hw-l4 README

TASK 1

Consider a **convex optimization** problem

$$\begin{aligned} \min \ & f(x) \\ \text{s.t.} \ & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

Its PHR Augmented Lagrangian is defined as

$$\mathcal{L}_\rho(x, \lambda, \mu) := f(x) + \frac{\rho}{2}\left\{\left\|h(x) + \frac{\lambda}{\rho}\right\|^2 + \left\|\max\left[g(x) + \frac{\mu}{\rho}, 0\right]\right\|^2\right\} - \frac{1}{2\rho}\left\{\|\lambda\|^2 + \|\mu\|^2\right\}$$

where $\rho > 0, \mu \succeq 0$.

Prove that **the PHR Lagrangian is always convex with respect to x**.

由已知条件可知

f(x),g(x),h(x)均是关于 x 的凸函数,要证明 PHR 增广拉格朗日函数为凸函数

$$\mathcal{L}_\rho(x, \lambda, \mu) := f(x) + \frac{\rho}{2}\left\{\left\|h(x) + \frac{\lambda}{\rho}\right\|^2 + \left\|\max\left[g(x) + \frac{\mu}{\rho}, 0\right]\right\|^2\right\}$$

已知正向加权和，逐点取大运算(point-wise max)不改变函数的凸性，那么证明对非负凸函数取 2 范数的平方是否改变凸性，即函数平方不改变凸性

设： $w(x) \in [0, +\infty)$ 为凸函数

在函数定义域内任取 $x_1, x_2, \forall x_1 \neq x_2 \in \mathbb{R}$

对凸函数不等式同取平方

$$w(\frac{x_1 + x_2}{2}) \leq \frac{w(x_1) + w(x_2)}{2} \tag{1}$$

$$\xi^2(\frac{x_1 + x_2}{2}) \leq \frac{\xi^2(x_1) + 2\xi(x_1)\xi(x_2) + \xi^2(x_2)}{4} \tag{2}$$

不等式等式右侧减去 $\frac{\xi^2(x_1)+\xi^2(x_2)}{2}$

整理得：

$$\frac{-(w(x_1) - w(x_2))^2}{4} \leq 0$$

等式恒成立

故凸函数的平方仍然是凸函数，可知 PHR 增广拉格朗日函数为凸函数

> 在网上看到一个有趣的思路

> 对
>
> $$\left\|\max\left[g(x)+\frac{\mu}{\rho},0\right]\right\|^2$$
>
> 当两边都为非负数时，两边同时平方就相当于两边同时乘以一个整数，而且越大的一方乘以了一个相对较大的数，所以不等式依然成立

TASK 2

Provided a low-dimensional strictly convex QP solver that only solves:

$$\min_{x\in\mathbb{R}^n}\frac{1}{2}x^{\mathrm{T}}M_Qx+c_Q^{\mathrm{T}}x,\ \mathrm{s.t.}\ A_Qx\le b_Q$$

M_Q>0

Please design a scheme to approximately solve the case where M_Q⩾0

You can combine the low-dimensional strictly convex QP solver with a “proximal” term.
A C++ version of the solver is provided as below:
https://github.com/ZJU-FAST-Lab/SDQP

A test example for a positive semi-definite case

A test example for a positive semi-definite case:

$$M_Q=\begin{pmatrix}8&-6&2\\-6&6&-3\\2&-3&2\end{pmatrix}\quad c_Q=\begin{pmatrix}1\\3\\-2\end{pmatrix}\quad A_Q=\begin{pmatrix}0&-1&-2\\-1&1&-3\\1&-2&0\\-1&-2&-1\\3&5&1\end{pmatrix}\quad b_Q=\begin{pmatrix}-1\\2\\7\\2\\-1\end{pmatrix}$$

$\begin{pmatrix}-\frac{103}{97}\\-\frac{93}{97}\\\frac{95}{97}\end{pmatrix}$ is an optimal solution.

