Treat All Integrals as Volume Integrals:

A unified, parallel, grid-based method for evaluation of volume, surface, and path integrals on implicitly defined domains

Mete Yurtoglu
Google
747 6th St S.
Kirkland, WA
myurtoglu@me.com

Molly Carton
Student Mem. ASME
University of Washington - Seattle
Mechanical Engineering
Seattle, WA 98195-2600
mcarton@uw.edu

Duane Storti
University of Washington - Seattle
Mechanical Engineering, Box 352600
Seattle, WA 98195-2600
storti@uw.edu

ABSTRACT

We present a unified method for numerical evaluation of volume, surface, and path integrals of smooth, bounded functions on implicitly defined bounded domains. The method avoids both the stochastic nature (and slow convergence) of Monte Carlo methods and problem-specific domain decompositions required by most traditional numerical integration techniques. Our approach operates on a uniform grid over an axis-aligned box containing the region of interest, so we refer to it as a grid-based method. All grid-based integrals are computed as a sum of contributions from a stencil computation on the grid points. Each class of integrals (path, surface, or volume) involves a different stencil formulation, but grid-based integrals of a given class can be evaluated by applying the same stencil on the same set of grid points; only the data on the grid points changes. When functions are defined over the continuous domain so that grid refinement is possible, grid-based integration is supported by a convergence proof based on wavelet analysis. Given the foundation of function values on a uniform grid, grid-based integration methods apply directly to data produced by volumetric imaging (including computed tomography and magnetic resonance), direct numerical simulation (DNS) of fluid flow, or any other method that produces data corresponding to values of a function sampled on a regular grid. Every step of a grid-based integral computation (including evaluating a function on a grid, application of stencils on a grid, and reduction of the contributions from the grid points to a single sum) is well-suited for parallelization. We present results from a parallelized CUDA implementation of grid-based integrals that faithfully reproduces the output of a serial implementation but with significant reductions in computing time. We also present example grid-based integral results to quantify convergence rates associated with grid refinement and dependence of the convergence rate on the specific choice of difference stencil (corresponding to a particular genus of Daubechies wavelet).

1 Introduction

Evaluation of a definite surface integral is typically defined as the sum of contributions from a finely-divided polygonal approximation to the domain of integration [1], and the statement extends to cover line and volume integrals if polygon is generalized to n-dimensional polytope. Methods for evaluating integrals based on such a definition require a finely divided set of polytopes customized to closely approximate the particular domain of integration. In cases where the domain is specified in terms of polytopes, the polytopes can be finely subdivided and such an approach is perfectly reasonable. In contrast, this paper focuses on the case when the domain is defined by an implicit function (or by an approximation based on a sampling of function values), so that a finely divided domain approximation is not available without further computation.

In the field of engineering design, evaluation of definite integrals is often associated with determining mass or surface properties of a solid model, and there is a considerable literature related to this task dating back to Sarraga [2] and Lee and Requicha [3, 4] who nicely set the problem context. In Part I of their work, Lee and Requicha [3] noted that the existing literature on computational multiple integration focuses on problems with a complicated integrand but a simple domain, and characterized mass property computation as the converse problem with relatively simple integrands on complicated domains. They also identified key issues: (1) the modeling scheme used to describe the solid dominates the design of integration algorithms, and (2) dominant errors are typically associated with errors in representing the domain. Their treatment starts with consideration of solids represented in terms of their polyhedral boundaries, and states a method that combines the Divergence Theorem with integration over the parametrized polygonal facets. While stating that integrals on regions with curved boundaries can be approximated by obtaining polygonal approximations of the curved surfaces or by applying Greens Theorem on the trimmed patch boundaries in the parameter plane, they are careful to note that that error estimates are not available for such methods. They go on to discuss Monte Carlo methods that employ computations at a large number N of sample points to provide reliable but slow ($O(\sqrt{}(N))$) convergence and, in Part II [4], methods based on conversion to cellular approximations.

From this early foundation, the literature goes on to explore a variety of specialized methods using particular representation schemes. One branch of exploration involves ray representations (or ray-reps [5–7]) which Ellis et al. characterize as sampled boundary representations which support efficient integration but with accuracy that is dependent on that of the ray-surface intersection algorithm. Another branch involves computing approximate contributions from surface patches [8–10] or cells [11] based on optimized points in the parametric domain. It is worth noting that integration has long been recognized as a challenging problem that would benefit from parallel computing techniques [5,8,10]. The common thread of the existing literature on computing integral properties of solid models is the existence of an explicit model involving either parametric surface patches or a spatial/cellular decomposition.

