Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Element-Wise Mixed Implicit-Explicit Integration for Stable Dynamic Simulation of Deformable Objects

B. Fierz† J. Spillmann‡ M. Harders§

ETH Zurich, Computer Vision Laboratory, Switzerland

Abstract

In order to evolve a deformable object in time, the underlying equations of motion have to be numerically integrated. This is commonly done by employing either an explicit or an implicit integration scheme. While explicit methods are only stable for small time steps, implicit methods are unconditionally stable. In this paper, we present a novel methodology to combine explicit and implicit linear integration approaches, based on element-wise stability considerations. First, we detect the ill-shaped simulation elements which hinder the stable explicit integration of the element nodes as a pre-computation step. These nodes are then simulated implicitly, while the remaining parts of the mesh are explicitly integrated. As a consequence, larger integration time steps than in purely explicit methods are possible, while the computation time per step is smaller than in purely implicit integration. During modifications such as cutting or fracturing, only newly created or modified elements need to be reevaluated, thus making the technique usable in real-time simulations. In addition, our method reduces problems due to numerical dissipation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The physically-based animation of deformable bodies is an important field of research in computer graphics. Dynamic effects such as vibrations and recoil contribute to the realism of the animations. In a quasi-static simulation, the solution depends only on time-invariant boundary conditions. In contrast, in a dynamic simulation, transient states are determined from previously computed steps. This has the unfortunate consequence that numerical errors can lead to injection or dissipation of energy.

More precisely, the underlying mechanical relations lead to a system of second-order partial differential equations. By spatial discretization we obtain second-order ordinary differential equations , which are then transformed into afirst-order equation system with unknown displacements $\mathbf{u}(t)$ and velocities $\dot{\mathbf{u}}(t)$. Then, either an *explicit* or an *implicit* numerical time-integrator is employed to evolve the object. An explicit integration scheme computes the future displacements $\mathbf{u}(t+h)$ and velocities $\dot{\mathbf{u}}(t+h)$ based on a direct computation of the forces $\mathbf{f}(t)$ at time t. For explicit methods, the Courant-Friedrichs-Lewy (CFL) condition must hold in order to ensure convergence, and consequently, only small time steps are possible. Explicit schemes are a good choice when the cost to compute one simulation step has to be minimal, for example, in haptic rendering loops.

In contrast, implicit methods compute the future displacements and velocities by solving the non-linear system of equations, and thus the cost per time step is considerably larger. In addition, some commonly used implicit solvers encounter problems with numerical dissipation.

In this work, we opt for a mixed strategy: Instead of selecting one particular scheme in advance, we combine both explicit and implicit integration methods within the same simulation, hoping to exploit their intrinsic benefits. This idea is not new and widely known by the term IMEX (IMplicit-EXplicit integration). In computer graphics, IMEX has in the past mainly been used to combine linear with non-linear forces in cloth simulation, *i. e.*, to employ an implicit scheme to integrate stiff linear in-plane forces while

† e-mail: bfierz@vision.ee.ethz.ch
‡ e-mail:jonas.spillmann@vision.ee.ethz.ch
§ e-mail: mharders@vision.ee.ethz.ch

SCA 2011, Vancouver, British Columbia, Canada, August 5 – 7, 2011.

DOI: 10.2312/SCA/SCA11/257-266

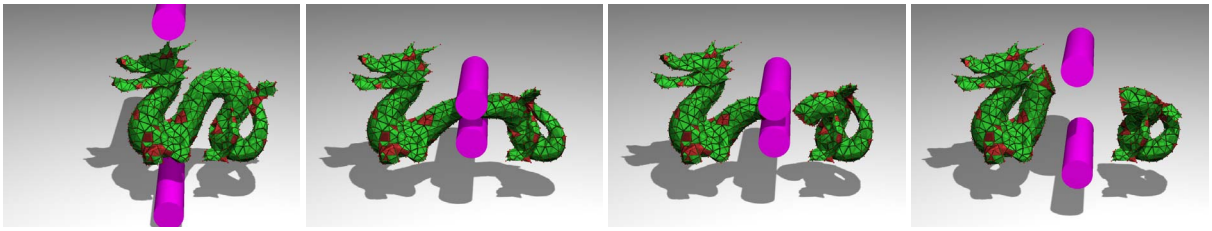


Figure 1: *When mesh cutting is realized through element subdivisions, a cut usually adds ill-shaped elements. In turn, this causes stability problems when using an explicit time-integrator to evolve the object. By integrating the ill-shaped elements implicitly, our approach circumvents these limitations.*

explicitly integrating the non-linear but weaker out-of-plane forces [EEH00]. IMEX schemes are also often used in other areas. For instance, in computational engineering, IMEX schemes were used to handle interface nodes that couple fluid with solid parts [BM78].

In contrast to these works, we propose using an element-wise IMEX scheme for the dynamic simulation of deformable objects in the context of linearized integration schemes. Our approach is based on the observation that there is a relation between the shapes of the simulation elements, and the maximum time step of an explicit scheme in order to avoid divergence [She02]. Recently, [FSAH11] have shown that it is possible to identify ill-shaped elements hindering stable numerical time-integration for a given target time step h_t. Based on this, we first determine all ill-shaped elements of a simulation mesh. It should be noted that this can be done as a pre-computation step when then elemental stiffness matrices are constant. Thereafter, during the simulation, the nodes of the identified elements, as well as any nodes directly adjacent to these, are integrated implicitly, while the remaining ones are evolved with an explicit scheme. In cases of topological changes, it is not necessary to reevaluate all the elements. Instead, only newly created ones and those neighboring changed elements have to be processed. For mesh sizes used in interactive simulations only limited computational effort is required, thus making the approach applicable to real-time modifications such as cutting procedures. Our approach has the following key benefits:

- In contrast to purely explicit integration schemes, we can take larger time steps, resulting in a significant speed-up in real-time scenarios. Further, the meshing process is eased since ill-shaped elements do not need to be strictly avoided. This makes the approach particularly attractive in scenarios involving topological changes, which potentially introduces elements of lower quality.
- Further, our approach comes with a reduction of the numerical dissipation encountered in some integration methods, thus improving the animation of dynamic scenes. Also, since only the ill-shaped elements are implicitly integrated, the cost per time step is much smaller.

