The Dikin's method and its solvers for linear programming

Name: Zhonghao ZHANG

Dept: EE

Lecturer: Prof. Chee Wei Tan

Date: Nov 16, 2010

Outline

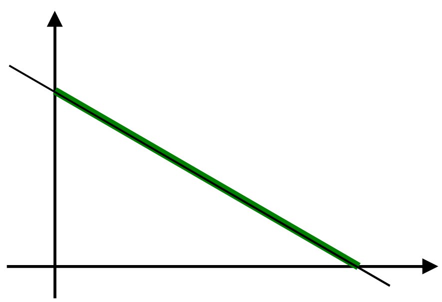
- The Dikin’s method for Linear Programming
  - Principle
  - Comparison with the Simplex Method
- Two MATLAB solvers using Dikin’s method
  - Complexity and stability

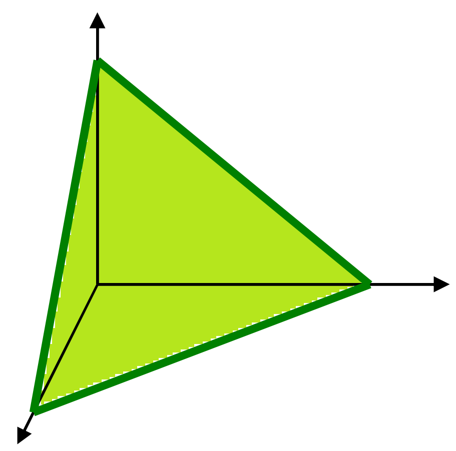
Linear Programming (LP)

- Standard form

$$\begin{aligned} &\text{minimize} && z = c^T x \\ &\text{subject to} && Ax = b \quad A : m \times n \\ & && x \geq 0 \end{aligned}$$

