A peer-reviewed version of this preprint was published in PeerJ on 25 November 2015.

View the peer-reviewed version (peerj.com/articles/cs-35), which is the preferred citable publication unless you specifically need to cite this preprint.

Klimach HG, Zudrop J, Roller SP. 2015. Generation of high order geometry representations in Octree meshes. PeerJ Computer Science 1:e35 https://doi.org/10.7717/peerj-cs.35

Generation of high order geometry representations in octree meshes

Harald Klimach[1], **Jens Zudrop**[1], **and Sabine Roller**[1]

[1]**University of Siegen, Hölderlinstr. 3, 57076 Siegen, Germany**

ABSTRACT

A robust method to obtain high order approximations of material distributions from arbitrary surface descriptions is presented. The method is used in the context of mesh generation with an octree structure and the resulting mesh and material distribution is intended to be used in a discontinuous Galerkin solver. The method is described in detail, its properties are discussed and some results from applying it to various geometries is shown. Its implementation is available in our open-source mesh generator *Seeder*.

Keywords: High order, discontinuous Galerkin, mesh generation, polynomial approximation

INTRODUCTION

High order approximations are attractive for numerical simulations on modern computing systems, due to their fast error convergence. They can solve complex problems accurately with few degrees of freedom and therefore, low memory consumption. As memory is an expensive resource, this is an important property on modern computing systems.

A relatively recent numerical method is the discontinuous Galerkin finite element method. It is gaining popularity in a wide range of application domains to solve partial differential equations. An introduction and overview is offered by Hesthaven and Warburton (2007). Besides the possibility to use high order schemes, they offer the additional advantage of high locality, as volume operations appear only locally within elements and exchanges are restricted to information on element surfaces. This property nicely fits the demands of massive parallel and distributed compute systems. Thus, a high order discontinuous Galerkin finite element discretization is a good candidate to solve large scale problems.

However, one drawback of high order methods is the need for appropriate descriptions of geometrical setups in the simulation with the same approximation accuracy as the numerical scheme. The presented method is intended to describe inhomogeneous material distributions. Specifically, we consider nonsmooth material distributions. This appears for example in the field of Maxwell equations with spatially non-constant dielectric parameters:

$$\nabla \cdot (\varepsilon \mathbf{E}) = \rho^e \tag{1}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2}$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0 \tag{3}$$

$$\frac{\partial \varepsilon \mathbf{E}}{\partial t} - \nabla \times (\mathbf{B}/\mu) = -\mathbf{j} \tag{4}$$

The electric field $\mathbf{E}$ and the magnetic field $\mathbf{B}$ depend not only on initial and boundary conditions, but also on the permittivity ε and permeability μ. Our goal is the conversion of surface descriptions that separate regions of different materials into functions of space $\varepsilon(x,y,z)$ and $\mu(x,y,z)$. Though, we will only consider the Maxwell equations in this work, the method is generic and can be used for any partial differential equation with spatially varying material parameters.

The target numerical method is the discontinuous Galerkin method with polynomial basis functions. Polynomials are therefore also used for the material distributions. However, other functions could also be generated by the described approach. The material representation is obtained during the mesh generation

for the discontinuous Galerkin solver and is attached as additional information to the elements of the mesh. An implementation of this approach is available as open source in the mesh generator *Seeder* (Klimach et al., 2015). It takes surface triangulations in form of STL (White, 2013) files and produces Octree based meshes with cubical elements and the high order polynomial geometry description.

RELATED WORK

Our approach is most closely related to embedded boundaries, as used in spectral discretizations. Examples for such approaches are the *Spectral Smoothed Boundary Method* (Bueno-Orovio and Pérez-García, 2006) and the *Fourier Spectral Embedded Boundaries* (Sabetghadam et al., 2009). Typically, they rely on simple domains and deploy regular Cartesian meshes for the sampling of geometrical information. We basically extend this concept with an octree mesh, where this method is applied in each of the elements of the mesh. Within the elements, we also make use of the octree bisection algorithm to gain a fast identification of surfaces and minimizing the work and data in the voxelization of the geometry.

Other methods, where actual meshes are deployed with an internal geometry representation are typically referred to as *Immersed Boundary Methods*, introduced by Peskin (2002), this approach has been improved and extended since. An overview to these methods is for example provided by Mittal and Iaccarino (2005). While these methods rely on meshes, they are typically targeting low order schemes. Our goal on the other hand is a mesh tailored towards the needs of high order discontinuous Galerkin methods. Thus, it can be understood as a middle ground between the two approaches, offering the advantages of both. It enables the exploitation of the fast convergence in spectral approximations but still allows for complex computational domains.

