Solving Contact Problems with Abaqus

Asim Rashid

Preface

The finite element method (FEM) is widely used for solving the contact problems in solid mechanics. Numerical solution techniques for contact problems have evolved significantly over the decades and now have matured to solve very complex engineering problems.

Many books are available on the topic of contact mechanics giving an insight into the basic principles of nonlinear continuum mechanics and their application to the contact problems. These books give a good insight into the basic theory, however these are not sufficient for a CAE Engineer/Analyst to solve the contact problems in a FEA software. This book aims to provide practical information to perform complex contact analysis in Abaqus, a leading FEA software. The book mainly consists of tutorials providing intensive instructions to perform analysis of contact problems. During such analysis it is very common to face convergence difficulties. Quite a few tutorials are devoted to diagnose such difficulties and take the corrective action.

The book begins with an introduction. In this section, an overview of contact capabilities and explanation of different formulations is given.

Contents

Table of Contents

Introduction

Contact is the principal way load is transferred from one body to another one. Almost all mechanical systems, e.g. brakes, bearings, clutches, combustion engines, tires and gear trains, involve contacting elements/components. Understanding the contact behavior of mechanical systems provides necessary information for the comfortable, reliable and energy efficient design. The study of stresses and deformations arising due to contact interaction of solid bodies is thus of paramount importance in many engineering application.

Contact problems are considered to be highly nonlinear and are one of the most difficult ones to solve. Many commercial FEA software offer the capabilities to solve contact problems. These capabilities are generally considered immature in comparison to the capabilities in the general area of nonlinear solid mechanics. This has motivated the analysts to assume simplified boundary conditions or make some other assumptions instead of solving a contact problem. In the previous decade there has been significant improvements in the algorithmic robustness. Furthermore remarkable increase in the computer power has made it practical to solve complex contact problems efficiently.

The general aim of performing a contact analysis is to determine contact traction and area on the surfaces where this traction acts. In a frictionless contact, a force normal to the contacting surfaces acts on the contacting bodies. If friction is present between the surfaces, shear forces may be generated that resist the tangential motion of the contacting bodies. The solution to a contact problem requires special constraints, known as Signorini conditions, to be considered. These constraints make sure that the penetration of contacting bodies does not take place and contact traction only becomes non-zero when contact is established.

Abaqus provides extensive capabilities to simulate contact problems accurately and efficiently. In Abaqus, contact simulations can be either surface based or contact-element based. In this work, only surface-based contact is discussed. In Abaqus, contact between two surfaces/bodies is defined by creating a contact interaction and specification of contact properties between contacting surfaces.

Contact interaction

A contact interaction contains all the information necessary to specify the interacting behavior of contacting surfaces. When a contact interaction is created, it is important to specify the step in which to activate the interaction, the type of interaction, and the region of the model to which the interaction will be applied.

Two primary algorithms available when using surface-based contact are: contact pair algorithm and general contact algorithm.

The general contact algorithm is highly automated and is based on an automatically generated all-inclusive surface definition. On the contrary, the contact pair algorithm requires to explicitly select surfaces that may potentially come into contact.

Contact property model

A contact property model contains a set of data, e.g. coefficient of friction and thermal conductance, that is referred by a contact interaction. Each contact interaction must refer to a contact interaction property that governs the interaction behavior.

In a mechanical contact simulations, often, a contact interaction property is defined to specify non-default tangential behavior (friction) or normal behavior (hard or soft contact). Most commonly used behaviors will be described here.

Hard Contact

The "hard" contact relationship is the default contact behavior in Abaqus. It implies that the contact pressure is applied on the contacting surfaces when the clearance between two surfaces becomes zero. When surfaces are in contact, any contact pressure can be transmitted between them. The surfaces separate if the contact pressure reduces to zero. The "hard" contact relationship is shown in Figure 1.

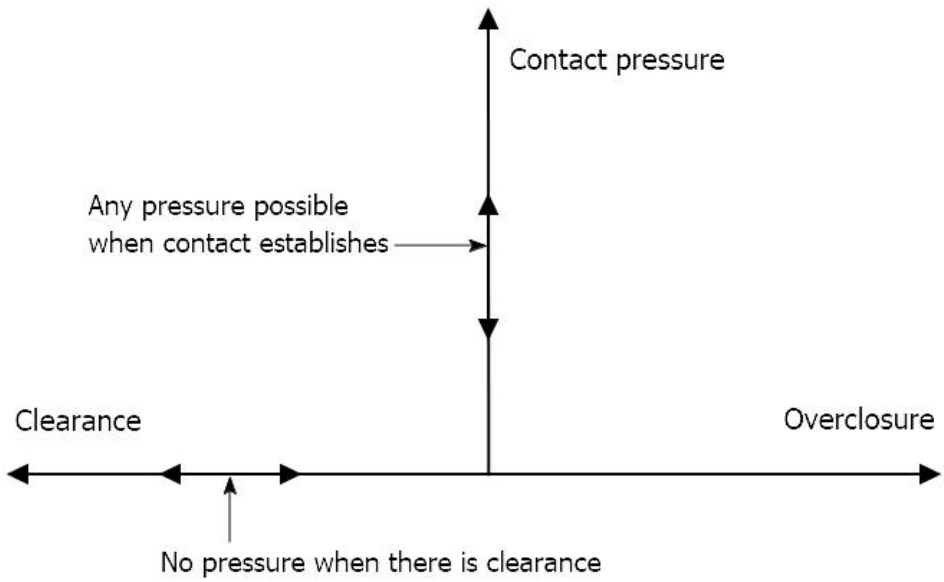

Figure 1: Hard contact relationship

The above figure implies zero-penetration of the contacting bodies, however, the zero-penetration condition may or may not be strictly enforced depending on the constraint enforcement method used.

