrm{lm}}^{\mathsf{T}}\left(\lambda_i \mathrm{diag}(\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J})\mathbf{h}_{\mathrm{lm}} + \mathbf{J}^{\mathsf{T}}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} \quad \boxed{\text{if using eq'n (13) for } \mathbf{h}_{\mathrm{lm}} \quad (16)}$$

[15-分母部分] scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);

[16-分母部分]

scale2 = delta_x_.transpose() * (currentLambda_ * Hessian_.diagonal() * delta_x_ + b_);

至此，ρ(rho) 计算所需的条件已经具备。

第一题-代码更改

T1.3：策略1-*IsGoodStepInLM*()

```
#if Lambda_update_math == 1

   double scale2 = 0;
   scale2 = delta_x_.transpose() * (currentLambda_ * Hessian_.diagonal() * delta_x_ +
b_);
   scale2 += 1e-3;    // make sure it's non-zero
   double rho = (currentChi_ - tempChi) / scale2;

   if (rho > 0 && isfinite(tempChi))   // last step was good, 误差在下降
   {
      currentLambda_ = std::max(currentLambda_/9.0, 1e-7);

      currentChi_ = tempChi; //残差更新
      return true;
   } else {
      currentLambda_ = std::min(currentLambda_ * 11.0, 1e+7); // 【new】
      return false;
   }


```

综上，策略1的剩余代码如左图所示。

第一题

深蓝学院 shenlanxueyuan.com

策略2程序的流程。

1. 计算ρ 更新需要的 α和δx

$$\alpha = \frac{((J^T W(y - \hat{y}(p)))^T h)}{(\frac{(\chi^2(p+h) - \chi^2(p))}{2} + 2(J^T W(y - y^\wedge(p)))^T h)} \tag{2}$$

2. 计算 $\delta x = \delta x * \alpha$ ， 并更新状态。

   δx 改变 $x = x + \delta x$ 中的 x 也发生改变， 所以残差 tempChi 即$\chi^2(p + h_{lm})$也要更新。

3. 计算 scale (公式15的分母)，之后计算 ρ 即可

$$\rho = \frac{\chi^2(p) - \chi^2(p + h_{lm}))}{h_{lm}^T(\lambda_i h_{lm} + J^T W(y - \hat{y}(p))} \tag{3}$$

4. 按照策略2的描述，进行 λ 的相应更新即可。

第一题-策略2

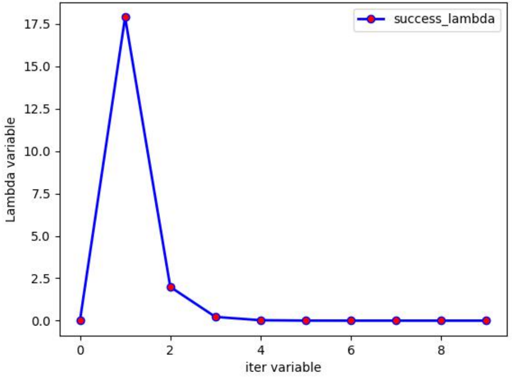
```
#elif Lambda_update_math == 2
    RollbackStates(); // delta_x的上一次取值|| 【更新-说明上次取得不太准呗，用更准确的】

    double JtwFH = b_.transpose() * delta_x_;
    double alpha = JtwFH / ((tempChi - currentLambda_) / 2. + 2 * JtwFH);
    alpha = std::max(0.1, alpha); //  a大于0.1 会无法迭代【测试】

    delta_x_ *= alpha; //重新计算 delta_x_ ，求出 alpha * delta_x_
    UpdateStates(); // 再更新状态

    // 【问题在这-更新】|| delta_x更新了，残差也要相应的更新
    tempChi = 0.0;
    for (auto edge: edges_) {
        edge.second->ComputeResidual();
        tempChi += edge.second->Chi2();
    }

    double scale = 0;
    scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);
    scale += 1e-3; // make sure it's non-zero :)
    double rho = (currentChi_ - tempChi) / scale;

    if (rho > 0 && isfinite(tempChi)) // last step was good, 误差在下降
    {
        currentLambda_ = std::max(currentLambda_ / (1 + alpha), 1e-7);
        currentChi_ = tempChi; //残差更新
        return true;
    }
    else
    {
        currentLambda_ += abs(currentChi_ - tempChi) / (2. * alpha);
        return false;
    }
```