- Examples

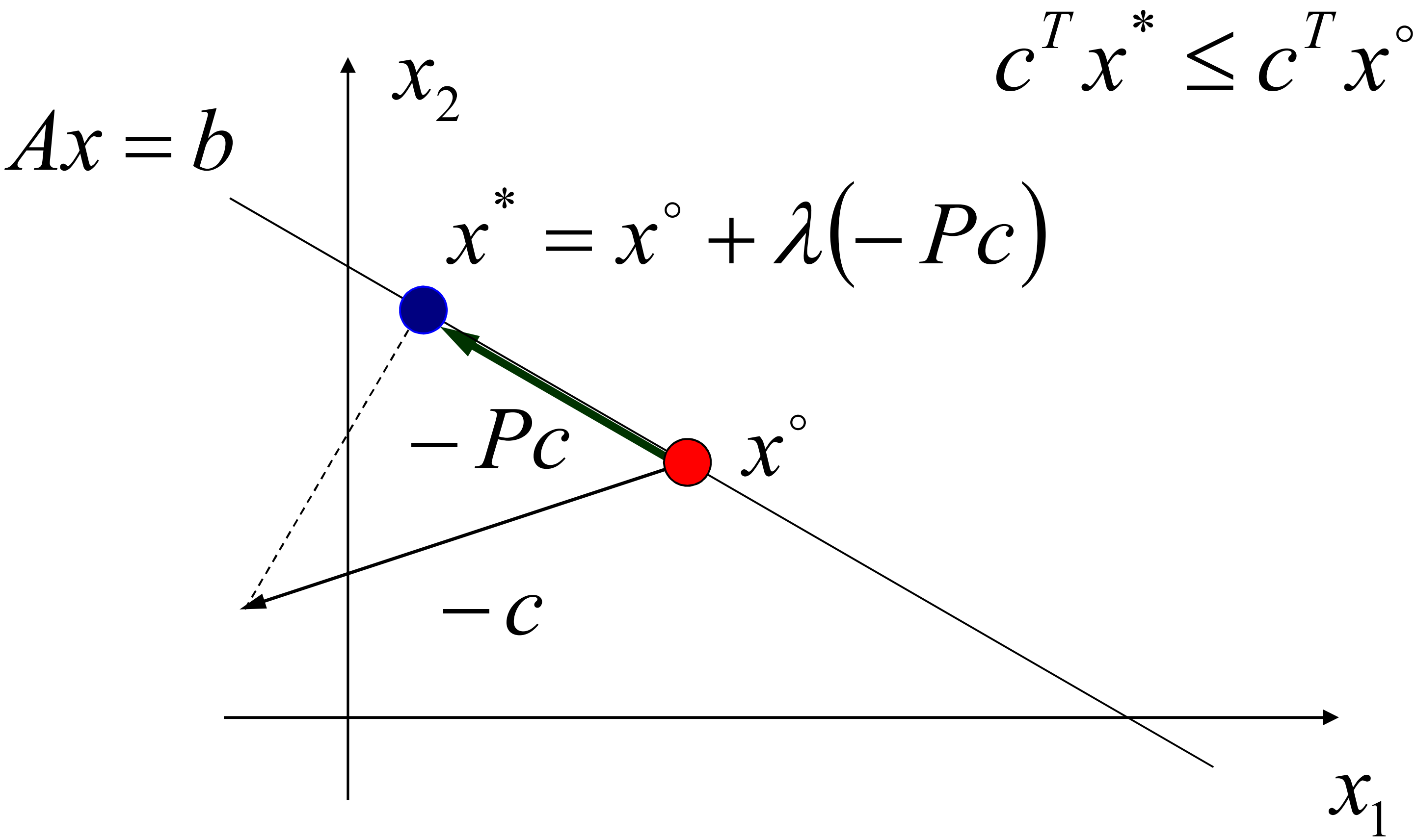




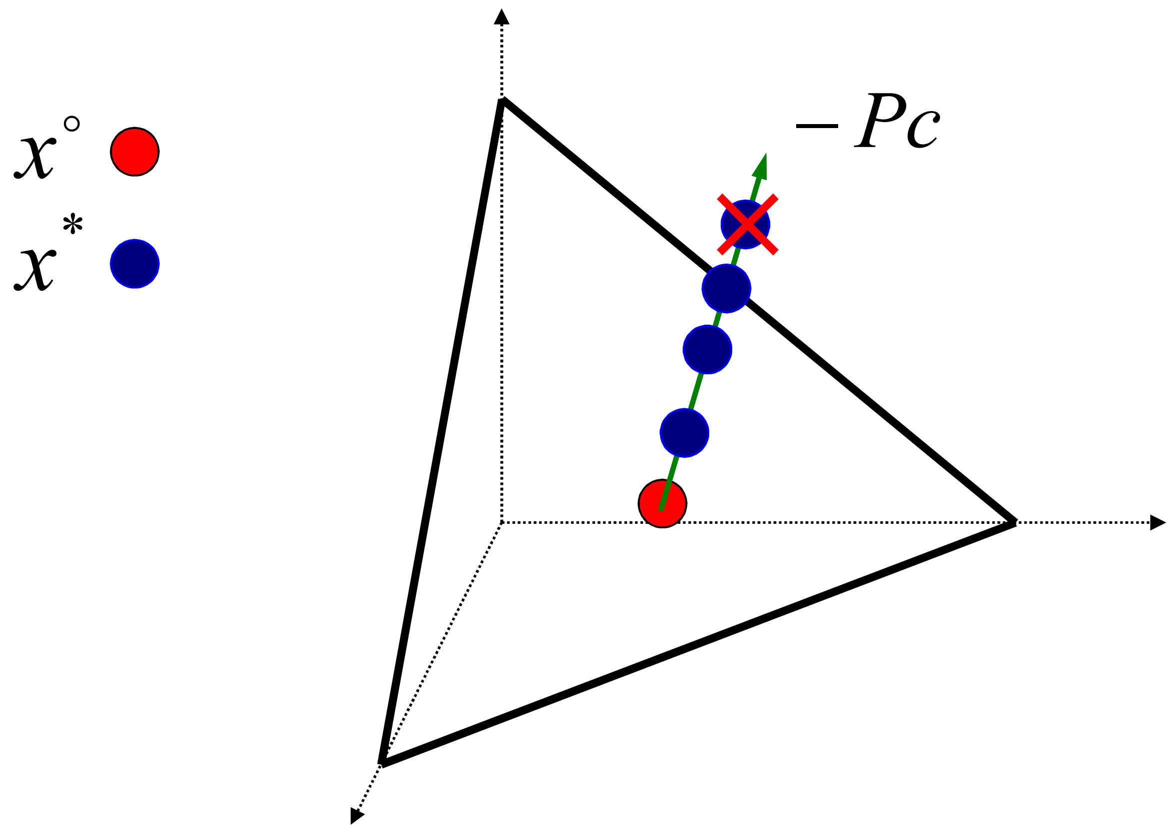
The Dikin’s method

- How does the Dikin’s method work?
  - Start with a feasible (interior) point
  - Iterative steps
    - Steepest descent(SD) algorithm
    - Affine scaling(AS) algorithm
  - Convergence test

Steepest descent (SD) algorithm



SD algorithm



Keep the point away from the boundary
and change the direction!

Affine scaling(AS) algorithm

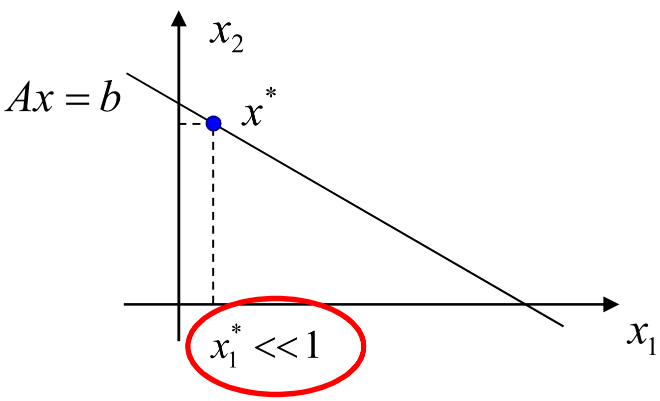
- Main idea:
  - *Scale the coordinate axes as*

$$x \Rightarrow x' = diag\left(1/x^*\right)x$$

$$x^* \Rightarrow diag\left(1/x^*\right)x^* = \begin{pmatrix}1, & 1, & \cdots & 1\end{pmatrix}^T$$

