Data-Driven Finite Elements for Geometry and Material Design

Desai Chen[1] David I.W. Levin[12] Shinjiro Sueda[123] Wojciech Matusik[1]

[1]MIT CSAIL [2]Disney Research [3]California Polytechnic State University

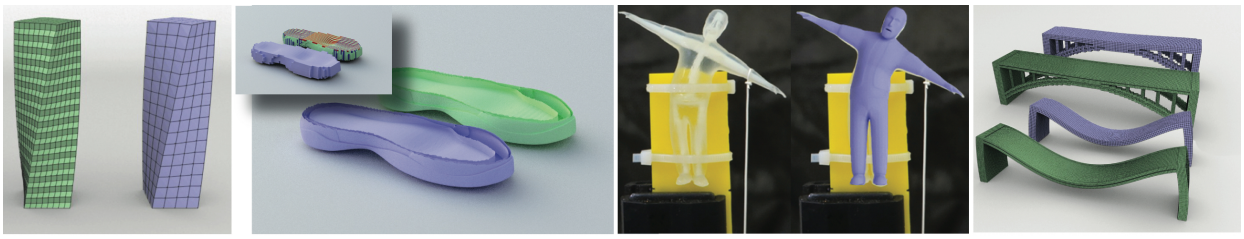


Figure 1: *Examples produced by our data-driven finite element method. Left: A bar with heterogeneous material arrangement is simulated 15x faster than its high-resolution counterpart. Left-Center: Our fast coarsening algorithm dramatically accelerates designing this shoe sole (up to 43x). Right-Center: A comparison to 3D printed results. Right: We repair a flimsy bridge by adding a supporting arch (8.1x) speed-up. We show a High-Res Simulation for comparison.*

Abstract

Crafting the behavior of a deformable object is difficult—whether it is a biomechanically accurate character model or a new multimaterial 3D printable design. Getting it right requires constant iteration, performed either manually or driven by an automated system. Unfortunately, previous algorithms for accelerating three-dimensional finite element analysis of elastic objects suffer from expensive precomputation stages that rely on *a priori* knowledge of the object's geometry and material composition. In this paper we introduce Data-Driven Finite Elements as a solution to this problem. Given a material palette, our method constructs a metamaterial library which is reusable for subsequent simulations, regardless of object geometry and/or material composition. At runtime, we perform fast coarsening of a simulation mesh using a simple table lookup to select the appropriate metamaterial model for the coarsened elements. When the object's material distribution or geometry changes, we do not need to update the metamaterial library—we simply need to update the metamaterial assignments to the coarsened elements. An important advantage of our approach is that it is applicable to non-linear material models. This is important for designing objects that undergo finite deformation (such as those produced by multimaterial 3D printing). Our method yields speed gains of up to two orders of magnitude while maintaining good accuracy. We demonstrate the effectiveness of the method on both virtual and 3D printed examples in order to show its utility as a tool for deformable object design.

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

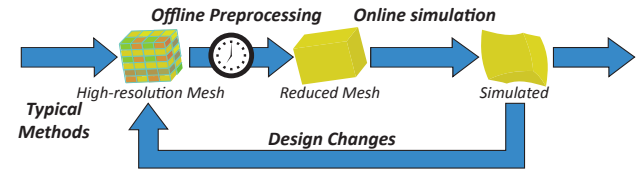
Keywords: Data-driven simulation, finite element methods, numerical coarsening, material design

1 Introduction

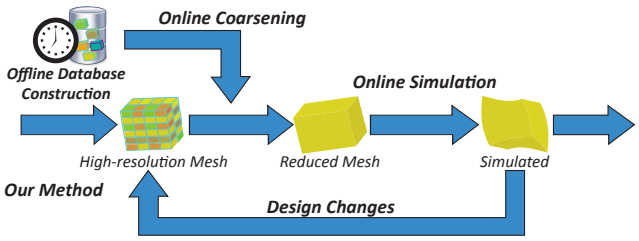
Objects with high-resolution, heterogeneous material properties are everywhere: from the output of multimaterial 3D printers to virtual characters gracing the screen in our summer blockbusters. Designing such objects is made possible by the tight coupling of design tools and numerical simulation which allows designers (or automatic algorithms) to update geometry or material parameters and subsequently estimate the physical effects of the change. Fast, accurate simulation techniques that can handle runtime changes in geometry and material composition are a necessity for such iterative design algorithms.

The gold standard technique for estimating the mechanical behavior of a deformable object under load is the finite element method (FEM). While FEM is accurate, its solution process is notoriously slow, making it a major bottleneck in the iterative design process. For this reason, there have been a large number of works on speeding up FEM simulations, and these speed improvements have enabled FEM to be used in many performance critical tasks such as computer animation, surgical training, and virtual/augmented reality. Unfortunately, even though techniques such as model reduction or numerical coarsening can achieve order-of-magnitude performance increases, they require expensive precomputation phases, typically on the order of minutes for large meshes. This precomputation requires knowledge of an object's geometry and material composition *a priori*, something that is not known during a design task. When the user updates the model by changing the geometry or the material distribution, the preprocessing step must be run again. As shown in Fig. 2a, since this step is inside the design loop, the user cannot get rapid feedback on the changes made to the object.

We propose *Data-Driven FEM* (DDFEM), a new simulation methodology that removes these limitations and is thus extremely well-suited to the types of design problems discussed above. We divide an object into a set of deformable voxels using embedded finite elements and coarsen these voxels hierarchically. A custom metamaterial database is populated with materials that minimize the error incurred by coarsening. This database is learned *once* in a completely offline fashion and depends only on the set of materials to be used by the deformable object and not on the actual material distribution and geometry. At runtime we use the database to perform fast coarsening of an FEM mesh in a way that is agnostic to changes in geometry and material composition of the object. The key features of the algorithm are its ability to handle arbitrary, nonlinear elastic



(a) *Typical methods*



(b) *Our method*

Figure 2: *(a) In a typical method, the preprocessing step is offline, making the design loop slow. (b) In our method, we move the time-consuming offline computation outside of the design loop.*

constitutive models as well as to avoid expensive precomputation within the design loop (Fig. 2b). DDFEM is the first algorithm optimized for interactive geometric and material design problems.

2 Related Work

Efficient FEM simulation plays an important role in designing deformable objects. As mentioned, these problems are common in engineering and graphics [Bendsoe and Sigmund 2003; Bickel et al. 2010; Kou et al. 2012; Skouras et al. 2013; Chen et al. 2013; Xu et al. 2014]. We can broadly partition the space of approaches for optimizing FEM simulation into two categories. We term the first category *numerical* approaches. These methods use fast matrix inversion techniques and other insights about the algebra of the finite element method to increase its performance. Simulators based on the multigrid method [Peraire et al. 1992; Zhu et al. 2010; McAdams et al. 2011] and Krylov subspace techniques [Patterson et al. 2012] have yielded impressive performance increases. Other hierarchical numerical approaches, as well as highly parallel techniques, have also been applied to improve the time required to perform complex simulations [Farhat and Roux 1991; Mandel 1993]. Finally, Bouaziz et al. [2014] propose specially designed energy functions and an alternating time-integrator for efficient simulation of dynamics.

The second set of methods are *reduction* approaches. These algorithms attempt to intelligently decouple or remove degrees of freedom (DOFs) from the simulated system. This leads to smaller systems resulting in a massive increase in performance, with some decrease in accuracy. Our algorithm falls into this category. Note, however, that numerical and reduction approaches need not be mutually exclusive. For example, our algorithm may potentially be used as a preconditioner for a numerical approach. Algorithms based on reduction approaches mitigate the inevitable increase in error using one or more of three approaches: Adaptive remeshing, higher-order shape functions, or by adapting the constitutive model.

