手写VIO-第三讲作业分享

主讲人 allenthree

作业内容

深蓝学院 shenlanxueyuan.com

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.

1.1 绘制阻尼因子mu变化曲线

深蓝学院
shenlanxueyuan.com

μ 以 currentLambda_这个成员变量出现在problem.cc文件的 bool Problem::Solve(int iterations) 中。 这里存储的是成功的迭代步之后的lambda，有时候迭代失败，就根据失败的方法更新lambda

记录所以的lambda数据，在problem.cc 大概110行：**oneStepSuccess = IsGoodStepInLM();** 之后将CurretLambda_存入文件当中

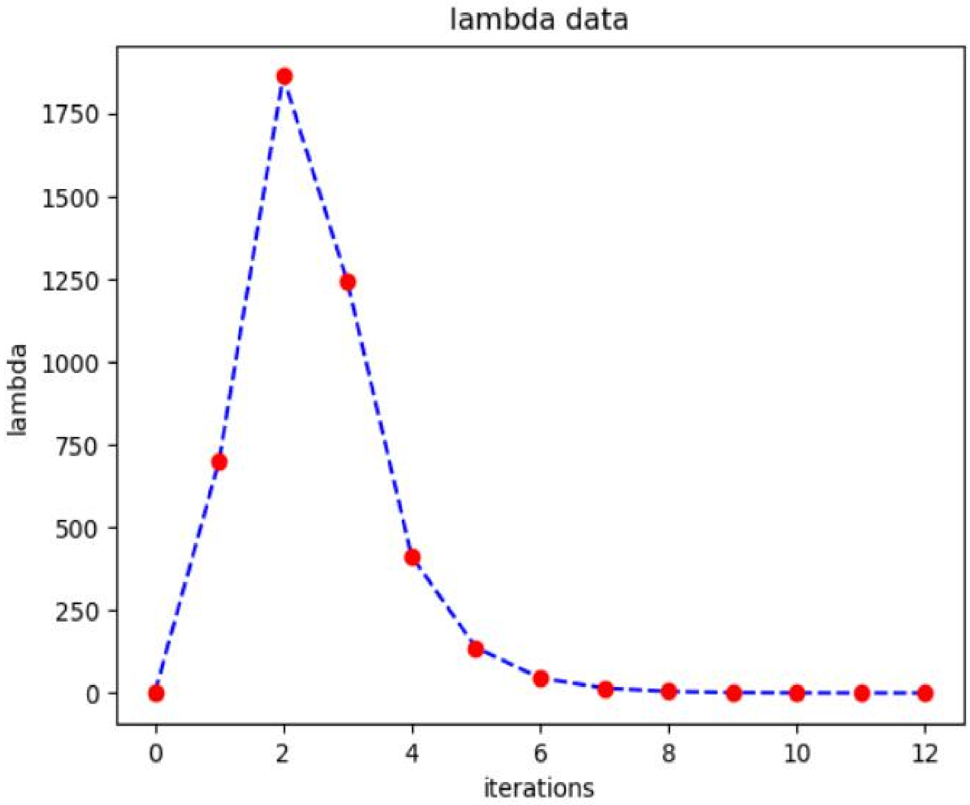
在1.3中对比时再讲它们的不同

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

    //**********************modified beginning**********************
    ofstream lambda_data("../../data/lambda_data.txt", ios_base::app);
    lambda_data << iter << "\t" << currentLambda_ << endl;
    lambda_data.close();
    //**********************modified ending**********************
  ...
}
```

1.1 绘制阻尼因子mu变化曲线

然后调用python程序绘制曲线图

```
#!/usr/bin/python
import matplotlib . pyplot as plt
filename = "lambda_data.txt"
X, Y = [],[]
for line in open(filename , 'r'):
value = [float (s) for s in line . split ()]
X.append(value [0])
Y.append(value [1])

plt . plot (X, Y, 'b--')
plt . plot (X, Y, 'ro')
plt . title ("lambda data")
plt . xlabel ("iterations")
plt . ylabel ("lambda")
plt . savefig ("./lambda_line_chart")
plt .show()
```



1.2 修改曲线方差，并计算其参数

将曲线函数改成 $y = ax^2 + bx + c$

残差模块的更改只需要将原函数 $y = \exp(ax^2 + bx + c)$ 更改为 $y = ax^2 + bx + c$，然后再减去观测值 y 即可 (在类中是成员变量 y_).

$y = ax^2 + bx + c$ 分别对 a, b, c 求导得到新的雅克比函数 $[x^2, x, 1]^\top$ 。然后用新的雅克比函数替换原程序中相应位置即可。