Soft Contact

The "soft" contact is usually used to model a soft, thin layer on surfaces. In Abaqus/Standard they are also sometimes used because they can make it easier to resolve the contact difficulties. Three types of softened contact relationships are available in Abaqus: exponential, linear and tabular. Here we will discuss tabular law briefly.

Tabular law is used to define a pressure-overclosure relationship in tabular form in which the contact pressure is a piecewise linear function of the overclosure between the surfaces. To define a piecewise-linear relationship in tabular form, data pairs (p_i, h_i) of pressure versus overclosure (clearance corresponds to negative overclosure) are specified as shown in Figure 2.

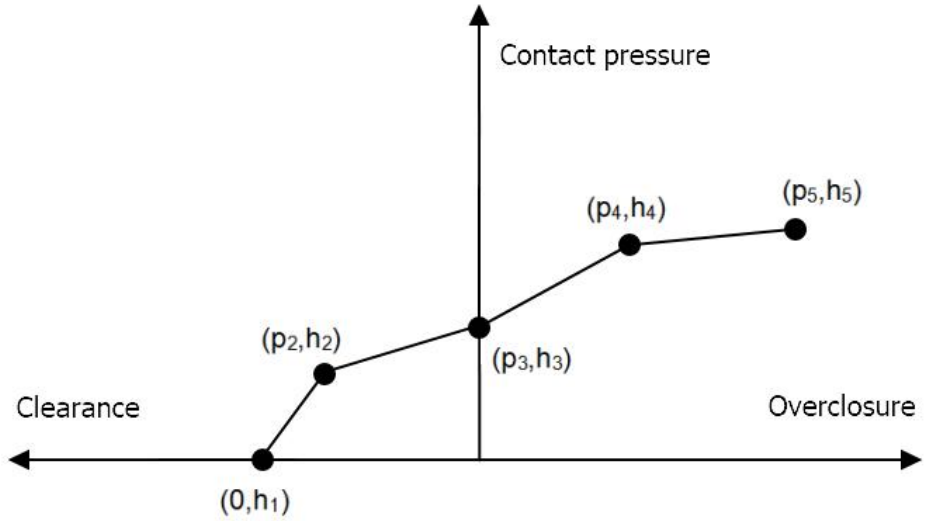

Figure 2: Tabular pressure-overclosure relationship

In this relationship the surfaces transmit contact pressure when the overclosure between them, measured in the contact (normal) direction, is greater than h_1, where h_1 is the overclosure at zero pressure. So the surfaces can transmit pressure even there is a clearance between them.

Friction models

By default, frictionless contact is enforced in tangential direction. Including the ideal Coulomb friction model in a contact property, could make the convergence very difficult. Therefor Abaqus provides a penalty friction formulation which allows some relative motion of the surfaces (called “elastic slip”) when they should be sticking. The penalty friction formulation works well for most of the analysis.

In Abaqus/Standard the the ideal Coulomb friction model between two surfaces can be enforced exactly by using the Lagrange multiplier formulation. With this method there is no relative motion between two closed surfaces. However, the Lagrange multipliers converges more slowly and require more computational resources.

Friction should be included in the a contact property model only when it has a significant effect on the simulation results.

Contact constraint enforcement methods

Several methods are available in literature to enforce the contact constraints, e.g. the Lagrange multiplier method, the penalty method, the augmented Lagrangian multiplier method and the Nitsche method. First three will be described in a very simplified way here.

We consider an elastic body with outward pointing unit normal vector **n**. The body comes into frictionless contact with a rigid obstacle as shown in Figure 3. This gives rise to contact pressure on the contacting surfaces.

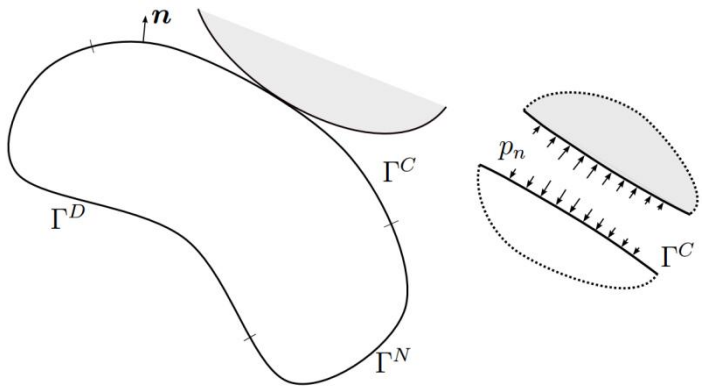

Figure 3: A deformable body in frictionless contact with a rigid obstacle

In the absence of the contact, response of the body can be computed by the following relation

$$\mathbf{Ku} = \mathbf{F} \qquad (1)$$

where **K** is the stiffness matrix, **u** is the displacement vector, and **F** is the external force vector.

When the body comes into contact, an additional term is added to the equilibrium equation to account for the contact pressure developed at the contact interface.

In the Lagrange multiplier method, the equilibrium equation is solved simultaneously with another equation which takes care of the contact constraints. So the following system of equations needs to be solved for **u** and $\mathbf{P_n}$ in the Lagrange multiplier method

$$\mathbf{Ku} + \mathbf{C_n}^T\mathbf{P_n} = \mathbf{F} \qquad (2)$$
$$\mathbf{C_n u} = \mathbf{g} \qquad (3)$$

where $\mathbf{C_n}$ is a transformation matrix containing the normal directions, $\mathbf{P_n}$ is the vector containing the Lagrange multipliers (contact pressure) and **g** vector contains all the initial gaps along **n**.

