A Relaxation Method for Simulating the Kinematics of Compound Nonlinear Mechanisms

Hod Lipson
Sibley School of Mechanical and Aerospace Engineering and Faculty of Computing and Information,
Cornell University,
Ithaca, NY 14853
e-mail: hod.lipson@cornell.edu

This paper describes a relaxation-based method for simulating 2D and 3D compound kinematic mechanisms. The relaxational process iteratively propagates node motions and degrees of freedom throughout a given kinematic mechanism. While relaxation methods were classically used to solve static problems, we show that the propagation of displacements during the calculation process itself reveals the kinematics of the structure. The method is slower than approaches based on solving simultaneous differential equations of motion, but provides several advantages: It achieves a higher level of accuracy, is more robust in handling transient singularities and degeneracies of the mechanism, and can handle more complex compound mechanisms with many links in multiple entangled kinematic chains. It also allows straightforward introduction of linkages with nonlinear behaviors such as wrapping strings, hydraulics, actuators, contacts, and other arbitrary responses. The basic simulation algorithm is presented, and a number of applications are provided including robotics, design, and biomechanics. [DOI: 10.1115/1.2198255]

Keywords: kinematics simulation, relaxation, robotics, biomechanics

Introduction

Kinematics—*the science of pure motion*—is concerned with the analysis and synthesis of mechanisms composed of connected elements. It deals with the relative geometric displacements of points and links of a mechanism, without regard to forces that generate those displacements or the physical embodiment that realizes them.

The science of kinematics has its origins in the need to systematically analyze and design machines at the beginnings of the industrial age [1,2], but in recent years the interest in kinematics has increased in scope beyond mechanical applications: Many applications today involve nonengineering usage, such as physically realistic computer gaming [3], biomechanics [4,5] and interactive education. Kinematics is also used as a component within larger design automation and control systems [6].

The invigorated interest in kinematics has led to new requirements that simulation codes must meet. Besides simulation fidelity, some application areas require stability in handling singularities, real-time computation, scalability to very large systems with hundreds of components, and simplicity of use. Accuracy may be traded for speed in applications where real-time or interactive-time performance is critical. Robustness may be more important in applications where users are not technically savvy, and automatic graceful handling of errors and transient singularities is expected. The expansion of application areas has led users to expect to create increasingly complex models, incorporating hundreds of kinematic elements, and a larger variety of element types and behaviors. The complexity of the code itself is also becoming an important factor: Simpler implementations are preferred when complex legacy codes are rewritten in favor of more maintainable and web-operable systems.

This paper describes a relaxation-based method for simulating the *kinematics* of mechanisms that provides increased robustness and scalability at the expense of not accounting for *dynamics*.

Contributed by the Mechanisms and Robotics Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received December 10, 2004; final manuscript received October 14, 2005. Review conducted by Gordon R. Pennock.

Kinematic Versus Dynamic Simulation

The most common way for simulating multibody *kinematics* is through *dynamic* simulation [7–9]. Dynamic behavior is typically calculated using numerical integration of a set of simultaneous differential equations that describe the forces and torques that a mechanism is subject to, under constraints at joints and contacts. Newton-Euler formalisms are based on Newtonian mechanics, and model constraints through additional forces that enforce them. Euler-Lagrange equations use variational calculus methods that enforce constraints by eliminating degrees of freedom from the equations. Other approaches solve the equations symbolically [10] or use case-based analysis [11]. Some methods use a symbolic reasoning approach that attempts to emulate the qualitative reasoning a human performs when predicting the function of a mechanism [12].

Many of the classical simulation methods do not scale well to complex mechanisms with entangled kinematic chains. Current commercial simulators often become unstable when handling systems composed of a few tens of coupled links, such as three or more chained Stewart platforms. These instabilities arise from the difficulties of solving multiple nonlinear differential equations simultaneously. But even if these equations were solved correctly, complex dynamical behavior of mechanisms is often too chaotic and too dependent on many elusive parameters such as friction and impact, to be quantitatively predictable at all. For example, no dynamic simulator is likely to faithfully predict the final resting position of rolling dice or the motion of a double pendulum over any prolonged period of time.

There are, however, simpler ways to determine the *kinematics* of a mechanism, without resorting to dynamics. Unlike *dynamics*, kinematic calculations are only concerned with the motion and displacement that a mechanism links can undergo. Forces, accelerations, mass properties, and damping rates can be ignored, and so many of the calculation complexities and sources of instability can be removed.

This paper describes a relaxation-based approach for simulating the geometric kinematics of mechanisms. The relaxational process iteratively propagates node motions and degrees of freedom

 Copyright © 2006 by ASME

throughout a given actuated mechanism. While relaxation methods were classically used to solve static problems, the key claim in this paper is that the *propagation of displacements during the calculation process itself* reveals the kinematics of the structure.

Of particular interest is the ability to handle kinematic links capable of excreting nonlinearly varying geometric constraints. A kinematic component enforces a certain geometric constraint between joints in space. A simple pin jointed bar, for example, holds the distance between two joints constant. The term nonlinear kinematics refers here to a situation where the geometric constraint enforced is not constant (or linear), and itself depends on the geometry of the joints it is controlling. For example, a string will constrain the distance between two joints so long as their motion is not toward each other. If their motion is toward each other, the constraint disappears. More complex constraints can be described, resulting in what we term as *nonlinear kinematics*.

The relaxation method proposed here provides a different tradeoff between the need for accuracy, speed, scalability, and robustness. It has several advantages and disadvantages compared to other classical methods. In particular, the method is slower than approaches based on solving simultaneous differential equations of motion, and completely ignores dynamic effects which are critical in many applications. However, it provides several advantages: It achieves a higher level of kinematic accuracy, is more robust in handling transient singularities and degeneracies of the mechanism, and can handle more complex compound mechanisms with many links in multiple entangled kinematic chains. It also allows straightforward introduction of linkages with nonlinear behaviors such as wrapping strings, hydraulics, actuators, contacts, and other arbitrary responses.

The following section provides an outline of the basic relaxation method and its implicit extension to simulating kinematics. The next section describes a number of extensions that allow simulation of a larger variety of element types. The last section demonstrates a number of applications, including robotics, gaming, and biomechanics.

Simulating Kinematics Using Relaxation

Historical Background. Relaxation methods were one of the earliest computational paradigms for solving relatively large scale engineering problems, even before the digital computer. In his 1940 textbook *Relaxation Methods in Engineering Science*, Southwell describes the key concept of these methods: "Their basis…is the notion that in any specific problem, although it may be difficult to obtain solutions by direct attack, it is easy to reverse this process… We can always determine the error of a trial solution…[then] we apply a series of operations, each one an indirect solution of a particularly simple kind, and in this way we "tune up" a trial solution (whether good or bad) until it conforms with some imposed standard of accuracy" (R. V. Southwell [13]).