Adaptive remeshing alters the resolution of the simulation discretization in response to various metrics (stress, strain etc.). Such methods seek to maintain an optimal number of elements and thus achieve reasonable performance. Adaptive remeshing has proven useful for simulating thin sheets such as cloth [Narain et al. 2012], paper [Narain et al. 2013], as well as elastoplastic solids [Wicke et al. 2010] and solid-fluid mixtures [Clausen et al. 2013]. More general basis refinement approaches have also been suggested [Debunne et al. 2001; Grinspun et al. 2002]. While these methods do improve the performance of simulation algorithms, they have some drawbacks. First, they often require complicated geometric operations which can be time consuming to implement. Second, they introduce elements of varying size into the FEM discretization. This can lead to poor numerical conditioning if not done carefully. Finally, in order to maintain accuracy, it may still be necessary to introduce many fine elements, leading to slow performance. Alternatively, one can turn to P-Adaptivity for help. This refers to adaptively introducing higher-order basis functions in order to increase accuracy during simulation [Szabó et al. 2004]. Unfortunately, these methods suffer from requiring complicated mesh generation schemes and are not well-suited for iterative design problems.

An alternative approach to remeshing is to use higher order shape functions in order to more accurately represent the object's motion using a small set of DOFs. Modal simulation techniques fall into this category [Shabana 1991; Krysl et al. 2001; Barbič and James 2005]. Substructuring [Barbič and Zhao 2011] decomposes an input geometry into a collection of basis parts, performing modal reduction on each one. These basis parts can be reused to construct new global structures. Other approaches involve computing physically meaningful shape functions as an offline preprocessing step. For instance, Nesme et al. [2009] compute shape functions based on the static configuration of a high resolution element mesh induced via a small deformation of each vertex. Faure et al. [2011] use skinning transformations as shape functions to simulate complex objects using a small number of frames. Gilles et al. [2011] show how to compute material aware shape functions for these frame-based models, taking into account the linearized object compliance. Both Nesme et al. [2009] and Faure et al. [2011] accurately capture material behavior in the linear regime, but, because their shape functions cannot change with the deformed state of the material, they do not accurately capture the full, non-linear behavior of an elastic object. Our non-linear metamaterials rectify this problem. Computing material aware shape functions improves both the speed and accuracy of the simulation. However, these methods require a precomputation step that assumes a fixed material distribution and geometry. If the material distribution changes, these shape functions must be recomputed, and this becomes a bottleneck in applications that require constantly changing material parameters.

The final coarsening technique involves reducing the degrees of freedom of a mesh while simultaneously augmenting the constitutive model at each element, rather than the shape functions. Numerical coarsening is an extension of analytical homogenization which seeks to compute optimal, averaged material for heterogeneous structures [Guedes and Kikuchi 1990; Farmer 2002]. Numerical coarsening, for instance, has been applied to linearly (in terms of material displacement) elastic tetrahedral finite elements [Kharevych et al. 2009]. These methods require an expensive precomputation step (a series of static solves) that must be repeated when the material content, or the geometry of an object changes. This holds these methods back from being suitable for iterative design problems.

Recently, three methods have been introduced that are similar to ours in spirit. Bickel et al. [2009] measured force-displacement to compute a spatially varying set of Young's moduli, interpolated in strain space. Our work also involves learning new constitutive models for finite element methods with several key differences. First, we present a more robust energy-based metamaterial model that does not require incremental loading during simulation. Second, the previous work relies on captured data to build constitutive model, while we use a new sampling strategy that allows us to build our metamaterial model virtually. This allows us to leverage large, high-performance compute clusters to speed up the process. Finally, their work is completely geometry dependent—their computed material models cannot be transferred to new meshes. Xu et al. [2014] and Li et al. [2014] computed material distribution given user specified forces

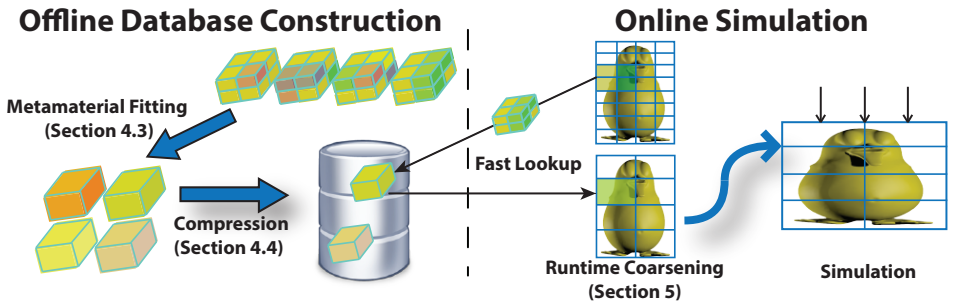


Figure 3: *An overview of DDFEM. The method is divided into an offline database construction phase and an online simulation phase. During database construction we build a collection of metamaterials which can be used for accurate coarsening. These metamaterials are compressed (optional) and stored in a database, indexed by material ID. During online simulation we coarsen a high-resolution simulation mesh by performing a fast lookup into the precomputed database. Finally, finite element simulation is performed on this coarsened mesh.*

and displacements. They compute materials in a low-dimensional space of material modes to speedup and regularize the solution. However, their method requires a known geometry, and furthermore, users cannot control per-element material assignment. Rather than computing material distribution, our method supports user specified topology changes and material assignment. These features are important due to the design-centric nature of our work.

Data-driven techniques have also been applied to add fine detail to coarse surgical simulations [Cotin et al. 1999; Kavan et al. 2011; Seiler et al. 2012]. While these methods do not help with the type of iterative design problems addressed here, they could be combined with our fast coarsening in order to quickly "upscale" coarsened simulation results to original simulation resolution.

2.1 Contributions

This work makes the following technical contributions:

- The first algorithm for fast runtime coarsening of arbitrary, nonlinear, elastic material models for finite element analysis.
- A compact metamaterial model used for coarsening.
- An efficient procedure for fitting our metamaterial model.

3 Overview

DDFEM is a combination of embedded finite elements and hierarchical coarsening. In this section, we discuss the problem of coarsening and introduce the notion of a material palette. We conclude by summarizing the two main stages of DDFEM—offline metamaterial construction and online coarsening.

Coarsening for finite elements The key component of our DDFEM is coarsening. Coarsening involves reducing the number of vertices in a finite element simulation mesh in order to improve runtime performance. Since simply removing vertices can greatly reduce the accuracy of the simulation, coarsening schemes also assign new materials to coarsened elements to minimize this effect.

We regard the global coarsening of a simulation mesh as the result of many local coarsening operations which map from contiguous subsets of fine elements with applied materials to coarse elements with new, optimized materials. Our goal is to precompute these optimized materials so that coarsening is fast at runtime. Below we discuss how to make such a precomputation tractable beginning with our choice of Finite Element simulation methodology.

Conforming vs. embedded finite elements The defining feature of conforming finite element methods is that the simulation mesh is aligned with the geometry of the object being simulated. One obvious feature of conforming meshes is that the mesh itself is a function of the input geometry. This means that the output of a local coarsening operator (the coarsened mesh) will also be a function of the input geometry. Also, the new material computed by each local coarsening operator will be a function of input geometry. This dependence on input geometry is a significant issue to overcome if we wish to precompute coarsened materials because, in design problems, the input geometry is in constant flux. The number of precomputed coarse materials now depends on the local material assignment on the simulation mesh and the input geometry. Thus space of coarsened materials is prohibitively large. To mitigate this we turn to embedded finite elements. These methods embed the geometry to be simulated into the simulation mesh with no regard for whether the mesh conforms to the geometry or not. Thus an identical simulation mesh can be used for any input geometry. Local coarsening operations on the embedded mesh yield identical coarse elements and the optimized coarse material depends only on the local material distribution on the simulation mesh. This significantly reduces the size of the coarsened material space. In this paper we embed all simulation geometry into a hexahedral simulation mesh.

