INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2013; **96**:323–338
Published online 3 September 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4561

A constrained-optimization methodology for the detection phase in contact mechanics simulations

Alejandro M. Aragón[1,*,†], Vladislav A. Yastrebov[2] and Jean-François Molinari[1]

[1]*School of Architecture, Civil and Environmental Engineering (ENAC), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland*
[2]*Centre des Matériaux, MINES ParisTech, CNRS UMR 7633, BP 87, 91003 Evry, France*

SUMMARY

The detection phase in computational contact mechanics can be subdivided into a global search and a local detection. When potential contact is detected by the former, a rigorous local detection determines which surface elements come or may come in contact in the current increment. We first introduce a rigorous definition of the closest point for non-differentiable lower-dimensional manifolds. We then simplify the detection by formulating an optimization problem subject to inequality constraints. The formulation is then solved using different techniques from the field of mathematical optimization, for both linear and quadratic finite element meshes. The resulting general and robust detection scheme is tested on a set of problems and compared with other techniques commonly used in computational geometry. Copyright © 2013 John Wiley & Sons, Ltd.

Received 18 April 2013; Revised 10 July 2013; Accepted 25 July 2013

KEY WORDS: contact mechanics; contact detection; collision detection; constrained optimization

1. INTRODUCTION

Mechanical contact comprises an important class of engineering problems, for which the mathematical formulation and analytical/numerical solutions encounter considerable difficulties. Extensive work in the field has allowed to partly overcome these difficulties, which include the strong nonlinearity of the contact problem, its non-differentiability, and the inequality nature of the contact constraints [1, 2]. With a formulation based on rigorous energetic considerations, the finite element method (FEM) is at the backbone of contact numerical simulations. Widely used in structural mechanics, the FEM may provide accurate approximate solutions of specific boundary value problems including contact interactions. The use of the FEM allowed the field of computational contact mechanics to find applications in a wide range of disciplines, including the locomotive industry (e.g., vehicle crashworthiness analysis [3], assembled structures such as engines [4], tire-road, and wheel-rail contact [5, 6], breaking systems [7], bearings [8]), metal-forming [9, 10], structural mechanics (e.g., buckling of shells [11], failure of masonry walls [12]), impact [13, 14], tectonics [15–17], and even biomechanics through the study of human joints [18].

Nowadays, the general numerical treatment of the contact problem can be divided into two stages: the contact detection and the contact resolution. The latter draws attention from the applied mathematics [1, 19] and computational mechanics [2, 20, 21] communities. Due to the complexity of the mathematical formulation and the unsatisfactory results of the early methods [22], the field of computational contact mechanics presents a very active area of research. Several methodologies have

*Correspondence to: Alejandro M. Aragón, School of Architecture, Civil and Environmental Engineering (ENAC), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland.

†E-mail: alejandro.aragon@fulbrightmail.org

been proposed in the literature for the contact resolution stage. They are distinguished by discretization type (e.g., node-to-node, node-to-face, face-to-face [23, 24], contact domain method [25, 26]) and the resolution method behind (e.g., penalty, Lagrange multipliers and augmented Lagrangian [27–29]). Recently, particular attention has been paid to face-to-face based Nitsche [30] and mortar [31–33] formulations, which provide an accurate integration over the contact interface. Every contact discretization operates on different components of the contacting surfaces (nodes, integration points, faces, and segments), so the detection phase has to be adjusted accordingly. All in all, the contact resolution phase is in general preceded by a detection stage, whose robustness and efficiency determine the accuracy and promptness of the entire resolution scheme, respectively.

The contact detection phase can in turn be decomposed into two distinct sub-stages: *(i)* a *global search* that roughly determines potential contacting solids; and *(ii)* a thorough *local detection* that takes into account the actual solid discretization primitives, that is, nodes and faces. The global search relies on representing the solids using bounding volumes for which the overlapping tests are fast to compute. In this way, the local detection phase is only performed over the intersection region of global bounding volumes, if any. Both global and local searches behave differently whether an implicit or an explicit integration scheme is used. Within an implicit integration, the contacting parts have to be known before the contact occurs in order to construct the proper data structures that are able to treat the contact within the resolution stage [2]. Consequently, the bounding volumes used to roughly represent the objects or primitives are extended to take into account their expected locations at the end of the current increment. In contrast, an explicit integration scheme does not require any *a priori* knowledge of contacting pairs and relies on the detection of penetration within the current configuration. Although the global search is completely independent on the employed contact discretization, the local detection should be adapted to the discretization used. We will confine our discussion to the case of node-to-face discretizations, but the underlying ideas can be readily applied to face-to-face discretizations. As it is common practice, the determination of a node's closest point is based either on the search for its closest mesh nodes with further projection tests on adjacent faces, or just by testing for projections on closest mesh elements. However, such procedures are not robust [11, 34, 35] because *(i)* the closest point does not always belongs to the face attached to the closest node; and *(ii)* the normal projection used to determine the closest point does not always exist. The latter is due to the inherent non-smooth FE representation of the contacting surfaces [36]. This article proposes an alternative robust approach to establish the correct contact elements and location of the closest point within the local search. For that we formulate a specific constrained optimization problem and solve it using techniques from the field of mathematical optimization. A similar approach to the one outlined in this paper was first proposed in the field of computer science for the development of game engines [37, 38]. The proposed formulation does not rely on normal projections and can therefore be applied to non-smooth contacting surfaces. Several techniques from the field of mathematical optimization are investigated for solving the proposed formulation, and they are compared with other procedures based on computational geometry. Recommendations are given for an efficient implementation of the local detection depending on the polynomial order of the FE discretization.

This article is organized as follows: a brief summary of previous work is given in Section 2. Section 3 introduces the contact mechanics problem. Section 4 reviews the procedure traditionally used to obtain the closest point to a piece-wise smooth surface and provides an alternative approach based on constrained optimization. Finally, the results of applying the described techniques to a set of problems are given in Section 5.

2. PREVIOUS WORK

First attempts to solve the contact boundary value problem, in the context of finite element (FE) formulations, were confined to the case of matching discretizations and their infinitesimal relative motions [27]. The associated contact discretization, called *node-to-node*, did not require a contact detection phase as all possible contacting node pairs were known *a priori* and were not allowed to change with deformation. The use of a contact detection phase emerges with the need of treating

DOI: 10.1002/nme

finite relative motions or non-matching meshes. With significant changes in the configurations of the contacting bodies during loading, the detection phase may become the bottle-neck of contact simulations. When performing a simulation of n contacting bodies/elements, the excessive $\mathcal{O}(n^2)$ complexity of the simple all-to-all detection algorithm led to the development of more efficient strategies [11].

Traditionally, the numerical treatment of contact problems introduces geometrical asymmetry, that is, one contact surface is treated as master and another as slave. This asymmetric treatment is then used in both detection and resolution stages. For example, in node-to-face discretizations, the assignment of master and slave surfaces is determined by their corresponding mesh densities and by their relative stiffness. In many problems it is possible to assign master and slave surfaces in advance. Nevertheless, there are cases where this designation is hard or even impossible to discern, as for example in the analysis of self-contact problems (e.g., post-buckling behavior of thin-walled structures), complex geometries (e.g., foams under high pressures, hair-like assemblies), and large relative displacements of heterogeneous meshes. Benson and Hallquist [11] first introduced a procedure for the analysis of self-contact, naming it the *single surface contact algorithm*. The algorithm used a bucket-based search, which was borrowed from hidden-line algorithms used in computer visualization [39]. By dividing the search space into buckets, the time complexity of the contact detection is significantly reduced compared with the naive all-to-all algorithm. The bucket search is also described thoroughly by Heinstein *et al.* [34], who further describe an alternative search methodology based on binary sorting of nodes, and by Fujun *et al.* [40], who used a node-to-surface discretization for local detections and linked-lists for the placement of nodes into buckets. Malone and Johnson describe a detection methodology based on search planes of the faces of FEs [41]. In a recent publication [42], Yang and Laursen presented a contact detection methodology based on bounding volume hierarchies defined by discrete orientation polytopes ($k-$DOPs) in the context of large-deformation mortar formulations.

