Appendix A: Velocity Verlet Time Integration

Let the position of a rigid body be $\boldsymbol{x}$ and the velocity $\boldsymbol{v}$ with respect to its center of gravity, the orientation $\boldsymbol{q}$ (represented by a unit quaternion) and the angular velocity $\boldsymbol{\omega}$. Let m be its mass and $\mathbf{I}$ be its orientation dependent inertia tensor. The sum of all external forces and torques acting on the body due to gravity and springs are denoted $\boldsymbol{f}$ and $\boldsymbol{\tau}$, respectively.

Quantities at the beginning $t = t^{(l)}$ and end $t^{(l+1)} = t^{(l)} + h$ of a time step of length h are denoted by $*^{(l)}$ or $*^{(l+1)}$, respectively. To achieve more accurate results, we slightly relaxed the restrictions mentioned in Section 3 and show how to replace the forward Euler time integration of [GBF03] with a more accurate Velocity Verlet method.

Let $\boldsymbol{q}(\triangle\boldsymbol{\omega})$ represent an incremental rotation and $\circ$ be the quaternion multiplication. The linear and angular accelerations are obtained from the equations of motion $\boldsymbol{\sigma} = \mathbf{I}^{-1}(\boldsymbol{\tau} - (\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}))$ and $\boldsymbol{a} = \frac{\boldsymbol{f}}{m}$. The Verlet method is given by first updating the body's location with

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{x}^{(l)} + h\,\boldsymbol{v}^{(l)} + \frac{h^2}{2}\boldsymbol{a}^{(l)} \tag{9}$$

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}(h\,\boldsymbol{\omega}^{(l)} + \frac{h^2}{2}\boldsymbol{\sigma}^{(l)}) \circ \boldsymbol{q}^{(l)} \tag{10}$$

and then the velocities with

$$\boldsymbol{v}^{(l+1)} = \boldsymbol{v}^{(l)} + \frac{h}{2}\left(\boldsymbol{a}^{(l)} + \boldsymbol{a}^{(l+1)}\right) \tag{11}$$

$$\boldsymbol{\omega}^{(l+1)} = \boldsymbol{\omega}^{(l)} + \frac{h}{2}\left(\boldsymbol{\sigma}^{(l)} + \boldsymbol{\sigma}^{(l+1)}\right) . \tag{12}$$

We reformulated this update to meet the requirements of the implicit contact- and collision distinction explained in [GBF03] and Section 3. In particular, all external forces should be applied in the linear and angular velocity update and not appear in the position and orientation updates.

The restructured Verlet time integration for a single body is obtained by reformulating equations (9) to (12) in terms of

$$\boldsymbol{v}_- = \boldsymbol{v} - \frac{h}{2}\boldsymbol{a}\ , \qquad \boldsymbol{v}_+ = \boldsymbol{v} + \frac{h}{2}\boldsymbol{a}$$

and

$$\boldsymbol{\omega}_- = \boldsymbol{\omega} - \frac{h}{2}\boldsymbol{\tau}\ , \qquad \boldsymbol{\omega}_+ = \boldsymbol{\omega} + \frac{h}{2}\boldsymbol{\tau}\,,$$

which leads to the following 3 step procedure:

1. Update velocities by applying external impulses

$$\boldsymbol{v}_+^{(l)} = \boldsymbol{v}_-^{(l)} + h\,\boldsymbol{a}^{(l)}$$
$$\boldsymbol{\omega}_+^{(l)} = \boldsymbol{\omega}_-^{(l)} + h\,{\mathbf{I}^{(l)}}^{-1}\boldsymbol{\tau}^{(l)}$$

2. Update body locations (done frequently during joint corrections)

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{x}^{(l)} + h\,\boldsymbol{v}_+^{(l)}$$
$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}\left(h\,\boldsymbol{\omega}_+^{(l)} - \frac{h^2}{2}{\mathbf{I}^{(l)}}^{-1}\left(\boldsymbol{\omega}^{(l)} \times \mathbf{I}^{(l)}\boldsymbol{\omega}^{(l)}\right)\right) \circ \boldsymbol{q}^{(l)} \tag{13}$$

3. Update velocities at the end of the time step

$$\boldsymbol{v}_-^{(l+1)} = \boldsymbol{v}_+^{(l)}$$
$$\boldsymbol{\omega}_-^{(l+1)} = \boldsymbol{\omega}_+^{(l)} + \frac{h}{2}\left({\boldsymbol{\sigma}^*}^{(l)} + {\boldsymbol{\sigma}^*}^{(l+1)}\right)$$