Here we focus on the alternative case of integrals for which the domain is implicitly defined by the isosurface(s) of one or more functions of the spatial coordinates. Problems of this type are commonly generated by users of computer-aided math systems; e.g. compute the area and/or moments of inertia of an ellipsoidal shell or compute the path length of the seam on a tennis ball described as the intersection between a sphere and a hyperbolic paraboloid [12]. In the solid modeling context, function-based representations or f-reps use these implicit domains to define solids [13, 14].

Monte Carlo methods can be applied directly to volumetric integrals (on an implicit domain of co-dimension zero) but, due to the previously mentioned issue of slow convergence, a significant literature has developed around attempts at conversion to alternative representations that support other methods. In the best case scenario, a significant amount of additional computation is involved, for example applying a tessellation method such as marching cubes [15] or a cell or ray decomposition. In the worst case (like the Schwarz lantern surface approximation of a cylindrical surface [16]), finely-divided polytopes may be obtained that do not provide good approximations of the domain even though the vertices provide a dense sampling of the domain, and in such cases the integral estimates can be unreliable. Edwards [1, p.26-27] points out that confusion between how integrals are defined (as sums of contributions from finely-divided polytopes approximating the domain) and how integrals are evaluated (using the Fundamental Theorem of Calculus) forms a significant obstacle to the proper application of calculus. Here we extend that advice to integration on implicit domains and adopt an approach based on the Fundamental Theorem in its various forms (including the Divergence Theorem and Stokes Theorem).

Our approach is motivated by the one scenario where a fine polytope subdivision can be readily and reliably obtained without doing a significant amount of work. When the domain B is an n-dimensional block defined by a finite interval along each of n orthogonal coordinate directions, the domain can be finely divided into sub-blocks defined by uniform subdivision of the coordinate intervals. An integral over the block is then defined as the limit of the sum of contributions equal to the product of the volume of each sub-block with the value of the integrand at a representative point in the sub-block.

This provides an approach to volume integrals based on a reliable partition into fine polytopes, and our approach will be to always convert the integral to be computed into a volume integral over the block B. The natural choice of taking the representative point for each block to be its center brings us to the idea of working with values of a function on a uniform grid (arising as the collection of the centers of the uniform array of sub-blocks). For a grid with N_x, N_y, N_z points with spacing $\Delta_x, \Delta_y, \Delta z$ along the x, y, z coordinate directions respectively, the generic grid point is located at $\mathbf{r}_{i,j,k} = (x_i, y_j, z_k) = (x_0 + i\Delta_x, y_0 + j\Delta_y, z_0 + k\Delta_z)$. The sum of sub-block contributions can now be identified as a sum over voxel values $g_{i,j,k} = g(x_i, y_j, z_k)$, and the integral takes the initial form:

$$\int_B g(\mathbf{r})dv = \lim_{\Delta_x,\Delta_y,\Delta_z \to 0} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \sum_{k=0}^{N_z-1} g_{ijk}\Delta_x\Delta_y\Delta_z \tag{1}$$

and we present the volumetric (or "volumometric" [17]) formulations for each class of integrals before getting into further details of discretization.

The approach to integrating over a box can be extended to a more interesting domain Ω by introducing the characteristic or occupancy function $\mathcal{X}(\mathbf{r})$ defined by:

$$\mathcal{X}(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} \in \Omega \\ 0 & \mathbf{r} \notin \Omega \end{cases} \tag{2}$$

Here we are especially interested in an implicitly defined region $\Omega = \{(x,y,z) \in \mathbb{R}^3 | f(x,y,z) < 0\}$ in which case the occupancy function becomes

$$\mathcal{X}(f(\mathbf{r})) = \begin{cases} 1 & f < 0 \\ 0 & f > 0 \end{cases} \tag{3}$$

Including $\mathcal{X}$ as a factor in the integrand, the domain of integration can be extended to to the box B containing the region Ω.

$$\iiint_{\Omega} g(\mathbf{r}) dv = \iiint_{B} \mathcal{X}(\mathbf{r}) g(\mathbf{r}) dv \tag{4}$$

We have now obtained our first formula for an integral (in particular a volume integral) defined on an implicit domain in terms of a volume integral over a box. We will produce similar formulas for surface and path integrals before returning to present an alternative volume integral formulation. See [18] for further details.