The detection phase increases its importance as the problem size increases, and in large problems its computational time accounts for a vast portion of the total time spent in the simulation, specially in explicit integration FE codes [42, 43]. Contact detection methodologies for large-scale simulations have also been proposed for distributed memory architectures [44, 45]. Malone and Johnson [44] conveyed the problem of load-imbalance and non-optimal efficiencies when applying their proposed *volume checking* algorithm based on the determination of bounding boxes resulting from a static mesh decomposition. Plimpton *et al.* [45] used a dynamic decomposition during the contact detection phase in order to achieve load balancing across processors, using *recursive coordinate bisectioning* (RCB) [46]. Yet, Attaway *et al.* [47] have reported that even with dynamic decomposition for load balancing, the contact detection phase takes between 30% to 60% of the computational time in the Cray Y-MP vector machine in transient solid dynamic simulations that combine the FEM with smoothed particle hydrodynamics. Some technical aspects of the bucket sort implementation for self-contact and distributed memory architectures are discussed by Yastrebov *et al.* [48].

3. PROBLEM DESCRIPTION

Contact problems in continuum mechanics are treated in a special way due to their geometrical discontinuous nature, as the contact interface transfers efforts between *separate* solids. This special treatment is also due to the strong nonlinearity, as the contact status and accompanied weak form may change abruptly, and also the friction force, which depends on the contact pressure, that is unknown before the solution is found. The contact conditions prescribed on surfaces are given as inequalities and rigorously, cannot be replaced by ordinary boundary conditions. The complexity comes from the reciprocal interaction between separate solids and the strong interconnection between kinematic and force quantities on both sides of the contact. For instance, the conditions of non-penetration and non-adhesion, also called the Hertz–Signorini–Moreau contact conditions [2], are formulated in terms of the contact pressure and the normal gap (or normal penetration)[‡].

[‡]These conditions are equivalent to the Karush–Kuhn–Tucker conditions in the field of mathematical optimization [49, 50], but called differently in the context of contact mechanics.

DOI: 10.1002/nme

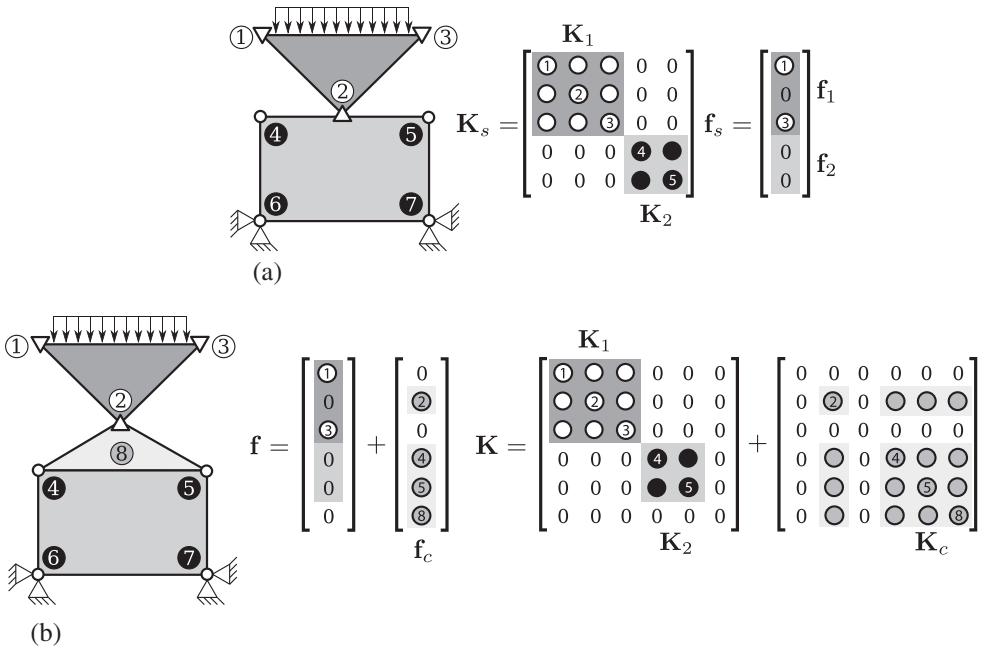


Figure 1. Example of two discretized solids in contact: finite element mesh, tangent matrix **K** and residual vector **f** for the cases (a) without contact, and (b) with contact constraints and additional degrees of freedom (node 8). Lower indices s and c refer to structural and contact parts, and indices 1 and 2 refer to upper and lower solids, respectively.

Thus, the interaction between deformable surfaces requires a coherent geometrical framework, that becomes even more important when contact problems are formulated within the FEM, where the surfaces are discretized and mutual penetrations are possible during the convergence of the resolution step. In the case of the numerical treatment, the geometrical similarity between surfaces is lost and different roles are assigned to them. This is performed in a way that the contact problem is formulated with respect to the penetration and sliding of parts of one (slave) surface with respect to another (master) surface.

A FE formulation can be obtained from the principle of virtual work, which is formulated as a weak (or integral) form of the balance of momentum. There are different ways to integrate the contact and friction conditions in this weak form. The particular form of the contact conditions that are included in the weak form, depends on the optimization method (e.g., penalty, Lagrange multipliers or augmented Lagrangian). For a detailed description the reader is referred to monographs [2, 35]. It is important to note that the contact conditions should be fulfilled locally at the level of the so-called *contact elements*, at which the normal gap, contact pressure, tangential traction, and the relative tangential displacement, can all be deduced during the convergence process.

The contact is incorporated into the system of equations via the change of the tangent matrix and the residual vector, as shown in Figure 1. Note that contrary to the penalty method, the method of Lagrange multipliers introduces additional degrees of freedom (the multipliers themselves) (see node 8 in Figure 1(b)). Contact elements are created based on the result of the detection procedure. In node-to-face discretizations, this translates to linking slave nodes to master faces, which may enter in contact during the resolution step (see element spanned by nodes 2-4-5 in Figure 1(b)). Therefore, it is crucial for the correct resolution step to possess a robust detection technique, which must link slave nodes with master faces correctly.

As explained in the introduction, the closest-node approach may fail in assigning the closest element to the contacting node. We recall that in a traditional master–slave approach, for each slave node ν^s of one surface, we first find the closest node ν^m on the opposite master surface, then we verify the faces adjacent to ν^m for proximity to ν^s. However, the closest face may not be connected

DOI: 10.1002/nme

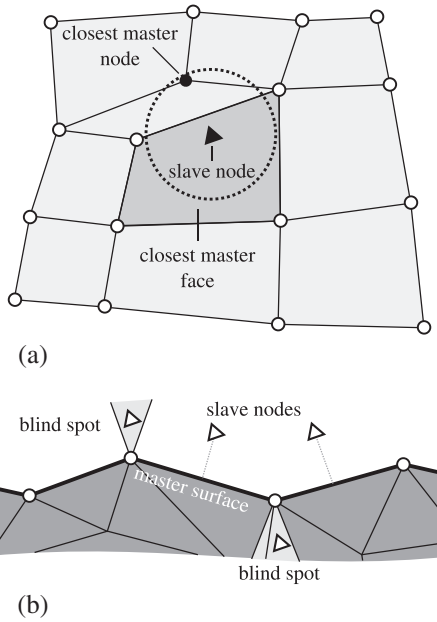


Figure 2. (a) Failure of the closest-node approach, the closest point lies on a face, which is not attached to the closest node; and (b) an example of so-called blind spots: regions where a normal projection on the master surface does not exist.