$-\dfrac{295}{97}$ is the constrained minimum

对于 $M_Q\ge 0$，需要将其转换为一个实对称正定矩阵。从而使用 SDQP 进行求解，参考网上的做法引入一个先验的 $\overline{x}$，使得目标函数近似转换为：

$$\begin{aligned}&\frac{1}{2}x^TM_Qx+c_Q^Tx+\frac{1}{2\rho}\|x-\overline{x}\|^2\\&=\frac{1}{2}x^TM_Qx+c_Q^Tx+\frac{1}{2\rho}(x-\overline{x})^T(x-\overline{x})\\&=\frac{1}{2}x^T\ \left(M_Q+\frac{1}{\rho}I\right)\ x+(c_Q-\frac{1}{\rho}\overline{x})^Tx+\frac{1}{\rho}\overline{x}^T\overline{x}\end{aligned}$$

$\rho\to+\infty$ 时等价与原目标函数，将 M_Q 转化为 $M_Q+\frac{1}{\rho}I>0$，则可使用 SDQP 求解器求解

```C++
#include <iostream>

#include "sdqp/sdqp.hpp"

using namespace std;
using namespace Eigen;

int main(int argc, char **argv)
{
```

```
    const int d=3; //variable dim
    const int m=5; //constraint dim
    Matrix<double,d,3> MQ, MQ_prox,QT;
    Matrix<double,d,1> cQ,cQ_prox;
    Matrix<double,d,1> x,x_ba,x_opt;
    Matrix<double,m,d> AQ;
    MatrixXd I=MatrixXd::Identity(3,3);
    VectorXd b(m);
    MQ << 8,-6,2,
          -6,6,-3,
          2,-3,2;
    cQ<<1,3,-2;
    AQ<< 0, -1,-2,
         -1,1,-3,
         1,-2,0,
         -1,-2,-1,
         3,5,1;
    b<<-1,2,7,2,-1;
    double dis=1e6;
    double rho=1.0;
    int iter=0;
    double minobj=1e6;
    while(dis>1e-3){
        MQ_prox=MQ+I/rho;
        cQ_prox=cQ-x_ba/rho;
        VectorXd eigen_vec=MQ_prox.eigenvalues().real().transpose();
        double eigen_min_value=eigen_vec.minCoeff();
        if(eigen_min_value)
          minobj= sdqp::sdqp<3>(MQ_prox,cQ_prox,AQ,b,x);
        else
        {
            cout<<"error"<<endl;
            break;
        }

        dis=(x-x_ba).lpNorm<Infinity>();
        x_ba=x;
        rho=min(rho*10,1e6);


        cout<<"x opt value="<<x.transpose()<<endl;
        cout<<"obj value=" <<0.5*x.transpose()*MQ*x+cQ.transpose()*x<<endl;
        cout<<"trans obj value ="<<minobj<<endl;
        iter++;
        cout<<"iter time"<<iter<<endl;
    }
    x_opt << -103.0 / 97.0, -93.0 / 97.0, 95.0 / 97.0;
    cout<<"given x_opt : "<<x_opt.transpose()<<endl;
    cout<<"given obj value: "<<-295.0/97.0<<endl;


    return 0;
}
```

输出结果：

```Python
x opt value=-0.706806 -0.623037  0.811518
obj value=-2.65012
trans obj value =-1.87696
iter time1
x opt value= -1.06151 -0.958994  0.979497
obj value=-3.04124
trans obj value =-3.10521
iter time2
x opt value= -1.06186 -0.958763  0.979382
obj value=-3.04124
```

```
trans obj value =-3.05627
iter time3
given x_opt :  -1.06186 -0.958763  0.979381
given obj value: -3.04124

Process finished with exit code 0
```

迭代三次就达到 1e-3 精度，与给定结果一致

TASK 3

Please solve the SOCP below via Conic ALM.