In the penalty method, penetration of the contact surface is penalized. In the beginning there is no contact pressure acting on the contacting surfaces so some nodes will penetrate the obstacle. As contact conditions are enforced, a contact pressure is generated which is proportional to the penetration of the contact surface i.e. $\mathbf{P_n} = \text{ϒ}[\mathbf{C_n u} - \mathbf{g}]_+$, where ϒ is a given penalty parameter and $[x]_+ = (x + |x|)/2$. So the larger the penetration, the greater the calculated contact pressure. With this method some degree of penetration will usually occur. The penalty method leads to a problem to be solved only in one variable, **u**, using the following equation

$$\mathbf{Ku} + \mathbf{C_n}^T\mathbf{P_n} = \mathbf{F} \qquad (4)$$

The augmented Lagrangian approach combines the Penalty and the Lagrange multiplier methods. The contact pressure is determined by $\mathbf{P_n} = [\mathbf{P_n} + \gamma(\mathbf{C_n u - g})]_+$ and the equation (4) is solved for **u**.

In the augmented Lagrangian method contact pressure is augmented in the computations which makes it less sensitive to the penalty parameter as compared to the Penalty method. It also improves the accuracy of contact pressure computations.

All of these contact enforcement methods are available in Abaqus. The terminology used in Abaqus is somewhat different. Now we will describe the enforcement methods as they are referred in Abaqus.

There are three contact constraint enforcement methods available in Abaqus/Standard: the direct method, the penalty method and the augmented Lagrange method.

1. The direct method attempts to strictly enforce a given pressure-overclosure behavior, without approximation or use of augmentation iterations. Because of its strict interpretation of contact constraints, hard contact simulations utilizing the direct enforcement method are susceptible to overconstraint issues. Lagrange multipliers are always used when direct method is specified for hard contact. When direct method is used with softened contact behavior, the constraints are enforced with or without Lagrange multipliers depending on the pressure-overclosure curve slope.

2. The penalty method is a stiff approximation of hard contact as it approximates hard pressure-overclosure behavior. Numerical softening associated with the penalty method can mitigate overconstraint issues and reduce the number of iterations required in an analysis.

3. The augmented Lagrange method uses the same kind of stiff approximation as the penalty method.

The penalty and augmented Lagrange methods sometimes provide more efficient solutions at some sacrifice in solution accuracy.

Abaqus/Explicit uses two different methods to enforce contact constraints: the kinematic contact method and the penalty method.

1. The kinematic contact algorithm uses a kinematic predictor/corrector contact algorithm to strictly enforce contact constraints.

2. The penalty contact algorithm has a weaker enforcement of contact constraints but allows for treatment of more general types of contact.

For general contact Abaqus/Explicit enforces contact constraints using the penalty contact method. The contact pair algorithm uses the kinematic contact formulation by default.

Relative sliding of surfaces

Abaqus makes a distinction between analyses where the magnitude of relative sliding is small and those where the magnitude of sliding may be finite. It requires less computational resources to model problems where the sliding between the surfaces is small. Small sliding means that a point contacting a surface does not slide more than a small fraction of a typical element dimension.

When using the small-sliding formulation, Abaqus determines at the beginning of the simulation which segment on the master surface will interact with each node on the slave surface. It maintains these

relationships throughout the analysis, never changing which master surface segments interact with which slave nodes. In contrast, when the finite-sliding contact formulation is used, Abaqus continually tracks which part of the master surface is in contact with each slave node. This is a very complex calculation and requires considerable computational resources. If you are not sure which sliding option is suitable for a given problem, it is always safe to specify finite sliding.

Slave and master surfaces

In a contact interaction, specification of master and slave roles is important as it could have significant impact on the results. It can also have an effect on performance and solution could become expensive .

Contact pairs in Abaqus/Standard use a pure master-slave contact algorithm. Therefore, it is important to select the slave and master surfaces correctly in order to achieve the best possible results. Some simple rules to follow are:

1. Analytical rigid surfaces and rigid-element-based surfaces must always be the master surface.
2. Among the two surfaces attached to deformable bodies, it is best to choose the smaller surface as the slave surface. If that distinction cannot be made, the master surface should be chosen as the surface of the stiffer body or as the surface with the coarser mesh if the two surfaces are on structures with comparable stiffness. If the stiffness and mesh density are the same on both surfaces, the preferred choice is not always obvious. (The stiffness of the structure and not just the material should be considered when choosing the master and slave surface. For example, a steel component may be less stiff than a large component of aluminum even though the steel has a higher modulus of elasticity than the aluminum.)

The choice of master and slave roles is particularly important for node-to-surface contact.

The general contact in Abaqus/Standard automatically assigns master and slave roles for contact interactions. The general contact algorithm in Abaqus/Explicit uses balanced master-slave weighting whenever possible (pure master-slave weighting is used for general contact interactions involving node-based surfaces). For balanced master-slave contact, Abaqus/Explicit calculates the contact constraints twice: once with the first surface acting as the master surface and once with the second surface acting as the master surface. The weighted average of the two corrections (or forces) is applied to the contact interaction. For the contact pair algorithm, Abaqus/Explicit will decide which type of weighting to use for a given contact pair based on the nature of the two surfaces involved and the constraint enforcement method used.

Discretization of contact pair surfaces

There are different possibilities to formulate the contact constraints. Similarly contact constraints can be applied at various locations on interacting surfaces. The conditions and location of these constraints depend upon the contact discretization applied to the interacting surfaces. There are different discretization available in literature, e.g. node-to-node, node-to-surface and surface-to-surface discretization.

Abaqus/Standard offers two contact discretization options: “node-to-surface” discretization and “surface-to-surface” discretization.

Node-to-surface contact discretization

With node-to-surface discretization, each node on the slave surface effectively interacts with a point of projection on the master surface on the opposite side of the contact interface. The slave nodes are constrained not to penetrate into the master surface. Figure 4 shows a finely meshed slave surface contacting a coarsely meshed master surface.

