Gauss’ least constraints principle and rigid body simulations
Stéphane Redon, Abderrahmane Kheddar, Sabine Coquillart

To cite this version:

HAL Id: hal-01147672
https://hal.archives-ouvertes.fr/hal-01147672
Submitted on 4 May 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Gauss’ Least Constraints Principle and Rigid Body Simulations

Stéphane Redon, Abderrahmane Kheddar, Sabine Coquillart

Abstract

Most of well-known approaches for rigid body simulations are formulated in the contact-space. Thanks to Gauss’ principle of least constraints, the frictionless dynamics problems are formulated in a motion-space. While the two formulations are mathematically equivalent, they are not computationally equivalent. The motion-space formulation is better conditioned, always sparse, needs less memory, and avoids some unnecessary computations. A preliminary experimental comparison suggests that an algorithm operating in the motion-space takes advantage of sparsity to perform increasingly better than a contact-space algorithm as the average number of contact points per object increases.

1 Introduction

Rigid body simulations have numerous offline or online applications: virtual environments, virtual prototyping, teleoperation, assembly tasks, interactive tolerance tests, video games. Many situations require the simulation of classical physics laws. This explains why rigid body simulation is such an active research field.

While moderately complex objects can be handled in real-time by the known simulation systems[4], complex objects may require many computations as the number of contact points increases, and the strategy is often to display complex objects while computing the simulation on simplified objects[9]. We believe that the reason for this is that the simulation systems do not make explicit use of the number of degrees of freedom (dof) in the simulation[1]. Thus, as the average number of contact points per object increases, the simulators are not as efficient as they could be.

Note that the average number of contact points per object can increase dramatically as the objects complexity increases. Consider two contacting cubes. If the cubes are aligned and one cube is slightly rotated, then eight contacts occur. If the cubes are replaced by two-sided polyhedral cylinders in a similar configuration, then 2n contacts occur. Other examples can easily be found: threaded screws insertion, tightly-packed objects, object clusters. Also, the fact that tolerance values are used in collision detection increases the number of contact points.

1. S. Redon and S. Coquillart are from INRIA-Rocquencourt - FRANCE.
2. A. Kheddar is from Université d’Evry CEMIF-SC - FRANCE
3. Contact author: stephane.redon@inria.fr

To our knowledge, most of recently proposed (constraint-based) solutions able to handle unilateral constraints formulate the dynamics problems (ie the constrained motion problem and the collision resolution problem) as a linear complementarity problem (LCP) (see for example [2][4][17][18][19]). The LCP relates contact forces and accelerations or velocities. For example, in a frictionless system, if \( \mathbf{a} \) and \( \mathbf{b} \) are two vectors in \( \mathbb{R}^m \) describing the normal contact forces and the relative normal accelerations at the \( m \) contact points, then there exists a \( m \times m \) matrix \( \mathbf{A} \) and a vector \( \mathbf{b} \) in \( \mathbb{R}^m \) such as:

\[
\mathbf{a}_p = \mathbf{A}\mathbf{f} + \mathbf{b}, \quad \mathbf{a}_p \geq 0, \quad \mathbf{f} \geq 0, \quad \mathbf{a}_p^T\mathbf{f} = 0
\]

As a result, the number of degrees of freedom of the unconstrained system is not explicit. In other words, problem (1) is a contact-space formulation[17].

This paper uses Gauss’ principle of least constraints to formulate both dynamics problems in the frictionless case in a motion-space. In the motion-space, an algorithm solving the dynamics problems is able to make explicit use of the number of dof in the simulation and avoids unnecessary computations. Note that some authors have presented formulations that explicitly contain the number of dof in the system. Lötstedt[13] uses a formulation closely related to Gauss’ principle, but doesn’t handle elastic collisions. Baraff[6] presents a linear-time algorithm for acyclic articulated bodies, but then uses a contact-space formulation[4] to handle unilateral contacts. Milenkovic[14], too, makes explicit use of the number of dof by formulating the problem as a quadratic programming problem, but requires many variables to enforce all the dynamics/collision conditions.

