

# gpcvx

## A Matlab Solver for Geometric Programs in Convex Form

Kwangmoo Koh  
deneb1@stanford.edu

Seungjean Kim  
sjkim@stanford.edu

Almir Mutapcic  
almirm@stanford.edu

Stephen Boyd  
boyd@stanford.edu

September 26, 2005

**gpcvx** solves a geometric program (GP) using a phase I/phase II method. The phase I and the phase II solutions are found using **gppd2**, a primal-dual interior point method described in the book *Convex Optimization* [BV04].

## 1 The problem

The *log-sum-exp function* on  $\mathbf{R}^k$  is defined as

$$\text{lse}(x) = \log(e^{x_1} + \cdots + e^{x_k}), \quad (1)$$

where  $x = (x_1, \dots, x_k)$ . The *entropy function* on  $\mathbf{R}_+^k$  is defined as

$$\text{entr}(y) = - \sum_{i=1}^k y_i \log y_i, \quad (2)$$

where  $y = (y_1, \dots, y_k)$ . For both functions, the number of terms  $k$  is determined from context.

**gpcvx** solves an optimization problem of the form

$$\begin{aligned} & \text{minimize} && \text{lse}(A^{(0)}x + b^{(0)}) \\ & \text{subject to} && \text{lse}(A^{(i)}x + b^{(i)}) \leq 0, \quad i = 1, \dots, m, \\ & && Gx + h = 0, \\ & && l \preceq x \preceq u, \end{aligned} \quad (3)$$

with variable  $x \in \mathbf{R}^n$  and parameters  $A^{(i)} \in \mathbf{R}^{K_i \times n}$ ,  $b^{(i)} \in \mathbf{R}^{K_i}$  for  $i = 0, \dots, m$ ,  $G \in \mathbf{R}^{p \times n}$ ,  $h \in \mathbf{R}^p$ , and  $l, u \in \mathbf{R}^n$ . Here  $\preceq$  means componentwise inequality between vectors. We refer

to the problem (3) as a *geometric program in convex form*. For more information about geometric programming, see [BV04, BKVH].

To make it easier to derive the dual problem, we form a problem equivalent to (3). We introduce new variables  $y^{(i)} \in \mathbf{R}^{K_i}$ , as well as new equality constraints  $y^{(i)} = A^{(i)}x + b^{(i)}$  for  $i = 0, \dots, m$ . Then we can write the problem (3) as

$$\begin{aligned} & \text{minimize} && \text{lse}(y^{(0)}) \\ & \text{subject to} && \text{lse}(y^{(i)}) \leq 0, \quad i = 1, \dots, m \\ & && A^{(i)}x + b^{(i)} = y^{(i)}, \quad i = 0, \dots, m \\ & && Gx + h = 0 \\ & && l \preceq x \preceq u, \end{aligned} \tag{4}$$

The dual problem of (4) is

$$\begin{aligned} & \text{maximize} && \sum_{i=0}^m b^{(i)T} \nu^{(i)} + h^T \mu + \lambda_l^T l - \lambda_u^T u + \text{entr}(\nu^{(0)}) + \sum_{i=1}^m \lambda^{(i)} \text{entr}(\nu^{(i)} / \lambda^{(i)}) \\ & \text{subject to} && \lambda^{(i)} \geq 0, \quad i = 1, \dots, m, \\ & && \nu^{(i)} \succeq 0, \quad i = 0, \dots, m, \\ & && \mathbf{1}^T \nu^{(0)} = 1, \\ & && \mathbf{1}^T \nu^{(i)} = \lambda^{(i)}, \quad i = 1, \dots, m, \\ & && \sum_{i=0}^m A^{(i)T} \nu^{(i)} + G^T \mu + \lambda_u - \lambda_l = 0. \end{aligned} \tag{5}$$

with variables  $\mu \in \mathbf{R}^p$ ,  $\lambda_l, \lambda_u \in \mathbf{R}_+^n$ ,  $\lambda^{(i)} \in \mathbf{R}_+$  for  $i = 1, \dots, m$ , and  $\nu^{(i)} \in \mathbf{R}_+^{K_i}$  for  $i = 0, \dots, m$ .

## 2 Calling sequences

The complete calling sequence of `gpcvx` is

```
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,l,u,quiet);
```

Input arguments represent the problem data of (4). Output arguments are the optimal point (if feasible), the solution status, and the dual variables.

### 2.1 Input arguments

- **A**: matrix with  $n$  columns and  $\sum_{i=0}^m K_i$  rows that stacks  $A^{(0)}, \dots, A^{(m)}$  in (4), *i.e.*,

$$A = \begin{bmatrix} A^{(0)} \\ \vdots \\ A^{(m)} \end{bmatrix}.$$

**A** can be in sparse format.

- **b**: vector of length  $\sum_{i=0}^m K_i$  that stacks  $b^{(0)}, \dots, b^{(m)}$  in (4), *i.e.*,

$$b = \begin{bmatrix} b^{(0)} \\ \vdots \\ b^{(m)} \end{bmatrix}.$$

- **szs**: vector of length  $m + 1$  that specifies the number of exponential terms in each objective and inequality constraints, *i.e.*,  $(K_0, \dots, K_m)$ .
- **G**: matrix with  $n$  columns and  $p$  rows that specifies  $G$  in (4). **G** can be in sparse format.
- **h**:  $p$ -vector that specifies  $h$  in (4).
- **l**:  $n$ -vector that specifies lower bounds on  $x$ . If not given, it will be set to the default lower bounds  $(-250, \dots, -250)$ .
- **u**:  $n$ -vector that specifies upper bounds on  $x$ . If not given, it will be set to the default upper bounds  $(250, \dots, 250)$ .
- **quiet**: boolean. Suppresses print messages during execution if **true**. The default value is **false**.

## 2.2 Output arguments

- **x**:  $n$ -vector. **x** is the optimal point of the problem if the problem is feasible, and **x** is the last primal iterate of phase I if the problem is infeasible.
- **status**: string; possible values are 'Solved', 'Infeasible' and 'Failed'.
- **lambda**: vector of length  $m + 2n$ ; the dual variables associated with inequality constraints if the problem is feasible. The first  $m$  elements, **lambda**(1:m), are the dual variables of the  $m$  inequality constraints, the next  $n$  elements, **lambda**(m+1:m+n), are those of the lower bound constraints ( $l \preceq x$ ), and the last  $n$  elements, **lambda**(m+n:m+2\*n), are those of the upper bound constraints ( $x \preceq u$ ). If the problem is infeasible, **lambda** is a dual variable vector of phase I, which is a certificate of infeasibility (see [BKVH, §5.8.1, §11.4.3]).
- **nu**: vector of length  $\sum_{i=0}^m K_i$ ; the dual variables associated with equality constraints ( $A^{(i)}x + b^{(i)} = y^{(i)}$ ) if the problem is feasible. If infeasible, **nu** is a dual variable vector of phase I, which is a certificate of infeasibility (see [BKVH, §5.8.1, §11.4.3]).
- **mu**: vector of length  $p$ ; the dual variables associated with equality constraints ( $Gx + h = 0$ ) if the problem is feasible. If the problem is infeasible, **mu** is a dual variable vector of phase I, which is a certificate of infeasibility (see [BKVH, §5.8.1, §11.4.3]). **mu** is an empty matrix when there is no equality constraint.

## 2.3 Other calling sequences

Other calling sequences supported by `gpcvx` are:

```
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,l,u);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,[],[],l,u);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h,[],[],quiet);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,[],[],l,u,quiet);
>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,[],[],[],[],quiet);
```

## 2.4 Caveats

- The equality constraint matrix,  $G$ , must be full rank.
- If your problem is large and sparse, be sure that  $A$  and  $G$  are in sparse format.
- Equality constraints should be explicitly specified as  $Gx + h = 0$ . You cannot represent an equality constraint as a pair of opposing inequality constraints.

## 3 Example

Consider the problem

$$\begin{aligned}
 &\text{minimize} && z_1^{-1} z_2^{-1/2} z_3^{-1} + 2.3 z_1 z_3 + 4 z_1 z_2 z_3 \\
 &\text{subject to} && (1/3) z_1^{-2} z_2^{-2} + (4/3) z_2^{1/2} z_3^{-1} \leq 1, \\
 &&& 0.1 z_1 + 0.2 z_2 + 0.3 z_3 \leq 1, \\
 &&& (1/2) z_1 z_2 = 1,
 \end{aligned} \tag{6}$$

with variables  $z_1$ ,  $z_2$  and  $z_3$ . This is a GP in posynomial form. (If you want to solve a GP in posynomial form directly, use `gpposy`.) This problem can be converted into the convex form (3) by a change of variables ( $x_i = \log z_i$ ) and a transformation of the objective and constraint functions [BV04, BKHV]. Then, the problem (6) can be converted into

$$\begin{aligned}
 &\text{minimize} && \text{lse}(A^{(0)}x + b^{(0)}) \\
 &\text{subject to} && \text{lse}(A^{(1)}x + b^{(1)}) \leq 0, \\
 &&& \text{lse}(A^{(2)}x + b^{(2)}) \leq 0, \\
 &&& Gx + h = 0, \\
 &&& l \preceq x \preceq u,
 \end{aligned} \tag{7}$$

with variable  $x = (x_1, x_2, x_3)$ . The problem data are

$$A^{(0)} = \begin{bmatrix} -1 & -0.5 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad A^{(1)} = \begin{bmatrix} -2 & -2 & 0 \\ 0 & 0.5 & -1 \end{bmatrix}, \quad A^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$b^{(0)} = \begin{bmatrix} \log(1) \\ \log(2.3) \\ \log(4) \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} \log(1/3) \\ \log(4/3) \end{bmatrix}, \quad b^{(2)} = \begin{bmatrix} \log(0.1) \\ \log(0.2) \\ \log(0.3) \end{bmatrix},$$

$$G = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \quad h = \log(0.5).$$

The Matlab code for solving this problem is as follows:

```
% Matlab script that solves the above problem

>> A0 = [ -1 -0.5 -1 ;...
          1  0  1 ;...
          1  1  1 ];
>> A1 = [ -2 -2 0 ;...
          0 0.5 -1 ];
>> A2 = [ 1 0 0 ;...
          0 1 0 ;...
          0 0 1 ];
>> A = [ A0; A1; A2 ];

>> b0 = log([ 1; 2.3; 4 ]);
>> b1 = log([ 1/3; 4/3 ]);
>> b2 = log([ 0.1; 0.2; 0.3 ]);
>> b = [ b0; b1; b2 ];

>> G = [ 1 1 0 ];
>> h = log(0.5);
>> szs = [ size(A0,1); size(A1,1); size(A2,1) ]; %i.e., [ 3; 2; 3 ]

>> [x,status,lambda,nu,mu] = gpcvx(A,b,szs,G,h);
```

After executing the code, you can see the result by typing `x` in Matlab.

```
>> x

ans =

    1.2465
   -0.5534
    0.0980
```

## 4 Performance

`gpcvx` is not optimized for performance, but shows reasonable speed. The following table shows the execution times of `gpcvx` on some typical problems. These times given are for a

2.8GHz Pentium 4, 512Mb RAM, Linux operating system.

Problem	$n$	$m$	$w$	$p$	nnz	Execution time
test1	100	100	5	50	5000	2sec
test2	1000	1000	5	0	50000	36sec
test3	1000	10000	5	0	500000	171sec
test4	100	1000	50	50	500000	53sec
test5	1000	1000	50	0	500000	138sec

Here,  $n$  is the number of variables,  $m$  is the number of inequality constraints,  $w$  is the number of terms (summands) per inequality constraint, and  $p$  is the number of equality constraints. The matrices **A** and **G** are sparse; **nnz** is the number of nonzero elements in the **A** matrix. The code that runs these experiments, as well as the test problems, can be found in the `examples_gpsolver` directory.

## References

- [BKVH] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. To appear in *Optimization and Engineering, 2005*. Available at [www.stanford.edu/~boyd/gp\\_tutorial.html](http://www.stanford.edu/~boyd/gp_tutorial.html).
- [BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Available at [www.stanford.edu/~boyd/cvxbook.html](http://www.stanford.edu/~boyd/cvxbook.html).