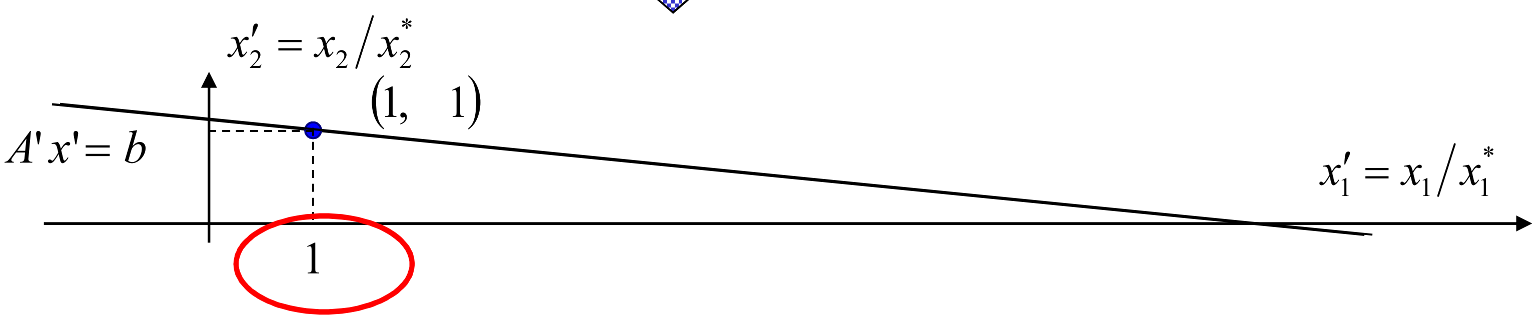
AS algorithm

$$\underbrace{\begin{bmatrix} 1/x_1^* & 0 \\ 0 & 1/x_2^* \end{bmatrix} x}_{x'} \qquad \underbrace{\begin{bmatrix} x_1^* & 0 \\ 0 & x_2^* \end{bmatrix} c}_{c'} \qquad \underbrace{A \begin{bmatrix} x_1^* & 0 \\ 0 & x_2^* \end{bmatrix}}_{A'}$$

