A Unified Approach for Subspace Simulation of Deformable Bodies in Multiple Domains

Xiaofeng Wu* Rajaditya Mukherjee* Huamin Wang*
The Ohio State University

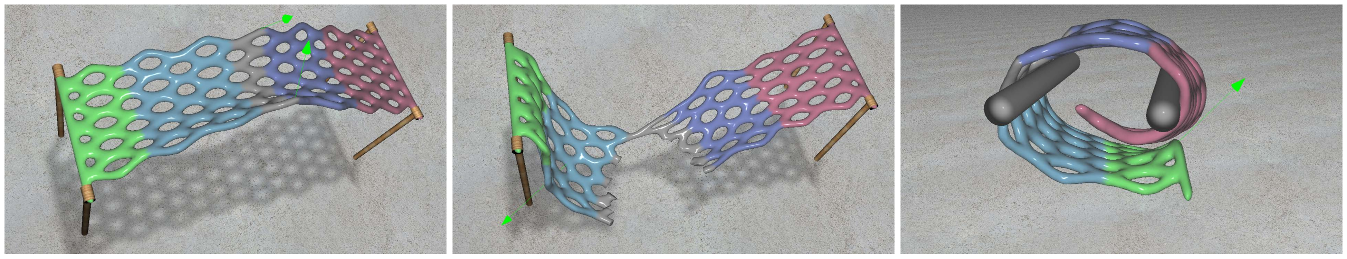

(a) Constrained deformation (b) Deformation after cut (c) Unconstrained deformation

Figure 1: *A hammock example. Under the vertex-based partitioning framework, we present a unified subspace simulation system to animate a multi-domain deformable body in real time, without any coupling constraint. Our experiment shows that this system can effectively avoid the limitations of many existing multi-domain subspace simulators and it can flexibly handle a variety of deformation cases.*

Abstract

Multi-domain subspace simulation can efficiently and conveniently simulate the deformation of a large deformable body, by constraining the deformation of each domain into a different subspace. The key challenge in implementing this method is how to handle the coupling among multiple deformable domains, so that the overall effect is free of gap or locking issues. In this paper, we present a new domain decomposition framework that connects two disjoint domains through coupling elements. Under this framework, we present a unified simulation system that solves subspace deformations and rigid motions of all of the domains by a single linear solve. Since the coupling elements are part of the deformable body, their elastic properties are the same as the rest of the body and our system does not need stiffness parameter tuning. To quickly evaluate the reduced elastic forces and their Jacobian matrices caused by the coupling elements, we further develop two cubature optimization schemes using uniform and non-uniform cubature weights. Our experiment shows that the whole system can efficiently handle large and complex scenes, many of which cannot be easily simulated by previous techniques without limitations.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation.

Keywords: subspace simulation, domain decomposition, finite element method, nonlinear elasticity, cubature approximation.

*e-mail: {wuxi, mukherjr, whmin}@cse.ohio-state.edu

1 Introduction

The fundamental idea behind *subspace deformation*, also known as *dimensional model reduction* or *reduced-order deformation*, is to constrain the deformation of an object into a precomputed subspace. The computational complexity of this method is $O(r^4)$ when using cubic polynomial [Barbič and James 2005], or even $O(r^3)$ when using cubature approximation [An et al. 2008], in which r is the number of deformation modes in the subspace basis. Since r can be significantly smaller than the number of vertices, subspace simulation can be several orders of magnitude faster than full-space simulation. Unfortunately, to handle detailed local deformation of a large object, subspace simulation must use a sufficiently large r, which compromises its efficiency and makes it less attractive.

One solution to this efficiency issue, known as *multi-domain subspace simulation*, is to divide an object into multiple domains and simulate the deformation of each domain in its own subspace. If d is the number of domains and each subspace contains r modes, then the computational cost can be as low as $O(dr^4)$ or even $O(dr^3)$ theoretically, which is much lower than the cost of using dr modes for the whole object. Most existing multi-domain techniques choose to partition mesh elements, i.e., tetrahedra, into disjoint domains. The challenge is how to handle the coupling among these domains, each of which has its own local deformation and rigid motion. If the interface between two domains is small and the domain connectivity contains no loop, the coupling process can be simplified by ignoring the deformation of the interface and attaching one child domain to its parent domain, as Barbič and Zhao [2011] showed. Unfortunately, such an assumption is not always plausible, such as the hammock example shown in Figure 1. To eliminate the gap between two adjacent domains, a straightforward idea is to enforce hard constraints on the interface vertices using Lagrangian multipliers [Huang et al. 2006]. Since these constraints may conflict with the domain subspaces, they can cause the locking issue that suppresses the local deformation effect. Instead of using hard constraints, Kim and James [2011] used spring forces as soft constraints to connect two domains. Their result depends on the choice of the spring stiffness: if the stiffness is higher, the gap becomes smaller but the deformation is more locked; and if the stiffness is lower, the deformation is less locked but the gap becomes larger. Their original implementation also considers subspace local deformation only, assuming that the rigid motion of each domain has

	Deformable Interface	Rigid Motion	Nonlinear Elasticity	Special Limitations
[Huang et al. 2006]	Yes	Yes	No	Mesh-dependent complexity
[Barbič and Zhao 2011]	No	Yes	Yes	Loop-free domains
[Kim and James 2011]	Yes	No	Yes	Stiffness parameter tuning
[Yang et al. 2013]	Yes	Yes	No	Restriction on the number of modes
Our method	Yes	Yes	Yes	See Section 7

Table 1: *The limitations of multi-domain subspace simulation techniques. By using the vertex-based mesh partitioning strategy and integrating rigid and non-rigid motions of all of the domains into a single solve, our multi-domain subspace simulation system effectively avoids many limitations of the existing techniques.*

been given by some earlier simulation. Recently, Yang and colleagues [2013] developed a boundary-aware method to construct linear deformation modes, so that both the gaps and the locking issues can be avoided. This method requires the number of deformation modes in each basis to be at least six times the number of its interfaces. Since these modes are specifically designed, they cannot be easily reused when the domains are assembled into other shapes. The method also has difficulty in handling large nonlinear elastic behaviors, even after using modal warping [Choi and Ko 2005]. In general, existing techniques have various limitations as shown in Table 1. How to flexibly and accurately handle the coupling among multiple domains still remains as an open problem.

In this paper, we propose to partition the vertices, not the elements, into multiple domains. The benefit of this new domain decomposition framework is straightforward: the domains are now indirectly coupled by the elastic forces of the coupling elements connecting the domains, so there is no need to use additional coupling constraints. To develop a practical multi-domain subspace simulation system under this framework, we made the following contributions.

- **A single elastic model.** We show that the same model can be used to handle the elastic deformation of the whole object, including the domains and the coupling elements. We can easily incorporate the elastic forces of the coupling elements, called the *coupling forces*, into the simulation of each domain to achieve the coupling effect, without parameter tuning.
- **A unified multi-domain system.** It is straightforward to separate the rigid motion of a single domain from its subspace deformation and solve them simultaneously as a single dynamical system. But it is not straightforward to do so for multiple domains that are connected by coupling forces. In this work, we formulate a unified system that contains rigid motions and subspace deformations of all of the domains, and we show this is needed to reduce artificial damping.
- **Cubature approximation.** A computationally expensive step in subspace simulation is the evaluation of the reduced elastic forces and their Jacobian matrices. Here we present two cubature approximation schemes to efficiently calculate the reduced forces and their Jacobian matrices. We show that by using different weights for different force components, one of the schemes can achieve better approximation results.

Our experiment reveals the capability of our system in handling large and complex deformable bodies, many of which cannot be easily simulated by previous multi-domain subspace simulation techniques. Despite being more versatile, our system can still run at a high frame rate (from 16.7 to 100 FPS). The system has no strict requirement on the subspace basis, so it is compatible with better basis construction methods and elastic models in the future.

2 Other Related Work

Simulation of deformable objects. The simulation of deformable objects has been an important research topic in computer graphics, since the early work by Terzopoulos and colleagues [1987]. A variety of techniques have been developed to simulate cloth [Baraff and Witkin 1998; Choi and Ko 2002; Chen et al. 2013], elastic rods [Bergou et al. 2008; Umetani et al. 2014], and volumetric deformable bodies [Teran et al. 2003; Müller et al. 2005; Wang et al. 2010]. Although the performance of graphics hardware has been significantly improved in recent years, it is still computationally expensive to simulate a large deformable object, which may take hours or even days. This restricts the simulators from using high-resolution meshes, when they are applied in real-time applications, such as artistic modeling, virtual surgeries, and electronic games.