This paper is organized as follows. Gauss’ principle is recalled in Section 2. A motion-space formulation of the frictionless dynamics problems is derived from Gauss’ principle in Section 3 to form similar minimization problems. Section 4 reduces the minimization problems to a well known nearest-point problem (NPP) which will be our motion-based formulation. Section 5 compares both formulations from a theoretical and practical point of view. It shows that while the formulations are mathematically equivalent, they are not computationally equivalent: the motion-space formulation is better conditioned, is always sparse, requires less memory to store the data and avoids some unnecessary computations. An experimental comparison is made between two typical algorithms that solve the LCP problem and the NPP problem. A preliminary comparison suggests that, thanks to a sparser formulation, the algorithm operating in the motion-space (ie solving the NPP problem) is able to perform increasingly better than the algorithm operating in the contact-space, as the average
number of contact points per object increases. Section 6 concludes.

2 Gauss’ least constraints principle

Few references about applications of Gauss’ principle of least constraints[10] can be found. However, it has recently prompted a renewed interest[7][20]. Especially, Gauss’ principle has been used in the special case of initially motionless objects[5] and, in robotics, to compute the dynamics of redundant manipulators[8]. Udwindia[20] shows that Gauss’ principle leads to a closed-form equation of motion when all the constraints are bilateral. An immediate advantage of Gauss’ principle over the principle of virtual work, used in LCP methods, is that it allows to give a very intuitive formulation of the motion of a constrained system. Actually, it is so intuitive that it is often rediscovered and/or used without mentioning Gauss ([14]).

Though it was initially expressed for a set of point masses, it applies also for a frictionless system of rigid bodies subject to geometrical (unilateral or bilateral) constraints[7][12], and can be simply expressed thanks to generalized accelerations and masses. Let’s consider a contact group² of n mobile objects. A rigid body i has only six degrees of freedom, and its acceleration is split into a translational term $a_i(G_i)$, which is the acceleration of the object’s center of mass, and a rotational term $\alpha_i$. These two vectors are in $\mathbb{R}^3$. The accelerations of the n (potentially) mobile objects can be stacked in a single $6n$-dimensional generalized acceleration vector $a$, while the masses and (time-dependant) inertia tensors can be stacked in a single $6n \times 6n$ generalized mass matrix $M$. This matrix is block-diagonal and is symmetric positive definite (SPD).

Now, let $a_\circ$ denote the (generalized) unconstrained acceleration of the contact group, that is, the accelerations the contact group would have were it not subject to geometrical constraints². $a_\circ$ is in $\mathbb{R}^{6n}$. Gauss’ principle states that the generalized constrained acceleration $a$ minimizes the following scalar function of $a$ over the set of possible accelerations:

$$G_a(a) = \frac{1}{2} (a - a_\circ)^T M (a - a_\circ) = \frac{1}{2} \|a - a_\circ\|^2_M \quad (2)$$

The possible accelerations are the accelerations which are compatible with the current contact group’s configuration. Note that since $M$ is SPD, $\|\cdot\|_M$ is a well-defined non-euclidean norm. By analogy with the kinetic energy $E = \frac{1}{2}v^T M v$ of the system, this norm is usually called kinetic norm. Thus, Gauss’ principle amounts to minimize the kinetic distance between the generalized accelerations $a$ and $a_\circ$, over the set of possible accelerations⁴. Making implicit the fact that the distance is the kinetic one, Gauss’ principle can be stated even more simply: at any moment, a contact group’s constrained acceleration is the closest possible acceleration to its unconstrained acceleration.

In this formulation, the unknowns are the objects’ accelerations, and the contact forces are implicit: the problem is stated in the motion-space, and not in the contact-space, as in LCP methods[17]. Thus, the number of dof (6n) of the unconstrained contact group is explicit. Figure 1 shows an application of Gauss’ principle for a particle laid on a slope and subject to gravity⁵.

3 Dynamics problems formulation

3.1 Constrained motion

The constrained motion problem for a contact group of n objects is a direct application of Gauss’ principle, and all we have to do is express the set of possible accelerations. This set can be derived from the traditional contact model[3][4][17]. Let’s denote two contacting objects by $i$ and $j$. $I$ is the contact point, $n$ is the surface normal at $I$, directed from $j$ to $i$. Depending on the object it belongs to, $I$ is denoted by $I_i$ or $I_j$. Using this notation, the non-penetration constraint on the contact points’ accelerations is[3]:

$$\langle a(I_i) - a(I_j) \rangle . n + 2. (v(I_i) - v(I_j)) . \frac{dn}{dt} \geq 0 \quad (3)$$

In a contact-space approach, this constraint would be used to form the matrix $A$ and the vector $b$ of the introduction, which relate contact forces and normal accelerations at the contact points. However, using Gauss’ principle, we need to express

²Throughout this paper, a contact group is a set of mobile objects. Two objects $i$ and $j$ are in the same contact group if and only if there exists a chain of contacting mobile objects from object $i$ to object $j$. At any moment, any contact group is dynamically independent of the others. Therefore, in a rigid body simulation, the contact groups can be examined separately when solving the dynamics problems.