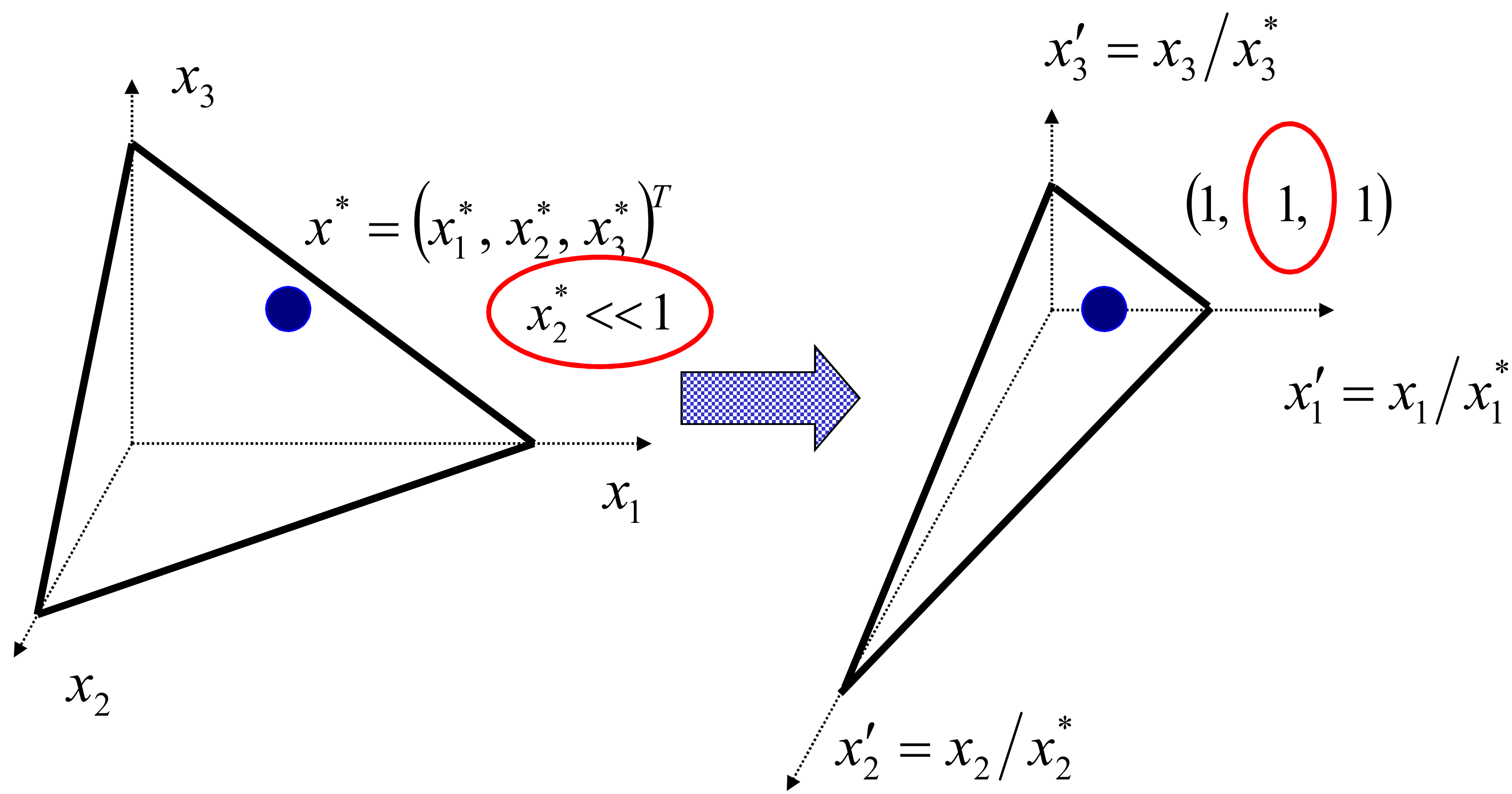


equivalent objective function : $z = c^T x = c'^T x'$

equivalent constraint : $b = Ax = A'x'$

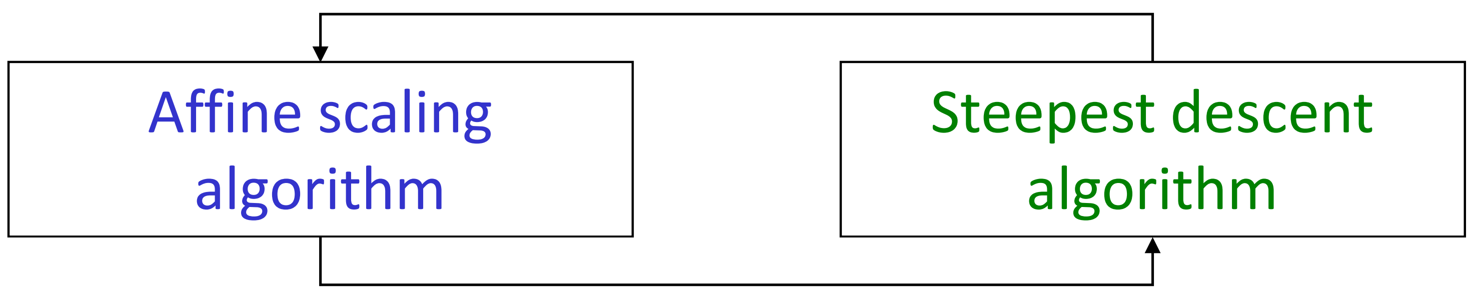


Affine scaling algorithm