Discontinuous Galerkin methods are suitable for unstructured, irregular meshes and another path towards high order surface representations is the deformation of elements to obtain superparametric, body fitted boundaries. Hindenlang et al. (2015) offers a method in this direction specifically designed for discontinuous Galerkin discretizations. However, the identification of such curved boundaries is much more complex and usually not used for internal interfaces like material changes, as both sides of the interface need to be considered. Such deformed, unstructured elements are also subject to varying mesh quality and prone to issues with geometrical constraints.

THE SEEDER MESH GENERATOR

Seeder (Harlacher et al., 2012) is an octree mesh generator. It basically produces voxelizations of complex geometries defined by surface triangulations. By voxelization we refer to the process of subdividing a given volume into smaller cubical elements (voxels). With the octree approach, these cubical elements are successively split into 8 smaller cubes, where needed. An example for the voxelization of a sphere is shown in Figure 1. *Seeder* is freely available online (Klimach et al., 2015) under a permissive BSD license and has been successfully compiled and applied on a wide range of architectures. In the following section the general voxelization method is briefly outlined. Afterwards the extensions to enable material definitions are explained.

Basic mesh generation procedure

To produce the voxelization, *Seeder* deploys an approach similar to the building cube method by Ishida et al. (2008). The basic idea is an iterative refinement towards geometry surfaces, followed by a flooding of the computational domain starting from a user defined seed. This flooding is limited by elements intersected by boundary objects and all flooded elements finally constitute the actual computational domain. For the refinement a bisection in each direction is used in each step, resulting in an octree mesh. Such tree structures are well established and wide spread in mesh generators to identify and sort geometrical objects fast, see for example Yerry and Shephard (1984) for an early adoption.

In *Seeder*, each geometry has some refinement level, defined by the user, attached to it. This level describes how many bisection steps should be done to resolve the surface. The higher this level the smaller the voxels to approximate the surface. Elements are refined iteratively if they intersect a geometry, until the desired refinement level is reached. After this step of boundary identification, the actual computational domain is identified by a 3D flood filling algorithm. This flooding is bound by all elements intersected by a geometry. To avoid unintentional spills, only direct neighbors are considered for flooding (*von Neumann neighborhood*). This mechanism, even though it requires the definition of seeds by the user, is chosen

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

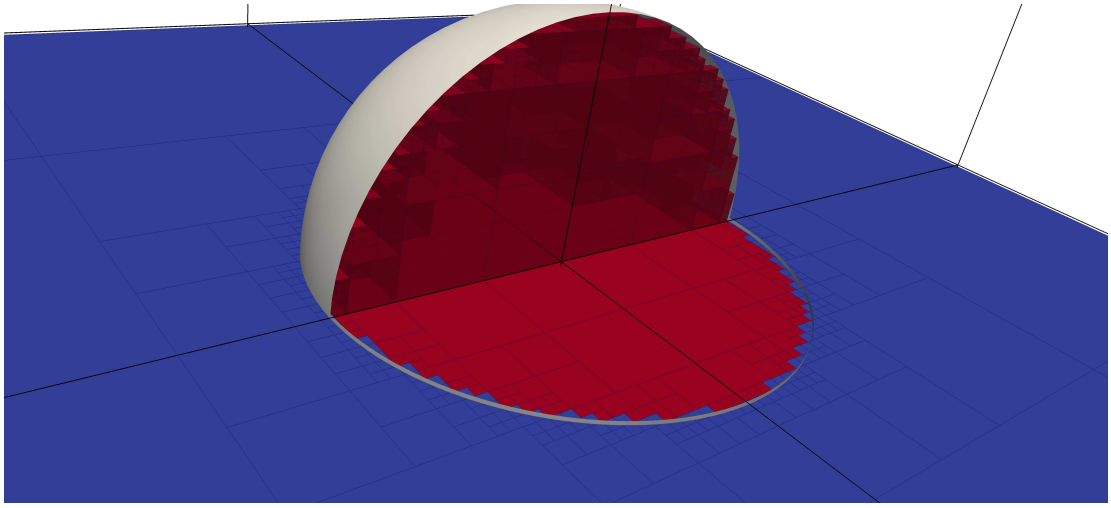

Figure 1. Illustration of the voxelization of a sphere within original mesh elements. The sphere is indicated by the white surface, while the thick black lines outline the elements of the actual mesh. The voxelization within elements follows the octree refinement towards the sphere and is indicated by the dark blue lines. Inside the sphere, voxels have been colorized by the flood-fill mechanism with a seed in the center.

as it provides a high robustness and indifference towards the triangle definitions in STL files. Small inaccuracies in the geometry definition are automatically healed, as long as they are below the resolution of the voxelization. This approach has proven to be robust and applicable to a wide range of complex geometries.