Subspace simulation of deformable bodies. Originally designed to solve problems in other engineering fields, the subspace simulation approach [Sifakis and Barbic 2012] has gained popularity in computer graphics recently. The early subspace simulator developed by Pentland and Williams [1989] uses modal analysis to build linear vibration modes as the subspace basis. Hauser and colleagues [2003] studied how to integrate collisions and other constraints into modal analysis. Choi and Ko [2005] developed the modal warping method to make linear modes more suitable for handling large deformations, by removing incorrect vertex deformation components caused by linear elasticity. To more accurately model nonlinear elasticity under large deformation, Barbič and James [2005] constructed nonlinear deformation modes from modal derivatives. An and collaborators [2008] proposed to calculate the reduced elastic force in the subspace by cubature approximation. Their method reduces the computational complexity of subspace simulation to $O(r^3)$, in which r is the number of modes in the subspace basis. Harmon and Zorin [2013] presented a basis augmentation scheme to help subspace simulation capture local deformation caused by collision contact. Recently, Hahn and colleagues [2014] used a pose-varying basis to simulate clothing with wrinkle details.

When simulating deformable objects in the subspace, an interesting question is whether collisions can be handled in the subspace as well. Based on the fact that two primitives of the same object cannot be in contact without sufficient deformation, the collision culling method developed by Barbič and James [2010] used subspace coordinates to check the existence of potential self collisions. Teng and colleagues [2014] explored the use of self contact patterns in subspace simulation of articulated bodies, and formulated a pose-space cubature scheme to quickly resolve collisions without checking colliding primitives. In this work, our research is focused on multi-domain subspace simulation and our system can benefit from the use of existing and future subspace collision handling methods.

Subspace fluid simulation. Graphics researchers have also investigated the use of subspace simulation in fluid animation. Treuille and colleagues [2006] showed that both the velocity field and the boundary coupling force can be reduced for subspace simulation. Wicke and collaborators [2009] later improved the usability of this method, by decomposing the volume into domains and enforcing velocity constraints at the domain interface. Given high-resolution fluid simulation data, Kim and Delaney [2013] built a reduced model to efficiently generate animation results under new dynamic conditions. Recently, Ando and colleagues [2015] combined a reduced pressure projection solver with a boundary conforming basis, to speed up the pressure projection step without violating the free-surface boundary condition.

3 Background

In this section, we will present the background knowledge of deformable body simulation in both the full space and the subspace.

Full-space simulation. Given a deformable body with N vertices, we can formulate the equation governing its motion as,

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{f}^{\text{ext}}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^{3N}$ is the stacked vertex displacement vector, $\mathbf{M} \in \mathbb{R}^{3N\times3N}$ and $\mathbf{D} \in \mathbb{R}^{3N\times3N}$ are the mass and damping matrices, and $\mathbf{f}^{\text{int}} \in \mathbb{R}^{3N}$ and $\mathbf{f}^{\text{ext}} \in \mathbb{R}^{3N}$ are the stacked internal and external force vectors. To stably integrate $\mathbf{u}$ over time by Equation 1, many full-space simulators use the implicit Euler method, which requires solving a large sparse linear system with $3N$ degrees of freedom.

Subspace simulation in an inertial frame. The basic idea behind subspace simulation is to constrain the displacement vector $\mathbf{u}$ into a subspace spanned by a set of r representative displacement vectors $\{\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^r\}$, also known as *deformation modes*. These vectors can be assembled into a $3N \times r$ matrix $\mathbf{U}$, as the *basis* for subspace simulation. So we can convert the displacement vector into $\mathbf{u} = \mathbf{U}\mathbf{q}$, in which $\mathbf{q} \in \mathbb{R}^r$ contains the reduced coordinates of $\mathbf{u}$ in the subspace. Following the method presented in [Barbič and James 2005], we use generalized eigenvalue decomposition to construct $\mathbf{U}$, such that $\mathbf{U}$ is mass orthogonal: $\mathbf{U}^{\mathsf{T}}\mathbf{M}\mathbf{U} = \mathbf{I}$, in which $\mathbf{I}$ is the $r \times r$ identity matrix. By combining $\mathbf{u} = \mathbf{U}\mathbf{q}$ and $\mathbf{U}^{\mathsf{T}}\mathbf{M}\mathbf{U} = \mathbf{I}$ with Equation 1, we now obtain the governing equation in the subspace:

$$\ddot{\mathbf{q}} + \mathbf{U}^{\mathsf{T}}\mathbf{D}\mathbf{U}\dot{\mathbf{q}} + \mathbf{U}^{\mathsf{T}}\mathbf{f}^{\text{int}}(\mathbf{U}\mathbf{q}) = \mathbf{U}^{\mathsf{T}}\mathbf{f}^{\text{ext}}. \tag{2}$$

Using implicit time integration, we can formulate Equation 2 into a dense $r \times r$ linear system. Since r can be significantly smaller than $3N$, subspace simulation can be orders of magnitude faster than full-space simulation.

Subspace simulation in a non-inertial frame. As Barbič and Zhao [2011] pointed out, Equation 2 is not suitable for simulating the rigid motion of a deformable object, since incorporating rigid modes into the basis $\mathbf{U}$ will cause $\mathbf{U}$ to be time-dependent and we cannot afford updating $\mathbf{U}$ over time. A more practical way to handle the rigid motion is to simulate it separately from subspace deformation by rigid body dynamics, as did in [Barbič and James 2005; Kaufman et al. 2008]. Because non-rigid deformation is now defined in a non-inertial frame, we must consider four fictitious forces applied at each vertex $\mathbf{x}_i$:

$$\begin{aligned} \mathbf{f}_i^{\text{cor}} &= -2 \cdot m \cdot \boldsymbol{\omega} \times (\mathbf{U}_i \cdot \dot{\mathbf{q}}), & \mathbf{f}_i^{\text{ine}} &= -m \cdot \dot{\mathbf{v}}, \\ \mathbf{f}_i^{\text{eul}} &= -m \cdot \dot{\boldsymbol{\omega}} \times \tilde{\mathbf{x}}_i(\mathbf{q}), & \mathbf{f}_i^{\text{cen}} &= -m \cdot \boldsymbol{\omega} \times \boldsymbol{\omega} \times \tilde{\mathbf{x}}_i(\mathbf{q}), \end{aligned} \tag{3}$$

which are the Coriolis force, the inertial force, the Euler force and the centrifugal force, respectively. Here $\mathbf{v}$ and $\boldsymbol{\omega}$ are the linear and angular velocities of the non-inertial frame, m is the vertex mass, $\mathbf{U}_i$ is the sub-matrix of $\mathbf{U}$ corresponding to vertex $\mathbf{x}_i$, and $\tilde{\mathbf{x}}_i(\mathbf{q})$ is the vertex position in the non-inertial frame. Note that Equation 3 requires all of the vectors to be defined in the non-inertial frame. By summing up all of the four forces and stack them into a vector $\mathbf{f}^{\text{fic}} \in \mathbb{R}^{3N}$, we can formulate subspace simulation of the non-rigid deformation in the non-inertial frame by:

$$\ddot{\mathbf{q}} + \mathbf{U}^{\mathsf{T}}\mathbf{D}\mathbf{U}\dot{\mathbf{q}} + \mathbf{U}^{\mathsf{T}}\mathbf{f}^{\text{int}}(\mathbf{U}\mathbf{q}) = \mathbf{U}^{\mathsf{T}}\left(\mathbf{R}^{\mathsf{T}}\mathbf{f}^{\text{ext}} + \mathbf{f}^{\text{fic}}(\dot{\mathbf{v}}, \boldsymbol{\omega}, \dot{\boldsymbol{\omega}}, \mathbf{q}, \dot{\mathbf{q}})\right), \tag{4}$$

