Comparison to CMA-ES

In this section we show convergence plots of optimizations using our differentiable simulation framework compared to a gradient-free CMA-ES approach. We use the CMA implementation by Nikolaus Hansen[1] in the first comparison (Fig. 1), and the implementation by Alexander Fabisch[2] in the second one (Fig. 2); both with their default parameter values.

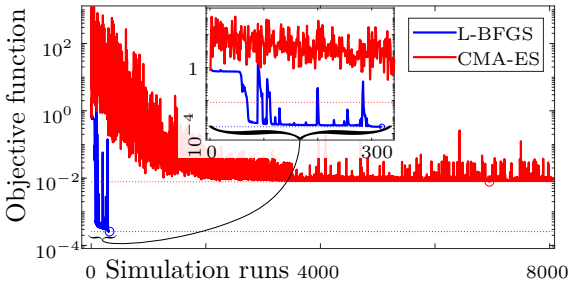

Figure 1: Optimizing throwing the bunny to a specific target. Graph shows all simulation runs, including ones during line search for L-BFGS. Circles mark best result per method.

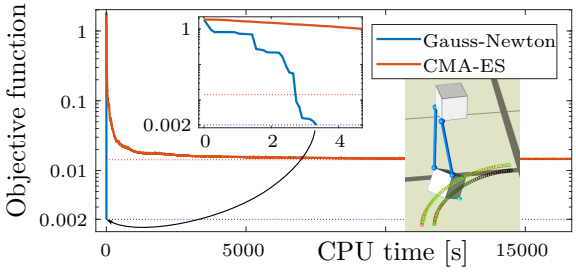

Figure 2: Optimizing control trajectories for dragging a cube along a circular arc. Graph shows best result per iteration (filtering out line-search evaluations for Gauss-Newton, and sub-optimal samples per generation for CMA).

Scaling of computation time

Here, we test how the computation time scales with increasing number of objects in a simulation scene. In this test case, a chain of N rigid bodies is simulated. The first rigid body is connected to two springs anchored in world space. Similarly, each following rigid body is connected to the previous one by two springs. For all rigid bodies, collision ($k_n = 10^3$, $k_d = 10^{-3}$) and friction ($\mu = 0.5$) with the ground plane is enabled. The total simulated time is $t = 2s$ with a time step of $\Delta t = 1/500$ s.

Figure 3 shows the resulting computation time for $N = 1..40$, exhibiting fairly linear scaling.

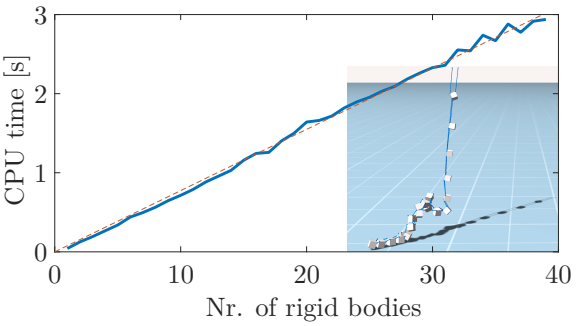

Figure 3: Computation time per number of rigid bodies in a "chain" setup. The inset image shows a screenshot of this test with $N = 20$.

Number of contact points

The number of contact points influences the collision response. Increasing the number of contact points, while keeping the contact penalties (k_n, k_d) fixed, results in a stiffer collision response. Conversely, when scaling the penalties by the surface area each contact point represents, we arrive at the same total contact force regardless of the number of contact points (for a planar collision geometry). In Fig. 4 the restitution is plotted against the number of contact points per edge of a rigid cube. For each sample we scale the penalty factor according to the surface area each contact point represents, e.g. $k_{n,N} = k_{n,0}/N^2$ and $k_{d,N} = k_{d,0}/N^2$ for N contact points per edge, where $k_{n,0}$, $k_{d,0}$ are nominal stiffness and damping coefficients.

High-stiffness elastic material

In this example we show an optimization using continuation of the penalty stiffness ($k_n = 100..10^6$) such that a deformable cube made of a stiff elastic material (shear modulus $\mu = 10$ MPa) reaches a given target position. The optimization must find the initial velocity, such that the cube drops to the ground and then slides to the target.

