# Some Pictures and Scripts for Teaching IPMs

Michal Černý, Irena Šindelářová[1]

**Abstract.** In the text we discuss some geometric aspects of linear programming and provide a visualisation tool for a type of a short-step central-path following algorithm. The tool is intended to help in teaching interior point algorithms.

**Keywords.** linear programming, central path algorithms, short-step algorithms, Khachiyan's grid, visualisation

The nice property of linear programming, contrary to many other fields of mathematics, is that its nature is essentially geometric. Therefore, such important terms as polyhedron, analytic center, central path etc. may be quite easily visualised when their definitions are taught. Moreover, one can often visualise also the ideas underlying particular important theorems: for instance, the duality theorem—probably the most important result in theory of linear programming—has an intuitive geometric, sometimes called "billiard", meaning (see [8], p. 93).

A lot of ideas underlying LP algoritms are of quite geometric nature, too. An example of a wonderful and easy geometric idea is the well-known Shor-Yudin-Nemirovskij's ellipsoid method. Even the Khachiyan's algortihm may be visualised: it may be regarded as the ellipsoid method where all "important" points, say vertices of the bounded full-dimensional rational polyhedron (if nonempty) and centres of the ellipsoids, lay on a special grid: a grid of finitely many rational points with bounded bit-size. Rounding of coefficients of matrices describing the ellipsoids may be also geometrically demonstrated: if the ellipsoid is coordinate-aligned and centered at zero, rounding eigenvalues to a restricted bit-size number may be seen as shortening semi-axes so that their lengths (and also squares of their lengths) are points on the Khachyian's grid. Hence, in some sense Khachyian's algorithm may be visualised as a discrete version of the ellipsoid method.

To give one more example: a well-known nice geometric idea is the Karmarkar's affine transformation, nicely described e.g. in [7]. Interesting geometric aspects of the Karmarkar potential are given in [9].

[1]Both authors: University of Economics Prague, Dept of Econometrics, Nám. Winstona Churchilla 4, CZ-130 67 Prague, Czech Republic; cernym@vse.cz, isin@vse.cz. Supported by Czech Science Foundation grant No. 402/06/0150.

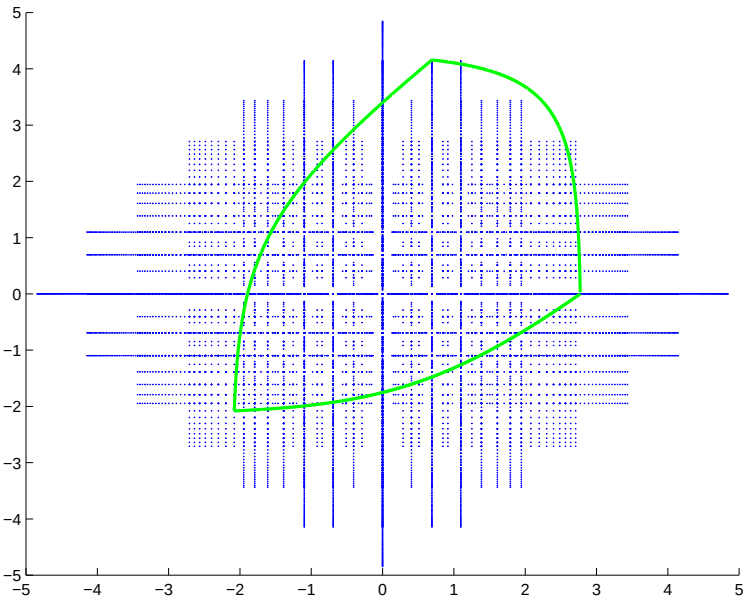


**Fig. 1.** Khachiyan's grid: points $[\frac{a}{b}, \frac{c}{d}] \in \mathbb{R}^2$ with integral $a, b, c, d > 0$ such that sum of bit sizes of $a, b, c$ and $d$ does not exceed 10 bits (10 bits are taken just as an example) and a polyhedron with vertices on the grid—a triangle in $\mathbb{R}^2$ with vertices $[\frac{1}{8}, \frac{1}{8}], [2, 64]$ and $[16, 1]$. Both axes are log-scaled.

The aim of this text is to provide teachers of linear programming with a simple visualisation tool of a version of a central-path algorithm which will be sketched later; it is a nice representative of the entire family of central-path short-step algorithms and is suitable to be presented in lectures. We choose this algorithm as (i) it is one of the algorithms with so far best-known complexity (its iteration bound is $O(\sqrt{n}L)$, where $n$ denotes dimension and $L$ is the bit-size of the linear program); (ii) its idea is nicely explained in the excellent book [4]. The book does not give full theoretical analysis, however the idea behind the algorithm is shown in a clear way which is suitable for many undergraduate linear-programming courses (nevertheless, a full analysis exceeds the level of such lectures; an almost-full analysis is found in [7]).

**The algorithm.** Assume we are given a full-rank matrix $\boldsymbol{A}$ and vectors $\boldsymbol{b}, \boldsymbol{c}$ (all vectors are column) and our task is either to find $\boldsymbol{x}^* \in \operatorname{argmax}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ (for our purposes, an approximate optimum is sufficient) if such exists or state that no such exists. (In theory, one usually works with rational arithmetic; here we omit the arithmetic-precision considerations. Our task is just to give a didactic tool to plot some pictures, so we may admit that $10^{-10} = 0$ or so.) Let us be given a small positive number $\epsilon$ which will govern the exactness of the algorithm.

For the purpose of visualisation we will assume the dimension $n = 2$ and $\mathcal{F} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ is nonempty, bounded and has dimension two. (This means

the the optimum exists.) However, we describe the algorithm in general. Let $m$ be the number of rows of $\boldsymbol{A}$.

First, we construct a type of a self-dual embedding of the linear program $\operatorname{argmax}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$. Let

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{A} & -\boldsymbol{b} \\ -\boldsymbol{A}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{c} \\ \boldsymbol{b}^{\mathrm{T}} & -\boldsymbol{c}^{\mathrm{T}} & 0 \end{pmatrix}, \quad \boldsymbol{d} = \boldsymbol{1} + \boldsymbol{D}\boldsymbol{1}, \quad \boldsymbol{E} = \begin{pmatrix} \boldsymbol{D} & -\boldsymbol{d} \\ \boldsymbol{d}^{\mathrm{T}} & 0 \end{pmatrix} \tag{1}$$

Now set

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \tau \\ \eta \end{pmatrix}, \quad \boldsymbol{e} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ n+m+2 \end{pmatrix}. \tag{2}$$

Here, $\boldsymbol{q}$ is a vector of variables: $\boldsymbol{x}$ has $n = 2$ components and $\boldsymbol{y}$ has $m$ components. Now the following holds:

(i) the linear program $\operatorname{argmax}\{-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{q} : \boldsymbol{E}\boldsymbol{q} \leq \boldsymbol{e}, \boldsymbol{q} \geq \boldsymbol{0}\}$ is bounded and feasible and for every optimal solution $(\boldsymbol{q}^*)^{\mathrm{T}} = ((\boldsymbol{y}^*)^{\mathrm{T}}, (\boldsymbol{x}^*)^{\mathrm{T}}, \tau^*, \eta^*)$, $\boldsymbol{x}^*, \boldsymbol{y}^*$ and $\tau^*$ form a solution to the Goldman-Tucker system

$$\boldsymbol{A}\boldsymbol{x} - \tau\boldsymbol{b} \leq \boldsymbol{0},\ -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \tau\boldsymbol{c} \leq \boldsymbol{0},\ \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq \boldsymbol{0},\ \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{y} \geq \boldsymbol{0},\ \tau \geq 0;$$

(ii) every strictly complementary optimal solution of $\operatorname{argmax}\{-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{q} : \boldsymbol{E}\boldsymbol{q} \leq \boldsymbol{e}, \boldsymbol{q} \geq \boldsymbol{0}\}$ is a solution of the Goldman-Tucker system with either $\tau > 0$ or $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} < 0$,

(iii) $\boldsymbol{q} = \boldsymbol{1}$, $\boldsymbol{p} = \boldsymbol{1}$ are feasible for the system $\boldsymbol{E}\boldsymbol{q} + \boldsymbol{p} = \boldsymbol{e}$, $\boldsymbol{q} > \boldsymbol{0}$, $\boldsymbol{p} > \boldsymbol{0}$.

By the Goldman-Tucker theorem, either there exists a solution to the Goldman-Tucker system with $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} < 0$, and then the real optimum of the original program $\operatorname{argmax}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ does not exist, or there exists a solution with $\tau > 0$, and then $\frac{1}{\tau}\boldsymbol{x}$ is optimum of the original linear program (and, moreover, $\frac{1}{\tau}\boldsymbol{y}$ is optimum of its dual program).

So now we have a system $\boldsymbol{E}\boldsymbol{q} + \boldsymbol{p} = \boldsymbol{e}$, $\boldsymbol{q} \geq \boldsymbol{0}$, $\boldsymbol{p} \geq \boldsymbol{0}$ where we know an interior feasible point in advance; moreover it is a "good" starting point for the algorithm.

To get an optimum of the original system (or conclude that it does not exist) we need to find a strictly complementary solution of another system. This is the essence of the algorithm: it gets the linear program

$$\operatorname{argmax}\{-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{q} : \boldsymbol{E}\boldsymbol{q} + \boldsymbol{p} = \boldsymbol{e}, \boldsymbol{q} \geq \boldsymbol{0}, \boldsymbol{p} \geq \boldsymbol{0}\} \tag{3}$$

as its input and will converge to a such a solution.

At the beginning, we set $\mu := 1$ and exponentially fast decreasing $\mu$ we will approximately follow the central path $\{\text{argmax}\{(\boldsymbol{q},\boldsymbol{p}) : \mu \sum_{i=1}^{m+n+2}(\ln \boldsymbol{p}_i + \ln \boldsymbol{q}_i) - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{q}\}; \mu \in (0,1]\}$. To (approximately) follow the central path (keeping within the quadratic-convergence region), we have to start in a "good" point: indeed, the initial point $\boldsymbol{p} = \boldsymbol{1}$, $\boldsymbol{q} = \boldsymbol{1}$ is a point on the central path with $\mu = 1$.

Given $\mu$, the point on the central path $\text{argmax}\{(\boldsymbol{q},\boldsymbol{p}) : \mu \sum_{i=1}^{m+n+2}(\ln \boldsymbol{p}_i + \ln \boldsymbol{q}_i) - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{q}\}$ may be written as a solution to a non-linear system $\boldsymbol{E}\boldsymbol{q} + \boldsymbol{p} = \boldsymbol{e}$, $\boldsymbol{q}_i \cdot \boldsymbol{p}_i = \mu$ for all $i = 1, 2, \ldots, n+m+2$, $\boldsymbol{p} \geq \boldsymbol{0}$, $\boldsymbol{q} \geq \boldsymbol{0}$. Given an approximate point near the central path $(\boldsymbol{q},\boldsymbol{p})$, we want to update $\boldsymbol{q} := \boldsymbol{q} + \boldsymbol{\alpha}$, $\boldsymbol{p} := \boldsymbol{p} + \boldsymbol{\beta}$ by performing a Newton-like step: substituting this into the system, we get $\boldsymbol{E}(\boldsymbol{q}+\boldsymbol{\alpha})+(\boldsymbol{p}+\boldsymbol{\beta}) = \boldsymbol{e}$, $(\boldsymbol{q}_i + \boldsymbol{\alpha}_i) \cdot (\boldsymbol{p}_i + \boldsymbol{\beta}_i) = \mu$ for all $i$'s, $(\boldsymbol{q} + \boldsymbol{\alpha}) \geq \boldsymbol{0}$, $(\boldsymbol{p} + \boldsymbol{\beta}) \geq \boldsymbol{0}$ and we neglect the second-order terms $\boldsymbol{\beta}_i \cdot \boldsymbol{\alpha}_i$. Hence, we solve the linear system

$$\begin{pmatrix} \boldsymbol{E} & \boldsymbol{J} \\ \text{diag}(\boldsymbol{q}) & \text{diag}(\boldsymbol{p}) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \mu \cdot \boldsymbol{1} - \boldsymbol{q} * \boldsymbol{p} \end{pmatrix} \tag{4}$$

($\boldsymbol{J}$ stands for unit matrix, $\text{diag}(\boldsymbol{q})$ for a diagonal matrix with entries of $\boldsymbol{q}$ on the diagonal and $\boldsymbol{q} * \boldsymbol{p} = (\boldsymbol{q}_1 \cdot \boldsymbol{p}_1, \boldsymbol{q}_2 \cdot \boldsymbol{p}_2, \ldots, \boldsymbol{q}_{n+m+2} \cdot \boldsymbol{p}_{n+m+2})^{\mathrm{T}}$) where $\boldsymbol{q},\boldsymbol{p}$ act as constants and $\boldsymbol{\alpha},\boldsymbol{\beta}$ as unknowns. It may be shown that such an „approximate" step from $(\boldsymbol{q},\boldsymbol{p})$ to $(\boldsymbol{q}+\boldsymbol{\alpha},\boldsymbol{p}+\boldsymbol{\beta})$ fulfills the constraints $\boldsymbol{q}+\boldsymbol{\alpha} > \boldsymbol{0}, \boldsymbol{p}+\boldsymbol{\beta} > \boldsymbol{0}$ and does not deviate from the exact central path too far. (With some further analysis, this implies that the procedure converges.) Now we may summarize the algorithm. Of course, we are not correct… To be fully correct, we would have to compute in rational arithmetic and perform further analysis, when it is appropriate to state that no optimum exists, and also determine the error of the approximate solution (or perform the usual final "$2^{-2L}$-truncation" step to get an exact optimum). We will not do this here; for purposes of visualisation, we assume the optimum exists, and we do not need to estimate the error in detail.

| | |
|---|---|
| input: | $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \epsilon$ |
| output: | approximate $\boldsymbol{x} \in \text{argmax}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ or report `optimum does not exist` |
| [1] | Construct $\boldsymbol{E}$ as in (1) and set $\mu := 1$, $\boldsymbol{q} := \boldsymbol{1}$, $\boldsymbol{p} := \boldsymbol{1}$; |
| [2] | if $\mu < \epsilon$ then: look at $\boldsymbol{q}^{\mathrm{T}}$ as $(\boldsymbol{y}^{\mathrm{T}}, \boldsymbol{x}^{\mathrm{T}}, \tau, \eta)$; if $\tau \approx 0$, report `optimum does not exist`; otherwise return $\frac{1}{\tau}\boldsymbol{x}$ as the approximate solution. Stop. |
| [3] | Set $\mu := (1 - \frac{1}{2\sqrt{n}})\mu$; |
| [4] | find $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by solving the linear system (4) with current $\boldsymbol{q}$ and $\boldsymbol{p}$, update $\boldsymbol{q} := \boldsymbol{q} + \boldsymbol{\alpha}$, $\boldsymbol{p} := \boldsymbol{p} + \boldsymbol{\beta}$ and go to [2]. |

Now it would be necessary to show that the procedure converges to the strictly complementary solution of (3); moreover, it holds that if an optimum exists, then the algorithm converges to the analytic centre of the optimal face.

We will plot $\mathcal{F} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ and the trajectory of the algorithm: to plot the trajectory, in each step, we look at current $\boldsymbol{q}^{\mathrm{T}}$ as $(\boldsymbol{y}^{\mathrm{T}}, \boldsymbol{x}^{\mathrm{T}}, \tau, \eta)$ as in step [2] and we plot the point $\frac{1}{\tau}\boldsymbol{x}$.

However, the algorithm in fact does not pass through $\mathcal{F}$ but a more-dimensional polyhedron

$$\{\boldsymbol{q} : \boldsymbol{E}\boldsymbol{q} \leq \boldsymbol{e}, \boldsymbol{q} \geq \boldsymbol{0}\}. \tag{5}$$

So it is instructive, in each step, to plot the projection of this polyhedron into the plane, too.

Given a definition $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}$ of a nonempty two-dimensional polyhedron, to visualise the analytic centre it suffices to run the algorithm with $\boldsymbol{c} = \boldsymbol{0}$; it is instructive to see how the analytic centre moves if we, for instance, add some redundant inequalities into the defining system (e.g., some inequalities are present more than once).

At the site `http://nb.vse.cz/~cernym/ipm.zip`, some MatLab scripts plotting the pictures, that may help in teaching, are available. The function

`ip(`$\boldsymbol{A}$`,`$\boldsymbol{b}$`,`$\boldsymbol{c}$`,mode)`

gets an $m \times 2$ matrix $\boldsymbol{A}$, a column vector $\boldsymbol{b}$ with $m$ entries and a column vector $\boldsymbol{c}$ with two entries such that the feasible region $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ is two-dimensional, nonempty and bounded and plots the trajectory of the described algorithm. If `mode` $= 1$, then the projections of (5) into $\mathbb{R}^2$ are in each step plotted too.

The function `getmatrix(`$\boldsymbol{x}$`,`$\boldsymbol{y}$`)`, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are column vectors with $l$ entries, first computes the convex hull $\mathcal{C}$ of the points $[\boldsymbol{x}_1, \boldsymbol{y}_1], \ldots, [\boldsymbol{x}_l, \boldsymbol{y}_l]$ and returns a matrix $(\boldsymbol{A} \;\; \boldsymbol{b})$ such that $\mathcal{C} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$. For instance, the command

```
Ab =  getmatrix([2 + cos([0:2*pi./7:2*pi])]',
      [3 + sin( [0:2*pi./7:2*pi])]')
```

forms a system of inequalities which define a regular septahedron centered at $[2, 3]$. Now calling

```
ip(Ab(:,1:2),Ab(:,3),[0 1]',0)
```

and

```
ip(Ab(:,1:2),Ab(:,3),[0 1]',1)
```

yields the following pictures.

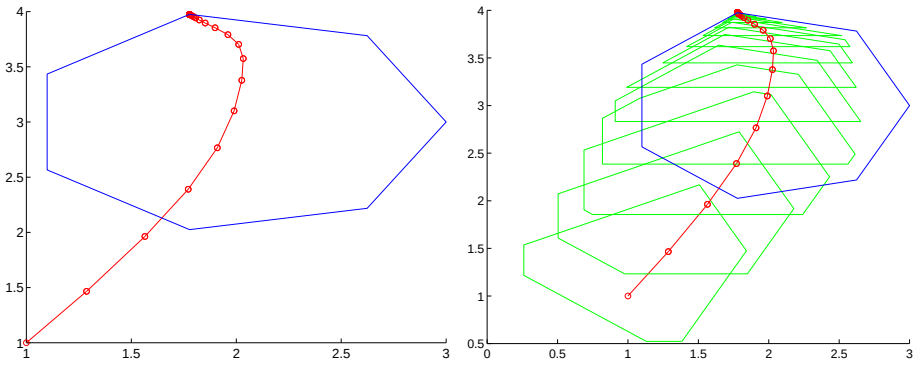

**Fig. 2.** Trajectory of the algorithm.

Calling `ip(Ab(:,1:2),Ab(:,3),[-1 0]',` *mode*`)` (again with either *mode*=`0` or *mode*=`1`) shows how the algorithm converges to the analytic centre of the optimal face (which is the vertical edge of the septahedron). Also note how the algorithm follows the central path $\{\text{argmax}\{(\boldsymbol{q},\boldsymbol{p}) : \mu \sum_{i=1}^{m+n+2}(\ln \boldsymbol{p}_i + \ln \boldsymbol{q}_i) - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{q}\}; \mu \in (0,1]\}$: first the direction is governed by the terms $\ln \boldsymbol{p}_i + \ln \boldsymbol{q}_i$ and then by $-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{q}$.

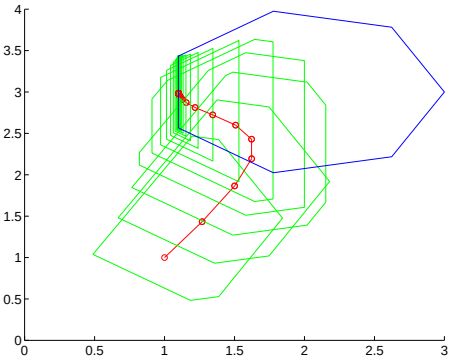

**Fig. 3.** Convergence to the analytic centre of the optimal face.

The function `ip2(`$\boldsymbol{A}_1$`,`$\boldsymbol{b}_1$`,`$\boldsymbol{c}_1$`,`$\boldsymbol{A}_2$`,`$\boldsymbol{b}_2$`,`$\boldsymbol{c}_2$`, mode)` plots two trajectories of two linear programs. It is, for instance, instructive to see how the analytic centre moves if we change the definition of the same polyhedron. For example, first generate a triangle

```
Ab =  getmatrix([2 + cos([0:2*pi./3:2*pi])]',
      [3 + sin( [0:2*pi./3:2*pi])]')
```

and call

```
ip2(Ab(:,1:2),Ab(:,3),[0 0]',
   [Ab(:,1:2); Ab(1, 1:2)], [Ab(:,3); Ab(1,3)], [0, 0]', 0)
```

The system $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$ defines the triangle with three inequalities; the system $\boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2$ with four inequalities: the fourth is a redundant copy of the first one. (Note that if $\boldsymbol{c} = \boldsymbol{0}$ then the optimal face is the entire polyhedron and hence the algorithm converges to its analytic centre.)

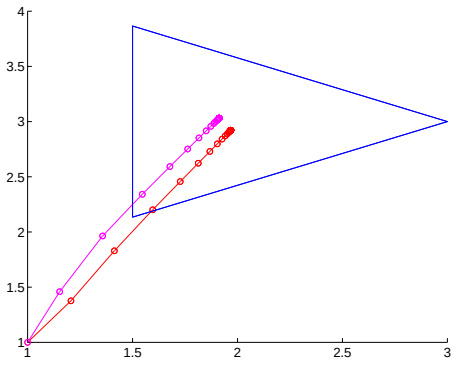


**Fig. 4.** Convergence to analytic centres of the same polyhedron with two different definitions.

As the algorithm converges to the analytic centre of the optimal region, it is suitable for $L_1$-regression. It is well-known that given points $[\boldsymbol{x}_i^{\mathrm{T}}; y_i]$, $i = 1, 2, \ldots, N$, finding $\widehat{\boldsymbol{\beta}}_{L_1} \in \operatorname{argmin}_{\boldsymbol{\beta}}\{\sum_{i=1}^{N} |y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_i|\}$ is equivalent to the linear program $\operatorname{argmax}_{(\boldsymbol{\beta},\boldsymbol{r})}\{-\boldsymbol{1}^{\mathrm{T}}\boldsymbol{r} : \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{r} \geq \boldsymbol{y}, \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{r} \leq \boldsymbol{y}, \boldsymbol{r} \geq \boldsymbol{0}\}$, where $\boldsymbol{X}$ is a matrix with rows of $\boldsymbol{x}_i^{\mathrm{T}}$'s and $\boldsymbol{y}$ is a vector of $y_i$'s. (Our algorithm requires variables to be nonnegative, hence we have to use the usual trick with $\boldsymbol{\beta} \equiv \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$; $\boldsymbol{\beta}^+, \boldsymbol{\beta}^- \geq \boldsymbol{0}$ and as we start the algorithm with $\boldsymbol{\beta}^+ = \boldsymbol{\beta}^- = \boldsymbol{1}$, the initial $\boldsymbol{\beta}$ is zero.) The optimum of the linear program need not be unique; then there is a question which solution shall be taken as $\widehat{\boldsymbol{\beta}}_{L_1}$. The analytic centre of the optimal region is a reasonable choice (as it is the "farthest-from-all-borders" point); unlike the simplex method, with our algorithm we get it free of charge.

The script `L1reg(degree,`$\boldsymbol{x}$`,`$\boldsymbol{y}$`)` gets two $N$-component column vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and fits the $L_1$-regression polynomial od degree `degree`. The pictures show how fast the regression function converges from $\boldsymbol{\beta}_{\mathrm{start}} = \boldsymbol{0}$ to the optimum. Just for illustration, the usual $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$-estimate is shown with a dashed line. For instance, generate some random data with `x = [0:0.1:3]'`,

```
y =  1 + 1.25.*x - 1.5.*x.^2 + 0.3.*x.^3 +
     random('norm', 0, 0.2, size(x)).
```

Then, calling `L1reg(1,x,y)` and `L1reg(3,x,y)` yields the following pictures.

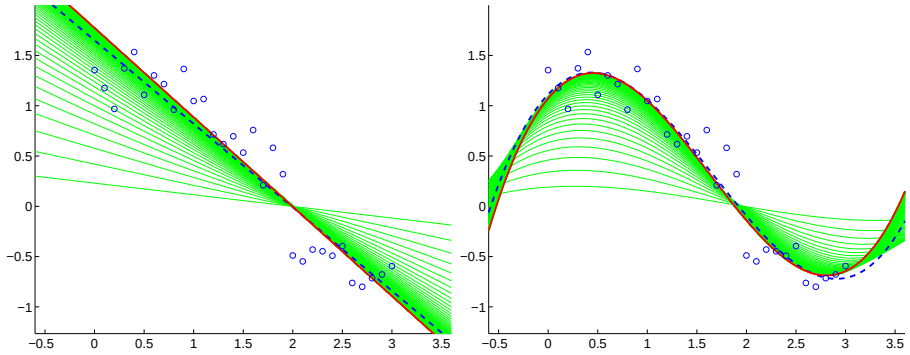


**Fig. 5.** Convergence to $L_1$-regression polynomials of degree 1 and 3 (compared to $L_2$-regression (dashed)).

The script also produces a picture of evolution of the regression coefficients over time (i.e. values of $\boldsymbol{\beta}$ in $k$-th iteration, where $k$ is on the x-axis). For instance, if we fit a polynomial of degree 9 to the data described, we get the following picture. (We stop if $\mu < 10^{-9}$.)

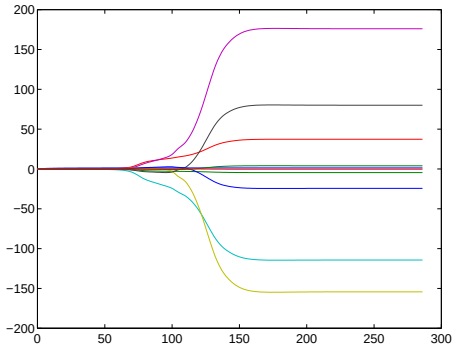


**Fig. 6.** Evolution of the regression coefficients during computation.

Now say we are fitting a line, hence $\boldsymbol{\beta}$ has two components. Fix the found (nearly) optimal residuals $\boldsymbol{r}$ and perform a slight $\delta$-relaxation of the polyhedron: we get the system $\boldsymbol{X\beta} + \delta\mathbf{1} \geq \boldsymbol{y} - \boldsymbol{r}, \boldsymbol{X\beta} - \delta\mathbf{1} \leq \boldsymbol{y} + \boldsymbol{r}$ in three variables, $\boldsymbol{\beta}_1$ (intercept), $\boldsymbol{\beta}_2$ (slope) and $\delta$. Now, given $\delta$, we may plot a "$\delta$-confidence polyhedron" which shows how $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ may change such that the sum of residuals does not increase more that $\delta N$. This is done by the script `confp(`$\boldsymbol{x}$`,`$\boldsymbol{y}$`,`$\delta$`)`, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are as before. If $\delta$ is a column vector, then for each component, one polyhedron is plotted. For our problem, calling `confp(x, y, [0:.01:.1]')` we get the following picture.