GENERATION OF POLYNOMIAL GEOMETRY APPROXIMATION

The cubical elements obtained by the mesh generation procedure described in the previous section, provide the frame wherein the high order surface representation can now be constructed. This basically means, we need to find a representation in the function space of the solver that approximates the geometrical information best. For material interfaces we need to deal with discontinuities as the material property jumps at the interface. In Figure 2 such a discontinuity and its approximation by an increasing number of Legendre polynomials is shown. While in this simple 1D example, the projection can be computed analytically, this is not possible anymore for higher dimensions with arbitrary jump definitions. We therefore introduce an algorithm in this section to find approximations of the projection numerically.

However, before polynomials can be computed, a concept to selectively attach attributes to regions of the mesh is needed to enable the definition of objects with different material. We can exploit the flood filling algorithm of the mesh generation for this requirement by using multiple fillings, which are confined by different geometrical objects respectively. This is similar to coloring an image, and we refer to those fillings as colors. In the following we briefly discuss the coloring concept and then move on to the generation of high order surface representations within the octree mesh.

Coloring

To allow for the definition of spatial areas with different properties, *Seeder* deploys not just a single flooding indicator, but multiple. We refer to them as colors. Each color requires its own seed definition and boundaries of the same color. The flooding spreads from the seeds and is limited by the boundaries. Boundaries of other colors do not affect the flooding and it is possible to have elements flooded by multiple colors. Colors provide a method to distinguish specific areas in the computational domain, such that the solver can attach individual material properties to them.

Subresolution

With the coloring principle described above, we now are equipped to define arbitrary material areas, but we still need to obtain high order surface approximations in the elements of the octree. As this provides

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

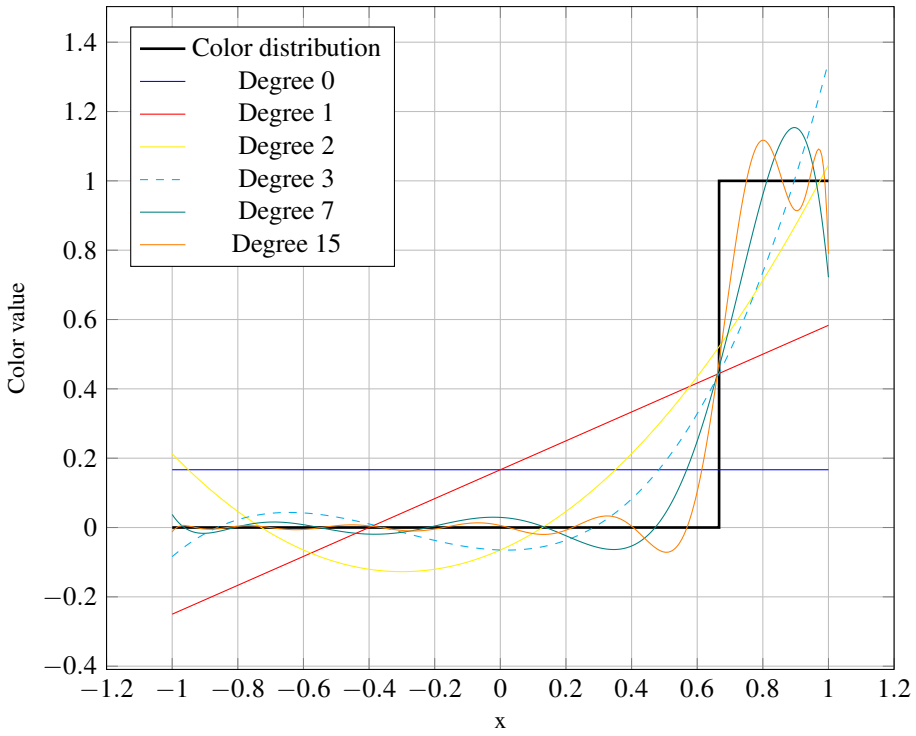

Figure 2. Projection of a step function, jumping at $x = \frac{2}{3}$ onto the space of Legendre polynomials. Shown is the step function along with its approximation by more and more Legendre basis functions obtained by analytical integration.

information beyond the resolution of the actual mesh, we refer to this as subresolution. The subresolution provides a more detailed surface description via polynomials in elements of the mesh that are intersected by colored boundaries. The used strategy consists of the following three steps:

- **Voxelization** (and flooding) within elements of the final mesh to identify color distributions
- **Probing** of integration points for their color status in the elements
- **Conversion** of point data to polynomial modes

Figure 1 illustrates the concept for a sphere. The sphere is shown by the white surface and cut open to reveal the voxelization within. Thick, black lines indicate the actual mesh elements, and the finer blue lines show the subresolution voxels within them. As described above, elements in the interior are flooded, which is indicated by the red coloring. The flooding is limited by the sphere and all elements outside the geometry are not flooded by this color.

