Supplemental to
Adaptive Nonlinearity for Collisions in Complex Rod Assemblies

Danny M. Kaufman — Adobe & Columbia University
Rasmus Tamstorf — Walt Disney Animation Studios
Breannan Smith — Columbia University
Jean-Marie Aubry — Weta Digital
Eitan Grinspun — Columbia University

Here we provide details of the pseudocode and our implementation of ADONIS – our adaptively nonlinear solver for rod assemblies. In what follows we employ superscripts to denote time. The subscripts j and k are reserved to denote contact indices, while the subscripts x and y are reserved for indexing rods and collision meshes. At each solve we have ℓ rods with configurations q_x, velocities $\dot{\mathsf{q}}_x$, displacements δ_x, masses M_x, and internal energies $V_x(\mathsf{q}_x)$. The rod assembly is subject to a contact set $\mathbb{K}$ of $|\mathbb{K}| = m$ detected contacts. Each contact is generated by either the contact between two rods, a rod and a collision mesh, or else by self contact. Each $k \in \mathbb{K}$ then records the global indices of the rods and/or collision meshes in contact, the local indices of the vertices, edges, and/or faces forming the contact, and, if CCD is employed, the time of collision. Quantities undecorated by a subscript are used to denote the full system variables composed by concatenation, e.g., $\mathsf{q} = (\mathsf{q}_1^T, ..., \mathsf{q}_\ell^T)^T \in \mathbb{R}^n$, $\mathsf{M} = \text{diag}(\mathsf{M}_1, ..., \mathsf{M}_\ell)$, and $\mathsf{G} = (\mathsf{G}_1, ..., \mathsf{G}_m) \in \mathbb{R}^{n\times 3m}$, and summation, e.g., $V(\mathsf{q}) = \sum_x V_x(\mathsf{q}_x)$; while H denotes the Hessian of V. A single time step of our algorithm then proceeds as given in Algorithm 1.

A number of additional points in the algorithm warrant further discussion:

Alg. 1, steps 3-4 These initial LU factorizations are readily available from the previously completed unconstrained solve in step 2. Rod Hessians are banded matrices with a width of eleven. We employ a sequentially banded LU factorization method dgbtrf (http://www.netlib.org/lapack/double/dgbtrf.f) for LU factorization of these Hessians.

Alg. 1, step 6 Collision detection generates a contact graph that is then subdivided into its connected components or *contact groups*. Each contact group is solved as an independent contact problem. For clarity of presentation and the avoidance of further subscripting we do not include this detail in our pseudocode. However, the chief difference is simply that each contact group (or "island") independently follows the pseudocode from line 7 onwards and is computed in parallel.

Alg. 1, steps 6-7 Broad-phase updates, narrow-phase collision detection queries, and contact point processing to build constraints are all performed in parallel.

Alg. 1, step 7 `Build Constraints` builds the contact frames for proximity detected contacts at the start of step position q^t, while the contact frames for CCD detected contacts are constructed at the linearly predicted collision states. The scripted displacements of the collision meshes at the detected points of contact, along the contact frame directions, at estimated times of collision, then give the offsets s.

Alg. 1, step 9 `Initialize Contact Force` can either set λ to 0 or else warm start based on tracking contact coherence over time steps. In our code we employ the latter warm starting strategy but have so far, in limited testing, observed little advantage in employing the warm starting except when rods are close to rest. Further investigation is certainly warranted.

Alg. 1, step 17 `Contact Error`(λ) returns the ∞-norm over all individual contact subproblem residuals.