```
// 计算残差对变量的雅克比
// 对a, b, c求导，  而不是对x求导
virtual void ComputeJacobians() override
{
Vec3 abc = verticies_ [0]−>Parameters();
double y = abc(0) * x_ * x_ + abc(1) * x_ + abc(2) ;

// 误差为1维，  状态量 3 个，  得1x3 的雅克比矩阵
Eigen :: Matrix<double, 1, 3> jaco_abc;
// jaco_abc << x_ * x_ * exp_y, x_ * exp_y , 1 * exp_y;
jaco_abc << x_ * x_ , x_, 1 ;
jacobians_ [0] = jaco_abc;
}
```

1.3 实现其他阻尼因子策略

深蓝学院 shenlanxueyuan.com

1. $\lambda_0 = \lambda_o$; λ_o is user-specified [8].
   use eq'n (13) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (16) for ρ
   if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{h}$; $\lambda_{i+1} = \max[\lambda_i / L_\downarrow, 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \min[\lambda_i L_\uparrow, 10^7]$;

4.1.1 algorithm1

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

3. $\lambda_0 = \lambda_o \max\left[\mathsf{diag}[\boldsymbol{J}^\mathsf{T}\boldsymbol{W}\boldsymbol{J}]\right]$; λ_o is user-specified [9].
   use eq'n (12) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (15) for ρ
   if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{h}$; $\lambda_{i+1} = \lambda_i \max\left[1/3, 1 - (2\rho_i - 1)^3\right]$; $\nu_i = 2$;
   otherwise: $\lambda_{i+1} = \lambda_i \nu_i$; $\quad \nu_{i+1} = 2\nu_i$;

1.3 实现其他阻尼因子策略

深蓝学院 shenlanxueyuan.com

$\mathrm{H}\Delta\mathrm{x}_{\mathrm{gn}} = \mathrm{b}$，称其为 normal equation.

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

algorithm 2&3

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

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

algorithm 1

```
bool oneStepSuccess = false;
int false_cnt = 0;
while (!oneStepSuccess)  // 不断尝试 Lambda, 直到成功迭代一步
{
    // setLambda (H + lamdaD^TD)*delta_x = b
    AddLambdatoHessianLM();
    // 第四步，解线性方程 H X = B
    SolveLinearSystem();
    // 还原Hessian矩阵
    RemoveLambdaHessianLM();
    // 优化退出条件1:  delta_x_ 很小则退出
    if (delta_x_.squaredNorm() <= 1e-8 || false_cnt > 10) {
        stop = true;
        break;
    }
    // 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
    oneStepSuccess = IsGoodStepInLM();
    lambda_data_all << iter << "\t" << currentLambda_ << endl;
    // 后续处理，
    if (oneStepSuccess) {
        // 在新线性化点 构建 hessian
        MakeHessian(); // 每一次迭代都有MakeHessian()
```

$$\left[J^{\top} W J + \lambda \operatorname{diag}(J^{\top} W J)\right] h_{\mathrm{lm}} = J^{\top} W (y - \hat{y}) , \tag{13}$$

H△x = b，求出 delta_x_

AddHessian的反向操作，方便后面对Hessian的其他操作

1.3 实现其他阻尼因子策略

深蓝学院 shenlanxueyuan.com

```
void Problem::AddLambdatoHessianLM() {
    ulong size = Hessian_.cols();
    assert(Hessian_.rows() == Hessian_.cols() &&
"Hessian is not square");
    for (ulong i = 0; i < size; ++i) {
        Hessian_(i, i) += currentLambda_; // algo 2 & 3
        Hessian_(i, i) *= (1+currentLambda_);  // algo 1
    }
}
```