The remainder of the paper is structured as follows: In Section 2 we provide a literature overview covering articles addressing similar goals. Section 3 discusses the preliminaries of our method and details explicit and implicit time integration. In Section 4 we present our IMEX method. Section 5 presents experiments which underline the performance of our method. Finally, in Section 6 we summarize the approach and present possible future extensions.

2. Related Work

Having in mind that both explicit and implicit schemes have their own advantages and drawbacks, there exists a wealth of approaches in the literature in order to exploit the benefits, and to overcome or at least dilute the limitations.

One prominent approach is to maximize the quality of the simulation elements in order to allow for larger time steps. Available meshing algorithms based on Delaunay tessellation [She98, ACSYD05] or advancing front methods [Sch97] generate meshes with a good average element quality. Unfortunately, none of these methods can guarantee a certain minimum quality. Thus, it is also not possible to guarantee a minimum time step for explicit integration. [LS07] carve a mesh out of a regular background grid based on a signed distance function. They can guarantee minimum and maximum dihedral angles. However, meshes created with this approach tend to have a higher resolution, resulting in higher simulation costs.

In cases where the mesh topology is static and preprocessing does not matter a different approach is possible. By performing the simulation in frequency space it is possible to omit vibration modes which are higher than the CFL would allow. This approach taken by [PW89, HSO03, BJ05] allows larger time steps with explicit integration. However, the transformation into the frequency space is expensive and not affordable in real-time settings where objects undergo frequent topological changes.

A potential solution which can handle elements of different quality is to use a time adaptive integration scheme. [BG00] describe a system where the nodes are grouped in queues based on their required minimum time step. Each

queue gets executed multiple times in order to meet a certain simulation time step for all nodes. [DDCB01] describe a space and time adaptive simulation system. They use the spatial adaptivity as a level of detail mechanism to focus computational effort where it is needed. Their time adaptivity method handles the different stability requirements resulting from the spatial adaptivity levels. Time adaptive integration methods are frequently used in molecular dynamics in order to separate fast from slow forces [HLW06]. In summary, although adaptive simulations can speed up the simulation massively, they lead to considerably complex codes, especially in the context of topological changes. In contrast, our method does not require special spatial modifications and no costly element management for time adaptivity.

A promising alternative to traditional implicit schemes are geometric integration schemes. They come with an excellent structure- and energy-preservation, as pointed out in [KYT*06]. Recently, a combination of a geometric material model with a fully variational geometric integrator has been proposed in [CPSS10]. While these works are conceptually orthogonal to our approach, we believe that augmenting these approaches with our IMEX scheme could further improve the simulation fidelity.

A number of publications are using IMEX schemes to accelerate mass-spring based cloth animations. [EEH00] separate the forces into linear and non-linear parts. The former are integrated explicitly and the latter implicitly. [BMF03] use an explicit scheme for the forces which are independent of the velocities and an implicit scheme for forces dependent on the velocity. [BA04] distinguish between different types of springs. While the bending springs are solved explicitly, the ones for shearing are treated implicitly, depending on a quality criterion. In contrast to these works, we propose to select the integration method based on the element quality.

In computational engineering, IMEX schemes are used to treat separate regions of partitioned meshes. In order to simplify parallel computation, [Ram02, KB04] partition a mesh into sub regions. The interface nodes between the sub meshes are integrated explicitly and distributed onto each processor. Each sub mesh is then integrated implicitly in isolation. [BM78] discuss the application of IMEX schemes to fluid-structure interactions, where the high oscillatory solid parts are integrated using implicit and the low oscillatory fluid parts are integrated using explicit schemes. [VB05] use implicit integration for both parts and explicit integration for the coupling terms. We show that a similar strategy can be employed by separating well-shaped from ill-shaped simulation elements.

Element-by-element implicit integrations [HLW83] were introduced to reduce the computation time of implicit solvers. [TL88] use both IMEX scheme and element-by-element implicit integration to reduce computation time. They select the explicit scheme based on a local stability criterion per step. Their method is tailored for fluid dynamics problems.

Recently, new strategies to deal with instabilities in explicit integration have been proposed. The key idea is to evaluate a CFL-like condition on a per-element basis. Any parts of the mesh with high vibrations are then excluded from the explicit integration. In [AFSH10, FSH10], this is accomplished by rigidifying the ill-shaped elements. Unfortunately, enforcing the resulting constraints with a manifold projection method turned out to be computationally too costly. Later, [FSAH11] have proposed to employ a shape-matching paradigm to simulate the ill-shaped regions. This improved the performance and avoided artificial stiffening. Still, both methods yield physically implausible results if larger parts of the mesh are ill-shaped. Additionally, shape-matching is known to have difficulties in handling static nodes. In this work we extend their approach. We rely on the same technique to detect the ill-shaped elements. However, in contrast to their work, we use an implicit solver to handle identified elements. This yields consistent results, even if large parts of the mesh are excluded from the explicit integration.