The angular acceleration $\boldsymbol{\sigma}^*$ does not include torque directly, meaning that

$$\mathbf{I}\boldsymbol{\sigma}^* = -\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = -\left(\boldsymbol{\omega}_+ - \frac{h}{2}\boldsymbol{\tau}\right) \times \mathbf{I}\left(\boldsymbol{\omega}_+ - \frac{h}{2}\boldsymbol{\tau}\right)$$

This introduces a velocity update at the end of each time step in addition to the application of external impulses (see Figure 2). Note that ${\boldsymbol{\sigma}^*}^{(l+1)}$ in the velocity update is not known and needs to be determined with Newton's method in a similar way as the original Verlet integration described in [KE04].

We neglect torque-dependent terms in the orientation update (we use $\boldsymbol{\omega}_+^{(l)}$ instead of $\boldsymbol{\omega}^{(l)}$ in Equation (13)), because of a more reliable convergence of joint corrections for cases involving strong rotational motors. This way, up to 6 times larger simulation step sizes could be chosen for the trike and Strandbeest examples. At the same time, we still enjoy some of the benefits of the accuracy of Verlet integration such as long-term energy conservation and momentum conservation. Although not as accurate as the Moser Veselov integrator [MV91] that conserves both energy and momentum completely as used by [SKV*12], our simulator is more general in the sense that it features different coefficients of restitution and sustained drift-free articulation and contact constraints.

Appendix B: A heuristic for splitting the kinematic graph

To show, that our splitting strategy yields linear time-complexity, we define the bandwidth $\beta(\mathbf{A})$ as in [BBK05]:

$$\beta(\mathbf{A}) := \max_{1\le i\le n}\{\beta_i(\mathbf{A})\}\ , \quad \beta_i(\mathbf{A}) := i - \min_{1\le j\le i}\{j|\mathbf{A}_{ij} \neq 0\}$$

Let the number of original constraints attached to each of the subparts k be z_k. The z_k can be freely chosen, as long as they add up to the total number of constraints of the original body. The rows corresponding to extra joints have the largest bandwidth (bottom row in Figure 4). Keeping in mind that extra joints constrain 6 degrees of freedom this means $\beta(\mathbf{A}) \le 11 + 2 \cdot z_{\max}$ with $z_{\max} = \max_{1\le k\le n_P}(z_k)$. Both the factorization and the solution has therefore linear time-complexity. Note, that this result can easily be extended to all multibody systems with a tree-like kinematic graph.

In practice, it is desirable to find a splitting strategy that maximizes performance. We found, that in contact-heavy scenarios more computation time of our simulator is spent on solving the system than factorizing it. Therefore the workload is roughly proportional to the number of off-diagonal matrix entries n_M. Considering again the different kinds of matrix entries (sorted by color as in Figure 4 (bottom row)), the number of off-diagonal matrix entries can be written for $n_P \geq 2$ as

$$n_M = \overbrace{\sum_{k=1}^{n_P} z_k(z_k-1)}^{black} + \overbrace{30(n_P-1)}^{red} + \overbrace{72(n_P-2)}^{green} + \underbrace{12(z_0+z_{n_P}) + 24\sum_{k=2}^{n_P-1} z_k}_{blue} \tag{14}$$

We allow breaking apart the constraints between a pair of bodies. For example, joints 3 and 4 in Figure 4 (bottom row) might be attached to the same pair of bodies in the original configuration. For a joint with 6 constraints that is cut in half, this causes a maximum of $3\cdot 3+3\cdot 3=18$ additional matrix entries. Note, that although this introduces a small closed kinematic loop, there is no fill-in during the factorization. For this reason we neglect this additional term and propose a heuristic to minimize n_M.