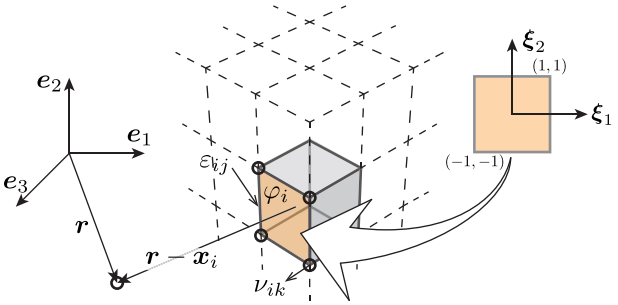


Figure 3. Schematic of the problem of finding the closest point from a point $\boldsymbol{r}$ to a face φ_i.

to the closest node (Figure 2(a)), and thus, the closest-node approach is not recommended for the general case. A more robust solution relies in finding the normal projections of nodes onto faces. Still, the closest element may be missed as the normal projection on faces does not always exist (Figure 2(b)). To improve the detection one should verify the proximity of the node not only to faces but also to edges and nodes of another surface [35]. In the following section, we suggest to combine these separate verifications (faces, edges, and nodes) in a single procedure, which results in a constrained optimization problem.

4. DETECTION OF THE CLOSEST POINT

A piece-wise smooth contact surface $\overline{\sigma}$ of a FE mesh can be considered as an assembly of smooth open faces φ_i and their closures $\partial\varphi_i$, that is, $\overline{\sigma} = \cup_i(\varphi_i \cup \partial\varphi_i)$. Each face is parametrized by the natural coordinate $\boldsymbol{\xi}$, as illustrated in Figure 3 for a quadrilateral face. For a given point $\boldsymbol{r} \in \mathbb{R}^d$, the problem is to find the closest point (or points) $\boldsymbol{x}_i(\boldsymbol{\xi}) \in \varphi_i$ on the surface $\overline{\sigma}$.

First, we consider a single smooth face φ_i and construct a distance functional

$$F_i(\boldsymbol{r}, \boldsymbol{x}_i(\boldsymbol{\xi})) = \frac{1}{2}[\boldsymbol{r} - \boldsymbol{x}_i(\boldsymbol{\xi})]^2, \quad \boldsymbol{x}_i(\boldsymbol{\xi}) \in \varphi_i. \tag{1}$$

DOI: 10.1002/nme

A normal projection $\boldsymbol{x}_i^{\perp} \equiv \boldsymbol{x}_i(\boldsymbol{\xi}^{\perp})$ of the point $\boldsymbol{r}$ on the open face φ_i exists, if and only if there is at least one coordinate $\boldsymbol{\xi}^{\perp}$ that satisfies the following system of equations (dropping the dependency of $\boldsymbol{x}_i$ on $\boldsymbol{\xi}$):

$$\frac{\partial F_i\left(\boldsymbol{r}, \boldsymbol{x}_i\right)}{\partial \xi_j} = \left(\boldsymbol{r} - \boldsymbol{x}_i\right) \cdot \frac{\partial \boldsymbol{x}_i}{\partial \xi_j} = 0, \tag{2}$$

where $j = 1, \ldots, d-1$. This system may have a unique solution, but it may also have no solutions or even an infinite number of them. If a unique solution exists, the corresponding normal projection is equivalent to the closest point if and only if

$$\det\left(\nabla_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\xi}} F\left(\mathbf{r}, \boldsymbol{x}_i\right)\big|_{\boldsymbol{\xi}^{\perp}}\right) \geqslant 0, \tag{3}$$

and the shortest distance between $\boldsymbol{r}$ and the ith face is given by

$$d\left(\boldsymbol{r}, \varphi_i\right) = \left\|\boldsymbol{r} - \boldsymbol{x}_i^{\perp}\right\|.$$

Let us denote by $\boldsymbol{X} = \cup_i \boldsymbol{x}_i^{\perp}$ the set of all points (finite or infinite) corresponding to the faces φ_i of the contact surface σ that satisfy conditions (2) and (3). Then, the global shortest distance from the point $\boldsymbol{r}$ to the surface σ is given by

$$d\left(\boldsymbol{r}, \sigma\right) = \min_{\forall \boldsymbol{x}_i^{\perp} \in \boldsymbol{X}} \left\|\boldsymbol{r} - \boldsymbol{x}_i^{\perp}\right\|,$$

and the closest point is determined as

$$\boldsymbol{x}^{\star} = \operatorname*{argmin}_{\forall \boldsymbol{x}_i^{\perp} \in \boldsymbol{X}} \left\|\boldsymbol{r} - \boldsymbol{x}_i^{\perp}\right\|. \tag{4}$$

4.1. Projection on faces, edges and vertices

Equation (4) does not always provide us with a solution because it obtains the closest point by only considering projections on separate faces. In general, the closest point does always exist for a surface $\overline{\sigma}$ but it does not always correspond to the solution of the minimization problem given by Equations (2) and (3). Nevertheless, even though the latter may not be unique, the probability to encounter an equally distant $\boldsymbol{r}$ from several points of the surface σ is negligible from a practical point of view. The problem with Equation (4) arises when the closest point is not located on a face but on an edge or a node. In this case, we cannot determine this point by solving Equation (2) as on edges and vertices $\nabla_{\boldsymbol{\xi}} \boldsymbol{x}$ is not defined. To avoid this incompatibility, edges and nodes are accounted for when computing the closest node [35]. Thus, in addition to the distance functional for faces given by Equation (1), let the distance functional defined over edges be

$$E_{ij}\left(\boldsymbol{r}, \boldsymbol{x}_{ij}\left(\boldsymbol{\zeta}\right)\right) = \frac{1}{2}\left[\boldsymbol{r} - \boldsymbol{x}_{ij}\left(\boldsymbol{\zeta}\right)\right]^2, \quad \boldsymbol{x}_{ij} \in \varepsilon_{ij} \subset \varphi_i, \tag{5}$$

where $\boldsymbol{\zeta}$ is the natural coordinate over the jth edge of face φ_i. As before, we drop the explicit dependence of $\boldsymbol{x}_{ij}$ on $\boldsymbol{\zeta}$. The normal projection $\boldsymbol{x}_{ij}^{\perp} \equiv \boldsymbol{x}_{ij}(\boldsymbol{\zeta}^{\perp})$ of a point $\boldsymbol{r}$ over edge ε_{ij} is obtained by solving

$$\left(\boldsymbol{r} - \boldsymbol{x}_{ij}\right) \cdot \frac{\partial \boldsymbol{x}_{ij}}{\partial \zeta_k} = 0. \tag{6}$$

Again, we denote by $\boldsymbol{Y} = \cup_i \left(\cup_j \boldsymbol{x}_{ij}^{\perp}\right)$ the set of all points that solve Equation (6) on all edges. Finally, let $\boldsymbol{Z}$ denote the set of coordinates corresponding to all nodes in the mesh, that is, $\boldsymbol{Z} = \{\cup \boldsymbol{x}_{ik}\}$ where $\boldsymbol{x}_{ik} \equiv \nu_{ik} \subset \varphi_i \subset \sigma$. Considering the contact surface $\overline{\sigma}$ as an ensemble of its faces $\cup_i \varphi_i$, together with their corresponding edges $\cup_i \cup_j \varepsilon_{ij}$ and nodes $\cup_i \cup_k \nu_{ik}$, that is,

$$\overline{\sigma} = \cup_i \left(\varphi_i \ \cup_j \varepsilon_{ij} \ \cup_k \nu_{ik}\right),$$

DOI: 10.1002/nme

the shortest distance from a point $\boldsymbol{r}$ to $\overline{\sigma}$ is given as

$$d(\boldsymbol{r},\overline{\sigma}) = \inf \begin{cases} \min\limits_{\forall \boldsymbol{x}_i^{\perp} \in \boldsymbol{X}} \left\| \boldsymbol{r} - \boldsymbol{x}_i^{\perp} \right\|, \\ \min\limits_{\forall \boldsymbol{x}_{ij}^{\perp} \in \boldsymbol{Y}} \left\| \boldsymbol{r} - \boldsymbol{x}_{ij}^{\perp} \right\|, \\ \inf\limits_{\forall \boldsymbol{x}_{ik} \in \boldsymbol{Z}} \left\| \boldsymbol{r} - \boldsymbol{x}_{ik} \right\|. \end{cases} \tag{7}$$

The closest point is determined according to the shortest distance, that always exists and can be obtained by Equation (7). Though, a separate consideration of faces, edges, and vertices may complicate the implementation of a program.