⁴For example, the unconstrained acceleration of a cube laid on a table and subject to gravity $g$ is precisely $g$.

⁵Since $G_a$ and $\sqrt{G_a}$ are minimized by the same $a_\circ$. For clarity, the examples given in this paper will concern particles. For isolated particles, the object space and the accelerations space can be superimposed, and the kinetic distance is proportional to the euclidean distance.
this constraint on objects’ accelerations. This is done simply: for any rigid object \( k \), we have
\[
a_k(\dot{x}_k) = a_k(G_k + \omega_k \wedge G_k \dot{x}_k + \omega_k \wedge (\omega_k \wedge G_k \dot{x}_k)),
\]
where \( \omega_k \) is the (known) rotational velocity of object \( k \). Since \( v_i(I_i), v_j(I_j) \) and \( dn/dt \) are known[3], inequation (3) is linear in the translational and rotational accelerations of the contacting objects. Stacking the \( m \) non-penetration constraints (3) yields a single constraint on the contact group’s acceleration \( J_a \geq c \), where \( J \) is the well-known \( m \times 6n \) jacobian[17], and \( c \) is in \( \mathbb{R}^m \). This general constraint defines the convex set of possible accelerations \( \{ a : J_a \geq c \} \).

Finally, the constrained accelerations are:
\[
a_c = \arg\min \left\{ \frac{1}{2} ||a - a_u||^2_M : J_a \geq c \right\}
\]
(4)

A common property of frictionless systems is immediately visible. Since \( a_c \) minimizes a (non-euclidean) distance over a convex set, it is unique. Note that equation (4) yields the constrained accelerations for a contact group containing any number of objects subject to any number of (unilateral or bilateral) constraints.

### 3.2 Resolving collisions

Let us now derive a formulation of the collision resolution problem in the velocities space from Gauss’ principle. Let \( v \) denote a generalized velocity of a contact group of \( n \) objects (\( v \) is in \( \mathbb{R}^{6n} \), and let \( v^- \) denote the (generalized) velocity that occurred immediately before the collision. The problem is to find \( v^+ \), the (generalized) velocity that will occur immediately after the collision. To do so, we make two classical assumptions[17]: the first assumption is that the collision duration is infinitesimal[16], and the second assumption is Poisson’s hypothesis[15], which relates the objects’ velocities at the contact points. As a result of the first assumption, we can consider that the object’s positions during the collision, as well as the constraints acting on them, remain unchanged. Moreover, the exterior actions can be neglected compared to the intensity of the actions of the constraints. Accordingly, \( a_u = 0 \). Let’s assume now that the generalized acceleration is constant over the whole duration of the collision \( dt \). This yields \( v^+ - v^- = a dt \). Since \( a_u = 0 \), Gauss’ principle allows to state that \( v^+ \) minimizes \( G \), where \( G(v) = \frac{1}{2} ||v - v^-||^2_M \), over the velocities that can occur immediately after the collision. These possible separating velocities are found simply. In the case of (possibly) simultaneous collisions, Poisson’s hypothesis yields the collision response constraint on the objects’ velocities, for any of the \( m \) contact points (colliding or not):
\[
(v_i^+ - v_j^+) \cdot n \geq -e(v_i^+ - v_j^+) \cdot n
\]
(5)

where \( e \) is the coefficient of restitution for the objects involved. Again, the velocities at the contact points depend on the objects’ translational and rotational velocities. Consequently, inequation (5) is a linear function of the translational and rotational velocities of objects \( i \) and \( j \). Stacking the \( m \) collision response constraints yields the set of possible (separating) velocities \( \{ v : Jv \geq d \} \), where \( J \) is a \( m \times 6n \) jacobian and \( d \) is in \( \mathbb{R}^m \). Finally, the velocities occurring immediately after the collision are:
\[
v^+ = \arg\min \left\{ \frac{1}{2} ||v - v^-||^2_M : Jv \geq d \right\}
\]
(6)

Once again, this result can be stated in a very elegant way: in a contact group, the velocities occurring immediately after a collision are the closest possible velocities to the velocities that occurred immediately before the collision. Figure 2 shows a collision resolution for a particle that has just collided a motionless object.