To describe the procedure in greater detail, we will use a setup in one dimension for a single element. A sketch of this overall scheme in one dimension is provided in Figure 3. It makes use of the same target step function, as in Figure 2, but outlines the individual steps we take to arrive at a numerical approximation of the analytical projection. The numerical scheme will enable the discretization of higher dimensional problems and arbitrarily complex geometries.

First, we need a way to describe the color distribution in greater detail than offered by the elements of the final mesh. Luckily, we already have a robust and fast method to identify color boundaries in volumes by the above described **voxelization** method. Thus, we can just deploy that mechanism with iterative refinement and flood filling within the elements for the final mesh. This is achieved by marking elements for final mesh output when the target resolution of the geometry is reached, but continuing the refinement for a given number of additional levels. The number of additional refinement levels L can be freely chosen in the configuration, but for accurate approximations it is related to the number of Chebyshev nodes N. The minimal distance between the first Chebyshev node and the element boundary is proportional to N^{-2}

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

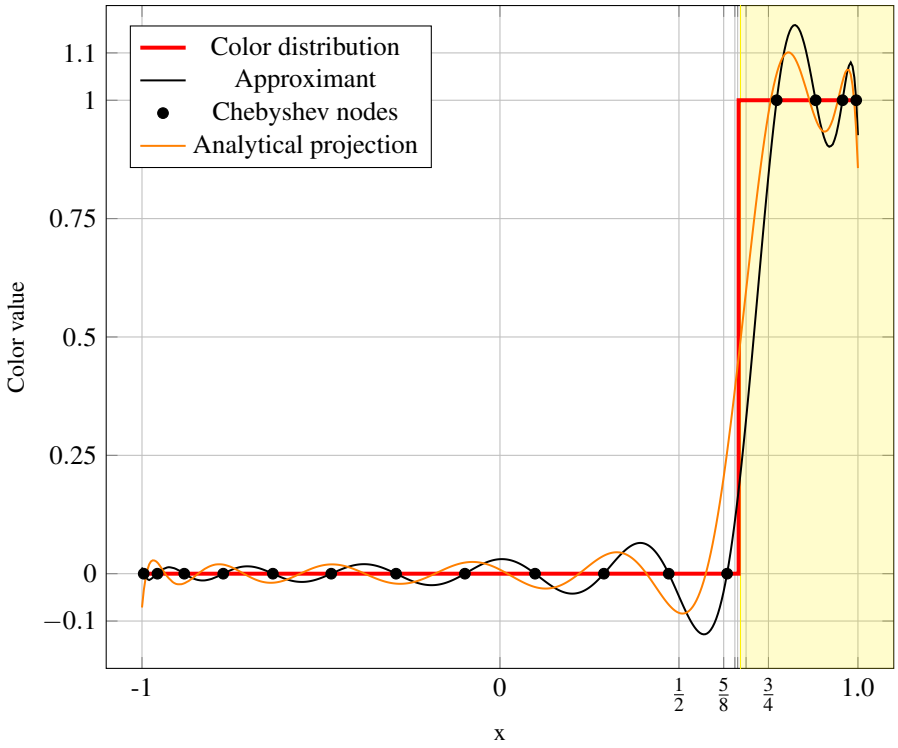

Figure 3. Illustration of the approximation method in 1D. For a single element and one discontinuity. The color value jumps from 0 to 1 at $x = \frac{2}{3}$, and is indicated by the red line. The bisection sequence is indicated by the grid lines, and the yellow area highlights the region, where the color value is identified to be 1 by the bisecting approximation. An approximant polynomial of degree 15 is constructed from the 16 shown Chebyshev nodes. The orange polynomial shows the analytical projection with degree 15, also depicted in Figure 2.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

and the length of the smallest voxel within the element is given by 2^{-L}. To resolve all node distances, it is therefore necessary to choose the number of additional levels according to

$$L \geq \lceil 2\log_2(N) \rceil \tag{5}$$

We can observe, that even though the voxelization is only a first order approximation, it is feasible to accurately represent the surface due to the exponential nature of the bisection approach in the octree. After all voxels are known, the mesh generation algorithm proceeds with flooding as described in the previous section. As the additional levels are integrated into the overall mesh, no special measures need to be taken at this point. The subresolution within the elements is automatically flooded along with the rest of the mesh elements. Figure 1 illustrates the mesh status after refinement and flooding. It shows an element of the final mesh that is intersected by a sphere.

Once the flooding status of all voxels is known, the status for each Chebyshev node can be **probed**. For this, numerical values need to be associated with the flooding status for each color. Usually we assume a value of 1 for flooded voxels, and a value of 0 for not flooded voxels. Due to the spatial discretization with an octree, the location of each Chebyshev node can be found fast with logarithmic computational complexity.