Since then, the process of iterative solution by optimization has been expanded from its original pin-joined frame application to almost all fields of engineering and science involving the approximate solution of large scale systems, including the solution by error minimization of general sets of nonlinear equations. One of the advantages of the relaxation approach was that it is intuitive: "What is of greater importance, even though a computer[1] may not comprehend the theoretical basis of his calculations he will have, throughout, a mental picture of what he is doing; he will see his task as that of bringing unaccounted or 'residual' quantities within a specified margin of uncertainty" (R. V. Southwell [13]).

The slow iterative convergence of the relaxation process ultimately led to its replacement by faster direct solution methods like finite elements, but it is these original properties that now merit reconsideration in light of the new requirements. The intuitive nature of the process had the advantage of allowing the persons doing the calculation to spot and correct mistakes early, to interject judgment to accelerate convergence, and to locally handle special cases such as transient singularities and nonlinearities. This is in contrast with noniterative methods that solve a system of equations simultaneously by inverting a large matrix. Errors are not known until the entire calculation is complete, and any local singularity will render the entire matrix singular. The local nature of the calculations also made it possible to distribute the computation among several persons, a factor that again is becoming of interest in today's distributed computation environments and multicore processors.

The Basic Relaxation Process for Static Structures. The basic relaxation process, as applied to pin-joined structures, comprises two steps applied iteratively. The first step computes the total net force $\mathbf{R}_i$ on each joint; this is the *residual* force, since these forces should be zero if the structure is in perfect equilibrium. The second step adjusts the position vector $\mathbf{x}_i$ of each joint so as to reduce the residual force acting upon it. These two steps are repeated until the maximum residual force goes below a desired threshold.

The residual force vector $\boldsymbol{R}_i$ acting on joint i is computed by summing all the forces produced by links connected to it

$$\mathbf{R}_i = \sum_{j \in \{L_i\}} k_j (l_j - l_{0j}) \hat{\mathbf{n}}_j + \mathbf{F}_i \tag{1}$$

where $\{L_i\}$ is the set of all links meeting at joint i, l_j is the current length of link j, $l_j = \|\mathbf{x}_{j2} - \mathbf{x}_i\|$ and j_2 is the index of the other joint to which link j is connected, l_{0j} is the given nominal length of link j, $\mathbf{x}_i$ is a vector of the coordinate of joint i, k_j is the stiffness of link j, (e.g., for a beam $k_j = E_j A_j / l_j$ where E is the modulus and A is the cross section area), $\hat{\mathbf{n}}_j$ is the unit vector in the direction of the link $\hat{\mathbf{n}}_j = (\mathbf{x}_{j2} - \mathbf{x}_i) / \|\mathbf{x}_{j2} - \mathbf{x}_i\|$, and $\mathbf{F}_i$ is any external force vector acting on joint i (e.g. gravity).

The effective stiffness vector $\mathbf{S}_i$ of each joint i is estimated by summing all the stiffness of all links connected to it

$$\mathbf{S}_i = \max\left[\sum_{j \in L_i} k_j abs(\hat{\mathbf{n}}_j), \mathbf{S}_{\min} \right] \tag{2}$$

where $abs(\mathbf{n}_j)$ is a unit vector composed of the absolute value of each term in the unit vector in the direction of the link, $abs(\hat{\mathbf{n}}_j) = \langle |\hat{n}_x|, |\hat{n}_y|, |\hat{n}_z| \rangle$. Note that to handle singularities, $\mathbf{S}_i$ must be ensured to be nonzero by not allowing it to go below $\mathbf{S}_{\min}$ (see the next section for a discussion of singularities). Given the residual force acting on each joint and its effective stiffness, the displacement vector Δ_i of joint i is computed as the ratio

$$\boldsymbol{\Delta}_i = \alpha \left\langle \frac{R_x}{S_x}, \frac{R_y}{S_y}, \frac{R_z}{S_z}, \right\rangle \tag{3}$$

where α is a relaxation factor typically in the range $0 < \alpha < 1$ but occasionally larger (over-relaxation). The displacement can be capped by some value $\Delta_{\max}$. The joint positions (of ungrounded nodes only) are then updated by

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \boldsymbol{\Delta}_i \tag{4}$$

Equations (1)–(4) are repeated until the overall largest residual force $\max(\mathbf{R}_i)$ goes below a desired threshold ϵ. Grounded nodes are omitted from this maximum.

The steps above are aggregated in the pseudocode shown in Fig. 1(*a*). For greater computational efficiency, many of the above steps can be calculated concurrently using the pseudocode shown in Fig. 1(*b*), resulting in faster performance but in some cases loosing guarantees on convergence. In practice, the relaxation factor α is often gradually reduced from a large number initially (e.g., 0.5) to a small number (e.g., 0.01) as the residuals diminish. This *relaxation schedule* may need to be adjusted dynamically for some structures.

[1] A *computer*, in those days, was a person hired to perform arithmetic calculations.

```
Repeat
     For all ungrounded joints j
        R←F_j
        S←0
        For all links i∈{L_j}
           j2←indx of other joint of link i
           n←(x_J2−x_j)/|x_J2−x_j|
           f←k_i(l_i−l_0i)
           R←R+f·n
           S←S+k|n|
        S←max(S,S_min)
        Δ←α·R/S
        x_j←x_j+Δ
Until max(R)≤ε for ungrounded joints
```

(a)

```
Repeat
     R_i←F_i for all joints i.
     S_i←0 for all joints i.
     For all links i
          j1←index of 1st joint of link i
          j2←index of 2nd joint of link i
          n←(x_J2i− x_J1i)/|x_J2i− x_J1i|
          f←k_j(l_i−l_0i)
          R_j1←R_j1+f·n
          R_j2←R_j2−f·n
          S_j1←S_j1+k|n|
          S_j2←S_j2+k|n|
     For all ungrounded joints i
          S_i←max(S_i,S_min)
          Δ←α·R_i/S_i
          x_i←x_i+Δ
Until max(R_i)≤ε for ungrounded joints
```

(b)

Fig. 1 Pseudocode for the simple kinematic relaxation algorithm for a static pin-joined frame; (a) asynchronous, (b) synchronous

Advantages Over Solutions Using Simultaneous Equations

It would clearly be faster to solve these stiffness and displacement relationships simultaneously rather than iteratively, but an iterative solution holds several advantages.

