IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Minor Change, Major Gains: The Effect of Orientation Formulation on Solving Time for Multi-body Trajectory Optimization

Alexander Knemeyer*[1], Stacey Shield[1], Amir Patel[1],

***Abstract*— Many different coordinate formulations have been established to describe the position of multi-body robot models, but the impact of this choice on the tractability of trajectory optimization problems has yet to be investigated. Relative formulations, which reference the position of each link to its predecessor, reduce the number of variables and constraints in the problem, but lead to cumbersome expressions for the equations of motion. By contrast, referencing the positions to an absolute frame simplifies these equations, but necessitates more coordinate variables and connection constraints. In this paper, we investigate whether changing the orientation coordinates of a multi-body system model from relative to absolute angles can reduce the time required to solve the problem. The two approaches are tested on a variety of two- and three-dimensional models, with and without unscheduled unilateral contacts. Across all cases, the absolute formulation was found to be the more successful option. The performance improvements increased with the complexity of the system and task, culminating in the challenging example of a 60-degree turn on a 3D quadruped model, which was only able to converge in the allotted time when absolute angles were used.**

I. Introduction

Trajectory optimization is a valuable tool that roboticists and biologists alike use to investigate motion. It has a variety of proven use-cases, including planning whole-body manoeuvres [1][2][3], identifying efficient gaits that exploit the body's passive dynamics [4][5], devising control policies for agile maneuvers [6], designing optimal robot morphologies [7][8][9] and exploring more conceptual questions about locomotion [10]. The major drawback, especially when more elaborate whole-body models or long time horizons are involved, is solving time.

Innovations such as improved integration methods [11], contact-invariant approaches [12] and warm-starts with simpler models [13] have made the generation of accurate representations of poorly-specified, intricate tasks on detailed models feasible, but an aspect of trajectory optimization that has received little attention regarding performance improvement is the kinematic formulation of the model itself.

Describing the robot's position in the joint space has the advantage of producing a minimal set of coordinate variables and ensuring that the links in the system remain connected without the need for explicit positional constraints, but it produces complicated Coriolis terms when used to describe long serial chains of links. While it has been observed that removing these problematic terms can allow the model to solve faster [14], neglecting them becomes detrimental to the accuracy of the simulation as the movement becomes more rapid and dynamic.

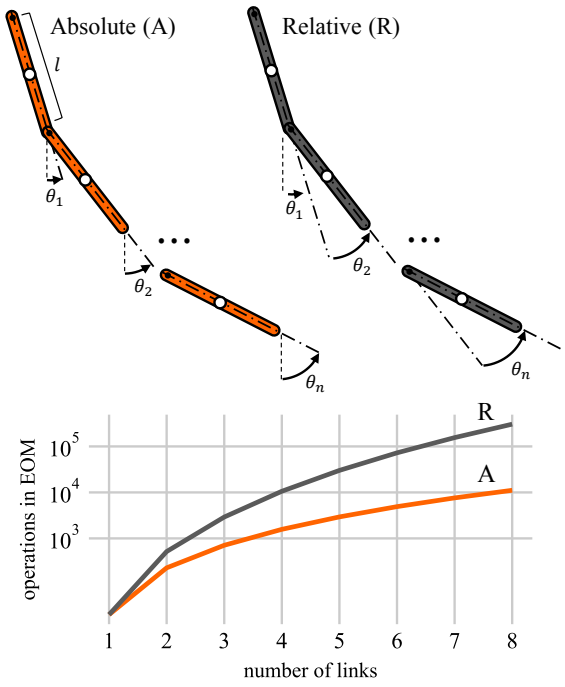

Fig. 1. Diagram of a 2D n-link pendulum, contrasting the absolute (A) and relative (R) angle formulations. The plot compares the number of operations in the symbolic equations of motion (EOM) as the number of links increases.

Formulations that reference positions and orientations to the inertial frame, rather than relative to preceding links, result in much simpler expressions for the equations of motion (EOM), at the cost of many more coordinate variables and explicit connection constraints. This is illustrated in Fig. 1, which shows how many operations make up the symbolic EOM for planar pendulums of increasing length using two different orientation formulations. When the angles are referenced to a world frame, each subsequent link adds far fewer operations to the computational burden.

In this paper, we compare the performance of trajectory optimization problems formulated using relative versus absolute orientation coordinates. First, we discuss the differences between the approaches we are considering and how they might impact the tractability of the problem. We then test the solving times using three models, each of which adds a

new aspect to the challenge:

1) long serial chains,
2) unscheduled foot-ground contacts,
3) 3D locomotion.

