Chapter 4

Relaxation algorithms

This chapter is intended as an expansion of the work of Chapter 3, where we have described our first steps into interactive deformation modeling. Our first approach is a completely linear model with an iterative solution based on the Conjugate Gradient algorithm. We have shown how mesh modification and deformation are easily combined with this method. However, linear elasticity has its limits: it assumes that deformations remain small. This assumption is questionable for soft material, such as soft tissue.

Other work in deformable objects primarily uses dynamic methods to compute deformations. Such methods compute the evolution of deformations over time as they move to a steady state. They are perhaps easier to understand than iterative static methods, since all intermediate results have a physical interpretation. Since they compute more physically relevant information, one could also expect that they are more expensive than a static method.

This chapter addresses both the extension to nonlinear material and convergence speed in more detail. We will extend the deformation framework of the previous chapter to include nonlinear deformations and a dynamic formulation. Using this framework, we benchmark the convergence speed of a static algorithm by comparing it to a dynamic method applied to the same problem. The rest of this chapter starts with detailing theoretical convergence of a dynamic method, then it introduces the convergence experiment, material models, and finally it shows and discusses the results.

4.1 Convergence of dynamic relaxation

The theoretical convergence speed of Conjugate Gradients (CG) has been analyzed extensively in literature, and was discussed in Section 2.4. In this section, we briefly analyze the convergence speed of dynamic relaxation in the case of linear elasticity. We will show how quickly a dynamic method will settle into a steady state, and find that the convergence speed of the dynamic problem is similar to that of CG.

We recall from (2.55) that the PDE for linear elasticity can be discretized into the following n-dimensional differential equation for the function $u(t) \in \mathbb{R}^n$

$$M\ddot{u} + C\dot{u} + Ku + f^{\text{ex}} = 0.$$

Here f^{ex} represents the external force, $M \in \mathbb{R}^{n\times n}$ is the mass matrix, representing the inertia of the object, and $C \in \mathbb{R}^{n\times n}$ the damping matrix, and $K \in \mathbb{R}^{n\times n}$ the stiffness matrix. The integration methods in (2.60) and (2.62) require diagonal M and C matrices for efficient time-stepping, so we use lumped masses. Since C must also be diagonal, we take $C = \eta M$ for some constant $\eta > 0$. This is a form of Rayleigh damping.

This dynamic solution has two parameters: η controls the amount of damping, and Δt is the time step of the integration scheme. Both parameters influence the speed of convergence towards the steady state. We want to determine how quickly this dynamic method reaches the steady state, so that the parameters η and Δt have to be chosen optimally. We determine these optimal parameters by analyzing the evolution of the solution error e over time. We define the error e in a solution u as being the difference between u and a static solution u_{static}. We have $Ku_{static} = f^{\text{ex}}$, so the error $e = u - u_{\text{static}}$ satisfies the homogeneous differential equation

$$M\ddot{e} + \eta M\dot{e} + Ke = 0. \tag{4.1}$$

Solutions of this equation are expressed in terms of generalized eigenvalues of M and K, i.e. solutions to

$$Kw = \lambda Mw$$

Both K and M are symmetric and positive definite, so this generalized eigenvalue problem has M-orthogonal eigenvectors with positive eigenvalues. There are eigenpairs (w_i, λ_i) from $\mathbb{R}^n \times \mathbb{R}^+$, such that $Kw_i = \lambda_i Mw_i$ for $i = 1, \ldots, n$. Since K and M are symmetric, the eigenvectors can be chosen to be M-orthogonal. In addition, the eigenvectors w_j can be normalized, so that we have

$$(w_i, w_j)_M = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases}$$

This expression uses the notation from Equation (2.44).

The vectors w_i represent normalized undamped vibration modes of the body, and form an M-orthonormal basis of $\mathbb{R}^n$. Therefore, we can decompose e into the eigenvectors w_i, writing

$$e(t) = \sum_j y_j(t) w_j, \qquad y_i(t) = (e(t), w_i)_M$$

Since K and M are positive definite, the eigenvalues are positive. We order the eigenvalues, so $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Analogous to Subsection 2.4.1, we analyse the error in the energy norm, which is given by $\|e\|_K$. Due to the M-orthogonality of the w_j, we find

$$\|e\|_K = \sqrt{\sum_j y_j(t)^2 \lambda_j}.$$

In other words, the solution error can be decomposed in its modal components. By taking the M-inner product of (4.1) and a vibration mode w_j, we get the following differential equation for the component y_j:

$$\ddot{y}_j(t) + \eta\dot{y}_j(t) + \lambda_j y_j(t) = 0. \tag{4.2}$$

This is a differential equation with constant coefficients. There are three cases for the general solution: the vibration is either underdamped, critically damped or overdamped. Let

$$\mu = -\eta/2,$$
$$\omega_j = \frac{1}{2}\sqrt{|4\lambda_j - \eta^2|}.$$

If $\eta < 2\sqrt{\lambda_j}$, then the system is underdamped, and solutions take the form of

$$y_j(t) = c_{1j}e^{\mu t}\sin(\omega_j t + \varphi_j), \qquad c_{1j}, \varphi_j \in \mathbb{R}.$$

If $\eta = 2\sqrt{\lambda_j}$, then the system is critically damped, and solutions take the form of

$$y_j(t) = (c_{1j} + c_{2j}t)e^{\mu t}, \qquad c_{1j}, c_{2j} \in \mathbb{R}.$$

If $\eta > 2\sqrt{\lambda_j}$, then the system is overdamped, and solutions take the form of

$$y_j(t) = c_{1j}e^{(\mu+\omega_j)t} + c_{2j}e^{(\mu-\omega_j)t}, \qquad c_{1j}, c_{2j} \in \mathbb{R}.$$

Graphs of these three cases are shown in Figure 4.1.

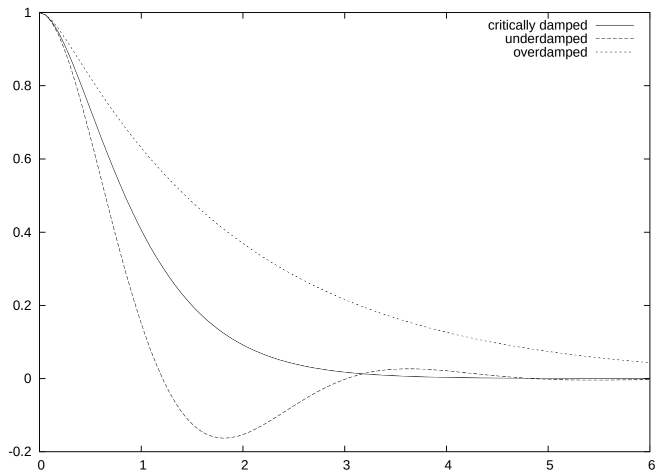


Figure 4.1: Three types of damping demonstrated for Equation (4.2), with critical damping, and $\eta = 1/2\eta_{\text{crit}}$ and $\eta = 2\eta_{\text{crit}}$. Begin values are $\lambda_j = 4$, $y_j(0) = 1$, $\dot{y}_j(0) = 0$.

We see that all modal components of the error diminish over time by $e^{-\eta t/2}$ in the underdamped and critically damped case. If η is larger than $2\sqrt{\lambda_j}$ for any j, then that mode is overdamped, and the corresponding error component will diminish by $e^{-(\eta/2-\omega_j)t}$, which is slower than $e^{-\eta t/2}$. Therefore, the quickest convergence is attained when η is as large as possible, but no mode is overdamped. This is when $\eta = 2\sqrt{\lambda_1}$.