[1] `https://github.com/cma-es/c-cmaes`
[2] `https://github.com/AlexanderFabisch/CMA-ESpp`

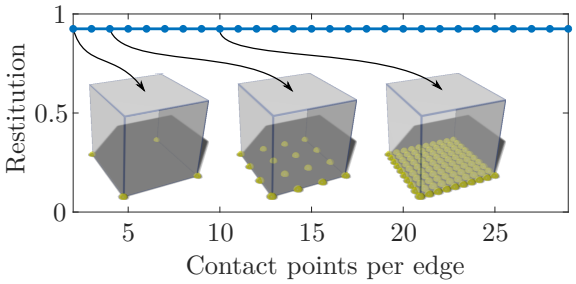

Figure 4: Restitution remains constant as the number of contact points increases when scaling penalty factors by surface area. Inset images show the contact point distribution for 2, 4, and 10 points per edge respectively.

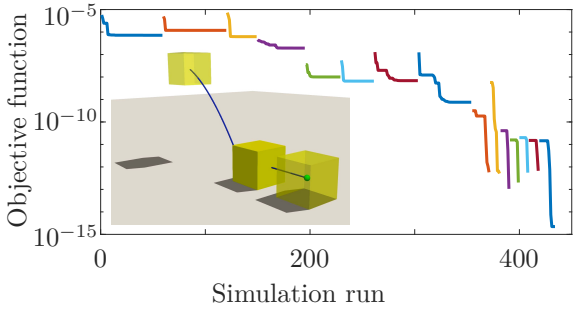

Figure 5: Best objective function value per simulation run using continuation on the contact penalty factor. In each continuation step (colours), the penalty factor doubles, starting from 100, up to the final value of 10^6. The inset image shows the final result.

Rigid body theory

Parameterization of rotation

We parameterize the rotation of a rigid body with exponential coordinates $\boldsymbol{\theta}$, using the exponential map $\mathbf{R}(\boldsymbol{\theta}) = \lim_{n\to\infty}(\mathbf{I} + [\boldsymbol{\theta}])^n$. For $||\boldsymbol{\theta}|| > \epsilon$ we compute $\mathbf{R}(\boldsymbol{\theta})$ with the Euler-Rodrigues formula, where we use $\epsilon = 10^{-8}$ in all our examples. For $||\boldsymbol{\theta}|| \leq \epsilon$, we use the first-order approximation to the exponential map $\mathbf{R}(\boldsymbol{\theta}) = \mathbf{I} + [\boldsymbol{\theta}]$. The derivative of $d\mathbf{R}/d\boldsymbol{\theta}$ is computed analytically using the derivation by Gallego and Yezzi [2015], and for $||\boldsymbol{\theta}|| \leq \epsilon$ we use $d\mathbf{R}/d\boldsymbol{\theta} = [\mathbf{I}]$. The second-order derivative of the rotation matrix is computed using symbolic differentiation.

Rigid bodies

A rigid-body frame is defined by its centre of mass $\mathbf{c}$ and rotational degrees of freedom $\boldsymbol{\theta}$. A point $\hat{\mathbf{x}}$ in rigid-body coordinates is transformed to world coordinates with $\mathbf{x} = \mathbf{c} + \mathbf{R}(\boldsymbol{\theta})\hat{\mathbf{x}}$. Similarly, $\mathbf{w} = \mathbf{R}(\boldsymbol{\theta})\hat{\mathbf{w}}$ maps a vector $\hat{\mathbf{w}}$ from rigid-body to world coordinates.

The linear velocity $\mathbf{v}$ is equal to the rate of change of the rigid body's centre of mass $\mathbf{v} = \dot{\mathbf{c}}$. The angular velocity $\boldsymbol{\omega}$, however, describes the rate $\|\boldsymbol{\omega}\|$ at which an object rotates around an axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\|$, and thus $\boldsymbol{\omega} \neq \dot{\boldsymbol{\theta}}$. To relate the angular velocity $\boldsymbol{\omega}$ to the rate of change of rotational degrees of freedom, $\dot{\boldsymbol{\theta}}$, consider a vector $\mathbf{a}$: its rate of change due to an angular velocity $\boldsymbol{\omega}$ is computed as $\dot{\mathbf{a}} = \boldsymbol{\omega} \times \mathbf{a}$. Since the column vectors of $\mathbf{R}$ are just the axes of the rigid body's coordinate frame, $\dot{\mathbf{R}}$ can therefore be computed with

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]\mathbf{R},$$

where $[\cdot]$ denotes the skew-symmetric matrix corresponding to the left-cross product. $\mathbf{R}$ being an orthogonal matrix, we find

$$[\boldsymbol{\omega}] = \dot{\mathbf{R}}\mathbf{R}^\mathsf{T} = \sum_j \frac{\partial \mathbf{R}}{\partial \theta_j}\mathbf{R}^\mathsf{T}\dot{\theta}_j = \sum_j [(\mathbf{J}_\omega)_j]\dot{\theta}_j,$$

where $(\mathbf{J}_\omega)_j$ is the j-th column vector of the Jacobian $\mathbf{J}_\omega$, representing the skew-symmetric matrix $(\partial\mathbf{R}/\partial\theta_j)\mathbf{R}^\mathsf{T}$.