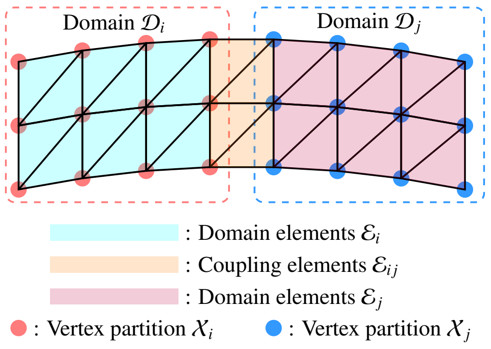

Domain $\mathcal{D}_i$
Domain $\mathcal{D}_j$
: Domain elements $\mathcal{E}_i$
: Coupling elements $\mathcal{E}_{ij}$
: Domain elements $\mathcal{E}_j$
: Vertex partition $\mathcal{X}_i$
: Vertex partition $\mathcal{X}_j$

Figure 2: *A 2D beam. We segment its vertices and elements into two domains, which are connected by coupling elements.*

where $\mathbf{R}^{\mathsf{T}}$ is a time-varying rotation matrix from the global frame to the non-inertial frame. Note that $\mathbf{R}^{\mathsf{T}}$ is not needed in front of $\mathbf{f}^{\text{fic}}$, since it has already been defined in the non-inertial frame. To efficiently project the fictitious forces into the subspace: $\mathbf{U}^{\mathsf{T}}\mathbf{f}^{\text{fic}}$, we adopt the fast sandwich transform method[1] developed by Kim and James [2011]. Basically, this method uses pre-computed matrix blocks to directly calculate $\mathbf{U}^{\mathsf{T}}\mathbf{f}^{\text{fic}}$, based on the fact that these forces can be treated as linear functions of $\mathbf{R}$, $\mathbf{q}$, and $\dot{\mathbf{q}}$.

4 Multi-Domain Subspace Simulation

In this section, we will investigate how to efficiently simulate both rigid and non-rigid motions of a deformable body made of multiple domains. To begin with, we will present our new vertex-based mesh partitioning strategy in Subsection 4.1. We will then discuss how to incorporate coupling forces into subspace simulation and rigid body simulation of each domain in Subsection 4.2. Finally, we will study numerical integration and linear solve in Subsection 4.3.

4.1 Mesh Partitioning

Given a mesh $\mathcal{M} = (\mathcal{E}, \mathcal{X})$, in which $\mathcal{E}$ is the set of its elements and $\mathcal{X}$ is the set of its vertices, we choose to partition $\mathcal{X}$ into d disjoint sets $\{\mathcal{X}_i | i = 1, 2, ..., d\}$ with $\mathcal{X} = \cup_{i=1}^{d} \mathcal{X}_i$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$, for any $i \neq j$. This partitioning can be automatically generated by mesh segmentation algorithms, or manually created from user input. Given $\{\mathcal{X}_i\}$, we can further partition the elements $\mathcal{E}$. Let $\mathcal{E}_i$ be a subset of $\mathcal{E}$, whose elements have vertices coming from $\mathcal{X}_i$ only. We define $\mathcal{D}_i = (\mathcal{E}_i, \mathcal{X}_i)$ as the i-th domain, as shown in Figure 2. If the vertices of an element belong to more than one domain, this element is a coupling element that connects these domains. The elements connecting domain $\mathcal{D}_i$ and domain $\mathcal{D}_j$ form an interface layer $\mathcal{E}_{ij}$. We name $\mathcal{D}_i$ and $\mathcal{D}_j$ as neighbors, if $\mathcal{E}_{ij} \neq \emptyset$. To simplify our discussions later, we assume that an element does not connect more than two domains. It should be straightforward to extend our methods to handle elements connecting three or more domains as well. Table 2 lists some of the symbols to be used in this paper.

4.2 Inter-Domain Coupling

Similar to many existing subspace simulators, our system simulates rigid body dynamics separately from non-rigid local deformation. The main question is: *how can we integrate the coupling force into the simulation of these two components for each domain*?

[1]Similar performance can be achieved by using precomputed quantities too, as Barbič and Zhao [2011] proposed.

Symbol	Definition
$\mathcal{D}_i$	The i^{th} domain of the mesh
$\mathcal{E}_i$	The element set of domain $\mathcal{D}_i$
$\mathcal{X}_i$	The vertex set of domain $\mathcal{D}_i$
$\mathcal{E}_{ij}$	The coupling elements connecting $\mathcal{D}_i$ and $\mathcal{D}_j$
$e_{ij,k}$	The k^{th} element of $\mathcal{E}_{ij}$
$\mathbf{x}^l_{ij,k}$	The l^{th} vertex of element $e_{ij,k}$ in the global frame
$\mathbf{R}_i$	The rotation matrix of domain $\mathcal{D}_i$
$\mathbf{v}_i$, $\boldsymbol{\omega}_i$	Domain $\mathcal{D}_i$'s linear and angular velocities
$\mathbf{U}^l_{ij,k}$	The basis for vertex $V^l_{ij,k}$
$\mathbf{f}^l_{ij,k}$	The coupling force on $\mathbf{V}^l_{ij,k}$ caused by $e_{ij,k}$
$\mathbf{F}^i_{ij}$	The reduced coupling force of $\mathcal{D}_i$ caused by $\mathcal{E}_{ij}$
$\mathbf{f}^i_{ij}$	The net coupling force on domain $\mathcal{D}_i$ caused by $\mathcal{E}_{ij}$
$\boldsymbol{\tau}^i_{ij}$	The net coupling torque on domain $\mathcal{D}_i$ caused by $\mathcal{E}_{ij}$

Table 2: *Symbols. This table summarizes some of the symbols.*

Coupling force in subspace simulation. Our system does not explicitly model the deformation of each coupling element. Instead, the motions of the adjacent domains indirectly cause the deformation, which triggers the coupling force exerted on the domains next. Let $\mathcal{E}_{ij}$ be an interface layer connecting domain $\mathcal{D}_i$ and domain $\mathcal{D}_j$. Its coupling force applied on $\mathcal{D}_i$ can be formulated in the subspace:

$$\mathbf{F}^i_{ij} = \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}^l_{ij,k}\in\mathcal{X}_i} (\mathbf{U}^l_{ij,k})^\mathsf{T}\mathbf{R}_i^\mathsf{T}\mathbf{f}^l_{ij,k}, \tag{5}$$

where $\mathbf{f}^l_{ij,k}$ is the force of element $e_{ij,k}$ applied on its vertex $\mathbf{x}^l_{ij,k}$ in the global frame, $\mathbf{R}_i$ is the rotation matrix of $\mathcal{D}_i$, and $\mathbf{U}^l_{ij,k}$ is the sub-matrix of domain $\mathcal{D}_i$'s basis $\mathbf{U}_i$ that corresponds to vertex $\mathbf{x}^l_{ij,k}$. Intuitively, Equation 5 first rotates the coupling force to the domain's non-inertial local frame, and then reduces it to the domain's subspace. We will use $\mathbf{F}$ and $\mathbf{f}$ to denote the forces in the subspace and the full space, respectively.

By adding all of the coupling forces applied on domain $\mathcal{D}_i$, we can extend Equation 4 to govern the non-rigid deformation of $\mathcal{D}_i$ as,

$$\ddot{\mathbf{q}}_i + \mathbf{U}_i^\mathsf{T}\mathbf{D}_i\mathbf{U}_i\dot{\mathbf{q}}_i + \mathbf{U}_i^\mathsf{T}\mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i) + \sum_{\mathcal{E}_{ij}}\mathbf{F}^i_{ij} = \mathbf{F}_i^{\text{sub}}, \tag{6}$$

in which $\mathbf{F}_i^{\text{sub}}$ is defined as:

$$\mathbf{F}_i^{\text{sub}} = \mathbf{U}_i^\mathsf{T}\left(\mathbf{R}_i^\mathsf{T}\mathbf{f}_i^{\text{ext}} + \mathbf{f}_i^{\text{fic}}(\dot{\mathbf{v}}_i, \boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \mathbf{q}_i, \dot{\mathbf{q}}_i)\right). \tag{7}$$

The terms with subscript i indicate that they belong to domain $\mathcal{D}_i$.