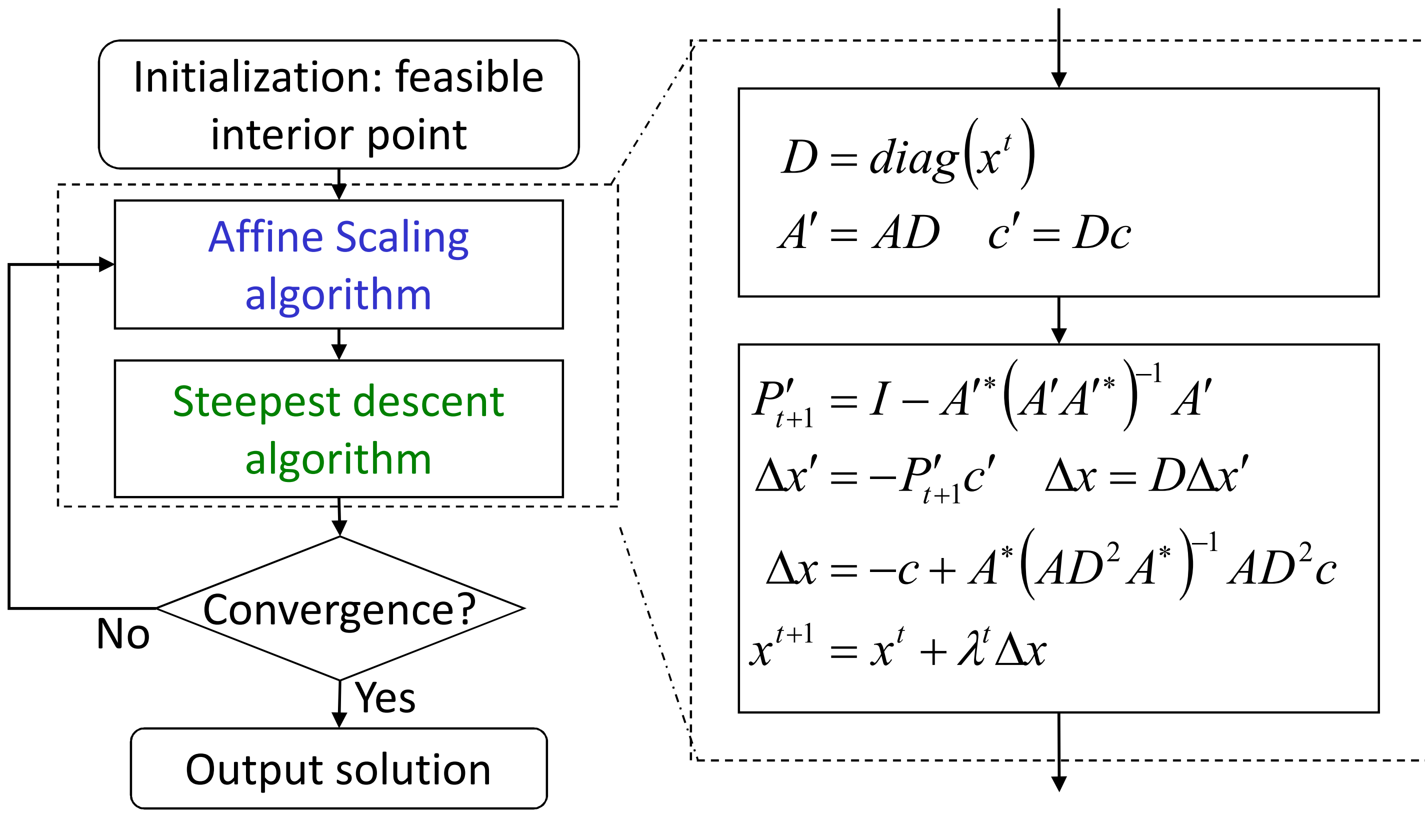


Affine scaling algorithm

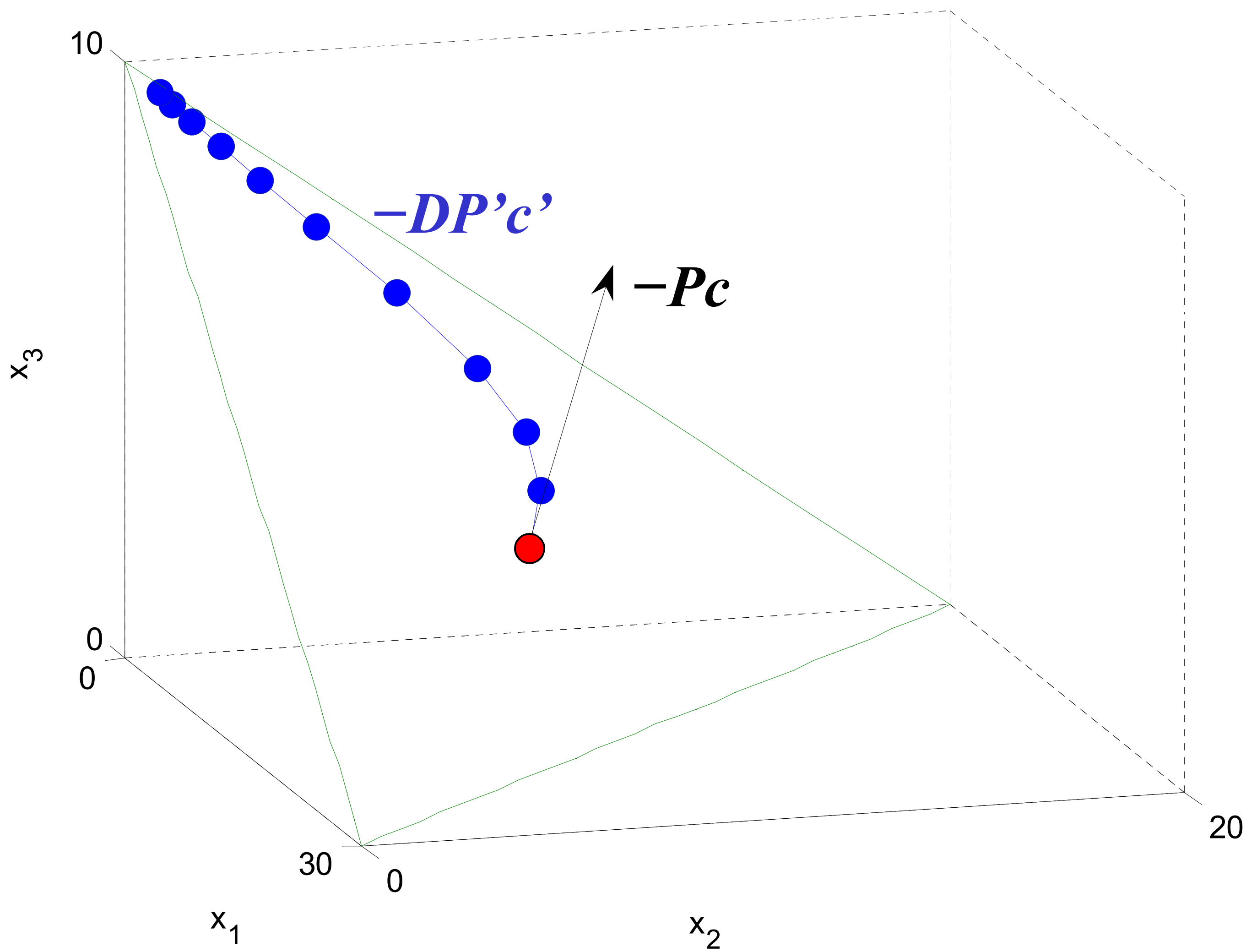
- Once we have "centered" the point x^* to $(1, 1, \ldots, 1)^T$, we then perform SD algorithm again.