In this case, we have

$$\|e\|_K = e^{-\eta t/2}\sqrt{\sum_j \lambda_j \tilde{y}_j^2(t)}, \qquad \tilde{y}_j(t) = e^{\eta t/2} y_j(t)$$

The contents of the square root are $\mathcal{O}(t)$, so the error is dominated by the exponential term $e^{-\eta t/2}$. Hence, when the equation is integrated over a time span T, then the magnitude all modal components decreases by $e^{-\eta T/2}$. For a reduction ε in error, we have to integrate over a fixed time span

$$T = \frac{-2\ln\varepsilon}{\eta} = \frac{-\ln\varepsilon}{\sqrt{\lambda_1}}.$$

The stability condition of the SS22 and related explicit second order integration methods for (4.2) is given by Zienkiewicz [103]: the time step Δt must satisfy

$$\Delta t^2 \leq \frac{4}{\lambda_j}, \qquad j = 1, \ldots, n.$$

The highest frequency mode is given by the largest eigenvalue λ_n, and this mode must also be stable, so we have

$$\Delta t \leq \frac{2}{\sqrt{\lambda_n}}.$$

If a modal component y_j is to decrease by a factor ε, then this takes at least N time steps, where

$$\begin{aligned} N &= \frac{T}{\Delta t} \\ &\geq \frac{-\ln(\varepsilon)\sqrt{\lambda_n}}{2\sqrt{\lambda_1}} \\ &= \frac{1}{2}\ln\left(\frac{1}{\varepsilon}\right)\sqrt{\kappa}, \qquad \kappa = \lambda_n/\lambda_1 = \text{cond}_2(M^{-1}K) \end{aligned}$$

Recalling Equation (2.47), we see that CG and dynamic relaxation offer similar performance in the linear case: the condition number of K determines the convergence speed. The effect of the mass matrix M is that of a preconditioner: if M were variable, and could be selected to decrease $\text{cond}_2(M^{-1}K)$, then larger time steps could be taken, leading to more rapid convergence. This "preconditioning" has a physical interpretation: when a discretisation has both small and large elements, increasing nodal masses of small elements decreases their vibration frequencies, thus it brings down λ_n. For a system with lumped masses, M is diagonal, so if we view M as a preconditioner, then increasing nodal masses is analogous to preconditioning with a diagonal matrix.

parameter	notation	value
gravity	g	$9.8\ \mathrm{m/s^2}$
density	ρ	$1000\ \mathrm{kg/m^3}$
Young modulus	E	$1.0 \cdot 10^4 \mathrm{Pa}$
Poisson ratio	ν	0.3
Material nonlinearity	γ	8

Table 4.1: Material parameters and constants for the experiments.

4.2 Experimental setup

Subsection 2.4.1 and 4.1 show that on theoretical grounds CG and dynamic relaxation have the same convergence speed. However, the estimate for CG is not tight. Moreover, the linear analysis does not necessarily extend to nonlinear problems. In order to assess the speed of both algorithms in practice, their convergence in terms of computational cost has to be measured when applied in a practical situation. In this section we will discuss the experimental setting and how convergence and computational cost are measured.

The test object is a horizontal cylinder of very soft material, fixed on one end. At the start of the experiment, the gravity force is applied, and the object moves to a new equilibrium state. We measure how quickly it reaches that state. Material parameters and constants are in Table 4.1. These parameters are in the same order of magnitude as a very soft tissue [7, 57]. The undeformed configuration of the cylinder is shown in Figure 4.2. Cantilever beams of soft material easily lead to large deformations, so they test the performance on nonlinear problems. Moreover elongated structures are also present in the human body, for example, in skeletal muscles and tendons.

The object is meshed using a Delaunay tetrahedrization [69] of cylindrical point clouds. We use two meshes, a coarse mesh of 1230 elements and a more fine grained mesh of 9300 elements. Properties of the meshes used are listed in Table 4.2. The meshes are very well-shaped: they have no extreme element sizes, and no extreme angles. It is unlikely that this quality can be maintained for unstructured meshes during online changes. To assess the impact of mesh quality deterioration, we will examine the influence of edge lengths on relaxation

The computational cost of the solution process iteration is measured in *flops*, floating point operations. During the computation, a flop count is maintained. The flop count per tetrahedron was manually determined for every material model. The resulting counts are shown in Table 4.2. During the computation, these numbers are added in a global variable. This flop count is independent of machine, compiler and timer resolution, and is not affected by any overhead of measuring the performance. Multiplications, divisions, sums and differences were counted as one flop, and 1 MFLOP = 10^6 flop. Compared to these counts, the exponential function was measured to take approximately 50 flops. Another instance of the program runs the same experiment with statistics turned off and maximum optimization settings, to determine the speed of the program in flops per second. By combining both numbers, the computional cost

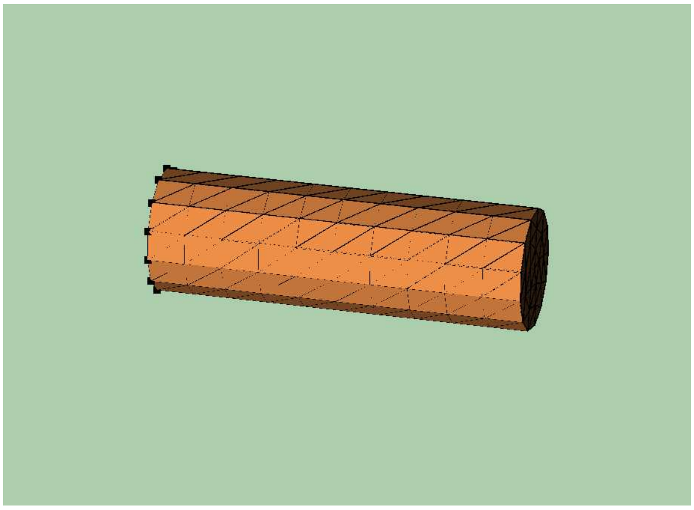


Figure 4.2: The cantilever beam, in undeformed configuration

	small mesh	large mesh
Mesh type	Delaunay	idem
Rod length	0.1m	idem
Rod radius	0.03m	idem
Elements	1230	9300
Nodes	308	1911
Edge lengths	0.05 –0.013m	0.0067 – 0.0025m.
Dihedral angles	$20^\circ - 140^\circ$	idem

Table 4.2: Geometry of the test input

can be expressed in seconds of computation time.

The rate of convergence was determined by comparing the approximation with an "exact" solution, a solution computed with a smaller error tolerance. This solution was obtained in a two step process first, a nonlinear CG iteration was used to find an approximate solution, such that the residual r satisfies $\|r\|_2 \leq 10^{-2}\|f^{ex}\|_2$. Then a truncated Newton-Raphson algorithm (discussed in Section 2.4.3) was used to obtain a solution such that $\|r\|_2 \leq 10^{-8}\|f^{ex}\|_2$ (except for the linear problem, where the tolerance was set at 10^{-12}).

Suppose that the exact solution of the problem is $\hat{u} \in \mathbb{R}^n$, and at some point, the error is $u - \hat{u} = e \in \mathbb{R}^n$. Hyperelastic mechanical problems are energy minimization problems, so we measure the error with the energy difference between the approximation and the 'exact' solution, i.e. the *energy error* $\Pi(u) - \Pi(\hat{u})$. When e is small, then we can rewrite this to

$$\begin{aligned}\Pi(u) - \Pi(\hat{u}) &= \Pi(\hat{u} + e) - \Pi(\hat{u}) \\ &= ((\partial\Pi/\partial u)(\hat{u}), e) + (K(\hat{u})e, e) + \mathcal{O}(\|e\|^3) \\ &= (e, e)_{K(\hat{u})} + \mathcal{O}(\|e\|^3).\end{aligned}$$

The first term of the last expression is an approximation of the energy difference. Since this expression is less susceptible to rounding errors, we will use it for measuring the convergence.

4.3 Hyperelastic compressible materials

Previous work in soft tissue modeling and deformable object simulation shows a variety of different models in use, both for off-line and on-line simulation. Therefore we use a number of different material models, which are discussed in this section. All of these are compressible, isotropic, hyperelastic models. We recall from Equation (2.12) and the discussion surrounding it, that hyperelastic models are defined by an energy density W, which depends on the three invariants ι_1, ι_2 and ι_3 of the Green deformation tensor $\mathbf{C}$. Some forms of anisotropy can also be added to hyperelastic models, by introducing other types of dependencies in W [51, 78].

We recall from (2.15) that the second Piola-Kirchoff stress tensor $\mathbf{S}$ for hyperelastic materials is given by

$$\mathbf{S} = 2\frac{\partial W}{\partial \mathbf{C}},$$

and elastic forces for the nodes of a tetrahedron are given in (2.35): they are represented in the 2-tensor

$$-\mathbf{T} \cdot \mathbf{Z}^{-*} = -\mathbf{F} \cdot \mathbf{S} \cdot \mathbf{Z}^{-*}. \tag{4.3}$$

In this expression $\mathbf{Z}$ is the tensor defined in (2.32). It represents the shape of the tetrahedron.

For Newton-Raphson methods we will also need the derivative of the nodal forces, relative to the tensor $\mathbf{U}$ representing node displacements. The derivative of the nodal forces can be expressed as a 4-tensor, a linear map that takes 2-tensors to 2-tensors.

It can be computed from (4.3) by applying the product rule, leading to the following derivative.

$$\mathbf{H} \mapsto \left(\mathbf{H} \cdot \mathbf{Z}^{-1} \cdot \mathbf{S} \cdot + \mathbf{F} \cdot \left(\frac{\partial \mathbf{S}}{\partial \mathbf{u}} : \mathbf{H}\right)\right) \cdot \mathbf{Z}^{-*}, \qquad \mathbf{H} \in \mathrm{Lin}\,. \tag{4.4}$$

The derivative of $\mathbf{S}$ is given by

$$\frac{\partial \mathbf{S}}{\partial \mathbf{u}} = \frac{\partial \mathbf{S}}{\partial \mathbf{C}} : \frac{\partial \mathbf{C}}{\partial \mathbf{u}},$$

and

$$\frac{\partial \mathbf{C}}{\partial \mathbf{u}} : \mathbf{H} = (\mathbf{H} \cdot \mathbf{Z}^{-1})^* \cdot \mathbf{F} + \mathbf{F}^* \cdot (\mathbf{H} \cdot \mathbf{Z}^{-1}).$$

Equation (4.4) includes $\mathbf{S}$, so when both the forces from (4.3) and their derivative from (4.4) are required, the calculations can be combined. Calculating both is only slightly more expensive than calculating the derivative only.