Coupling force in rigid body simulation. For interactive applications, we cannot assume that the rigid motion of a domain has already been given, as in [Kim and James 2011]. So we need to know how the coupling force affects the rigid motion as well. According to rigid body dynamics [Baraff 1997], we have the equations describing the linear and rotational motions of domain $\mathcal{D}_i$:

$$\begin{cases} m_i\dot{\mathbf{v}}_i + \mathbf{D}_{\mathbf{v}_i}\mathbf{v}_i = \mathbf{f}_i^{\text{ext}} - \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}^l_{ij,k}\in\mathcal{X}_i}\mathbf{f}^l_{ij,k} = \mathbf{f}_i^{\text{ext}} - \mathbf{f}^i_{ij}, \\ \mathbf{I}_i\dot{\boldsymbol{\omega}}_i + \mathbf{D}_{\omega_i}\boldsymbol{\omega}_i = \boldsymbol{\tau}_i^{\text{ext}} - \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}^l_{ij,k}\in\mathcal{X}_i}\boldsymbol{\tau}^l_{ij,k} = \boldsymbol{\tau}_i^{\text{ext}} - \boldsymbol{\tau}^i_{ij}, \end{cases} \tag{8}$$

in which m_i and $\mathbf{I}_i$ are domain $\mathcal{D}_i$'s mass and inertia matrix, $\boldsymbol{\tau}_i^{\text{ext}}$ is the total external torque, $\boldsymbol{\tau}^l_{ij,k}$ is the coupling torque caused by the coupling force $\mathbf{f}^l_{ij,k}$, and $\mathbf{D}_{\mathbf{v}_i}$ and $\mathbf{D}_{\omega_i}$ are two damping matrices. For Rayleigh damping, the damping matrices can be computed from m_i, $\mathbf{I}_i$ and the Jacobian matrices. For simplicity, we use $\mathbf{f}^i_{ij}$ and $\boldsymbol{\tau}^i_{ij}$ to denote the net coupling force and torque caused by the elastic forces of the coupling elements in $\mathcal{E}_{ij}$.

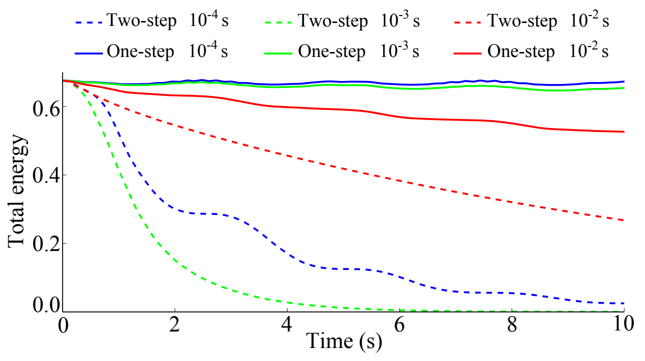


Figure 3: *The energy dissipation process of a 3D beam after initial deformation. This plot visualizes how the total energy dissipates when using different integration schemes and time steps, without damping. It indicates that the one-step integration scheme can preserve the energy well, even when using a large time step.*

By combining Equation 6 with Equation 8, we now obtain a dynamical system that governs both non-rigid and rigid motions of the domains, which are partitioned from the original mesh.

4.3 Numeric Integration

Given the dynamical system presented in Subsection 4.2, we can now study its numerical integration schemes and matrix solvers.

Two-step integration scheme. Since we separate the rigid body motion of each domain from its non-rigid motion, a natural way to simulate a body with multiple domains is to solve rigid and non-rigid motions separately. Specifically, we propose a two-step integration scheme to solve body motions within one time step. First, we assume that a deformed domain is rigid and we use the backward Euler method to integrate Equation 8. After that, we treat the local frame of each domain to be static and we use the backward Euler method to integrate Equation 6.

Unfortunately, our experiment shows that this two-step integration scheme can cause very large artificial damping on the rigid motion, unless a sufficiently small time step is used. Figure 3 shows how the total energy dissipates when we use this scheme to simulate a two-domain beam example. As the time step gets larger, the artificial damping effect becomes more obvious and the energy dissipation rate is significantly increased. We believe this problem is fundamentally due to the fact that the two-step scheme ignores the interplay between the rigid motion and the non-rigid motion within one time step. The magnitude of this ignored term scales with $O(\Delta t^2)$, which gradually vanishes when the time step Δt becomes smaller. To solve this problem, we tested many ideas, such as asynchronous integration and exchanging rigid and non-rigid information multiple times within one time step. None of these ideas is effective, as our experiment shows.

One-step integration scheme. To use larger time steps without artificial damping, we propose to integrate rigid and non-rigid motions of all of the domains into a unified system, and solve it by a single backward Euler step. Since the fictitious forces contain nonlinear terms, we integrate them explicitly and treat the overall external force $\mathbf{F}_i^{\text{sub}}$ applied in the subspace of domain $\mathcal{D}_i$ as constant. Let primed symbols represent the states at the next time $t + \Delta t$ and non-primed symbols represent the states at the current time t. By linearizing the elastic forces in Equation 6, we obtain the equation for simulating non-rigid deformation as:

$$\left(\mathbf{I} + \mathbf{U}_i^{\mathsf{T}}(\Delta t\mathbf{D}_i + \Delta t^2\mathbf{K}_i)\mathbf{U}_i\right)\dot{\mathbf{q}}_i' + \Delta t \sum_{\mathcal{E}_{ij}} \frac{\partial \mathbf{F}_{ij}^i}{\partial \mathbf{Q}_{ij}}\mathbf{Q}_{ij}' \\ = \Delta t\left(\mathbf{F}_i^{\mathrm{sub}} - \mathbf{U}_i^{\mathsf{T}}\mathbf{f}_i^{\mathrm{int}} - \sum_{\mathcal{E}_{ij}} \mathbf{F}_{ij}^i\right) + \dot{\mathbf{q}}_i, \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{r_i \times r_i}$ is the identity matrix, $\mathbf{K}_i \in \mathbb{R}^{r_i \times r_i}$ is domain $\mathcal{D}_i$'s tangent stiffness matrix at time t, and $\mathbf{Q}_{ij} = \left[\dot{\mathbf{q}}_i^{\mathsf{T}}, \mathbf{v}_i^{\mathsf{T}}, \omega_i^{\mathsf{T}}, \dot{\mathbf{q}}_j^{\mathsf{T}}, \mathbf{v}_j^{\mathsf{T}}, \omega_j^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{12+r_i+r_j}$ is the stacked velocity vector of domain $\mathcal{D}_i$ and domain $\mathcal{D}_j$, including the reduced deformation velocities, the linear velocities, and the angular velocities. Here r_i is the number of modes in the subspace of Domain $\mathcal{D}_i$. By applying the same idea to linearize Equation 8, we obtain the linear equation for simulating the rigid motion:

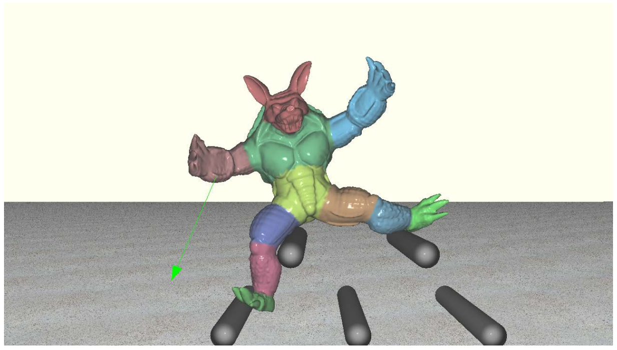
$$\begin{cases} m_i(\mathbf{v}_i' - \mathbf{v}_i) + \Delta t\left(\sum_{\mathcal{E}_{ij}}\left(\frac{\partial \mathbf{f}_{ij}^i}{\partial \mathbf{Q}_{ij}}\mathbf{Q}_{ij}' + \mathbf{f}_{ij}^i\right) + \mathbf{D}_{\mathbf{v}_i}\mathbf{v}_i'\right) = \Delta t\mathbf{f}_i^{\mathrm{ext}}, \\ \mathbf{I}_i(\omega_i' - \omega_i) + \Delta t\left(\sum_{\mathcal{E}_{ij}}\left(\frac{\partial \boldsymbol{\tau}_{ij}^i}{\partial \mathbf{Q}_{ij}}\mathbf{Q}_{ij}' + \boldsymbol{\tau}_{ij}^i\right) + \mathbf{D}_{\omega_i}\omega_i'\right) = \Delta t\boldsymbol{\tau}_i^{\mathrm{ext}}. \end{cases} \tag{10}$$

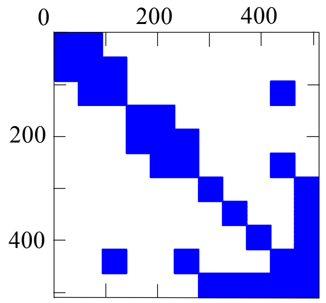
Details about Equation 9 and 10 are in the appendix. Once we formulate the non-rigid deformation and the rigid motion of each domain using the two equations, we combine them into a single linear system and update the domains through a single linear solve. Figure 3 shows that this one-step integration scheme suffers less from the artificial damping issue, even when using a large time step.