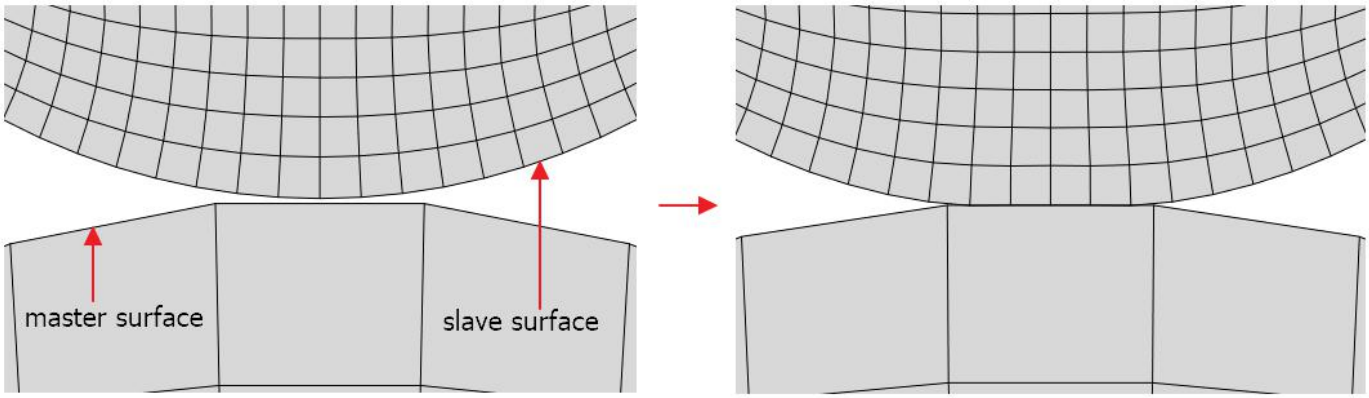

Figure 4: Node-to-surface discretization with a finely meshed slave surface

The algorithm places no restriction on the master surface; it can penetrate the slave surface. Figure 5 shows a finely meshed master surface contacting a coarsely meshed slave surface. It can be seen that master surface has penetrated the coarsely meshed slave surface.

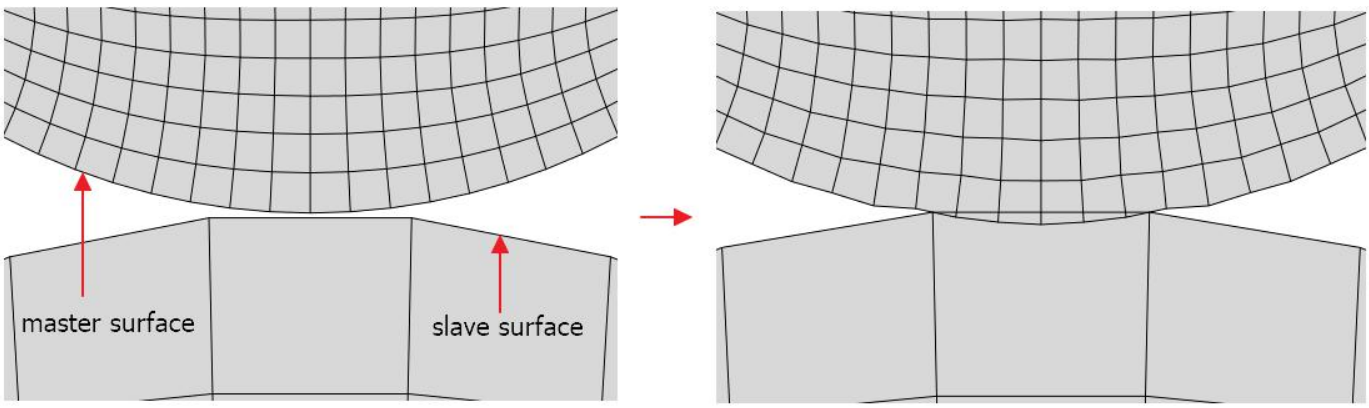

Figure 5: Node-to-surface discretization with a finely meshed master surface

The only information needed for the slave surface is the location and surface area associated with each node. Thus a a node-based surface (a surface consisting of a group of nodes) can be used as the slave surface. Node-to-surface discretization is available even if a node-based surface is not used in a contact pair definition.

Surface-to-surface contact discretization

The surface-to-surface formulation enforces contact conditions in an average sense over regions nearby slave nodes rather than only at individual slave nodes. The averaging regions are approximately centered on slave nodes, so each contact constraint will predominantly consider one slave node but will also consider adjacent slave nodes. Some penetration may occur at individual nodes; however, large penetrations of master nodes into the slave surface do not happen.

Figure 6 shows a finely meshed slave surface contacting a coarsely meshed master surface when surface-to-surface discretization is used.

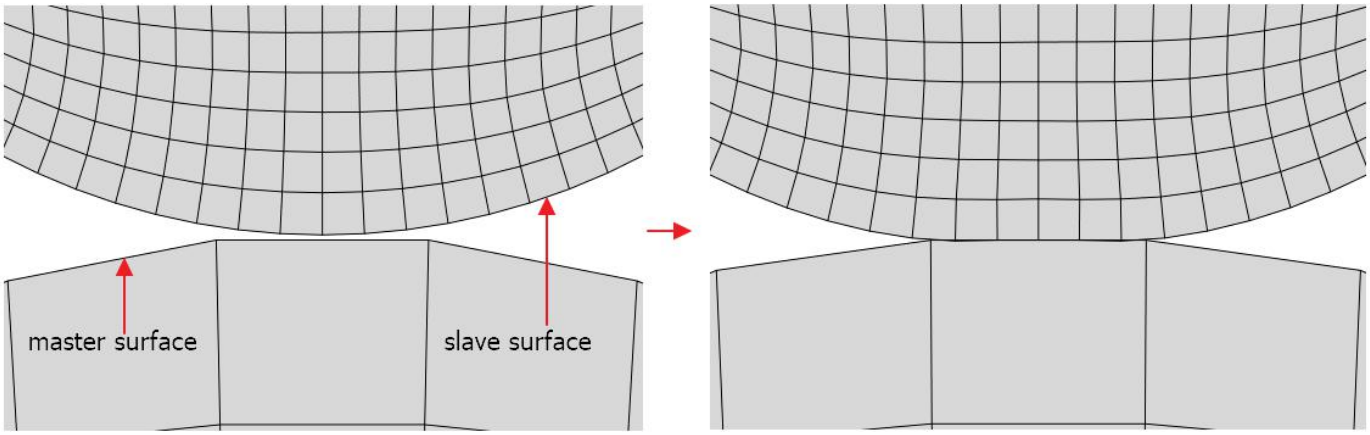

Figure 6: Surface-to-surface discretization with a finely meshed slave surface

Figure 7 shows a finely meshed master surface contacting a coarsely meshed slave surface when surface-to-surface discretization is used. It can be seen that only a small penetration has occurred as compared to the node-to-surface discretization.