```
void Problem::RemoveLambdaHessianLM() {
    ulong size = Hessian_.cols();
    assert(Hessian_.rows() == Hessian_.cols() &&
"Hessian is not square");
    for (ulong i = 0; i < size; ++i) {
        Hessian_(i, i) -= currentLambda_;         // algo 2 & 3
        Hessian_(i, i) /= (1+currentLambda_);       // algo 1
    }
}
```

$\mathrm{H}\Delta \mathrm{x}_{\mathrm{gn}} = \mathrm{b}$，称其为 normal equation.

正常就是 H=JTWJ, 现在加入阻尼因子，所以有上面两个函数

$$\left[\boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{J} + \lambda \boldsymbol{I}\right] \boldsymbol{h}_{\mathsf{lm}} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \;, \tag{12}$$

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

1.3 实现其他阻尼因子策略

计算h_{lm}(delta_x_) 和 **ρ(rho)**   4.1.1 algorithm1

```
scale = delta_x_.transpose() * (currentLambda_ * Hessian_.diagonal() * delta_x_ + b_); // algo 1
```

```
scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);                 // algo 3
```

$$\rho_i(\boldsymbol{h}_{\text{lm}}) = \frac{\chi^2(\boldsymbol{p}) - \chi^2(\boldsymbol{p}+\boldsymbol{h}_{\text{lm}})}{(\boldsymbol{y}-\hat{\boldsymbol{y}})^\mathsf{T}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}) - (\boldsymbol{y}-\hat{\boldsymbol{y}}-\boldsymbol{J}\boldsymbol{h}_{\text{lm}})^\mathsf{T}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}-\boldsymbol{J}\boldsymbol{h}_{\text{lm}})} \tag{14}$$

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

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

1.3 实现其他阻尼因子策略

深蓝学院
shenlanxueyuan.com

判断本次迭代 IsGoodStepInLM()　　**4.1.1 algorithm1**

```
// algorithm 1 in 4.1.1
  if (rho > 0 && isfinite(tempChi)){
    currentLambda_ = (std::max) (currentLambda_/9. , 1e-7);
    currentChi_ = tempChi;
    return true;
  }
  else {
    currentLambda_ = (std::min) (currentLambda_ * 11, 1e7);
    return false;
  }
```

算法

1. $\lambda_0 = \lambda_o$; λ_o is user-specified [8].
   use eq'n (13) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (16) for ρ
   if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{h}$; $\lambda_{i+1} = \max[\lambda_i / L_\downarrow, 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \min[\lambda_i L_\uparrow, 10^7]$;

1.3 实现其他阻尼因子策略 Algo 2

判断本次迭代 IsGoodStepInLM()　　**4.1.1 algorithm2**

```
bool Problem::IsGoodStepInLM() {
    double scale = 0;

    double tempChi = 0.0;

    // algorithm 2 in 4.1.1
    Eigen::MatrixXd JWfh =  b_.transpose() * delta_x_;

    double d_JWfh = JWfh(0, 0);    // 这个矩阵只有一维，但是还是要取这个维
度

    double alpha = d_JWfh / ((currentChi_ - tempChi) / 2 + 2 * d_JWfh);

    alpha = (std::min)(0.1, alpha);
    RollbackStates();
    delta_x_ *= alpha;
    UpdateStates();
    scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);
    scale += 1e-3;   // make sure it's non-zero :)
```

```
// recompute residuals after update state
    // 统计所有的残差
    tempChi = 0.0;
    for (auto edge: edges_) {
        edge.second->ComputeResidual();
        tempChi += edge.second->Chi2();
    }
    // rho 比例因子，判断迭代是否good
    double rho = (currentChi_ - tempChi) / scale;