Finally, the **conversion** of the nodal information to a suitable function space for the solver has to be done. A typical choice for discontinuous Galerkin methods are Legendre polynomials. To obtain the Legendre modes, we apply the fast polynomial transformation proposed in Alpert and Rokhlin (1991). However, other target functions with different point sets could also be plugged into the described machinery. With this step, we now have a volumetric description that yields a high order approximation of the surface. Note, that only intersected elements need to get this information added, all other elements have constant colors. Thus, the need for volume information is limited to a small area around the surface.

To summarize the subresolution algorithm, we walk through the necessary steps in a 1D example with a single step in the color distribution. This example is shown in Figure 3 and the starting point is the step function indicated by the red line. In 1D the reference element is the interval [-1,1], and the color distribution jumps from 0 to 1 at $x = \frac{2}{3}$. First, the surface is approximated by bisecting refinements towards the location of the jump as indicated by the grid. Then a flood filling is applied, which identifies the region with coloring, indicated by the yellow area in Figure 3. With 5 bisections in the voxelization, this results in an approximation of the jump location at $x = 0.671875$. To obtain a polynomial of degree 15, we employ 16 Chebyshev nodes, as indicated by the black dots. The value at each node is obtained by checking the flooding status of the interval in which the node is found. Finally, the approximating Legendre polynomial is found by a fast polynomial transformation. Due to the discontinuity of the step function, the representation in polynomial space is an infinite series and the finite approximation suffers from the Gibbs phenomenon (Wilbraham, 1848) but besides this fundamental problem, also inaccuracies due to the numerical integration can be seen as the numerical (orange) and the analytical (black) projections do not coincide. These result in a distorted location of the jump. In the next section we will have a closer look at these numerical issues.

NUMERICAL PROPERTIES

In this section we investigate the numerical properties of the described approximation method. Though, the one dimensional problem with a single jump is much simpler than a real three dimensional geometry, it is still instructive for the fundamental properties of the algorithm. Let us recall that the goal is an appropriate representation of the geometry in a high order discontinuous Galerkin solver. Typically, the employed functions in the solver are smooth within elements. Here we consider specifically Legendre polynomials, which are attractive due to their orthogonality. The representation of a non-smooth material distribution in the finite smooth function space therefore can only be approximate and the best possible approximation is found by the L^2 projection.

Table 1 shows the convergence behavior for the series of Legendre polynomials, obtained by L^2 projections of the step function at $x = \frac{2}{3}$ in the interval $[-1, 1]$. A slow convergence can be observed, which is well known for high order approximations of such discontinuous functions. But the error outside a small band around the discontinuity can be improved later on by a post-processing step, as for example shown in Gottlieb and Hesthaven (2001). The error is bound to this band at the discontinuity, and this

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

Degree	L^2-Error
0	0.527046
1	0.402538
3	0.226028
7	0.167786

Degree	L^2-Error
15	0.122322
31	0.086937
63	0.060674
127	0.043065

Degree	L^2-Error
255	0.030481
511	0.021513
1023	0.015218
2047	0.010763

Table 1. Convergence of the series of Legendre polynomials towards the step function with jump at $x = \frac{2}{3}$.