主要流程[红色] 细节之类[绿色]

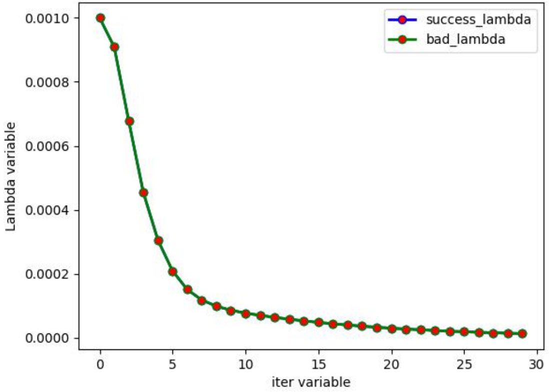
第一题

下面曲线基于exp(ax^2+bx+c)， 如果过于简单迭代次数区别不一定明显。

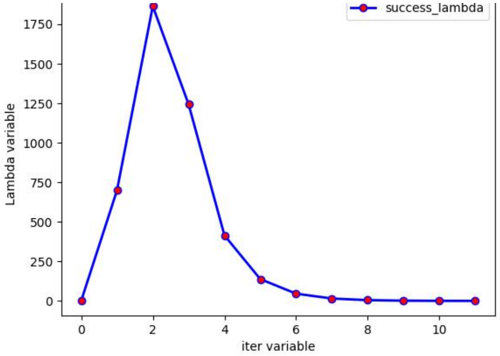
策略1



策略2



策略3



策略1： 成功迭代次数 10/30

策略2： 成功迭代次数 30/30

策略3： 成功迭代次数 12/30

第二题

- T2:求解f15

已知:

$$\alpha_{b_k b_{k-1}} = \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \tag{4}$$

$$a = \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a)) \tag{5}$$

$$= \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}w\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)) \tag{6}$$

$$w = \frac{1}{2}((w^{b_k} - b_k^g) + (w^{b_{k+1}} - b_k^g)) \tag{7}$$

$$= \frac{1}{2}(w^{b_k} + w^{b_{k+1}}) - b_k^g \tag{8}$$

$$f_{15} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta b_k^g} = \frac{\partial \frac{1}{4} q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(w - b_k^g)\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial b_k^g} \tag{9}$$

$$= \frac{\partial \frac{1}{4} R_{b_i b_k} exp([(w - b_k^g)\delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial b_k^g} \tag{10}$$

$$= \frac{\partial \frac{1}{4} R_{b_i b_k} exp([w\delta t]_\times) exp([(-J_r(w\delta) b_k^g \delta t)]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial b_k^g} \tag{11}$$

$$= \frac{\partial \frac{1}{4} R_{b_i b_{k+1}}(I - [J_r(w\delta) b_k^g \delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial b_k^g} \tag{12}$$

$$= \frac{\partial \frac{1}{4} R_{b_i b_{k+1}}[-J_r(w\delta) b_k^g \delta t]_\times (a^{b_{k+1}} - b_k^a)\delta t^2}{\partial b_k^g} \tag{13}$$

$$= -\frac{\partial \frac{1}{4} R_{b_i b_{k+1}}[(a^{b_{k+1}} - b_k^a)\delta t^2]_\times (-J_r(w\delta) b_k^g \delta t)}{\partial b_k^g} \tag{14}$$

$$= -\frac{1}{4} R_{b_i b_{k+1}}[(a^{b_{k+1}} - b_k^a)\delta t^2]_\times (-J_r(w\delta)\delta t) \tag{15}$$

$$= -\frac{1}{4} R_{b_i b_{k+1}}[(a^{b_{k+1}} - b_k^a)\delta t^2]_\times (-\delta t) \tag{16}$$

$$= \frac{1}{4} R_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a)^\wedge \delta t^3 \tag{17}$$

