Chapter 6

2D needle insertion

Inserting needles into soft tissue is one of the most often performed medical procedures. In some procedures, needles have to be inserted deeply into soft tissue to reach a target. For example, certain types of biopsies (taking tissue samples from organs) are performed with needles with a specialized needle tip, designed to extract a tissue specimen. Another application is *brachytherapy*, treating cancer by inserting radioactive seeds directly in the tumor. The primary application of brachytherapy is prostate cancer. In this application the seeds are delivered with a needle [3]. In general, when a needle is inserted into soft material, such as tissue, the material and the needle interact through friction forces, resulting in deformation of the material. This deformation makes it harder to reach the desired target and avoid vital organs. Simulations of needle insertions that take into account this deformation may help train and plan for such difficult cases.

6.1 Related work

Simulations of needle insertion for training purposes have been implemented earlier using ad-hoc models, e.g. [15, 60], which rely solely on force measurements performed at the inserting end. The work by DiMaio and Salcudean [36–38] presents a breakthrough. They present a planar virtual environment for needle insertion, where the mechanics of the system are derived from complete 2D measurements of deformation. These measurements are performed by recording a standardized needle insertion in a flat square slab of soft material with a camera. The video sequence is used to measure the deformation of the slab. By comparing the measured deformations with a FEM simulation of the same material, the friction forces along the needle shaft are estimated. An interactive simulation is then built, where the same friction profile is used to model the interaction between tissue and needle. This simulation uses a static linear elasticity model on a fine 2D grid. During the simulation, only a small portion of nodes is 'visible', i.e. relevant to the system response: the nodes on the boundary are needed for visualization, and those that interact with the needle are necessary for computing interactions with the tissue. The responses of this subset can be can be condensed into a

small stiffness matrix, which is inverted and updated on the fly. As the needle advances into the tissue, more nodes are added to the condensed system. Updates of the inverted matrix take $\mathcal{O}(s^2)$ operations, where s is the number of visible nodes. Similarly, the effect of friction is to change the boundary conditions for nodes on the needle; these updates can also be done in $\mathcal{O}(s^2)$. The condensation process requires that the stiffness matrix corresponding to all the nodes of the mesh is inverted off-line.

Alterovitz et al. [2,3] also simulate needle insertion in FEM meshes. They use a linear elasticity model on a regular 2D grid, but use an explicit GN22 scheme to dynamically solve these equations. GN22 is similar to the SS22 scheme from Equation (2.60). The model runs at visual update rates (25 Hz) for a 1250 triangle mesh. They specifically target brachytherapy for prostate cancer treatment. By varying tissue and friction parameters in the model, the effect of tissue parameters and needle characteristics on seed placement accuracy can be predicted.

6.2 Needle insertion in 3D

The needle applies force to the material, and can be considered a part of the boundary for the boundary value problem. In a two-dimensional setting, the boundary is one-dimensional, and can be represented accurately by a small set of edges. In 3D, forces should be distributed over surfaces. In other words, for a physically valid discretization in 3D, the needle cannot be represented as a set of edges, but must be represented as a surface. To represent this surface, the mesh must include elements with a size comparable to the needle diameter.

If we assume that a needle has a diameter of 1 mm, then the mesh should have elements of that size in the vicinity of the inserted needle. It is clear that there will be huge disparity between element size and object size. Let us assume that the object is cube shaped, is 10 cm in length in every axis, and is discretized using a uniformly refined mesh. The mesh would have approximately $10\,\text{cm}/1\,\text{mm} = 100$ nodes for each side, leading to approximately 10^6 nodes and $3 \cdot 10^6$ degrees of freedom. The dimensions of the stiffness matrix are $3{\cdot}10^6 \times 3{\cdot}10^6$. Simulations of this size cannot be run interactively in the forseeable future. For example, condensation of internal nodes requires storing the inverse of the stiffness matrix. The inverse is dense, and storing a dense $3 \cdot 10^6$ DOF matrix requires approximately 67 terabytes of memory. Moreover, for nonlinear elasticity, the stiffness matrix depends on the deformations. Since the stiffness matrix is not constant, precomputing matrix inverses does not make sense.

For large or nonlinear problems, iterative methods have to be used, and these do not require precomputed structures. If we do not rely on precomputed structures, then there is no need to start with a fine mesh, but we can refine the mesh where necessary: this is where the needle is inserted. In this chapter, we will use these observations to produce a 2D simulation that is functionally equivalent to the one shown by DiMaio and Salcudean, but uses an iterative solution method and adaptive mesh resolution. With this method, it is possible to exchange computation time and accuracy. To asses whether this approach can be called interactive, we assess how quickly our implementation runs for reasonable accuracy requirements. Before we give these results, we show how the problem is modeled, and how the simulation is implemented.

6.3 Physical model

The physical behavior of the system has two major aspects. The mechanical behavior of the tissue itself is governed by equations for two-dimensional elasticity, so-called *plane stress*. The interaction between the needle and the tissue is a form of *stick-slip* friction. When the needle moves slowly, material sticks to the needle. At higher speeds the material slips across the needle. Hence, the needle forms a part of the boundary where either displacements (stick friction) or tractions (slip friction) are prescribed.

6.3.1 Elasticity

For problems where loads and deformations occur in a plane, one coordinate can be removed from the 3D elasticity formulation. We will analyze the situation shown in Figure 6.1. We describe $\mathbb{R}^3$ in matrices with respect to the unit basis $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$. The coordinates are also written as $\boldsymbol{z} = (x, y, z)$. The object that we will analyze is a slab of material, lying in the xy-plane, symmetrically around $z = 0$. We assume that deformations in the z-coordinate are uniform, so we describe the z-coordinate of a motion as $p_3(x, y, z) = zq(x, y)$, for some function q. The motion $\mathbf{p}$, mapping reference points to deformed locations, is given by

$$\mathbf{p}(x, y, z) = \begin{pmatrix} p_1(x, y) \\ p_2(x, y) \\ zq(x, y) \end{pmatrix},$$

for some functions $p_1, p_2 : \mathbb{R}^2 \to \mathbb{R}$.

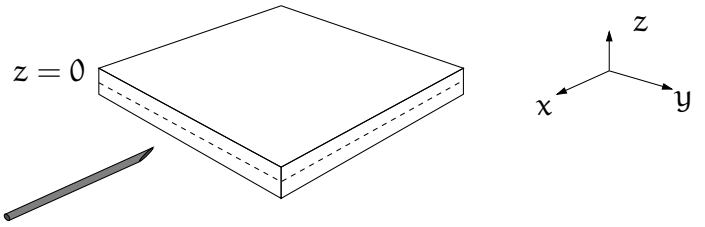


Figure 6.1: Plane stress. When loading thin slabs of material, the z sides are unloaded. Without loss of generality we assume that the slab is positioned around the $z = 0$ plane.