Handling Transient Singularities and Degeneracies. Consider a structure that is inherently singular, such as the frame shown in Fig. 2(*a*). At its initial state, the structure cannot support a vertical load **F** at joint *a*, because joint *a* has zero stiffness in the vertical direction. Solving this structure using simultaneous methods would yield a singular global stiffness matrix. However, using relaxation, we identify this situation by checking for small values of S_i terms, and ensuring they do not go below a minimal value

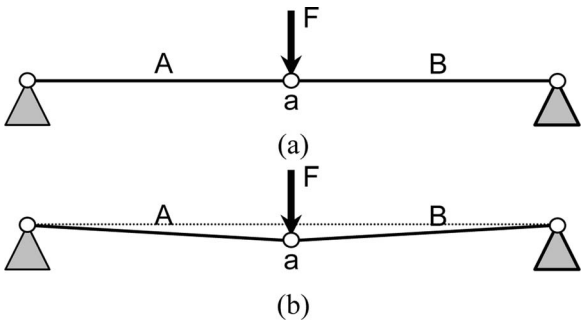


Fig. 2 Handling of a singular frame

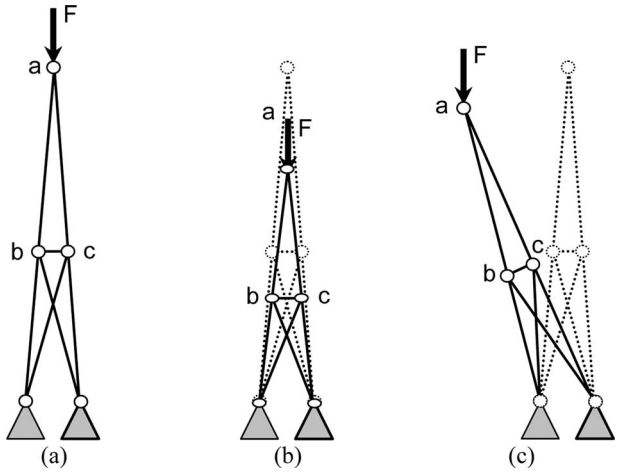


Fig. 3 Handling of unstable equilibrium

$S_{\min}$. This causes joint *a* in the example above to move vertically by an arbitrary small amount in the correct direction. This small amount will ensure that at the next iteration, the joint will have some valid vertical stiffness and consequently move toward its final equilibrium position as shown in Fig. 2(*b*). The equilibrium position is handled correctly only since the direction cosines are updated at every iteration. Such case-specific approaches may be introduced for handling other forms of singularities. Note that while singular structures may be rare in static analysis, they occur often during kinematic analysis as a frame transitions through a series of configurations.

We are not aware of a systematic way to determine the value $S_{\min}$, and currently tune it manually per application. A starting guess is to set it to a fraction of $\mathbf{F}_{\max}/\Delta_{\max}$ where these values represent the maximum anticipated force and the maximum allowable displacement per iteration, respectively.

Handling Unstable Equilibrium. A second degenerate case is unstable equilibrium. Consider the slender symmetrical structure shown in Fig. 3(*a*), subjected to a compression load **F** at joint *a*. Simultaneous solution of the equations would yield the deformed, symmetrical structure shown in Fig. 3(*b*). However, that solution may be unstable. Under some conditions, a more realistic outcome could be the structure shown in Fig. 3(*c*) (or its symmetrical counterpart with the structure leaning to the right, not shown).

This possibly misleading unstable equilibrium of the solution shown in Fig. 3(*b*) is avoided by the arbitrary order of node updates in the asynchronous version of the algorithm. Once force **F** is applied to node *a*, the stiffness $\mathbf{S}_a$ of node *a* is calculated and node *a* is moved vertically down a little. Next, either node *b* or *c* are computed. If node *b* is computed first, it is at that stage only subject to force from node *a*, and therefore it will move leftward. This motion will have broken the symmetry of the structure, causing subsequent calculation of node *c* to include a leftward force too. This asymmetry will amplify until an asymmetric equilibrium state of Fig. 3(*c*) is reached.

This symmetry breaking can also be introduced in the synchronous version of the algorithm (where nodes are updated simultaneously) by temporarily adding very small noise to forces, stiffnesses, or displacement vectors. The correctness of the final solution is unaffected as these perturbations are only added temporarily, but physically unstable equlibria are escaped. Symmetry breaking noise may be more than a mathematical tool; it may be an important factor governing the behavior of realistic mechanisms [14].

Incorporating Nonlinear and Variable Behavior. Since element behaviors are calculated anew at each iteration, we may vary their properties in arbitrary ways. For example, the link stiffness k_i

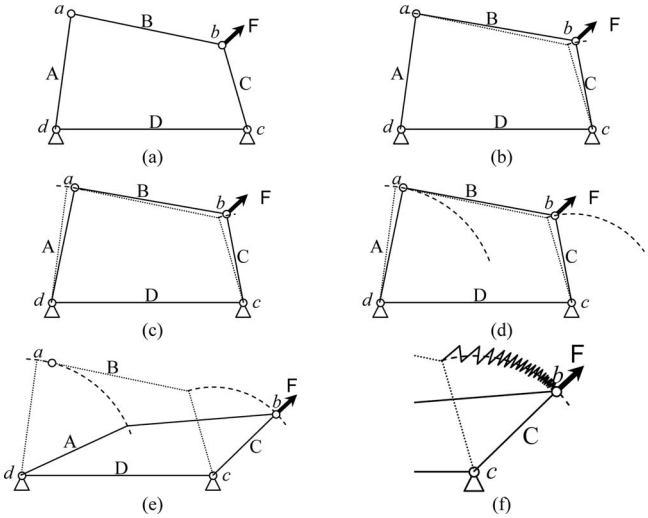


Fig. 4 Simulation of a four-bar mechanism

need not be constant, and we can replace it with a function k_i (l) that depends on the current length. A function for describing a string (that can undergo tension with stiffness K but sustain no load in compression) would be (but see also Eq. (7))

$$k_i(l) = \begin{cases} K & \text{if} \quad l \geq l_0 \\ 0 & \text{if} \quad l < l_0 \end{cases} \tag{5}$$

Similarly, a property may change over time in kinematic analysis. For example, we can describe an actuator as a link which changes its l_0 over time.

The Basic Relaxation Process for Kinematic Structures. While the original goal of the relaxation process was to arrive at the equilibrium state and thus solve for static conditions, application of the same process to a statically underconstrained structure reveals its kinematics.

It is important to note that the algorithm presented here only addresses the geometrical kinematics, without regard to forces, accelerations, time, or material effects such as buckling or bending. Terms like "elongation," "stiffness," and "force" are used only metaphorically: The elongation of the links are not physical elongations, but are mathematical "errors" that represent the geometrical discrepancy between the nominal link length and the current distance between its endpoints. This error is transient, because if the mechanism is actuatable (does not lock), then these elongations are propagated until they vanish arbitrarily close to zero. Similarly, the stiffness coefficient governs the rate by which this mathematical error propagates, and a force is a means for causing geometric perturbation. One can always choose an arbitrarily high stiffness or arbitrarily small force so that buckling and bending effects are smaller than any desired limit. The choice of stiffness and force merely affect the rate of convergence versus the convergence error—they act more like a relaxation factor than a physical stiffness or force.