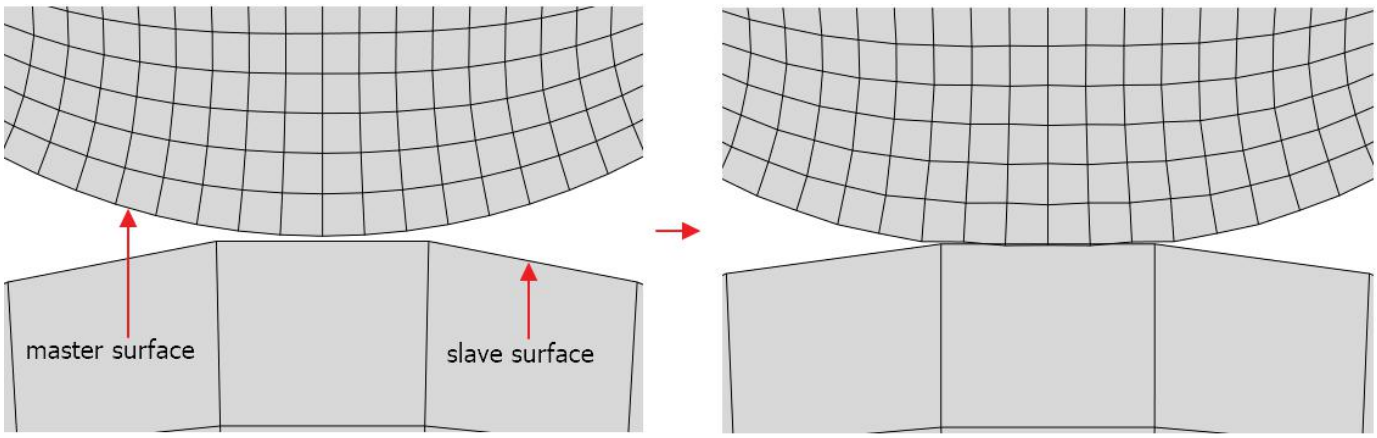

Figure 7: Surface-to-surface discretization with a finely meshed master surface

In general, surface-to-surface discretization provides more accurate stress and pressure results than node-to-surface discretization. As the mesh is refined, the difference in simulation results between the discretizations reduces, but for a given mesh refinement the surface-to-surface approach tends to provide more accurate stresses. Contact using surface-to-surface discretization is also less sensitive to master and slave surface designations than node-to-surface contact.

The primary mechanism for enforcing contact in Abaqus/Explicit is node-to-face contact.

Defining Surfaces

An important aspect of defining contact interactions is to specify surfaces on the bodies that may potentially come into contact. The following type of surfaces can be defied in Abaqus.

- Element-based surface
- Node-based surfaces
- Analytical rigid surfaces

Element-based surface

Element-based surfaces are defined on the faces or edges of elements. If the surface is defined on rigid elements, it is called a rigid surface and if it is defined on deformable elements, it is called a deformable surface. In Abaqus/CAE, when a surface is defined by picking geometry on a part instance, Abaqus creates the element-based surface internally by grouping the faces of underlying elements. This is illustrated schematically in the Figure 8.

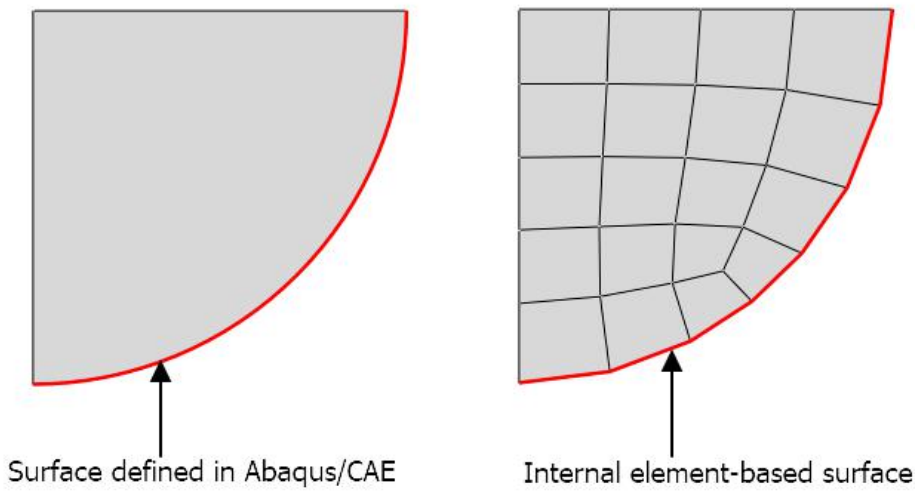

Figure 8: Definition of an element-based surface

Node-based surfaces

Node-based surfaces are defined by specifying the nodes or node sets. A node-based surface can only be used as a slave surface.

Element-based surfaces contain more intrinsic information, e.g. surface area, than node-based surfaces. If an element-based surface is used in a contact analysis, Abaqus can calculate the contact stress accurately as it can associate surface area with each node.

Analytical Rigid Surfaces

Analytical rigid surfaces are geometric surfaces with profiles that can be described with straight and curved line segments. These profiles can be extruded or rotated about an axis to form a three-dimensional surface.

