Interior Point Methods

- Relatively new method to solve LP's (invented in 1980's; Simplex in 1950's)

  but as powerful as Simplex.

- Solution concept

	Simplex	Int. Point Methods
similarities	iterative	iterative
	trial soln at each iteration	trial soln at each iteration
differences	trial solns are CPF's	trial solns are interior points (not on boundary)

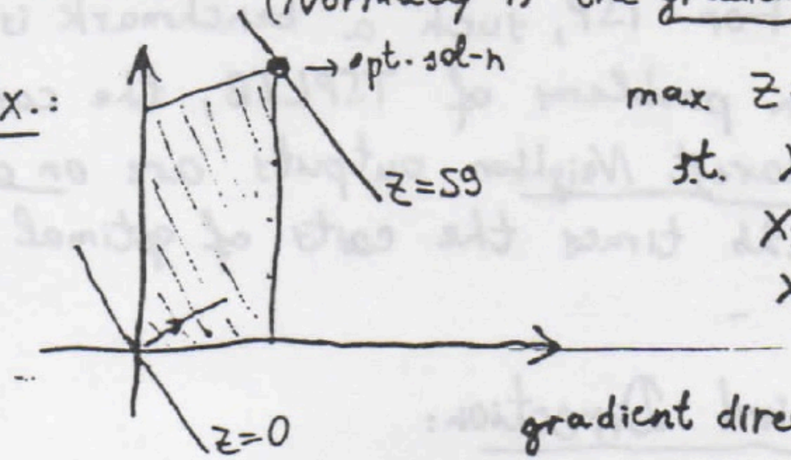
- Some general idea how Int. Point Algorithm works

  ▼ Trial solns: Interior of feasible region.

  ▼ Direction to move: Direction that

improves obj. f-n value at the fastest possible rate. (Normally is the gradient direction)

Ex.:



max $Z = 2X_1 + X_2$

s.t. $X_1 \leq 3$

$X_2 - X_1 \leq 50$

$X_1, X_2 \geq 0$

▼ How far to move on the current best direction?

If gets very close to boundary then it will be hard to move fast in subsequent iterations.

How to provide that it's not getting too close to boundary?

Interior Point Methods (cont.)

- How to provide that we are not getting too close to boundary?

Different variants of Int. Point Methods deal differently with this situation. A popular method is using barrier functions.

- Idea: Switch from original LP to a related nonlinear f-n incorporating barrier for constraints.

E.g., in our problem substitute LP by

$$\max\ 2X_1 + X_2 + \mu \cdot \ln(3 - X_1) + \mu \cdot \ln(50 - X_2 + X_1) + \mu \cdot \ln X_1 + \mu \cdot \ln X_2$$

where μ is (small) positive number

- logarithms provide that original constraints are still strictly satisfied.
- new obj. f-n is concave → can maximize by finding stationary point (i.e., gradient = 0)
- the smaller μ the closer new point to boundary.

- There are optimality criteria to check how close is the interior point to optimum
- Stop the algorithm when it is sufficiently close to optimum.

Comparison with Simplex (in terms of performance)

- Iterations of Int. point Methods are more complicated and many times longer than Simplex iterations.
- But number of iterations for Int. Point Methods is much smaller than number of Simplex iterations:

# of functional constr.	# of iterations: Simplex	# of iterations: int. point
10	20	20
10,000	20,000	<100

- Thus,
  - for small problems (<100 constraints), Simplex is more efficient

- for problems of medium size (several hundred constraints), performances of 2 algorithms are comparable
- for larger problems, int.-point methods are more efficient

- A drawback of Int.-Point Methods:
  limited capabilitity to perform Sensitivity Analysis

To overcome it,
try to switch to Simplex method Once Int.-Point Algorithm has finished