4.2. Constrained optimization approach

To avoid the separate consideration of lower dimensional manifolds, as outlined previously, the closest point for a given face $\overline{\varphi}$ may be given as the solution of the following minimization problem

$$\boldsymbol{x}_i^{\star} = \underset{\boldsymbol{g}(\boldsymbol{\xi}) \geq \boldsymbol{0}}{\operatorname{argmin}} \; F_i\left(\boldsymbol{r}, \boldsymbol{x}_i\left(\boldsymbol{\xi}\right)\right), \tag{8}$$

where $\mathbf{g}(\boldsymbol{\xi})$ is a vector of constraints that depends on the type of element outlined over the natural coordinate $\boldsymbol{\xi}$, for example, $\boldsymbol{g}(\boldsymbol{\xi}) = \{\xi_1 + 1, 1 - \xi_1, \xi_2 + 1, 1 - \xi_2\}$ for the quadrangular element shown in Figure 3. This problem has at least one solution for any $\boldsymbol{r}$ and any given face φ_i, and we again denote by $\boldsymbol{X} = \cup_i \, \boldsymbol{x}_i^{\star}$ their ensemble. Considering the piece-wise smooth contact surface $\overline{\sigma}$ as an ensemble of closed-set faces, that is, $\overline{\sigma} = \cup_i \varphi_i \cup_j \partial_j \varphi_i$, the shortest distance to the surface

$$d(\boldsymbol{r},\overline{\sigma}) = \inf_{\forall \boldsymbol{x}_i^{\star} \in \boldsymbol{X}} \left\| \boldsymbol{r} - \boldsymbol{x}_i^{\star} \right\|,$$

can be shown to be equivalent to the shortest distance found according to Equation (7).

The optimization problem stated by Equation (8) can be solved by defining the Lagrangian

$$\mathcal{L}_i\left(\boldsymbol{\xi}, \boldsymbol{\lambda}\right) = F_i\left(\boldsymbol{r}, \boldsymbol{x}_i\left(\boldsymbol{\xi}\right)\right) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{g}\left(\boldsymbol{\xi}\right), \tag{9}$$

where $\boldsymbol{\lambda}$ is a vector of the Lagrange multipliers [51]. For a given point $\boldsymbol{r}$, the combination $\left(\boldsymbol{\xi}^{\star}, \boldsymbol{\lambda}^{\star}\right)$ is said to be a local solution to the optimization problem if it satisfies the Karush–Kuhn–Tucker conditions [49,50]:

$$\begin{aligned} \nabla \mathcal{L}_i\left(\boldsymbol{\xi}^{\star}, \boldsymbol{\lambda}^{\star}\right) &= \boldsymbol{0}, \\ g_i\left(\boldsymbol{\xi}^{\star}\right) &\geq 0, \\ \lambda_i^{\star} &\geq 0, \\ \lambda_i^{\star} g_i\left(\boldsymbol{\xi}^{\star}\right) &= 0. \end{aligned} \tag{10}$$

The first and the second gradients of the Lagrangian are given by[§]

$$\nabla \mathcal{L}\left(\boldsymbol{\xi}, \boldsymbol{\lambda}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\xi}} \mathcal{L} \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L} \end{bmatrix}, \quad \nabla\nabla \mathcal{L}\left(\boldsymbol{\xi}, \boldsymbol{\lambda}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\xi}} \mathcal{L} & \nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\xi}} \mathcal{L} \\ \nabla_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\lambda}} \mathcal{L} & \nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}} \mathcal{L} \end{bmatrix}, \tag{11}$$

where the first gradients, dropping the dependence of $\boldsymbol{x}$ and $\boldsymbol{g}$ on $\boldsymbol{\xi}$, are

$$\begin{aligned} \left(\nabla_{\boldsymbol{\xi}} \mathcal{L}\right)_i &= \left(\boldsymbol{r} - \boldsymbol{x}\right) \cdot \frac{\partial \boldsymbol{x}}{\partial \xi_i} + \boldsymbol{\lambda}^{\mathsf{T}} \frac{\partial \boldsymbol{g}}{\partial \xi_i}, \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L} &= \boldsymbol{g}. \end{aligned} \tag{12}$$

[§]Some optimization methodologies may require the computation of the second gradient of the Lagrangian to obtain the search direction.

DOI: 10.1002/nme

and the second gradients, considering $\boldsymbol{\xi} = (\xi_1, \xi_2)$[¶], are given by

$$
\begin{aligned}
\nabla_{\xi_1}\nabla_{\xi_1}\mathcal{L} &= (\boldsymbol{r}-\boldsymbol{x})\cdot\frac{\partial^2\boldsymbol{x}}{\partial\xi_1^2} - \left(\frac{\partial\boldsymbol{x}}{\partial\xi_1}\right)^2,\\
\nabla_{\xi_1}\nabla_{\xi_2}\mathcal{L} &= \nabla_{\xi_2}\nabla_{\xi_1}\mathcal{L} = (\boldsymbol{r}-\boldsymbol{x})\cdot\frac{\partial^2\boldsymbol{x}}{\partial\xi_1\partial\xi_2} - \frac{\partial\boldsymbol{x}}{\partial\xi_1}\cdot\frac{\partial\boldsymbol{x}}{\partial\xi_2},\\
\nabla_{\xi_2}\nabla_{\xi_2}\mathcal{L} &= (\boldsymbol{r}-\boldsymbol{x})\cdot\frac{\partial^2\boldsymbol{x}}{\partial\xi_2^2} - \left(\frac{\partial\boldsymbol{x}}{\partial\xi_2}\right)^2,\\
\nabla_{\xi_1}\nabla_{\boldsymbol{\lambda}}\mathcal{L} &= \nabla_{\boldsymbol{\lambda}}\nabla_{\xi_1}\mathcal{L} = \frac{\partial\boldsymbol{g}}{\partial\xi_1},\\
\nabla_{\xi_2}\nabla_{\boldsymbol{\lambda}}\mathcal{L} &= \nabla_{\boldsymbol{\lambda}}\nabla_{\xi_2}\mathcal{L} = \frac{\partial\boldsymbol{g}}{\partial\xi_2},\\
\nabla_{\boldsymbol{\lambda}}\nabla_{\boldsymbol{\lambda}}\mathcal{L} &= 0.
\end{aligned}
\tag{13}
$$

The second gradient of the constraint function vanishes because all constraints are linear. This is because the FE has straight edges in natural coordinate space regardless of its polynomial interpolation order. It is also worth noting that the first right-hand term in expressions for $\nabla_{\xi_1,\xi_2}\nabla_{\xi_1,\xi_2}\mathcal{L}$ vanishes for linear FEs.

Note that the computation of the Hessian matrix $\nabla\nabla\mathcal{L}(\boldsymbol{\xi},\boldsymbol{\lambda})$ may be avoided for many optimization methodologies. A quasi-Newton approximation that is obtained by Broyden–Fletcher–Goldfarb–Shanno (BFGS) updates can be used instead [51]. Even though this technique increases the number of iterations needed to obtain an optimal solution with respect to using the consistent Hessian, the optimization algorithms are more robust as they guarantee the use of a positive-definite approximation of the Hessian matrix in the search.

The optimization problem stated by Equation (8) could be solved by an *active-set* strategy that identifies the active constraints at the solution. Starting from a feasible point, the active-set method finds an optimum solution by solving a series of quadratic subproblems where some inequality constraints may be treated as equality constraints (and are therefore identified as the active-set). In a primal active-set strategy, new iterates are guaranteed to remain feasible by the aid of a step-length parameter. *Interior-point* methods provide yet another technique for solving Equation (8). The original problem is transformed into the following equally-constrained optimization:

$$
\boldsymbol{x}_i^{\star} = \underset{\boldsymbol{z}-\boldsymbol{g}(\boldsymbol{\xi})=\boldsymbol{0}}{\operatorname{argmin}} \; F_i\left(\boldsymbol{r}, \boldsymbol{x}_i\left(\boldsymbol{\xi}\right)\right) - \mu\sum_i \ln\left(z_i\right), \tag{14}
$$

where $\mu > 0$ is the barrier parameter, and z_i are positive slack-variables. Finally, Equation (8) could also be solved by *sequential quadratic programming* (SQP), where a solution is also determined by replacing the original problem into a sequence of quadratic subproblems. The SQP implementation chosen for this work uses a merit function to determine the new iterates through a line search and BFGS approximations of the Hessian matrix. For more details on the different optimization techniques and their implementations, the reader is referred to Nocedal and Wright [51], and the references therein.

