### 1 - Gas Molecules Simulation

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Correctness</th>
<th>Style</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The function header matches the one provided in the handout (1S). There is a concise comment describing what the function does and what the input parameters are (1S).</td>
<td>1</td>
<td></td>
<td>* Please grade this on completeness; i.e. if description is wrong but student attempted to comment, reward point. Alert student about the mistake.</td>
</tr>
<tr>
<td>2</td>
<td>The block of graphics commands in the handout (page 2) and hold off are incorporated into the code (1C).</td>
<td>1</td>
<td></td>
<td>* Note that the order might not be the same as in the handout depending on the Matlab version. Please see announcement on course website dated 3/7.</td>
</tr>
<tr>
<td>3</td>
<td>Code draws n molecules (where n is the length of x) at their corresponding positions; can't assume fixed/max value for n (1C). Each molecule must have radius r (1C). The first molecule must be magenta while all other molecules must be blue (1C).</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>There is a figure that is produced which contains a plot of the molecules.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>The code uses tools/functions which are not allowed in this course and/or defeats the purpose of the project.</td>
<td>-1</td>
<td></td>
<td>* examples: break, continue</td>
</tr>
<tr>
<td>6</td>
<td>The function header matches the one provided in the handout (1S). There is a concise comment describing what the function does (1S) and what the input and output parameters are (1S + 1S).</td>
<td>4</td>
<td></td>
<td>* Please grade this on completeness; i.e. if description is wrong but student attempted to comment, reward point. Alert student about the mistake.</td>
</tr>
<tr>
<td>7</td>
<td>The first action of this code is to draw the molecules at their initial locations using drawMolecules.m. In particular, students must call the function instead of copying the contents over.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A loop is used to perform the simulation over T time steps.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>For all molecules (can't assume fixed/max value for number of molecules) (1C), code correctly checks if each molecule hits a vertical wall (1C) or horizontal wall (1C). If the molecule hits a wall, the position and velocity for that molecule is updated correctly. (1C + 1C for horizontal and vertical wall)</td>
<td>5</td>
<td></td>
<td>* Note: students can do these in any order: bouncing off the wall, collisions, and calculating the new position</td>
</tr>
<tr>
<td>10</td>
<td>For each pair of molecules, the checkCollision function is called correctly with the right inputs (1C) and the outputs are assigned to correct variables (1C). The pairwise checking of molecules must be done efficiently (1S).</td>
<td>2</td>
<td></td>
<td>* Note: Students can use for j = 1:n, k = 1:n as long as there's an if condition inside to avoid redundancy. However, if student uses code similar to page 4 of handout, take off the style point. Also, it is not considered inefficient if student uses 2 separate loops for 12 and 13 as opposed to how it's done in the solutions.</td>
</tr>
<tr>
<td>11</td>
<td>The new positions of the molecules are calculated after one time step.</td>
<td>1</td>
<td></td>
<td>* Vectorized code is OK</td>
</tr>
<tr>
<td>12</td>
<td>The drawMolecules code is called correctly inside the loop and there is a pause of 0.01s for each time step.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The values of xFinal and yFinal are appropriately set.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>The code uses tools/functions which are not allowed in this course and/or defeats the purpose of the project.</td>
<td>-1</td>
<td></td>
<td>* examples: break, continue</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
<td>Correctness</td>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>The function header matches the one provided in the handout (1S). There is a concise comment describing what the function does (1S) and what the input and output parameters are (1S + 1S).</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Code correctly checks if 2 molecules collide.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>If no collision occurs, the velocities returned should be the same.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>If collision occurs, the post collision velocities are correctly computed (1C + 1C for each molecule).</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>The code uses tools/functions which are not allowed in this course and/or defeats the purpose of the project.</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Please grade this on completeness; i.e. if description is wrong but student attempted to comment, reward point. Alert student about the mistake.

| 20 | The function header has title coinToss and has input and output parameters as specified in the handout (1S). There is a concise comment describing the code (1S) and a concise comment describing the input and output parameters (1S + 1S). | 4           |       |
| 21 | If T > 20, then only a numerical simulation must be performed. Otherwise, a numerical simulation and a figure displaying the trials must be shown. (1C). There must be a pause of p seconds for the latter case (1C). | 2           |       |