Analytical rigid surfaces as compared to rigid surfaces formed by element faces result in a smoother surface description, which can reduce contact noise, and also results in decreased computational cost for a contact analysis. However the range of shapes that can be created with an analytical rigid surface are limited.

Solution Algorithm

Before describing the solution algorithm for contact problems, it is important to describe the solution procedure used for solving nonlinear problems.

In a nonlinear problem the structure's stiffness changes as it deforms. A typical nonlinear load-displacement curve for a structure is shown in Figure 9. The objective of a nonlinear analysis is to determine this response. This requires to apply the load in a series of small increments so that the nonlinear curve could be followed. At the end of an increment, internal and external forces are usually not in balance and hence further iterations are required to find an equilibrium state. A number of incremental-iterative solution procedures are available to determine the nonlinear response. Most popular among those are full Newton (also called Newton-Raphson) and quasi-Newton techniques. First we will describe the full Newton technique.

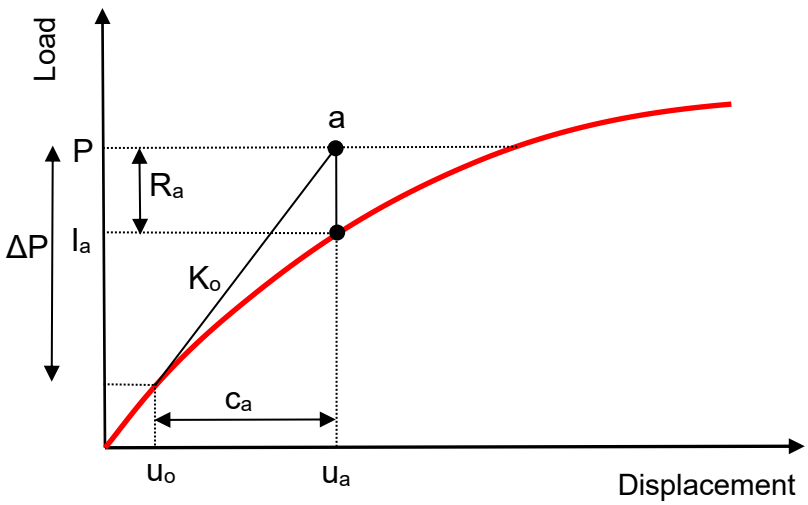

Figure 9: First iteration in an increment

We consider the structure at a configuration u_o with an initial stiffness K_o. The structure is subjected to an external load increment ΔP. The analysis of the structure requires to determine a configuration which produces an internal force vector balancing externally applied load. The response of the structure to the load increment ΔP can be determined by a linear approximation as $c_a = K_o^{-1} \Delta P$. The computed displacement correction, c_a, is then used to update the structure's configuration to $u_a = u_o + c_a$. In this updated configuration, internal forces (I_a) are also calculated. For the structure to be in equilibrium, internal and external forces have to be in balance. To determine if the the structure is in equilibrium, the force residual is calculated as $R_a = P - I_a$. If R_a is zero at every node in the model, point a would lie on the load-displacement curve and structure would be in equilibrium. For a nonlinear problem it is almost impossible that force residuals are zero, therefore force residuals are compared to a tolerance value. If force residual is less than this tolerance, u_a is accepted as the equilibrium configuration. If equilibrium is not established, a new iteration is performed, as shown in Figure 10, to bring the internal and external forces into balance. Updated stiffness matrix, K_a, which is computed at the u_a is used to compute new response by using $c_b = K_a^{-1} R_a$. The computed displacement correction, c_b, is then used to update the structure's configuration to $u_b = u_a + c_b$. To determine if the the structure is in equilibrium, the force residual is calculated as $R_b = P - I_b$ and compared to the tolerance value. If the force residual is less than the tolerance value, u_b is accepted as the equilibrium configuration otherwise further iterations are performed until the solution converges.

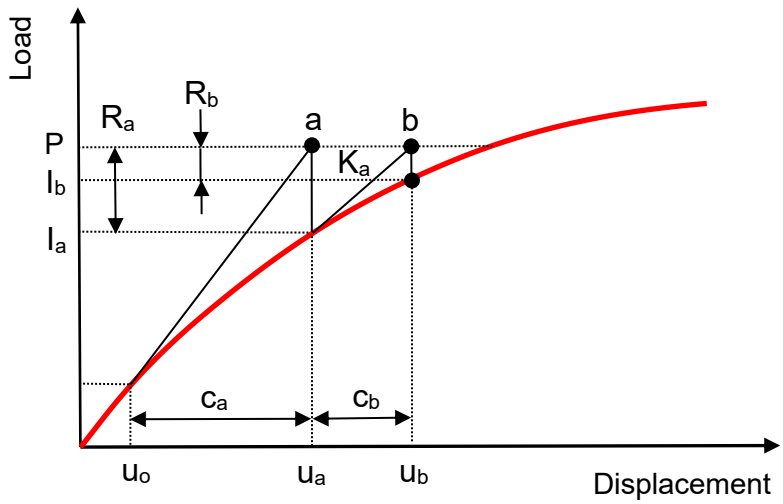

Figure 10: Second iteration

In the Newton-Raphson solution procedure, during each iteration structure's stiffness matrix is updated which results in very expensive analysis as compared to a linear analysis. Quasi-Newton methods can be cost-effective alternative for large nonlinear problems where stiffness matrix does not change significantly from one iteration to the next. In a quasi-Newton method, stiffness matrix is not updated at every iteration instead obtained by a secant approximation. The BFGS (Broyden, Fletcher, Goldfarb, Shanno) method is implemented in Abaqus/Standard as a quasi-Newton solution procedure.