### 4 Solving the dynamics problems

Section 3 has reduced both dynamics problems to the same minimization problem:
\[
x^* = \arg\min \left\{ \frac{1}{2} ||x - x_u||^2_M : Jx \geq c \right\}
\]
(7)

where \( x_u, M, J \) and \( c \) are known. If the contact group contains \( n \) objects subject to \( m \) (unilateral) constraints, then \( x \) and \( x_u \) are in \( \mathbb{R}^{6n} \), \( c \) is in \( \mathbb{R}^m \), and \( J \) is in \( \mathbb{R}^{m \times 6n} \). Since \( M \) is SPD, it can be factored as the product of two positive definite \( 6n \times 6n \) matrices\(^7\): \( M = Q^T Q \).

It is now possible to make the contact forces (or impulses) visible, thanks to the lagrangian method. However, we will end with a motion-based formulation. The lagrangian function associated to problem (7) is:
\[
L(x, \lambda) = \frac{1}{2} ||x - x_u||^2_M - \lambda^T (Jx - c)
\]
(8)

The vector \( \lambda \) is in \( \mathbb{R}^m \) and represents contact forces or impulses, according to the dynamics problem being solved.

\(^{7}\)The objects’ local inertia tensors are constant and can be factored offline. The matrix \( Q \) can thus be computed in \( O(n) \).
Since 
\[
\frac{\partial L}{\partial x} = 0 \iff x = M^{-1}J^T\lambda + x_n, \tag{9}
\]
the variable \(x\) can be eliminated (temporarily losing sight of the objects’ motion). Thus, we must minimize:
\[
f(\lambda) = \frac{1}{2}J^TJM^{-1}J^T\lambda - \frac{1}{2}\lambda^T(c - Jx_n) \tag{10}
\]
under the constraint \(\lambda \geq 0\). Assuming the contact group’s state is consistent, there exists a vector \(s\) such that \(K = c\). We now use the factorization of \(M\) to retrieve a motion-based formulation. Let \(s = Q(k - x_n)\) and \(J_Q = JQ^{-1}\). Minimizing (10) is equivalent to the following non-negative least-squares problem (NNLSP):
\[
\lambda^* = \arg\min \left\{ \frac{1}{2}|J_Q^T\lambda - s|^2 : \lambda \geq 0 \right\} \tag{11}
\]
Since \(J_Q^T\lambda\) and \(s\) are both in \(\mathbb{R}^m\) and since, from equation (9), \(x = Q^{-1}J_Q^T\lambda + x_n\), this formulation is motion-based.

Geometrically, problem (11) consists in projecting \(s\) on the positive cone generated by the rows of \(J_Q\): \(C = \{ x : x = J_Q^T\lambda\text{ and }\lambda \geq 0 \}\). Problem (11) is thus a nearest-point problem (NPP).

5 Comparing formulations

5.1 Mathematical equivalence

The equivalent contact-space formulation is obtained from problem (11). Indeed, the necessary and sufficient conditions of problem (11) are:
\[
\begin{align*}
JM^{-1}J^T\lambda + Jx_n - c & \geq 0, \\
\lambda & \geq 0, \\
(JM^{-1}J^T\lambda + Jx_n - c)^T\lambda & = 0.
\end{align*} \tag{12}
\]

This formulation is exactly\(^8\) the one given in Baraff[4] or Ruspinii[17]. In this formulation, the number of dof in the contact group is implicit, since all vectors are in \(\mathbb{R}^m\), and since the factorization of the operational inertia matrix \(JM^{-1}J^T\) is hidden.

At this point, the reader may wonder why we did not derive problem (11) directly from the LCP formulation (12). After all, the only tricks involved are factoring the mass matrix \(M\) and noticing that \(Jx_n - c\) is in the column space of \(JQ^{-1}\). However, we believe that the insight into the intuitive underlying physics offered by Gauss’ principle will help to understand why an algorithm solving problem (11) should be able to perform better than an algorithm solving problem (12).

\(^8\)Note that for clarity, the formulations given in this paper are for 6-dof rigid bodies. However, Gauss’ principle holds in reduced coordinates, and can be used along with the general contact model of Ruspinii[17]. Besides, the mathematical equivalence given in this section is a proof of this, since all that is required is factoring \(M\).