3. Preliminaries

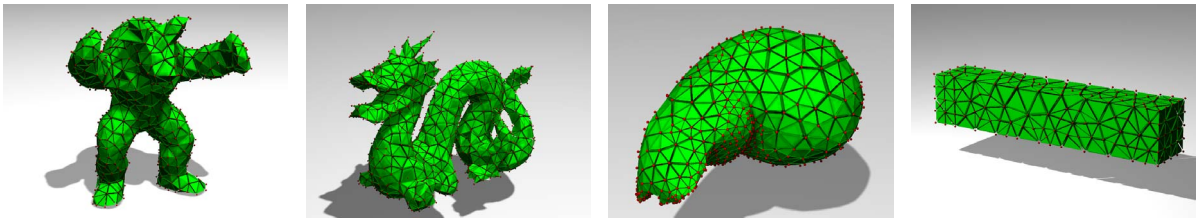
The equations of motion of a deformable object with nodal displacements $\mathbf{u}$ are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}, \tag{1}$$

where $\mathbf{M}, \mathbf{K}$ are the body's mass and stiffness matrices, $\ddot{\mathbf{u}}$ the nodal accelerations, and $\mathbf{f}$ the external forces such as gravity and user interactions, respectively. Note that physical damping is not considered. How these matrices are obtained is not important for our approach, as long as the following holds:

- The mass matrix $\mathbf{M}$ should be lumped in order to allow efficient explicit integration.
- A local stiffness matrix $\mathbf{K}_i$ should be computable per element.
- The stiffness matrix per element should only depend on the initial positions, which is the case for linearized integration methods. In turn, this enables the detection of the ill-shaped elements as a pre-computation step.

In order to solve (1), either an explicit or an implicit numerical time-integration scheme can be employed. Both approaches have advantages and drawbacks. For our interactive simulations we employ the co-rotational approach presented in [MG04]. They linearize the non-linear stress-strain relationship and use a co-rotational technique to account for large displacements. In this context the explicit solver requires a matrix-vector product, while the implicit method requires the solution of a linear system of equations.



(a) Armadillo (b) Dragon (c) Liver (d) Bar

Figure 2: *Models used to compare the simulation times of explicit, IMEX, and implicit methods for different time steps. The geometric characteristics are given in Table 1.*

3.1. Explicit integration

If we apply an explicit scheme to solve (1), the unknown future velocities and displacements appear only on the left hand side, thus an efficient direct solution is possible. In the past, it has been shown that the semi-implicit (symplectic) Euler scheme, which computes the velocities explicitly and the displacements implicitly, preserves the energy particularly well:

$$\begin{aligned} \mathbf{v}(t+h) &= \mathbf{v}(t)+h\mathbf{M}^{-1}(\mathbf{f}-\mathbf{K}\mathbf{u}(t)) \\ \mathbf{u}(t+h) &= \mathbf{u}(t)+h\mathbf{v}(t+h), \end{aligned} \tag{2}$$

The semi-implicit Euler scheme is, like all explicit integration schemes, only stable for h within a small stability region. This region is characterized by the CFL condition, which relates the vibration modes of the mesh to the time step,

$$h \leq \frac{2}{}. \tag{3}$$

The magnitude of the largest vibration mode $_{max}$ is obtained by taking the square root of the largest eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$. A derivation which relates (2) to (3) can be found in [MHTG05]. Further details are provided in [She02].

3.2. Implicit integration

Implicit integration schemes discretize (1) such that the unknown future displacements $\mathbf{u}(t+h)$ appear on the right hand side,

$$\begin{aligned} \mathbf{v}(t+h) &= \mathbf{v}(t)+h\mathbf{M}^{-1}(\mathbf{f}-\mathbf{K}\mathbf{u}(t+h)) \\ \mathbf{u}(t+h) &= \mathbf{u}(t)+h\mathbf{v}(t+h). \end{aligned} \tag{4}$$

This linearized implicit formulation results in a linear system of equations in the unknowns $\mathbf{v}(t+h)$,

$$(\mathbf{M}+h^2\mathbf{K})\mathbf{v}(t+h) = \mathbf{M}\mathbf{v}(t)+h(\mathbf{f}-\mathbf{K}\mathbf{u}(t)). \tag{5}$$

The system matrix $\mathbf{M}+h^2\mathbf{K}$ has to be symmetric and sparse in order to allow for an efficient solution of the resulting system. For linear elements, $\mathbf{K}$ is constant throughout the simulation and can be pre-computed in order to allow for an efficient iterative solution. In contrast, co-rotational and non-linear finite elements come with frequent updates of the system matrix and thus a reduced performance.

In contrast to explicit integration methods, implicit schemes are unconditionally stable, *i. e.* the solution converges for any h. However, this does not imply that the method does not introduce numerical errors. In fact, for linearized systems, energy can dissipate. In the extreme case, a large amount of kinetic energy can be drained [DSB99]. Although this artifact has negative consequences for many animation scenarios, researchers often opt for using implicit schemes. A potential strategy to reduce energy dissipation is to choose a smaller time step or to fall back on specially designed energy-preserving methods. Unfortunately, these solutions can result in an increased computational overhead. For a more immersed discussion, we refer to [HL06]. It should also be noted that in the field of fluid and particle dynamics, a considerable effort has been made to control energy dissipation [MCP*09, SN10].