For ease of notation, we leave out the (x, y) dependencies, and denote $\partial f/\partial x$ by f_x. The deformation gradient is given by

$$\mathbf{F} = \begin{pmatrix} p_{1,x} & p_{1,y} & 0 \\ p_{2,x} & p_{2,y} & 0 \\ zq_x & zq_y & q \end{pmatrix}.$$

The strain tensor $\mathbf{C}$ can be written as

$$\begin{pmatrix} p_{1,x}^2 & p_{2,x}p_{1,y} & 0 \\ p_{2,x}p_{1,y} & p_{2,y}^2 & 0 \\ 0 & 0 & q^2 \end{pmatrix} + z \begin{pmatrix} zp_{1,x}^2 & zp_{2,y}p_{2,x} & qq_x \\ zp_{2,y}p_{2,x} & zp_{2,y}^2 & qq_y \\ qq_x & qq_y & 0 \end{pmatrix}.$$

In the $z = 0$ plane, the right term disappears. Except for the q^2 term, the left term is the exact 2D analogon of $\mathbf{C}$.

The q^2 term is eliminated by using the constitutive equations: when we suppose that no stresses act in the third dimension, then by the definition of the first Piola-Kirchoff stress tensor, we should have

$$\mathbf{T} \cdot \boldsymbol{e}_3 = 0.$$

The second Piola-Kirchoff stress tensor $\mathbf{S}$ is defined by $\mathbf{F}^{-1} \cdot \mathbf{T}$, so we have $\mathbf{S} \cdot \boldsymbol{e}_3 = 0$ as well. Since $\mathbf{S}$ is symmetric, we conclude that

$$\mathbf{S} = \begin{pmatrix} \tilde{\mathbf{S}} & 0 \\ 0 & 0 \end{pmatrix}, \tag{6.1}$$

where $\tilde{\mathbf{S}}$ is the 2D restriction of the second Piola-Kirchoff stress tensor.

We will first analyze the results for the linear material from Equation (2.16),

$$\mathbf{S} = \mu(\mathbf{C} - \mathbf{I}) + \frac{\lambda}{2} \operatorname{trace}(\mathbf{C} - \mathbf{I})\mathbf{I}. \tag{6.2}$$

It follows from Equation (6.1) and Equation (6.2) that

$$0 = \mu(q^2 - 1) + \frac{\lambda}{2}(\operatorname{trace}(\tilde{\mathbf{C}} - \tilde{\mathbf{I}}) + (q^2 - 1)),$$

where $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{I}}$ are the 2D analogons of $\mathbf{C}$ and $\mathbf{I}$. Therefore,

$$q^2 - 1 = \frac{2\mu\lambda}{\lambda + 2\mu} \operatorname{trace}(\tilde{\mathbf{C}} - \tilde{\mathbf{I}}).$$

It follows that

$$\tilde{\mathbf{S}} = \mu(\mathbf{C} - \mathbf{I}) + \frac{2\mu\lambda}{\lambda + 2\mu} \operatorname{trace}(\tilde{\mathbf{C}} - \tilde{\mathbf{I}})\tilde{\mathbf{I}}, \tag{6.3}$$

We see that the reduction from 3D to 2D only impacts the volume-preservation part of the equations, and can be derived by setting $\boldsymbol{e}_3 \cdot \mathbf{S} \cdot \boldsymbol{e}_3 = 0$, and substituting the result for q^2 back into the constitutive equation. More complex constitutive equations give more complex expressions for q^2. However, if we set $\lambda = 0$ (or equivalently, $\nu = 0$), then the material has no volume preservation at all, and the 3D formulas can be translated to 2D directly. For example, Equation (4.6) specifies that neo-Hookean material satisfies

$$\mathbf{S} = \mu\mathbf{I} + (-\mu + \lambda(\sqrt{\iota_3} - 1)\sqrt{\iota_3})\mathbf{C}^{-1}.$$

When $\lambda = 0$, then setting $\boldsymbol{e}_3 \cdot \mathbf{S} \cdot \boldsymbol{e}_3 = 0$ implies $q^2 = 1$, and we can set

$$\tilde{\mathbf{S}} = \mu(\tilde{\mathbf{I}} - \tilde{\mathbf{C}}^{-1}). \tag{6.4}$$

We see that Equations (6.4) and (6.3) are the direct 2D analogons of 3D equations in (2.22) and (4.6). The tensors $\mathbf{T}, \mathbf{S}$, and $\mathcal{E}$ have their familiar meanings, but now they are located in $\mathbb{R}^{2\times 2}$. We use linear triangle elements, so the gradient $\mathbf{G}$ of the deformation is given by $\mathbf{U} \cdot \mathbf{Z}^{-1}$. Nodal elastic forces are given by $\mathbf{T} \cdot \mathbf{Z}^{-*}$.

A difference between 2D and 3D is formed by the traction. If the 2D traction (unit N/m), is denoted by $\tilde{t}$, then the 3D traction t (unit N/m^2) is given by

$$t = \tilde{t}/d,$$

where d is the thickness of the slab of material.

6.3.2 Stick-slip friction

During a needle insertion, the needle and the material interact, exchanging forces. These forces are caused by friction. The magnitude of the friction depends on the relative velocity of the needle, i.e. the difference between the velocity of the needle and the material.

Stick-slip friction for an elastic system with one degree of freedom, like the one in Figure 6.2, leads to an ordinary differential equation. In this case, a mass m is attached to a spring with spring constant k, and the mass is subject to stick-slip friction f^{fr} applied by a needle moving at velocity v_{needle}. The friction depends on the relative velocity $v_{needle} - v$. The differential equation describing the movement u and velocity v of the mass are given by

$$\begin{pmatrix} \dot{v} \\ \dot{u} \end{pmatrix} = \begin{pmatrix} \frac{1}{m}(ku + f^{fr}(v_{needle} - v)) \\ v \end{pmatrix}. \tag{6.5}$$

If the relative velocity is nonzero, then the friction force is nonzero and has the opposite direction. An example of a velocity/friction function is in Figure 6.3. In general, the right-hand side of differential equation (6.5) is discontinuous. Integrating such systems requires special precautions, and small time steps [59].

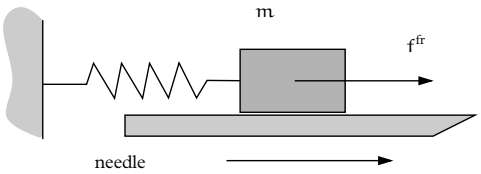


Figure 6.2: A single-degree of freedom stick-slip system

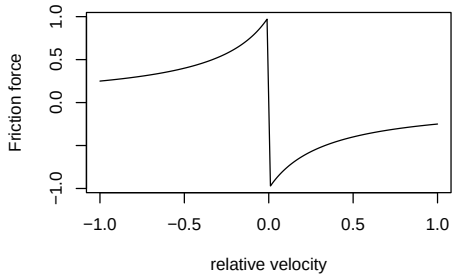


Figure 6.3: A typical stick-slip friction curve, after [59]. The force has a discontuity at $v_{needle} - v = 0$, causing problems during time integration.

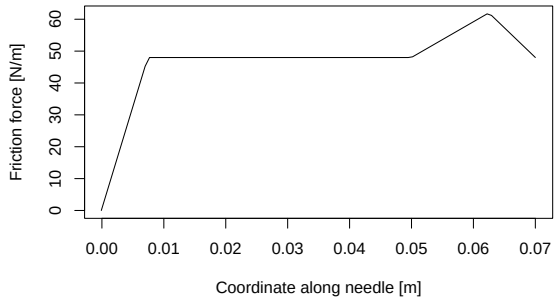


Figure 6.4: The force distribution used for experiments, for the insertion. Friction force varies along the shaft of the needle, and this graph shows how. For stick/slip friction, this distribution is used with deformed coordinates along the x-axis.