Matrix solver. The one-step integration scheme results in a linear system $\mathbf{AQ}' = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{(6d+\sum r_i)\times(6d+\sum r_i)}$ is a block-wise sparse symmetric matrix. Figure 4 shows a partitioned armadillo example and its corresponding block matrix structure, in which the rows and the columns are in the order of $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_d$. Intuitively, each diagonal block represents a domain and two symmetric off-diagonal blocks indicate an interface layer between two domains.

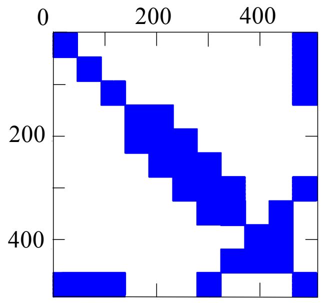
Our experiment shows that the interplay between rigid motions and non-rigid motions can cause $\mathbf{A}$ to become ill-conditioned. This interplay is represented by off-diagonal matrix blocks that exchange information between rigid motions and non-rigid motions. This information exchange process demands more iterations, which can be manifested as a high condition number. For example, the condition number of the octopus example shown in Figure 9 can be as high as 10^8 to 10^9. In contrast, if we choose the two-step integration scheme instead, the matrix is decoupled into a block diagonal matrix and the condition number drops to 10^3 to 10^4 immediately. This explains why the two-step scheme cannot benefit from the use of multiple iterations, because it is equivalent to a block descent solve suffering from the slow convergence issue. This issue exists even if we use other iterative solvers, such as conjugate gradient and generalized minimal residual.

So we choose to use sparse block Cholesky decomposition to solve the linear system directly. To reduce the number of new non-zero blocks introduced by the decomposition process, we re-order the domains so that the decomposition order follows the domain connectivity. In an ideal case when the domain connectivity has no loop, no new block will be introduced and the computational cost of the direct solve is linear to the number of domains, as in [Barbič and Zhao 2011]. A similar idea was also used in [Hecht et al. 2012] for fast Cholesky re-factorization. To apply Cholesky decomposition, the matrix must be symmetric positive definite. Using the invertible FEM method [Irving et al. 2004; Teran et al. 2005], we ensure that the stiffness matrix is symmetric and semi-positive definite. Theoretically, the matrix may still have zero eigenvalues, but they are extremely rare as Teran and colleagues [2005] pointed out and we did not notice any failure in our experiment. Figure 4 visualizes the domains of an armadillo example and its matrix layouts.


(a) An armadillo with unconstrained motion



(b) The matrix layout in our system



(c) The matrix layout after `symamd`

Figure 4: *An armadillo example. We segment this model into 11 domains. As a result, the system matrix has 11 blocks in its rows and columns, as shown in (b). After using our domain re-ordering method, Cholesky decomposition creates a matrix with 33K nonzeros. In contrast, after using the matlab command* `symamd`*, Cholesky decomposition creates a matrix with 39K nonzeros shown in (c).*

5 Cubature Approximation

To improve the system performance, we must know how to quickly evaluate the reduced elastic forces and their Jacobian matrices in the subspace. In general, there are two efficient techniques to accomplish this task: cubic polynomial proposed by Barbič and James [2005], and cubature approximation developed by An and colleagues [2008]. The cubic polynomial method is exact, but it has a high computational cost $O(dr_{\max}^4)$, where $r_{\max}$ is the maximum number of modes used in the subspaces. So we choose cubature approximation instead, whose computational complexity is $O(dr_{\max}^3)$.

To determine the cubature samples and weights for the elastic forces inside of a domain, we follow the non-negative-least-squares-based optimization approach presented in [An et al. 2008]. We will discuss how to generate the training and testing data for cubature optimization in Section 6. Our focus in this section is on cubature optimization and approximation of the coupling forces.

Uniform weight cubature optimization. The deformation of a coupling element causes not only the reduced coupling force in each domain's subspace, but also the linear force and the torque that affect the rigid motion of each domain. Since all of these forces must be evaluated, we propose to use cubature approximation for all of them and we handle them together in cubature optimization. For simplicity, we define the overall "force" vector generated by an interface layer $\mathcal{E}_{ij}$ as:

$$\mathsf{F} = \left[\frac{\mathbf{F}_{ij}^{i\,\mathsf{T}}}{\left\|\mathbf{F}_{ij}^i\right\|}, \frac{\mathbf{F}_{ij}^{j\,\mathsf{T}}}{\left\|\mathbf{F}_{ij}^j\right\|}, \frac{\mathbf{f}_{ij}^{i\,\mathsf{T}}}{\left\|\mathbf{f}_{ij}^i\right\|}, \frac{\mathbf{f}_{ij}^{j\,\mathsf{T}}}{\left\|\mathbf{f}_{ij}^j\right\|}, \frac{\boldsymbol{\tau}_{ij}^{i\,\mathsf{T}}}{\left\|\boldsymbol{\tau}_{ij}^i\right\|}, \frac{\boldsymbol{\tau}_{ij}^{j\,\mathsf{T}}}{\left\|\boldsymbol{\tau}_{ij}^j\right\|}\right]^{\mathsf{T}}, \tag{11}$$

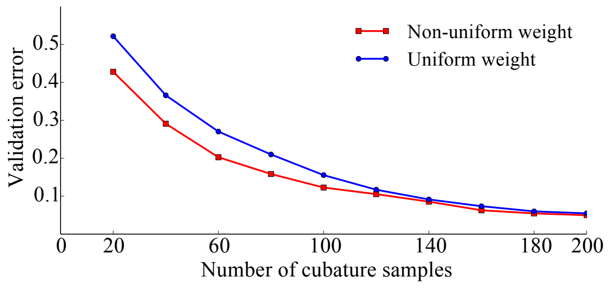


Figure 5: *The errors of using different cubature schemes. This plot shows that using non-uniform weights provides more accurate approximation than using uniform weights. However, their difference becomes less significant as the total number of samples increases.*

which contains all of three forces applied on both domain $\mathcal{D}_i$ and domain $\mathcal{D}_j$. We normalize each force component, to avoid any bias during the optimization process. Once we obtain a large set of these high-dimensional force vector data, we use the optimization framework [An et al. 2008] to generate cubature samples and weights. Since these forces are collected into a single vector and optimized together, they will be evaluated with the same cubature weights. Figure 5 shows the approximation errors using different numbers of samples, when we use this optimization strategy.

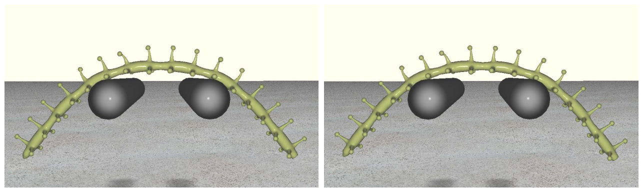
Non-uniform weight cubature optimization. We can make cubature approximation more accurate, by using different samples and weights for different forces. If we use different samples for different forces, we need to evaluate more samples in total, which becomes less efficient in practice. So we choose to still use the same samples, but different weights for different forces.

Specifically, we first run the uniform weight optimization strategy to obtain cubature samples and uniform weights as before. We then allow the weights to vary, by running four additional optimization processes for $\mathbf{F}^i_{ij}$, $\mathbf{F}^j_{ij}$, $(\mathbf{f}^i_{ij}, \mathbf{f}^j_{ij})$, and $(\boldsymbol{\tau}^i_{ij}, \boldsymbol{\tau}^j_{ij})$, respectively. Here we still use the same cubature weights for $\mathbf{f}^i_{ij}$ and $\mathbf{f}^j_{ij}$, and $\boldsymbol{\tau}^i_{ij}$ and $\boldsymbol{\tau}^j_{ij}$, to ensure linear and angular momentum conservation in rigid body motion. Note that the use of non-uniform weights will cause the linear system presented in Section 4.3 to be asymmetric, so we replace Cholesky decomposition by LU decomposition in our direct solver accordingly. Since the solver contributes a small portion of the overall computational cost, the use of an asymmetric matrix has limited influence on the system performance. Figure 5 compares the approximation errors of using uniform and non-uniform cubature weights. In general, the non-uniform weight scheme is more accurate, especially when the number of samples is small. Figure 6 shows the result of using cubature approximation is visually indistinguishable from the result of exact evaluation.