Material palette We further shrink the space of coarsening operators using an observation about material design. Designers do not work in a continuous space of materials but limit themselves to a relatively compact set (e.g. rubber, wood, steel) related to their problem domain (Fig. 4). We call these discrete sets of materials palettes and denote them $\mathcal{P} = \{\mathcal{M}^0, \mathcal{M}^1, \ldots, \mathcal{M}^n\}$. Here $\mathcal{M}^i$ denotes a specific material model in $\mathcal{P}$, and n is the size of the material palette. In this work we limit ourselves to (nonlinear) hyper-elastic materials, which means that each $\mathcal{M}_i$ can be represented by a strain energy density function. We also include a void (or empty) material in every palette. This allows us to perform topology changes in the same manner in which we perform material assignment updates. In subsequent sections, we use a left superscript to indicate the level of coarsening. For example, ${}^0\mathcal{P}$ denotes a material palette at the fine scale while ${}^1\mathcal{P}$ denotes the new palette of metamaterials that results from the first coarsening step.

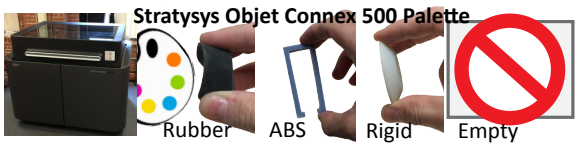


Figure 4: *An example material palette for the Stratasys Objet Connex 500 3D printer.*

Algorithms With the material palette in hand, we can now define our algorithm, which is divided into two distinct phases: an **offline database construction** stage and an **online coarsening** stage. Below we detail the input, output, and steps of each stage:

Offline Database Construction

- **INPUT:** A palette of materials to be applied to high-resolution hexahedral simulation meshes ${}^0\mathcal{P}$
- **OUTPUT:** A new palette of coarse metamaterials, ${}^1\mathcal{P}$, and a mapping from fine material combinations to the coarsened materials in ${}^1\mathcal{P}$.
- **STEPS:**
- **FOR EACH** material combination applied to a 2×2×2 cube of high resolution elements
  - Sample potential energy function of 2×2×2 block
  - Fit metamaterial for coarse hexahedral element
  - Add metamaterial to ${}^1\mathcal{P}$ using high resolution material IDs as database key
- **END**

Online Coarsening

- **INPUT:** High resolution hexahedral simulation mesh with material IDs and coarsened hexahedral simulation mesh
- **OUTPUT:** Metamaterial assignments for coarse mesh
- **STEPS:**
- **FOR EACH** 2×2×2 block in the high resolution mesh
  - Replace with single coarse element
  - Assign material from ${}^1\mathcal{P}$ using high resolution material IDs as database key
- **END**

Hierarchical coarsening We stress that both stages of the DDFEM algorithm can be applied hierarchically. Given the first level of metamaterials, ${}^1\mathcal{P}$, we can construct a metamaterial library, ${}^2\mathcal{P}$, for the second level by using ${}^1\mathcal{P}$ as an input material palette. At runtime, the coarsening algorithm looks up materials from ${}^2\mathcal{P}$ to replace each 2×2×2 coarse block with a single element.

Having introduced the broad strokes of the DDFEM scheme, we move on to a detailed explanation of each algorithmic component. First we discuss database construction in §4, followed by the runtime component in §5. We end by demonstrating the speed and accuracy of DDFEM in §6.

4 Metamaterial Database Construction

We construct our metamaterial database using a potential energy fitting approach. This is valid due to the hyperelastic materials that make up our material palettes. Material fitting considers 2×2×2 blocks of high-resolution hexahedral elements (denoted ${}^0\mathcal{E}$). For each element ${}^0E_k \in {}^0\mathcal{E}$, its material is referred to as ${}^0\mathcal{M}_k \in {}^0\mathcal{P}$. (Note that $\mathcal{E}$ refers to a set of elements and E refers to a single element.) Given ${}^0\mathcal{E}$, we can sample its deformation space, and using ${}^0\mathcal{M}_k$, compute the potential energy ${}^0V$ for each sample. Now we must find a metamaterial that, when applied to a single coarse element ${}^1E$ best approximates ${}^0V$. This is accomplished by fitting a metamaterial potential energy function , ${}^1V$, to the set of deformation/energy samples. The fitted metamaterial is stored in the metamaterial database and indexed by the material indices of ${}^0\mathcal{M}$.

4.1 Metamaterial Model

Our fitting approach depends on choosing a good metamaterial model. In order to ensure that our model meets the criteria for a material energy function [Marsden and Hughes 2012], we choose our metamaterial model for ${}^1E$ as a combination of material models

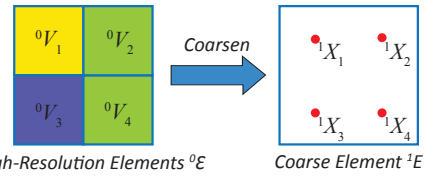


Figure 5: *The relationship between high-resolution and coarsened elements. At each quadrature point ${}^1\boldsymbol{X}_k$, the coarse element copies the corresponding energy density function ${}^0V_k$ from the high-resolution element.*

of ${}^0\mathcal{M}_k$:

$$ {}^1V({}^1\boldsymbol{u}, {}^1\boldsymbol{p}) = \sum_{k=1}^{8} w_k \, {}^0V_k({}^0\boldsymbol{u}_k, {}^1\boldsymbol{p}_k, {}^1\boldsymbol{X}_k), \tag{1} $$

where ${}^0V_k$ is the strain energy density of ${}^0\mathcal{M}_k$ at quadrature point position ${}^1\boldsymbol{X}_k$ (Fig. 5). Here ${}^1\boldsymbol{u}$ is the vector of nodal displacements associated with ${}^1E$ while ${}^0\boldsymbol{u}_k$ are displacements for the k^{th} element at level 0 reconstructed using trilinear interpolation from ${}^1\boldsymbol{u}$. The vector ${}^1\boldsymbol{p}$ stores the material parameters for the coarse metamaterial and consists of the stacked material parameter vectors for each material in ${}^0\mathcal{M}_k$, themselves denoted by ${}^1\boldsymbol{p}_k$. w_k is the standard Gaussian quadrature weight. We note that our model incurs slight computational overhead at runtime because we must evaluate potential energy functions at 8 quadrature points. However, the speed improvement gained by coarsening makes the remaining, per-quadrature point expense negligible.

We observe that even if the individual base material models are isotropic, the metamaterial can become anisotropic by assigning different material parameters at the quadrature points. We counter this by augmenting the metamaterial model with an anisotropic term, which improves fitting. The complete model is then given by

$$ \begin{aligned} {}^1V({}^1\boldsymbol{u}, {}^1\boldsymbol{p}, C) = \sum_{k=1}^{8} \Bigg(& w_k \, {}^0V_k({}^0\boldsymbol{u}, {}^1\boldsymbol{p}_k, {}^1\boldsymbol{X}_k) \\ & + C_k \left(\sqrt{\boldsymbol{v}^T \boldsymbol{F}_k^T \boldsymbol{F}_k \boldsymbol{v}} - 1 \right)^2 \Bigg), \end{aligned} \tag{2} $$

where $\boldsymbol{v}$ is a unit-length direction of anisotropy and C_k is the scaling parameter at the k^{th} quadrature point.