The Dikin's method



Example



Comparison to Simplex method

- The ***simplex method*** is another competitive tool to solve LP problem.
- The classical simplex method jumps from one vertex of feasible set (simplex) to another vertex seeking the optimal one.
- In the worst case, simplex method is NP hard; while the Dikin's method is P hard problem.

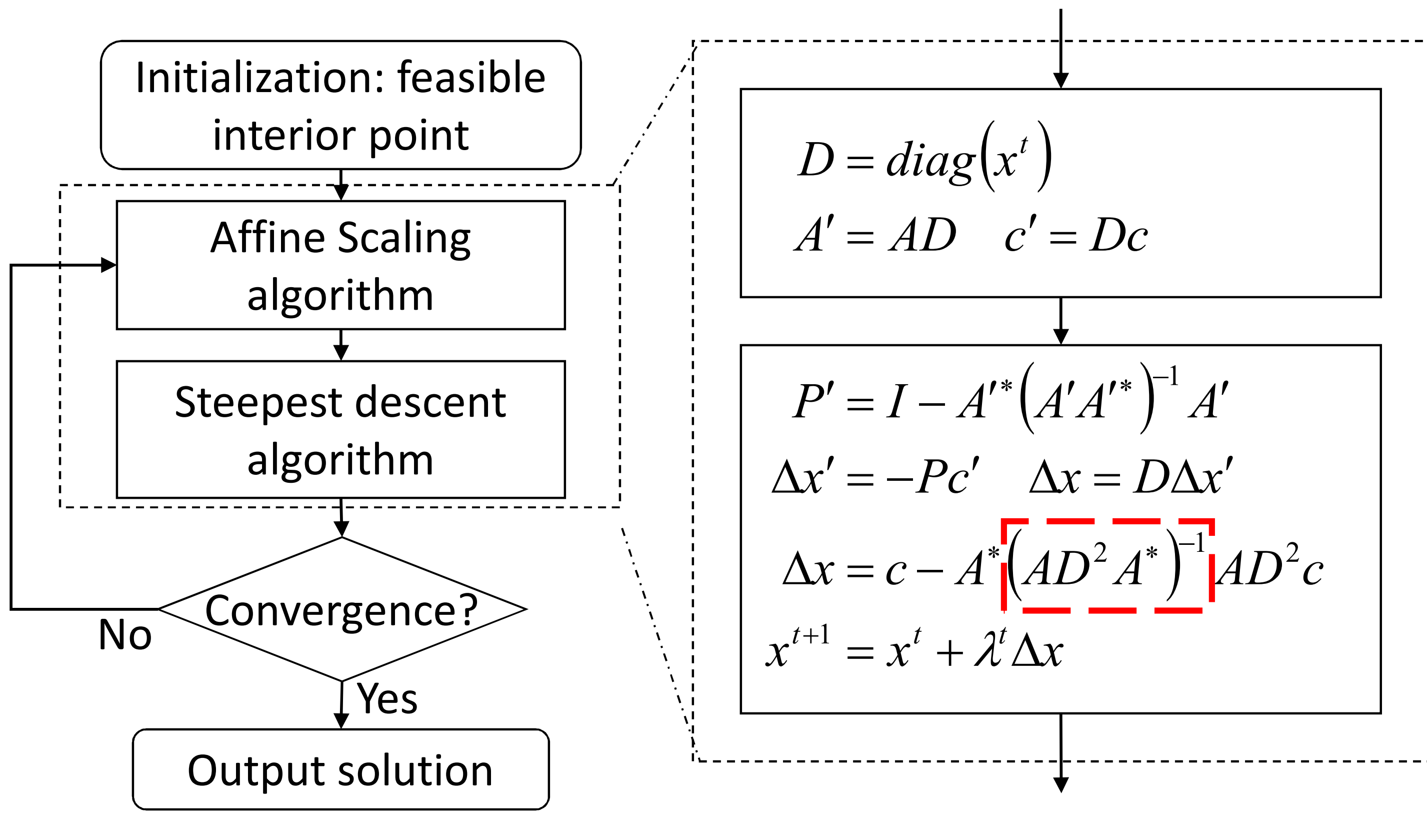
Two Dikin's solvers for LP

- A 75-character MATLAB solver
- My MATLAB solver.