    // algorithm 2 in 4.1.1
    if(rho > 0 && isfinite(tempChi)){
        currentLambda_ = (std::max)(10e-7, currentLambda_ / (1 + alpha));
        currentChi_ = tempChi;
        return true;
    }
    else {
        currentLambda_ += std::abs((currentChi_ - tempChi) / (2 * alpha));
        RollbackStates();  // 失败之后RollBack
        return false;
    }
}
```

1.3 实现其他阻尼因子策略 Algo 2

深蓝学院 shenlanxueyuan.com

判断本次迭代 IsGoodStepInLM() **4.1.1 algorithm2**

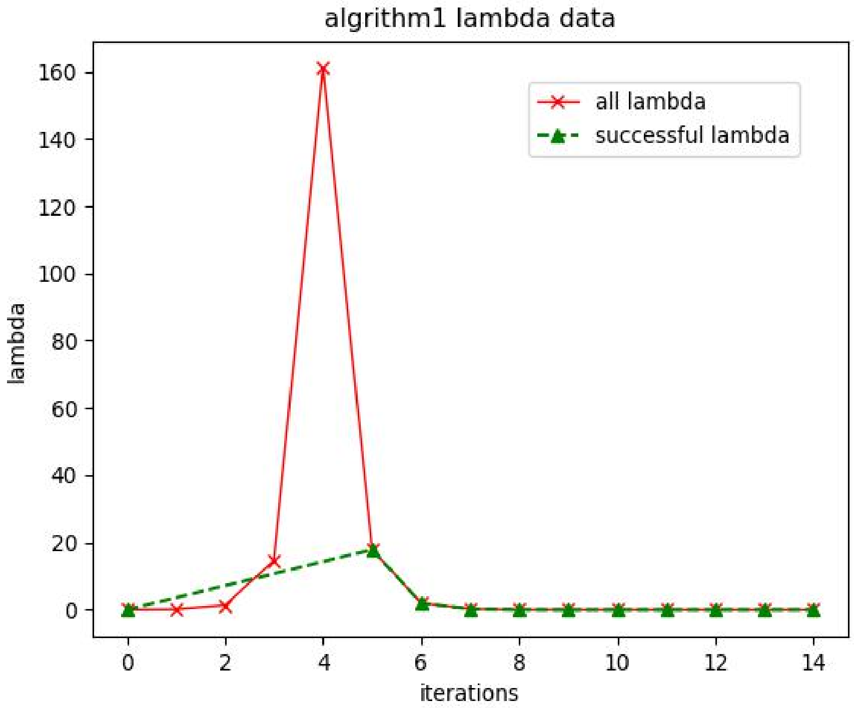
```
double tempChi = 0.0;

// algorithm 2 in 4.1.1
Eigen::MatrixXd JWfh =  b_.transpose() * delta_x_;

double d_JWfh = JWfh(0, 0);   // 这个矩阵只有一维，但是还是要取这个维度

double alpha = d_JWfh / ((currentChi_ - tempChi) / 2 + 2 * d_JWfh);

alpha = (std::min)(0.1, alpha);  // α大于0.1会无法迭代
RollbackStates();
delta_x_ *= alpha;  // 重新计算 delta_x_ 后再计算ρ
UpdateStates();
scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);
scale += 1e-3;    // make sure it's non-zero :)
```

```
// recompute residuals after update state
    // 统计所有的残差
    tempChi = 0.0;
    for (auto edge: edges_) {
        edge.second->ComputeResidual();
        tempChi += edge.second->Chi2();
    }
    // rho 比例因子，判断迭代是否good
    double rho = (currentChi_ - tempChi) / scale;