6 Implementation

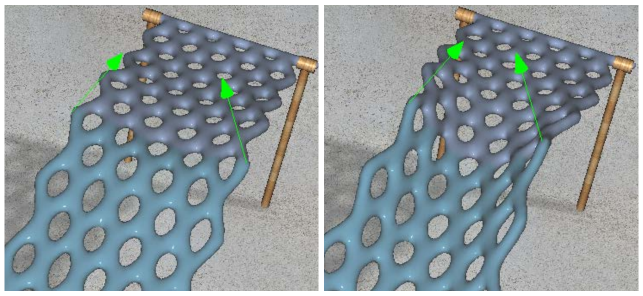
Here we will provide some implementation details of our system.

Subspace basis construction. A nice feature of our system is that it does not have any strict requirement on the subspace basis. In fact, the basis of each domain can be generated using many existing techniques, including motion capture data, pre-computed simulation data, and modal analysis. In our current implementation, we use linear modal analysis and modal derivatives proposed in [Barbič and James 2005] to generate both linear and nonlinear deformation modes in the basis. Our system allows the basis to be generated separately for each domain. In our experiment, it took less than five minutes to finish constructing subspace bases for all of the domains. In contrast, Kim and James [2011] constructed the basis for the whole mesh first and then divided each deformation mode into separate ones for the domains. Their method becomes computationally expensive if the mesh is too large. The method also has difficulty in factoring out the rotational components in the resulting basis of each domain. Finally, their bases may cause the system matrix to be nearly singular, as our experiment shows.



(a) Using cubature approximation (b) Using exact evaluation

Figure 6: *A caterpillar example. Our cubature approximation scheme accurately estimates the reduced forces within the domains and at the interfaces. So the result of using cubature approximation in (a) is comparable to the result using exact evaluation in (b).*



(a) Rigid interface (b) Non-rigid interface

Figure 7: *Comparison of rigid and non-rigid interfaces. Our system allows the interface to be naturally deformed, as shown in (b).*

Cubature optimization. Since the scope of our method is not limited to skinned characters, we can not sample skeleton poses to generate pose data for cubature optimization as did in [Kim and James 2011]. Instead, we apply random forces on each domain to get randomly deformed configurations for a non-skinned mesh. We then use the invertible FEM method [Irving et al. 2004; Teran et al. 2005] to generate full-space simulation data. Given the training data, we perform intra-domain cubature optimization in the same way as described in [An et al. 2008]. We process interface cubature optimization differently, as discussed previously in Section 5. In our experiment, the training data took 2 to 10 minutes to generate and the cubature optimization step took 5 to 30 minutes.

7 Results

We tested our system on an Intel Core i7-2600 3.4GHz processor. We use the OpenMP library to parallelize our simulation steps. All of our examples are simulated at the time step $\Delta t = 1/90$s. Currently, our examples use the St. Venant-Kirchhoff material model, whose stiffness is parameterized by two Lamé coefficients.

Figure 7b shows our system can simulate natural deformation of the whole object, including the interface between two domains. In contrast, if we do not allow the interface to deform, the multi-domain problem can be simplified but the result looks strange, as Figure 7a shows. Here we generate the result in Figure 7a by enforcing the rigidity constraint on the interface during basis reconstruction.

Name and Statistics	# of Modes	DoFs	Interface Tets	Cubature Tets (interface / total)	Domain Dynamics	Domain Coupling	Cholesky Factorization	Full-space Conversion	Total Cost
Armadillo	20	280	7,441	395/ 928	4.9ms	3.4ms	0.9ms	4.1ms	16.7ms
62K nodes, 241K tets	30	390	7,441	492/1,702	13.4ms	4.8ms	2.0ms	5.0ms	30.3ms
11 domains, 10 interfaces	40	500	7,441	563/2,051	23.2ms	6.2ms	3.2ms	6.2ms	45.1ms
Octopus	20	436	2,807	478/1,335	5.6ms	3.3ms	1.0ms	2.4ms	16.1ms
24K nodes, 84K tets	30	606	2,807	798/3,284	19.4ms	5.7ms	2.3ms	3.3ms	38.4ms
17 domains, 16 interfaces	40	776	2,807	951/3,945	33.9ms	7.9ms	3.8ms	4.4ms	59.8ms
Hammock	20	118	1,105	160/ 404	3.7ms	1.7ms	0.3ms	1.4ms	10.0ms
14K nodes, 42K tets	30	168	1,105	201/ 678	6.4ms	2.0ms	0.7ms	2.1ms	14.4ms
5 domains, 4 interfaces	40	218	1,105	260/ 961	14.0ms	3.1ms	1.3ms	2.4ms	23.9ms
Caterpillar	20	202	500	165/ 740	3.2ms	1.0ms	5.6ms	2.8ms	11.4ms
31K nodes, 98K tets	30	282	500	254/1,457	9.1ms	1.3ms	1.2ms	4.2ms	19.8ms
8 domains, 7 interfaces	40	362	500	346/1,967	17.0ms	2.2ms	2.2ms	5.0ms	32.7ms

Table 3: *Mesh and performance statistics. This table lists the number of modes per domain, the total degrees of freedom, the number of interface tetrahedra, the number of interface cubature tetrahedra, the total number of cubature tetrahedra, the computational cost of each simulation step, and the total computational cost per time step.*

Name	# of Modes	Frame Rate		
		single domain	no inter. cubature	ours
Armadillo	20	0.53	16.0	59.9
	30	0.09	12.2	33.0
	40	0.04	8.6	22.2
Octopus	20	0.09	32.8	62.2
	30	0.02	15.5	26.1
	40	0.01	11.4	16.7
Hammock	20	17.7	71.1	100.0
	30	5.07	43.8	69.5
	40	1.54	28.9	41.8
Caterpillar	20	2.51	67.3	87.5
	30	0.31	37.6	50.6
	40	0.10	24.6	31.5

Table 4: *Frame rates. This table lists the simulation frame rates.*

Performance evaluation. Table 3 summarizes the statistics of the 3D models and their performances in our experiment. It shows that the domain dynamics step, which constructs the unified system for linear solve, is typically the computational bottleneck. When a model (such as the armadillo example) contains many interface tetrahedra, the use of interface cubature is necessary to reduce coupling force and matrix evaluation costs. In contrast, if a model (such as the caterpillar example) does not contain many interface tetrahedra, the performance improvement provided by interface cubature becomes small, as Table 4 shows. Table 4 also illustrates that the simulation speed is much lower, if the system uses a single domain with the same number of degrees of freedom. This is because the matrix would become dense, which is more computationally expensive to construct and solve. Note that our system is compatible with parallel computing and it has a good potential to run faster on GPU.

An interface domain in the full space. If the interface between two subspace domains contains a thick layer of elements, we can treat it as a domain as well and simulate it in the full space. To do that, we simply incorporate its implicit time integration into the linear system and handle the coupling between the interface domain and the subspace domain in the same way as before. Since the interface domain is defined in the global frame, there is no need to simulate its rigid motion separately. One use of these interface domains is to allow user to edit the mesh. For example, user can cut the mesh in the interface domains, as shown in Figure 1b and 9c. This cannot be done if the domain is simulated in the subspace.

Comparison to spring-based coupling. An interesting question is what if we couple the domains directly by springs, not by coupling elements. To answer this question, we implemented a spring-based coupling method in our system, similar to [Kim and James 2011]. We also simulated the ground truth in full-space. Figure 8 compares the results of these methods. It indicates that the result quality of the spring-based coupling method really depends on the spring stiffness parameter. If the stiffness is too low, the springs cannot keep the two domains well attached, and if the stiffness is too high, locking and artificial damping issues will become obvious. The stiffness value depends on the mesh resolution and the time step, so it needs to be tuned for different interfaces and simulation cases. In contrast, our coupling method allows all of the elements to use the same stiffness and it does not need to tune the stiffness of the coupling elements separately, although the stability issue still occurs if the stiffness is too high.