The 75-character solver

Matrix inversion, perhaps **unstable**

```
for i=1:50,p=diag(x)^2;r=p*(c-A'*(A*p*A'\A*p*c));x=x-r*min(x./abs(r))/2;end
```



The 75-character solver

- 75-character solver
  - It calculates the projection matrix P' using:

$$P' = I - A'^{*}\left(A'A'^{*}\right)^{-1} A'$$

  - Actually, there are other ways to obtain P', for example:

$$P' = U_{\text{null}} U_{\text{null}}^{*} \qquad U_{\text{null}} : n \times \left(n - rank\left(A'\right)\right)$$

where columns of U_{null} are the bases of null space of A'.

Determine it using the single value decomposition (**SVD**)

The 69-character solver

- 69-character solver using SVD.

```
for i=1:50,X=diag(x);F=X*null(A*X);d=F*F'*c;x=x-d/max(X\abs(d))/2;end
```

  - Almost the same as 75-character solver, the only difference is calculation of projection matrix P'.

Comparison

- Complexity of each iteration
  - Assume $A' : m \times n$ and $A'A'^*$ is invertible.
  - 75-character solver:

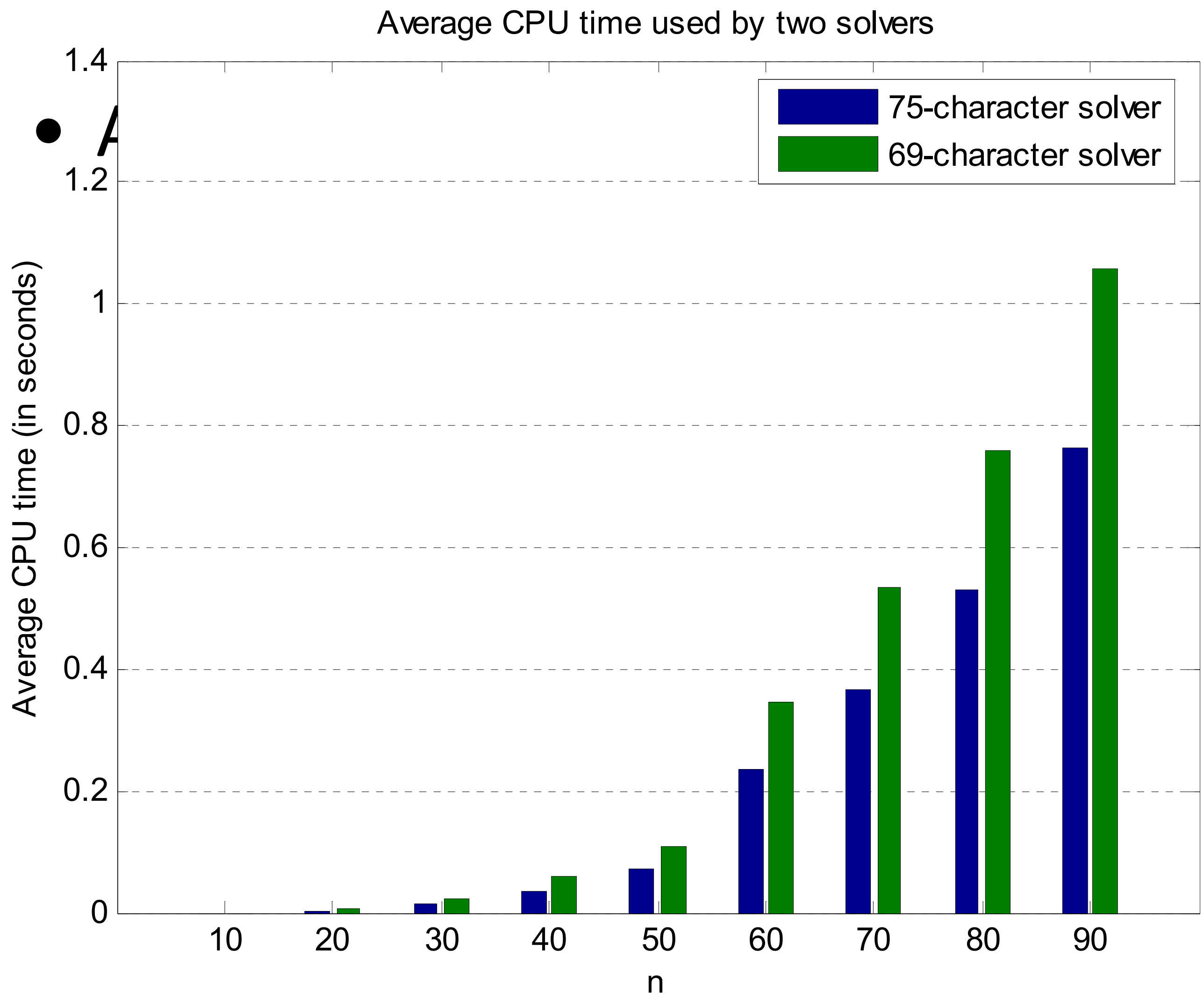
    matrix inversion $O\left(m^3\right)$