Due to these numerical problems, we use a quasi-static formulation. Nodes that lie on the needle, *needle nodes*, can have two states: either a node is fixed (sticking), which means that its location is attached to the user controlled needle and $v_{\text{needle}} = v$, or its movement is constrained to be parallel to the needle shaft (slipping). A configuration of slipping and sticking nodes leads to a single static deformation problem. When the solution of this problem is found, the elastic reaction forces can be used to rearrange the boundary conditions. For every needle node, a static and a dynamic friction threshold is defined. Fixed nodes whose reaction forces exceed the static friction threshold are loosened, and slipping nodes whose elastic forces are less than the dynamic friction threshold are fixed again. If any of the conditions on the nodes are changed, then this rearrangement defines a new static problem, and the procedure is repeated. Once no further rearrangements are necessary, the final solution has been reached. In effect, this procedure is an iterative approach. To distinguish this outer-loop iteration from the relaxation procedure (an inner loop), we shall refer to these iterations as *rearrangements*.

The magnitude of the friction forces on points along the needle shaft is variable. DiMaio and Salcudean demonstrated [36] that the force distribution is similar to the one plotted in Figure 6.4, which is what we will use throughout this chapter. The force bulge near the tip accounts for the force required to make the needle tip cut into the material.

6.4 Needle representation

We will discretize the elasticity problem using the Finite Element Method on a triangle mesh. As we explained above, the needle surface is part of the boundary of the domain. In a *conforming* finite element method, the space of shape functions should be a subspace of the total solution set of the original equation. In the case of needle insertion, this implies that the needle should be represented in the mesh by a set of connected

edges. In previous work, this was achieved by either relocating existing mesh nodes to be on the needle surface [2, 36], by inserting new nodes [2], or by deforming material locally to force nodes to be on the needle [38].

In Chapter 3 and Chapter 5 changes to the mesh were driven by visual reasons. Without subdivision or relocation, cuts will have a jagged appearance which looks unrealistic. For needle insertion, the needle itself is not visible. When we look at the solution accuracy, we observe that in general, the accuracy of a FEM solution is bounded by h, the size of the element, and p, the smoothness of the basis functions. For a given problem with solution $\hat{u}$, a linear FEM discretization will satisfy the following error bound for the approximation u_h [14]

$$\begin{aligned} \|\hat{u} - u_h\|_\infty &= \mathcal{O}(h^2 |\log(h)|^{3/2}), \qquad h \to 0 \\ \|\hat{u} - u_h\|_2 &= \mathcal{O}(h^2), \qquad h \to 0. \end{aligned} \tag{6.6}$$

Neither relocating nodes in the mesh, nor unjudicious use of subdivision decreases h, so in general, neither will improve the accuracy of the solution.

In Chapter 3 we have seen that node relocation can easily lead to flattened elements, and in Chapter 5, we have seen that accomodating surface in the mesh by subdividing intersected elements can easily lead to short edges and a significant increase in mesh size. These problems motivate us to experiment with a scheme where nodes are not moved within the mesh.

The needle is assumed to be an inflexible line segment. When the needle tip enters an element through an edge, the closest node of the edge is marked as a *needle node*. The location of the node is decomposed in two components, as shown in Figure 6.5. The distance along the needle shaft between nodes, d_{needle}, is used as a basis for friction computations. The component perpendicular to the needle, d_{perp}, is used to compute new node positions when the needle is displaced.

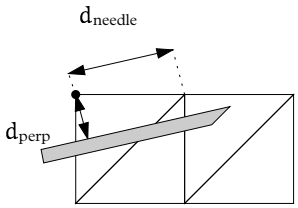


Figure 6.5: When needle nodes are intercepted, their position is decomposed in a component d_{perp} perpendicular to the needle, and a component d_{needle} along the needle. The component d_{perp} is held fixed during needle movements and d_{needle} is used for friction calculations.

6.5 Edge bisection

We will consider a square object that can be meshed with regularly arranged right-angled triangles, as shown in Figure 6.8. This is not an essential restriction, but it does

ease the discussion and the implementation. To accomodate the needle, the resolution of the mesh is locally increased during a simulation. This refinement is done using *edge bisection*. In this case, the basis of edge bisection is a *simple bisect*, splitting an edge which is the hypotenuse of its two incident triangles. The following pseudo-code demonstrates the bisection of an edge (p, q), and is illustrated in Figure 6.6.

procedure simple-bisect (p, q):
  $n \leftarrow$new node halfway between p and q.
  let f_1, f_2 be the faces incident with (p, q)
  if $f_1 \neq$ null:
    $f_1' \leftarrow f_1$ with $p := n$ substituted
    $f_1'' \leftarrow f_1$ with $q := n$ substituted
  if $f_2 \neq$ null:
    $f_2' \leftarrow f_2$ with $p := n$ substituted
    $f_2'' \leftarrow f_2$ with $q := n$ substituted
  replace $\{f_1, f_2\}$ with $\{f_1', f_1'', f_2', f_2''\}$.

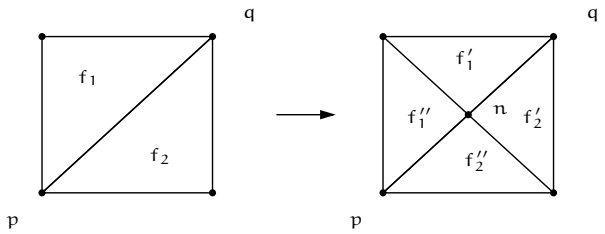


Figure 6.6: In a simple bisection, the faces $\{f_1, f_2\}$ are replaced by $\{f_1', f_1'', f_2', f_2''\}$.

The bisected edge must be the hypotenuse of its incident triangles. If this is not the case, the following recursive procedure first bisects neighbors before it bisects the specified edge e. It is illustrated in Figure 6.7.

procedure bisect(e):
  let f_1, f_2 be incident with e
  if $f_1 \neq$ null $\wedge\, e \neq$ hypotenuse(f_1):
    bisect (hypotenuse (f_1))
  if $f_2 \neq$ null $\wedge\, e \neq$ hypotenuse(f_2):
    bisect (hypotenuse (f_2))
  simple-bisect (e)

For a mesh consisting of right-angled triangles, this refinement process is equivalent to two other techniques. The hypotenuse is always the longest edge in a triangle, and hence this procedure is equivalent to the so-called longest edge bisection [81]. When

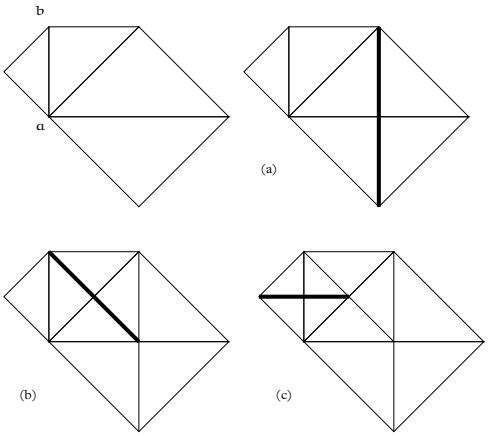


Figure 6.7: When an edge is bisected that is not the hypotenuse of its incident triangles, then its neighbors are first bisected: in pictures (a), and (b) neighbors are bisected, in (c), finally ab is bisected.

bisecting a hypotenuse, the angle opposite the hypotenuse is bisected. Since that angle is always the newest vertex in a triangle, we have a form of newest node bisection [85].