We assume that the reference configuration of the object is in a stress-free state, so $\mathbf{S} = \mathbf{0}$ when $\mathbf{C} = \mathbf{I}$. The function W represents potential energy, so we arbitrarily set $W = 0$ for $\mathbf{C} = \mathbf{I}$. We introduce the following models.

- St. Venant-Kirchoff material
- St. Venant-Kirchoff material with the linear geometry approximation
- neo-Hookean material
- Veronda-Westmann

The cost of computing an elastic force from the deformation of a tetrahedron varies across these models. The costs are listed in Table 4.3.

Model	Force	Derivative
Linear material/strain	129	129
St. Venant-Kirchoff	235	421
neo-Hookean	277	595
Veronda-Westmann	347	797

Table 4.3: Cost in flops of computing elastic forces and their derivatives in a single tetrahedron, measured by counting operations in the formulas.

For small deformations, all these models reduce to the second model, which allows the computations to be verified using the deformation test of Chapter 3. We express the material parameters for all models using the Lamé constants λ and μ.

4.3.1 St. Venant-Kirchoff elasticity

St. Venant-Kirchoff elasticity addresses the linear geometry approximation. It was used by Zhuang and Canny [101] in a dynamic simulation with non-lumped damping, by Picinbono et al. [78] in a dynamic simulation with lumped mass and damping, and by Debunne et al. [32] in a dynamic simulation with adaptive mesh resolutions.

We recall Equation (2.17) for the St. Venant-Kirchoff model, discussed in Section 2.1.

$$\begin{aligned}
W(\iota_1, \iota_2) &= \frac{1}{2}\left(\left(-\mu - \frac{3\lambda}{2}\right)\iota_1 + \left(\frac{\lambda}{4} + \frac{\mu}{2}\right)\iota_1^2 - \mu\iota_2\right), \\
\mathbf{S} &= \mu(\mathbf{C} - \mathbf{I}) + \frac{\lambda}{2}(\iota_1 - 3)\mathbf{I}, \qquad (4.5) \\
\partial\mathbf{S}/\partial\mathbf{C} : \mathbf{H} &= \frac{\lambda}{2}\,\mathrm{trace}(\mathbf{H})\mathbf{I} + \mu\mathbf{H}.
\end{aligned}$$

The result of applying the St.Venant-Kirchoff model to our test object is shown in Figure 4.3. The energy function does not have an energy term that prevents material inversion. This is reflected in the result: elements are inverted near the attachment point of the rod.

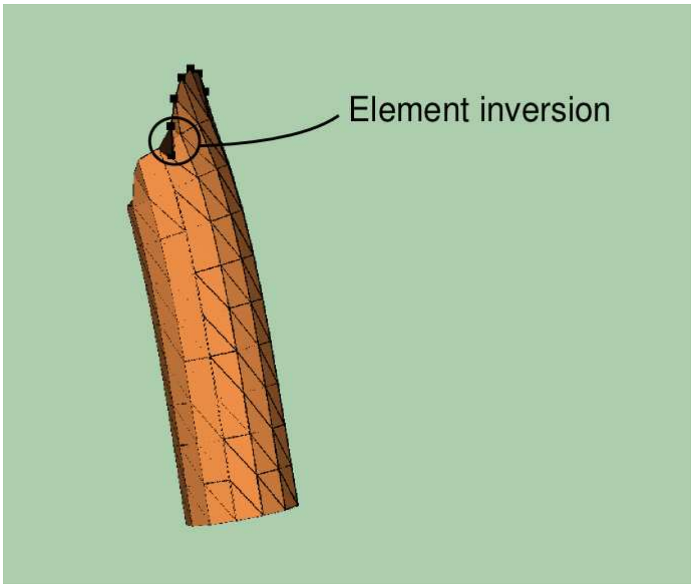


Figure 4.3: St. Venant-Kirchoff elasticity. Elements are inverted where the beam is fixed at the left.

4.3.2 Linear geometry approximation

If this model is combined with the linear geometry approximation, then we obtain linear elasticity, which was discussed earlier in Section 3.1. Linear elasticity was prevalent in early work in surgery simulation [18, 27, 49]. It is also used when high update rates are required. When using the Boundary Element Method [53] or static condensation [18, 36] it is possible to precompute all deformations of an object in advance. With this technique, the high update rates required for haptic interaction can be achieved.

The linear geometry approximation is shown in Figure 4.4. Evidently, the assumption of small deformations does not hold in this situation.

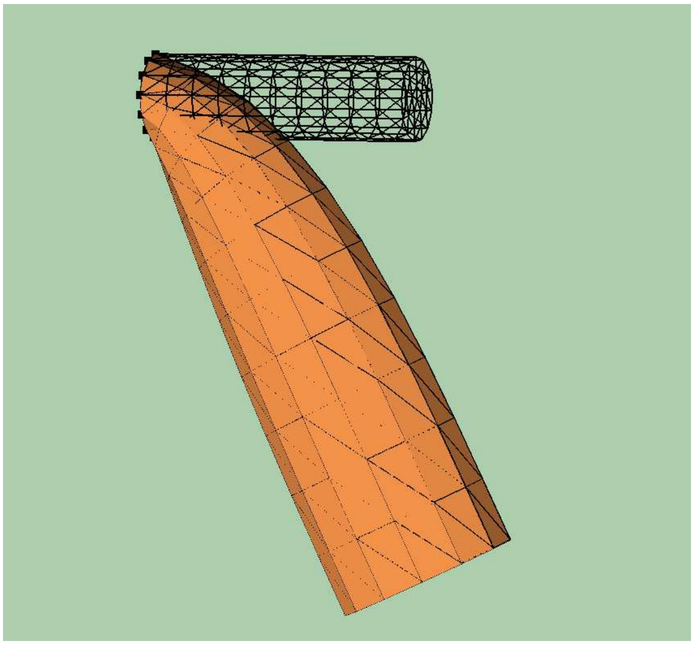


Figure 4.4: The result of applying the linear model to our standard test. The undeformed configuration is shown as a wire frame mesh.

4.3.3 Neo-Hookean elasticity

The compressible neo-Hookean elasticity model is a generalization of the St.Venant-Kirchoff model, and it is used for describing rubbery materials. It has also been used as a material model for interactive deformation by Székely et al. [92] and Wu et al. [100]. The energy density function that we use is given by [65, 102].

$$W(\iota_1, \iota_3) = \frac{1}{2}\left(\mu(\iota_1 - 3) - \mu \ln(\iota_3) + \lambda(\sqrt{\iota_3} - 1)^2\right). \tag{4.6}$$

The stress and its derivative are as follows:

$$\begin{aligned}
\mathbf{S} &= (\mu\mathbf{I} + (-\mu + \lambda(\sqrt{\iota_3} - 1)\sqrt{\iota_3})\mathbf{C}^{-1}) \\
\partial\mathbf{S}/\partial\mathbf{C} : \mathbf{H} &= -(\lambda(\sqrt{\iota_3} - 1)\sqrt{\iota_3} - \mu)\,\mathbf{C}^{-1} \cdot \mathbf{H} \\
&\quad + \frac{\lambda}{2}(2\sqrt{\iota_3} - 1)\sqrt{\iota_3}(\mathbf{C}^{-1} : \mathbf{H})\mathbf{I} \cdot \mathbf{C}^{-1}
\end{aligned}$$

Compression makes ι_3 tend to zero, so the logarithm tends to minus infinity: the material resists inversion, which is visible in the result shown in Figure 4.5.

For small strains, we have $\mathbf{C} \approx \mathbf{I}$, so $\mathbf{C} - \mathbf{I} = \mathcal{O}(\varepsilon)$ for some small $\varepsilon > 0$. In a linear approximation, we have

$$\begin{aligned}
\sqrt{\iota_3} &= 1 + \operatorname{trace}(\mathbf{C} - \mathbf{I})/2 + \mathcal{O}(\varepsilon^2), \\
\mathbf{C}^{-1} &= \mathbf{I} - (\mathbf{C} - \mathbf{I}) + \mathcal{O}(\varepsilon^2).
\end{aligned}$$

For small deformations, this reduces to the stress for linear elasticity in Equation (4.5).