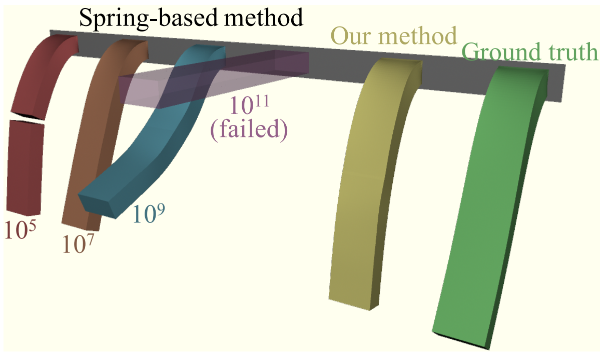


Figure 8: *A 3D beam example, partitioned into two domains. This example compares the results of spring-based coupling, element-based coupling (as our method), and full-space simulation (as ground truth). Our method allows all of the elements to use the same stiffness, so it avoids the coupling problems that may occur in stiffness parameter tuning.*

8 Conclusions

We present a new multi-domain subspace simulation technique, based on the vertex-based mesh partitioning strategy. Our research shows that this technique can overcome the limitations of many previous techniques, and our system can be built upon the same elastic model without additional hard or soft coupling constraints. Our experiment demonstrates the flexibility of our system in handling large and complex deformable bodies, and its performance is comparable to those of the existing subspace simulation techniques.

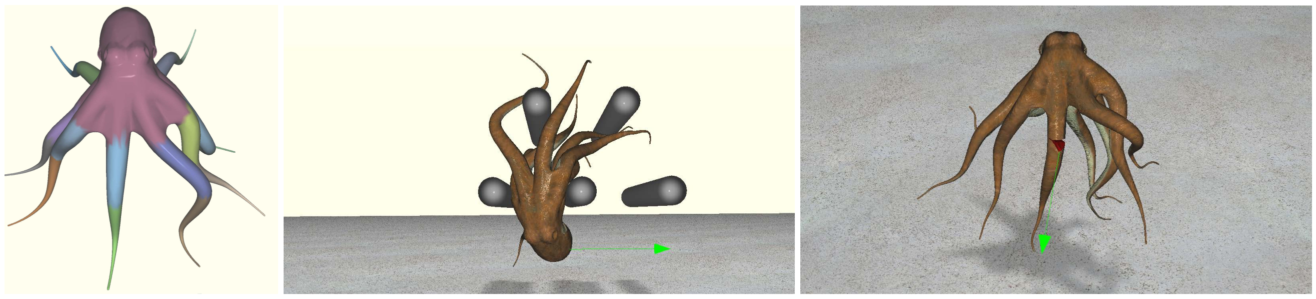

(a) The domains (b) Unconstrained deformation (c) Deformation after cut

Figure 9: *An octopus example. Our system efficiently simulates this octopus example in real time, which contains 17 domains.*

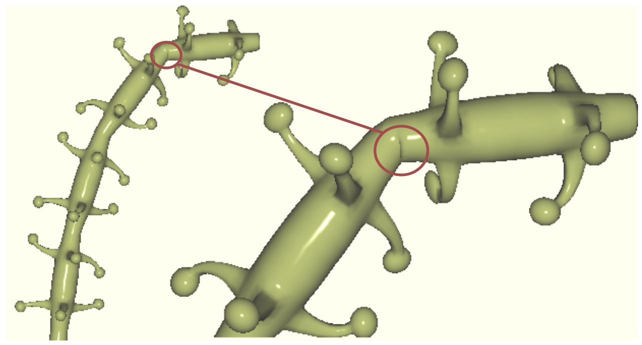


Figure 10: *Discontinuity artifact. This example demonstrates the discontinuity artifact at the interface, due to large deformation.*

Limitations and future work. Our system does not solve the fundamental discontinuity issue among multiple domains simulated in their own subspaces. Because of this, the surface may be uneven and the coupling elements may be inverted, when the body undergoes large deformation as shown in Figure 10. One solution to this problem is to extend the boundary-aware idea [Yang et al. 2013] to nonlinear deformation modes, which however is not so straightforward and needs further research in the future. The coupling force in our system needs additional cubature samples for evaluation, which increases the overall computational cost. Our plan is to implement the whole system on GPU, so that forces and Jacobian matrices can be quickly evaluated in parallel. Our system separates rigid motions from non-rigid motions. Although it provides a practical solution to dynamic simulation, it is not strictly correct and the result can be different from the result of full-space simulation. Currently, our system integrates nonlinear fictitious forces explicitly and implements implicit time integration by a single Newton iteration. As a result, the system can become unstable when it handles complex hyperelastic materials or fast rotations. We can integrate fictitious forces implicitly and use multiple Newton iterations in the future, if the computational resource allows. Finally, our system does not consider collisions yet and it is interesting to know how self collisions can be handled in multi-domain subspace simulation.

9 Acknowledgments

We thank Nvidia and Adobe for funding and equipment support. This work was supported in part by NSF grant IIS-1524992.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. (SIGGRAPH Asia) 27*, 5 (Dec.), 165:1–165:10.

ANDO, R., THÜREY, N., AND WOJTAN, C. 2015. A dimension-reduced pressure solver for liquid simulations. *Computer Graphics Forum (Eurographics) 34*, 2, 473–480.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 43–54.

BARAFF, D. 1997. An introduction to physically based modeling: Rigid body simulation i - unconstrained rigid body dynamics. In *An Introduction to Physically Based Modelling, SIGGRAPH '97 Course Notes*, 97.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. (SIGGRAPH) 24*, 3 (July), 982–990.

BARBIČ, J., AND JAMES, D. L. 2010. Subspace self-collision culling. *ACM Trans. Graph. (SIGGRAPH) 29*, 4 (July), 81:1–81:9.

BARBIČ, J., AND ZHAO, Y. 2011. Real-time large-deformation substructuring. *ACM Trans. on Graphics (SIGGRAPH 2011) 30*, 4, 91:1–91:7.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. *ACM Trans. Graph. (SIGGRAPH) 27*, 3 (Aug.), 63:1–63:12.

CHEN, Z., FENG, R., AND WANG, H. 2013. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 88:1–88:8.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Trans. Graph. (SIGGRAPH) 21*, 3 (July), 604–611.

CHOI, M. G., AND KO, H.-S. 2005. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Trans. Vis. Comp. Graph. 11*, 1 (Jan.), 91–101.

HAHN, F., THOMASZEWSKI, B., COROS, S., SUMNER, R. W., COLE, F., MEYER, M., DEROSE, T., AND GROSS, M. 2014. Subspace clothing simulation using adaptive bases. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (July), 105:1–105:9.

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 107:1–107:10.

HAUSER, K. K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, 247–256.

HECHT, F., LEE, Y. J., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2012. Updated sparse Cholesky factors for corotational elastodynamics. *ACM Trans. Graph. 31*, 5 (Sept.), 123:1–123:13.

HUANG, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. An efficient large deformation method using domain decomposition. *Comput. Graph. 30*, 6 (Dec.), 927–935.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of SCA*, 131–140.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph. (SIGGRAPH Asia) 27*, 5 (Dec.), 164:1–164:11.

KIM, T., AND DELANEY, J. 2013. Subspace fluid re-simulation. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 62:1–62:9.

KIM, T., AND JAMES, D. L. 2011. Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of SCA*, 63–72.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. (SIGGRAPH) 24*, 3 (July), 471–478.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph. 23*, 3 (July), 207–214.

SIFAKIS, E., AND BARBIC, J. 2012. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, 20:1–20:50.

TENG, Y., OTADUY, M. A., AND KIM, T. 2014. Simulating articulated subspace self-contact. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (July), 106:1–106:9.

TERAN, J., BLEMKER, S., HING, V. N. T., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of SCA*, 68–74.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of SCA*, 181–190.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '87, 205–214.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. Graph. (SIGGRAPH) 25*, 3 (July), 826–834.