Finally, we use a complicated manoeuvre on a high degree-of-freedom (DOF) system to demonstrate that this seemingly-minor adjustment affects the performance enough to make or break the successful convergence of a demanding problem.

These experiments contribute to the trajectory optimization literature by showing that the choice of coordinates is a significant aspect of problem design, and guiding readers towards approaches that are conducive to fast, reliable solving. While the examples we chose are specific to legged robotics, the results are relevant to a broad range of motion planning and optimal control problems.

II. Multi-body dynamics

Changing the orientation formulation of a model affects two aspects of the trajectory optimization problem: the equations of motion, and the joint constraints necessary to restrict the movement of rigid bodies relative to each other.

A. Equations of motion

The dynamics of a rigid multi-body system are often expressed in the manipulator equation form:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{G} = \mathbf{B}\mathbf{u} + \mathbf{J}_L^T \lambda \tag{1}$$

where $\mathbf{q}$ is the vector of generalized coordinates for the system, $\mathbf{M}$ represents the mass matrix, $\mathbf{C}$ the Coriolis and centrifugal matrix, $\mathbf{G}$ the gravitational force, $\mathbf{B}$ the input mapping, and $\mathbf{u}$ the generalized input. Interactions with objects and the environment are captured by the contact Jacobian $\mathbf{J}_L$ and the contact forces λ. n and p represent the number of the input forces and contact points respectively.

Using an absolute orientation formulation makes the elements in these matrices easier for a nonlinear programming (NLP) solver to process in two ways: it simplifies them - that is, reduces the number of operations required to calculate them (as shown in Fig. 1) - and improves their sparsity.

These improvements can be credited to changes in the expressions relating the model's coordinates to the world frame. Consider the n-link pendulum in Fig. 1: when relative angles are used, the absolute orientation of the nth link must be calculated by adding the orientations of all previous links. This leads to a Jacobian where the elements mapping joint angles to translational degrees of freedom depend on all preceding joint angles. Most importantly from the solver's point-of-view, this causes the equations of motion and contact equations to have many more nonzero partial derivatives.

When the orientation is already expressed in the world frame, the corresponding Jacobian elements each depend on only one angular position, and therefore, one non-zero partial derivative is created where there would otherwise have been n. As an illustration of how these improvements carry over to the equations of motion, Fig. 2 compares the sparsity of each of the Coriolis terms $\mathbf{C}\dot{\mathbf{q}}$ for a 4-link planar pendulum.

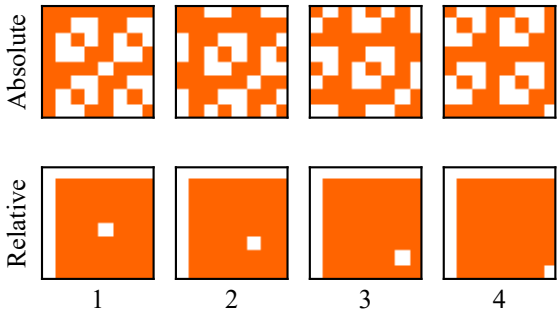

Fig. 2. Sparsity of the Hessians of the Coriolis terms for each link of a planar 4-link pendulum arising from relative and absolute orientation formulations.

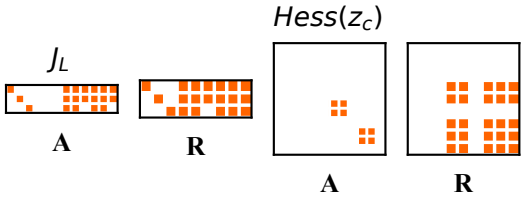

Fig. 3. Sparsity patterns for the contact Jacobian J_L and Hessian of the contact height z_c for the 3D hopper using absolute (A) and relative (R) orientation formulations.

Especially because contact interactions typically occur at a robot's extremities, simplifying the Jacobian also results in a much more tractable contact model.

Fig. 3 compares the sparsity patterns for the contact Jacobians, and the Hessians of the foot height for the 3D monoped model described later in this paper. Depending on how the model is configured, the Jacobian itself will not necessarily have fewer non-zero elements, expressions defining the auxiliary contact variables will have sparser derivatives. The work of Geilinger, et al. [15] demonstrates how a more favourable Hessian can have a notable impact on the convergence of an optimal legged locomotion problem.