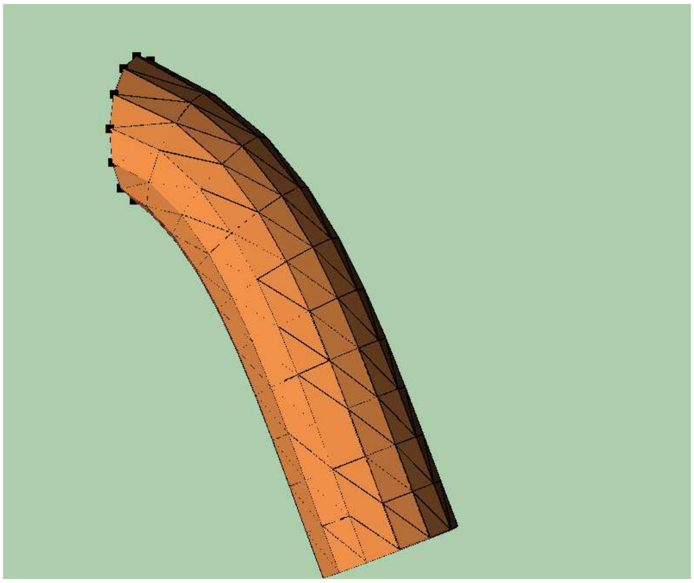


Figure 4.5: Compressible neo-Hookean material.

4.3.4 Veronda-Westmann elasticity

Veronda and Westmann [98] have proposed a three-dimensional constitutive description of soft tissue based on measurements of cat skin. Their work was also discussed in Section 2.6. This model has been used in offline simulations of soft tissue [51, 79]. Veronda and Westmann propose the following energy density:

$$W(\iota_1, \iota_2, \iota_3) = c_1(e^{\gamma(\iota_1 - 3)} - 1) + c_2(\iota_2 - 3) + g(\iota_3).$$

The function g was not specified further. In the compressible case, we should have $g(\iota_3) \to \infty$ if $\iota_3 \to 0$. For small deformations, we have $\iota_1 \approx 3$, so the exponential term can be linearized to $c_1\gamma(\iota_1 - 3)$. The effect of the exponential term is to resist stretching more when strains are large. This is consistent with the stress-strain relations for most types of soft tissue. The parameter γ measures the amount of nonlinearity.

We assume that the reference state is stress free ($\mathbf{S} = 0$ if $\mathbf{C} = \mathbf{I}$). To ensure consistency with the linear model, we require that for $\gamma \to 0$, the exponential term reduces to $2\mu(\iota_1 - 3)$. The following function fits this template:

$$W(\iota_1, \iota_2, \iota_3) = \frac{1}{2}\left(\frac{2\mu}{\gamma}\left(e^{\gamma(\iota_1-3)} - 1\right) - \mu(\iota_2 - 3) + \frac{\lambda}{2}(\iota_3 - 1 - \ln(\iota_3))\right).$$

This energy density leads to the following stress tensor

$$\mathbf{S} = \left(2\mu e^{\gamma(\iota_1-3)} - \mu\iota_1\right)\mathbf{I} + \mu\mathbf{C} + \frac{\lambda}{2}(\iota_3 - 1)\mathbf{C}^{-1}. \tag{4.7}$$

The stress derivative is given by

$$\begin{aligned} \partial\mathbf{S}/\partial\mathbf{C} : \mathbf{H} = \mu\left(2e^{\gamma(\iota_1-3)}\gamma - 1\right)\operatorname{trace}(\mathbf{H})\mathbf{I} + \mu\mathbf{H} \\ + \frac{\lambda}{2}(\iota_3(\mathbf{C}^{-1} : \mathbf{H})\mathbf{I} - (\iota_3 - 1)\mathbf{C}^{-1} \cdot \mathbf{H}) \cdot \mathbf{C}^{-1}. \end{aligned} \tag{4.8}$$

When the nonlinearity γ tends to 0, and we assume small strains ($\mathbf{C} = \mathbf{I} + \mathcal{O}(\varepsilon)$), then (4.7) tends to

$$\begin{aligned} \mu(2 - \operatorname{trace}(\mathbf{C} - \mathbf{I}) - \operatorname{trace}(\mathbf{I}) + \mathbf{C}) + \frac{\lambda}{2}\operatorname{trace}(\mathbf{C} - \mathbf{I})(\mathbf{I} - (\mathbf{C} - \mathbf{I})) + \mathcal{O}(\varepsilon^2) \\ = \mu(\mathbf{C} - \mathbf{I}) + (\mu + \frac{\lambda}{2})(\operatorname{trace}(\mathbf{C} - \mathbf{I}))\mathbf{I} + \mathcal{O}(\varepsilon^2). \end{aligned}$$

The $\operatorname{trace}(\mathbf{C} - \mathbf{I})$ term, corresponding with volume preservation in the linear model, is not consistent with the linear case. The result of applying Veronda-Westmann to the test object is shown in Figure 4.6. Due to the exponential term, the object resists stretching more, and bends less. This results in a smaller tip deflection than the neo-Hookean material model.

4.4 Relaxation algorithms

The two relaxation algorithms tested are explicit SS22 time-integration with lumped masses and lumped damping, and the nonlinear CG algorithm. In this section we discuss how parameters for the dynamic algorithm were chosen, and how the line search for the CG algorithm was implemented.

4.4.1 Dynamic parameters

The implementation of a dynamic relaxation is straightforward, but running requires η and Δt to be set. In the linear case, we can compute the optimal choice for both parameters. In the nonlinear case, we must resort to a heuristic.

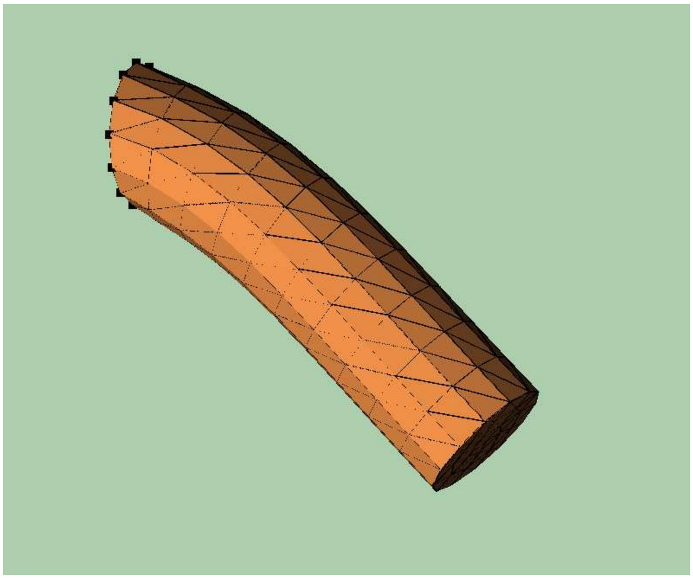


Figure 4.6: Veronda-Westmann material.

The critical time step can be computed exactly for linear elasticity. Nonlinear material can react more strongly to a change in deformation, and requires shorter time steps. Therefore, the critical time step is found by the following empirical procedure. A time step is considered stable if an undamped simulation does not blow up within 50 MFLOPs. A simulation is considered blown up if $\|u\|$ exceeds 10^{15}. An initial time step is estimated using the Courant-Friedrichs-Lewy criterion (2.54), and then it is repeatedly lowered by 15 % until a stable time step is found.

The critical damping was also determined by trial and error. Damping higher than η_{crit} yields smooth and slower convergence, while lower damping yields slower, oscillatory convergence. By manually trying out different η values and selecting the value yielding the fastest convergence, we can find the optimal η, listed in Table 4.4. To verify that η_{crit} is close to optimal, all convergence graphs also show results for damping of $2/3\eta_{\text{crit}}$ and $3/2\eta_{\text{crit}}$.

material	η_{crit}
linear	17
St. Venant-Kirchoff	20
neo-Hookean	21
Veronda Westmann	27

Table 4.4: Damping parameters for the experiments discussed

4.4.2 Line search

The nonlinear CG algorithm is a generalization of the linear CG method. It was discussed in Section 2.4.2. An implementation of the nonlinear CG algorithm requires a line search strategy. Such a strategy improves the energy $\Pi(x)$ of the current solution $x \in \mathbb{R}^n$ by taking a step $\alpha > 0$ in a given direction $d \in \mathbb{R}^n$. The optimal step is given by

$$\min_{\alpha \in \mathbb{R}^+} \Pi(x + \alpha d).$$

For a differentiable Π, the steplength follows from $g(\alpha) = 0$, where

$$g(\alpha) = \left(\frac{\partial \Pi}{\partial x}(x + \alpha d), d \right).$$

This is a one-dimensional equation, which may be solved with a Newton iteration. The Newton iteration was discussed in Section 2.4.3. In this case, the iteration can be defined as follows.