The number of matrix entries (14) can be written as

$$n_M = \left(\sum_{k=2}^{n_P-1} z_k^2 + 23z_k\right) + z_0^2 + z_{n_P}^2 + 11(z_0+z_{n_P}) + 102n_P - 174\,.$$

Minimizing this function is an integer programming problem. Except for the obvious difference between inner parts and the two end parts, the z_k should be chosen as equal as possible for minimal n_M. Therefore we constrain the number of inner (and outer) constraints per part to only differ by 1 at most. n_M is reformulated in terms of the total number of constraints $n=\sum_{k=1}^{n_P} z_k$, the number of constraints of inner parts $m=\sum_{k=2}^{n_P-1} z_k$ and the number of inner parts $p=n_P-2$. With the ceil and floor functions $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$, let a inner parts contain $z_{lower}=\lfloor\frac{m}{p}\rfloor$, and the others $z_{lower}+1$ constraints. We then have

$$\begin{aligned} z_0+z_{n_P} &= n-m \\ z_0^2+z_{n_P}^2 &= \lceil\frac{(n-m)^2}{2}\rceil \\ \sum_{k=2}^{n_P-1} z_k^2 &= a\,z_{lower}^2 + (p-a)(z_{lower}+1)^2 \\ &= p(\lfloor\frac{m}{p}\rfloor+1)^2 - (\underbrace{p-m \bmod p}_{a})(2\lfloor\frac{m}{p}\rfloor+1)\,. \end{aligned}$$

n_M can be written in a form that yields a lower bound $f(m,p)\leq n_M$:

$$n_M = \underbrace{\frac{1}{2}n^2+11n+\left(\frac{1}{p}+\frac{1}{2}\right)m^2+12m-mn+102p}_{f(n,m,p)}+\gamma$$

with $\quad \gamma\in[0,\lceil\frac{p}{4}\rceil+1]$

A splitting strategy can be obtained by minimizing f for a given n with respect to m and p, yielding

$$p^* = \frac{n-12}{\sqrt{102}}-2$$
$$m^* = \frac{(n-12)p^*}{p^*+2}\,.$$

We found, that this strategy was optimal for $n\in[38,218]$. For $n>78$, we noticed that 16 constraints for the 2 outer parts and 10 for the inner parts seems to always be optimal. To get the maximum number of inner parts with 10 constraints, p needs to be $\lfloor\frac{m}{10}\rfloor$ or $\lfloor\frac{m}{10}\rfloor+1$.

Our heuristic to find p and m that minimize n_M can be given by the following algorithm that takes n as input:

```
if n < 27 // no split
  p = -1
  m = 0
else if n < 37 // one split
  p = 0
  m = 0
else if n < 219 // n < 79 should work, too
  p = n==37 ? 1 : round((n-12)/sqrt{102}-2)
  m = (n-12)*p/(p+2)
else
  m = n-32
  p = floor(m/10)
  if p + 1 - m mod (p+1) < m mod p
    p = p + 1
end if
```

We verified the optimality of this heuristic empirically for $n<10^8$. A thorough proof, however, might be exaggerated at this point, since n_M is not a very accurate performance measure anyway.

Appendix C: An alternative to matrix modification for a single contact

There exist two closely related versions of the Cholesky decomposition of a symmetric positive definite matrix $\mathbf{A}\in\mathbb{R}^{n_C\times n_C}$. The first is defined as

$$\mathbf{A}=\mathbf{G}\mathbf{G}^T\,, \tag{15}$$

where $\mathbf{G}$ is a lower triangular matrix and the second is given by

$$\mathbf{A}=\mathbf{L}\mathbf{D}\mathbf{L}^T\,. \tag{16}$$

$\mathbf{L}$ is also a lower triangular matrix, but has ones on the diagonal and $\mathbf{D}$ is a diagonal matrix.

We show how to add n_a asymmetric rows and columns to the matrix $\mathbf{A}$ (as for the case of Coulomb's model for dynamic friction) based on an existing Cholesky decomposition of $\mathbf{A}$ in the form (15) or (16). For our cases, n_a will not be greater than three, because we only insert one (frictional) contact at a time.

Since we add rows/columns to the bottom/right of $\mathbf{A}$, the decomposition does not need to be changed explicitly. Instead, this change can be emulated during the solution process. All this is done in a block-based manner, handling all n_a constraints at once.
For our derivations, we denote modified parts of the system matrix and the decomposition with $\bar{*}$. The additional n_a constraints yield the new system matrix

$$\bar{\mathbf{A}} = \begin{pmatrix} \mathbf{A} & \bar{\mathbf{A}}_{1,2} \\ \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2} \end{pmatrix} .$$

The dimensions of the new parts are $\bar{\mathbf{A}}_{2,1} \in \mathbb{R}^{n_C \times n_a}$, $\bar{\mathbf{A}}_{1,2} \in \mathbb{R}^{n_C \times n_a}$ and $\bar{\mathbf{A}}_{2,2} \in \mathbb{R}^{n_a \times n_a}$.