4.2 Force Space Sampling

As mentioned previously, we take a sampling-based approach to metamaterial fitting. In order to fit our model (Eq. 2) to ${}^0V$ we first draw a number of samples from the deformation space of ${}^0\mathcal{E}$ and compute ${}^0V$ for each sample. If a user has prior knowledge of the set of meshes and simulations that they will require, then the best way to draw the samples is to run a number of anticipated simulations with various material combinations. In this paper, we provide a more general method to draw samples for a metamaterial. Initially, we attempted sampling by applying a random deformation to the corners of ${}^1E$; however, this led to many infeasible samples for very stiff materials. In order to alleviate this problem we perform sampling in the force space.

For each element ${}^0E \in {}^0\mathcal{E}$ we apply a set of randomly generated forces. We solve an elastostatic problem to compute the deformation of ${}^0\mathcal{E}$, using constraints to remove rigid motion. Recall that this is fast because ${}^0\mathcal{E}$ consists of just 8 elements. Each sample is then a tuple $\{{}^0\boldsymbol{u}, {}^0V\}$ (Fig. 6) where ${}^0\boldsymbol{u}$ are the nodal displacements of ${}^0\mathcal{E}$, and ${}^0V$ is the strain energy density value of this deformed configuration.

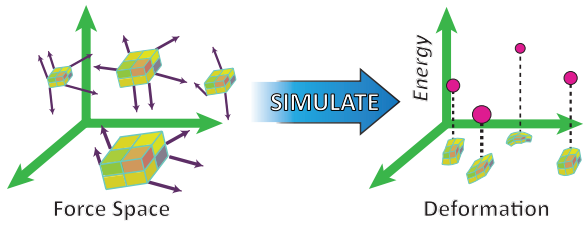


Figure 6: *We sample the energy function of a 2×2×2 block of hexahedra by applying random forces and deforming the block using FEM. Each set of forces results in a deformation-energy tuple which is used during fitting.*

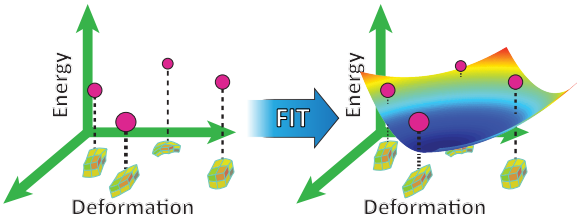


Figure 7: *Metamaterial potential energy functions are fitted to the deformation-energy samples using a least-squares optimization.*

4.3 Fitting

Given a set of deformation samples, $\{{}^0\boldsymbol{u},\ {}^0V\}$, we perform a least squares fit to determine the parameters, ${}^1\boldsymbol{p}$, for a given metamaterial (Fig. 7):

$$ {}^1\boldsymbol{p}^* = \underset{{}^1\boldsymbol{p}}{\operatorname{argmin}} \sum_{s=1}^{n_s} \left({}^0V_s - {}^1V\left(\boldsymbol{r}\left({}^0\boldsymbol{u}_s\right), {}^1\boldsymbol{p}\right)\right)^2, \qquad (3) $$

where $\boldsymbol{r}$ constructs ${}^1\boldsymbol{u}$ from ${}^0\boldsymbol{u}$, n_s is the total number of samples, and s indexes all samples. In our experiments we use the simplest form of $\boldsymbol{r}$ choosing it to extract the displacements of the corners of ${}^0\mathcal{E}$.

Fitting in the Presence of Anisotropy If performed naively this optimization is nonlinear because we must simultaneously solve for $\boldsymbol{v}_k$, the preferred direction of anisotropy. This can severely slow the fitting procedure, especially in cases where it would otherwise be a linear least squares problem (i.e if all fine-scale materials are Neo-Hookean or a similarly simple material model). To avoid this problem we first estimate all anisotropy directions, and then solve Eq. 3 for the remaining material parameters. Our intuition is that anisotropy manifests itself as preferential stretching along a particular axis. To find this axis, we apply stretching forces to a block in a discrete set of directions uniformly sampled over a sphere. If the stretching force is close to the direction of anisotropy, then the amount of stretching deformation is reduced. For any given stretching direction $\mathbf{v}$, we apply a stretching force and compute the deformation gradient $\mathbf{F}$ of each quadrature point. Under $\mathbf{F}$, a unit length vector in direction $\mathbf{v}$ is stretched to a new length $l = \|\mathbf{F}\mathbf{v}\|$. The set of all 3D vectors $l\mathbf{v}$ forms an ellipse-like shape. We find the principal axes of the ellipse (via SVD) and use them as directions of anisotropy.

Regularization Since vastly different material assignments, ${}^0\mathcal{M}_k$, can produce the same metamaterial, our naïve cost function (Eq. 3) can produce very large parameter values and even non-physical negative ones. For example, consider a homogeneous material assignment at the high-resolution level. The same metamaterial can be achieved by interleaving hard and soft materials at each fine element or by assigning a single, well chosen material to all fine elements. To overcome this, we add a regularization term to control the parameter ranges and prevent overfitting of the training samples. Our modified error function takes the following form:

$$ \sum_{s=1}^{n_s} \left({}^0V_s - {}^1V\left(\boldsymbol{r}\left({}^0\boldsymbol{u}_s\right), {}^1\boldsymbol{p}\right)\right)^2 + \lambda \sum_k ({}^1\boldsymbol{p}_k - {}^0\boldsymbol{p}_k)^2, \qquad (4) $$

which prevents material parameters from deviating too far from ${}^0\mathcal{M}_k$. We chose the regularization constant $\lambda = 0.01$ for the results in this paper. In our experiments, since the base energy functions are linear with respect to the material parameters, the fitting problem can be solved by linear regression with regularization.

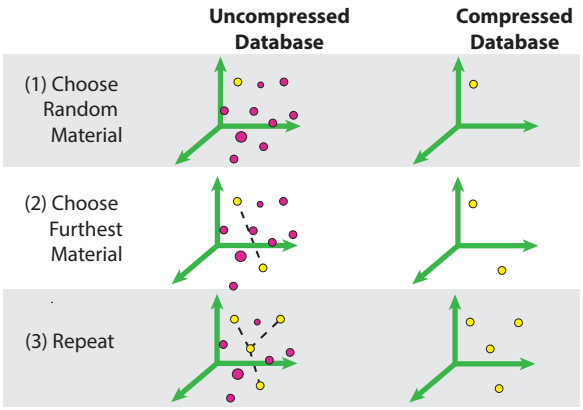


Figure 8: *Database compression step adds metamaterials to a compressed database by a farthest point sampling strategy.*

4.4 Database Compression

Given n materials in the palette, the number of material combinations in a 2×2×2 block is n^8. In modern hardware, it is impossible to compute and store all material combinations even for a moderately-sized palette with 100 materials. In order to compress the number of materials stored in our metamaterial database, we select a small number of representative material combinations and remove all others. We compare materials in a feature space. In order to construct metamaterial feature vectors, we first select a common subset of deformations from all computed deformation samples. We then evaluate the potential energies of each metamaterial at each deformation sample. The stacked vector of energies becomes a metamaterial feature vector.

Since our base materials differ in stiffness by orders of magnitude, we take the logarithm to measure the difference in ratio. Let D be the L^2 norm of log-energies between the two materials given by

$$ D(A,B) = \sqrt{\sum_s (\log({}^0V_s^A) - \log({}^0V_s^B))^2}, \qquad (5) $$

where A and B denote two distinct metamaterials in the database. Given the distance metric, we can select k representatives materials using farthest point sampling [Eldar et al. 1997]. We randomly choose an initial metamaterial and then repeatedly select the material combination furthest away from any previous representatives – continuing until we obtain k representatives (Fig. 8). This compression algorithm chooses k samples that equally cover the metamaterial energy space, helping to preserve good behavior in our coarse simulations.