UMETANI, N., SCHMIDT, R., AND STAM, J. 2014. Position-based elastic rods. In *ACM SIGGRAPH 2014 Talks*, ACM, New York, NY, USA, SIGGRAPH '14, 47:1–47:1.

WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. (SIGGRAPH Asia) 29*, 6 (Dec.), 156:1–156:10.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph. (SIGGRAPH) 28*, 3 (July), 39:1–39:8.

YANG, Y., XU, W., GUO, X., ZHOU, K., AND GUO, B. 2013. Boundary-aware multidomain subspace deformation. *IEEE Trans. Vis. Comp. Graph. 19*, 10 (Oct), 1633–1645.

A Derivation of Equation 9

Since only internal elastic forces are integrated implicitly, we can discretize Equation 6 in time by plugging in $\ddot{\mathbf{q}} = (\dot{\mathbf{q}}' - \dot{\mathbf{q}})/\Delta t$:

$$(\dot{\mathbf{q}}' - \dot{\mathbf{q}})/\Delta t + \mathbf{U}_i^\mathsf{T}\mathbf{D}_i\mathbf{U}_i\dot{\mathbf{q}}'_i + \mathbf{U}_i^\mathsf{T}\mathbf{f}_i^{\text{int}\prime}(\mathbf{U}_i\mathbf{q}'_i) + \sum_{\mathcal{E}_{ij}} \mathbf{F}_{ij}^{i\,\prime} = \mathbf{F}_i^{\text{sub}}. \quad (12)$$

By linearizing the internal forces at time t, we can get the approximate internal force at the next time instant $t + \Delta t$:

$$\begin{aligned}\mathbf{f}_i^{\text{int}\prime}(\mathbf{U}_i\mathbf{q}'_i) &= \mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i) + \tfrac{\partial \mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i)}{\partial \mathbf{q}_i}(\mathbf{q}'_i - \mathbf{q}_i)\\ &= \mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i) + \tfrac{\partial \mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i)}{\partial \mathbf{u}_i}\tfrac{\partial \mathbf{u}_i}{\partial \mathbf{q}_i}(\mathbf{q}'_i - \mathbf{q}_i)\\ &= \mathbf{f}_i^{\text{int}}(\mathbf{U}_i\mathbf{q}_i) + \Delta t\mathbf{K}_i\mathbf{U}_i\dot{\mathbf{q}}'_i.\end{aligned} \quad (13)$$

With similar linearization and derivation, we get the equation describing the reduced coupling force as:

$$\sum_{\mathcal{E}_{ij}} \mathbf{F}_{ij}^{i\,\prime} = \sum_{\mathcal{E}_{ij}} \left(\mathbf{F}_{ij}^i + \frac{\partial \mathbf{F}_{ij}^i}{\partial \mathbf{Q}_{ij}}\mathbf{Q}'_{ij}\right). \quad (14)$$

After merging Equation 12 to 14, we obtain Equation 9.

B Equations for Implicit Integration

Let $\mathbf{x}_{ij,k}^l$ be the l-th vertex of element $e_{ij,k}$ in the global frame and $\tilde{\mathbf{x}}_{ij,k}^l$ be its position in the local frame of domain $\mathcal{D}_i$. The matrix-vector product term in Equation 9 can be evaluated as:

$$\begin{aligned}&\sum_{\mathcal{E}_{ij}}\left(\frac{\partial \mathbf{F}_{ij}^i}{\partial \dot{\mathbf{q}}_i}\dot{\mathbf{q}}'_i + \frac{\partial \mathbf{F}_{ij}^i}{\partial \mathbf{v}_i}\mathbf{v}'_i + \frac{\partial \mathbf{F}_{ij}^i}{\partial \boldsymbol{\omega}_i}\boldsymbol{\omega}'_i + \frac{\partial \mathbf{F}_{ij}^i}{\partial \dot{\mathbf{q}}_j}\dot{\mathbf{q}}'_j + \frac{\partial \mathbf{F}_{ij}^i}{\partial \mathbf{v}_j}\mathbf{v}'_j + \frac{\partial \mathbf{F}_{ij}^i}{\partial \boldsymbol{\omega}_j}\boldsymbol{\omega}'_j\right)\\ &= \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}_{ij,k}^{l_0},\mathbf{x}_{ij,k}^{l_1}\in\mathcal{X}_i}\left(\Delta t(\mathbf{R}_i\mathbf{U}_{ij,k}^{l_0})^\mathsf{T}\frac{\partial \mathbf{f}_{ij,k}^{l_0}}{\partial \mathbf{x}_{ij,k}^{l_1}}(\mathbf{R}_i\mathbf{U}_{ij,k}^{l_1}\dot{\mathbf{q}}'_i + \mathbf{v}'_i + \boldsymbol{\omega}'_i \times \mathbf{R}_i\tilde{\mathbf{x}}_{ij,k}^{l_1})\right) +\\ &\quad \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}_{ij,k}^{l_0}\in\mathcal{X}_i,\mathbf{x}_{ij,k}^{l_1}\in\mathcal{X}_j}\left(\Delta t(\mathbf{R}_i\mathbf{U}_{ij,k}^{l_0})^\mathsf{T}\frac{\partial \mathbf{f}_{ij,k}^{l_0}}{\partial \mathbf{x}_{ij,k}^{l_1}}(\mathbf{R}_j\mathbf{U}_{ij,k}^{l_1}\dot{\mathbf{q}}'_j + \mathbf{v}'_j + \boldsymbol{\omega}'_j \times \mathbf{R}_j\tilde{\mathbf{x}}_{ij,k}^{l_1})\right),\end{aligned} \quad (15)$$

in which the term $\partial \mathbf{f}_{ij,k}^{l_0}/\partial \mathbf{x}_{ij,k}^{l_1}$ is one of the 3×3 sub-blocks of the element stiffness matrix corresponding to element $e_{ij,k}$. The matrix-vector product terms in Equation 10 can be expanded in a similar fashion.

Matrix symmetry. Although it is not intuitive, the system matrix assembled from Equation 9 and 10 is indeed symmetric. This is because every term in these equations has a corresponding symmetric term in one of the other equations. For example, the term $\sum_{\mathcal{E}_{ij}} \partial \mathbf{F}_{ij}^i/\partial \mathbf{v}_j$ in Equation 15 is one sub-block of the system matrix $\mathbf{A}$, whose its transpose is:

$$\begin{aligned}\sum_{\mathcal{E}_{ij}}\left(\frac{\partial \mathbf{F}_{ij}^i}{\partial \mathbf{v}_j}\right)^\mathsf{T} &= \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}_{ij,k}^{l_0}\in\mathcal{X}_i,\mathbf{x}_{ij,k}^{l_1}\in\mathcal{X}_j}\left(\Delta t(\mathbf{R}_i\mathbf{U}_{ij,k}^{l_0})^\mathsf{T}\frac{\partial \mathbf{f}_{ij,k}^{l_0}}{\partial \mathbf{x}_{ij,k}^{l_1}}\right)^\mathsf{T}\\ &= \sum_{e_{ij,k}\in\mathcal{E}_{ij},\mathbf{x}_{ij,k}^{l_0}\in\mathcal{X}_i,\mathbf{x}_{ij,k}^{l_1}\in\mathcal{X}_j}\left(\Delta t\frac{\partial \mathbf{f}_{ij,k}^{l_1}}{\partial \mathbf{x}_{ij,k}^{l_0}}\mathbf{R}_i\mathbf{U}_{ij,k}^{l_0}\right) = \sum_{\mathcal{E}_{ij}}\frac{\mathbf{f}_{ij}^j}{\dot{\mathbf{q}}_i}.\end{aligned} \quad (16)$$

Here, the symmetry of element $e_{ij,k}$'s tangent stiffness matrix, i.e., $(\partial \mathbf{f}_{ij,k}^{l_0}/\partial \mathbf{x}_{ij,k}^{l_1})^\mathsf{T} = \partial \mathbf{f}_{ij,k}^{l_1}/\partial \mathbf{x}_{ij,k}^{l_0}$ is used in this derivation. Equation 16 indicates that $\sum_{\mathcal{E}_{ij}} \partial \mathbf{F}_{ij}^i/\partial \mathbf{v}_j$ and $\sum_{\mathcal{E}_{ij}} \mathbf{f}_{ij}^j/\dot{\mathbf{q}}_i$ are two symmetric blocks. Similar relationships can be derived for other terms.