4. Mixed Implicit-Explicit Integration

In this section, we present an IMEX scheme which selects the nodes to be integrated implicitly or explicitly based on the CFL condition stemming from the element shapes. At first, we group the mesh nodes into a set $\mathcal{I}'$ of m nodes i' needing higher update rates than a target time step h_t, and a set $\mathcal{I}$ of $n-m$ nodes i being integrable with the target time step, where n is the total number of nodes in the mesh. This grouping is based on the CFL condition relating the shape of the elements to the maximum possible time step (3).

4.1. Element identification

In order to keep the implicit set as small as possible the accurate identification of ill-shaped elements is a central element of our method.

The identification is accomplished by using the approach presented in [FSAH11]. For each element in the mesh, we build the local stiffness and mass matrices, $\mathbf{K}^o$ and $\mathbf{M}^o$, of its *1-ring* neighborhood, *i. e.*,of the element nodes and their neighbors. Setting a zero displacement constraint on

Model	# elements	# nodes	Bounding box [cm]	E [kPa]		[kg m^{-3}]
Armadillo	2629	758	$2.6 \times 3.1 \times 2.4$	30	0.3	1
Dragon	4163	1270	$20 \times 14 \times 9$	30	0.3	1
Liver	2749	825	$19 \times 15 \times 15$	30	0.3	1
Bar	2323	592	$4 \times 4 \times 20$	30	0.3	1

Table 1: *Material and mesh parameters for the models used in the experiments.*

the boundary, we extract the square submatrix of $\mathbf{M}^{o\,-1}\mathbf{K}^o$ corresponding to the nodes of the examined element, *i. e.* a 12×12 matrix for a linear tetrahedron. The largest eigenvalue of this submatrix is determined via Jacobi iterations. The identification yields the set of ill-shaped elements. The nodes of these elements are added to the implicit set $\mathcal{I}'$. In contrast to other methods, this metric has the advantage that it is solely based on the elemental mass and stiffness matrices. Further, in contrast to global methods such as described in [She02], it allows us to specifically identify single elements.

In our application the identification of ill-shaped elements can be done a priori. We only need to reevaluate those elements during runtime, which change their rest shape or connectivity. Typically, this occurs after mesh modifications such as cutting, tearing, involves only a few elements, and thus does not impact the performance significantly. Again note that we assume an approach with constant elemental stiffness matrices.

The accuracy of the method to identify critical elements with little false positives is crucial, since the runtime costs of the implicit part scales with the number of nodes in the implicit set $\mathcal{I}'$. [BM78] prove that such a combination of explicitly and implicitly integrated nodes within the same mesh is stable as long as the highest oscillation frequency of the explicit partition fulfills the CFL criteria. This is true using the identification method from [FSAH11].

4.2. A matrix-free IMEX solver

The main part of our approach is the time evolution of the mesh nodes grouped into an explicitly integrated nodes $\mathcal{I}$ and implicitly integrated nodes $\mathcal{I}'$. First, we perform an explicit integration step for the nodes $i \in \mathcal{I}$. This is carried out in a node-wise fashion, *i. e.*,

$$\begin{aligned} \mathbf{v}_i(t+h) &= \mathbf{v}_i(t) + h\frac{1}{m_i}\left(\mathbf{f}_i - \mathbf{K}_i\mathbf{u}(t)\right) \\ \mathbf{u}_i(t+h) &= \mathbf{u}_i(t) + h\mathbf{v}_i(t+h) \end{aligned} \tag{6}$$

where $\mathbf{K}_i \in \mathbb{R}^{3\times 3n}$ is the nodal stiffness matrix assembled from the contributions of the adjacent elements, and m_i is the nodal mass. The first step results in future displacements $\mathbf{u}_i(t+h)$ of all nodes which are adjacent to *only* well-shaped elements.

In order to update the implicitly-integrated nodes $i' \in \mathcal{I}'$,

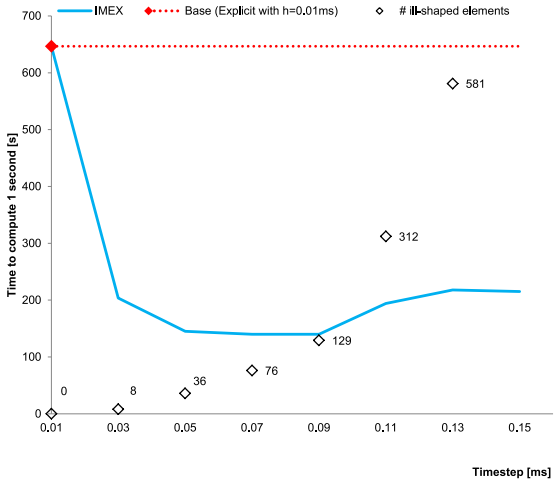
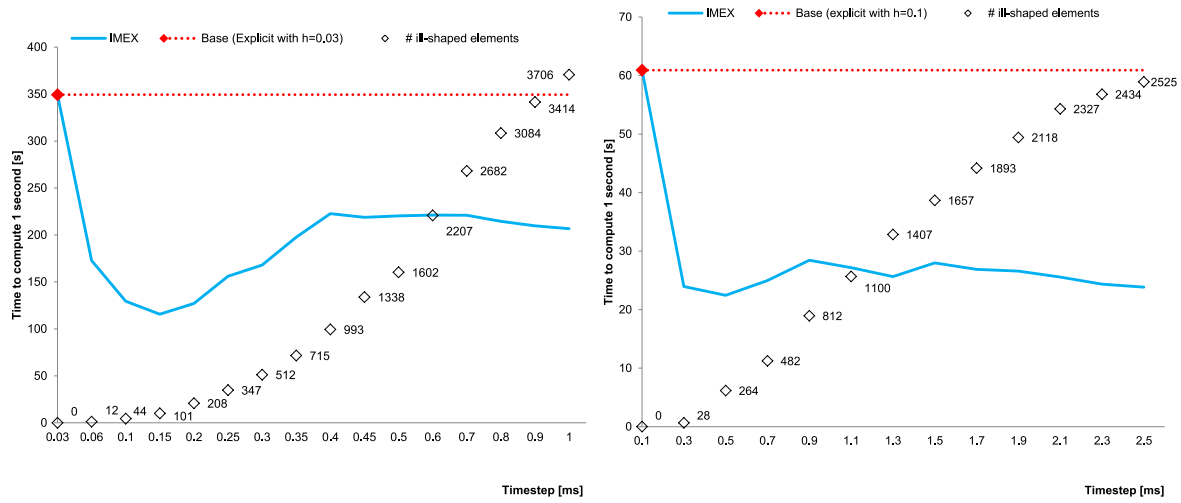