    // algorithm 2 in 4.1.1
    if(rho > 0 && isfinite(tempChi)){
      currentLambda_ = (std::max)(10e-7, currentLambda_ / (1 + alpha));
        currentChi_ = tempChi;
        return true;
    }
    else {
        currentLambda_ += std::abs((currentChi_ - tempChi) / (2 * alpha));
        return false;
    }
}
```

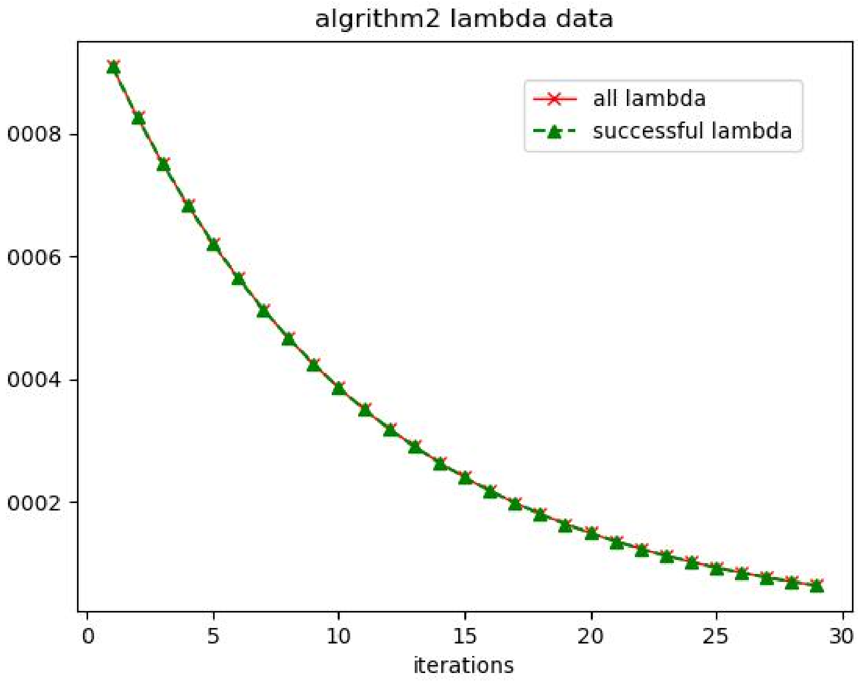
个人感觉不是特别elegant

1.3 实现其他阻尼因子策略

problem solve cost: 0.74ms **around 1 ms**
makeHessian cost: 0.34ms **algo1**



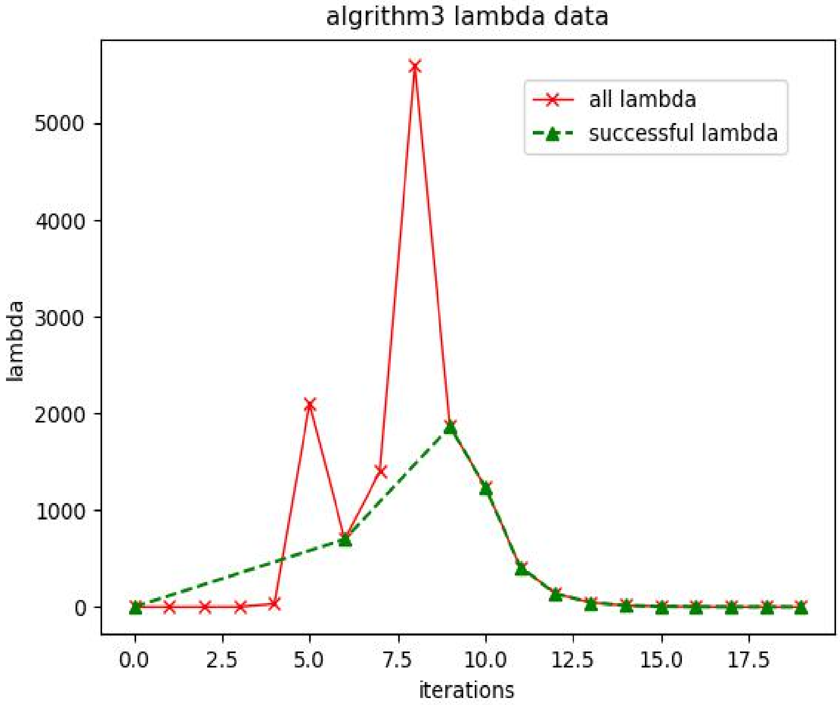
problem solve cost: **around 2ms**
makeHessian cost: 0.8ms **algo3**

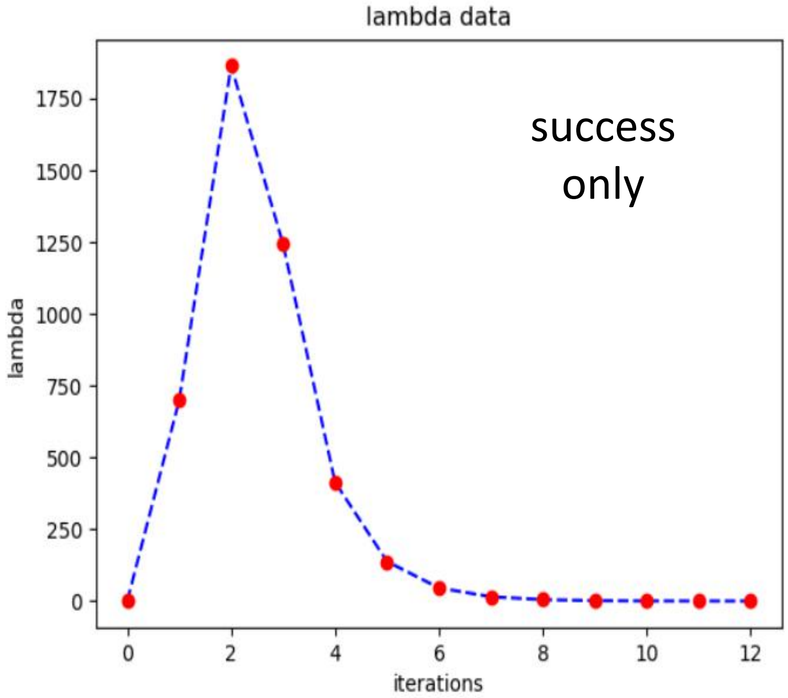


1.3 实现其他阻尼因子策略

problem solve cost: **around 2ms**

makeHessian cost: 0.8ms **algo3**





success only

2 证明题

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

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

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

$$= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}\left(\frac{1}{2}\left(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\right)\right)\delta t^2$$

2 证明题

深蓝学院
shenlanxueyuan.com

$$a = \frac{1}{2}\left(q_{b_ib_k}(a^{b_k} - b_k^a) + q_{b_ib_{k+1}}(a^{b_{k+1}} - b_k^a)\right)$$

$$= \frac{1}{2}\left(q_{b_ib_k}(a^{b_k} - b_k^a) + q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\right)$$

$$\alpha_{b_ib_{k+1}} = \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}a\delta t^2$$

$$= \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}\left(\frac{1}{2}\left(q_{b_ib_k}(a^{b_k} - b_k^a) + q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\right)\right)\delta t^2$$

2 证明题-f_{15}

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

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

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

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

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