B. Joint constraints

In two dimensions, changing the orientation representation to an absolute frame does not require any further modification to the broader formulation: it is as simple as depicted in Fig. 1. In 3D, however, motion constraints that are implicit in the relative version must be included explicitly, together with the constraint forces necessary to facilitate them [16]. These forces are added to the state vector at each collocation point, and chosen by the solver as needed to satisfy the constraints. In the absence of added joint constraints, all connections behave as spherical "ball-and-socket" joints.

This is no longer a minimal coordinate representation: a body connected to its predecessor by a single-DOF rotary

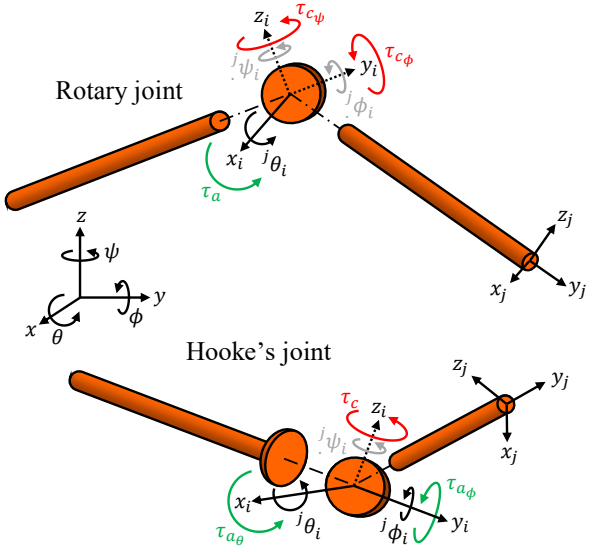

Fig. 4. Variables associated with two types of joint used in 3D systems. The additional angular coordinates required in an absolute formulation are noted in grey, and the constraint torque variables are shown in red.

joint requires an additional two coordinates when its orientation is expressed in world-frame terms. Redundant coordinates managed by connection constraints and constraint forces are an aspect this approach shares with the maximal coordinate representations used in areas such as computer graphics [17]. Since the translational coordinates are still referenced recursively, the connections between links remain implicit, and there is no need for the positional constraints that would be required to keep the links joined in a fully global formulation.

The canonical form of the manipulator equation, modified to include these constraint torques, is as follows:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{G} = \mathbf{B}\mathbf{u} + \mathbf{J}_L^T\lambda + \mathbf{J}_c^T\mathrm{T}_c \tag{2}$$

where T_c is a vector of constraint forces and torques mapped into the equations of motion via the Jacobian of the angle constraints, $\mathbf{J}_c$. Note that, although we use the same names for these components because they fulfil the same general roles, the matrices in the modified manipulator equation are not the same as those in (1) - they are the ones generated using absolute referencing. Similarly, to preserve familiarity, we use $\mathbf{q}$ to symbolize the vector of coordinates including the absolute angles, not the joint-space coordinates.

The following examples describe the constraints that must be added for two kinds of joint, shown in Fig. 4 and used in models in this paper. A rigorous treatment with more joint types can be found in [18].

1) Rotary joint: A rotary joint (also known as a hinge or revolute joint) allows to bodies to rotate about a common line $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_j$ (using the example in Fig. 4). This type of joint might be used to remove two degrees of freedom from a knee in a biped. The constraints to enforce this are:

$$\hat{\mathbf{x}}_i \cdot \hat{\mathbf{y}}_j = 0 \qquad \hat{\mathbf{x}}_i \cdot \hat{\mathbf{z}}_j = 0 \tag{3}$$

They are implemented by means of the constraint torques $\tau_{c\phi}$ and $\tau_{c\psi}$ that act on the restricted degrees of freedom.

2) Hooke's joint: A Hooke's joint (also known as a universal joint) permits two successive rotations between rigid bodies. A defining result is that lines $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$ remain perpendicular. This type of joint might be used to remove one degree of freedom from a hip in a quadruped. The constraint to enforce this is calculated using,

$$\hat{\mathbf{y}}_i \cdot \hat{\mathbf{x}}_i = 0 \tag{4}$$

Once again, the constraint torque τ_c is required to make the constraint feasible.

Particularly for systems that largely consist of single-DOF joints, the number of decision variables in the problem will be much larger when the absolute formulation is used. Considering positions, velocities and accelerations for each coordinate, the formulation change adds eight variables for each rotary joint. It also adds more constraints, as range-of-motion restrictions that could be imposed with variable bounds in a relative formulation now require equations to specify. Depending on the NLP solving algorithm used, this change from variable bounds to constraint equations could affect the way these constraints are handled.