Figure 3: *Performance graph (blue) of one simulated second of the vibrating armadillo (see Figure 2(a)) with different time steps. The red line is the performance of an explicit method running at a time step of* 0.01 *ms. The measurements reveal that for any time step $h > 0.01$ ms, our method outperforms the explicit solver, with a maximum speed-up of 4.6. The diamonds show the number of detected ill-shaped elements.*

we consider a global system where the previously computed future displacements of the explicit nodes are conceptually treated as boundary conditions, as proposed in [BYM79]. Given the CFL condition the solution of the explicit integration is stable and does not need to be changed for this time step. The computed positions are used as the future positions in the implicit step. That is, we solve the modified system

$$(\mathbf{M}' + h^2\mathbf{K}')\mathbf{v}'(t+h) = \mathbf{M}'\mathbf{v}'(t) + h\mathbf{f}' - h\mathbf{K}'\mathbf{u}'(t) - h\mathbf{f}^{\text{impl}} \tag{7}$$

for the future velocities $\mathbf{v}'(t+h) \in \mathbb{R}^{3m}$, where the $3m \times 3m$ matrices $\mathbf{M}'$ and $\mathbf{K}'$ are composed from the contributions of the implicit nodes, and likewise for the vectors $\mathbf{v}'(t), \mathbf{u}'(t)$ and $\mathbf{f}'$ of size $3m$. Further, the vector $\mathbf{f}^{\text{impl}}$ is constructed from the nodal forces $\mathbf{f}_i^{\text{impl}} = \mathbf{K}_i\mathbf{u}(t+h)$ of the explicitly integrated *interface nodes*, *i. e.*, those explicitly integrated nodes $i \in \mathcal{I}$ being adjacent to an implicitly integrated node $i' \in \mathcal{I}'$. $\mathbf{f}^{\text{impl}}$ effectively encodes the aforementioned boundary conditions. Of note is that the system (7) is of size $3m \times 3m$, compared to a full implicit system of size $3n \times 3n$, which



(a) Dragon (b) Liver

Figure 4: *Computation time for one simulated second for the Dragon and the Liver model at different time steps. Our method outperforms the explicit solver for any time step* $h > 0.03$ *ms, and* $h > 0.1$ *ms, respectively. The maximum speed-ups are* 3.3, *and* 2.7, *respectively.*

depending on the number of nodes in $\mathcal{I}'$ can be considerably smaller.

In order to solve (7), we use a conjugate gradient (CG) method implemented following [She94]. Instead of precomputing the matrix for the solver beforehand, we compute the matrix vector products on the fly by summing the element-wise contributions to the result. Thus, each step of the CG algorithm can be formulated in terms of nodal and elemental updates, and the result vector is accumulated accordingly to account for the global solution. In turn, this has the advantage that a large data structure containing the global connectivity information is not necessary. Instead, we can formulate the CG algorithm in terms of elemental and nodal connectivity information blocks, and thus greatly improve the efficiency in a massive parallel computation environment. Additionally, any changes occurring to the structure and connectivity of the mesh does not require any expensive recomputation of a global matrix layout.

5. Results

In this section, we compare the performance of our IMEX formulation to fully implicit and explicit algorithms. We further show applications of our method in the context of interactive cutting. Deformations were computed using a linear co-rotational finite element method [MG04]. The timings were obtained on a single core of an Intel Core2 Quad Q9400 CPU running at 2.66 GHz.

5.1. Performance

The main benefit of our method is that it combines the advantages of both explicit and implicit schemes. In this section, we compare the performance of our approach to both purely explicit and implicit schemes. We also illustrate the relation between the time step of the numerical integration, and the numerical dissipation in implicit schemes.

Of note is that our method only provides a benefit within a certain region of time step values. For time steps smaller than a minimum time step $h_{\min}$, all simulation nodes can be integrated explicitly, thus our method corresponds to an explicit method. For time steps larger than a maximum time step $h_{\max}$, all nodes must be simulated implicitly, thus our method corresponds to an implicit method. The values $h_{\min}$ and $h_{\max}$ cannot be obtained analytically, but instead depend on the geometry of the mesh, and on the parameters of the modeled material. As a rule of thumb, inhomogeneous meshes where elements differ significantly in shape and size will result in a larger range $[h_{\min}, h_{\max}]$, while homogeneous meshes without ill-shaped elements will result in a very narrow range.