$$\begin{aligned} \alpha_0 &\leftarrow 0 \\ \alpha_{n+1} &\leftarrow \alpha_n - \left(\frac{dg}{d\alpha}(\alpha_n) \right)^{-1} g(\alpha_n), \qquad n \geq 0 \end{aligned}$$

We have

$$\frac{dg}{d\alpha}(\alpha) = (d, K(x + \alpha d))$$

The line search algorithm stops when the if needs more than $j_{\max}$ iterations, or if the update of α is small enough, as measured by a tolerance $\varepsilon_{\text{newton}}$. This leads to the following algorithm.

```
$j \leftarrow 0$
$\alpha_0 \leftarrow 0$
**while** $j < j_{\max}$:
  $r_j \leftarrow -(\partial \Pi / \partial x)(x + \alpha_j d)$
  $s_j \leftarrow -K(x + \alpha_j d) d$
  $\delta_j \leftarrow (r_j, d)/(s_j, d)$
  **if** $j > 0$ and $|\delta_j| < \varepsilon_{\text{newton}} |\delta_0|$:
    exit loop
  $\alpha_{j+1} \leftarrow \alpha_j - \delta_j$
  $j \leftarrow j + 1$
```

The update δ_j is not added to α_j if it is too small. Instead, the corresponding residual r_j is used to update the elastic force vectors in the main loop of the iteration.

In this Newton iteration, the result of the last calculation of $K(x + \alpha_j d)d$ is never used, which is wasteful. Therefore we propose a secant method [102]: the derivative g' in the Newton scheme is replaced by the finite difference approximation

$$\frac{dg(\alpha_n)}{d\alpha} \approx \frac{g(\alpha_n) - g(\alpha_{n-1})}{\alpha_n - \alpha_{n-1}}. \tag{4.9}$$

This leads to the following pseudo code:

```
$j \leftarrow 0$
$\alpha_0 \leftarrow 0$
while $j < j_{max}$:
  $r_j \leftarrow -(\partial\Pi/\partial x)(x + \alpha_j d)$
  $\gamma_j \leftarrow (s_j, d)$
  if $j = 0$:
    $s_j \leftarrow -K(x + \alpha_j d)d$
    $\delta_j \leftarrow \gamma_j/(d, s_j)$
else $\delta_j \leftarrow \gamma_j(\alpha_j - \alpha_{j-1})/(\gamma_j - \gamma_{j-1})$
  if $j > 0$ and $|\delta_j| < \varepsilon_{newton}|\delta_0|$:
    exit loop
  $\alpha_{j+1} \leftarrow \alpha_j - \delta_j$
  $j \leftarrow j + 1$
```

Since no evaluations of $\partial^2\Pi/\partial^2 u$ are left unused, we can expect that this method is more efficient. We may further speed up this algorithm by replacing the evaluation of $K(x)d$ in the first step by the finite difference approximation from (4.9). This introduces a scale-dependent parameter, since α_{-1} must be chosen for the problem at hand. We will refer to this algorithm as the scale-dependent secant algorithm.

4.5 Results

The hyperelastic models discussed were implemented, along with iteration methods for nonlinear CG and SS22 time-integration. This was done in the framework that we wrote for the work in Chapter 3. In addition, a truncated Newton algorithm, as discussed in Subsection 2.4.3, was implemented to compute reference solutions at stricter tolerances.

4.5.1 Tuning CG

The performance of the three line search algorithms (Newton, secant, scale-dependent secant) from the previous section is plotted in Figure 4.7. The scale-dependent secant algorithm is the fastest method. For our test cases, $\alpha_{-1} = -0.001$ was sufficient to obtain convergence. The line search algorithms all use a tolerance parameter ε. Figure 4.8 shows how different settings affect the convergence, and is representative of other models: The iteration converges within a few iterations, so the precise value of ε makes little difference in the convergence behavior.

There are different strategies for determining β in the nonlinear Conjugate Gradient algorithm. Both the Fletcher-Reeves strategy from Equations (2.48) and Polak-Ribière from (2.49) were implemented, but for our test problem there was no difference in performance. The rest of the experiments were conducted with normal secant line search, Polak-Ribière β selection and a large tolerance ($\varepsilon_{newton} = 0.1$) for the line search.

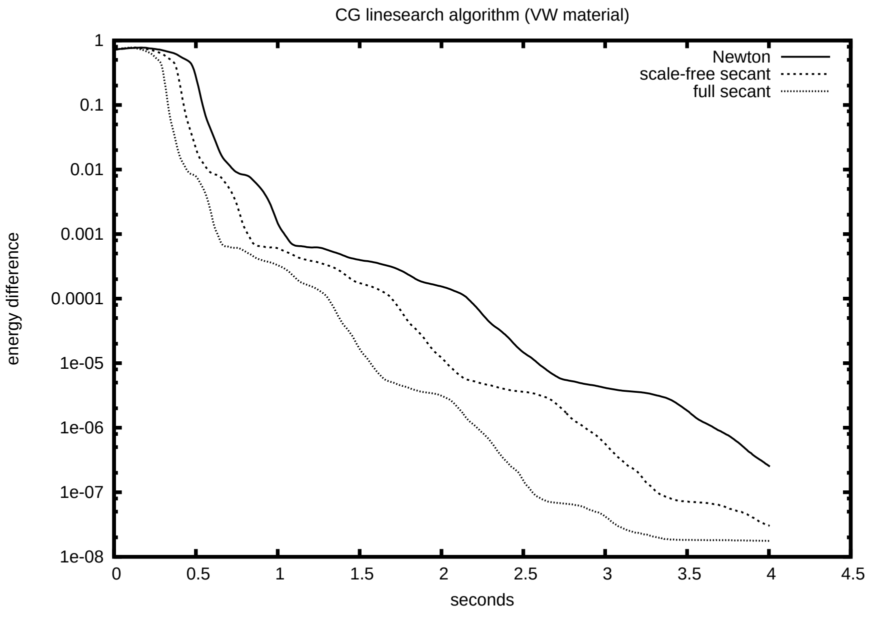


Figure 4.7: The performance of different line searches for the Veronda-Westmann problem.

4.5.2 Performance

By comparing FLOP count and processor time used, we can also estimate the MFLOP per second rate, which indicates how efficiently the CPU is used during computations. These numbers are given in Table 4.5. The baseline for the MFLOP/second rate was a repeated double vector add, coded in C++. For repeated adds of a 1024-double vector, the machine, a 1 Ghz Pentium 3, achieved 246 MFLOP/sec. The programs were compiled with GNU C++ version 3.2, with maximum optimization switched on, and visualization and convergence statistics turned off.

The dominating cost in computation were computations of the elastic forces, taking up 99 to 99.5 % of the operations. The MFLOP rates range between 50 and 90 % of the peak speed, indicating that the implementation performs in the order of magnitude of the machine peak speed.

4.5.3 Influence of mesh size

In Chapter 3, we have demonstrated that large meshes are needed for accurate results. Figure 4.9 compares the convergence of the large mesh and the small mesh from Table 4.2. The larger mesh leads to slower convergence, but static and dynamic are slowed down by the same amount, so all other experiments are performed on the small mesh.