The possible advantage of absolute over relative coordinates hinges on one question: *will the simplified expression of the dynamics offset this many-fold increase in problem size?*

III. Method

In this section, we describe the approach to trajectory optimization used for the experiments that follow. The models used and the tasks they perform are described in the context of each experiment. In each case, we compare the same system using relative angles against the absolute angle formulation. Some familiarity on the subject is assumed - for a detailed introduction, the reader is referred to the tutorial in [19].

A. Direct collocation

Direct collocation is used to formulate the dynamics constraints without the need for forward integration as used by shooting methods, which have known disadvantages [19]. Specifically, we use the Radau quadrature-based approach described in [11]. This has an accuracy of order $\mathcal{O}\left((h^{2K-1}\right)$ [20] where h is the duration of each time step and K is the number of collocation points in each finite element. We use a three-point formulation.

To improve performance, we use a two-stage solving approach: first, a coarse solution is generated using implicit Euler integration, then the values at each element are used to seed the values at all three collocation points of the corresponding element for the subsequent Radau stage.

B. Contact-invariant Approach

Some of the models incorporate unilateral contact constraints to model foot-ground impacts. We enforce these

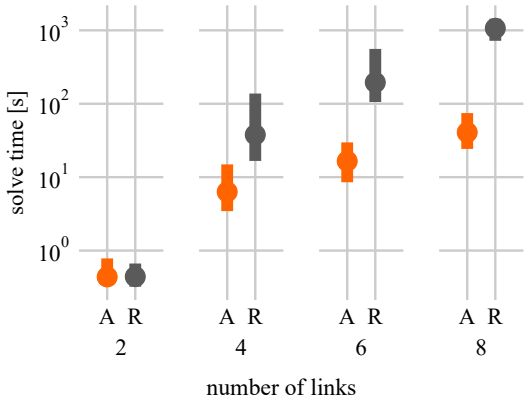

Fig. 5. Solving time for planar n-link pendulums, using absolute (A) and relative (R) angle formulations. The circle and bar represent the median and interquartile range, respectively.

through complementarity constraints, using the method described in [12] adapted to work with higher-order collocation [11]. This models the contacts as inelastic collisions and incorporates sliding corresponding to a Coulomb coefficient of friction μ. We use a penalty method to make the associated equality constraints more tractable for the solver [21]. This involves setting the jth such constraint at the ith collocation point equal to some penalty variable $p_i j$, and then minimizing the sum of these penalties (P) to below the constraint tolerance as a term in the objective function. A tunable scalar ρ is used to weight P to at least two orders of magnitude larger than the objective term.

A variable time step is also used in all problems involving such constraints. For a problem consisting of N elements, this is implemented by using a nominal timestep $h_n = \frac{T}{N}$, where T is a reasonable approximation of the total time required for the manoeuvre, and allowing the timestep h_i at each element to take on values within 20 percent of h_n. Since the contact states can only change from timestep to timestep, this eliminates some unnatural movements which might otherwise arise.

C. Initialization

Unless otherwise specified, the state variables of all models are initialized from vectors of small random values prior to the first-order pre-solve stage. The same seeds are applied to both configurations in each test, to prevent the performance of either from being biased by an especially poor starting point. In general, ten random seeds were attempted for each configuration.

D. Solving

All experiments were written in the Python optimization library Pyomo [22][23]. The NLP solver IPOPT [24] with the linear solver MA86 [25] was used for all tests. The solving times stated were observed on an AMD Ryzen 7 1700 Eight-Core Processor. Animations showing sample results from each experiment are included in the supplementary video.

IV. Experiments

A. Planar n-link pendulum swing-up

The first test system is a planar n-link pendulum, shown in Fig. 1. There was no actuator at the base joint, and the torque at all other joints was limited to the product of the weight and length of one link.

1) Experiment: The model performed a swing-up maneuver from rest, minimizing

$$J = \sum_{j=2}^{m} \sum_{i=1}^{N} \tau_{ji}^2 \tag{5}$$

where τ_{ji} is the torque acting at the top of the jth link at element i. The swing-up was specified by having all links start at rest at zero radians, and end at rest at π radians relative to the inertial frame. A fixed time of 3 seconds discretized into $N = 50$ elements was allocated for the motion.