4.5 Hierarchical Coarsening

While one level of coarsening can yield significant speed-ups, DDFEM can also be applied hierarchically. As discussed in §4.4, the exponential growth of metamaterials palettes at each level makes it

prohibitively expensive to perform fitting. We address this by changing our coarsening strategy. Instead of choosing ${}^0\mathcal{E}$ to be a $2\times2\times2$ block we choose it to be a $2\times1\times1$ block, which we coarsen. We construct an intermediate database of materials and compress. We then choose ${}^0\mathcal{E}$ to be a $1\times2\times1$ block, coarsen and compress, and finally a $1\times1\times2$ block, coarsen and compress. Intermediate compression greatly reduces the number of samples we need to generate in order to populate the material parameter database for the next coarsening level. It is important to note that our intermediate databases only store lookup tables which allow us to extract appropriate material IDs for the next coarsening stage. Material parameters need only be stored in the final database since it is these elements that are simulated.

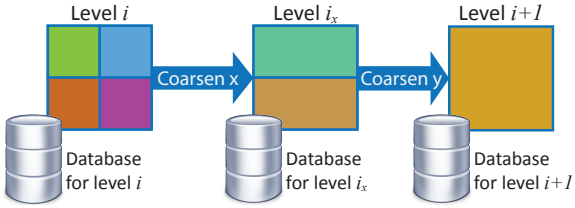


Figure 9: *Hierarchical coarsening operates on one dimension at a time, performing clustering at each intermediate stage (here denoted i_x). This allows our compression algorithm to be applied aggressively, greatly reducing the number of energy samples we need for fitting material parameters.*

5 Runtime Simulation

Once our metamaterial database, ${}^1\mathcal{P}$, has been constructed we can use it to perform fast online coarsening. Initially, the user loads geometry which is embedded in a hexahedral grid for simulation. Prior to simulation we iterate over all $2\times2\times2$ blocks of hexahedral elements and perform mesh coarsening by replacing these 8 elements with a single coarse element. We perform a database lookup into ${}^1\mathcal{P}$, using the material ID numbers of the 8 original elements, to quickly retrieve the optimal metamaterial for this coarse element. Database lookup is fast (even using our unoptimized, single-threaded implementation), and this is what makes DDFEM so appealing. We achieve significant simulation speed-up from coarsening, retain accuracy in the simulation, and reduce the cost of material coarsening at runtime to a negligible amount. Our material model can be used in any simulation algorithm suitable for non-linear elasticity. In our experiments, we use Coin-IpOpt [Wächter and Biegler 2006] to implement static and dynamics simulations with tolerance ("tol" option) set to 0.5. We use Pardiso as our linear solver. For timing purposes, we limit Pardiso to single thread mode. The pseudo-code for static simulation is shown in Alg. 1.

6 Results and Discussion

All the results shown here are simulated using nonlinear constitutive models at the fine scale. This and coarsening speed are the key differentiating factors between DDFEM and other coarsening algorithms such as Nesme et al. [2009] and Kharevych et al. [2009]. Our database starts with three Neo-hookean base materials with Young's modulus $1e5, 1e6, 1e7$ and Poisson's ration 0.45. For comparison with 3D-printed objects, we used two base materials with measured Young's moduli. We use 500 force directions, and sample 5 magnitudes in each direction, resulting in 2500 force samples for each material combination. In addition, we generate 500 stretching samples for computing the direction of anisotropy. During fitting, we use shear modulus and Lamé's first parameter, as well as the spring stiffness. We repeat the same process for the second level of coarsening, using 6561 materials in the first level as base materials.

Algorithm 1: Static Simulation

repeat
  $\mathbf{f}$: global force vector
  L: triplet list for global stiffness matrix
  for each element e **do**
    compute elastic force $\mathbf{f}_e$
    add $\mathbf{f}_e$ to $\mathbf{f}$
  end for
  add external force $\mathbf{f}_{ext}$ to $\mathbf{f}$
  for each element e **do**
    $\mathbf{K}_e$: element stiffness matrix
    for each quadrature point q **do**
      compute stiffness matrix $\mathbf{K}_q$ at quadrature point
      $\mathbf{K}_e += \mathbf{K}_q$
    end for
    append entries of $\mathbf{K}_e$ to L
  end for
  sort L to get sparse stiffness matrix $\mathbf{K}$
  set entries in $\mathbf{K}$ and $\mathbf{f}$ for fixed vertices
  $\Delta\mathbf{x} = \mathbf{K}^{-1}\mathbf{f}$
  compute step size h using line-search
  $\mathbf{x} += h\Delta\mathbf{x}$
until convergence

Example	grid size	rel sp	time/iter	iters	error
Pushing(0)	16×16×16	1.0	1.010	5	-
Pushing(1)	8×8×8	11.5	0.087	5	8.91e-4
Pushing(2)	4×4×4	31.4	0.032	5	1.36e-2
Bending(0)	8×32×8	1.0	0.270	28	-
Bending(1)	4×16×4	12.6	0.028	22	5.60e-2
Bending(2)	2×8×2	22.7	0.015	22	8.88e-2
Twisting(0)	8×32×8	1.0	0.300	16	-
Twisting(1)	4×16×4	15.2	0.031	10	1.56e-2
Twisting(2)	2×8×2	20.7	0.019	12	3.28e-2
Buckling(0)	128×8×16	1.0	8.85	32	-
Buckling(1)	64×4×8	50.1	0.28	20	7.24e-3
Buckling(2)	32×2×4	331.8	0.12	7	3.14e-2
Fibers(0)	32×100×32	1.0	193.85	17	-
Fibers(1)	16×50×16	51.2	4.95	13	2.94e-2
Fibers(2)	8×25×8	489.5	0.96	7	4.26e-2
Bridge(0)	56177	1.0	43.44	14	-
Bridge(1)	9727	8.4	4.88	15	4.39e-3
Bridge-arch(0)	65684	1.0	54.99	3	-
Bridge-arch(1)	11695	8.1	7.84	3	3.68e-4
George(0)	46152	1.0	52.19	23	-
George(1)	6755	16.4	3.49	21	2.86e-2
George-bone(0)	46152	1.0	41.35	12	-
George-bone(1)	6755	13.2	2.70	14	2.99e-2

Table 1: *Relative performance, absolute performance in seconds and average vertex error relative to the bounding box size for full-resolution and coarsened simulations. Relative performance illustrates the performance increase gained by coarsening with respect to the time taken for the high-resolution static simulation to converge. Bracketed numbers after each example name indicate the number of coarsening levels with 0 indicating the high-resolution simulation. All computation times are recorded using Coin-IpOpt running in single threaded mode on a 2.5 GHz Intel Core i7 processor.*

We select 400 representatives at each intermediate level.