The bisection technique we described has the following properties.

- It is easy to implement.
- Triangles are naturally ordered in a hierarchy. This hierarchy can be stored in a binary tree. This tree is in effect a BSP tree, suitable for efficient point-location.
- All triangles are congruent to triangles of the starting mesh, and therefore well shaped.
- It can be extended to simplexes in arbitrary dimensions, and this extension maintains the above properties [62]. The only requirement is that the nodes are ordered in some way.

The starting mesh can be refined uniformly by subdividing all hypotenuses in the mesh repeatedly. The mesh can also be arbitrarily refined in a region, by repeated bisection. The following procedure refines around the point $\mathbf{x}$ until the edge lengths around $\mathbf{x}$ are less than h.

```
procedure refine-around (x, h)
  t ← triangle containing x
  if |hypotenuse (t)| > h:
    bisect(hypotenuse (t))
    refine-around (x, h)
```

Figure 6.8 illustrates this refinement process.

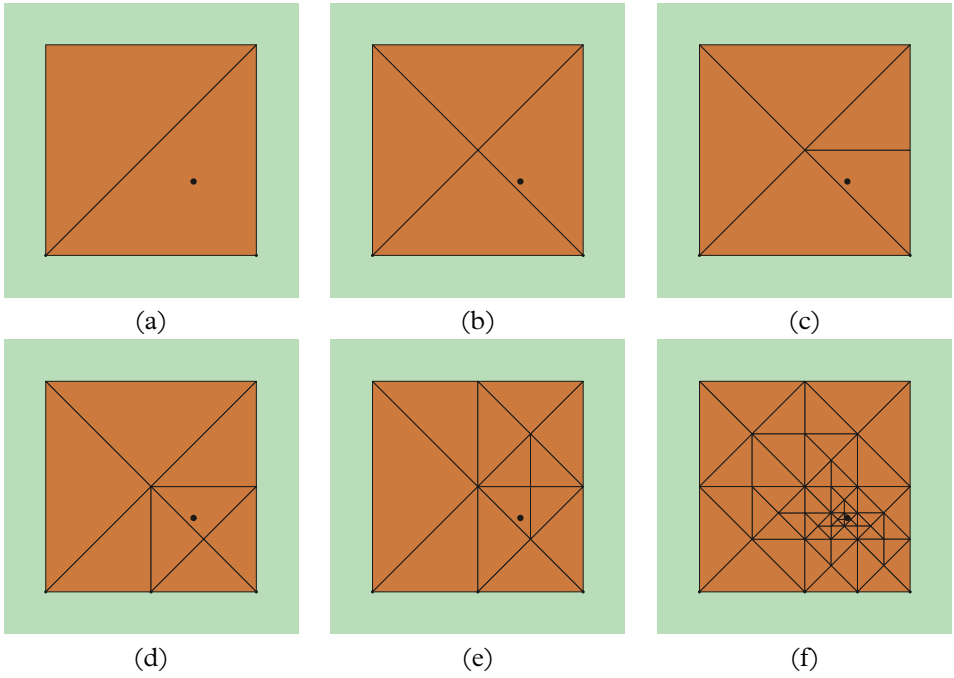


Figure 6.8: Recursive edge bisection, with refinement around the point marked with a dot. Pictures (a) shows the starting mesh. Picture (b) and (c) show simple bisections. In Picture (d) neighbors are bisected as well, and in Picture (e), the bisections propagate even further. Finally, Picture (f) shows the result after 10 refinements.

6.6 Solution method

We will use the CG method, both in its linear and nonlinear form, to compute solutions to the FEM problem. Two modifications with respect to the method of Chapter 3 are necessary. There, we have only considered fixed nodes. In the case of needle insertion, nodes can also be constrained to move along a line. In effect, both types of constraint restrict the minimization problem on $\mathbb{R}^n$ to a subspace $W \subset \mathbb{R}^n$. In other words if u is a given starting configuration, and W the space of allowed node movements, then the solution is given by

$$\min_{w \in W} \Pi(u + w).$$

If P_W is the orthogonal projection on W, then we can rewrite this as

$$\min_{v \in \mathbb{R}^n} \Pi(u + P_W v). \tag{6.7}$$

The latter is a minimization problem in $\mathbb{R}^n$, and hence we can solve it with the techniques discussed in Section 2.4, using the function in (6.7) as objective In other words, we take

$$P_W \left(\frac{\partial \Pi}{\partial u}(u + P_W v) \right) \quad \text{and} \quad P_W \left(\frac{\partial^2 \Pi}{\partial u^2}(u + P_W v) \right) P_W$$

for the derivative and second derivative. In effect, search directions and second derivatives are first projected onto the constraints before they are used in the algorithms.

A second complication is the stopping criterion. The stopping in Chapters 3 and 4 uses the 2-norm of the residual force vector r, i.e.,

$$\|r\|_2 \leq \varepsilon \|f^{ex}\|_2, \tag{6.8}$$

for some *relaxation tolerance* $\varepsilon > 0$. This criterion does not work when nonzero displacements are prescribed without applying external forces: then the right handside is 0, which is impossible to satisfy due to rounding errors. We do have access to f^{react}, the reaction forces necessary to keep nodes in their constrained positions; these can also be regarded as a form of external force. They can be found as

$$(I - P_W) \frac{\partial \Pi}{\partial u}(u).$$

Therefore, we propose the following stop criterion for the relaxation loop.

$$\|r\|_2 \leq \varepsilon \sqrt{\|f^{ex}\|_2^2 + \|f^{react}\|_2^2} + \frac{E d H}{\sqrt{n}} \varepsilon_{round}. \tag{6.9}$$

In this criterion, $\varepsilon_{round} \ll 1$ is a separate tolerance, H is the diameter of the object, and E is the Young modulus. In this equation, the first term is the analogon of Equation (6.8) that also uses reaction forces. The second term of the right hand side becomes significant when the first term is almost 0 due to rounding errors. In that case, the first term may be hard to satisfy due to rounding errors in the evaluation of $\|r\|$. The second term is a rough estimate of the rounding errors in both quantities. The quantity dH is

a measure of the area of the object, so EdH is a scale-free measure of the force magnitude. The factor $\varepsilon_{round}/\sqrt{n}$, with , is a measure for the roundoff errors that accumulate during the evalution of $\|r\|$ and $\sqrt{\|f^{ex}\|_2^2 + \|f^{react}\|_2^2}$.

This stopping criterion does not use the linear elasticity assumption in any way, and may therefore be used for nonlinear iterations as well. For 3D, the term dH should be changed, for example to H^2.

6.7 Implementation

The previous sections provide enough technniques to implement a needle insertion simulation based on edge bisection and static elasticity. The implementation was based on the deformation and relaxation framework written in Chapter 4, and the mesh data structure that will be described in Chapter 7.