2) Results: The results in Fig. 5 show that the absolute angle formulation solves significantly faster, and scales better with an increasing number of links. We observed that both formulations required around the same number of algorithm iterations to solve, with each iteration for the absolute angle model completing far faster. This is further supported by the solver log files, which show an average of 86% of the solving time for the relative-angle model being spent on NLP function evaluations, compared to 52% for the absolute-angle model. This mirrors the observation about the number of operations in the equations of motion, shown in Fig. 1.

B. Planar monoped hopper

The second test system is a planar monoped hopper with a two-link leg, shown in Fig 6A. The three serial links in the system are similar to a 3-link pendulum with an added unilateral foot-ground contact. This similarity is highlighted so that any performance differences due to the addition of a contact model can be better isolated.

1) Experiment: The model performed a 5 meter *missing the boat* [10] sprint from rest, minimizing actuator effort according to

$$J = \sum_{i=1}^{N} \tau_{1i}^2 + \tau_{2i}^2 + \rho P, \tag{6}$$

where τ_{1i} and τ_{2i} are the input joint torques at the ith element of the hip and knee respectively. A total time of $T = 2$ seconds, discretized into $N = 100$ elements, was allocated to perform the maneuver. The initial and final poses were not specified beyond the requirement that it start at $x = 0m$ and finish with $x = 5m$.

2) Results: The solving times for each angle configuration are shown in Fig. 7. Again, the absolute formulation performs better, but the more interesting aspect is *how much* better: compared to the relatively minor difference in the case of similarly-sized pendulum models, the improvement is greater. This indicates that the contact constraints specifically are made more tractable by the change. As with the

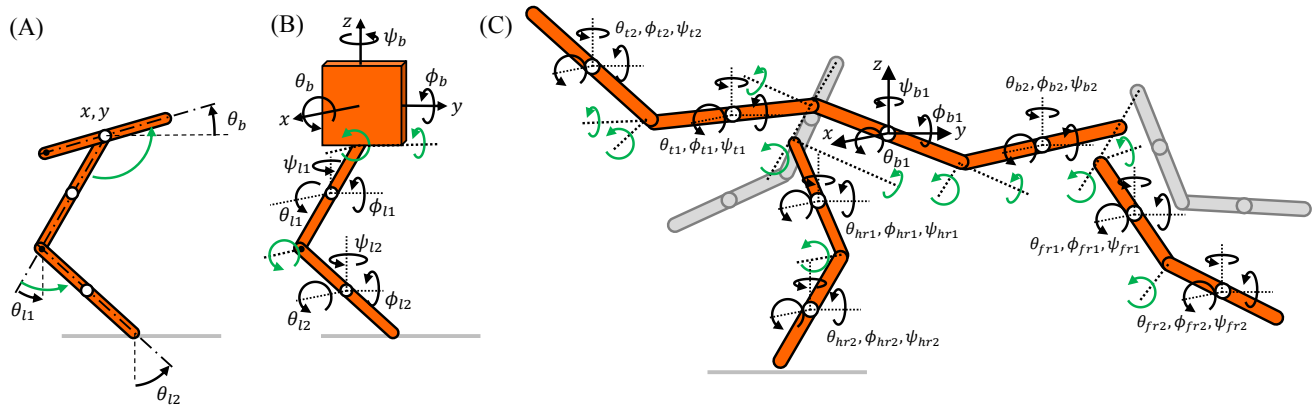

Fig. 6. Models used in each experiment: (A) planar monoped, (B) spatial monoped, (C) spatial quadruped, with its left side coloured gray to differentiate from its right. The parameters of the models are provided in the Appendix.

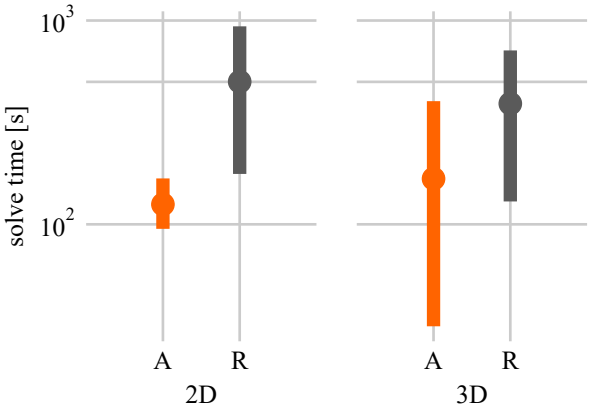

Fig. 7. Solving time for planar and spatial monoped models. The circle and bar represent the median and interquartile range, respectively.

pendulum, a smaller portion of the solving time was spent on NLP function evaluations for the absolute-angle model. It used an average of 20%, while the relative-angle model used 46%.