5. NUMERICAL EXPERIMENTS

The results of applying the proposed methodology to two scenarios are reported next. First, the formulation based on mathematical optimization is investigated with a single FE of varying polynomial interpolation order. A comparison is carried out between the different optimization methodologies. The starting guess used in the optimization is also examined. Then, the procedure is investigated in the context of a large-scale problem involving two rough surfaces in proximity. In the

[¶]This formulation corresponds to a three-dimensional surface, where the contact faces are parametrized by a two-dimensional coordinate.

DOI: 10.1002/nme

latter, the methodology is compared with more traditional techniques borrowed from computational geometry. The computational times reported henceforth are obtained running the MATLAB (The MathWorks, Inc., Natick, Massachusetts, USA.) implementation of the considered methods on a computer with a 2.6 GHz Intel Core i7 processor and 16 GB of RAM memory.

5.1. Single element optimization study

This section summarizes the optimization tests performed on FEs of both linear and quadratic interpolation functions. Each result reported is the outcome of 10,000 optimizations, as the position of point $\boldsymbol{r}$ is moved over a 100×100 square lattice outlined on $\Theta \subset \mathbb{R}^2$ on the same plane of the FE. The domains are $\Theta = [-2, 2] \times [-2, 2]$ and $\Theta = [-6, 6] \times [-6, 6]$ for linear and quadratic elements, respectively. A comparison of the different methodologies available for the optimization is also performed.

The effect of the starting guess $\boldsymbol{\xi}^0$ is also investigated, by selecting it from either of the following alternatives:

(I) all corners and the center of the element are tested for the proximity to the slave node, and the closest one is chosen as the initial guess,
(II) the center of the element is chosen as the initial guess.

For both linear and quadratic triangular elements, the center is $\boldsymbol{\xi} = (1/3, 1/3)$. Similarly, for the quadrangular elements, the center is $\boldsymbol{\xi} = (0, 0)$. Yet, for the computations that use the interior-point method, only option **II** is used.

The constraints could be taken into account as linear or nonlinear functions, or as bounds on the variables. For the triangular elements, the constraint vector is $\boldsymbol{g}(\boldsymbol{\xi}) = \{\xi_1, \xi_2, 1 - \xi_1 - \xi_2\}$ regardless of the interpolation order. However, two of these constraints can be imposed as lower bounds because $\xi_1 \geqslant 0$ and $\xi_2 \geqslant 0$. The vector of constraints for quadrangular elements is given in Section 4.2, but the four constraints can be imposed as bounds $-1 \leqslant \xi_i \leqslant 1$. Yet another way to impose the constraints takes advantage of their linearity, thus $\boldsymbol{g}(\boldsymbol{\xi}) \geqslant \mathbf{0}$ can be written as $\mathbf{A}\boldsymbol{\xi} \geqslant \mathbf{b}$.

Figure 4 shows the results of the optimization for different types of FEs using SQP [51], even though other methodologies are investigated later on. Each plot in Figure 4 shows the number of iterations that the algorithm requires to converge to a local optimum, that is, the solution to Equation (8). The results for a three-node triangular element are illustrated in Figure 4(a), where dashed lines show the mapped equations $\xi_1 = 0$, $\xi_2 = 0$ and $\xi_1 + \xi_2 = 1$, thus delineating the location of the triangular element. As seen in the figure, there are regions within the plane of the triangular element that require a single iteration to converge. These regions correspond to the Voronoi regions of the triangle's vertices. Given a triangle $\varphi_i \subset \mathbb{R}^d$, the Voronoi region (also called Dirichlet region) of its vertex v_{ik} (with coordinate $\boldsymbol{x}_{ik}$) is defined as

$$V(v_{ik}) := \{\boldsymbol{x} \,|\, \|\boldsymbol{x} - \boldsymbol{x}_{ik}\| \leqslant \|\boldsymbol{x} - \boldsymbol{y}\|, \boldsymbol{y} \in \varphi_i \setminus \boldsymbol{x}_{ik}\}.$$

The number of iterations increases to 2 or 3 in locations that have an orthogonal projection to one of the edges of the triangle, to a maximum number of eight iterations in some locations laying within the triangle. The starting guess for the plots in this figure is chosen as the point within the parent element closest to $\boldsymbol{r}$, taken between the element center and each of its corners (option **I**). As a result, an average of 1.6 iterations is executed for each optimization on the linear triangle case when considering all 10,000 optimizations. By choosing only the center of the element as the starting guess (option **II**), the average number of iterations needed to converge increases to 2.8 iterations per optimization. Similar results are shown in Figure 4(b) for a four-node bilinear quadrangular element, where the dashed lines correspond to $\xi_1 = \pm 1$ and $\xi_2 = \pm 1$. As in the previous figure, a single iteration is sufficient in those locations outside the element that do not have a normal projection to its edges. The number of iterations is again maximized in locations within the element. For the linear quadrangular element, the average number of iterations increases from 2.4 to 4.7 when choosing only the center of the element as the starting guess (i.e., going from option **I** to **II**).

Figure 4(c) and (d) correspond to the results obtained with 6-node quadratic and 8-node biquadratic elements, respectively. Dashed lines are now curved because of the higher polynomial

DOI: 10.1002/nme

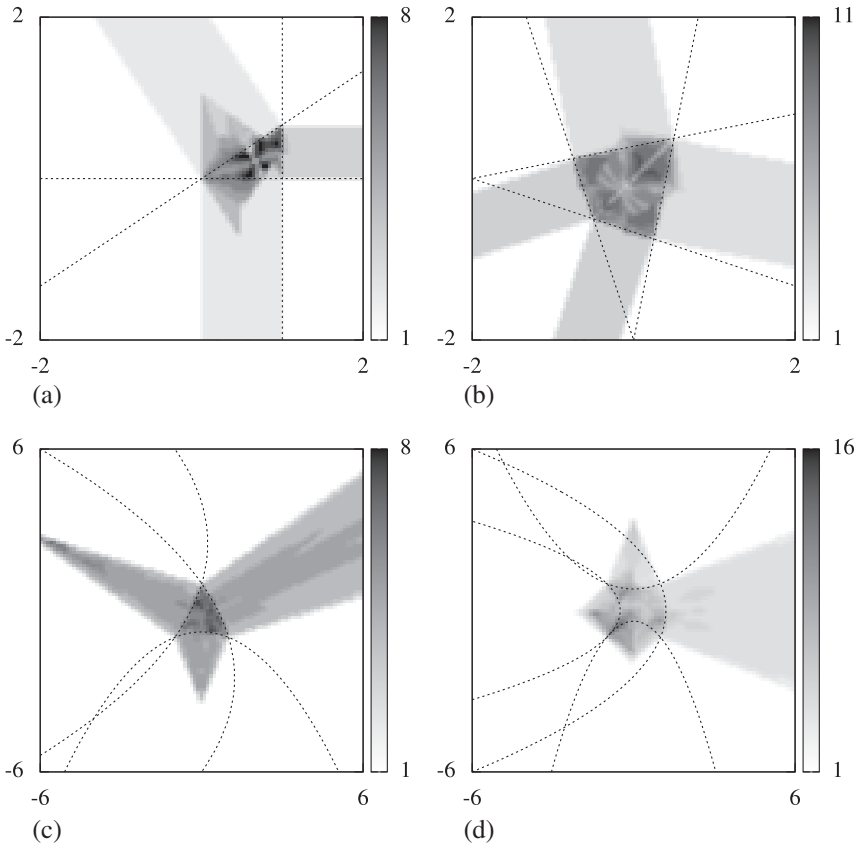


Figure 4. Results for the optimization problem stated by Equation (8) to find the minimum distance from a point $\boldsymbol{r} \in \mathbb{R}^2$ to an element of type: (a) three-node linear triangle, (b) four-node bilinear quadrangle, (c) six-node quadratic triangle, and (d) eight-node bi-quadratic quadrangle. Each figure shows a color map of the number of iterations required by the sequential quadratic programming algorithm to converge to a local optimum, and it is the result of 10,000 computations, as the point $\boldsymbol{r}$ moves on a 100×100 square lattice laid out on the same plane of the element.

order of the elements[‖]. The results show that concave edges reduce the number of locations that contain a normal projection on edges, thus they increase the region where the algorithm converges in a single iteration. On the contrary, convex edges increase the number of locations with a normal projection, as shown by the right edges of both elements. As with the case of linear elements, the starting guess influences the number of iterations that the algorithm takes to converge to the optimum. For the quadratic triangular element, the average number of iterations increases from 1.8 (**I**) to 3.2 (**II**), consequently raising the computational time by 12%. Likewise, this number increases from 1.7 (**I**) to 3 (**II**) for the quadratic quadrangle, but this time with only 2% additional computational cost. The extra time spent in computing the closest point in the parent element balances the additional time spent per iteration. Option **I** is used henceforth for the remainder of the results.