5.2 Computational difference

Whereas problems (11) and (12) are mathematically equivalent, they are not computationally equivalent for essentially four reasons:

- **ill-conditionned matrix**: the condition number of \(JM^{-1}J^T\) is the square of that of \(J_Q\) (see Gill[11], for example). As a result, problem (12) is much more ill-conditionned than problem (11), and is therefore much more sensitive to round-off errors.

- **sparsity**: Since \(M\) and \(Q\) are block-diagonal matrices, and since \(J\) is always sparse, the matrix \(J_Q\) involved in problem (11) is always sparse as well. On the contrary, \(JM^{-1}J^T\) may be dense[6].

- **memory**: When the involved matrices are considered dense, it requires much more memory to store \(JM^{-1}J^T\) \((O(m^2))\) than to store \(J_Q\) \((O(nm))\) as the number of contact points per object increases. Moreover, when the matrices are considered sparse, \(J_Q\) always require \(O(m)\) only, while \(JM^{-1}J^T\) may still be dense.

- **unnecessary computations**: an algorithm solving problem (12) will have to enforce conditions on both contact forces (or impulses) \(\lambda\) and objects’ accelerations (or velocities) at the contact points \(x_{cp}\). As a result, this algorithm will have to maintain the \(m\) coordinates of \(x_{cp}\) while solving problem (12). However, the \(m\) coordinates of \(x_{cp}\) are not independent, since the contact points belong to rigid bodies: these coordinates could be deduced from the objects’ motions. Since these objects’ motions are not readily available in a contact-space formulation, the algorithm will perform unnecessary computations. In the motion-space formulation, an algorithm is able to operate on the \(6n\) truly independant motion coordinates only, and on the corresponding coefficients in \(J_Q\).

As a preliminary conclusion, let’s state the difference between a contact-space formulation and a motion-space formulation more simply: in a contact-space approach, the contact forces are computed for themselves, and are then used to compute the object’s motions. In a motion-space approach, the contact forces are computed along with the objects’ motions\(^9\).

5.3 Experimental comparison

There are numerous ways to solve the nearest-point problem (12). However, its geometrical structure has led the optimization community to develop specific algorithms, generally split into two categories: combinatorial algorithms (active-set methods), and descent methods. Algorithms in the first category look for the solution by moving from face to face in the cone. Algorithms in the second category are interior-point methods or exterior-point methods[1]. For typical rigid body simulations, however, it is simpler to use active-set algorithms.

\(^9\)Since the objects’ motions are directly related to \(J_Q^T\).
(as is the case in contact-space formulations: at least for frictionless systems, the LCP problems are solved by Lemke’s algorithm or Dantzig’s algorithm[2][4][17][18][19]).

In an attempt to make an experimental comparison of both approaches, we chose to implement Baraff’s well-known algorithm[4] to solve problem (12). This algorithm, in the frictionless case, is equivalent to Dantzig’s algorithm for LCPs. For the NPP problem (11), we chose Wilhelmsen’s algorithm[21] for several reasons. It is an active-set method, easy to implement and, most of all, is closely related to Dantzig’s algorithm, making thus the comparison more pertinent. Actually, as most pivotal methods, both algorithms have an outer loop and an inner loop. In the inner loop, both algorithms need to solve linear systems involving submatrices of $J^{-1}J^T$ (note that consequently, the assessment about ill-conditioning doesn’t hold anymore). However, this allows to compare efficiency under equivalent robustness assumptions. The difference between the algorithms resides in the fact that Wilhelmsen’s algorithm operates on $J^T\lambda$ and $\lambda$, while Dantzig’s algorithm operates on $x_p$ and $\lambda$. We used the same guidelines to implement both algorithms. The matrix $J_0$ and the matrix $J^{-1}J^T$ were treated as dense in a version of the algorithms, and as sparse in another. Cholesky factorizations were used to solve the linear systems in both cases. However, we did not implement an incremental factorization routine, for either algorithm, as suggested in [4]. Since preliminary tests did show that both algorithms performed approximately the same number of linear systems resolutions, no algorithm would have benefitted significantly more than the other. Note that the same tolerance value was used in both algorithms ($\varepsilon = 10^{-5}$).