Note that $\mathbf{J}_\omega$ is a function of $\boldsymbol{\theta}$. We can consequently write the angular velocity in terms of the generalized coordinates and their time derivative:

$$\boldsymbol{\omega}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{J}_\omega(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$$

Rigid-body dynamics

Using the above mapping from $\dot{\boldsymbol{\theta}}$ to $\boldsymbol{\omega}$, we can write the Newton-Euler equations in the form of Eq. (1), where the generalized mass and forces are

$$\hat{\mathbf{M}} = \begin{pmatrix} m\mathbf{I} & 0 \\ 0 & \mathbf{J}_\omega^\mathsf{T}\mathbf{I}_c\mathbf{J}_\omega \end{pmatrix}$$

and

$$\hat{\mathbf{f}} = \begin{pmatrix} \mathbf{I} \\ \mathbf{J}_\omega^\mathsf{T}[\hat{\mathbf{x}}] \end{pmatrix}\mathbf{f} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}),$$

respectively. The first term in $\hat{\mathbf{f}}$ maps forces from the body's surface to generalized coordinates and the fictitious force term $\mathbf{C}$ is defined as

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 \\ \mathbf{J}_\omega^\mathsf{T}\mathbf{I}_c\dot{\mathbf{J}}_\omega\dot{\boldsymbol{\theta}} + \mathbf{J}_\omega^\mathsf{T}[\mathbf{J}_\omega\dot{\boldsymbol{\theta}}]\mathbf{I}_c\mathbf{J}_\omega\dot{\boldsymbol{\theta}} \end{bmatrix}.$$

This term appears because $\dot{\boldsymbol{\theta}}$ is *not* the same as the angular velocity $\boldsymbol{\omega}$.

To solve rigid-body dynamics implicitly, we need to compute the following derivatives:

$$\left[\frac{\partial(\mathbf{J}_{\boldsymbol{\omega}})_j}{\partial\theta_i}\right]_\times = \frac{\partial^2\mathbf{R}}{\partial\theta_j\partial\theta_i}\mathbf{R}^\mathsf{T} + \frac{\partial\mathbf{R}}{\partial\theta_j}\frac{\partial\mathbf{R}}{\partial\theta_i}^\mathsf{T},$$

$$\begin{aligned}\left[\frac{\partial^2(\mathbf{J}_{\boldsymbol{\omega}})_j}{\partial\theta_i\partial\theta_k}\right]_\times =& \frac{\partial^3\mathbf{R}}{\partial\theta_j\partial\theta_i\partial\theta_k}\mathbf{R}^\mathsf{T} + \frac{\partial^2\mathbf{R}}{\partial\theta_j\partial\theta_i}\frac{\partial\mathbf{R}}{\partial\theta_k}^\mathsf{T} \\ &+ \frac{\partial^2\mathbf{R}}{\partial\theta_j\theta_k}\frac{\partial\mathbf{R}}{\partial\theta_i}^\mathsf{T} + \frac{\partial\mathbf{R}}{\partial\theta_j}\frac{\partial^2\mathbf{R}}{\partial\theta_i\partial\theta_k}^\mathsf{T},\end{aligned}$$

and

$$\frac{\partial\dot{\mathbf{J}}_{\boldsymbol{\omega}}}{\partial\theta_i} = \sum_j \frac{\partial^2\mathbf{J}_{\boldsymbol{\omega}}}{\partial\theta_j\theta_i}\dot{\boldsymbol{\theta}}.$$

The rotated moment of inertia is computed as $\mathbf{I}_c = \mathbf{R}^\mathsf{T}\mathbf{I}_0\mathbf{R}$ (where $\mathbf{I}_0$ is the inertia in rigid-body coordinates). Its derivative with respect to $\boldsymbol{\theta}$ is

$$\frac{\partial\mathbf{I}_c}{\partial\theta_i} = \frac{\partial\mathbf{R}}{\partial\theta_i}\mathbf{I}_0\mathbf{R}^\mathsf{T} + \mathbf{R}\mathbf{I}_0\frac{\partial\mathbf{R}}{\partial\theta_i}^\mathsf{T}.$$