Figure 5 presents a comparison with other methods used in constrained optimization. The plots in the figure are to be compared with those for the SQP optimization given for the quadratic triangular and quadrangular elements of Figures 4(c) and (d), respectively. In the latter, the SQP optimization run on an average of 6.5 μs for the triangular and 6.7 μs for the quadrangular element, both with a standard deviation $s = 1.5\,\mu$s. Figure 5(a) and (b) show similar results by using the interior-point method. As the results of the optimization illustrate, this method takes considerably more iterations to converge than the SQP. An average of 10.8 iterations is obtained for the triangular element, with

[‖]And because the nodes over edges are not aligned in the current configuration.

DOI: 10.1002/nme

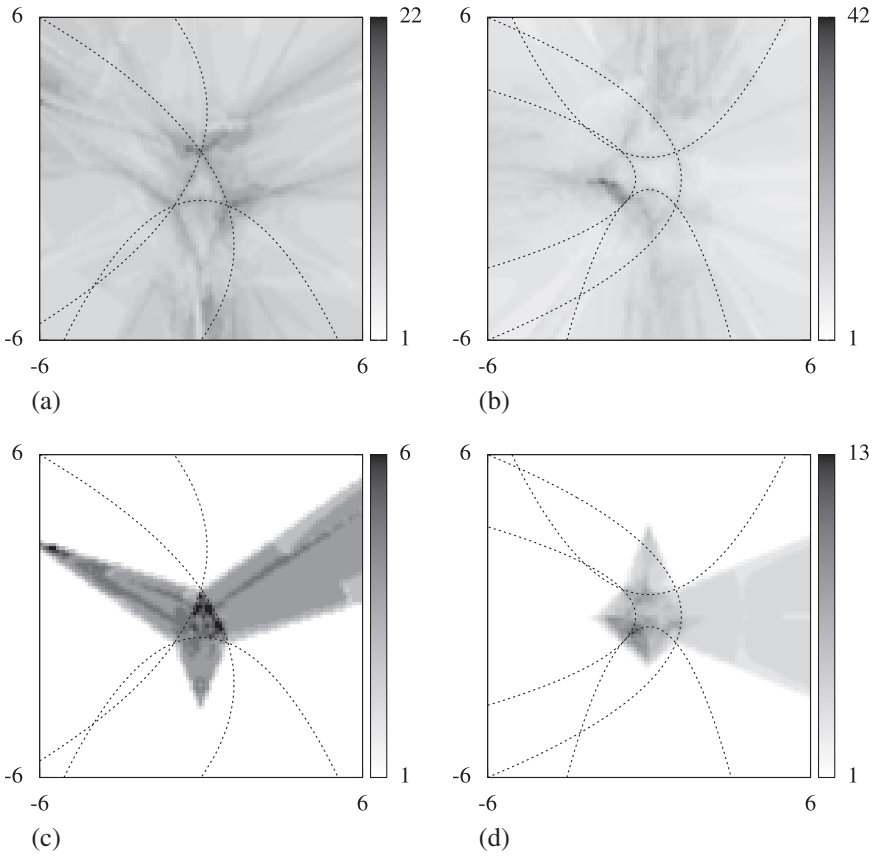


Figure 5. Comparison with interior-point (a and b) and active-set (c and d) methods.

an average time of 24.4 μs per iteration. Likewise, for the quadrangular element an average of 9.8 iterations is obtained, with a slightly lower average computational time of 22.8 μs. The interior-point method does not only take more time to converge, but there is also a region where the algorithm had difficulties in converging to the solution. Finally, Figure 5(c) and (d) show the results of yet another method, the active-set methodology. This method gives similar results to those obtained by SQP, as there is an average of 1.7 iterations for each quadratic triangle optimization, with an average time of 5.5 μs. With the same number of average iterations, the average computational time for the quadrangular element is 5.4 μs. This method requires roughly 20% less computational time than the SQP implementation.

Table I summarizes the results hitherto for both linear and quadratic FEs. Each reported result is, as explained before, the average obtained over 10,000 optimizations. It is worth noting that comparisons between the different methodologies cannot be drawn between linear and quadratic elements, as their corresponding computational domains are different.

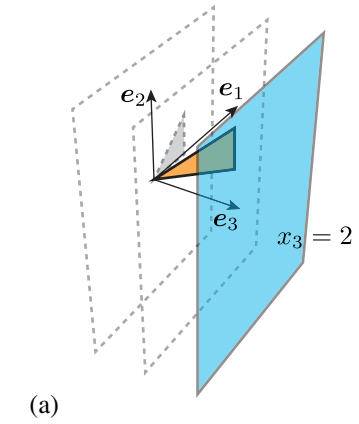
For the three-dimensional optimization results given in Figure 6, the point $\boldsymbol{r}$ is laid out on a subset of the plane $x_3 = 2$, that is, $\Theta = [-4, 4] \times [-4, 4] \times 2 \subset \mathbb{R}^3$. This plane does not correspond to that of the triangle, as the right-most nodes of the triangular element used to obtain the results in Figure 4(a) now have a coordinate $x_3 = 1$, and the left node a coordinate $x_3 = 0$. The schematic of the problem is presented in Figure 6(a). The optimization results of Figure 6(b) show similar results as those presented in Figure 4(a), where the locations with the most number of iterations correspond to those that contain a normal projection laying within the face of the element.

The average computational times are 6.7 μs and 5.8 μs for the SQP implementation and active-set implementations, respectively.

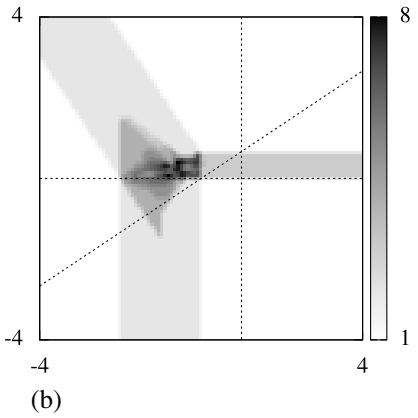
DOI: 10.1002/nme

Table I. Summary of the comparison between the different optimization strategies applied to both linear and quadratic finite elements. The table lists the number of iterations and the computational time in μs, as averages of 10,000 optimizations.

		linear		quadratic	
					
SQP	iterations	1.6	2.4	1.8	1.7
	CPU time	6	7.4	6.5	6.7
IPM	iterations	9	7.6	10.8	9.8
	CPU time	21.1	19	24.4	22.8
active-set	iterations	1.5	2.2	1.7	1.7
	CPU time	5.2	6	5.5	5.4



(a)



(b)

Figure 6. (a) Configuration of the problem in three-dimensional space, where the point $\boldsymbol{r} \in \mathbb{R}^3$ moves on a 100×100 square lattice laid out on a different plane (subset of $x_3 = 2$) than that of the three-node linear triangular element. For this problem, the right-most nodes of the triangle in Figure 4(a) are moved to the plane $x_3 = 1$. (b) SQP results of the optimization.