For a given time step $h \in [h_{\min}, h_{\max}]$, the element identification method detects the set of nodes $i' \in \mathcal{I}'$ which cannot be integrated stably with h, thus those nodes are integrated implicitly. As a consequence, the performance of our method for $h \in [h_{\min}, h_{\max}]$ depends on both the number and the connectivity of the nodes in $\mathcal{I}'$, but it is commonly better than the performance of explicit solvers. We note that the timings do not include the time to identify the ill-shaped elements, because this is commonly done as a pre-processing step. Re-

sults concerning the online identification of elements will be provided below in Section 5.2. Timings show that the identification test takes about 0.013 ms per element.

Comparison with explicit methods In order to compare our method to explicit solvers, we perform a series of experiments. Elastic objects are first deformed, while one part is subject to boundary displacement constraints. The loads are then removed and the velocities are set to zero, thus causing objects to vibrate about their resting state. The simulation for all integration methods start from the same deformed state and thus with the same deformation energy. Note again that we do not include physical damping in the tests. We first determine the minimum time step $h_{\min}$ at which all nodes can be integrated explicitly. Then, the performance of a semi-implicit Euler method running at a time step of $h_{\min}$ is used as basis for comparison.

We stage this experiment for a number of models shown in Figure 2 (Armadillo, Dragon, Liver, and Bar). The model parameters are summarized in Table 1. The results of the experiment for the Armadillo model are shown in Figure 3. The time to compute one simulation second is plotted for different time steps $h > h_{\min} = 0.01$ ms. Also, the number of elements classified as ill-shaped for a selected time step are provided. Note that for the purpose of comparison the resulting time of performing a purely explicit integration with $h_{\min}$ is visualized by projecting the value as a line into the plot. This only serves as a reference (the explicit method becomes unstable for $h > h_{\min}$). As can be seen, our method results in a reduction of the computation time by up to a factor of 4.6. For time steps $h > h_{\max} = 0.15$ ms, all nodes have to be simulated implicitly, thus our method corresponds to a purely implicit method.

Similar results have been found for the Dragon model (Figure 4(a)) and the Liver model (Figure 4(b)). We obtain a maximum speed-up of 3.3 and 2.7, respectively. The Bar model (Figure 5) turns out to be a 'worst-case scenario'. The elements in the mesh have similar shapes and sizes. There are no singular, particularly badly shaped elements present in the mesh. As a consequence, the explicit time step $h_{\min} = 0.5$ ms is already comparatively large. Our method only exhibits an improvement over the explicit method within a relatively small region of 0.5 ms$< h <$0.7 ms. For larger time steps 0.7 ms$\leq h <$2 ms, the overhead of solving parts of the mesh with the implicit method outbalances the reduction in the number of steps. Thus, for a mesh with equally shaped elements our method will likely result only in smaller improvements.

Comparison with implicit methods Implicit methods, being unconditionally stable, allow to take considerably larger time steps. However, for larger steps, numerical dissipation can reduce the simulation quality of moving objects. This is particularly true when the absolute displacement dominates the deformation, as, *e. g.*, in a swinging deformable pendulum. To quantify these effects, and to show that our method comes with both a reduced numerical dissipation and a better performance for a range of time steps, we repeat the experiment described above using implicit integration. The results were similar for all objects; as an example case we present the data for the Armadillo model. We employ an implicit Euler method to evolve the object. In Figure 6, we plot the total energy $E_{\text{tot}} = E_{\text{kin}} + E_{\text{def}}$, obtained as the sum of the kinetic energy $E_{\text{kin}} = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}$ and deformation energy $E_{\text{def}} = \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u}$, after one simulated second, for both methods at different time steps. Note that the energy level at the start of the simulation is also plotted as a reference line.

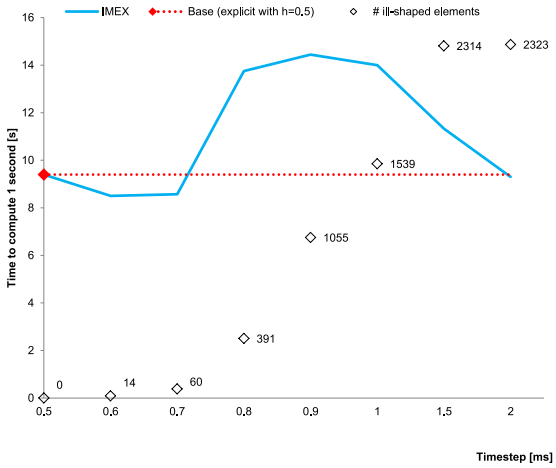


Figure 5: *Computation time for one simulated second for the Bar model at different time steps. The Bar is a 'worst-case' scenario for our method since it is only more efficient for a narrow range of 0.5 ms $< h <$0.7 ms. For larger time steps, the overhead for solving the implicit system outbalances the reduced number of steps.*

A higher value indicates that less energy has drained due to numerical dissipation. The measurements illustrate that for time steps $h < h_{\max} = 0.15$ ms, the dissipation rate of our method is more than three times smaller, as compared to the implicit scheme. For time steps $h \leq 0.01$, our approach corresponds to an explicit method. Note that a small amount of energy is injected as our method behaves like an explicit scheme for the smallest time step. For time steps $h \geq h_{\max}$, our scheme corresponds to an implicit integration. Moreover, in Figure 7 we plot the time to compute one simulation second for different time steps. The measurements indicate that our method is always at least as efficient as the linearized implicit integrator.