  - 69-character solver:

    SVD: $4mn^2 + 8n^3$

Comparison – complexity

- A … vers



Approximately the same

Comparison – Stability

- Stability analysis
  - 75-character solver:

$$\Delta x = -c + A^* \left(A D^2 A^* \right)^{-1} A D^2 c$$

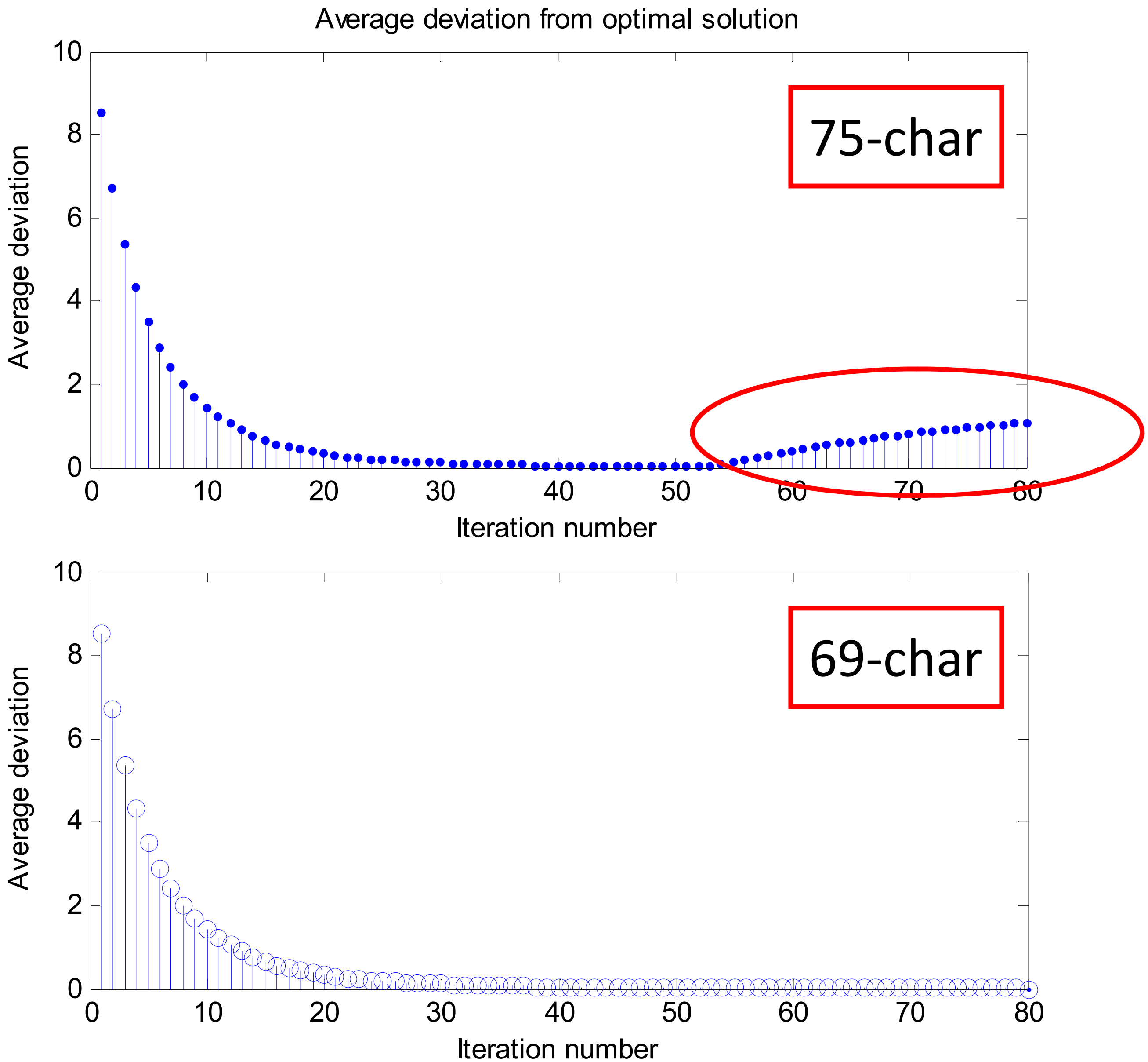
  - 69-character solver:

$$U_{\text{null}} = \text{null}\left(AD\right) \quad \Delta x = -c + D U_{\text{null}} U_{\text{null}}^* D c$$

Comparison – Stability

- Deviation from the optimal solution

$$\left\| x^t - x_{opt} \right\|_2^2$$


Average deviation from optimal solution
75-char
Average deviation
Iteration number
69-char
Average deviation
Iteration number

Comparison

- Conclusion

	75-char	69-char(using SVD)
Complexity	slightly lower	slightly higher
Stability	poor	nice

Thanks!

Q & A