DOI: 10.1002/nme

5.2. *Rough surface example problem*

The proposed formulation is now tested with a large-scale problem consisting of two interpenetrating rough surfaces as shown in Figure 7. Even though this configuration is not physically feasible, interpenetrating thin layers are used to ensure that there is a number of slave nodes that have in their immediate neighborhood one or more master elements to which a distance computation can be performed. Figure 7 shows the discretization consisting of linear tetrahedra, but we also examine a quadratic discretization.

For the linear case, the performance of the proposed optimization-based methodology will be compared with more traditional approaches based on computational geometry. The first one uses proof by exhaustion, that is, *brute force*. The slave node is projected to the plane of a triangle on the master surface and if the projection does not lay within the triangle bounds, we then proceed to determine the closest point to each of the triangle edges. The second approach, taken from Ericson [52], uses Voronoi regions to find the closest point to a triangle. The algorithm quickly determines if the location of a slave node can be found to lay within the Voronoi region of one of the triangle vertices, thus assigning the corresponding vertex as the closest point. When vertex Voronoi regions do not contain the slave node, the algorithm then determines if the latter can be found in the regions that have projections to the edges. Finally, if all precedent cases are ruled out, the closest point is computed as laying within the face of the element. For these two computational geometry approaches, the distance computation for each slave is performed 10 times during the complete contact detection, and its average is reported.

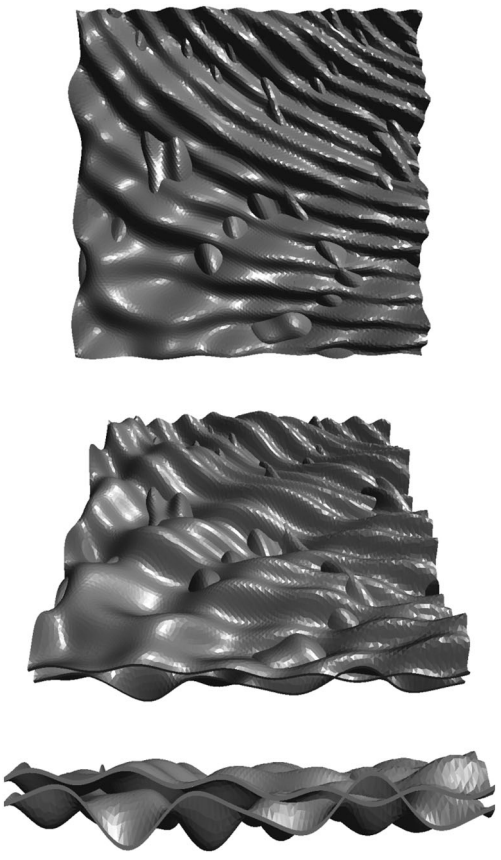


Figure 7. Three different rendered views of the finite element discretizations of the two interpenetrating rough surfaces used in the example of Section 5.2. The meshes comprise of a total of 149,162 linear tetrahedra and 48,426 nodes.

DOI: 10.1002/nme

The linear discretization of the thin layers that represent the two rough surfaces contain a total of 48,426 nodes and 149,162 linear tetrahedra. The seed used to create the meshes is such that a single tetrahedron is contained through the thickness. In order to avoid the $\mathcal{O}(n^2)$ time complexity associated with checking a slave node against every element on the master surface, a simple grid space decomposition is used to reduce the search region to those elements in the neighborhood of the slave node. Details of this technique, which is referred in the literature as a *bucket search*, can be found in [11, 34, 40]. The survey zone consists of $211 \times 211 \times 9$ buckets, with 20,589 and 115,499 non-empty node and element buckets, respectively. Coincidentally, there is a total of 20,589 slave nodes for which the closest distance is computed. For each slave node, master elements in the resulting neighboring buckets are obtained, resulting in an average of 7.8 master elements (with minimum and maximum of 1 and 37 elements, respectively). A total of 161,450 distance computations are performed for this mesh. The Voronoi region strategy turns out to be the most efficient for finding distances to linear triangles. The average distance computation time considering all slave nodes is $\bar{t} = 0.18\,\mu$s, with a standard deviation of $s = 0.03\,\mu$s. The brute force approach yields $\bar{t} = 0.7\,\mu$s and $s = 0.06\,\mu$s, with an increase in time by almost a factor of four. From the optimization techniques investigated in the previous section, only the SQP methodology is used here due to its robustness, resulting in an average computational time of $\bar{t} = 10.8\,\mu$s and $s = 4.5\,\mu$s. Overall, it took an average of 4.6 iterations per slave node. The optimization-based method takes 62 times more time than that of the Voronoi regions. Therefore, for linear elements, this latter computational geometry approach clearly outperforms any other method. Nevertheless, the proposed methodology is general and can be used regardless of the polynomial interpolation of the mesh, whereas the brute force and Voronoi region approaches are applicable only to linear meshes.

The discretization that uses quadratic elements contains a total of 195,035 nodes, four times the number of nodes of the linear mesh. The bucket array structure now contains $209 \times 209 \times 9$ buckets, with 76,052 and 135,370 non-empty node and element buckets, respectively. With a total of 628,017 distance computations, the proposed technique results in an average computational time $\bar{t} = 12.3\,\mu$s and a standard deviation $s = 6\,\mu$s, which are only slightly larger than the values for the linear mesh. There is an average of 8.26 master elements per slave node, with bounds [1, 38], resulting in an average of five iterations per distance computation.

6. CONCLUSIONS

An optimization-based formulation has been presented for local contact detection. The procedure does not rely on orthogonal projections so it can be used correctly for non-smooth surface geometries, common in FE analysis. It has been shown that the formulation takes a single iteration to converge when slave nodes are located within the Voronoi region of a point of the master element. The formulation is general in the sense that it does not change with increasing polynomial order, and the constraints remain always linear. Several techniques from the field of mathematical optimization were investigated for the given formulation, and SQP was the methodology of choice in Section 5.2 due to its robustness. The initial guess was also investigated, showing that it is worth starting from element vertices than from its center. Even though general and robust, the main drawback of the proposed methodology is its computational time when comparing it to other computational geometry strategies for linear elements. This was demonstrated in the example of two interpenetrating rough surfaces, where bucket sort was used to decompose the space and reduce the local search to elements intersecting with the slave's neighboring element buckets. Numerical results indicated that the proposed technique takes excessive computational time for linear meshes, resulting in roughly sixty times the time required for a closest-node computation based on Voronoi regions. Nevertheless, the approaches based on computational geometry cannot be used for quadratic meshes, for which the optimization-based distance computation did not increase significantly in cost. The proposed constrained-optimization method thus appears robust and efficient for meshes where the polynomial interpolation is higher than linear.

DOI: 10.1002/nme

ACKNOWLEDGEMENTS

This research is supported by the European Research Council ERCstg UFO-240332.