C. 3D monoped hopper

Our third test is of a 3D monoped hopper, shown in Figure 6B. It is similar to the planar model with an additional input torque for abduction. Put another way, a Hooke's joint was used for the hip while a rotary joint was used for the knee. The floating base body of the hopper is exactly the same for both formulations: $\mathbf{q}_b = [x\ y\ z\ \phi\ \theta\ \psi]^T$ where ϕ, θ and ψ are the roll, pitch and yaw in the 3D rotation Euler-321 [26] and (x, y, z) is the absolute position in space.

The approaches differ significantly for the leg, however. The absolute angle model was constrained using the equations described in sections II-B.1 and II-B.2. This resulted in 6 angles in the leg's state vector, along with the definition of three additional constraint forces.

In contrast, the top link in the relative-coordinate model was expressed as two rotations from the base, while the bottom link was a single further rotation from the top link. This means that the position of the foot in this relatively simple model is separated from the inertial frame by six successive rotations - three for the body, two at the hip and one at the knee.

1) Experiment: The hopper was made to find a minimum-effort periodic gait with an average velocity of $5m/s$. Periodicity was enforced by constraints specifying that the initial and final position and velocity values were equal. The initial and final states were not specified otherwise. We allocated $T = 0.7$ seconds for the stride, discretized into $N = 30$ elements. The cost function was the same as for the planar monoped and the implicit Euler method was used in the pre-solve stage.

2) Results: The results in Fig. 7 show that the absolute angle formulation converges significantly faster, often finding minimum-effort gaits in under 3 minutes. As in previous tests, 65% of the solve time spent on NLP function evaluations for the absolute-angle model was significantly less than the 83% spent by the relative-angle one.

D. 3D quadruped

Besides the ability to find trajectories that minimize some cost, in many applications, simply the ability to find trajectories *at all* is what makes optimization such a useful tool. By solving the forward and reverse kinematics problems simultaneously, it allows feasible motions to be simulated when neither the actions nor the forces required to drive them are known beforehand. This is especially valuable when the motions-of-interest are dangerous or difficult to coax out of human or animal subjects.

One such example is dynamic quadrupedal turning. Several aspects make it an especially challenging trajectory optimization problem:

- It cannot be reduced to a single plane of motion, so a 3D model with many degrees of freedom is required.
- The optimal foot contact order is not known *a priori*.
- It potentially requires a time horizon equivalent to several gait cycles.

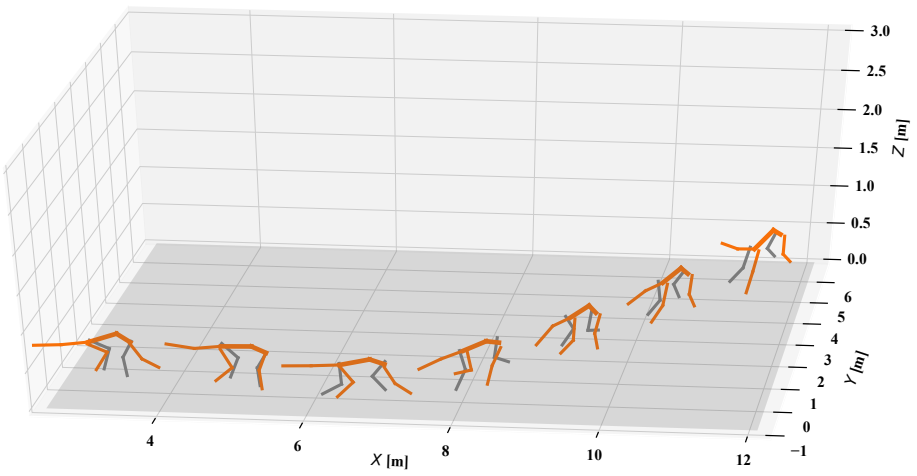

Fig. 8. Keyframes of a 3D quadruped performing a dynamic 60° turn to its left. The final stance on the right side at $t = T$ was constrained to be the same as the initial stance at $t = 0$, but rotated by 60 degrees about the vertical axis.

Previously, we attempted to generate this motion using a model with a relative angle configuration, but it could not successfully converge even when allowed 12 hours to solve. Here, we demonstrate that the changes described in this paper have made it possible to find a trajectory for a 3D quadruped performing a dynamic 60 degree turn with an unscheduled contact order.