Note that we are totally aware that, despite all our efforts to implement both algorithms with the same guidelines, it may still be argued that comparing execution times is not always pertinent. However, we are far less interested in the ratio of execution times than in the behavior of this ratio. For a given test, a random matrix $J$ was computed in the following way. For any row, two integers where randomly chosen between 1 and $n$. These (possibly equal) integers where used to place two sets of six coefficients in a row, resulting in a typical jacobian matrix. The fact that the chosen integers could be equal enabled to simulate object/environment contacts. To obtain a vector $c$ consistent with $J$, a random vector $k$ was chosen in $R^n$ and $c$ was set to $Jk$. The entries of the equivalent LCP problem were computed as in (12), with $M$ set to the $6n \times 6n$ identity matrix (which has obviously no impact on the comparison). The two parameters of the tests were the number of objects $n$ and the (average) number of unilateral constraints per object $p$. For a given pair $(n,p)$, 50 execution times were obtained for Dantzig’s and Wilhelmsen’s algorithms: $t^D$ and $t^W$, respectively, for $1 \leq i \leq 50$. The ratio of average execution times was then:

$$r(n,p) = \frac{\sum_{i=1}^{50} t_i^D}{\sum_{i=1}^{50} t_i^W}$$

Note that we chose $n$ and $p$ such as both problems could be stored in memory and didn’t need disk access. The results are reported in Table 1 for the dense case (ie dense for both algorithms), and in Table 2 for the sparse case (ie sparse for both algorithms). Again, let’s emphasize that we are not really interested in the ratio of execution times for itself, but in its behavior. This behavior is demonstrated in Tables 1 and 2: the algorithm operating in the motion-space takes (a better) advantage of sparsity to perform increasingly better than the algorithm operating in contact-space, as the number of contact points per object increases. Considering now the ratio in itself, and considering the fact that we used the same guidelines to implement both algorithms, we may (carefully) state that a motion-space algorithm performs better than a contact-space algorithm (at least for our implementation and for the tests we did). It should be clear that when the objects are simple and/or that there are few contacts per object, a contact-space approach will (not surprisingly) perform better. As the objects complexity increases, however, the motion-space formulation seems to be preferable.

### 6 Conclusion and results

Thanks to Gauss’ principle of least constraints, the dynamics problems occurring in a frictionless rigid body simulation (the constrained motion problem and the collision resolution problem) have been formulated in the motion-space. While the resulting formulations are mathematically equivalent, they are not computationally equivalent. The main reasons for this is that the motion-space formulation is better conditioned, is always sparse, takes less memory, and is able to avoid some unnecessary computations. A preliminary experimental study suggests that an algorithm operating in the motion-space is

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n=2$</th>
<th>$n=7$</th>
<th>$n=12$</th>
<th>$n=17$</th>
<th>$n=22$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.60</td>
<td>0.50</td>
<td>0.51</td>
<td>0.54</td>
<td>0.52</td>
</tr>
<tr>
<td>15</td>
<td>0.85</td>
<td>1.08</td>
<td>1.23</td>
<td>1.47</td>
<td>1.63</td>
</tr>
<tr>
<td>25</td>
<td>0.85</td>
<td>0.90</td>
<td>1.07</td>
<td>1.21</td>
<td>1.05</td>
</tr>
<tr>
<td>35</td>
<td>0.94</td>
<td>0.93</td>
<td>1.31</td>
<td>1.15</td>
<td>1.01</td>
</tr>
<tr>
<td>45</td>
<td>1.01</td>
<td>1.08</td>
<td>1.5</td>
<td>1.08</td>
<td>1.03</td>
</tr>
</tbody>
</table>

**Table 1:** Dense case: The ratio of execution times exhibits no significant behavior.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n=2$</th>
<th>$n=7$</th>
<th>$n=12$</th>
<th>$n=17$</th>
<th>$n=22$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.51</td>
<td>0.62</td>
<td>0.80</td>
<td>0.91</td>
<td>0.93</td>
</tr>
<tr>
<td>15</td>
<td>0.71</td>
<td>1.19</td>
<td>1.52</td>
<td>1.95</td>
<td>2.19</td>
</tr>
<tr>
<td>25</td>
<td>0.70</td>
<td>1.04</td>
<td>1.60</td>
<td>1.98</td>
<td>1.90</td>
</tr>
<tr>
<td>35</td>
<td>0.76</td>
<td>1.11</td>
<td>2.08</td>
<td>2.20</td>
<td>2.05</td>
</tr>
<tr>
<td>45</td>
<td>0.78</td>
<td>1.34</td>
<td>2.44</td>
<td>2.43</td>
<td>2.20</td>
</tr>
</tbody>
</table>

**Table 2:** Sparse case: the motion-space algorithm takes advantage of sparsity to perform increasingly better than the contact-space algorithm, as the number of contacts per object increases.