第二题

- T2:求解g12

已知：

$$\alpha_{b_k b_{k-1}} = \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \tag{18}$$

$$a = \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a)) \tag{19}$$

$$= \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}w\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)) \tag{20}$$

$$w = \frac{1}{2}((w^{b_k} + n_k^g - b_k^g) + (w^{b_{k+1}} + n_{k+1}^g - b_k^g)) \tag{21}$$

$$= \frac{1}{2}(w^{b_k} + w^{b_{k+1}}) - b_k^g + \frac{1}{2}n_k^g + \frac{1}{2}n_{k+1}^g \tag{22}$$

$$g_{12} = \frac{\partial \frac{1}{4} q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(w + \frac{1}{2}n_k^g)\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^g)\delta t^2}{\partial n_k^g} \tag{23}$$

$$= \frac{\partial \frac{1}{4} R_{b_i b_k} exp([w\delta t]_\times) exp([J_r(w\delta t)\frac{1}{2}n_k^g\delta t]_\times)(a^{b_{k+1}} - b_k^g)\delta t^2}{\partial n_k^g} \tag{24}$$

$$= -\frac{\partial \frac{1}{4} R_{b_i b_{k+1}}[(a^{b_{k+1}} - b_k^g)\delta t^2]_\times J_r(w\delta t)\frac{1}{2}n_k^g\delta t}{\partial n_k^g} \tag{25}$$

$$= -\frac{1}{4}R_{b_i b_{k+1}}[(a^{b_{k+1}} - b_k^g)\delta t^2]_\times(\frac{1}{2}\delta t) \tag{26}$$

$$= -\frac{1}{8}R_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^g)^\wedge \delta t^3 \tag{27}$$

第三题

深蓝学院 shenlanxueyuan.com

- 证明：

$$(J^TJ+\mu I)\Delta x_{lm} = (V\Lambda V^T+\mu IVV^T)\Delta x_{lm} = V(\Lambda+\mu I)V^T\Delta x_{lm} \quad (28)$$

$$= -J^Tf = -F'^T \quad (29)$$

所以：$\Delta x_{lm} = -\frac{V^TF'^T}{\Lambda+\mu I}V = -\sum_{j=1}^{n}\frac{v_j^TF'^T}{\lambda_j+\mu}v_j \quad (30)$

所以：$\Delta x_{lm} = -V(\Lambda+\mu I)^{-1}V^TF'^T \quad (31)$

$$= -[v_1, v_2 \quad \cdots \quad v_n]\begin{bmatrix} \frac{1}{\lambda_1+\mu} & & & \\ & \frac{1}{\lambda_2+\mu} & & \\ & & \ddots & \\ & & & \frac{1}{\lambda_n+\mu}\end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T\end{bmatrix}F'^T \quad (32)$$

$$= -[v_1, v_2 \quad \cdots \quad v_n]\begin{bmatrix} \frac{v_1^TF'^T}{\lambda_1+\mu} \\ \frac{v_2^TF'^T}{\lambda_2+\mu} \\ \vdots \\ \frac{v_n^TF'^T}{\lambda_n+\mu}\end{bmatrix} \quad (33)$$

$$= -(\frac{v_1^TF'^T}{\lambda_1+\mu}v_1 + \frac{v_2^TF'^T}{\lambda_2+\mu}v_2 + \cdots + \frac{v_n^TF'^T}{\lambda_n+\mu}v_n) = -\sum_{j=1}^{n}\frac{v_j^TF'^T}{\lambda_j+\mu}v_j \quad (34)$$

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening