

THE LP PROBLEM IN STANDARD FORM

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c'x, \\ & Ax = b, \quad x \geq 0. \end{aligned}$$

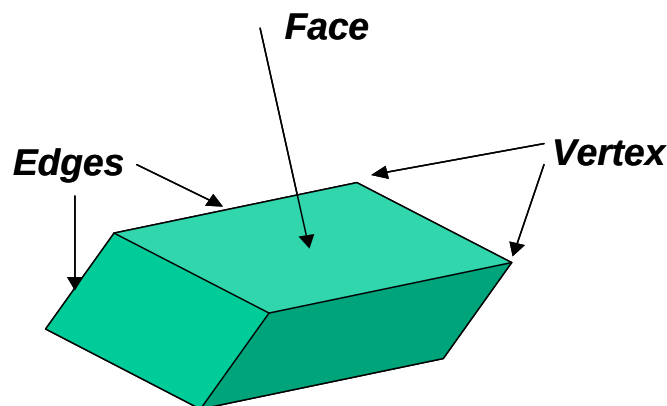
- $x \geq 0$ means $x_i \geq 0$, $i = 1, \dots, n$.
- A of size $r \times n$ is supposed to have full rank r .
- Ω is a **polytope** (**polyhedron** if bounded).
- This is a *convex* optimization problem $\Rightarrow$ KKT conditions sufficient for a global minimum.

GEOMETRY OF THE FEASIBLE SET

Definition: The point $x_e \in \partial \Omega$ is an *extreme point*, or a *vertex*, if

$$x_e = \theta y + (1 - \theta) z, \quad y, z \in \Omega, \quad 0 < \theta < 1$$

implies that $y = z = x_e$.



Definition: A feasible point x satisfying $Ax = b$ is called a *basic point* if there is an index set $\mathcal{B} = \{i_1, \dots, i_r\}$, where $\{a_{i_1}, \dots, a_{i_r}\}$ are linearly independent and $x_i = 0$ for all $i \notin \mathcal{B}$.

If x_i happens to be 0 also for some $i \in \mathcal{B}$, we say that the basic point is *degenerate*.

For a basic point, the corresponding $r \times r$ matrix

$$B = [a_{i_1}, \dots, a_{i_r}],$$

will be *non-singular*, and the equation $Bx_B = b$ has a unique solution.

The Fundamental Theorem for LP (N&W Theorem 13.2):

1. *If $\Omega \neq \emptyset$, it contains basic points.*
2. *If there are optimal solutions, there are optimal basic points.*

Theorem (N&W Theorem 13.3): *The basic points are the extreme points of Ω .*

The number of basic points is somewhere between 1 (since the rank of A is r) and $\binom{n}{r}$

THE SIMPLEX ALGORITHM

- The *Simplex Algorithm* is reported to have been discovered by G. B. Dantzig in 1947.
- The idea of the Simplex Algorithm is to search for the minimum by going from vertex to vertex (from basic point to basic point) in Ω .
- Hand calculations are never used anymore!

The Simplex Iterative Step

We assume that the problem has the standard form and that we are located in a basic point,

$$x = \begin{bmatrix} x_B \\ 0 \end{bmatrix},$$

where the partition is according to $A = [B \ N]$, B non-singular, and

$$Ax = [B \ N] \begin{bmatrix} x_B \\ 0 \end{bmatrix} = Bx_B = b.$$

Split x in the same way,

$$A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Bx_1 + Nx_2 = b.$$

Hence,

$$x_1 = B^{-1} (b - Nx_2) = x_B - B^{-1}Nx_2.$$

Note also that

$$\begin{aligned} f(x) &= c'x = [c_1 \ c_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= c'_1x_1 + c'_2x_2 \\ &= c'_1(x_B - B^{-1}Nx_2) + c'_2x_2 \\ &= c'_1x_B + (c'_2 - c'_1B^{-1}N)x_2 \end{aligned}$$

Around $(x_B \ 0)$, we may express both x_1 and f in terms of x_2 !

We are located at $x_1 = x_B$, $x_2 = 0$, and try to change one of the components $(x_2)_j$ of x_2 so that

$$f(x) = c'_1 x_B + (c'_2 - c'_1 B^{-1} N) x_2$$

decreases.

- If $(c'_2 - c'_1 B^{-1} N) > 0 \Rightarrow$ **FINISHED!**

Assume that $(c'_2 - c'_1 B^{-1} N)_j < 0$:

- Assume that all components of x_1 increase when $(x_2)_j$ increases, then

$$\min c'x = -\infty.$$

$\Rightarrow$ **FINISHED!**

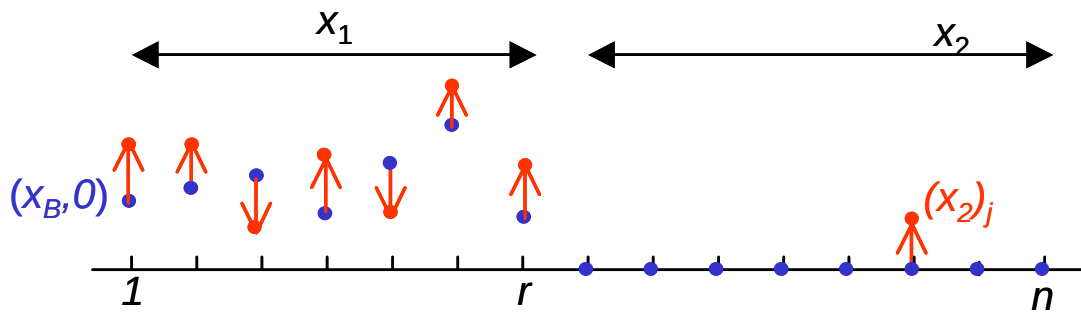


Figure 1:

- The Simplex algorithm always converges if all basic points are non-degenerate.
- Degenerate basic point: Try a different component of x_2 !
- It is straightforward to construct a generalized Simplex Algorithm for bounds of the form

$$l_i \leq x_i \leq u_i, \quad i = 1, \dots, n.$$

- If we LU -factorize B once, we can update the factorization with the new column without making a complete new factorization (N&W, Sec. 13.4).

- It is often preferable to take the "steepest ridge" (fastest decrease in the objective) out from where we are (N&W, Sec. 13.5).

Starting the Simplex method

The Simplex method consists of two phases:

- Phase 1: Find a first basic point
- Phase 2: Solve the original problem.

The Phase 1 algorithm:

1. Turn the signs in $Ax = b$ so that $b \geq 0$.

2. Introduce additional variables $y \in \mathbb{R}^r$ and solve the extended problem

$$\min (y_1 + \cdots + y_r),$$
$$[A \quad I] \begin{bmatrix} x \\ y \end{bmatrix} = b, \quad x, y \geq 0.$$

(Note that $(0 \ b)'$ already is a basic point for this problem!).

Assume that the solution of the extended problem is

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}.$$

- If $y_0 \neq 0$, then the original problem is infeasible ($\Omega = \emptyset$).
- If $y_0 = 0$, then x_0 is a basic point (= possible start for the original problem).
- This is not the only Phase 1 algorithm.

EPILOGUE

- Open Problem: *Are there LP algorithms of polynomial complexity?*
- The Simplex Method has exponential complexity in the worst case (*Kree–Minty–Cheval counterexample*)
- Interior Point Methods (*Khatchiyan, 1978*): $\#Op \propto \mathcal{O}(n^4 L)$
- Karmarkar (1984): $\#Op \propto \mathcal{O}(n^{3.5} L)$
- Current record (??) *Interior Barrier Primal–Dual methods*, $\#Op \propto \mathcal{O}(n^3 L)$. (We return to this method after discussing penalty and barrier methods)

- Solving large LP problems is BIG business!
- A missing conversion between the commercial MPS data format and Matlab has (so far) prevented me from giving you realistic LP-problems

LINEAR PROGRAMMING IN MATLAB OPTIMIZATION TOOLBOX

Basic function: linprog

Solves the general LP-problem

$$\min_x f'x,$$

$$Ax \leq b$$

$$A_{eq}x = b_{eq}.$$

$$lb \leq x \leq ub$$

where f , x , b , b_{eq} , lb , and ub are vectors and A , A_{eq} are matrices (may be entered as *sparse* matrices)

Syntax:

x	=	linprog(f,A,b,Aeq,beq)
x	=	linprog(f,A,b,Aeq,beq,lb,ub)
x	=	linprog(f,A,b,Aeq,beq,lb,ub,x0)
x	=	linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval]	=	linprog(...)
[x,fval,exitflag]	=	linprog(...)
[x,fval,exitflag,output]	=	linprog(...)
[x,fval,exitflag,output,lambda]	=	linprog(...)

Example: The Standard form:

$$\min c'x,$$

$$Ax = b,$$

$$x \geq 0.$$

$x = \text{linprog}(c,[],[],A,b,\text{zeros}(\text{size}(c)),[])$

- Note the Matlab convention with *placeholders*, "[]"

INPUT:

x0: Start point. Used only for medium problems.

Options: Structure of parameters

(NB! Syntax change in Matlab 7 not yet modified in the written documentation. See examples below)

LargeScale: 'on'/'off'

Display: 'off'/'iter'/'final' (large scale problems)

MaxIter: Max number of iterations

Simplex: 'on'/'off' ('on' ignores x0)

TolFun: Objective tolerance (large scale problems)

OUTPUT:

x,fval: Solution and objective

exitflag:

- 1 Iteration terminated OK
- 0 Number of iterations exceeded MaxIter
- 2 No feasible point found
- 3 Problem is unbounded
- 4 NaN value encountered
- 5 Both primal and dual are infeasible
- 7 Search direction became too small

output: Structure of iteration onfo.

iterations: Number of iterations

algorithm: Algorithm used

cgiterations: The number of PCG iterations (large-scale algorithm only)

message: Output message

lambda: Structure of Lagrange multipliers

ineqlin: for linear inequalities $Ax \leq b$,

eqlin for linear equalities $A_{eq}x = b_{eq}$,

lower for lb,

upper for ub.

ALGORITHMS:

Small/Medium scale: SIMPLEX-like including Phase 1

Large scale: Primal-dual inner method

EXAMPLES FROM THE DOCUMENTATION

A. Small problem

Find $\mathbf{x}$ that minimizes

$$f(\mathbf{x}) = -5x_1 - 4x_2 - 6x_3$$

subject to

$$x_1 - x_2 + x_3 \leq 20$$

$$3x_1 + 2x_2 + 4x_3 \leq 42$$

$$3x_1 + 2x_2 \leq 30$$

$$0 \leq x_1, 0 \leq x_2, 0 \leq x_3$$

First, enter the coefficients, then call **LINPROG**:

```
f = [-5 -4 -6];
```

```
A = [ 1 -1  1  
      3  2  4  
      3  2  0];
```

```
b = [20 42 30];
```

```
lb = zeros(3,1);
```

```
[x,fval,exitflag,output,lambda] = ... linprog(f,A,b,[],[],lb);
```

```
x          =      [0 15  3]
```

```
fval       =      -78.0
```

```
output:
```

```
iterations: 6
```

```
algorithm: 'large-scale: interior point' (!)
```

```
cgiterations: 0
```

```
message: 'Optimization terminated.'
```

```
lambda.ineqlin =      [0 1.5 0.5]
```

```
lambda.lower   =      [1  0  0]
```

For solution by the Simplex method:

```
f = [-5 -4 -6]';  
A = [ 1 -1  1  
      3  2  4  
      3  2  0];  
b = [20 42 30]';  
lb = zeros(3,1);  
options = optimset('LargeScale','off','Simplex','on');  
[x,fval,exitflag,output,lambda] = ...  
linprog(f,A,b,[],[],lb,[],[],options);
```

(NB! If you forget enough placeholders, [], you get the error message "LINPROG only accepts inputs of data type double")

Now **output** gives:

iterations:	3
algorithm:	'medium scale: simplex'
cgiterations:	[]
message:	'Optimization terminated.'

(same solution!)

B Medium Problem

This problem is stored as a Matlab MAT-file.

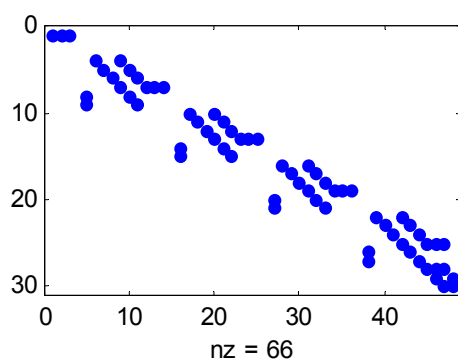
- 48 unknowns
- 30 inequality constraints
- 20 equality constraints
- $x \geq 0$

Entered into Matlab simply by

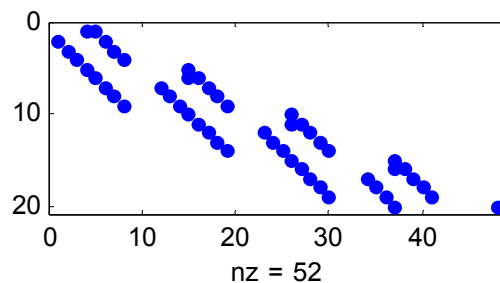
load sc50b

A	30x48	(sparse)
Aeq	20x48	(sparse)
b	30x1	
beq	20x1	
f	48x1	
lb	48x1	

Sparsity patterns:



A (inequalities)



Aeq (equalities)

```
load sc50b
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output,lambda] = ...
    linprog(f,A,b,Aeq,beq,lb,[],[],options);

x = [ 30  28 42 ... 102.4870]
```

Only `lambda.ineqlin(2)` and `lambda.ineqlin(3)` equal to 0:
only inequality 2 and 3 non-active.

`max(lambda.lower)= 8.2808e-015`: $x_i > 0$ for all $i = 1, \dots, 48$.

```
output =
    iterations:    43
    algorithm:     'medium scale: simplex'
    cgiterations:  []
    message:       'Optimization terminated.'
```

Large scale option:

```
options = optimset('LargeScale','on');
[x,fval,exitflag,output,lambda] = ...
    linprog(f,A,b,Aeq,beq,lb,[],[],options);
```

```
output =
    iterations:    8
    algorithm:     'large-scale: interior point'
    cgiterations:  0
    message:       'Optimization terminated.'
```

Same solution!

With display of results for each iteration:

```
options = optimset('LargeScale','on','Display','iter');
```

Residuals:		Primal Infeas $A \cdot x - b$	Dual Infeas $A' \cdot y + z - f$	Duality Gap $x' \cdot z$	Total Rel Error

Iter	0:	1.50e+03	2.19e+01	1.91e+04	1.00e+02
Iter	1:	1.15e+02	3.18e-15	3.62e+03	9.90e-01
Iter	2:	8.32e-13	1.96e-15	4.32e+02	9.48e-01
Iter	3:	3.51e-12	1.87e-15	7.78e+01	6.88e-01
Iter	4:	1.81e-11	3.50e-16	2.38e+01	2.69e-01
Iter	5:	2.63e-10	1.23e-15	5.05e+00	6.89e-02
Iter	6:	5.88e-11	2.72e-16	1.64e-01	2.34e-03
Iter	7:	2.61e-12	2.59e-16	1.09e-05	1.55e-07
Iter	8:	7.97e-14	5.67e-13	1.09e-11	3.82e-12

Optimization terminated.

OPTIMIZATION SOFTWARE

<http://www-fp.mcs.anl.gov/otc/Guide/>

- The [AIMMS](#) modeling language.
- The [AMPL](#) modeling language.
- [ANALYZE](#) linear programming model analysis.
- [ASA](#) - adaptive simulated annealing.
- [BPMPD](#) - linear programming.
- [BQPD](#) - quadratic programming.
- [BT](#) - minimization.
- [BTN](#) - block truncated Newton.
- [CML](#) - constrained maximum likelihood.
- [CNM](#) - linear algebra and minimization.
- [CO](#) - constrained optimization.
- [COMPACT](#) - design optimization.
- [CONOPT](#) - nonlinear programming.
- [CONSOL-OPTCAD](#) - engineering system design.
- [CONTIN](#) - systems of nonlinear equations.
- [CPLEX](#) - linear programming.
- [C-WHIZ](#) - linear programming models.
- [DATAFORM](#) - model management system.
- [DFNLP](#) - nonlinear data fitting.
- [DOC](#) - Design Optimization Control Program.
- [DONLP2](#) - nonlinear constrained optimization.
- [DOT](#) - Design Optimization Tools.
- [EASY FIT](#) - parameter estimation in dynamic systems.
- [Excel and Quattro Pro Solvers](#) - spreadsheet-based linear programming
- [EZMOD](#) - modeling environment for decision support systems
- [FortMP](#) - linear and mixed integer quadratic programming.
- [FSQP](#) - nonlinear and minmax constrained optimization, with feasible iterates.
- [GAMS](#) - modeling language.
- [GAUSS](#) - matrix programming language.
- [GENESIS](#) - structural optimization software.
- [GENOS 1.0](#) - nonlinear network optimization.
- [GINO](#) - nonlinear programming.
- [GRG2](#) - nonlinear programming.
- [GOM: Global Optimization for Mathematica](#).

- [HOMPACK](#) - nonlinear equations and polynomials.
- [HOPDM](#) - linear programming (interior-point).
- [HARWELL Library](#) - linear and nonlinear programming, nonlinear equations, data fitting.
- [HS/LP Linear Optimizer](#) - linear programming.
- [LOG](#) - constraint-based programming and nonlinear optimization.
- [IMSL](#) - Fortran and C Library.
- [KORBX](#) - linear programming.
- [LAMPS](#) - linear and mixed-integer programming.
- [LANCELOT](#) - large-scale problems.
- [LBFGS](#) - unconstrained minimization.
- [LBFGS-B](#) - bound-constrained minimization.
- [LGO IDE](#) - continuous and Lipschitz global optimization.
- [LINDO](#) - linear, mixed-integer and quadratic programming.
- [LINGO](#) - modeling language.
- [LIPSOL](#) - linear programming.
- [LNOS](#) - linear programming/network flow problems.
- [LOQO](#) - Linear programming, unconstrained and constrained nonlinear optimization.
- [LP88 and BLP88](#) - linear programming.
- [LSGRG2](#) - nonlinear programming.
- [LSNNO](#) - large scale optimization.
- [LSSOL](#) - least squares problems.
- [M1QN3](#) - unconstrained optimization.
- [MATLAB](#) - optimization toolbox.
- [MAXLIK](#) - maximum likelihood estimation.
- [MCS](#) - global optimization.
- [MILP88](#) - mixed integer programming.
- [MINOS](#) - linear programming and nonlinear optimization.
- [MINTO](#) - mixed integer linear programming.
- [MINPACK-1](#) - nonlinear equations and least squares.
- [MIPIII](#) - mixed integer programming.
- [MODLER](#) - linear programming modeling language.
- [MODULOPT](#) - unconstrained problems and simple bounds.
- [MOSEK](#) - linear programming and convex optimization.
- [MPL](#) - modeling system
- [MPSIII](#) - mathematical programming system.

- [NAG C Library](#) - nonlinear and quadratic programming, minimization
- [NAG Fortran Library](#) - nonlinear and quadratic programming, minimization
- [NETFLOW](#) - network optimization.
- [NITSOL](#) - systems of nonlinear equations.
- [NLPE](#) - minimization and least squares problems.
- [NLPJOB](#) - Mulicriteria optimization.
- [NLPQL](#) - nonlinear programming.
- [NLPQLB](#) - nonlinear programming with constraints.
- [NLSSOL](#) - constrained nonlinear least squares problems.
- [NLPSPR](#) - nonlinear programming.
- [NOVA](#) - nonlinear programming.
- [NPSOL](#) - nonlinear programming.
- [ODRPACK](#) - NLS and ODR problems.
- [OML](#) - linear and mixed-integer programming, model management.
- [OPL Studio](#) - optimization language and solver environment.
- [OPTDES](#) - design optimization tool.
- [OPTTECH](#) - global optimization.
- [OptiA](#) - unconstrained, constrained, quadratic, minimax, nonsmooth, and global optimization
- [OPTIMA Library](#) - optimization and sensitivity analysis.
- [OPTIMAX](#) - component software for optimization
- [OPTMUM](#) - optimization.
- [OPTPACK](#) - constrained and unconstrained optimization.
- [OptQuest](#) - global optimization
- [OSL](#) - linear, quadratic and mixed-integer programming.
- [PCOMP](#) - modelling language with automatic differentiation.
- [PCx](#) - linear programming with a primal-dual interior-point method.
- [PDEFIT](#) - parameter estimation in partial differential equations.
- [PETSc](#) - parallel solution of nonlinear equations and unconstrained minimization problems.
- [PLAM](#) - algebraic modeling language for mixed integer programming, constraint logic programming, etc.
- [PORT 3](#) - minimization, least squares, etc.
- [PROC LP](#) - linear and integer programming.
- [PROC NETFLOW](#) - network optimization.

- [PROC NLP](#) - various quadratic and nonlinear optimization problems.
- [Q01SUBS](#) - quadratic programming for matrices.
- [QAPP](#) - quadratic assignment problems.
- [QL](#) - quadratic programming.
- [QPOPT](#) - linear and quadratic problems.
- [RANDMOD](#) - linear programming model randomizer.
- [SIMUSOLV](#) - modeling software.
- [SPRNLP](#) - sparse and dense nonlinear programming, sparse nonlinear least squares, including the [SOCS](#) package for optimal control
- [SPEAKEASY](#) - numerical problems and operations research.
- [SNOPT](#) - large-scale quadratic and nonlinear programming problems.
- [SQOPT](#) - large-scale linear and convex quadratic programming problems.
- [SQP](#) - nonlinear programming.
- [SYNAPS Pointer](#) - multidisciplinary design optimization software
- [SYSFIT](#) - parameter estimation in systems of nonlinear equations.
- [TENMIN](#) - unconstrained optimization.
- [TENSOLVE](#) - nonlinear equations and least squares.
- [TN/TNBC](#) - minimization.
- [TNPACK](#) - nonlinear unconstrained minimization.
- [TSA88](#) - network linear programming.
- [UNCMIN](#) - unconstrained optimization.
- [VE08](#) - nonlinear optimization.
- [VE10](#) - nonlinear least squares.
- [VIG and VIMDA](#) - decision support system.
- [What'sBest](#) - linear and mixed integer programming.
- [WHIZARD](#) - linear programming, mixed-integer programming.
- [XLSOL](#) - Linear, integer and nonlinear programming for AMPL models
- [XPRESS-MP](#) from Dash Associates - linear and integer programming.