In the implementation, the starting mesh is uniformly refined until edge lengths have a specified length h_{start}. Then the simulation enters the following interaction loop where needle movement, friction and refinement are handled.

while true:
  move needle
  refine mesh down to h_{ref} around the needle tip
  add intercepted nodes to needle
  compute solution using CG
  rearrange boundary conditions
  unconstrain nodes that slid off the needle

When the needle is moved, the locations of all needle nodes are expressed in d_{perp} and d_{needle}. Needle nodes are moved by computing d_{perp} and d_{needle} relative to the old needle position, and using them as coordinates relative to the new needle position.

6.8 Computational experiments

The question to be answered is whether the simulation is practically useful. As far as the physical model is concerned, both our scheme and the one of DiMaio and Salcudean are equivalent, since they use the same elasticity model, and the same friction parameters. Therefore, we can achieve the same result, provided that we may spend as much computation as necessary to attain the desired accuracy. Therefore, the real question is: how accurate is the system for a given amount of computation? We will measure accuracy in terms of average and maximum errors in the displacement. We set the desired accuracy at approximately 1 mm, which roughly corresponds to the resolution of CT scan and MRI scans.

The simulation runs a number of static linear problems in sequence. The question of accuracy versus speed will be answered by first considering the error in the simulation under static load. This gives us a combination of relaxation tolerance and mesh

resolution. These settings are used to quantify the buildup of errors during a simulation. This error is considered by running an automated, standardized needle insertion at various insertion speeds. Timings of the same experiment show to what extent the simulation is interactive.

Finally, we will consider the error that is introduced by not relocating nodes, the *conformance error*. An advantage of our approach, which also allows nonlinear material, is demonstrated by comparing insertions for nonlinear and linear material.

6.8.1 Test situation

The test object is a slab with dimensions $0.10 \times 0.10 \times 0.01$ m, and material properties $E = 34\,\text{kPa}$, $\nu = 0.34$. We assume that the object is lying in in the xy-plane, with $0 \leq x \leq 0.1\,\text{m}$ and $0 \leq y \leq 0.1\,\text{m}$. Refinements of element size are done by factor 2: we set $h_k = 0.1\,\text{m} \cdot 2^{-k}$, and select both h_{ref} and h_{start} from $h_1, \ldots, h_7$. We have used $\varepsilon_{\text{round}} = 10^{-8}$ as the tolerance for rounding errors.

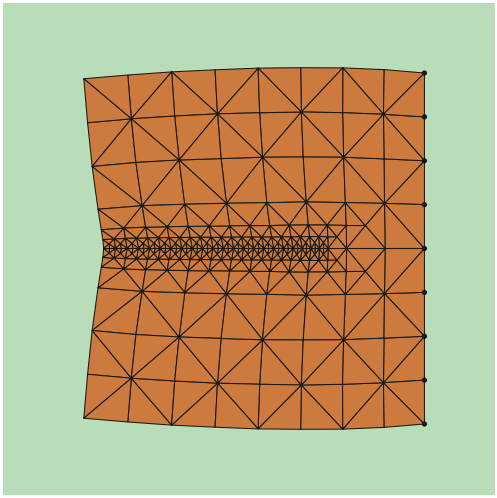


Figure 6.9: Experimental load for determining speed/accuracy. The object is fixed on the right, and a load is applied to the center, symmetrically. This picture shows $h_{\text{start}} = h_3$, $h_{\text{ref}} = h_6$.

The speed versus accuracy experiments are performed in a symmetric setting: in the static experiment, forces are applied along a line at $y = 0.05\,\text{m}$, while the boundary at $x = 0.1\,\text{m}$ is held fixed. This configuration is shown in Figure 6.9. It leads to a symmetric deformation. The "needle" is represented by edges in the mesh, and hence, the accuracy assessment is not affected by conformance errors. In the static problem, load is applied to the center of the object over a length of 70 mm, using the distribution in Figure 6.4. In the dynamic problem, the needle is inserted to a depth of 70 mm, with dynamic friction thresholds derived from the given force distribution. The static friction threshold is set at twice the dynamic threshold.

6.8.2 Residual tolerance

The stopping criterion in (6.9) is parameterized by a tolerance ε. In Table 6.1 shows how the the value of the tolerance affects displacement errors, by comparing solutions of the same problem calculated with different tolerances. We can see that very modest tolerances already are enough to make displacement errors small.

relaxation tolerance ε	average error [mm]	maximum [mm]
0.3	0.74	2.98
0.1	0.061	0.18
0.03	0.034	0.083
0.01	0.011	0.022
0.003	0.0038	0.0096
0.001	0.0008	0.0021

Table 6.1: Error dependency on tolerance for the residual force, relative to the solution computed at $\varepsilon = 10^{-8}$. This has been run on a mesh with $h_{\text{ref}} = h_{\text{start}} = h_7$.

6.8.3 Mesh density

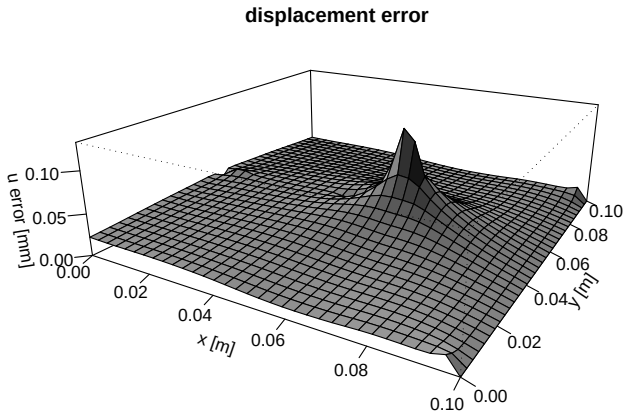
The impact of mesh resolution on the solution error was analyzed by comparing the results for $h_{\text{ref}} = h_{\text{start}} = h_7$ with results for coarser meshes. These results are given in Table 6.3. Here, both h values are varied from h_1 to h_7.

Plots of the displacement error for two combinations of h_{ref} and h_{start} are in Figure 6.10. It is evident that errors are especially present in the vicinity of "interesting" regions: regions where force varies, or near the corners of the fixed boundary. Hence, the mesh can be left more coarse outside these regions. This distributes the displacement error all over the object. Moreover, convergence is also quicker, as demonstrated in Table 6.2.

$\downarrow h_{\text{start}}/h_{\text{ref}} \rightarrow$	h_3	h_4	h_5	h_6	h_7
h_1	22	34	45	62	84
h_2	27	36	46	64	84
h_3	46	49	58	73	93
h_4		81	86	96	115
h_5			204	208	194
h_6				278	285

Table 6.2: CG Iteration counts for different combinations of h_{start} and h_{ref}. Reducing mesh resolution also decreases convergence requirements: information travels faster over coarser mesh parts, thus speeding up the relaxation.

The accuracy results for varying h_{start} and h_{ref} are in Table 6.3. In general, decreasing h_{start} and decreasing h_{ref} improves the accuracy. For $h_{\text{start}} = h_3$ and $h_{\text{ref}} = h_6$, we get an error of 0.44 mm, which is small enough by our standards.