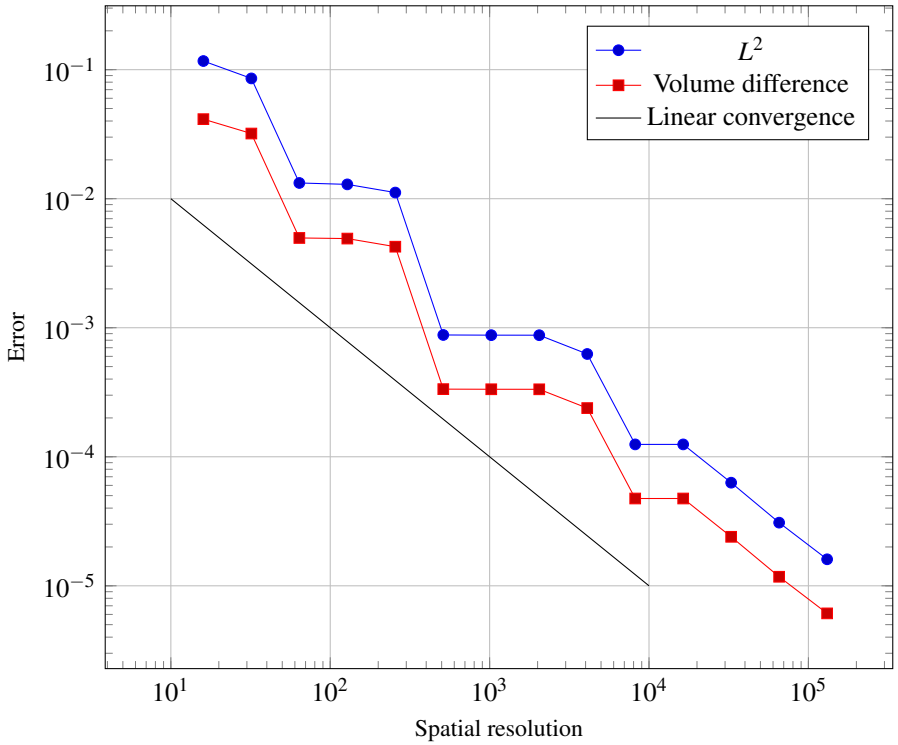

Figure 4. Error convergence of the numerical approximation towards the analytical projection for a polynomial of degree 15. The blue line shows the L^2 error over all 16 modes, while the red line shows the absolute error in the first mode, which represents the volume. On average a convergence rate of 0.973 is achieved. Keep in mind that in comparison to the actual step function, the error from Table 1 always remains.

decreases with the number of degrees of freedom. While this analytic projection can be computed for the simple setup with a single discontinuity in one dimension, this is not possible anymore in multiple dimensions and more complex geometries. Thus, we need the previously described numerical approach to approximate the projection. In the following we now first analyse how well the numerical scheme recovers the optimal solution given by the L^2 projection for the simple one dimensional discontinuity.

Figure 4 shows the convergence of the numerical procedure towards the analytical projection for polynomial space of maximal degree 15. Plotted is the error over the spatial resolution, where the spatial resolution is given by the number of Chebyshev nodes used for the numerical integration. The difference in terms of the L^2 norm to the analytical projection is represented by the blue line and covers all modes of the polynomial. With the red line, the absolute difference in the volume is provided. Note, that the volume is given by the first mode of the Legendre polynomial and thus, easily obtained. Yet, the two curves show a similar behavior, and the volume error can be used as a good indicator for the qualitative behavior of this numerical approximation. Apparently, the error does not converge uniformly, but on average we observe a convergence rate of roughly 1. This is similar to the error in the voxelization and so these two resolutions (number of voxels and number of integration points) should be in the same range.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

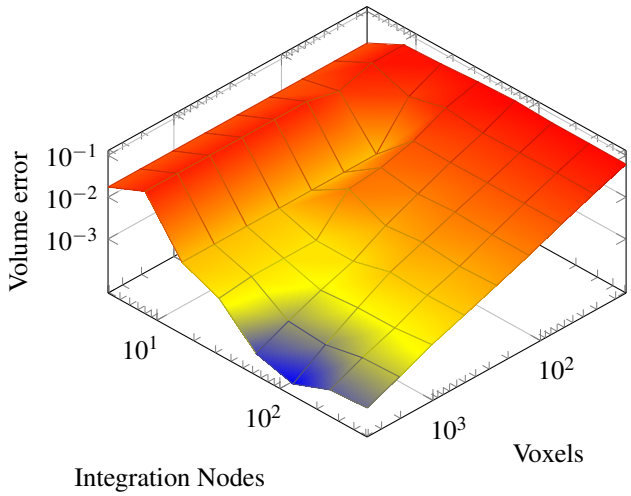

Figure 5. Error in the volume approximation for a sphere with a radius of $\frac{1}{3}$. The mesh consists of 8 elements with a common vertex in the center of the sphere.

To investigate the mechanism in 3D, we look at three different geometrical objects: a sphere, a cube and a tetrahedron. In each case the overall domain is a cubical box $[-1,1]\times[-1,1]\times[-1,1]$ subdivided by 8 cubical elements. The sphere is put in the middle of the domain, with its center at $(0,0,0)$ and has a radius of $\frac{1}{3}$. Similarly, the cube has an edge length of $\frac{2}{3}$ and its barycenter is placed at the center of the domain at $(0,0,0)$. As a third basic geometry the tetrahedron again is similarly defined with its barycenter in the center of the domain and an edge length of 1.

The error in the volume approximation is used to assess the quality of the polynomial representation. In Figure 5 this measure is plotted for the sphere over the two available parameters voxelization resolution and numerical integration points. It can be observed, that the error is mostly bounded by either one of the parameters and for a minimal computational effort it indeed is necessary to change them according to the relation in Equation 5.

Figure 6 illustrates the projection of the sphere on a 3D polynomial representation in the eight elements of the mesh. From left to right it shows improved accuracies. In yellow the isosurface of a color value of 0.5 is shown and in comparison the half of the reference sphere is shown in blue. The leftmost image shows the sphere in comparison to the eight elements, indicated by the black wireframe. In this, a very rough estimation of the sphere is shown with a polynomial of degree 15 and a low voxelization resolution. Clearly the stair cases from the voxelization are visible in the polynomial representation here. The next image shows a zoom in for finer voxelization, but still a polynomial degree of 15, in the left half the reference sphere is again depicted in blue. While the finer resolution in the voxelization yields a better approximation now, there are relatively strong oscillations visible, especially close to the element boundaries. To allow for a better representation of the sphere we move to a polynomial degree of 31 in the third image. The voxelization is chosen with an appropriate resolution in this case, but the numerical integration results in aliasing issues, exihibiting a staircase like effect for the isosurface of the polynomial. Finally in the rightmost image, we see the effect of a higher number of points for the numerical integration, resulting in a much smoother geometry for the shown polynomial of degree 31.

Figure 7 illustrates the approximation of the cubical geometry. All images show the isosurface for polynomial representation of degree 15, but the accuracy of the numerical approximation increases from left to right. The number of integration points increases from 16 in the leftmost image over 32 and 48 to 64 in the rightmost image and the voxelization is chosen according to Equation 5. Edges and corners get smoothed out, but only little oscillations are observed for this basic, axis aligned, geometry.

Finally, in Figure 8 a study on the 3D polynomial approximation of a tetrahedron is depicted. The maximal polynomial degree increases from 7 up to 63 and twice as many integration points as polynomial modes are used in each approximation. It can be observed, how the sharp edges and corners are smoothed out at low orders, but are increasingly well recovered in the higher resolved polynomials. Also oscillations in the planes of the tetrahedron get smaller in amplitude. With a polynomial of degree 63 the original

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

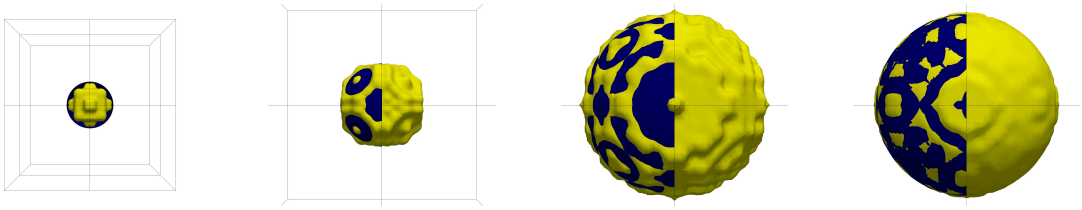

Figure 6. Illustration of sphere approximations, with increasing accuracy. Blue is the sphere and yellow the isosurface of the color value at 0.5. On the left the sphere is shown in the embedding domain with the 8 elements. Voxelization and integration points increase from left to right.

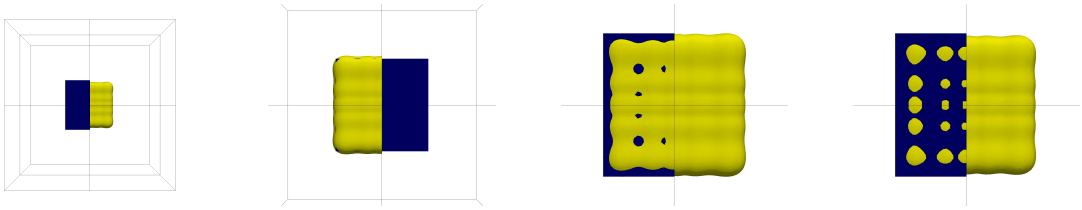

Figure 7. Representation of the cube in 8 elements with polynomials of degree 15. From left to right an increasing number of integration points is used. The leftmost image shows the cube in relation to the 8 elements of the mesh. The reference geometry is drawn in blue and the isosurface of the color value 0.5 in yellow. To allow a better comparison, the reference is cut in the middle, except for the second image, where it is the other way around an the isosurface is cut.

shape is well captured. This shows, that a high order polynomial can nicely be constructed, even for objects with sharp corners and edges.

Figure 9 shows the application of the described method to a more complex geometry. Depicted is in yellow the isosurface of a polynomial approximating a porous medium, described by a triangulation in a STL file shown in dark blue. This geometry features small bridges and holes. Those are well recovered by the polynomial approximation, only the edges get a little bit smoothed out. Keep in mind, that we are only using cubical elements, which can be exploited by the numerical scheme. Also, no bad elements arise resulting in a highly robust mesh generation.

SUMMARY AND OUTLOOK

Seeder implements a robust first order method to obtain polynomial representations of geometrical objects to describe nonsmooth material distributions. The robustness is due to the low order geometry approximation and the usage of a flood filling algorithm, eliminating the need for high requirements on the quality of the geometry representation. It has been shown, that the numerical approximation indeed converges towards the optimal L^2 projection of the nonsmooth distribution onto the polynomial function space, used in high order discontinuous Galerkin solvers. A possible improvement over the current staircase representation of the surface could for example be achieved by computing an approximate