2 Surface Integral Formulation

Now we consider integrating a scalar function over an implicitly defined surface. Where we previously considered the integral over a volume Ω, here we consider the surface integral I_S over the boundary $\partial\Omega$ of such a volume:

$$I_S = \iint_{\partial\Omega} g(\mathbf{r}) ds \text{ where } \partial\Omega = \{\mathbf{r} \in \mathbb{R}^3 | f(\mathbf{r}) = 0\} \tag{5}$$

We convert the surface integral to a volume integral over a box following earlier treatments by Resnikoff and Wells [19] and Storti [20]. After introducing the outward normal unit vector $\hat{n} = \nabla f / |\nabla f|$ (for which $\hat{n} \cdot \hat{n} = 1$ wherever ∇f exists), the surface integral becomes

$$I_S = \iint_{\partial\Omega} g(\mathbf{r}) ds = \iint_{\partial\Omega} g(\mathbf{r}) \hat{n} \cdot \hat{n} ds \tag{6}$$

allowing application of the Divergence theorem which produces the volume integral

$$I_S = \iiint_{\Omega} \nabla \cdot (g(\mathbf{r}) \hat{n}) dv \tag{7}$$

The domain of this integral can also be extended to $\mathbb{R}^3$ by introducing the occupancy function, giving:

$$I_S = \iiint_{\mathbb{R}^3} \mathcal{X}(f(\mathbf{r})) \nabla \cdot (g(\mathbf{r}) \hat{n}) dv \tag{8}$$

Integration by parts yields:

$$I_S = -\iiint_{\mathbb{R}^3} g(\mathbf{r})\nabla\mathcal{X}(f(\mathbf{r}))\cdot\hat{n}dv + \iint_{\partial\mathbb{R}^3}\mathcal{X}(f(\mathbf{r}))g(\mathbf{r})ds \tag{9}$$

Because $\mathcal{X}$ and $\nabla\mathcal{X}$ vanish outside Ω, the second integral vanishes and the domain of the remaining volume integral term can be restricted to the box B containing Ω:

$$I_S = \iint_{\partial\Omega} g(\mathbf{r})ds = -\iiint_B g(\mathbf{r})\nabla\mathcal{X}(f)\cdot\nabla f/|\nabla f|dv \tag{10}$$

which provides the desired formula for an integral over an implicit surface in terms of a volume integral over a box.

3 Path Integral Formulation

Here we consider the case of an integral along a path determined by the intersection of two surfaces. Current approaches involve attempts to create a parametrized path by (1) creating a spline to approximate the intersection of the surfaces (which may not be feasible especially for digitized functions) or (2) polygonizing each implicit surface and attempting to compute the intersection of the surface polygonizations. Alternatively, to evaluate the path integral I_P of a function $g(\mathbf{r})$ over the path I_P corresponding to the intersection of the implicit surfaces $\delta\Omega_1$ and $\delta\Omega_2$ (where $f_1(\mathbf{r}) = 0$ and $f_2(\mathbf{r}) = 0$ respectively), we begin by substituting in the unit tangent vector identity $\hat{t}\cdot\hat{t} = 1$ (where $\hat{t} = \hat{n}_1 \times \hat{n}_2$, $\hat{n}_1 = \frac{\nabla f_1}{|\nabla f_1|}$, and $\hat{n}_2 = \frac{\nabla f_2}{|\nabla f_2|}$).

$$I_P = \int_{\partial\Omega_1\cap\partial\Omega_2} g(\mathbf{r})dl = \int_{\partial\Omega_1\cap\partial\Omega_2} g(\mathbf{r})(\hat{t}\cdot\hat{t})dl \tag{11}$$

and apply Stokes' Theorem on equation 11 to obtain the surface integral

$$I_P = \iint_{\partial\Omega_1\cap\Omega_2} (\nabla\times g(\mathbf{r})\hat{t})\cdot\hat{n}_1 ds \tag{12}$$

Introducing the occupancy functions $\mathcal{X}(f_1)$ for the volume Ω_1 and $\mathcal{X}(f_2)$ for the volume Ω_2 and following equations 8 through 10 of the surface integral formulation in section 3 produces the formula for an integral over the path corresponding to the intersection of two implicitly defined surfaces:

$$I_P = -\iiint_B g(\mathbf{r})\nabla\mathcal{X}(f_1)\cdot(\mathcal{X}(f_2)\nabla\times\hat{t})dv \tag{13}$$

The occupancy functions $\mathcal{X}(f_1)$ and $\mathcal{X}(f_2)$ in this volume integral can be exchanged by symmetry. However, exchanging f_1 and f_2 results in a sign change of the tangent vector $\hat{t}$ due to the antisymmetry of the cross product, reversing the sign of the result.