Consider the grounded four-bar mechanism *A-B-C-D* shown in Fig. 4(*a*), with the force **F** applied to joint *b*. The mechanism is statically underdetermined and therefore could not undergo conventional static analysis. However, applying relaxation causes it to move through its kinematic trajectory. In the first iteration, only node *b* has a nonzero residual in the direction of **F**, and it will consequently move in the direction of **F**, to a new location as shown in Fig. 4(*b*). In the second iteration, node *a* will have a nonzero residual due to the elongation of link *B*, and will therefore move toward joint *b* as shown Fig. 4(*c*). Subsequently, the residual of joint *b* will be updated with the force exerted by links *B* and *C*. Link *B* will not apply much force because of the motion of joint *a*, but the force exerted by link *C* will cause it to move toward the theoretical dotted arc in Fig. 4(*d*). In subsequent iterations, joints *a* and *b* will progress further until they reach a final equilibrium point as shown in Fig. 4(*e*).

The trajectory taken by the joints as their positions are updated approximates the theoretical path of the mechanism. A typical trajectory is shown in Fig. 4(*f*), balancing the effects of the external force **F** and the restoring force exerted by link *C*. The accuracy of the simulation, determined by the magnitude of the deviation from the theoretical path, is proportional to the relaxation factor α and inversely proportionate to the stiffness of the links. The rate of simulation is inversely proportional to the accuracy.

Qualitative Dynamics. It is interesting to note that although the simulation process is geometrical and does not account for dynamics in any way, the computational propagation of motions throughout the structure *qualitatively* emulates first-order (viscous) dynamics. In this sense, the relaxation factor α is a proxy for damping (small α corresponds to large damping coefficient), and can be adjusted for different components independently to observe qualitative damping effects.

Mass inertia effects can be *qualitatively* emulated by replacing the displacement update rule of Eq. (3) with

$$\boldsymbol{\Delta}_i^{(t+1)} = \eta\alpha\left\langle \frac{R_x}{S_x}, \frac{R_y}{S_y}, \frac{R_z}{S_z}, \right\rangle + (1-\eta)\boldsymbol{\Delta}_i^{(t)} \tag{6}$$

Setting $\eta = 1$ will result in the original fully damped (no oscillations) behavior as described earlier. Setting η to other values in the range $[0,1]$ causes pseudoinertial effects as nodes tend to persist in their previous displacement rates. The value of η can be different for each node to emulate non-uniform mass distribution.

Simulating Various Joint and Link Types. One way to expand the relaxation process for other structures and joint types would be to derive new update rules. An alternative approach is to emulate these structures and joints using equivalent assemblies of pin-joined frames.

Table 1 lists pin-joined equivalents to some common higher-level kinematic elements. Functional equivalence is achieved by adding auxiliary structure; this structure is often made hidden and omitted from contact and collision calculations. More complex mechanical components and behaviors can be represented through specially tailored update rules that are executed at each iteration of the relaxation process. Some examples are shown in Table 2.

Limitations. Implementations of relaxation kinematics may run into several problems such as slow or lack of convergence and incorrect solutions. Some of these situations are outlined below.

Convergence limitations. The relaxation algorithm generally converges as long as the element behavior is positive and monotonic (i.e., larger elongation implies equal or larger resisting force) and the relaxation factor is small enough. When each joint is handled in turn, then the total energy of the mechanism due to strain of the elements and external forces is reduced at each iteration, and thus the process will converge; however this is not guaranteed if nodes are updated simultaneously in the synchronous implementation. Since this reasoning assumes local linear behavior, then highly nonlinear behaviors will require small relaxation factors around nonlinearities. Abrupt discontinuities are therefore a source of problems, and "soft" implementations of contact and collision will also help. For example, a mechanism that contains a string element implemented using Eq. (5) may create oscillations if the point of equilibrium is near the point of switching. To overcome this difficulty, a more robust implementation of a string element would be

Table 1 Some higher-level elements and their pin-joined equivalents

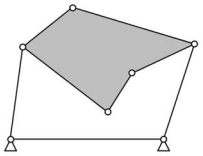
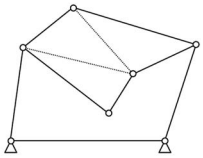
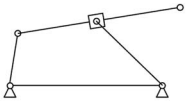
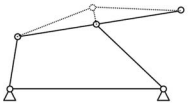
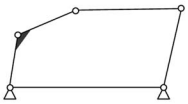
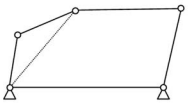
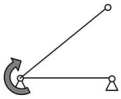
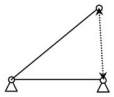
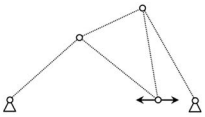
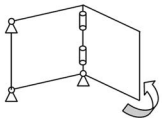
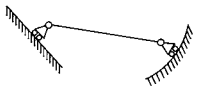
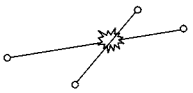
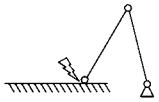
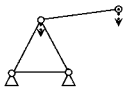
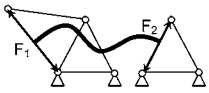
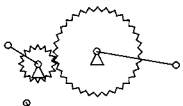
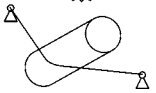
High-level element type	Target structure	Pin-joined equivalent (Added links shown dotted)
Rigid components can be represented as a triangular mesh		
Intermediate connections can be represented by adding auxiliary support structure		
Welded joint can be represented by adding a link across the joint.		
Rotary actuation can be represented by adding link across the angle, whose l_0 (rest length) can be adjust according to sine law.		
Prismatic joints can be represented by any straight line mechanism, such as Robert's linkage (shown here), or by enforcing constraints on displacements (see Table 2).		
3D Hinges can be represented by auxiliary structure.		

Table 2 Some higher-level effects and their update rules

High-level behavior	
Constrained motion can be enforced by breaking down the displacement vector into allowed and disallowed components, and removing the disallowed component. For example, forcing planar motion involves removing the part normal to the constraint plane.	
Collisions can be detected by calculating the distance between two segments and if below a threshold, applying repelling forces at the four endpoints. The magnitude of the force depends on the collision 'softness', and it is distributed among four endpoints in inverse proportion to their distance from the collision point.	
Static Friction can be approximated by breaking down the residual force into parallel F_p and normal F_n components, and zeroing the parallel component if $F_p \leq \mu F_n$ where μ is the Newtonian friction coefficient.	
Gravity and other field effects can be emulated by applying calculated external forces at every joint.	
Hydraulics can be emulated by updating rest lengths and stiffnesses (l_0, k) of links arbitrarily or in some proportion to lengths of other links, but making k zero in response to tension.	
Gears and other mechanisms can be emulated by updating the angle of one link in some fixed proportion to the angle of another link (see Table 2 for enforcing angles)	
Wrapping effects can be calculated by braking down a link into smaller segments and performing collision/contact calculation at each intermediate node.	