First version with $\mathbf{A} = \mathbf{G}\mathbf{G}^T$
The modified decomposition reads

$$\underbrace{\begin{pmatrix} \mathbf{A} & \bar{\mathbf{A}}_{1,2} \\ \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2} \end{pmatrix}}_{\bar{\mathbf{A}}} = \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{0} \\ \bar{\mathbf{G}}_{2,1}^T & \mathbf{1}_{n_a} \end{pmatrix}}_{\bar{\mathbf{G}}} \underbrace{\begin{pmatrix} \mathbf{G}^T & \bar{\mathbf{U}}_{1,2} \\ \mathbf{0} & \bar{\mathbf{D}}_{2,2} \end{pmatrix}}_{\bar{\mathbf{U}}} \tag{17}$$

Note, that the block $\bar{\mathbf{D}}_{2,2} \in \mathbb{R}^{n_a \times n_a}$ is not a diagonal matrix and asymmetric in general, since this is actually a block-based LU decomposition for asymmetric matrices, mixed with a usual Cholesky decomposition. Because the additional rows/columns are added to the bottom/right of $\bar{\mathbf{A}}$, the factor $\mathbf{G}$ does not change. For a purely symmetric modification method that is able to place constraints at arbitrary positions in the matrix, see [DH05].
Matrix equations for the blocks $\bar{\mathbf{A}}_{1,2}$, $\bar{\mathbf{A}}_{2,1}$ and $\bar{\mathbf{A}}_{2,2}$ can be isolated from (17) to compute $\bar{\mathbf{U}}_{1,2}$, $\bar{\mathbf{G}}_{2,1}$ and $\bar{\mathbf{D}}_{2,2}$ once for every contact:

$$\mathbf{G}\bar{\mathbf{U}}_{1,2} = \bar{\mathbf{A}}_{1,2} \quad \Rightarrow \quad \bar{\mathbf{U}}_{1,2} \tag{18}$$

$$\mathbf{G}\bar{\mathbf{G}}_{2,1} = \bar{\mathbf{A}}_{2,1} \quad \Rightarrow \quad \bar{\mathbf{G}}_{2,1} \tag{19}$$

$$\bar{\mathbf{G}}_{2,1}^T\bar{\mathbf{U}}_{1,2} + \bar{\mathbf{D}}_{2,2} = \bar{\mathbf{A}}_{2,2} \quad \Rightarrow \quad \bar{\mathbf{D}}_{2,2} \tag{20}$$

Because $\mathbf{G}$ is a lower triangular matrix, computing $\bar{\mathbf{U}}_{1,2}$ and $\bar{\mathbf{G}}_{2,1}$ from (18) and (19) amounts to n_a forward substitutions each. The computation of $\bar{\mathbf{D}}_{2,2}$ in (20) involves a sparse multiplication $\bar{\mathbf{G}}_{2,1}^T\bar{\mathbf{U}}_{1,2}$ and a subtraction.
The original, unmodified linear system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ can be solved with a forward substitution $\mathbf{G}\boldsymbol{y} = \boldsymbol{b}$, followed by a back substitution $\mathbf{G}^T\boldsymbol{x} = \boldsymbol{y}$. To solve the modified system, we need to first do a forward substitution of

$$\begin{pmatrix} \mathbf{G} & \mathbf{0} \\ \bar{\mathbf{G}}_{2,1}^T & \mathbf{1}_{n_a} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}$$

and then a back substitution of

$$\begin{pmatrix} \mathbf{G}^T & \bar{\mathbf{U}}_{1,2} \\ \mathbf{0} & \bar{\mathbf{D}}_{2,2} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} .$$

More precisely, the overall solution procedure to obtain $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ consists of the following steps:

1. compute $\boldsymbol{y}_1$ by forward substitution of $\mathbf{G}\boldsymbol{y}_1 = \boldsymbol{b}_1$
2. $\boldsymbol{x}_2 \leftarrow \bar{\mathbf{D}}_{2,2}^{-1} \underbrace{\left(\boldsymbol{b}_2 - \bar{\mathbf{G}}_{2,1}^T\boldsymbol{y}_1\right)}_{\boldsymbol{y}_2}$
3. compute $\boldsymbol{x}_1$ by back substitution of $\mathbf{G}^T\boldsymbol{x}_1 = \boldsymbol{y}_1 - \bar{\mathbf{U}}_{1,2}\boldsymbol{x}_2$