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

2 证明题-g_{12}

深蓝学院
shenlanxueyuan.com

$$\alpha_{b_ib_{k+1}} = \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}a\delta t^2$$

$$= \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}\left(\frac{1}{2}\left(q_{b_ib_k}(a^{b_k} - b_k^a) + q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\right)\right)\delta t^2$$

$$g_{12} = \frac{\partial \alpha_{b_ib_{k+1}}}{\partial n_k^g} = \frac{\partial \frac{1}{4} q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{4}n_k^g\delta t \end{bmatrix}(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_k^g}$$

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

$$= \frac{1}{4}\frac{\partial R_{b_ib_{k+1}} exp\left(I + \left[\frac{1}{2}n_k^g\delta t\right]_\times\right)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_k^g}$$

$$= -\frac{1}{4}\frac{\partial R_{b_ib_{k+1}}([(a^{b_{k+1}} - b_k^a)\delta t^2]_\times)\left(\frac{1}{2}n_k^g\delta t\right)}{\partial n_k^g}$$

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

3 证明题

$$(J^T J+\mu I)\Delta x_{lm}=(V\Lambda V^T+\mu I)\Delta x_{lm}=(V(\Lambda+\mu I)V^T)\Delta x_{lm}=-J^T f=-F'^T$$

所以：

$$\Delta x_{lm}=-V(\Lambda+\mu I)^{-1}V^T F'^T=-[v_1\ v_2\ \cdots\ v_3]\begin{bmatrix}\frac{1}{\lambda_1+\mu} & & \cdots & \\ & \frac{1}{\lambda_2+\mu} & \cdots & \\ & & \ddots & \\ & & & \frac{1}{\lambda_n+\mu}\end{bmatrix}\begin{bmatrix}v_1^T\\ v_2^T\\ \vdots\\ v_n^T\end{bmatrix}F'^T$$

$$=-[v_1\ v_2\ \cdots\ v_3]\begin{bmatrix}\frac{v_1^T F'^T}{\lambda_1+\mu}\\ \frac{v_2^T F'^T}{\lambda_2+\mu}\\ \vdots\\ \frac{v_n^T F'^T}{\lambda_n+\mu}\end{bmatrix}=-\left(\frac{v_1^T F'^T}{\lambda_1+\mu}v_1+\frac{v_2^T F'^T}{\lambda_2+\mu}v_2+\cdots+\frac{v_n^T F'^T}{\lambda_n+\mu}v_n\right)=-\sum_{j=1}^{n}\frac{v_j^T F'^T}{\lambda_j+\mu}v_j$$

在线问答

Q&A

connection lost

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening