1 准备工作

- 下载CVX. 目前的下载网址为: http://cvxr.com/cvx/download/.
- 以CVX2.0在Window系统64位微型计算机简介其安装过程. “cvx_w64.zip”压缩文件下载到计算机上, 解压存储在计算机的一个目录下, 如目录D:\cvx-w64下. 运行Matlab, 在Matlab中通过文件浏览将工作路径设置在D:\cvx-w64\cvx 下. 然后在命令行窗口运行cvx_setup后, 可看到运行的结果, 其中有“2 solvers initialized”和“3 solvers skipped”等信息. 此时, SeDuMi和SDPT3已成功链接, 可以使用CVX的的核心软件了, 但添加的商业软件则被跳过链接而无法使用.
- 运行商业软件在D:\cvx-w64\cvx下, cvx_setup以一个cvx_setup.m文件形式存储, 可在Matlab 编辑器中打开cvx_setup.m文件, 并且运行这个文件, 则得到与命令行窗口运行cvx_setup相同的结果.

  若需要使用CVX添加的商业软件CPLEX和GUROBI, 则需要联系软件开发方得到一个名称为cvx_license.mat使用许可密码文件, 若存储在D:\cvx-w64\cvx\lic下, 则完全遵循上述的步骤, 启动命令cvx_setup\lic\cvx_license.mat即可.

2 编程及简单算例

例 1 考虑如下的线性规划的标准模型

$$\begin{aligned} \min \quad & -x_1 - 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 40 \\ & 2x_1 + x_2 + x_4 = 60 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

*CVX*编程为

```
cvx_begin                          % CVX的标准开始语句.
    A = [1 1 1 0; 2 1 0 1];        % 约束的2 × 4的矩阵.
    b = [40 60]';                  % 约束的右端项.
    c = [-1 -2 0 0]';              % 目标函数的系数.
    variable x(4);                 % 设定x为变量.
    minimize(c' * x);              % 目标函数.
    subject to                     % 与模型一致, 不起任何作用.
        A * x == b;                % 约束方程.
        x == nonnegative(4);       % 变量为第一卦限.
cvx_end                            % CVX的标准结束语句.
x                                  %注意写在CVX模块之外.
```

计算输出包括调用的核心软件、变量个数、约束方程的情况、调用软件的具体算法及其参数等、算法的迭代次数、目标和对偶目标值、偏差程度、解的近似程度和计算时间等, 最终可看到输出：

Status : Solved

Optimal value(cvx_optval) : -80

x = 0 40 0 20.

表示该问题得以求解, 最优解$x = (0, 40, 0, 20)^T$, 最优目标值为：-80.

- 程序以模块形式表达, cvx_begin开始, cvx_end结束, 中间采用Matlab语言和CVX特定的函数语言编写.
- 在优化目标和约束描述之前给出变量的定义. 用variable定义一个变量, variables定义多个变量, 变量之间用空格间隔. 变量必须明确其维数.
- 优化目标和约束描述仿实际优化模型形式编写. 目标命令为minimize()或maximize(), 分别表示对括号内的目标函数极

小化和极大化. 约束通过命令subject to 来表示, 实际上这个命令不起任何作用, 省略后对程序没有影响.

- 约束在目标命令之后, 都采用Matlab中的逻辑关系“ >= ”, “ <= ”和“ == ”表示.
- 有逻辑运算符号的表达式被CVX认定为约束语句.
- 锥的形式语言, 如上述程序的x==nonnegative(4), 表示限定在第一卦限.
- CVX模块中的逻辑关系“ >= ”, “ <= ”可以用“ > ”, “ < ”替代, 它们具有等价的作用.
- 需要注意模块之内约束符号“==”与赋值“=”的差别. 它们之间不能替代, 否则将出现错误.
- CVX中的variable x(n)为模块内的全程变量, 模块内不能重新计算赋值, 如在模块内出现x(1)=1, 则出现赋值覆盖, 是一个错误. 当约束要求第一个分量为1时, 则在模块内写成约束形式x(1)==1.
- cvx_end后的x表示输出最优解x的结果.
- x == nonnegative(n) 也可以直接用Matlab语言写成$x >= 0$.
- 上述的0被默认为一个n维列向量. 一般的$l \leq x \leq u$中的l和u需赋予具体的n维列向量数值, 如$0 \leq x_1 \leq 1, 3 \leq x_2 \leq 4$可写成

l=[0 3]';
u=[1 4]';
$l <= x <= u$;

3 二阶锥优化问题

- CVX特别将二阶锥$(x,y)^T \in \mathcal{L}^{n+1}$表示成

  variables x(n) y ;
  {x, y} <In> lorentz(n);

- 需要注意, {x, y} <In> lorentz(n)的表达式中是lorentz(n), 而不是lorentz(n+1)!

- 在Matlab中有二阶锥等价的表示语句,

  variables x(n) y ;
  norm(x, 2)<= y;

- 凸性规则. 在CVX中, 要特别注意函数的凸性规则, 下列Matlab语句虽也起到与上函数要求的相同功效,

  variables x(n) y ;
  sqrt(x′∗x)<= y;

  但对凸函数$f(x)$, $\sqrt{f(x)}$不一定是凸函数, CVX不认为是合法语句, 会输出

  Disciplined convex programming error:
  Illegal operation: sqrt(convex).

  的错误信息, 表示不满足CVX的凸性规则, 这一点需特别注意.

例 2 某公司有6个建筑工地要开工, 每个工地的位置*(*用平面坐标*a, b*表示, 距离单位：公里*)*及水泥日用量*d(*吨*)*由表1 给出. 现规划建立一个新的料场并假设从料场到工地之间均有直线道路相连, 试选定料场位置使总的吨公里数最小.

	1	2	3	4	5	6
a	1.25	8.75	0.5	5.75	3	7.25
b	1.25	0.75	4.75	5	6.5	7.75
d	3	5	4	7	6	11

Table 1: 工地的位置(a,b)及水泥日用量d

解： 设待建料场位置为$x,y,$ 则该问题的优化模型为：

$$\begin{aligned}
\min \quad & 3t_1+5t_2+4t_3+7t_4+6t_5+11t_6 \\
\text{s.t.} \quad & \sqrt{(x-1.25)^2+(y-1.25)^2}\le t_1 \\
& \sqrt{(x-8.75)^2+(y-0.75)^2}\le t_2 \\
& \sqrt{(x-0.5)^2+(y-4.75)^2}\le t_3 \\
& \sqrt{(x-5.75)^2+(y-5)^2}\le t_4 \\
& \sqrt{(x-3)^2+(y-6.5)^2}\le t_5 \\
& \sqrt{(x-7.25)^2+(y-7.75)^2}\le t_6 \\
& x,y,t_i(i=1,2,\ldots,6)\in\mathbb{R}.
\end{aligned}$$

上述模型是第一章第二节的*Torricelli*点问题的变形, 是一个二阶锥规划问题. *CVX*的程序如下*:*

```
cvx_begin
    a = [1.25 8.75 0.5 5.75 3 7.25]';
    b = [1.25 0.75 4.75 5 6.5 7.75]';
    d = [3 5 4 7 6 11]';
    variables x(2) t(6);
    minimize(d' * t);
    subject to
        norm(x - [a(1) b(1)]') <= t(1);                  % 采用2范数程序语言.
        norm(x - [a(2) b(2)]') <= t(2);
        norm(x - [a(3) b(3)]') <= t(3);
        norm(x - [a(4) b(4)]') <= t(4);
        {x - [a(5) b(5)]', t(5)} < In > lorentz(2);      %采用二阶锥程序语言.
        {x - [a(6) b(6)]', t(6)} < In > lorentz(2);
cvx_end
x                                                        %输出选定料场位置.
```

以上算例得以成功计算, 目标的最优值为: 117.855(吨公里), 新建料场位置为：$(5.7279, 5.0414)^T$.

4 半定优化问题

- $X \in \mathcal{S}^n$, CVX写成

  variable X(n, n) symmetric

- 半正定约束$X \in \mathcal{S}_+^n$用

  X==semidefinite(n)

  表示.

- 半正定锥上的关系$X \succeq Y$表示成

  X-Y==semidefinite(n)

- n阶矩阵与$X \in \mathcal{S}^n$的Frobenius内积$A \bullet X$写成

  trace(A∗X)

例 3 如下的程序

```
n = 6;
A = ones(n, n);  C = eye(n);  b = 2;
cvx_begin
variable X(n, n) symmetric;
minimize(trace(C * X));
subject to
    trace(A * X) >= b;
    X(1, 1) == 1;
    X == semidefinite(n);
cvx_end
X
```

求解下列半定规划问题

$$\begin{aligned} \min \quad & C \bullet X \\ \text{s.t.} \quad & A \bullet X \geq b \\ & x_{11} = 1 \\ & X = (x_{ij}) \in \mathcal{S}_+^n \end{aligned}$$

其中, $n = 6$, C为单位矩阵, A为元素都为1的矩阵, $b = 2$. 计算输出的部分结果为：

Status : Solved

Optimal value (cvx_optval) : +1.03431

X =

1.0000 0.0828 0.0828 0.0828 0.0828 0.0828

0.0828 0.0069 0.0069 0.0069 0.0069 0.0069

0.0828 0.0069 0.0069 0.0069 0.0069 0.0069

0.0828 0.0069 0.0069 0.0069 0.0069 0.0069

0.0828 0.0069 0.0069 0.0069 0.0069 0.0069

0.0828 0.0069 0.0069 0.0069 0.0069 0.0069

5 对偶变量的处理

- 变量定义

  dual variable z

  表示对偶变量, 而不需要给出其维数. 在约束给出对偶变量的匹配, 如线性规划标准型的例子加入

  z : A∗x == b

  或等价的

  A∗x == b : z

就可以得到对偶解的信息.

- 对于对偶变量的设定, 必须有线性锥优化理论的基础, 才可能准确定义出对偶变量的对应关系.

  对于线性规划标准形模型的CVX程序, 修改为

```
cvx_begin
A=[1 1 1 0; 2 1 0 1];
b=[40 60]';
c=[-1 -2 0 0]';
variable x(4);
dual variables y z;                % 设定y和z为对偶变量.
minimize(c'*x);
subject to
        y : A*x==b;                % 对应等式约束的对偶变量, 2维.
        z : x==nonnegative(4);     % 对应变量x的对偶变量, 4 维.
cvx_end
x
y                                  % 输出对偶变量值.
z                                  % 输出对偶变量值.
```

  输出的部分结果为：

```
Status: Solved
Optimal value (cvx_optval): -80
x = -0.0000   40.0000   0   20.0000
y =-2   0
z= 1   0   2   0
```

- CVX目前无法给出本书第五章第二节有关二阶锥规划各模型的二阶锥对偶变量信息. 实际上CVX给出的是Lagrange对偶的Lagrange乘子的信息(参考第四章).

- 对于半定规划, 则可以得到对偶变量的信息, 如上述的半定规划例子修改后

```
n = 6;
A = ones(n, n);
C = eye(n);
b =2;
cvx_begin
variable X(n, n) symmetric;
dual variables y1 y2 V;
minimize( trace( C*X ) );
subject to
        y1 : trace( A*X ) ¿= b;
        y2 : X(1, 1)==1;
        V : X == semidefinite(n);
cvx_end
y1
y2
V
```

部分输出结果为：

Status: Solved

Optimal value (cvx_optval): +1.03431

y1 = 0.0586

y2 = 0.9172

V =

0.0243 -0.0586 -0.0586 -0.0586 -0.0586 -0.0586

-0.0586 0.9414 -0.0586 -0.0586 -0.0586 -0.0586

-0.0586 -0.0586 0.9414 -0.0586 -0.0586 -0.0586

-0.0586 -0.0586 -0.0586 0.9414 -0.0586 -0.0586

-0.0586 -0.0586 -0.0586 -0.0586 0.9414 -0.0586

-0.0586 -0.0586 -0.0586 -0.0586 -0.0586 0.9414

6 可计算凸优化规则及核心函数库

CVX可求解的凸优化问题必须满足其设定的凸优化规则(disciplined convex programming). CVX 可以作为一个平台软件的核心是给出了这些可计算规则, 同时以一个核心函数库的方式记录那些可计算凸优化问题并允许不断的扩展.

CVX的表达式中仅考虑常数、线性函数、凸函数和凹函数四类. 优化问题限定为三类问题：第一类为极小化问题, 要求目标函数为凸函数, 分无约束和有约束两种情形; 第二类为极大化问题, 要求目标函数为凹函数, 分无约束和有约束两种情形; 第三类为可行性问题, 包含至少一个约束.

有效约束必须满足以下三个规则：

- 等号约束, 采用关系符号“==”, 如果两端都是函数形式, 要求等号两端必须为变量的线性函数；如果一端为集合, 另一端必须为变量的线性函数.
- 小于等于约束, 关系符号“<=” 或“<”都可以等价使用, 不等号

的左端是凸函数且右端为凹函数.

- 大于等于约束, 关系符号“>=”或“>”都可以等价使用, 不等号的左端是凹函数且右端为凸函数.

不等号约束“~=”不容许使用. 上述约束逻辑运算符号的两端可以是向量的形式, 但必须符合要求.

对于目标函数或约束中的表达式, 有效的表达形式必须满足下列规则：

- 常数表达式, 是Matlab的数值运算且结果为有限值.
- 线性表达式, 具有下列情形之一：
  - 常数表达式,
  - 表示变量所在的集合,
  - 调用已标明为线性的Matlab中或自定义的核心函数,
  - 线性函数的和或差,
  - 线性函数与常数表达式的乘积.
- 凸函数表达式, 具有下列情形之一：
  - 调用已标明为凸的Matlab中或自定义的核心函数,
  - 线性函数的偶数幂函数,
  - 凸表达式的和,
  - 一个凸表达式和一个凹表达式的差,
  - 一个凸表达式和一个非负常数的乘积,
  - 一个凹表达式和一个非正常数的乘积.
- 凹函数表达式, 具有下列情形之一：
  - 调用已标明为凹的Matlab中或自定义的核心函数；
  - 线性函数的p幂函数, 其中$p \in (0, 1)$;

  - 凹表达式的和；
  - 一个凹表达式和一个凸表达式的差；
  - 一个凹表达式和一个非负常数的乘积；
  - 一个凸表达式和一个非正常数的乘积.

- 复合函数$f(g(x))$表达式, 具有下列情形之一：
  - $f(u)$为凸、凹或线性函数, $g(x)$为线性函数; $f(g(x))$ 分别为凸、凹或线性函数.
  - $f(u)$是非减且凸函数, $g(x)$是凸函数, $f(g(x))$是凸函数；
  - $f(u)$是非增且凸函数, $g(x)$是凹函数, $f(g(x))$是凸函数；
  - $f(u)$是线性函数, $g(x)$是线性函数, $f(g(x))$是凸函数；
  - $f(u)$是非减且凹函数, $g(x)$是凹函数, $f(g(x))$是凹函数；
  - $f(u)$是非增且凹函数, $g(x)$是凸函数, $f(g(x))$是凹函数；
  - $f(u)$是线性函数, $g(x)$是线性函数, $f(g(x))$是凹函数.

上述所有的线性、凸、凹、非增、非减等函数必须是CVX核心函数库认定的合法函数, 这样才能进行上述的规则判定. 当上述表达式中$g(x)$为向量函数(泛函)时, 线性、凸、凹、非增、非减都按向量函数的对应性质讨论.

如果一个表达式无法满足上述规则, 则CVX将返回错误信息而拒绝进一步计算. 如

$$\text{sqrt}(\text{x}'*\text{x})<= \text{y};$$

CVX不认为是有效语句, 会输出

```
Disciplined convex programming error:
Illegal operation: sqrt(convex).
```

的错误信息.

简单例子

- 错误表示：sqrt($x' * x$). $f(u) = \sqrt{u}$没有在CVX的核心函数库中被定义为合法凸函数, 如当$g(x) = x(x-1), x \in \mathbb{R}$时, $\sqrt{x(x-1)}$在$x \geq 1$或$x \leq 0$才有定义, 不能认为其为$\mathbb{R}$的凸函数.

- 正确表示. norm()被CVX核心函数库接受. 下面的表达式

  $$\text{norm(A*x-b)+c*norm(x, 1)}$$

  A*x-b为线性向量函数, norm()为单调升的凸函数, norm(A*x-b)和norm(x, 1)符合复合函数的第一条规则, 整体表达式norm(A*x-b)+c*norm(x, 1)符合凸函数表达式规则之三.

- 注意一些理论上的凸(凹)函数可能不满足这些规则而不能被CVX核心函数库认为合法.

  如$(x^2+1)^2 = x^4+2x^2+1, x \in \mathbb{R}$明显为一个凸函数, 但Matlab的

  $$\text{square(square(x) + 1)}$$

  命令不被CVX认为是凸函数. 究其原因是其采用了函数复合的形式, $f(u) = u^2$, $g(x) = x^2 + 1$. 此时来看, $g(x)$明显是一个凸函数, $f(u)$是一个凸函数但不是非减, 不满足复合函数的规则, 因此CVX认为无效. 如$g(x) = x^2 - 1$, 有$(x^2 - 1)^2$不是凸函数.

- l_1范数模型的实现

  $$\min_{x \in \mathbb{R}^n} \|Ax - b\|_1$$

  当n, m, A和b已输入, l_1范数模型的编程为：

cvx_begin

variable x(n);

minimize(norm(A∗x-b, 1)); % 其中norm(A∗x-b, 1) 表示l_1范数

cvx_end

- l_∞范数模型

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|_\infty$$

的编程为：

cvx_begin

variable x(n);

minimize(norm(A∗x-b, Inf)); % 其中norm(A∗x-b, Inf)表示l_1范数

cvx_end

- l_2范数. norm(A∗x-b, 2)可默认地写成norm(A∗x-b).

- 直接库函数命令：

$$\text{quad\_over\_lin(A}*\text{x-b, c}'*\text{x+d)}$$

其中A为$m\times n$实系数矩阵, x为n维列决策变量, b为m 维列系数向量, c为n维列系数向量, d为常数, 其表达的函数关系为

$$\frac{(Ax-b)^T(Ax-b)}{c^Tx+d}.$$

若用上述命令求解, 首先将V分解成$V=A^TA$, 其中A为n阶实矩阵；然后

```
A=[.....];                                  % 假设已输入A矩阵.
b=[.....]';                                 % 假设已输入b向量.
n=size(b);                                  % 计算出b的维数, 也是A的阶数.
cvx_begin
variable x(n);
minimize( quad_over_lin( A*x, b'*x) );
subject to
        b'*x¿=0;
        ones(1, n)*x==1;
        x¿=0;
cvx_end
```

就是求解的程序.

注意quad_over_lin(A∗x-b, c′∗x+d)命令中, 分子是$(Ax-b)^T(Ax-b)$的形式, 自然保证这是一个凸函数. 对目标函数的分子x^TVx, 如要采用CVX的quad_over_lin(A∗x-b, c′∗x+d)命令还需做分解$V = A^TA$. 当V为半正定矩阵时, x^TVx是一个凸函数, 我们可以自己设计一个计算的程序加载到CVX核心函数库中, 就不需要这样的分解了.

quad_over_lin(A∗x-b, c′∗x+d)作为一个Matlab中的函数可在CVX环境外使用. 当$c^Tx+d \geq 0$时, 它输出$\frac{(Ax-b)^T(Ax-b)}{c^Tx+d}$函数值, 当$c^Tx+d < 0$时, 它输出$+\infty$.

7 参数控制

CVX目前支持的核心算法为SeDuMi和SDPT3,其中SDPT3为默认算法.

- 调用sdpt3

```
cvx_solver sdpt3
```

- 调用SeDumi

cvx_solver sedumi

目前运算的结果比较, SeDuMi比SDPT3普遍较快. 如果在Matlab的程序文件.m中插入上述命令, 则只在当时的运行环境中调用指定的算法, 程序运算结束后恢复默认的SeDuMi指定算法.