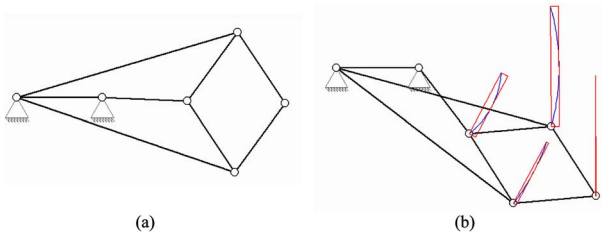


Fig. 5 Test case 1: The Peaucellier mechanism (8 links, 1 DoF) was simulated to accuracy 10^{-5}

$$k_i(l) = \begin{cases} K & l > l_0 + a \\ K \cdot \dfrac{1 - l_0 + a}{2a} & l_0 - a \leq l \leq l_0 + a \\ 0 & l < l_0 - a \end{cases} \tag{7}$$

where a is a radius around l_0 where the sharp step function of Eq. (5) is replaced with a smooth transition in a radius of a around l_0. Other methods to reduce oscillations are to actively reduce the relaxation factor for fluctuating nodes, or to average their stiffness over time rather than update them every iteration.

Overconstrained mechanisms. Simulation of overconstrained mechanisms will yield solutions that will account for internal stresses in the mechanism. In this case the relaxation process will converge correctly (residual forces will vanish) but the internal loads (i.e., f in Fig. 1) will remain high, whereas in a true mechanism operating within its degrees of freedom, all internal loads should vanish. This situation is easily be detected by measuring internal loads after the relaxation process has converged.

Incorrect solutions. Many multi-DoF mechanisms have several mathematically possible solutions, and the relaxation solver may sometimes reach an impossible or implausible one. This typically occurs when the relaxation factor is too large or when changes are made abruptly, thereby skipping the physical propagation of forces. For example, if actuators are used, they should be changed in very small increments resembling their physical realization.

Performance

In addition to various successful applications, we evaluated the kinematic simulator on a number of test problems of varying degrees of complexity.

Peaucellier Mechanism. The first test was conducted on the Peaucellier mechanism—a classic single degree of freedom mechanism from 1876, designed to trace an exact straight line [1]. The mechanism is composed of eight links arranged in a two-dimensional (2D) configuration as shown in Fig. 5(a). The simulation error was computed by putting a tight bounding box around all points in the traces produced by the joints of the mechanism as is moves (Fig. 5(b)). The aspect ratio of the bounding box provides an approximation of the error, as the theortical mechanism should trace an exact straight line and the aspect ration should approach zero. The relaxation kinematics solver traced a curve with an aspect ratio of 10^{-5}.

Chained Stewart Platforms. The second test case explored the performance of the relaxation simulator on a series of mechanisms with a range of complexities. The mechanisms were composed of several Stewart platforms chained back-to-back in a series. Each Stewart platform is a parallel mechanism with six degrees of freedom, composed of six linear actuators connecting two triangular plates [15]. The actuators are shown in red, and the plates in blue, in Fig. 6. There were two reasons for choosing this test case: First, we had encountered many difficulties simulating even two connected Stewart platforms using publicly available simulator code [16] earlier, and so this appears to be a nontrivial problem. Second, by chaining a series of platforms we can create an arbitrarily complex mechanism with certain known solutions, using which we can obtain absolute error measures.

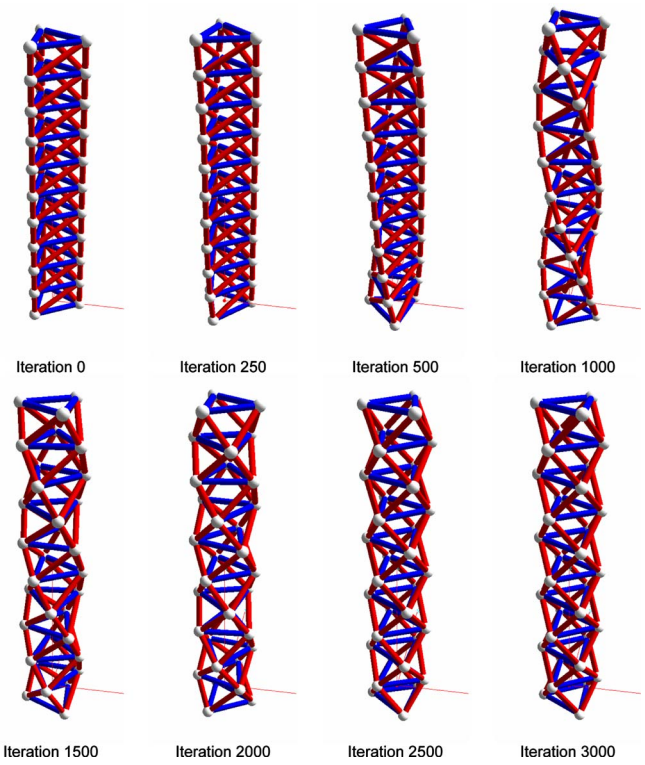


Fig. 6 Test case II: Snapshots from a kinematic simulation of 10 chained Stewart platforms. Shown 60 actuators (red) and 11 rigid platforms (blue)

A mechanism comprising ten Stewart platforms was initialized as shown in Fig. 6 (iteration 0), with all plates aligned, diagonal actuators set at length 20 units and vertical actuators set to length of ten units. The resulting structure stands 100 units tall. No gravitational loads or collision detection were used in this simulation. We then actuated all 60 actuators and changed their lengths to 15 units. Figure 6 shows a sequence of snapshots of the structure as it twists into its target configuration spanning about 1 s on a Pentium IV 2.4 GHz CPU. We measured the absolute error by calculating the deviation from colinearity of three points along the side of the structure that are known to be collinear in the theoretical target solution. We also estimated the error by recording the magnitude of the maximal residual (unaccounted) force in the mechanism.

Figure 7(a) tracks the error as function of iterations for solving the ten-platform case using the synchronous implementation (Fig. 1(b)). The actuators were actuated by changing their relaxed length in increments of 0.002 units every iteration for the first 2500 iterations. During that period, the residual forces fluctuate but remain on the order of 100 N. As the actuation terminates, both the absolute error and the residual error drop asymptotically, as seen by the linear curves in the log plot.