6.1 Database

One advantage of our compact metamaterial representation is the small amount of storage it requires. In fact we require only $6 \times 8 = 48$ floating-point values for each material at the first coarsening level and $6 \times 64 = 384$ values for the second level. (For each

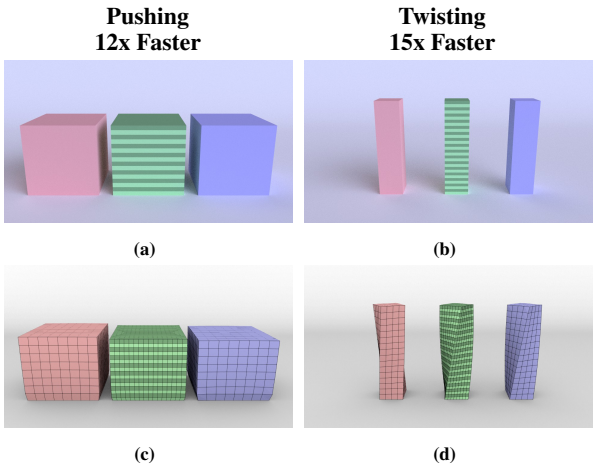


Figure 10: *Examples of pushing a cube (a - Initial State, c - Compressed) and twisting a bar (b - Initial State, d - Compressed), both with heterogeneous material distribution. We compare DDFEM to Naïve Coarsening and the ground-truth, High-Res Simulation. We render wire frames to show the simulation meshes.*

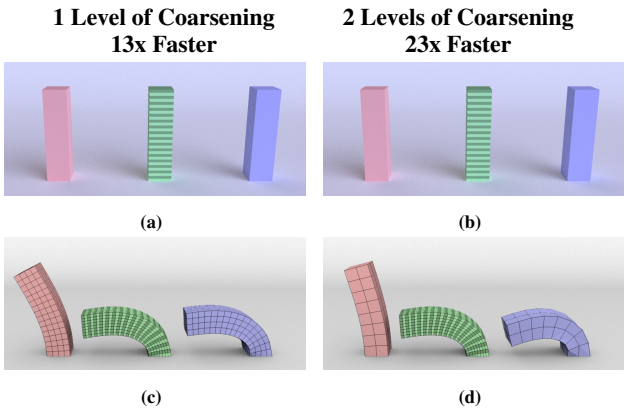


Figure 11: *Bending a heterogeneous bar: We compare DDFEM to Naïve Coarsening and a High-Res Simulation. Subfigures (a, c) show comparison for 1 level of coarsening, and (b, d) show 2 levels of coarsening. The naïve coarsening approach results in a much stiffer behavior, whereas our fitted model more closely approximates the fine model.*

finer element, ${}^0\boldsymbol{p}$ contains 2 material moduli plus $C, \boldsymbol{v}$.) For further recursive levels, we can limit ourselves to 320 values per material. Our current 3 material database is 4 megabytes in size.

6.2 Simulation Results

We show results from elastostatic simulations performed using DDFEM. We also demonstrate its performance advantages over high-resolution simulations. We render wire frames to show the discretizations of the high-resolution and coarse meshes. We first show examples of two simple simulations, the pushing and twisting of a rectangular object with heterogeneous, layered material distribution (Fig. 10). Note that in all cases DDFEM qualitatively matches the behavior of the high-resolution simulation. We also compare the performance of DDFEM to a naïve coarsening method that uses the material properties from $2\times2\times2$ element blocks of the high-resolution simulation mesh at each corresponding quadrature point. In our supplemental video we compare to a second baseline model which averages material parameters inside each coarse element. This average model is less accurate than the Naïve model in all cases.

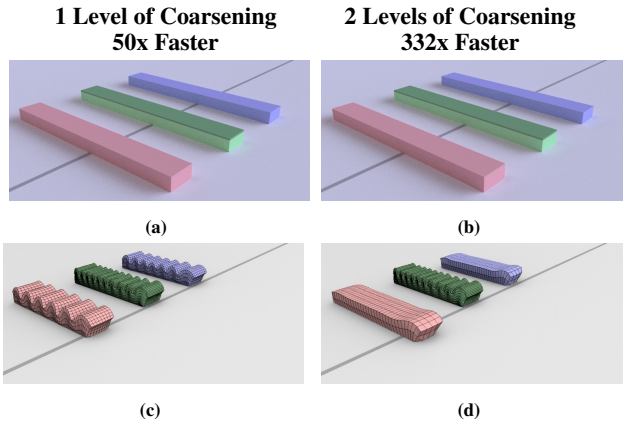


Figure 12: *Compressing a heterogeneous slab using Naïve Coarsening (1 level and 2 levels of coarsening), DDFEM (1 level and 2 levels of coarsening) and a High-Res Simulation. The top, darker layer is stiffer, causing the object to buckle. The bottom vertices are constrained to stay on the floor. Figure (a,b) shows the slabs before compression, figure (c,d) shows the slabs after compression and figure. Notice that, after 1 level of coarsening, Naïve Coarsening neither compresses nor buckles as much as either DDFEM or High-Res Simulation. After 2 levels of coarsening, the buckling behavior is lost. The Naïve Coarsening fails to capture the compressive behavior of High-Res Simulation, whereas DDFEM does.*

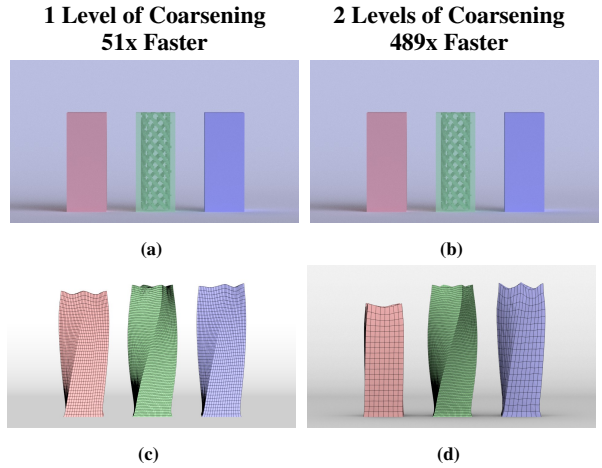


Figure 13: *Simulating a bar with an embedded set of fibers using Naïve Coarsening, DDFEM and a High-Res Simulation. Note that DDFEM captures the characteristic twisting motion of the bar better than Naïve Coarsening. (a,b) shows the initial state of both bars while (c,d) shows the deformed state after pulling on the top of the bars.*

Naïve approaches often exhibit pathological stiffness for heterogeneous materials (illustrated by the lack of compression of the box and lack of twisting of the bar) [Nesme et al. 2009]. In these cases, DDFEM yields good speed ups while maintaining accuracy. For a single level of coarsening we achieve *8 times or greater* speed ups for all examples. Performance numbers and mean errors are listed in Table 1. Since the fine simulation and the coarse simulation have different numbers of vertices, we create a fine mesh from the coarse simulation by trilinearly interpolating the fine vertices using the coarse displacements. The errors are measured by computing the average vertex distance relative to the longest dimension of the bounding box in rest shape. We also examine the behavior of DDFEM

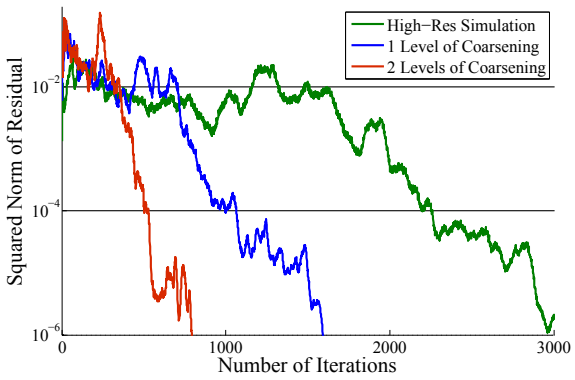


Figure 14: *Comparison of CG iterations on high-resolution and coarsened meshes of the George-bone example. The squared residual is measured as* $\|\mathbf{Kx}-\mathbf{f}\|^2$. *We observe that CG converges much faster on coarse meshes.*

Example	grid size	quad/iter (12)	time/iter (2-21)	iters
Bending(0)	8×32×8	0.11	0.27	28
Bending(1)	4×16×4	0.015	0.028	22
Bending(2)	2×8×2	0.014	0.015	22
Buckling(0)	128×8×16	1.0	8.8	32
Buckling(1)	64×4×8	0.11	0.28	20
Buckling(2)	32×2×4	0.10	0.12	7
Fibers(0)	32×100×32	10.0	193	17
Fibers(1)	16×50×16	1.0	4.9	13
Fibers(2)	8×25×8	0.68	0.96	7

Table 2: *Portion of time in seconds used by quadrature computation during static simulation in seconds. Bracketed numbers indicate corresponding lines in Algorithm 1.*

during bending (Fig. 11). Yet again the naïve coarsening method completely fails to capture the behavior of the high-resolution result, while DDFEM offers a much better approximation.