5.2. Application

The considerations presented above have been made with a specific type of real-time application in mind, where us-

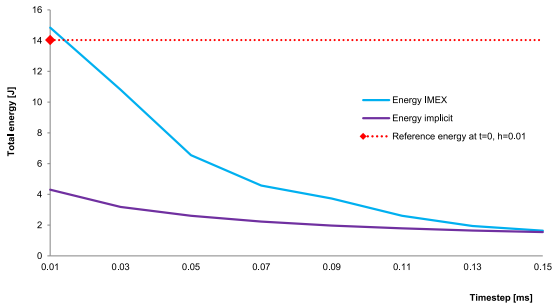


Figure 6: *Remaining total energy of the vibrating Armadillo model after one simulated second. The curves indicate that our method yields a reduced numerical dissipation for time steps $h < h_{\max} = 0.15$ ms. The red line indicates the initial energy. For a time step of $h = 0.01$ ms, our method corresponds to a purely explicit integration method, and thus, a small amount of energy is added (notice that we do not model physical damping).*

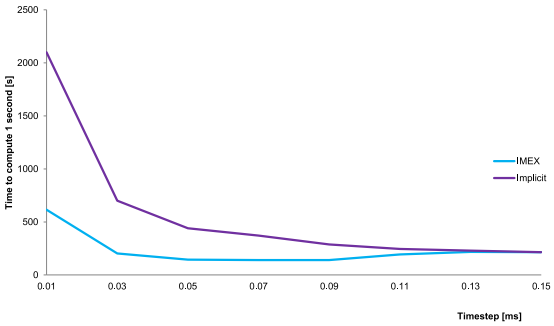


Figure 7: *Performance graph of one simulated second of the armadillo model. The curves indicate that our method requires lower computation time than an implicit method for time steps $h < h_{\max} = 0.15$ ms. For larger time steps, our method corresponds to the implicit method.*

ing small time steps can be desirable; for instance in surgical simulations. Using small time steps eases typical required tasks such as collision handling or haptic rendering. Thus, the results have to be seen in the light of an informed decision to use explicit integration. The advantage of our method becomes apparent when considering cutting procedures in this context. As soon as the simulation elements are subdivided, or nodes are snapped to cutting planes, the element quality deteriorates dramatically. This has the unfortunate consequence that the time step has to be adjusted accordingly in order to satisfy the CFL condition. Our method overcomes this limitation since we can easily detect the ill-shaped elements after the cutting. Then, the time step can be maintained with only a minor computational overhead due to the partial implicit integration. Nevertheless, it should also be noted that our method is always at least as fast as a purely implicit method.

In order to illustrate this application context, we perform an experiment where the Dragon model is non-progressively cut into two pieces (see Figure 1). The cutting path is approximated through straightforward element subdivision. The desired time step of the numerical integration is 0.3 ms. Before the cut, 512 elements have been detected as ill-shaped for the desired time step. Thus, 483 affected nodes have to be integrated implicitly. The average cost for computing one time step is 26 ms. After the cut, the total number of simulation elements increases to 4370 tetrahedra. 122 additional ill-shaped elements appeared after the cut, resulting in a total of 606 nodes to be integrated implicitly. This raises the average costs for computing one time step to 36 ms. The time to compute one physical second is 120 seconds. In contrast to this, had a purely explicit scheme been employed, the time step before the cut should have been set to 0.03 ms to maintain stability. After the cut, the time step has to be further reduced to 0.01 ms due to newly generated ill-shaped elements. In this case, computing one physical second would take up to 880 seconds. Note that the region which is cut initially does not contain any ill-shaped elements. These results demonstrate the advantage of using a mixed implicit-explicit approach to treat any dynamically appearing ill-shaped elements.

Finally, it should be noted that the time to perform the element identification after the cutting is only 0.013 ms per element. The complete online identification for this case was performed in 3.4 ms. Also note that the presented method is very well suited for progressive cutting, since only a small number of elements near the cut tip need to be reevaluated.

6. Conclusion

We have presented a mixed implicit-explicit (IMEX) approach to provide stable dynamic simulations of deformable objects with reduced numerical dissipation. In a preprocessing step we identified those mesh nodes which cannot be simulated stably, because they are near to or part of ill-shaped elements. Then, our method only integrates those nodes implicitly, while the remaining ones are evolved explicitly. As a consequence, we can benefit from both the excellent energy preservation characteristics of the explicit semi-implicit Euler method and the unconditional stability of the implicit Euler method. The results reveal that in many scenarios, our method provides a better performance than either a purely explicit or a purely implicit scheme, also showing a reduced numerical dissipation rate.

One limitation of our method is that the speed-up over the explicit method is not guaranteed. Instead, it depends on the geometry and topology of the underlying simulation mesh. We currently do not determine beforehand whether the employ of our method pays out for a given object. As future work, we investigate into a more formal way to determine

the expected speed-up. Furthermore, we plan to examine the performance of our IMEX solver using higher order integration methods, *e. g.* Runge-Kutta. Further, we want to inspect if it is possible to use our method in conjunction with fully non-linear models, including hyper-elastic materials.

Acknowledgements

We want to thank the reviewers for their valuable input. This research was supported by the EU project PASSPORT FP7 ICT-2007-223894.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational Tetrahedral Meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617 – 625. 2

[AFSH10] AGUINAGA I., FIERZ B., SPILLMANN J., HARDERS M.: Filtering of High Modal Frequencies for Stable Real-time Explicit Integration of Deformable Objects using the Finite Element Method. *Progress in biophysics and molecular biology 103*, 2-3 (2010), 225–235. 3

[BA04] BOXERMAN E., ASCHER U.: Decomposing Cloth. In *Proc. Symposium on Computer Animation* (2004), pp. 153 – 161. 3

[BG00] BIELSER D., GROSS M. H.: Interactive Simulation of Surgical Cuts. In *Proc. Pacific Conference on Computer Graphics and Applications* (2000), pp. 116–125. 2