Second version of decomposition with $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$
In case the decomposition exists in the form $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, we can solve the modified system in a similar manner. The modified decomposition is given by

$$\begin{pmatrix} \mathbf{A} & \bar{\mathbf{A}}_{1,2} \\ \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2} \end{pmatrix} = \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \bar{\mathbf{L}}_{2,1}^T & \mathbf{1}_{n_a} \end{pmatrix} \begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{D}}_{2,2} \end{pmatrix} \begin{pmatrix} \mathbf{L}^T & \bar{\mathbf{U}}_{1,2} \\ \mathbf{0} & \mathbf{1}_{n_a} \end{pmatrix}$$

This time, the matrices $\bar{\mathbf{U}}_{1,2}$, $\bar{\mathbf{L}}_{2,1}$ and $\bar{\mathbf{D}}_{2,2}$ can be obtained analogously from

$$\begin{aligned} \mathbf{L}\mathbf{D}\bar{\mathbf{U}}_{1,2} = \bar{\mathbf{A}}_{1,2} \quad &\Rightarrow \quad \bar{\mathbf{U}}_{1,2} \\ \mathbf{L}\mathbf{D}\bar{\mathbf{L}}_{2,1} = \bar{\mathbf{A}}_{2,1} \quad &\Rightarrow \quad \bar{\mathbf{L}}_{2,1} \\ \bar{\mathbf{L}}_{2,1}^T\mathbf{D}\bar{\mathbf{U}}_{1,2} + \bar{\mathbf{D}}_{2,2} = \bar{\mathbf{A}}_{2,2} \quad &\Rightarrow \quad \bar{\mathbf{D}}_{2,2} \end{aligned}$$

and the solution procedure reads:

1. compute $\boldsymbol{y}_1$ by forward substitution of $\mathbf{L}\boldsymbol{y}_1 = \boldsymbol{b}_1$
2. $\boldsymbol{x}_2 \leftarrow \bar{\mathbf{D}}_{2,2}^{-1} \underbrace{\left(\boldsymbol{b}_2 - \bar{\mathbf{L}}_{2,1}^T\boldsymbol{y}_1\right)}_{\boldsymbol{y}_2}$
3. compute $\boldsymbol{x}_1$ by back subst. of $\mathbf{L}^T\boldsymbol{x}_1 = \mathbf{D}^{-1}\boldsymbol{y}_1 - \bar{\mathbf{U}}_{1,2}\boldsymbol{x}_2$

Permutation

Due to fill-in minimization, the given factors $\mathbf{G}$ (or $\mathbf{L}$ and $\mathbf{D}$, respectively) are based on a reordered version $\mathbf{A}$ of the original matrix $\mathbf{A}^*$, that can be denoted in terms of a permutation matrix $\mathbf{P}$:

$$\mathbf{A} = \mathbf{P}\mathbf{A}^*\mathbf{P}^T$$

For this reason, the additional matrix parts $\bar{\mathbf{A}}_{1,2}$ and $\bar{\mathbf{A}}_{2,1}$ must be obtained by permuting their original counter-parts before computing $\bar{\mathbf{U}}_{1,2}$, $\bar{\mathbf{L}}_{2,1}$ and $\bar{\mathbf{D}}_{2,2}$:

$$\bar{\mathbf{A}}_{1,2} = \mathbf{P}\bar{\mathbf{A}}_{1,2}^*$$

$$\bar{\mathbf{A}}_{2,1} = \mathbf{P}\bar{\mathbf{A}}_{2,1}^*$$

The given right-hand side $\boldsymbol{b}_1^*$ is permuted accordingly with $\boldsymbol{b}_1 = \mathbf{P}\boldsymbol{b}_1^*$. To obtain the upper part $\boldsymbol{x}_1^*$ of the final solution, the inverse permutation is used:

$$\boldsymbol{x}_1^* = \mathbf{P}^T\boldsymbol{x}_1$$

Of course, the reordering does not need to be performed explicitly. Instead, it is achieved by accessing the elements in the right order, which amounts to a simple book-keeping.