Abaqus/Standard uses the Newton-Raphson algorithm by default to obtain solution for nonlinear problems. It can take several iterations to determine an acceptable equilibrium solution to a given load increment. With every iteration, the solution should be closer to equilibrium. However if the solution appears to move away from equilibrium and convergence is not expected, the current increment is aborted and a new attempt is made by reducing the increment size. When at the end of an iteration, an equilibrium solution is

obtained, the increment is complete and output results become available. This solution procedure is shown schematically in the Figure 11.

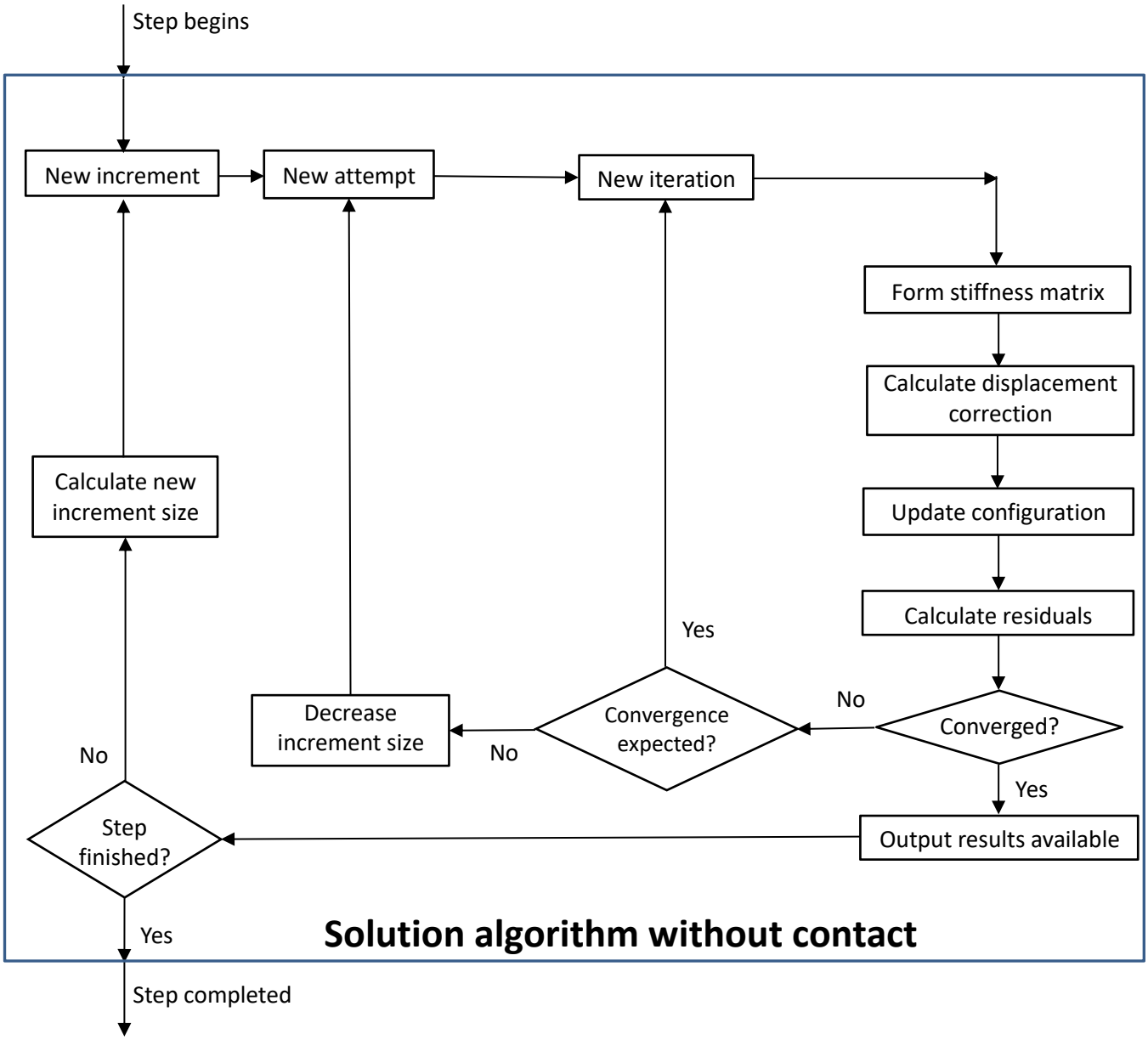

Figure 11: Solution algorithm for solving nonlinear problems in Abaqus/Standard

For contact problems, Abaqus/Standard checks the status of all contacting surfaces at the beginning of each increment to determine whether slave nodes are open or closed. Then an iteration is performed and updated configuration of the model is determined. In the updated configuration changes in contact conditions at slave nodes are checked. If the clearance becomes negative (overclosure) or zero at any node, its status is changed to closed. If the contact pressure becomes negative at any node, its status is changed to open. An iteration in which the contact state at the end of the iteration is different from the initially determined state is classified as severe discontinuity iteration (SDI). Figure 12 shows the different states of slave nodes when “hard” contact behavior is enforced.