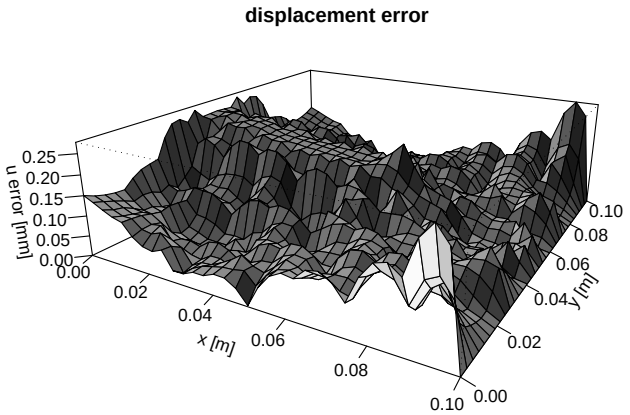


Figure 6.10: Displacement errors, at the top $h_{start} = h_6$ and $h_{ref} = h_6$, on the bottom $h_{start} = h_3$, $h_{ref} = h_6$. Larger errors occur close to varying loads: the corners of the fixed sides, and close to the needle tip and entry point. Therefore, a finer mesh is only necessary in these regions. On the bottom, the mesh is coarser away from the needle, which leads to larger errors in those regions.

Average error [mm]					
$\downarrow h_{start}/h_{ref} \rightarrow$	h_3	h_4	h_5	h_6	h_7
h_1	0.34	0.33	0.33	0.30	0.28
h_2	0.26	0.24	0.24	0.21	0.20
h_3	0.15	0.13	0.13	0.10	0.08
h_4		0.09	0.08	0.04	0.03
h_5			0.07	0.03	0.00
h_6				0.02	0.00

Maximum error [mm]					
$\downarrow h_{start}/h_{ref} \rightarrow$	h_3	h_4	h_5	h_6	h_7
h_1	1.01	0.88	0.88	0.87	0.87
h_2	0.91	0.68	0.61	0.54	0.53
h_3	0.66	0.54	0.51	0.29	0.27
h_4		0.45	0.43	0.19	0.14
h_5			0.41	0.17	0.06
h_6				0.16	0.03

Table 6.3: Displacement errors for different mesh resolutions. These tables list mean and maximum displacement errors in mm (compared with a uniform mesh with element size h_7), with h_{start} vertically, and h_{ref} horizontally. Half of the table is empty, since $h_{ref} \leq h_{start}$. We see that smaller h size leads to smaller errors in general.

6.8.4 Insertion speed

If the needle moves very quickly during an insertion, it will pierce elements without requiring force. For example, if the needle moves from outside the object to its final position in one step, no forces will be exchanged between needle and tissue. For this reason, it can be expected that the insertion speed influences the end result. We assume that the better solutions are obtained when the speed of the needle is smaller. Hence, we measure that influence by performing automated insertions at different speeds, and comparing them to the results for a very slowly inserted needle.

The automated insertion is performed as follows: after each cycle of the interaction loop, the needle is advanced by a fixed amount, until it reaches the desired penetration depth. The loop continues until no more rearrangements of needle boundary conditions are needed. The system then has reached an equilibrium situation.

Results of this experiment are in Table 6.4. These results do not show a clear trend. For insertions speeds smaller than $1.0h_{ref}$ per update, we get errors that are in the same order of magnitude as the discretization error.

6.8.5 Timings

When the experiment of the previous subsection is timed, we can estimate how interactive the simulation is. Table 6.5 lists for each insertion speed the amount of CPU time used, and how many times the interaction loop from Section 6.7 is called. Di-

speed $[\frac{h_{ref}}{\text{rearrangement}}]$	avg error [mm]	max error [mm]
3.00	0.62	1.60
1.00	0.68	1.96
0.30	0.15	0.34
0.10	0.18	0.68
0.03	0.36	1.08
0.01	0.07	0.27

Table 6.4: Error introduced by insertion speed, for $h_{start} = h_3$ and $h_{ref} = h_6$, compared to insertion at the speed $0.005 h_{ref}$. For this resolution, the discretization error was 0.11 mm average/0.29 mm maximum.

viding both gives an average update rate. We see that low insertion speeds give better update rates. This is confirmed by the average number of CG iterations necessary to find a solution. Smaller insertion speeds lead to a higher amount of coherence between subsequent solutions, thus reducing the average number of CG iterations required. For the lowest insertion speeds ($0.10\ h_{ref}$/rearrangement and lower, we obtain update rates suitable for haptic interaction).

The experiment was done with two different refinement resolutions, $h_{ref} = h_6$ and $h_{ref} = h_7$. The number of rearrangements doubles when going from h_6 to h_7 (a decrease of a factor 2), this is consistent with the fact that insertion speed is proportional to h_{ref} in this experiment.

The average update rate is better than the worst update rate, which is indicated by the maximum number of CG iterations required. The maximum is larger than average. These larger counts are necessary during the beginning of the insertion, when only a small portion of the needle is in the tissue. Although the mesh is still relatively small at this stage, it might be necessary to set a fixed bound on the number of CG iterations when driving a force-feedback device.

6.8.6 Relocation errors

In the following experiment, we assess the magnitude of conformance errors. To this end, a node repositioning scheme similar to the one in Chapter 3 was introduced. Nodes selected as needle nodes are projected orthogonally onto the needle. Again, we quantify errors by applying a static load to the mesh, according to the friction distribution from Figure 6.4, but now forces are applied over a tilted line, so that the "needle" does not coincide with mesh edges. The resulting deformation is shown in Figure 6.11. By measuring the difference between a solution with relocation and without relocation to the solution a finer mesh, we can estimate the impact of relocation. Table 6.6 shows that these errors are smaller than the discretization errors, thus validating the approach.

6.8.7 Nonlinearity effects

Our solution method does not exploit the linearity of the problem. Hence we can also use nonlinear material models. In Chapter 4 we saw that this is necessary when large

speed $\left[\frac{h_{ref}}{update}\right]$	CPU time [s]	rearrangements	updates [Hz]	CG iters (avg)	CG (max)
$h_{start} = h_3, h_{ref} = h_6$					
3.00	0.80	157	196.25	11.81	41
1.00	0.90	232	257.78	11.52	62
0.30	0.94	308	327.66	10.87	70
0.10	1.07	504	471.03	8.14	61
0.03	2.24	1478	659.82	5.31	57
0.01	5.77	4187	725.65	3.87	51
$h_{start} = h_3, h_{ref} = h_7$					
3.00	2.46	243	98.78	12.67	57
1.00	2.80	401	143.21	10.88	82
0.30	3.15	561	178.10	10.03	66
0.10	3.48	1000	287.36	6.63	67
0.03	6.34	2966	467.82	4.18	53
0.01	15.68	8505	542.41	3.15	48

Table 6.5: Timings for an insertion procedure at different speeds, done with two resolutions. Smaller movements decrease the number of CG iterations necessary, and hence increase update frequencies. We can see that decreasing h_{ref} by a factor 2 doubles the number of boundary condition rearrangements required. Timings were done on a Pentium III/1 Ghz, with visualization switched off. The program was compiled with GCC 3.2 with maximum optimization options switched on.