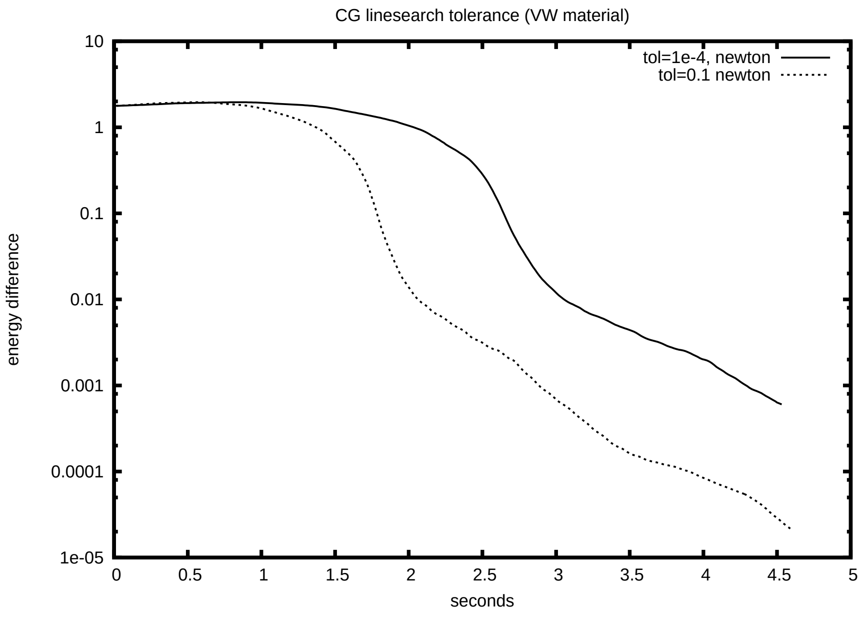
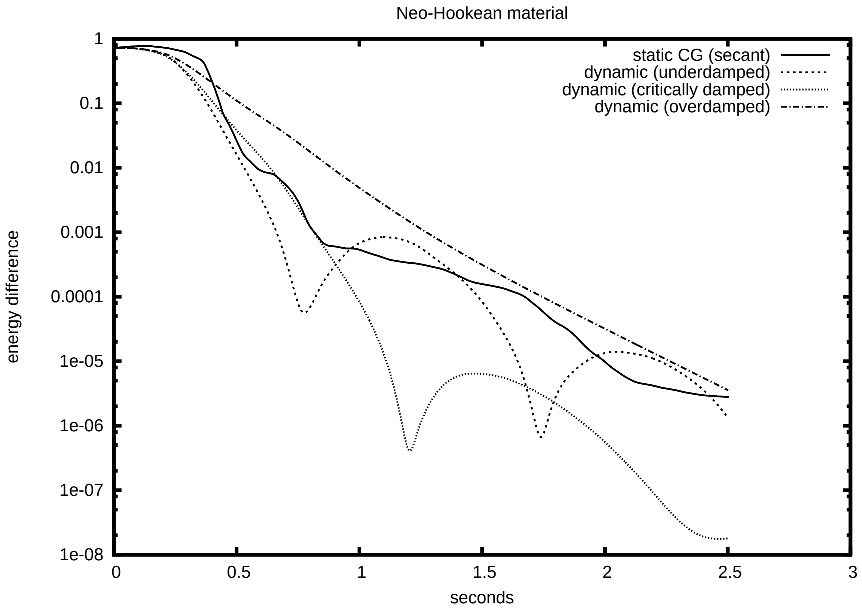


Figure 4.8: Impact of the Newton tolerance in the line search. (Veronda-Westmann problem, Newton line search). The performance is only slightly affected, but a looser tolerance is quicker.

material	relaxation	line search	% peak	iter/sec	MFLOP/iter
Linear elasticity	dynamic		49	628	0.2
	static		56	708	0.2
St Venant Kirchoff	static	Newton	48	71	1.7
	static	Secant	53	113	1.2
	dynamic		62	426	0.4
neo-Hookean	static	Newton	85	109	1.9
	static	Secant	77	144	1.3
	dynamic		54	356	0.4
Veronda-Westmann	static	Newton	90	91	2.4
	static	Secant	80	120	1.7
	dynamic		61	327	0.5
neo-Hookean	static	Newton	73	12	14.2
(large mesh)	static	Secant	66	16	9.9
	dynamic		49	41	2.9

Table 4.5: Machine dependent performance numbers for a PIII/1Ghz machine captured from the first 1.0 seconds that experiments ran. Timings per second are approximate numbers, and vary by a few percent across runs. The peak MFLOP rate is defined to be 246 MFLOP/sec: the performance for repeated double vector add of size 1024.



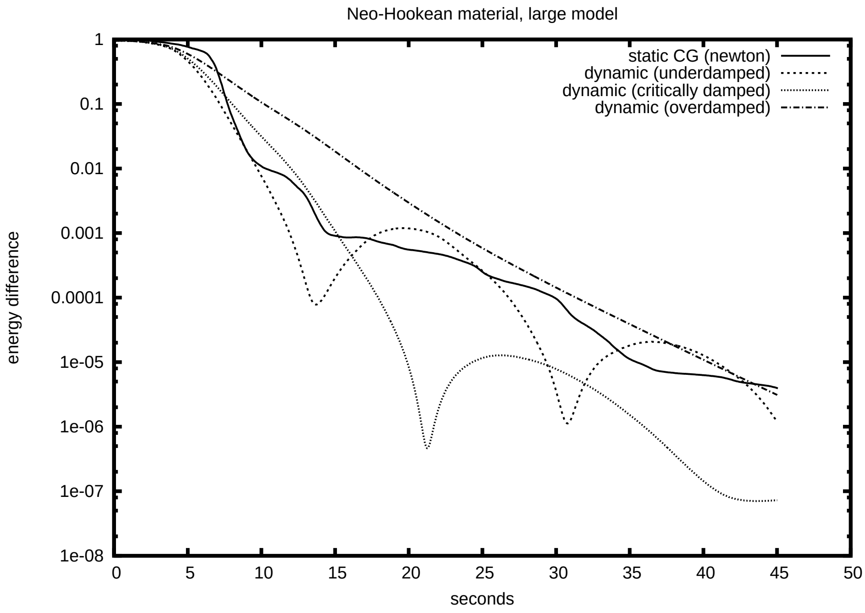


Figure 4.9: Cantilever experiment with neo-Hookean elasticity for different mesh sizes. The small model (top) and the large model (bottom) offer similar convergence.

4.5.4 Linear elasticity

Figure 4.10 shows the convergence of the linear case. In this case, the CG iteration outperforms dynamic relaxation, by approximately a factor 5 to 10. For example, an energy error of less than 10^{-4} takes 0.10 seconds with linear CG, and 0.89 seconds with dynamic relaxation.

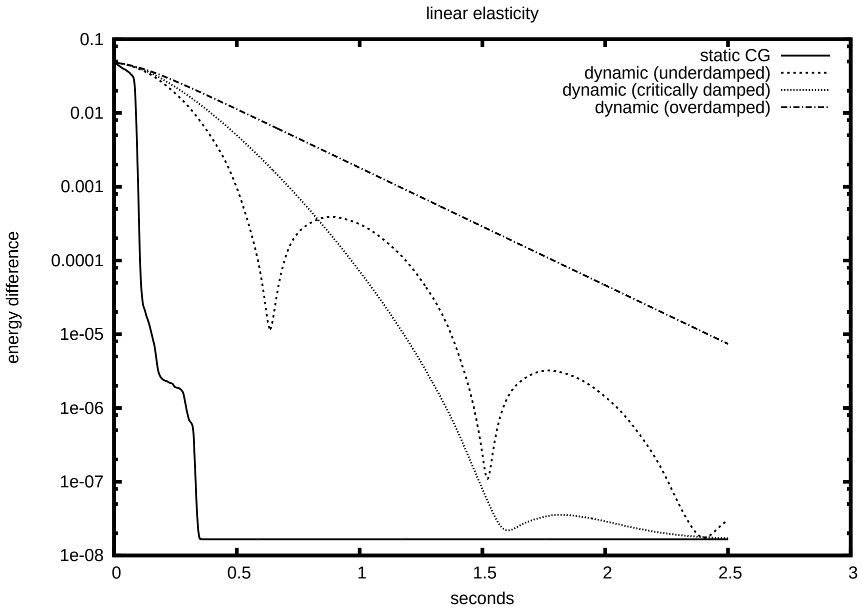


Figure 4.10: Convergence speed for the linear model, energy error

4.5.5 Nonlinear material models

For the nonlinear case, the CG iteration is as quick as a dynamic relaxation with optimal parameters; this is independent of mesh size and material characteristics. This can be seen in Figures 4.11 and 4.9. For St. Venant Kirchoff elasticity in Figure 4.12, dynamic relaxation is at a slight advantage. This seems to be caused by the element inversion. Stiffer material does not lead to element inversion. When the same experiment is repeated with $E = 2 \cdot 10^4$, both algorithms again have roughly the same speed, shown in Figure 4.13.

The lack of physical interpretation of the intermediate results of the CG iteration is evident in Figure 4.14. The static solution itself has a minimal residual force, but during the iteration the residual decreases erratically, and does not even descend monotonously. On the other hand, residual forces decrease smoothly during dynamic relaxation.