Each leg of the quadruped contains the same degrees of freedom as the monoped leg. The body consists of four links connected by Hooke's joints (two for the spine, and two for the tail) modelled in the same way as the legs.

1) Experiment: Since the relative-angle model was not able to solve the turn in a reasonable time, we selected a simple dropping motion to provide a performance comparison and to show that both can converge. For this task, the initial state of the model is set to rest in a fixed position a short distance above the ground, while the final state is unspecified. The cost function was the same contact penalty as the 3D monoped hopper, for all four legs. The relative-angle quadruped converged in an average of 1240 seconds, while the absolute-angle quadruped took an average of 47 seconds. There was little variance in the solve time for both.

The quadruped turn proof-of-concept was solved in two stages: first, a straight-line $N = 50$ element, $8\ m/s$ periodic gallop was found using the same weighted contact penalty with minimum torque effort cost as was used for the 3D monoped.

Next, the turn: the position and velocity data of a point sampled from the periodic gait were used to fix the initial and final points, with the final position corresponding to a 60 degree yaw rotation of the initial orientation. The final x, y position was unspecified as the amount of space required to perform the turn was not known. An approximate version of the turn, created by concatenating 3 constant-speed gallops (for a total of $N = 150$) and interpolating the rotation linearly over the course of the trajectory, was used as an initial seed for the second stage. Random noise was added and several such seeds were tried to reduce the chances of optimizing into a poor local minimum, and the same cost function was used.

2) Results: The periodic gait for the absolute-referenced quadruped reliably converged in under 30 minutes, while the rapid turn converges in one to three hours (depending on the initial seed). An image of keyframes from an optimal solution can be found in Fig. 8.

E. Summary

The results for all models and configurations are summarized in Table I. It shows that the absolute-angle formulation allows trajectories of equivalent quality to be generated in less time.

TABLE I

AVERAGE SOLVE TIME AND COST OF ALL EXPERIMENTS

	Time [s]		Cost	
	Absolute	Relative	Absolute	Relative
pend. 2	0.437	0.441	1549	1247
pend. 4	6.31	37.8	1788	1504
pend. 6	16.5	194	1288	1213
pend. 8	40.7	1066	1314	1223
mono. 2D	125	500	2.25	2.21
mono. 3D	120	401	3.23	4.74
quad. 3D	1240	47	0	0

V. DISCUSSION

A. Build time

Table II gives the build time for each of the models. This is the time required to generate the symbolic equations and construct the optimization problem as an object that can be passed to the solver.

TABLE II

AVERAGE BUILD TIME FOR ALL EXPERIMENTS

	Absolute [s]	Relative [s]
pend. 2	0.620	1.00
pend. 4	2.61	7.94
pend. 6	3.09	23.5
pend. 8	4.73	78.4
mono. 2D	3.93	1.00
mono. 3D	369	5913
quad. 3D	3360	23790

TABLE III

PARAMETERS FOR EACH LINK IN THE 3D QUADRUPED MODEL

	mass [kg]	length [m]	radius [m]
$fr1$, $fl1$	0.171	0.254	0.012
$fr2$, $fl2$	0.068	0.247	0.005
$hr1$, $hl1$	0.210	0.281	0.010
$hr2$, $hl2$	0.160	0.287	0.011
$t1$	0.400	0.380	0.005
$t2$	0.200	0.380	0.005
$b1$, $b2$	13.00	0.310	0.080

For the 3D models, simplifying the symbolic EOM using functions included in Sympy (the Python symbolic mathematics library) was found to improve performance significantly. This is an extremely time-consuming process, however, which accounts for the large increase in the stated build times for the 3D monoped over its 2D counterpart. Without simplification, the 3D monoped can be built in 140 seconds for the absolute-angle model or 155 seconds for the relative-angle model.

For the quadruped, the relative-angle model could not be fully simplified within 12 hours, despite simplifying the equations in parallel on a 16-core computer. The stated build time for the model corresponds to a version where all terms in the EOM were simplified except the Coriolis term, which would have pushed the build time far beyond 12 hours.

B. Maximal coordinates

If the idea that simpler equations leads to better solver performance is taken to its logical conclusion, the next step would be to try a maximal coordinate [17] formulation: representing each 3D body as an individual 6-DOF system (with 3-DOF for 2D bodies) referenced to the world frame, connected by constraint forces included in the state variable at each collocation point. This results in extremely simple equations of motion, though many more variables and constraints are required to implement the connections.