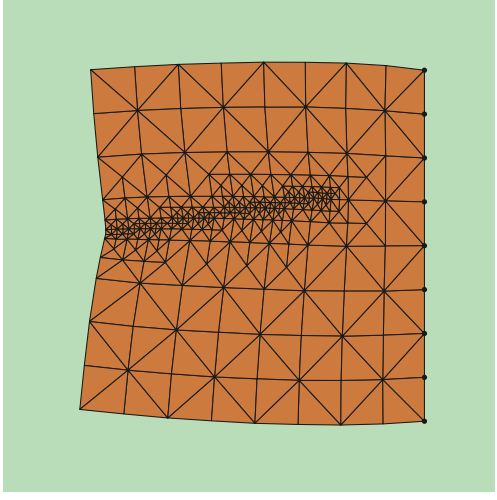


Figure 6.11: Applying forces along an angled trajectory.

	average error [mm]	max error [mm]
with relocation vs. without	0.019	0.23
discretization error with relocation	0.094	0.43
discretization error without relocation	0.095	0.37

Table 6.6: Magnitude of conformance errors for $h_{start} = h_3, h_{ref} = h_6$. We see that the difference introduced by node repositioning is smaller than the discretization error, computed relative to h_7 mesh resolution.

deformations are simulated. In Figure 6.12 a scenario is shown that involves large deformations. The needle is inserted sideways into a slab of material fixed at the bottom. In this case, the needle describes a trajectory that is curved, when viewed in the reference configuration. The material used is neo-Hookean material[1] versus linear material, with $E = 34\,\text{kPa}$ and $\nu = 0$. The difference between the final location of needle tip in both experiments is approximately 8 mm.

6.9 Discussion

In this chapter we have presented a novel method for computing simulated needle insertions into 2D elastic material. The method builds on previous work by its use of a quasi-static model of stick/slip friction with plane-stress elasticity. It is different in that it uses an iterative (and optionally nonlinear) relaxation algorithm, and adaptive mesh resolution. The mesh used has a high degree of regularity, and is refined near the inserted needle to improve the local accuracy. Nodes are not moved within the mesh, so the refinement technique, edge bisection, does not cause element shape deterioration. To assess the cost/accuracy ratio of this method, it was implemented in a prototype and subjected to a number of computational experiments. On the basis of these experiments, we conclude that accuracies around 1 mm for insertion in linearly elastic objects of size $10 \times 10\,\text{cm}$ can be achieved at haptic update rates on a 1 Ghz PC.

The performance of the method is related to coarseness of the mesh, and material model used. Both factors are related to the accuracy of the end result, so it is possible to improve response times by sacrificing accuracy. This compromise is controlled with the parameters ε, the tolerance for the relaxation, h_{ref}, the amount refinement around the needle and h_{start}, the global mesh resolution. For the scenarios analyzed, this proved to give adequate control. This manual step could be automated by estimating errors of the FEM discretization automatically, and refining the mesh adaptively during the simulation [32, 100]. Such estimations would also allow a more rigorous error analysis than those provided in Section 6.8.

In the current simulation, iteration counts for the CG algorithm are very low both due to the high spatial coherence between solutions, and due to the limited mesh resolution outside the region of interest. When finer meshes are necessary, additional techniques must be used to decrease iteration counts. Multigrid techniques can solve

[1] St. Venant-Kirchoff material was also tried, but caused element inversion, which lead to stalling convergence in the nonlinear CG relaxation

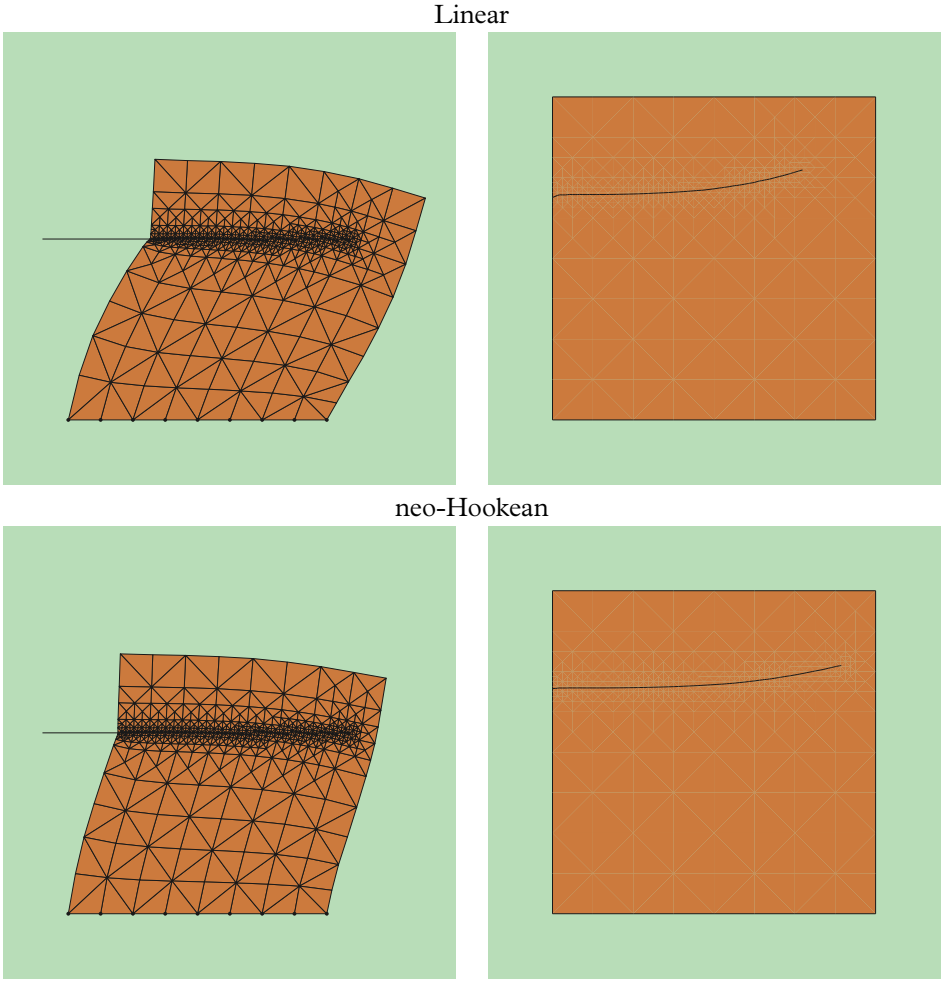


Figure 6.12: Linear elasticity (top) and neo-Hookean elasticity (bottom) compared, both in deformed configuration (left), and undeformed configuration (right). This was done with Poisson ratio $\nu = 0$, and insertion speed $0.01 h_{\text{ref}}$/update. The distance between the tip reference locations in both experiments is approximately 8 mm. The linear experiment took 16 seconds of CPU time; the neo-Hookean experiment took 38 seconds, and ended with a 625 triangle mesh.

elliptic FEM problems in $\mathcal{O}(\log n)$ iterations, where n is the number of degrees of freedom. Multigrid methods run *smoothing* relaxations at different resolutions of the same problem. Refinement by edge bisection naturally produces such hierarchical meshes. Smoothing techniques, like the Gauss-Seidel iteration, work by traversing the stiffness matrix row-by-row or column-by-column. This implies that all edges of the mesh should be tracked across mesh changes. Since these smoothing techniques are also used as preconditioners for the CG iteration, a first step in this direction is to implement them as preconditioners.

The extensions mentioned in the previous paragraphs are aimed at improving response times for larger and more accurate simulations. In this light, it is instructive to compare the magnitude of different types of solution errors. From the computational experiments in Section 6.8 we can deduce that these errors have the following causes (in order of decreasing impact).