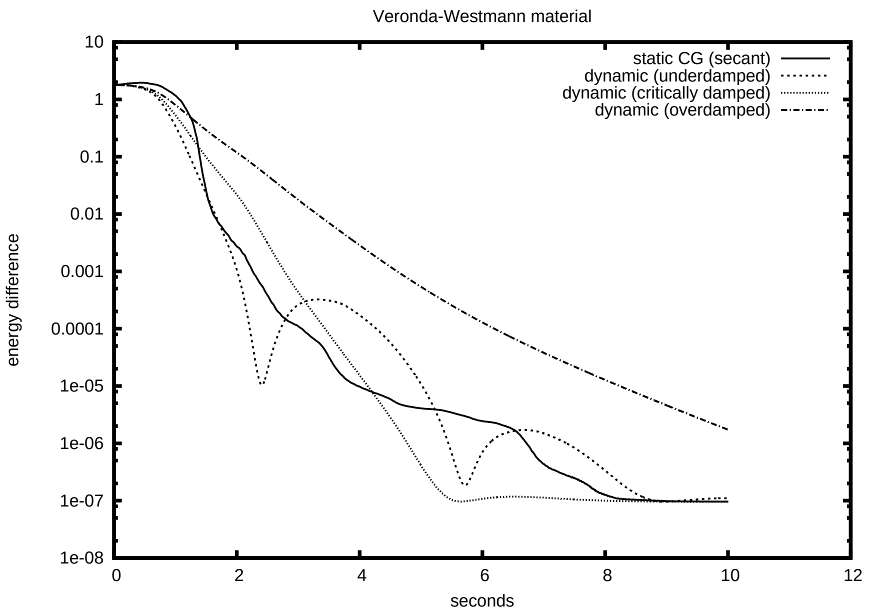


Figure 4.11: Convergence speed for exponential Veronda-Westmann

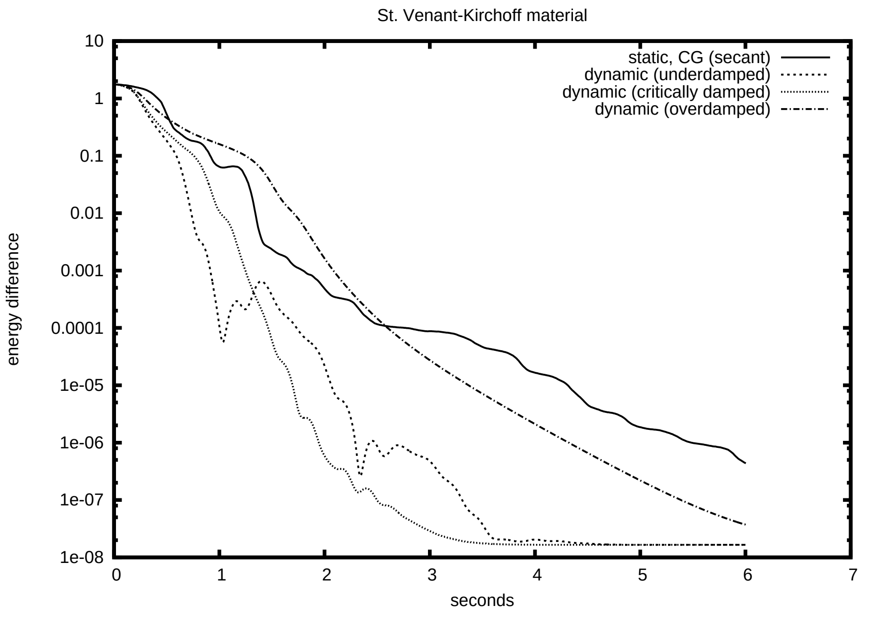


Figure 4.12: Convergence speed for St. Venant-Kirchoff material

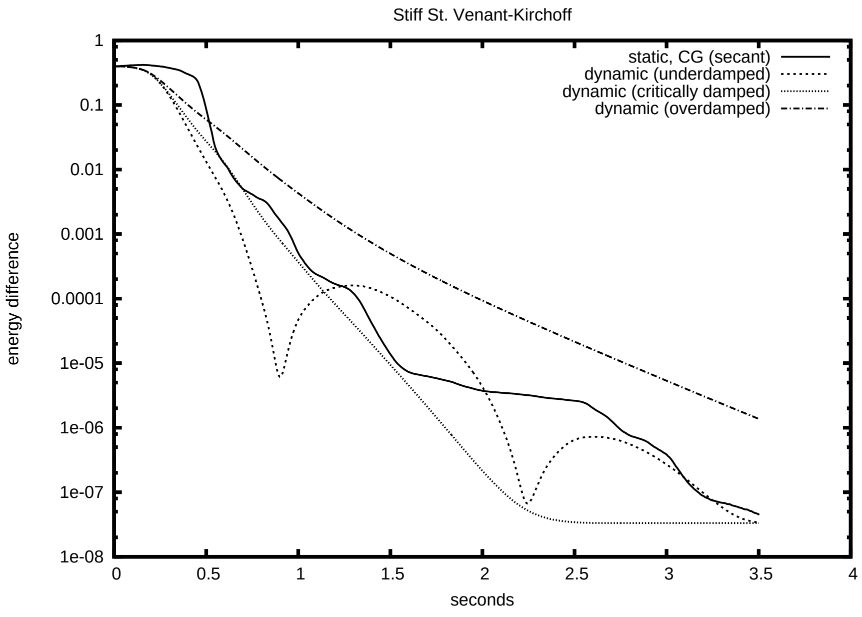


Figure 4.13: St. Venant-Kirchoff material with stiffer material ($E = 2 \cdot 10^4, \eta = 27s^{-1}$).

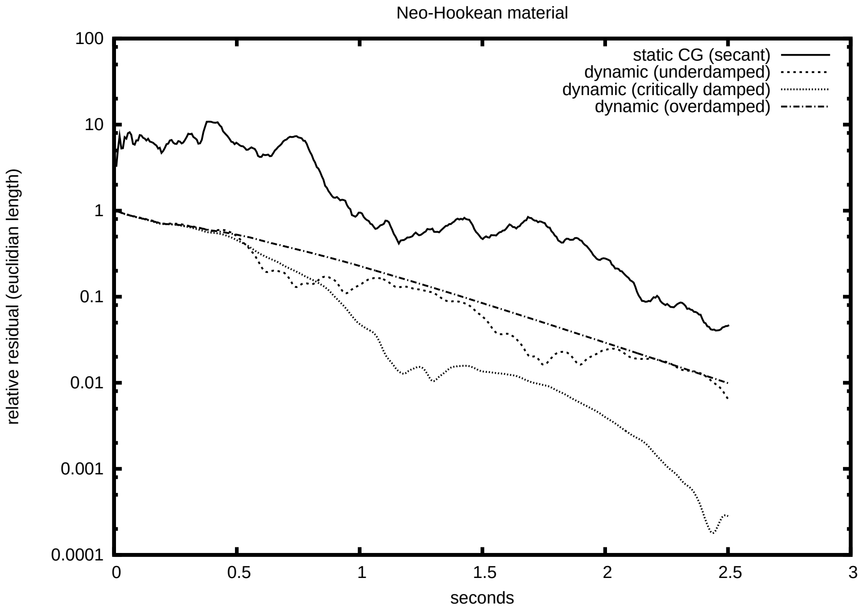
4.5.6 Influence of mesh quality

The influence of mesh quality is demonstrated in Figure 4.15. A single short edge was introduced in the mesh, by inserting a node close to an existing node, using a Delaunay incremental flip algorithm [40]. The effect on linear CG is negligable. This can be attributed to the observation in Subsection 2.4.1 that the magnitude of isolated eigenvalues does not affect the performance of linear CG. In a dynamic setting, the critical time step is inversely proportional to the smallest edge length, hence element shape severely influences the convergence. The influence on the nonlinear CG iteration is also noticeable, but has a much smaller impact. In this experiment, the scale-dependent secant method failed to converge, showing its limited usefulness in practice.

4.6 Discussion

We have compared dynamic relaxation and iterative optimization as methods for finding the steady state of a solution. Dynamic relaxation has linear convergence for the linear problem: the number of iterations is proportional to $\sqrt{\text{cond}_2(M^{-1}K)}$, where M is the lumped mass matrix. For static CG the number of iterations depends on the set of eigenvalues, and in the worst case, it is bounded by $\sqrt{\text{cond}_2 K}$. For uniform meshes and constant mass density, M is almost a multiple of I which suggests that the performance of both is similar.

There are more similarities: dynamic relaxation can be speeded up by increasing



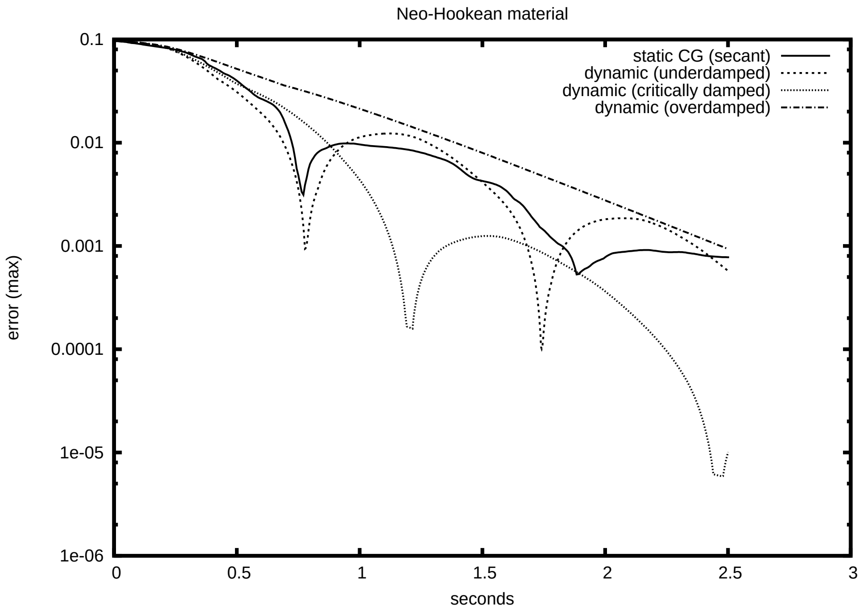


Figure 4.14: Evolution of the relative residual force $\|r\|_2/\|f^{ex}\|_2$ for neo-Hookean elasticity (top). Dynamic relaxation shows a smoother decrease than CG.