Figure 7(b) examines the trend of absolute error for mechanisms composed of 6 to 20 Stewart platforms (labeled S6 to S20, respectively). Mechanisms S6-S14 were actuated for the first 2500 iterations. Mechanisms S16-S18 were actuated for the first 5000 iterations, and mechanism S20 was actuated during the first 10,000 iterations. One can again see fluctuations in the error during actuation, but these subside exponentially after actuation has stopped. Slow actuation is necessary to force the mechanism to take a mechanically realistic trajectory; if actuation is performed

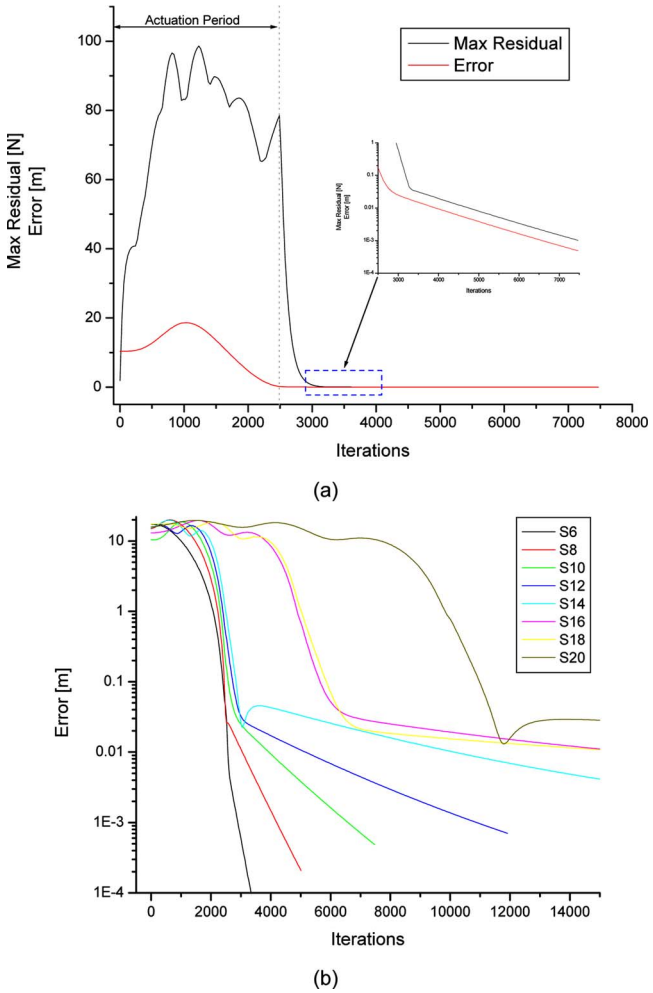


Fig. 7 Errors during simulation of various linked Stewart platforms. (*a*) Maximum residual and absolute error versus iterations while simulating ten-linked platforms, (*b*) absolute error trends for a number of mechanism ranging from six Stewart platforms (S6) to 20 linked platforms (S20).

any faster, the mechanism may reach a stable, mathematically correct solution which is mechanically implausible, such as a topologically impossible collapsed state.

Random Mechanisms. The third test case explored the performance of the relaxation simulator on a randomly generated mechanism. The mechanism was generated by scattering 100 points in space and connecting them with links arbitrarily. A grounded structure with n joints and l links has at least $3n$-l-6 degrees of freedom due to topological considerations. To create a highly underconstrained mechanism, we linked the 100 joints with only 200 links, and replaced 20 links with actuators that were actuated to a random length. No absolute error could be calculated, but we verified that the residual error converged asymptotically to zero. Figure 8(*a*) shows the initial structure and Fig. 8(*b*) shows the final mechanism after 10,000 iterations. Figures 8(*c*)–8(*f*) plot the maximum residual (unaccounted) force in the mechanism for ten independent runs on two types of mechanisms: One with 100 DoF and one with 50 DoF. In each run, a random mechanism was generated from scratch, and tested using two implementations: The synchronous and the asynchronous relaxation. In all cases the actuation period continued until iteration 30,000.

It is evident that convergence is slower for more complex mechanisms (more links and fewer degrees of freedom), but difference between the synchronous and asynchronous implementations are minor. The asynchronous implementation tends to have less transient error but each iteration of the asynchronous implementation takes longer to complete. The actual increase in time depends on what computation, such as collision detection, is included in the inner loop.

These three test cases were achieved with relatively little tuning of the algorithm coefficients $S_{\min}$ and α, and the maximum actuation rate. These tests do not provide an exhaustive evaluation but they demonstrate robust performance in a number of cases where conventional simulation approaches are known to experience difficulty.

Applications

We have used the relaxation process a number of applications where commercial or publicly available simulation software would not meet a variety of requirements. Some of these cases, their special requirements and the relaxation implementation are described below.

Evolutionary Robotics. Evolutionary robotics methods use algorithms inspired by natural evolution to automatically design robotic systems. These methods usually operate in simulation, where populations of hundreds of robots are evaluated over hundreds of generations and undergo natural selection for a desired task. During evaluation, robotic systems including both morphology and control need to be simulated. A key requirement is that the simulator is robust enough to handle a large variety of machines that are produced through mutation and recombination. They may contain many singularities, degeneracies, and complex entangled kinematic chains that were not designed by a human and must be handled completely automatically. The machines were relatively slow so dynamic effects could be neglected. A typical run would evaluate on the order of 10^5 machines with 10–20 components on average. Some of the results of this evolutionary process are shown in Fig. 9(*a*). Some of the machines were fabricated and tested in reality where they performed as they did in simulation [6].

Biomechanical Simulation. The anatomy of musculoskeletal systems is notoriously complex, and no simulator currently exists that can model the interconnections between bone, ligaments, muscles, and the tendon network approaching its true complexity. Most current biomechanical simulators simplify this structure using bulk parameters and articulated joints. However, in order to correctly predict the outcomes of injury, repair, and treatment, more accurate and elastic simulators are needed. Difficulties with conventional simulations arise because of the complex wrapping of tendons around hard tissue. We have recently developed a simulator using relaxation methods, that is capable of handling these types of models. Current models of the finger as shown in Fig. 9(*b*) are able to predict important observed properties of these biomechanical systems that are not captured by simpler approaches [17].

Interactive Models of Historical Kinematic Machines. Franz Reuleaux (1829–1905) of Berlin created the world's largest collection of kinematic models at the Technical University of Berlin with over 800 models. Most of this collection was destroyed in the second world war, but Cornell University was fortunate to have purchased a substantial part of the Voigt-Reuleaux replica models in 1882. A history of this collection [19] shows that these models were used for teaching machine design up until the 1970s. A new generation of academics has again found these models useful for teaching and research, and we have decided to document and make the Collection available on the Internet [18,19]. The kinematic collection comprises 220 models from the Voigt catalog, a dozen models from the Schroeder works, and a Robert Willis model made in Paris [14], as well as various other instruments. The online museum contains most of the Reuleaux models with