- The tissue model.
- The error buildup during insertion, an aggregrate of the error sources below.
- The discretization error, caused by using a coarse mesh.
- The conformance error, caused by not moving nodes onto the needle.
- The relaxation error, caused by using an iterative algorithm.

We see that a good tissue model is crucial to making accurate predictions of the effect of tissue deformations. Therefore, research into improvements in computational techniques should be accompanied by a more in-depth analysis of the mechanical properties of the organs being modeled. A complete analysis should also include sensitivity to needle flexion (we assume the needle to be rigid) and variation in material parameters.

The deformable object was a square slab of material. This simplifies implementation, but it is not an essential restriction. The refinement technique is easy to understand for regular simplicial grids of the square, but its only requirement is that vertices are ordered [62]. Moreover, for triangulations in 2D, there is a variety of techniques based on Delaunay refinement [25, 82] that can naturally accomodate more complex object geometries. However, this approach has been chosen with the intent of generalizing to 3D insertions easily. In 3D, both generating the Delaunay tetrahedralization for objects with complex geometrical shapes is much harder [86], and can generate slivers, a type of badly shaped elements. In contrast, edge bisection generalizes naturally to higher dimensions [62].

The 3D generalization of this method is work in progress. Figure 6.13 shows edge bisection in 3D, and Figure 6.14 shows needle insertions in a cube-shaped object. When we look at the computational aspects of the extension to 3D of our method, then we can note that the performance of CG is proportional to the root of the condition number of the stiffness matrix. This quantity is proportional to $1/h$, where h is the element size [6]. Since element size does not change when lifting the simulation to 3D, we can expect that the same number of CG iterations are required. The cost of a single iteration does change. For representing the needle shape accurately in 3D, more refinement is needed, and tetrahedralization require more elements per vertex. The fine-grained simulation in

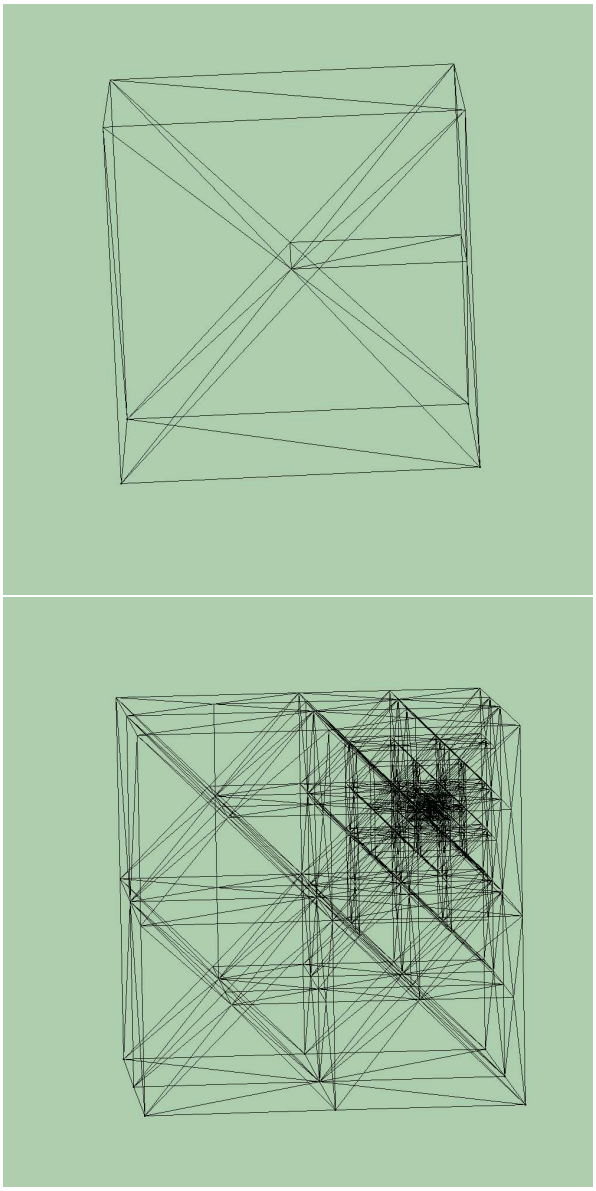
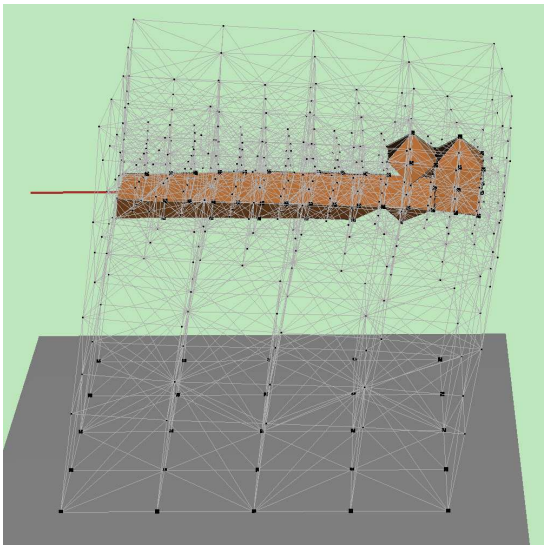
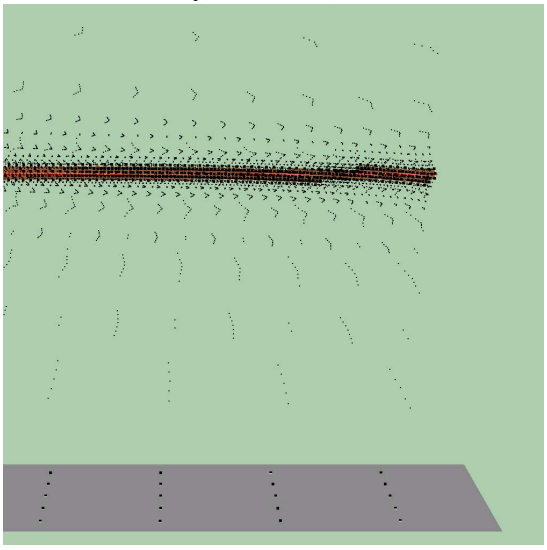


Figure 6.13: Recursive edge bisection in 3D, with refinement around a single point. On the top 3 levels of refinement, on the bottom 20 levels.



linear elasticity, 5 seconds, 2420 elements



neo-Hookean material, 1364 seconds, 29562 elements.

Figure 6.14: Insertions of a needle (radius 1 mm) from the left into a elastic cube (10 cm $\times$ 10 cm $\times$ 10 cm) fixed on the bottom. The needle is represented in the mesh by a jagged surface, which is shown in solid color. The geometry of the needle surface is accounted for in the friction forces. On the top: linear material, with 12 fold refinement around the needle. On the bottom a detail from a similar insertion into neo-Hookean material, with 20 fold refinement around the needle (Mesh edges are not shown). Timings were done on a P3/1Ghz.

Figure 6.14 corresponds to refinement to h_7. The simulation has approximately 30,000 elements, a factor 50 more than the example in Figure 6.12. Matrix operations are also more expensive in 3D. For example, a matrix/matrix multiply costs 12 flops in 2D and 45 flops in 3D, almost a factor 4. Combining these, we can estimate that each iteration in 3D is roughly 200 times more expensive. To reach the same update rates as the 2D simulation, techniques should be used that increase raw computation power: dedicated hardware and parallel processing.