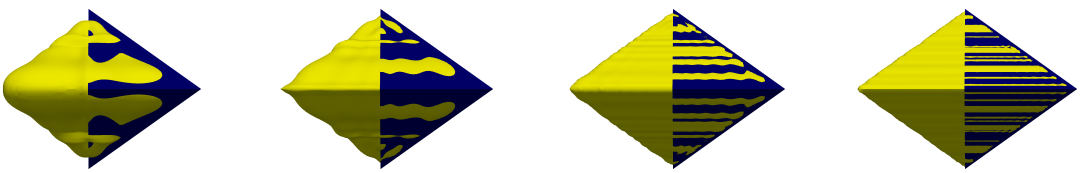

Figure 8. Approximation of the tetrahedron with an increasing polynomial degree from left to right. Starting on the left with a polynomial degree of 7 and increasing over 15 and 31 to 63 in the rightmost image. Shown is the isosurface of the polynomial at a value of 0.5 in yellow and for comparison the reference geometry cut in half with a blue coloring.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015

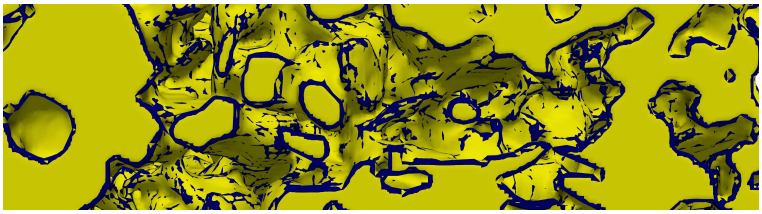

Figure 9. Isosurface of a porous medium (yellow) in comparison to the original STL data (blue). The geometry is well recovered, only egdes are smoothed out a little.

plane for the geometry within intersected voxels. While such a computation would introduce additional complexity and potentially expensive computations, it would reduce the number of voxels required to resolve the shortest distances between integration nodes. Nevertheless, the pure voxelization scheme currently deployed is already capable to discretize large an complex settings and has been successfully used for highly detailed simulations. Though the errors from the Table 1 remain, it is proven in Zudrop and Hesthaven (2015) that high order information can be recovered for such nonsmooth setups by an appropriate post-processing.

REFERENCES

Alpert, B. K. and Rokhlin, V. (1991). A Fast Algorithm for the Evaluation of Legendre Expansions. *SIAM Journal on Scientific and Statistical Computing*, 12(1):158–179.

Bueno-Orovio, A. and Pérez-García, V. M. (2006). Spectral smoothed boundary methods: The role of external boundary conditions. *Numerical Methods for Partial Differential Equations*, 22(2):435–448.

Gottlieb, D. and Hesthaven, J. (2001). Spectral methods for hyperbolic problems. *Journal of Computational and Applied Mathematics*, 128(1–2):83 – 131. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

Harlacher, D. F., Hasert, M., Klimach, H., Zimny, S., and Roller, S. (2012). Tree based voxelization of stl Data. In Resch, M., Wang, X., Bez, W., Focht, E., Kobayashi, H., and Roller, S., editors, *High Performance Computing on Vector Systems 2011*, pages 81–92. Springer Berlin Heidelberg.

Hesthaven, J. S. and Warburton, T. (2007). *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, 1 edition.

Hindenlang, F., Bolemann, T., and Munz, C.-D. (2015). Mesh curving techniques for high order discontinuous galerkin simulations. In *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pages 133–152. Springer.

Ishida, T., Takahashi, S., and Nakahashi, K. (2008). Efficient and Robust Cartesian Mesh Generation for Building-Cube Method. *Journal of Computational Science and Technology*, 2(4):435–446.

Klimach, H., Masilamani, K., Harlacher, D., and Hasert, M. (2015). Seeder. https://bitbucket.org/apesteam/seeder. Last accessed on 2015-06-04.

Mittal, R. and Iaccarino, G. (2005). Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261.

Peskin, C. S. (2002). The immersed boundary method. *Acta Numerica*, 11:479–517.

Sabetghadam, F., Sharafatmandjoor, S., and Norouzi, F. (2009). Fourier spectral embedded boundary solution of the poisson's and laplace equations with dirichlet boundary conditions. *J. Comput. Phys.*, 228(1):55–74.

White, E. (2013). What Is An STL File? http://www.3dsystems.com/quickparts/learning-center/what-is-stl-file. Last accessed on 2015-06-04.

Wilbraham, H. (1848). On a certain periodic function. *The Cambridge and Dublin Mathematical Journal*, 3:198–201.

Yerry, M. A. and Shephard, M. S. (1984). Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, 20(11):1965–1990.

Zudrop, J. and Hesthaven, J. S. (2015). Accuracy of high order and spectral methods for hyperbolic conservation laws with discontinuous solutions. *SIAM Journal on Numerical Analysis*, 53(4):1857–1875.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1316v1 | CC-BY 4.0 Open Access | rec: 20 Aug 2015, publ: 20 Aug 2015