Algorithm 1 `ADONIS`$(\mathsf{q}^t, \dot{\mathsf{q}}^t, h, t)$

1: **for** x in $\{1, ..., \ell\}$ **do** in parallel
2: $\quad\delta_x \leftarrow$ `solve DEL`$(\mathsf{q}_x^t, \dot{\mathsf{q}}_x^t, h)$
3: $\quad\mathsf{H}_x \leftarrow H_x(\mathsf{q}_x^t + \delta_x)$
4: $\quad\{\mathsf{L}_x, \mathsf{U}_x\} \leftarrow$ `LUFactorize`$\big(\mathsf{M}_x + h^2\mathsf{H}_x\big)$
5: **end for**
6: $\mathbb{K} \leftarrow$ `Collision Detection`(δ, q^t)
7: $\{\mathsf{G}, \mathsf{s}\} \leftarrow$ `Build Constraints`$(\mathbb{K}, \delta, \mathsf{q}^t)$
8: $\mathbb{S} \leftarrow \mathbb{K}$
9: $\lambda \leftarrow$ `Initialize Contact Force`$()$
10: **while** $\mathbb{S} \neq \emptyset$ **do**
11: $\quad\mathsf{P} \leftarrow$ `Assemble Compliance`$(\mathbb{S})$
12: $\quad$**for** x in $\{1, ..., \ell\}$ **do** in parallel
13: $\quad\quad\mathsf{b}_x \leftarrow h\mathsf{M}_x\dot{\mathsf{q}}_x^t - h^2\nabla V_x(\mathsf{q}_x^t + \delta_x) + h^2\mathsf{H}_x\delta_x$
14: $\quad\quad\delta_x \leftarrow$ `LUSolve`$(\mathsf{L}_x, \mathsf{U}_x, \mathsf{b}_x)$
15: $\quad$**end for**
16: $\quad$`gs_itr` $\leftarrow 0$
17: $\quad$**while** `Contact Error`$(\lambda) >$ `contact_tol`
18: $\quad\quad$**&** `gs_itr` $<$ `gs_max` **do**
19: $\quad\quad$**for** k in $\mathbb{K}$ **do**
20: $\quad\quad\quad\mathsf{c} \leftarrow \sum_{j\in\mathbb{K}\neq k} \mathsf{P}_j\lambda_j$
21: $\quad\quad\quad\lambda_k \leftarrow$ `solve` $: \lambda_k \in \mathcal{R}_k(\delta + \mathsf{P}_k\lambda_k + \mathsf{c})$
22: $\quad\quad$**end for**
23: $\quad\quad$`gs_itr` $\leftarrow$ `gs_itr` $+ 1$
24: $\quad$**end while**
25: $\quad\delta \leftarrow \delta + \sum_{j\in\mathbb{K}} \mathsf{P}_j\lambda_j$
26: $\quad\mathbb{S} \leftarrow \emptyset$
27: $\quad$**for** x in $\{1, ..., \ell\}$ **do** in parallel
28: $\quad\quad$**if** `Stretch`$(x) >$ `stretch_tol` **then**
29: $\quad\quad\quad\mathsf{H}_x \leftarrow H_x(\mathsf{q}_x^t + \delta_x)$
30: $\quad\quad\quad\{\mathsf{L}_x, \mathsf{U}_x\} \leftarrow$ `LUFactorize`$\big(\mathsf{M}_x + h^2\mathsf{H}_x\big)$
31: $\quad\quad\quad\mathbb{S} \leftarrow \mathbb{S} \cup$ `Get Contacts For Rod`(x)
32: $\quad\quad$**end if**
33: $\quad$**end for**
34: **end while**
35: $\dot{\mathsf{q}}^{t+1} \leftarrow \frac{1}{h}\delta$
36: $\mathsf{q}^{t+1} \leftarrow \mathsf{q}^t + \delta$
37: **return** $(\mathsf{q}^{t+1}, \dot{\mathsf{q}}^{t+1})$

Algorithm 2 `Collision Detection`(δ, q^t)

1: **if** `rod_ccd` **then**
2: $\quad\mathbb{K} \leftarrow$ rod-rod CCD on the trajectory from q^t to $\mathsf{q}^t + \delta$
3: **else**
4: $\quad\mathbb{K} \leftarrow$ rod-rod proximity collision-detection at q^t
5: **end if**
6: $\mathbb{K} \leftarrow \mathbb{K} \cup$ rod-mesh CCD on the trajectory from q^t to $\mathsf{q}^t + \delta$
7: **return** $\mathbb{K}$

Algorithm 3 `Stretch`(x)

1: $\text{sf}_x \leftarrow \max_i \big(|\boldsymbol{e}_i|/|\bar{\boldsymbol{e}}_i| - 1\big), \forall$ edges i in rod x
2: **return** sf_x

Algorithm 4 `Assemble Compliance`$(\mathbb{K})$

1: **for** k in $\mathbb{K}$ **do**
2: $\{x, y\} \leftarrow$ `Get Contacting Objects Indices`(k)
3: **if** $x = y$ or y is a collision mesh **then**
4: $\mathsf{P}_k \leftarrow$ `BlockLUSolve`$(x, \mathsf{L}_x, \mathsf{U}_x, \mathsf{G}_k)$
5: **else**
6: $\mathsf{P}_k \leftarrow$ `BlockLUSolve`$(x, \mathsf{L}_x, \mathsf{U}_x, \mathsf{G}_k)$
7: $\mathsf{P}_k \leftarrow$ `BlockLUSolve`$(y, \mathsf{L}_y, \mathsf{U}_y, \mathsf{P}_k)$
8: **end if**
9: **end for**
10: **return** $\mathsf{P} = \{\mathsf{P}_1, ..., \mathsf{P}_m\}$

Alg. 1, steps 16-24 The Gauss-Seidel loop here is parallelized by graph-color partitioning. I.e., all contacts are colored such that no two contacts with overlapping stencils have the same color. Our convergence tolerance is `contact_tol` $= 10^{-6}$.

Alg. 1, step 21 We use the hybrid solver of Daviet et al. [2011].

Alg. 1, step 31 `Get Contacts For Rod`(x) returns the contacts that rod x is involved in.

Alg. 4, step 2 `Get Contacting Objects Indices`(k) returns the indices of the rods and/or collision meshes that form contact k. If $x = y$ we have a self collision. Collision meshes, when present, are given by the second index y.

Alg. 4, steps 4 & 6-7 The three columns of each $\mathsf{G}_k \in \mathbb{R}^{n\times 3}$ are sparse with non-zero entries in a small subset of the rows corresponding to the DoFs of either one (for rod/mesh or self contact) or two (in contact) rods. In lines 4 and 6 `BlockLUSolve` applies an LU solve to the $|\mathsf{q}_x| \times 3$ block in G_k corresponding to rod x's DoFs, while in line 7 an analogous solve is applied to the $|\mathsf{q}_y| \times 3$ block in P_k corresponding to rod y's DoFs. As the solves in 6 and 7 are on disjoint elements of G_k they can be performed in parallel.

References

DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. 2011. A Hybrid Iterative Solver for Robustly Capturing Coulomb Friction in Hair Dynamics. *ACM Trans. Graph. 30*, 6 (Dec.), 139:1–139:12.