- input and output parameters can be in any order. Please grade this on completeness; i.e. if description is wrong but student attempted to comment, reward point. Alert student about the mistake.

| 22 | The figure must contain an array of n x n tiles (any color) (1C), the coin on the tiles at each trial (1C), and text in the figure window indicating the number of trials in which the coin is entirely within a tile (1C). There must be a space along the border to allow for cases when the coin hangs over an edge (1S). The grid must have n x n boxes of equal dimension (1S). | 3           | 2     |

- It's OK if grid has tick marks on axes; efficiency is not important here. Also, it's OK to show all previous coins.

| 23 | For each time step, the position of the center of the coin is correctly generated (1C) and the code correctly checks if the coin lies entirely inside the tile (1C). | 2           |       |
| 24 | If the coin lies inside the tile, the coordinates of the center of the coin are correctly stored into xOnTile and yOnTile (1C). The value of prob is correctly computed as well (1C). | 2           |       |

- Percentage is OK!

| 25 | There is at least 1 subfunction to accomplish the tasks above (1C). The subfunction must have a comment to describe what the code does (1S) and to describe the input and output (if applicable) parameters (1S + 1S). | 1           | 3     |

| 26 | The code uses tools/functions which are not allowed in this course and/or defeats the purpose of the project. | -1          |       |

- Note that students can use drawrect, drawdisk, drawdisknoborder, drawstar

### 4 - General Style (award point if not applicable)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Correctness</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Body of code is sufficiently, but not excessively, commented (1pt).</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Line lengths are not too long, i.e. do not often exceed 80 columns (it's ok if a few lines are long, especially for print statements)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>No extra output, e.g. debugging output, is produced</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Proper indentation is always used</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Meaningful variable names are used and do not overwrite MATLAB keywords</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Important parameters (constants) are named as variables</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>No superfluous code, e.g. an empty if/else branch or useless loop</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Reasonably efficient code</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Description of Coin Toss

- The code correctly checks if 2 molecules collide.
- If no collision occurs, the velocities returned should be the same.
- If collision occurs, the post collision velocities are correctly computed (1C + 1C for each molecule).
- The code uses tools/functions which are not allowed in this course and/or defeats the purpose of the project.

### Description of General Style

- Body of code is sufficiently, but not excessively, commented (1pt).
- Line lengths are not too long, i.e. do not often exceed 80 columns (it's ok if a few lines are long, especially for print statements)
- No extra output, e.g. debugging output, is produced
- Proper indentation is always used
- Meaningful variable names are used and do not overwrite MATLAB keywords
- Important parameters (constants) are named as variables
- No superfluous code, e.g. an empty if/else branch or useless loop
- Reasonably efficient code
Semicolons are not placed at incorrect places, e.g. at the end of: 'if', 'elseif', 'else', 'for', 'while', 'function'

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Student's code does not execute (or student provides a script when a function is required and vice-versa)</td>
<td>-1 from final score</td>
</tr>
<tr>
<td>P2</td>
<td>Student's code crashes or does not terminate (infinite loop) for normal cases.</td>
<td>-1 from final score</td>
</tr>
<tr>
<td>P3</td>
<td>There are function headers and file names that do not match those specified in the project description exactly.</td>
<td>-1 from final score</td>
</tr>
</tbody>
</table>

* Since all files to be submitted are functions, please disregard this penalty and only penalize in 1,6,15,20

### 6 - Grade Calculation

<table>
<thead>
<tr>
<th></th>
<th>TC = 31</th>
<th>TS = 29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total possible correctness points</td>
<td>C = min( ) + 1 free point, TC)</td>
<td>S = min( ) + 1 free point, TS)</td>
</tr>
<tr>
<td>Student correctness Points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student style points</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exceptions: If any file is missing/unacceptable, subtract n style points from each missing file where n = floor(total general style points / total number of files that need to be submitted)

**Final score = [(C/TC) + (S/TS)] * 5 - Penalties**

Score is out of 10 and has one decimal place, rounded to nearest value. No negative scores.