An exhaustive analysis of LCP solver performance on randomly generated rigid body contact problems

Motivation and goal
The Linear Complementarity Problem (LCP) arises in rigid body contact problems. Simulation robustness and performance are directly affected by the particular solver employed. Important considerations include runtime, solution quality, and reliability. In this work, we seek a systematic, comparative understanding of the available solvers, and ultimately hope to identify trade-offs between solvers and underlying contact methods.

Generating random contact problems
One challenge is that the underlying physical problem imposes constraints on the LCP inputs. A meaningful analysis must consider these aspects.

The solvers
The following four methods were employed:
- **Lemke**
  A C++ version of the LEMKE Matlab library produced by Fackler and Miranda [1].
- **PATH**
  An interface to Ferris and Munson's commercial grade LCP solver [2].
- **SOR scheme**
  An implementation of the projected symmetric successive over-relaxation scheme [3].
- **Interior point method**
  An implementation of the primal-dual interior point method for solving convex LCPs described in [4].

Difference between solvers
The following summarizes of the solution frequency of the tested solvers.

<table>
<thead>
<tr>
<th>Solver</th>
<th>For all</th>
<th>Average</th>
<th>Any parameterization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lemke</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>PATH</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>SOR</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Interior point</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
<th>For all</th>
<th>Average</th>
<th>Any parameterization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lemke</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>PATH</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>SOR</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Interior point</strong></td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Evans Drumwright

Dylan Shell

Sensitivity to parameters
The following summarizes of the solution frequency and quality of the tested solvers.

LCPs as Convex and Strictly Convex QPs
Definition: These so-called monotone problems arise for an LCP (q,M) where M is positive semi-definite. When q is in the range of M, the LCP always has a solution.

Significance: LCPs with PD M arise in [7].

Generating Random Convex LCPs:
1. Randomly pick the number of generalized coordinates (n) in range [2,11].
2. Randomly pick the number of generalized coordinates (nSIG) in range [6,24].
3. Randomly pick the number of polygon edges in the friction cone (nk) in 4, 6, 8, ..., 40.
4. Generate a gcx1 vector (a) with each component uniformly randomly selected from range [0,1].
5. Generate a gcx2 vector (b) with each component uniformly randomly selected from range [-1,1].
6. Generate a gcxk matrix (N) with each element uniformly randomly selected from range [-1,1].
7. Generate a gcxk+nk matrix (D) with elements uniformly randomly selected from range [-1,1]; D is constructed such that, for every column d, the column -d also appears in D.
8. Randomly make some columns of N and the corresponding columns of D, linearly dependent.
9. Generate gcxgcxgcx symmetric generalized inertia matrix (G) either:
   (a) the identity matrix,
   (b) a PD matrix generated through summing nge rank-1 updates.
10. Generate the LCP as in Anitescu-Potra [5].