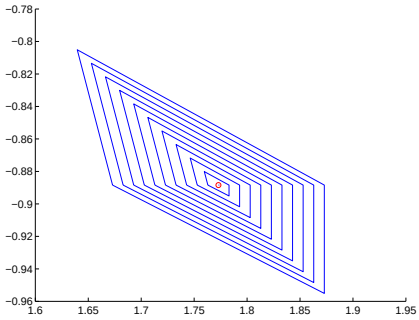


**Fig. 7.** Example of "$\delta$-confidence" polyhedra.

There are more examples of linear programming problems that nicely show the convergence of the interior-point algorithm. Our last example concerns the problem of inscribing the largest circle into a polyhedron defined by a set of inequalities $\boldsymbol{Ax} \leq \boldsymbol{b}$. This issue is easily solved by a linear program: maximize $r$ subject to the following constraints: (i) if a line $y = ax+b$ bounds the polyhedron from below, add a constraint $s_y - as_x - \sqrt{a^2+1} \cdot r \geq b$; (ii) if a line $y = ax + b$ bounds the polyhedron from above, add a constraint $s_y - as_x + \sqrt{a^2+1} \cdot r \leq b$; (iii) if a line $x = b$ bounds the polyhedron from left, add a constraint $s_x - r \geq b$, (iv) if a line $x = b$ bounds the polyhedron from right, add a constraint $s_x + r \leq b$, (v) $r \geq 0$. $s_x$ and $s_y$ are variables for the centre of the inscribed circle and $r$ its diameter. (For our algorithm, we need five nonnegative variables: $r$, $s_x^+$, $s_x^-$, $s_y^+$, $s_y^-$: $s_x \equiv s_x^+ - s_x^-$, $s_y \equiv s_y^+ - s_y^-$.)

The script `circ(`$\boldsymbol{A}$`,`$\boldsymbol{b}$`)` plots how the algorithm approximates the largest circle inscribed into the nonempty polyhedron $\{\boldsymbol{x} : \boldsymbol{Ax} \leq \boldsymbol{b}\}$. For instance, generating a septahedron with the command

```
Ab =  getmatrix([4 + cos ([0:2*pi./7:2*pi])]',
      [3 + .5.*sin([0:2*pi./7:2*pi])]')
```

and calling `circ(Ab(:,1:2), Ab(:,3))` will yield the last picture of this article.

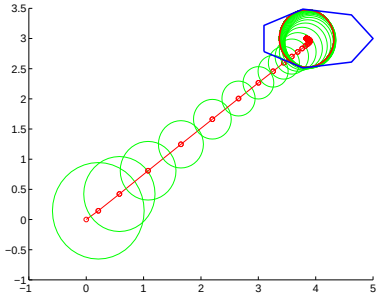


**Fig. 8.** Convergence to the largest circle inscribed into a polyhedron.

# References

[1] Coppersmith D. — Winograd S. (1990), *Matrix Multiplication via Arithmetic Progressions.* Journal of Symbolic Computation 9, pp. 251–280.

[2] Černý, M., *Linear programming is in* **P** (in Czech, 2007). Internet: `http://nb.vse.cz/~cernym/lpinp.pdf`.

[3] den Hertog, D. (1994), *Interior Point Approach to Linear, Quadratic and Convex Programming.* Mathematics and Its Applications vol. 277, Kluwer Academic Publishers.

[4] Matoušek J. — Gärtner B. (2007), *Understanding and Using Linear Programming.* Springer Verlag, Berlin.

[5] Nazareth, J. L. (2003), *Differentiable Optimization and Equation Solving: A Treatise on Algorithmic Science and the Karmarkar Revolution.* Springer Verlag, Berlin.

[6] Regenar, J. (2001), *A Mathematical View of Interior-Point Methods in Convex Optimization.* MPS/SIAM Series on Optimization, Philadelphia.

[7] Roos C. — Terlaky T. — Vial J.-P. (2006), *Interior Point Methods for Linear Optimization.* Springer Verlag, Berlin.

[8] Schrijver A. (1998), *Theory of Linear and Integer Programming.* Wiley and sons, New York.

[9] Ye, Y. (1997), *Interior Point Algorithms: Theory and Analysis.* Wiley and sons, New York.