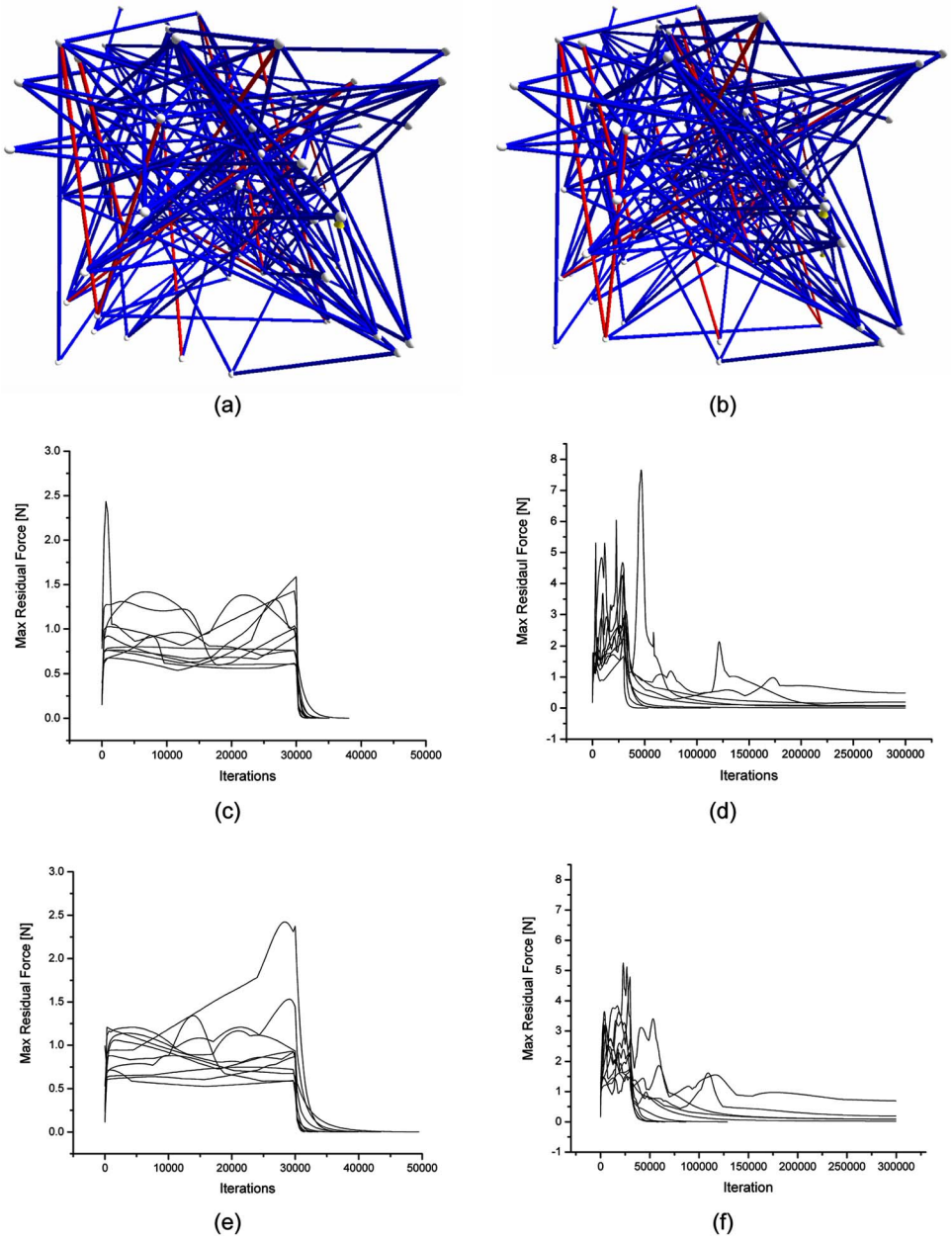


Fig. 8 Test case III: Random underconstrained mechanisms with 200 links and 100 degrees of freedom. (*a*) Initial state, (*b*) final actuated state; (*c*)–(*f*) progress of maximum residual force for ten random mechanisms (*c*) 100 DoF, synchronous relaxation, (*d*) 50 DoF, synchronous relaxation, (*e*) 100 DoF, asynchronous relaxation (*d*) 50 DoF, asynchronous relaxation.

mathematical and historic annotation, pictures, movies, animations, and simulations, as well as downloadable 3D-printing (stereolithography) files. The purpose of the simulations is to go beyond passive experiences to allow visitors to actively manipulate a mechanism, examine its behavior in within and outside it operating regime, and modify its parameters and topology. A critical aspect of this simulator is that it can be operated as a web applet and run in real time on a variety of platforms including relatively slow school machines. Most commercial simulators cannot be posted on the web for proprietary reasons, and publicly available codes are not easily rewritten as web applets. A simple java applet based on the relaxation code allowed us to post interactive simulations of about 30 machines. Two of these are shown in Fig. 9(*c*), and others can be viewed on site [18].

Conceptual Design by 3D Sketching. Freehand sketching is the informal written language of design, and with the recent advent of sketching hardware (like pen-based phones and tablet computers) there is a growing interest in sketched-based design. We have developed a computer-aided-design interface for creating 3D geometries through a combination of interactive 2D sketching and algorithms reverse projection from 2D to 3D. The relaxation algorithm is used in this system in two ways: First, it allows us to modify the geometry of the reconstructed model when user specified various constraints. To do this, the entire object is considered as a mechanism, and constraints are transformed into "actuators." The relaxation algorithm is also used to animate the entire object as a 3D mechanism, allowing users to sketch and explore 3D kinematics interactively [22]. The requirements that led to the use of the relaxation solver are the complexity of the mechanism (essentially a sketch), and the need for interactive-time performance.