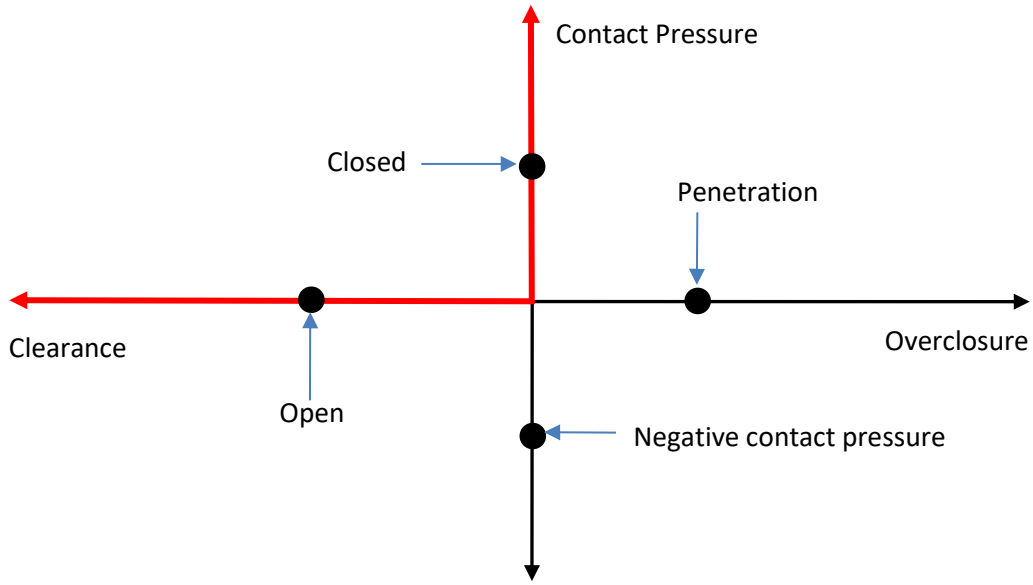

Figure 12: Different states of slave nodes in a hard contact relationship

By default, Abaqus/Standard continues to iterate until the severe discontinuities are sufficiently small or no severe discontinuities happen. So if the magnitude of the penetration and force errors are within the tolerances, the contact changes will not affect the solution convergence although the iteration will be labeled as SDI . The solution procedure for problems with contact interactions is shown schematically in the Figure 13.

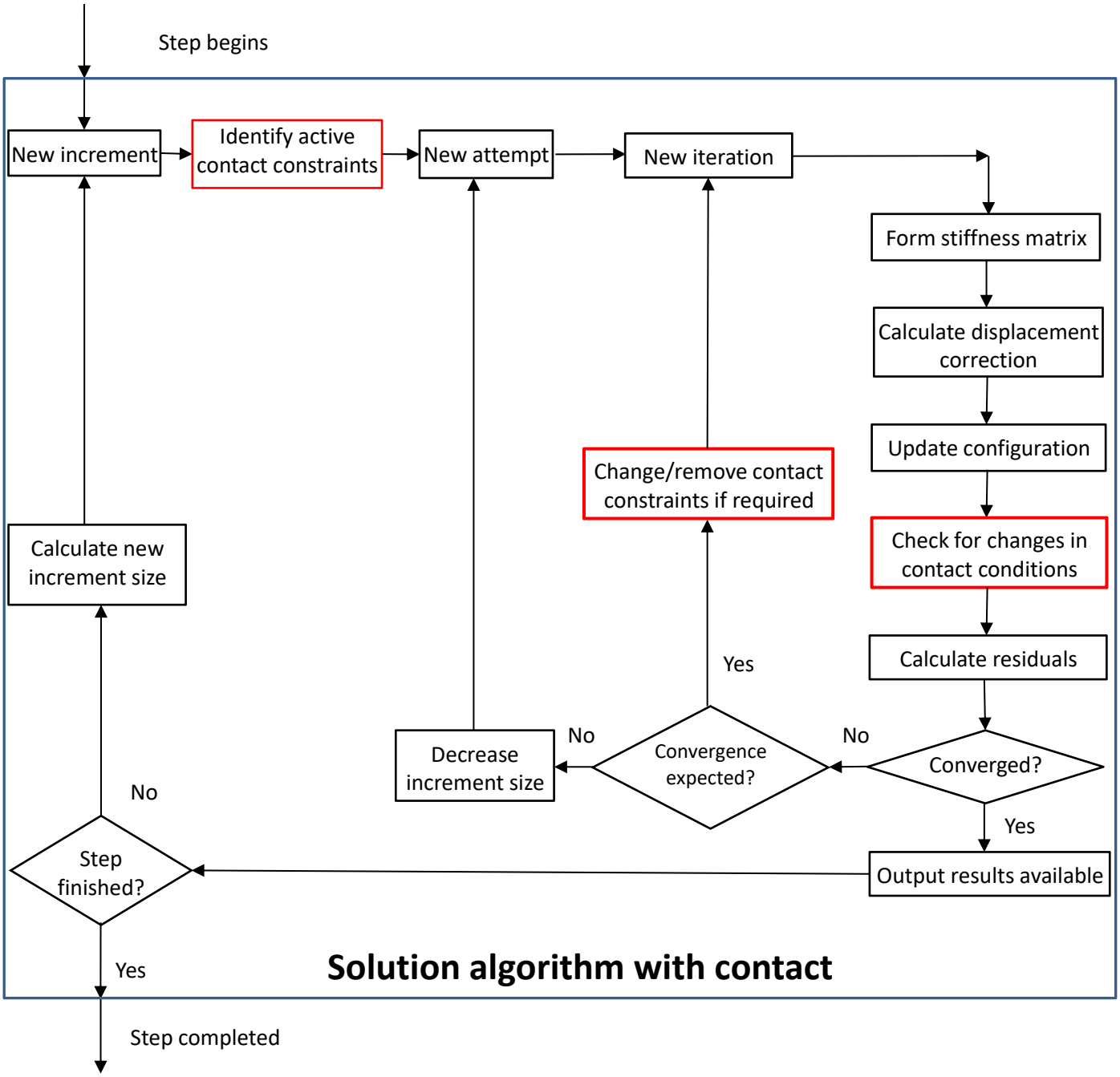

Figure 13: Solution algorithm for solving problems with contact in Abaqus/Standard

Abaqus/Standard distinguishes between an equilibrium iteration (an iteration in which no severe discontinuities occur) and a severe discontinuity iteration (an iteration in which contact changes occur). Job Diagnostics dialog box gives the summary for each increment providing the information how many iterations are severe discontinuity iterations and how many are equilibrium iterations.

Abaqus/Explicit determines a solution to a given problem without iterating by explicitly advancing the kinematic state from the previous increment. Abaqus/Explicit usually resolves complicated contact problems and other discontinuous nonlinearities more easily than Abaqus/Standard.