[BJ05] BARBIC J., JAMES D. L.: Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Transactions on Graphics 24*, 3 (2005), 982. 2

[BM78] BELYTSCHKO T., MULLEN R.: Stability of Explicit-Implicit Mesh Partitions in Time Integration. *International Journal for Numerical Methods in Engineering 12*, 10 (1978), 1575–1586. 2, 3, 5

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of Clothing with Folds and Wrinkles. In *Proc. Symposium on Computer Animation* (2003), pp. 28 – 36. 3

[BYM79] BELYTSCHKO T., YEN H., MULLEN R.: Mixed Methods for Time Integration. *Computer Methods in Applied Mechanics and Engineering 17-18* (1979), 259–275. 5

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A Simple Geometric Model for Elastic Deformations. *ACM Transactions on Graphics 29*, 4 (2010), 1 – 6. 3

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic Real-Time Deformations using Space & Time Adaptive Sampling. In *ACM SIGGRAPH Computer Graphics* (2001), pp. 31 – 36. 3

[DSB99] DESBRUN M., SCHRÖDER P., BARR A.: Interactive Animation of Structured Deformable Objects. In *Proc. Graphics Interface* (1999), pp. 1 – 8. 4

[EEH00] EBERHARDT B., ETZMUSSO., HAUTH M.: Implicit-Explicit Schemes for Fast Animation with Particle Systems. In *Eurographics Computer Animation and Simulation Workshop* (2000), pp. 137–151. 2, 3

[FSAH11] FIERZ B., SPILLMANN J., AGUINAGA I., HARDERS M.: Maintaining Large Time Steps in Explicit Finite Element Simulations using Shape Matching. *IEEE Transactions on Visualization and Computer Graphics* (2011). 2, 3, 4, 5

[FSH10] FIERZ B., SPILLMANN J., HARDERS M.: Stable Explicit Integration of Deformable Objects by Filtering High Modal Frequencies. *Journal of WSCG 18*, 1 - 3 (2010), 81 – 88. 3

[HL06] HAURET P., LE TALLEC P.: Energy-controlling Time Integration Methods for Nonlinear Elastodynamics and Low-velocity Impact. *Computer Methods in Applied Mechanics and Engineering 195*, 37-40 (2006), 4890–4916. 4

[HLW83] HUGHES T. J. R., LEVIT I., WINGET J.: Element-by-Element Implicit Algorithms for Heat Conduction. *Journal of Engineering Mechanics 109*, 2 (1983), 576–585. 3

[HLW06] HAIRER E., LUBICH C., WANNER G.: *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006. 3

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive Deformation Using Modal Analysis with Constraints. In *Proc. Graphics Interface* (2003), pp. 247 – 256. 2

[KB04] KECKEISEN M., BLOCHINGER W.: Parallel Implicit Integration for Cloth Animations on Distributed Memory Architectures. In *Eurographics Symposium on Parallel Graphics and Visualization* (2004). 3

[KYT*06] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, Variational Integrators for Computer Animation. In *Proc. Symposium on Computer Animation* (2006), pp. 43–51. 3

[LS07] LABELLE F., SHEWCHUK J. R.: Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles. *ACM Transactions on Graphics 26*, 3 (2007), 57. 2

[MCP*09] MULLEN P., CRANE K., PAVLOV D., TONG Y., DESBRUN M.: Energy-preserving Integrators for Fluid Animation. *ACM Transactions on Graphics 28*, 3 (2009), 1. 4

[MG04] MÜLLER M., GROSS M.: Interactive Virtual Materials. In *Proc. Graphics Interface* (2004), pp. 239 – 246. 3, 6

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless Deformations Based on Shape Matching. *ACM Transactions on Graphics 24*, 3 (2005), 471 – 478. 4

[PW89] PENTLAND A., WILLIAMS J.: Good Vibrations: Modal Dynamics for Graphics and Animation. *ACM SIGGRAPH Computer Graphics 23*, 3 (1989), 207 – 214. 2

[Ram02] RAMA MOHAN RAO A.: A Parallel Mixed Time Integration Algorithm for Nonlinear Dynamic Analysis. *Advances in Engineering Software 33*, 5 (May 2002), 261–271. 3

[Sch97] SCHÖBERL J.: NETGEN An Advancing Front 2D/3D-Mesh Generator Based on Abstract Rules. *Computing and Visualization in Science 1*, 1 (July 1997), 41–52. 2

[She94] SHEWCHUK J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep., 1994. 6

[She98] SHEWCHUK J. R.: Tetrahedral Mesh Generation by Delaunay Refinement. In *Proc. Symposium on Computational Geometry* (1998), pp. 86 – 95. 2

[She02] SHEWCHUK J. R.: What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proc. International Meshing Roundtable* (2002), pp. 115 – 126. 2, 4, 5

[SN10] SCHAFER N., NEGRUT D.: A Quantitative Assessment of the Potential of Implicit Integration Methods for Molecular Dynamics Simulation. *Journal of Computational and Nonlinear Dynamics 5*, 3 (2010). 4

[TL88] TEZDUYAR T. E., LIOU J.: Element-by-element and Implicit-explicit Finite Element Formulations for Computational Fluid D ynamics. In *International Symposium on Domain Decomposition Methods for Partial Differential Equations* (1988), pp. 281–300. 3

[VB05] VANZUIJLEN A., BIJL H.: Implicit and Explicit Higher Order Time Integration Schemes for Structural Dynamics and Fluid-Structure Interaction Computations. *Computers Structures 83*, 2-3 (2005), 93–105. 3