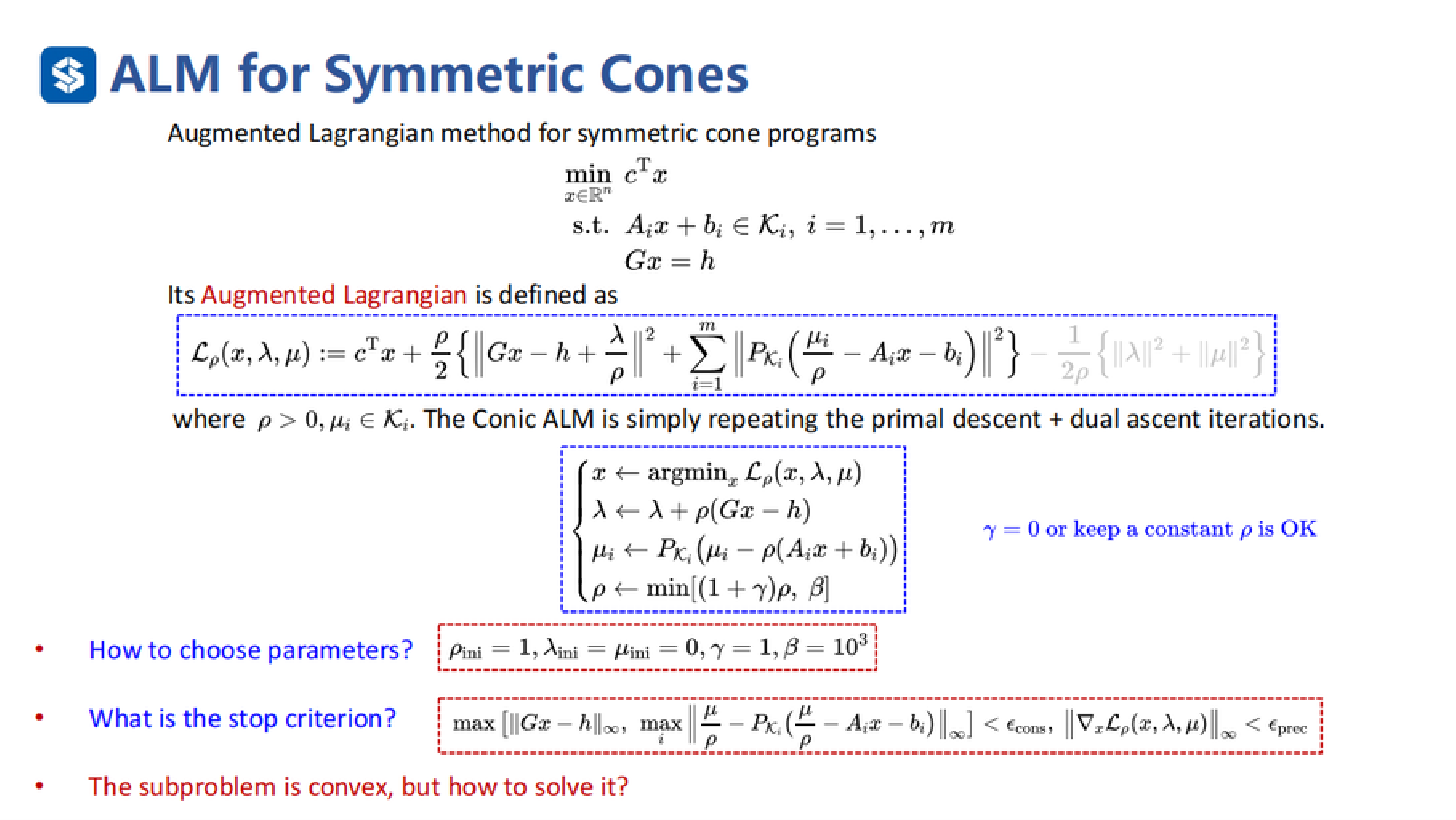
$$
\begin{aligned}
\min_{a,b,c,d,e,f,g\in\mathbb{R}} \quad & a+2b+3c+4d+5e+6f+7g \\
\text{s.t.} \quad & \|(7a+1,6b+3,5c+5,4d+7,3e+9,2f+11,g+13)\| \le a+1.
\end{aligned}
$$

An approximate optimal solution is:

$$(-0.127286, -0.506097, -1.01317, -1.77744, -3.06097, -5.66462, -13.7682)$$

An approximate constrained minimum is: -156.9589

题目需要转化为 如 PPT 中 conic ALM 形式



首先将问题转化成如下 SOCP 标准形式

$$
\begin{aligned}
\min_{x\in\mathbb{R}^7} \quad & f^T x \\
\text{s.t.} \quad & \|Ax+b\| \le c^T x + d.
\end{aligned}
$$

则原问题对应参数

$f=[1,2,3,4,5,6,7]^T$, x为优化变量, $b=[1,3,4,7,9,11,13]^T$, $c=[1,0,0,0,0,0,0]^T$

$$A=\begin{bmatrix} 7 & & & & & & \\ & 6 & & & & & \\ & & 5 & & & & \\ & & & 4 & & & \\ & & & & 3 & & \\ & & & & & 2 & \\ & & & & & & 1 \end{bmatrix}$$