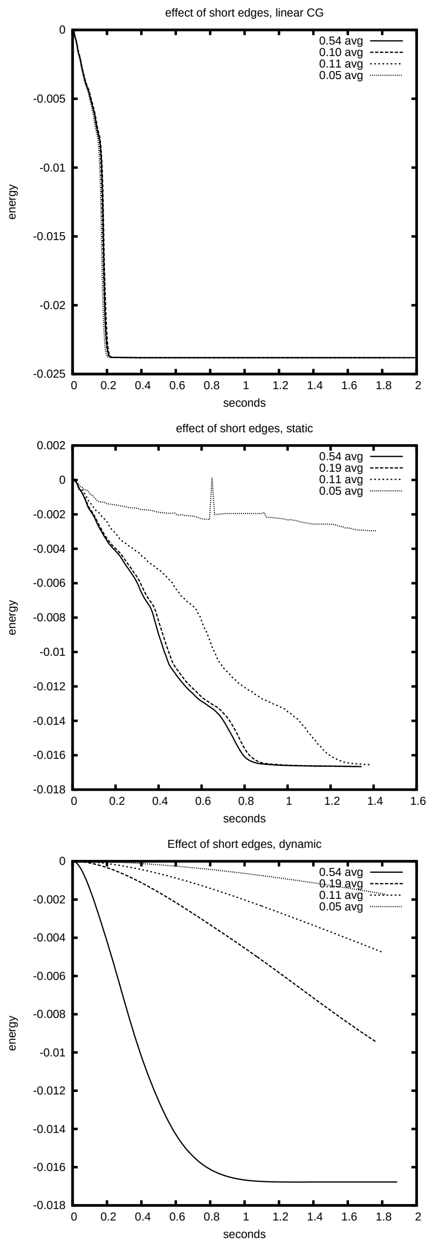


Figure 4.15: Energy decrease over time when a single short edge introduced in the mesh, for the Neo-Hookean material model. Top linear CG. In center the CG based static approach, and on the bottom the dynamic approach. The static approach is hampered less by short edges.

nodal masses of small elements. Analogously, a Conjugate Gradient iteration could be speeded up by using diagonal preconditioning. Parallels between time stepping algorithms for differential equations and optimization based solutions have been pointed out before [50]. In this case, considering the convergence analysis of CG, a direct parallel does not hold.

In the test-case that we have presented, the difference in performance between static CG and dynamic relaxation in the linear case is large: a factor 5 to 10. This could be caused by the symmetry of the test object: this symmetry implies that the stiffness matrix has duplicated eigenvalues. This favors static iteration, as clustered eigenvalues accelerate the convergence of linear CG.

For nonlinear models, the experiments indicate that the performance is comparable. There are qualitative differences between both methods: the physical underpinnings of relaxation ensure a smooth decrease in residual force, and non-conservative forces, such as friction to be added to the model. However, for fast convergence, both Δt and η must be selected experimentally for the situation at hand, and both parameters directly influence convergence speed. Moreover, they also depend on mesh characteristics, and in the nonlinear case the time step also depends on the magnitude of the forces applied. If a simulation includes online mesh changes or nonlinear elasticity, both parameters must be adjusted continuously.

Our test situation is inspired by a soft tissue simulation scenario. However, it has limited use in predicting the applicability of an algorithm in practice. The reason is that the experiment uses gravity, instantaneously switched on, as a test load. First, gravity is a load that is distributed over the entire body, while loads in interactive simulations are typically effected by simulated instruments, which act locally. Such localized loads have more high-frequency components, and this leads to more high-frequency components in the error. On the other hand, quick convergence requires choosing η low. In this case, the high-frequency components of the error will be underdamped and will persist for a long time. This will be noticeable as a "jelly like" vibrations. Secondly, the load is switched on instantaneously while the object is far from its resting position. Interactive simulations run at high update rates, and so loads change slowly between iteration steps. In practice, a deformation computed in the previous iteration step will be a good starting solution for the next step.

For the small mesh and the material parameters selected, the critical time-step is approximately 1 ms and requires an update rate of 1000 Hz. Our machine runs the dynamic simulation at 300 to 600 Hz, depending on the material model. This is close to real-time. Yet, it is not clear whether meaningful simulations can be constructed with meshes as small as these. Moreover, an accurate simulation should simulate mechanical properties of soft tissue, which are known to include viscoelastic effects and incompressibility. Both lead to larger FEM problems. For viscoelasticity, the history of deformation adds extra degrees of freedom. Incompressible problems introduce pressure as an additional variable to the problem, and require more degrees of freedom for the displacement functions to ensure existence of solutions.

In summary, the approach presented in this chapter already reaches the limits of interactive computation, while the material models used do not reflect real tissue behavior. We must address these limits for better simulations. We can distinguish three limits:

- the cost per iteration step,
- the condition number of the problem,
- the relaxation algorithms used.

A cost of a single iteration is determined by the number of elements and the per-element cost of the force computations. This implies that mesh change routines should keep the mesh size low, and only refine meshes where needed. Additional speedups can be gained by using parallel processing, at the cost of synchronization overhead and increased hardware costs.

Badly shaped elements increase the condition number of the problem slowing down the convergence of iterative methods. Explicit dynamic methods are especially susceptible to instabilities caused by small elements, so small elements require the use of small time-steps. A way to cope with such instabilities, is to use adaptive time steps for parts of the mesh [11]: this addresses the problem of instabilities, but introduces some overhead in keeping track of mesh parts that use different time steps. To a lesser degree, badly shaped elements also slow down static techniques. Therefore, it seems worthwhile to prevent such degenerate elements from occuring in the first place.

For FEM discretizations where element sizes are proportional to h, and volumes inversely proportional to the element count, the condition number of the stiffness matrix satisfies $\mathrm{cond}_2(K) \sim \frac{1}{h^2}$ [6]. A larger mesh, needed for a more accurate discretization, not only has more expensive iteration steps, but also requires a larger number of them. This is partly caused by the techniques that we have used: they are naive in the sense that they only use localized displacement/force calculations: applying a force locally to an object causes it to deform globally. For a global deformation, information must travel from the location where force is applied through the entire mesh. Both CG and dynamic relaxation update the location of a node using information from neighboring elements. The convergence of such an iterative method is therefore bound by the diameter of the mesh, since that determines the speed at which deformations propagate across the mesh.

This suggests that more advanced techniques should be used for improving the performance of relaxations. Preconditioning reduces the complexity of CG iterations, but is usually implemented with explicitly stored stiffness matrices. *Multigrid methods* [14] can solve FEM problems with n degrees of freedom in $\mathcal{O}(\log(n))$ iterations, by running iteration steps at multiple resolutions. The method requires that the problem is simulated on meshes with lower resolutions as well. For general unstructured meshes, computing such coarser grids is a complex task in itself [1], which suggests the use of structured meshes.

4.7 Conclusion

We have compared nonlinear Conjugate Gradients and dynamic relaxations for a scenario that is inspired by simulation of soft tissue, and found that nonlinear CG offers similar convergence as dynamic relaxation with optimal parameters. Therefore, both methods should be distinguished by their qualitative differences.

Dynamic relaxation offers a physical interpretation of the iteration results, but requires manual selection of simulation parameters, and the process is vulnerable to instabilities. Static iterative methods are more robust, but their intermediate results have no physical interpretation and are produced at lower frequencies.

Both approaches are affected by mesh quality and mesh size. In other words, better meshes promote faster convergence per iteration step, and smaller meshes offer cheaper iteration steps. Therefore, mesh modification algorithms, such as cuts, should keep mesh size down and mesh quality high. This observation is taken to its consequences in Chapter 5.

It can be argued that the experiment is limited in its scope, and not directly relevant to interactive surgery simulations. This suggests that more carefully setup experiments would give a better appraisals of both techniques. However, given the size of the problem analyzed and the performance numbers in Table 4.5, it seems more worthwhile to direct future research towards techniques to speed up either relaxation technique. Even when using high-quality meshes, both unpreconditioned CG and dynamic relaxation easily strain current computing hardware beyond its limits. It is therefore necessary to use more advanced algorithmic techniques to speed up deformation calculations. This observation will be taken to its consequences in Chapter 6.