REFERENCES

1. Kikuchi N, Oden JT. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*. SIAM: Philadelphia, 1988.
2. Wriggers P. *Computational Contact Mechanics*, second edition. Springer-Verlag: Berlin, Germany, 2006.
3. Schweizerhof K, Nilsson L, Hallquist J. Crashworthiness analysis in the automotive industry. *International Journal of Computer Applications in Technology, Special Issue on the Industrial Use of Finite-element Analysis* 1992; **5**(2/3/4):134–156.
4. Avery P, Farhat C. The FETI family of domain decomposition methods for inequality-constrained quadratic programming: application to contact problems with conforming and nonconforming interfaces. *Computer Methods in Applied Mechanics and Engineering* May 2009; **198**(21-26):1673–1683.
5. Nackenhorst U. The ALE-formulation of bodies in rolling contact: theoretical foundations and finite element approach. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**(39-41):4299–4322.
6. Hu G, Wriggers P. On the adaptive finite element method of steady-state rolling contact for hyperelasticity in finite deformations. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(13-14):1333–1348.
7. Choi JH, Lee I. Finite element analysis of transient thermoelastic behaviors in disk brakes. *Wear* 2004; **257**(1-2):47–58.
8. Leblanc A, Nelias D, Defaye C. Nonlinear dynamic analysis of cylindrical roller bearing with flexible rings. *Journal of Sound and Vibration* 2009; **325**(1-2):145–160.
9. Doltsinis IS. Aspects of modelling and computation in the analysis of metal forming. *Engineering Computations* 1990; **7**(1):2–20.
10. Stupkiewicz S, Mróz Z. A model of third body abrasive friction and wear in hot metal forming. *Wear* 1999; **231**(1):124–138.
11. Benson DJ, Hallquist JO. A single surface contact algorithm for the post-buckling analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering* 1990; **78**(2):141–163.
12. Snozzi L, Molinari JF. A cohesive element model for mixed mode loading with frictional contact capability. *International Journal for Numerical Methods in Engineering* 2012; **93**(5):510–526.
13. Camacho G, Ortiz M. Adaptive lagrangian modelling of ballistic penetration of metallic targets. *Computer Methods in Applied Mechanics and Engineering* 1997; **142**(3–4):269–301.
14. Molinari J, Ortiz M. A study of solid-particle erosion of metallic targets. *International Journal of Impact Engineering* 2002; **27**(4):347–358.
15. Aagaard BT. Finite-element simulations of earthquakes. *PhD Thesis*, California Institute of Technology, 2000.
16. Lapusta N, Rice JR, Ben-Zion Y, Zheng G. Elastodynamic analysis for slow tectonic loading with spontaneous rupture episodes on faults with rate- and state-dependent friction. *Journal of Geophysical Research* 2000; **105**(B10):23 765–23 789.
17. Kammer DS, Yastrebov VA, Spijker P, Molinari JF. On the propagation of slip fronts at frictional interfaces. *Tribology Letters* 2012; **48**:27–32.
18. Heegaard J, Leyvraz PF, Curnier A, Rakotomanana L, Huiskes R. The biomechanics of the human patella during passive knee flexion. *Journal of Biomechanics* 1995; **28**(11):1265–1279.
19. Duvaut G, Lions JL. *Inequalities in Mechanics and Physics*, Grundlehren der mathematischen Wissenschaften. Springer-Verlag: Berlin, Germany, 1976.
20. Laursen TA. *Computational Contact and Impact Mechanics: Fundamentals of Modeling Interfacial Phenomena in Nonlinear Finite Element Analysis*. Springer-Verlag: Berlin, Germany, 2002.
21. Zhong ZH. *Finite Element Procedures for Contact-Impact Problems*. OUP Oxford: Oxford, United Kingdom, 1993.
22. Taylor RL, Papadopoulos O. On a patch test for contact problems in two dimensions. In *Nonlinear Computational Mechanics*, Wriggers P, Wagner W (eds). Springer: Berlin, Germany, 1991; 690–702.
23. Zavarise G, Wriggers P. A segment-to-segment contact strategy. *Mathematical and Computer Modelling* 1998; **28**:497–515.
24. Yang B, Laursen TA. A large deformation mortar formulation of self contact with finite sliding. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(6 - 8):756–772.
25. Hartmann S, Oliver J, Cante JC, Weyler R, Hernández JA. A contact domain method for large deformation frictional contact problems. part 2: numerical aspects. *Computer Methods in Applied Mechanics and Engineering* 2009; **198**:2607–2631.
26. Oliver J, Hartmann S, Cante JC, Weyler R, Hernández JA. A contact domain method for large deformation frictional contact problems. part 1: theoretical basis. *Computer Methods in Applied Mechanics and Engineering* 2009; **198**:2591–2606.
27. Alart P, Curnier A. A mixed formulation for frictional contact problems prone to Newton like solution methods. *Computer Methods in Applied Mechanics and Engineering* 1991; **92**(3):353–375.
28. Simo J, Laursen T. An augmented lagrangian treatment of contact problems involving friction. *Computers & Structures* 1992; **42**(1):97–116.

DOI: 10.1002/nme

29. Pietrzak G, Curnier A. Large deformation frictional contact mechanics: continuum formulation and augmented lagrangian treatment. *Computer Methods in Applied Mechanics and Engineering* 1999; **177**(3–4):351–381.
30. Heinrich B, Nicaise S. The Nitsche mortar finite-element method for transmission problems with singularities. *IMA Journal of Numerical Analysis* 2003; **23**(2):331–358.
31. Belgacem F, Hild P, Laborde P. The mortar finite element method for contact problems. *Mathematical and Computer Modelling* 1998; **28**(4–8):263–271.
32. McDevitt TW, Laursen TA. A mortar-finite element formulation for frictional contact problems. *International Journal for Numerical Methods in Engineering* 2000; **48**(10):1525–1547.
33. Yang B, Laursen TA, Meng X. Two dimensional mortar contact methods for large deformation frictional sliding. *International Journal for Numerical Methods in Engineering* 2005; **62**(9):1183–1225.
34. Heinstein MW, Mello FJ, Attaway SW, Laursen TA. Contact–impact modeling in explicit transient dynamics. *Computer Methods in Applied Mechanics and Engineering* 2000; **187**(3–4):621–640.
35. Yastrebov VA. *Numerical Methods in Contact Mechanics*. ISTE/Wiley: New York City, NY, USA, 2013.
36. Kane C, Repetto E, Ortiz M, Marsden J. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering* 1999; **180**(1-2):1–26.
37. Eberly DH. *Game Physics*. Elsevier Science Inc.: New York, NY, USA, 2003.
38. Eberly D. *3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic*, The Morgan Kaufmann Series in Interactive 3D Technology. Elsevier Science: Amsterdam, the Netherlands, 2004.
39. Newman S. *Principles of Interactive Computer Graphics*, 2nd edn. Mc Graw-Hill: Singapore, 1979.
40. Fujun W, Jiangang C, Zhenhan Y. A contact searching algorithm for contact-impact problems. *Acta Mechanica Sinica* 2000; **16**:374–382.
41. Malone JG, Johnson NL. A parallel finite element contact/impact algorithm for non-linear explicit transient analysis: Part I–the search algorithm and contact mechanics. *International Journal for Numerical Methods in Engineering* 1994; **37**(4):559–590.
42. Yang B, Laursen TA. A contact searching algorithm including bounding volume trees applied to finite sliding mortar formulations. *Computational Mechanics* 2008; **41**:189–205.
43. Brown K, Attaway S, Plimpton S, Hendrickson B. Parallel strategies for crash and impact simulations. *Computer Methods in Applied Mechanics and Engineering* 2000; **184**(2-4):375–390.
44. Malone JG, Johnson NL. A parallel finite element contact/impact algorithm for non-linear explicit transient analysis: Part II–parallel implementation. *International Journal for Numerical Methods in Engineering* 1994; **37**(4):591–603.
45. Plimpton S, Attaway S, Hendrickson B, Swegle J, Vaughan C, Gardner D. Parallel transient dynamics simulations: algorithms for contact detection and smoothed particle hydrodynamics. *Journal of Parallel and Distributed Computing* 1998; **50**(1-2):104–122.
46. Berger M, Bokhari S. A partitioning strategy for nonuniform problems on multiprocessors. *Computers, IEEE Transactions on* 1987may; **C-36**(5):570–580.
47. Attaway SW, Hendrickson BA, Plimpton SJ, Gardner DR, Vaughan CT, Brown KH, Heinstein MW. A parallel contact detection algorithm for transient solid dynamics simulations using PRONTO3D. *Computational Mechanics* 1998; **22**:143–159.
48. Yastrebov VA, Cailletaud G, Feyel F. A local contact detection technique for very large contact and self-contact problems: sequential and parallel implementations. In *Trends in Computational Contact Mechanics*, Vol. 58, Zavarise G, Wriggers P (eds), Lecture Notes in Applied and Computational Mechanics. Springer: Berlin, Germany, 2011; 227–251.
49. Karush W. Minima of functions of several variables with inequalities as side constraints. *Master's Thesis*, Dept. of Mathematics, Univ. of Chicago, Illinois, 1939.
50. Kuhn HW, Tucker AW. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. University of California Press: Berkeley and Los Angeles, 1951; 481–492.
51. Nocedal J, Wright SJ. *Numerical Optimization*, 2nd edn., Springer series in operations research. Springer: Berlin, Germany, 2006.
52. Ericson C. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2004.

DOI: 10.1002/nme