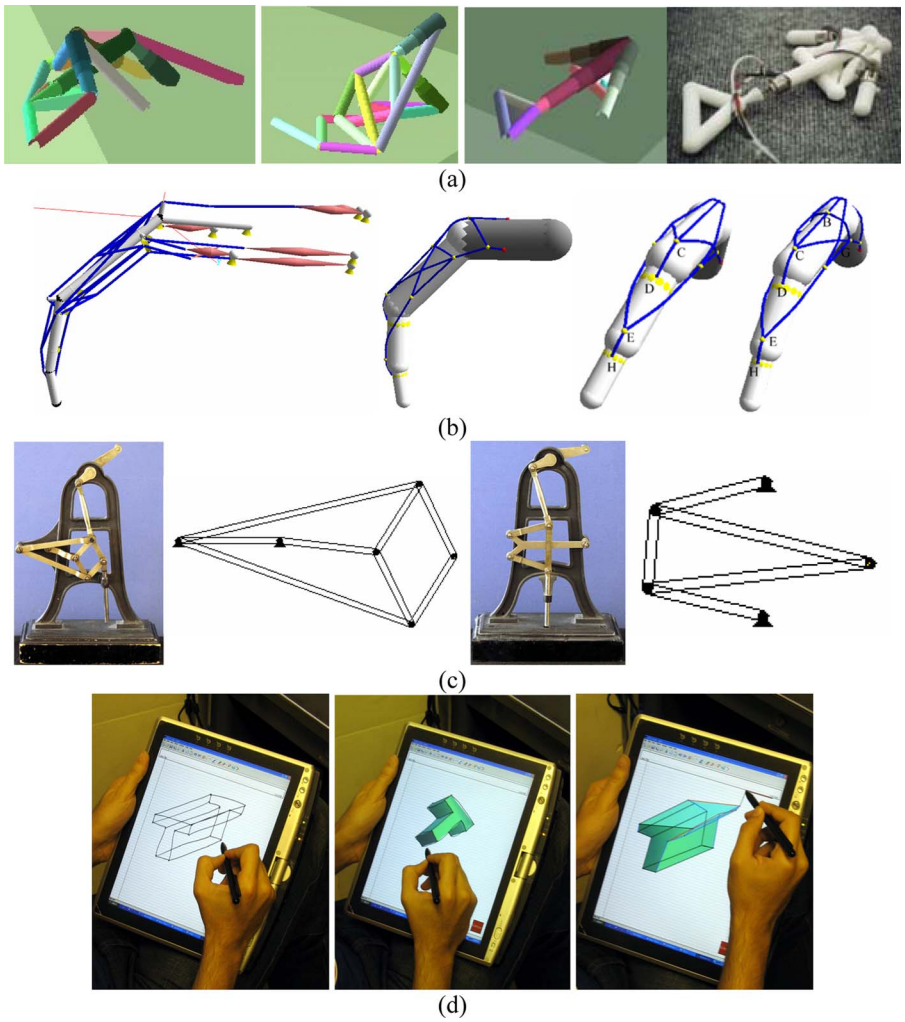


Fig. 9 Some application examples

The accuracy of the solver is not critical as long as qualitative behavior is preserved, since the sketch is a rough freehand drawing anyway.

Kinematic relaxation was also implemented in other applications, such as a 2D kinematics exploratory tool [20] and a kinematics topological design automation system [21].

Conclusions

This paper described a relaxation-based method for simulating 2D and 3D compound kinematic mechanisms. The relaxational process iteratively propagates node motions and degrees of freedom throughout a given actuated mechanism. Though relaxation-based methods have a long history, we exploit the propagation of the displacements during calculation process itself to reveal the kinematics of the structure. The method is slower than approaches based on solving simultaneous differential equations of motion, and does not account for dynamics, but provides several advantages: It achieves a higher level of accuracy, is more robust in handling transient singularities and degeneracies of the mechanism, and can handle more complex compound mechanisms with many links in multiple entangled kinematic chains. It also allows a straightforward introduction of linkages with nonlinear geometric constraints such as wrapping strings, hydraulics, actuators, contacts, and other arbitrary variable geometrical constraints. The relatively large number of applications that benefited from this algorithm and the relatively straightforward implementation may inspire revisiting relaxation methods in other engineering simulation domains.

Acknowledgment

This work was supported in part by U.S. National Insititute of Health Grant No. R01-AR052345 and by Department of Energy, Grant No. DE-FG02-01ER45902. An implementation of this simulation with a sketch-based interface is available for download at http://ccsl.mae.cornell.edu

References

[1] Ferguson, E. S., 1962, "Kinematics of Mechanisms from the time of Watt," Smithsonian Institution Bulletin No. 228, pp. 185–230 (available online at [18]).
[2] Reuleaux, F., 1876, "The Kinematics of Machinery: Outlines of a Theory of Machines," London: Macmillan (available online at [16]).
[3] Kent, S. L., 2002, "Engines and Engineering: What to Expect in the Future of PC Games," gamespy.com (http://archive.gamespy.com/futureofgaming/engines).
[4] Zatsiorsky, V. M., 1997, *Kinematics of Human Motion*, Human Kinetics Publishers, Champaign, IL.
[5] Sharma, G., Badescu, M., Dubey, A., Mavroidis, C., Tomassone, S. M., and Yarmush, M. L., 2005, "Kinematics and Workspace Analysis of Protein Based Nano-Actuators," J. Mech. Des., **127**(4), pp. 718–727.
[6] Lipson, H., and Pollack, J. B., 2000, "Automatic Design and Manufacture of Robotic Lifeforms," Nature (London), **406**, pp. 974–978.

[7] Woods, R. L., and Lawrence, K. L., 1997, *Modeling and Simulation of Dynamic Systems*, Prentice-Hall, Englewood Cliffs, NJ.
[8] Coutinho, M., 2001, *Dynamic Simulations of Multibody Systems*, Springer-Verlag, Berlin.
[9] Shabana, A. A., 2003, *Dynamics of Multibody Systems*, Cambridge University Press, Cambridge, England.
[10] Kramer, G., 1992, *Solving Geometric Constraint Systems*, MIT Press, Cambridge, MA.
[11] Sacks, E., and Joskowicz, L., 1993, "Automated Modeling and Kinematic Simulation of Mechanisms," Comput. Aided Geom. Des., **25**(2), pp. 106–118.
[12] Pu, P. 1991, "Simulating Both Dynamic and Kinematic Behaviors of Mechanical Mechanisms," Artif. Intell. Eng., **6**(3), pp. 136–155.
[13] Southwell, R. V. 1940, *Relaxation Methods in Engineering Science*, Oxford University Press, Oxford, England.
[14] Moon, F., 2003, "From Clocks to Chaos: Origins of Noise in Machines," *Proceedings of Modern Trends in Theoretical and Applied Mechanics Workshop*, University College, London, 23–24 April 2003.
[15] Innocenti, C., 2001, "Forward Kinematics in Polynomial Form of the General Stewart Platform," J. Mech. Des., **123**(2), pp. 254–260.
[16] Smith, R., 2004, Open Dynamics Engine, http://ode.org/
[17] Valero-Cuevas, F. J., and Lipson, H., 2004, "A Computational Environment to Simulate Complex Tendinous Topologies," *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, San Francisco, CA.
[18] Saylor, J., Walker, K., Moon, F. C., Henderson, D. W., Daimina, D., and Lipson, H., Cornell University Kinematic Models Digital Library (KMODDL), http://kmoddl.library.cornell.edu
[19] Lipson, H., Moon, F. C., Hai, J., and Paventi, C., 2004, "3D-Printing the History of Mechanisms," J. Mech. Des., **127**, pp. 1029–1033.
[20] Lipson, H., 2002, "Mechanisms: An Exploratory Tool for Mechanism Design," available at http://www.mae.cornell.edu/ccsl/downloads/mechanisms.htm
[21] Lipson, H., 2004, "How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis," *Proceedings of Genetic and Evolutionary Computation Conference*, Late Breaking Paper, GECCO, AAAI Press.
[22] Masry, M., Kang, D. J., and Lipson, H., 2005, "A Pen-Based Freehand Sketching Interface for Progressive Construction of 3D Objects," J. Compt. Graph., **29**, pp. 563–575.