Performance Analysis Table 2 shows time spent in quadrature evaluation versus in solver at coarse and fine levels. We use 8 quadrature points for the first level of coarsening and 64 for the second level. The time for computing the local stiffness matrix for one element increases from 0.1ms to 1ms. In the second level, the speedup comes from the reduced number of elements over which to perform quadrature and the time required for the linear solver.

To further investigate the performance of our coarsened simulations we replaced Pardiso with an assembly-free Jacobi-preconditioned conjugate gradient (CG) linear solver and used this to simulate our George-bone test case. While the overall runtime of the high-resolution simulation increased from 496 to 2082 seconds (most likely do to the unoptimized nature of our solver) our coarsened model achieved 20x and 67x speedups using one level and two levels of coarsening respectively. One might expect no benefit from the second level of coarsening since the number of quadrature points remains constant. However, the number of CG iterations is roughly proportional to the number of vertices in the simulation mesh and thus the coarse model converges more quickly (Fig. 14). Since our metamaterial models are not restricted to use a fixed number of quadrature points, one could design coarse models that are more tailored towards assembly-free solvers by reducing the number of quadrature points and simplifying the strain energy expressions.

Complex material behavior DDFEM can capture the gross behavior of complex, spatially-varying material distributions. Fig. 12 shows the results of applying DDFEM to a non-linearly elastic slab

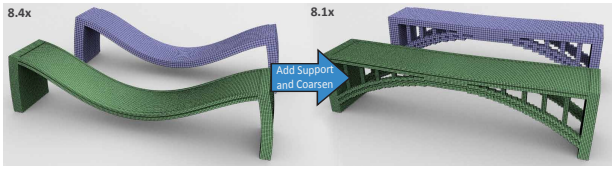


Figure 16: *Accelerating geometry change: We repair a structurally unsound bridge by adding a supporting arch (8x faster).*

with a stiff "skin." The bottom of the slab is constrained to slide along the ground with one end fixed. When force is applied to the free end of the slab, buckling occurs. DDFEM captures the gross behavior of the bar and approximates the overall amount of compression well. However, it cannot replicate the frequency of the high-resolution buckling pattern due to the coarseness of the simulation mesh. Fig. 12 also shows a comparison with 2nd level coarsening. In this case, the overall compression of the bar is still captured accurately. For this example DDFEM affords *50 times* (1 level of coarsening) and *332 times* (2 levels of coarsening) performance improvements over the high-resolution simulation. The artificial stiffness of the naïve model can be seen in the reduced buckling and compression when compared to DDFEM at both coarsening levels.

Anisotropic material distribution Next we explore the ability of DDFEM to handle highly anisotropic material distributions (Fig. 13). We embed a helical set of stiff fibers in a soft, non-linearly elastic matrix. Pulling on the object induces a twisting. Again, at one coarsening level DDFEM captures this anisotropic behavior well, much better than the naive approach, and gains a *51 times* speed up over the high-resolution simulation. Worth noting is that the DDFEM bar is slightly softer in the y-direction. This kind of inaccuracy should be expected. Since our method builds a low-dimensional approximation of a potential energy function we cannot hope to accurately reproduce the complete behavior of the high-resolution simulation. What is important is that DDFEM captures the salient global behavior, in this case, the twisting of the bar.

Geometry and material design We present three examples of using DDFEM for geometry and material design. In the first example, we edit the material composition of the sole of a running shoe in order to stabilize it. Fig. 15 shows the effect of the three material edits as well as relative speed up achieved over the full resolution simulation and coarsening time. DDFEM performance is always an order of magnitude more than that of the high-resolution simulation, and, most importantly, our coarsening times are on the order of milliseconds. We stress that our current implementation is completely single threaded and that coarsening, which in our case involves a simple database lookup, is inherently parallel. In the second example, we add a supporting arch to a bridge. Prior to the addition of the support structure, the bridge sags catastrophically. The fast coarsening of DDFEM allows us to achieve an 8 fold increase in simulation performance using a single coarsening pass. In the third example, we add a rigid skeleton to a deformable character (George) in order to control his pose. Here our single threaded, data-driven coarsening only takes 200ms.

Dynamics Though the examples shown in this paper focus on static analysis, DDFEM is equally applicable to dynamic simulations. At its core, DDFEM simply supplies new, more accurate material models for use during simulation. This makes the method useful for accelerating various animation tasks as well. In the accompanying videos we show a dynamic simulation of our fiber embedded bar, computed using a standard linearly-implicit time integrator.

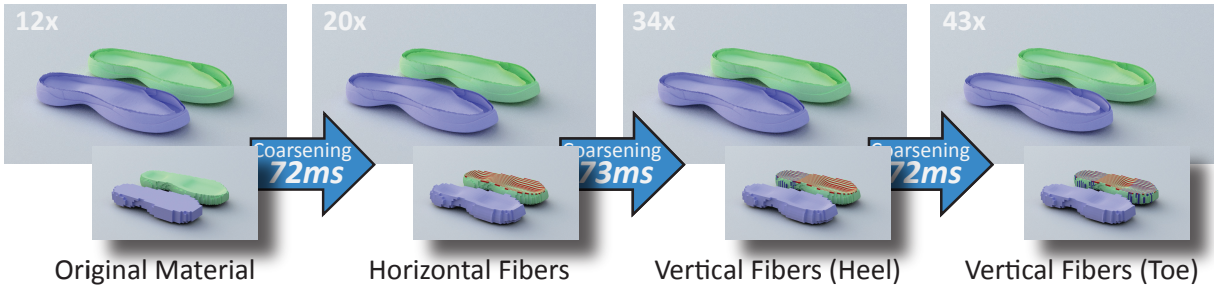


Figure 15: *Designing a shoe sole: We compare the performance of DDFEM to that of High-Res Simulation in the context of a material design problems. Large images show the effect of material changes on the sole of the shoe, which is being deformed under a "foot-like" pressure field. Inset images show the materials assigned to the shoe sole and the embedded finite element simulation mesh. Numbers within arrows show coarsening times between editing steps and the numbers in the upper left corner of each image show the relative performance of DDFEM to High-Res Simulation. While the DDFEM sole is made up of many metamaterials, we display it as a single color to distinguish it from the High-Res Simulation.*

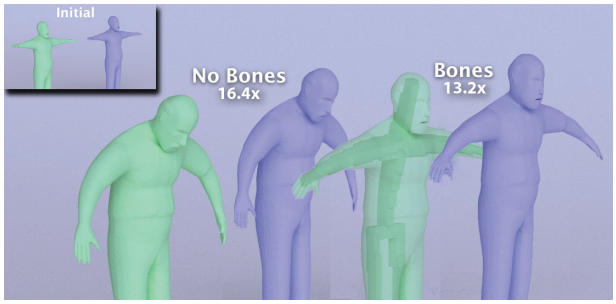


Figure 17: *Correcting George's posture using a rigid skeleton (High-Res Simulation and DDFEM).*