这里将等于情况加入，则：

$$\overline{A}x+\overline{b}=\begin{pmatrix} c^{\mathrm{T}} \\ A \end{pmatrix}x+\begin{pmatrix} d \\ b \end{pmatrix}=\begin{pmatrix} c^{\mathrm{T}}x+d \\ Ax+b \end{pmatrix}\in\mathcal{Q}^8$$

对应 conic ALM 为：

$$\mathcal{L}_\rho(x,\mu)=f^{\mathrm{T}}x+\frac{\rho}{2}\left\|P_{\mathcal{K}=\mathcal{Q}^n}\left(\frac{\mu}{\rho}-\overline{A}x-\overline{b}\right)\right\|^2$$

其中投影锥函数，

$$P_{\mathcal{K}=\mathcal{Q}^n}(v)=\begin{cases} 0, & v_0\le-\|v_1\|_2 \\ \frac{v_0+\|v_1\|_2}{2\|v_1\|_2}\left(\|v_1\|_2,v_1\right)^{\mathrm{T}}, & |v_0|<\|v_1\|_2 \\ v, & v_0\ge\|v_1\|_2 \end{cases}$$

对正的部分保留，对表达式进行谱分解

梯度函数：

$$\nabla_x\mathcal{L}_\rho(x,\mu)=f-\rho\overline{A}^T P_{\mathcal{K}=\mathcal{Q}^n}\left(\frac{\mu}{\rho}-\overline{A}x-\overline{b}\right)$$

根据下面的进行迭代即可

$$\boxed{\begin{cases} x\leftarrow\operatorname{argmin}_x\mathcal{L}_\rho(x,\lambda,\mu) \\ \lambda\leftarrow\lambda+\rho(Gx-h) \\ \mu_i\leftarrow P_{\mathcal{K}_i}\left(\mu_i-\rho(A_ix+b_i)\right) \\ \rho\leftarrow\min[(1+\gamma)\rho,\ \beta] \end{cases}}$$

结果：

```Python
================================
Iteration: 19
Function Value: -125.3
Gradient Inf Norm: 8.012e-08
Variables:
-0.1273 -0.5061  -1.013  -1.777  -3.061  -5.665  -13.77
```

使用 lbfgs 迭代两次就收敛了

Problem

可以使用 Hessian 去做，

Conic ALM

The gradient can be directly computed as

$$\nabla_x \mathcal{L}_\rho(x,\lambda,\mu) = c + G^{\mathrm{T}}(\lambda + \rho(Gx - h)) - \sum_{i=1}^{m} A_i^{\mathrm{T}} P_{\mathcal{K}_i}(\mu - \rho(A_i x + b_i))$$

SDP

The so-called generalized Hessian, i.e., B-subdifferential of the gradient is thus

$$\partial_B \nabla_x \mathcal{L}_\rho(x,\lambda,\mu) = \rho\Big(G^{\mathrm{T}}G + \sum_{i=1}^{m} A_i^{\mathrm{T}} \partial_B P_{\mathcal{K}_i}(\mu - \rho(A_i x + b_i)) A_i\Big)$$

Specifically, for SOCP the B-subdifferential can be chosen as

$\partial_B P_K$

$$\partial_B P_{\mathcal{K}_i}(x) \ni \begin{cases} I_n & \|x_2\| \le x_1, \\ 0 & \|x_2\| \le -x_1, \\ \begin{pmatrix} \frac{1}{2} & \frac{x_2^{\mathrm{T}}}{2\|x_2\|} \\ \frac{x_2}{2\|x_2\|} & \frac{x_1 + \|x_2\|}{2\|x_2\|} I_{n-1} - \frac{x_1 x_2 x_2^{\mathrm{T}}}{2\|x_2\|^3} \end{pmatrix} & \|x_2\| > |x_1|. \end{cases}$$

P_K

但如何求谱分解的 B-subdifferential 需要求教一下助教老师

ref

Convex Optimization: 3 Convex functions - winechord - 博客园

这是凸优化第三章的笔记文章目录Definit...

www.cnblogs.com

优秀学员

助教