CVX误差精度(tolerance)采用三个评价标准, 分别记为内部精度(solver tolerance)ϵ_{solver}, 标准精度(standard tolerance)$\epsilon_{standard}$和容忍精度(reduced tolerance)$\epsilon_{reduced}$, 满足关系$\epsilon_{solver} \leq \epsilon_{standard} \leq \epsilon_{reduced}$, 它们的标准为：内部精度为算法的内设精度；标准精度给出一个模型得以求解的临界值；容忍精度给出一个算法没有精确求解的临界值, 当不高于这个精度并高于标准精度时, CVX认为该模型没有精确求解, 当高于这个精度时, CVX报告计算失败的结果. 默认的三个精度值为$[\epsilon_{solver}, \epsilon_{standard}, \epsilon_{reduced}] = [\epsilon^{\frac{1}{2}}, \epsilon^{\frac{1}{2}}, \epsilon^{\frac{1}{4}}]$, 其中$\epsilon = 2.22 \times 10^{-16}$为机器精度. CVX设置了五个可以选择的精度标准, 分别为

- cvx_precision low: $[\epsilon^{\frac{3}{8}}, \epsilon^{\frac{1}{4}}, \epsilon^{\frac{1}{4}}]$.
- cvx_precision medium: $[\epsilon^{\frac{1}{2}}, \epsilon^{\frac{3}{8}}, \epsilon^{\frac{1}{4}}]$.
- cvx_precision default: $[\epsilon^{\frac{1}{2}}, \epsilon^{\frac{1}{2}}, \epsilon^{\frac{1}{4}}]$.
- cvx_precision high: $[\epsilon^{\frac{3}{4}}, \epsilon^{\frac{3}{4}}, \epsilon^{\frac{3}{8}}]$.
- cvx_precision best: $[0, \epsilon^{\frac{1}{2}}, \epsilon^{\frac{1}{4}}]$.

注意最佳精度中的第一项$\epsilon_{solver} = 0$, 表明只要没有达到0 这个精度, 则算法就一直算下去.

精度要求的实现通过在CVX内部或外部加入命令实现, 如

```
cvx_begin
cvx_precision high
...
cvx_end
```

表示在cvx_begin到cvx_end内部采用cvx_precision high精度. 若cvx_precision high在cvx_begin到cvx_end之外出现, 则cvx_precision high为全局精度.

当CVX程序正确无误后, 计算结果可能有以下六种情况输出, 分别罗列如下：

- Solved: 对偶互补的最优解得到, 存贮在CVX程序定义的变量中, 最优值为cvx_optval.
- Unbounded: 对于极小化目标函数的模型, 表明原问题沿一个方向无界, 这个方向通过原始问题的变量表出, 目标值cvx_optval 为-Inf. 对于极大化问题可以得到类似信息. 需要注意的是：无界情况下的无界方向通过原始问题的变量输出, 其不一定是原问题的可行解.
- Infeasible: 通过对偶问题的无界方向推出原问题无可行解. 输出的原始问题的变量值为NaNs, 极小化问题的目标值cvx_optval为+Inf, 极大化的目标值cvx_optval为-Inf.
- Inaccurate 有时因算法的设计和精度的问题, 无法肯定算例的计算结果, 因此出现：
  - Inaccurate/Solved: 算例可能有互补对偶解,
  - Inaccurate/Unbounded: 算例可能无界,
  - Inaccurate/Infeasible: 算例可能不可行.
- Failed: 算法无法得到满足要求的解, 此时变量和目标值都标识NaNs.
- Overdetermined: CVX的预处理发现算例的约束个数大于变量个数.

由于SeDumi和SDPT3都采用原始和对偶解信息, 通过原始和对偶解的可行和正交互补条件来判断是否达到最优解, 加之计算机本

身计算误差的问题, 使用者对于计算软件的输出结果不应无条件相信, 最好对计算结果加以验证和分析.

8 核函数库

Matlab本身提供函数的增加功能, 如何区别满足CVX的核函数与Matlab的函数？在读者自设的CVX程序实现中一定要注意函数内部的开始和结束语句一定用cvx_begin和cvx_end标识, 这样其中的语句一定满足CVX可计算的基本规则, 整个也就自然满足可计算的那些基本规则. 例如在上一节中, 我们知道Matlab命令

square(square(x)+1)

无法被CVX认为是一个凸函数, 现构造一个Matlab的函数

```
function cvx_optval =square_pos(x)
v=max(0, x);
cvx_optval =square(v);
```

则可将这个函数增加在CVX的核函数库中, 这时

square_pos(square(x)+1)

按复合函数的可计算规则是凸函数.