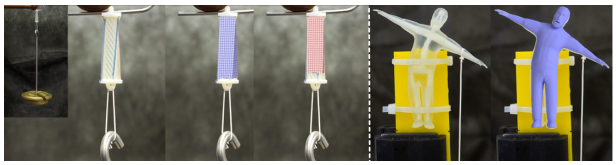


Figure 18: *A comparison of DDFEM (2 levels of coarsening) to real world deformations of 3D printed, multi-material designs. DDFEM captures the twisting behavior of an anisotropic bar much more accurately than Naïve Coarsening. Similarly DDFEM accurately predicts the deformation of our heterogeneous George character.*

6.3 Fabricated Results

Finally, we test the accuracy of our simulation against fabricated results, created using a Stratysys Object Connex 500 multimaterial 3D printer. We fabricated a bar with embedded helical fibers as well as our George character and applied specified loads to both. We show qualitative comparison of the deformed configurations of these real-world examples to our simulated results (*2 levels of coarsening*-Fig. 18). Note that the simulation does an excellent job of predicting the deformed configuration of both objects.

7 Limitations and Future Work

Because DDFEM relies on a database compression step to combat the combinatorial explosion of metamaterials, accuracy is heavily influenced by the set of representative metamaterials. Finding a better way to select metamaterial structures is an interesting area of future work. Second, in our attempt to make our method geometry independent, some accuracy when dealing with partially filled boundary finite elements is sacrificed. Adding a parameterized boundary representation to the method, in order to more correctly handle non-axis aligned boundary conditions, could also be explored. Third, the method acts on discrete materials. While we believe that this is reasonable, considering the way that engineers and designers approach material design, a method that coarsens continuous spaces of non-linear materials could be beneficial.

Many avenues of future work are promising. First, one could explore topologically aware meshing (e.g. Nesme et al. [2009]) to allow better handling of models with large empty regions. In fact shape function learning approaches, such as Nesme et al. [2009] could be combined with our material learning approach to produce even more accurate simulations. Including these shape functions in our database could, for instance, allow us to capture the wrinkles in our buckling example. Second, extending DDFEM to more complex material models, such as those involving plasticity, would be a useful exercise. Third, DDFEM can be combined with an adaptive voxel grid as well as other dynamic meshing approaches to obtain further speed-ups. Finally, exploring hierarchical solvers based on DDFEM coarsening is a very attractive direction. Solvers such as multigrid methods rely on good coarse approximations to accelerate fine scale simulations. Using DDFEM for these approximations could improve the convergence rate, and thus the performance of such algorithms.

8 Conclusions

In this paper we presented the data-driven finite element method, a sampling-based coarsening strategy for fast simulation of non-linearly elastic solids with heterogeneous material distributions. Our method is unique in that changing material parameters requires no additional computation at runtime, making it well-suited for iterative design problems. We have shown the utility of our method on several examples wherein it garners a *8 to 489 times* speed-up over corresponding high-resolution simulations.

Acknowledgements

Thank you to the Anonymous Reviewers, Etienne Vouga, Maria Shugrina and Melina Skouras for helpful suggestions and James Thompkin for video assistance. This research was funded by NSF grant CCF-1138967 and DARPA #N66001-12-1-4242.

References

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

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

BENDSOE, M., AND SIGMUND, O. 2003. *Topology Optimization: Theory, Methods and Applications*. Engineering online library. Springer.

BICKEL, B., BÄCHER, M., OTADUY, M. A., MATUSIK, W., PFISTER, H., AND GROSS, M. 2009. Capture and modeling of non-linear heterogeneous soft tissue. *ACM Trans. Graph. 28*, 3 (July), 89:1–89:9.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph. 29*, 4 (July), 63:1–63:10.

BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July).

CHEN, D., LEVIN, D. I. W., DIDYK, P., SITTHI-AMORN, P., AND MATUSIK, W. 2013. Spec2Fab: A reducer-tuner model for translating specifications to 3D prints. *ACM Trans. Graph. 32*, 4 (July), 135:1–135:10.

CLAUSEN, P., WICKE, M., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph. 32*, 2 (Apr.), 17:1–17:15.

COTIN, S., DELINGETTE, H., AND AYACHE, N. 1999. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics 5*, 1, 62–73.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH '01*, Annual Conference Series, 31–36.

ELDAR, Y., LINDENBAUM, M., PORAT, M., AND ZEEVI, Y. Y. 1997. The farthest point strategy for progressive image sampling. *Image Processing, IEEE Transactions on 6*, 9, 1305–1315.

FARHAT, C., AND ROUX, F.-X. 1991. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering 32*, 6, 1205–1227.

FARMER, C. 2002. Upscaling: a review. *International Journal for Numerical Methods in Fluids 40*, 1-2, 63–78.

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July), 73:1–73:10.

GILLES, B., BOUSQUET, G., FAURE, F., AND PAI, D. K. 2011. Frame-based elastic models. *ACM Trans. Graph. 30*, 2 (Apr.), 15:1–15:12.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. Charms: A simple framework for adaptive simulation. *ACM Trans. Graph. 21*, 3 (July), 281–290.

GUEDES, J., AND KIKUCHI, N. 1990. Preprocessing and postprocessing for materials based on the homogenization method with adaptive finite element methods. *Computer methods in applied mechanics and engineering 83*, 2.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A. W., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 Papers*, ACM, New York, NY, USA, SIGGRAPH '11.

KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph. 28*, 3 (July), 51:1–51:8.

KOU, X., PARKS, G., AND TAN, S. 2012. Optimal design of functionally graded materials using a procedural model and particle swarm optimization. *Computer-Aided Design 44*, 4, 300–310.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering 51*, 4, 479–504.

LI, S., HUANG, J., DE GOES, F., JIN, X., BAO, H., AND DESBRUN, M. 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Trans. Graph. 33*, 4 (July).

MANDEL, J. 1993. Balancing domain decomposition. *Communications in numerical methods in engineering 9*, 3, 233–241.

MARSDEN, J., AND HUGHES, T. 2012. *Mathematical Foundations of Elasticity*. Dover Civil and Mechanical Engineering.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July), 37:1–37:12.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. 31*, 6 (Nov.), 152:1–152:10.

NARAIN, R., PFAFF, T., AND O'BRIEN, J. F. 2013. Folding and crumpling adaptive sheets. *ACM Trans. Graph. 32*, 4 (July), 51:1–51:8.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3 (July), 52:1–52:9.

PATTERSON, T., MITCHELL, N., AND SIFAKIS, E. 2012. Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Trans. Graph. 31*, 6 (Nov.), 197:1–197:10.

PERAIRE, J., PEIRO, J., AND MORGAN, K. 1992. A 3D finite element multigrid solver for the Euler equations. *AIAA paper 92*, 0449.

SEILER, M., SPILLMANN, J., AND HARDERS, M. 2012. Enriching coarse interactive elastic objects with high-resolution data-driven deformations. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 9–17.

SHABANA, A., 1991. Theory of vibration, vol. ii.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph. 32*, 4 (July), 82:1–82:9.

SZABÓ, B., DÜSTER, A., AND RANK, E. 2004. The p-version of the finite element method. *Encyclopedia of computational mechanics*.

WÄCHTER, A., AND BIEGLER, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming 106*, 1.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29*, 4 (July), 49:1–49:11.

XU, H., LI, Y., CHEN, Y., AND BARBIČ, J. 2014. Interactive material design using model reduction. Tech. rep., University of Southern California.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2 (Apr.), 16:1–16:18.