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.

  \[ \max 2x_1 + x_2 + M \cdot \ln(3-x_1) + M \cdot \ln(50-x_1x_2) + M \cdot x_1 + M \cdot \ln x_2 \]

  where \( M \) is (small) positive number

- Logarithms provide that original constraints are still strictly satisfied.
- New obj f-n is concave \( \rightarrow \) can maximize by finding stationary point (i.e., gradient = 0)
- Smaller \( M \) the closer new point to boundary.

- 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 capability to perform Sensitivity Analysis

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

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 may times longer than Simplex iterations.
- But number of iterations for Int. Point Methods is much smaller than number of Simplex iterations:

<table>
<thead>
<tr>
<th># of functional constr.</th>
<th># of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>20</td>
</tr>
<tr>
<td>20,000</td>
<td>&lt;100</td>
</tr>
</tbody>
</table>

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