As an experiment, we repeated the planar pendulum test with this configuration but the results were not promising: the maximal model required much longer times to solve than either the relative or absolute angle coordinate options, and was far less robust, often failing to solve altogether. Further research is required to see whether the maximal formulation is beneficial in other cases, and whether there are modifications that can be made to improve its performance.

C. Limitations of the study

Only the IPOPT algorithm was considered in this study. It is indeed possible that the difference in performance between the two configurations might not be as notable for a different solver. We repeated the pendulum swing-up comparison for 4-link pendulums across a range of available linear solvers for IPOPT: MUMPS, MA57, MA77, MA86 and MA97 [25]. Although there were differences in the overall performance, the relative performance between the configurations did not significantly change.

VI. CONCLUSION

Trajectory optimization is a useful tool, but its major drawback is its potentially slow convergence times for large multi-body systems. Methods to decrease the time until convergence often sacrifice accuracy, which may invalidate the generated trajectories if they differ too much from reality. Here, we have shown how a minor modelling change - writing the orientation of every rigid body as a rotation from the inertial frame, instead of as a relative rotation - more efficiently deals with the complex Coriolis terms that arise from long serial changes. This formulation opens up the possibility of applying trajectory optimization to new, large, high-speed models in both two and three dimensions.

VII. APPENDIX

A. Pendulum parameters

Each link of the pendulum is modeled as a infinitesimally thin rod with unit mass and unit length.

B. Monoped parameters

The 2D monoped has a body segment of mass $m_b = 5$ kg and length $l_b = 1$ m. Both leg segments have mass $m_l = 1$ kg and length $l_l = 0.5$ m. All segments are assumed to be thin rods.

The parameters of the 3D monoped are the same, but the body is modelled as a cube with a side length $s_b = 0.4$ m, and the leg mass is reduced to $m_l = 0.5$ kg.

C. Quadruped parameters

The parameters of the 3D quadruped share the subscripts associated with the coordinates shown in Fig. 6. All links are modeled as cylinders.

The values of the parameters are provided in Table III.

REFERENCES

[1] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur, "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 684–689, IEEE, 2015.

[2] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, 2009.

[3] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on mechatronics*, vol. 15, no. 5, pp. 783–792, 2009.

[4] W. Xi and C. D. Remy, "Optimal gaits and motions for legged robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3259–3265, IEEE, 2014.

[5] M. Srinivasan, "Fifteen observations on the structure of energy-minimizing gaits in many simple biped models," *Journal of The Royal Society Interface*, vol. 8, no. 54, pp. 74–98, 2010.

[6] C. Gehring, S. Coros, M. Hutler, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, *et al.*, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[7] A. Blom and A. Patel, "Investigation of a bipedal platform for rapid acceleration and braking manoeuvres," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 426–432, IEEE, 2018.

[8] C. Fisher, S. Shield, and A. Patel, "The effect of spine morphology on rapid acceleration in quadruped robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2121–2127, IEEE, 2017.

[9] C. Paul and J. C. Bongard, "The road less travelled: Morphology in the optimization of biped robot locomotion," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1, pp. 226–232, IEEE, 2001.

[10] C. Hubicki, M. Jones, M. Daley, and J. Hurst, "Do limit cycles matter in the long run? stable orbits and sliding-mass dynamics emerge in task-optimal locomotion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113–5120, IEEE, 2015.

[11] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, 2019.

[12] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[13] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, IEEE, 2014.

[14] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid bipedal gait design using c-frost with illustration on a cassie-series robot," *arXiv preprint arXiv:1807.06614*, 2018.

[15] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[16] D. T. Greenwood, *Advanced dynamics*. Cambridge University Press, 2006.

[17] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *SIGGRAPH*, vol. 96, pp. 137–146, Citeseer, 1996.

[18] C. Roithmayr, *Relating constrained motion to force through Newton's second law*. PhD thesis, Georgia Institute of Technology, 2007.

[19] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[20] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, vol. 10. Siam, 2010.

[21] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *International Symposium on Robotics Research (ISRR), Puerto Varas, Chile*, 2017.

[22] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo–optimization modeling in python*, vol. 67. Springer Science & Business Media, second ed., 2017.

[23] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[25] HSL, "A collection of fortran codes for large-scale scientific computation," *See http://www.hsl.rl.ac.uk*, 2007.

[26] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.