4 Volume Integral Formulation

Rather than directly evaluating a volume integral based on Equations 8-9, for a broad class of integrands it is possible to rewrite a volume integral in a more easily evaluated alternative formulation. In particular, for any integrand $g(\mathbf{r})$ that can be analytically integrated with respect to one or more of the coordinates, a vector potential Φ can be obtained such that $g(\mathbf{r}) = \nabla\cdot\Phi$ and the volume integral becomes:

$$I_V = \iiint_\Omega g(\mathbf{r})dv = \iiint_\Omega (\nabla \cdot \Phi)dv \tag{14}$$

Introducing the occupancy function as in the previous section, we can extend the domain to the box B containing Omega. Applying the product rule, this becomes

$$I_V = \iiint_B \mathcal{X}(\nabla \cdot \Phi)dv = -\iiint_B \Phi \cdot \nabla \mathcal{X} dv + \iiint_B \nabla \cdot (\mathcal{X}\Phi)dv \tag{15}$$

Noting that Chi vanishes on ∂B, application of the Divergence Theorem causes the last term to vanish and produces the formula

$$I_V = -\iint_B \Phi \cdot \nabla \mathcal{X} dv \tag{16}$$

Since $\nabla\mathcal{X}$ corresponds to a generalized delta function on $\partial\Omega$, the contributions to this integral come not from the entire domain but only from the neighborhood of the boundary of the domain, which can significantly reduce the operations necessary to compute the value of the integral.

5 Discretization

Now that we have obtained formulas for each class of integral, the remaining step towards a numerical method is discretization. As discussed in the introduction, we do this in a straightforward, reliable manner by dividing the box uniformly into sub-boxes and choosing the center of each sub-box as the representative point at which to estimate the contribution from the sub-box. The center points of the sub-boxes form a regular three-dimensional grid over the box, and the sampled function values correspond to a voxel set.

Note that after each class of integrals is converted to a volume integral over the box, the integrand involves gradients so a numerical derivative estimator needs to be applied. (To simplify the discussion, we focus on equal grid spacing Δ along each coordinate direction. However, the method generalizes directly for coordinate-specific spacing by inclusion of appropriate scale factors in the sub-block volume and the derivative estimates.) To gain access to reliable theoretical underpinnings, we can think of the data on the grid to be sampled values of a function approximated by a tensor product of Daubechies wavelets [19] (i.e., we treat each coordinate direction in turn using conventional 1-dimensional Daubechies wavelet analysis). This provides two distinct benefits: (1) There are known approaches to computing the vector of connection coefficients that serves as the stencil for computing sampled values of the derivative of a wavelet function from sampled values of the function itself [21, 22]. (2) If information is available to compute function values on dyadic refinements of the grid, we can invoke key convergence theorems from Resnikoff and Wells [19]:

> **11.7** Suppose Ω is a bounded set of finite perimeter and that the function F is twice differentiable, then $L^j(\partial\Omega)$ converges to the length of $\partial\Omega$ as $j \to +\infty$.
>
> **11.8** Assume that $\partial\Omega$ is Lipschitz and piecewise smooth with a finite number of nonsmooth points. Then, as $j \to +\infty$, $L^j(\partial\Omega)$ converges to $L(\partial\Omega)$, the length of $\partial\Omega$.

Here we follow the notation of Resnikoff and Wells in which j indicates the level of wavelet analysis. (Elsewhere j is the grid index along the y direction.) These theorems are stated specifically for computing path length of implicitly defined planar curves, and $L^j(\partial\Omega)$ refers to the level j estimate of the length of the boundary $\partial\Omega$ which corresponds to our equation 18. As noted in [19], the theorems generalize to higher dimensions and more general integrands.

The analysis leading to the theorems applies in general for each member of the Daubechies family of wavelets, so there is freedom to employ a particular genus G of Daubechies wavelets for integral computations. Genus $G = 0$ corresponds to Haar wavelets which are piece-wise constant and have jump discontinuities, so they are not of great interest for applications where derivative expressions are important. Daubechies wavelets with $G > 0$ are continuous and have a well-defined connection

coefficient vector of finite length that can be used as a stencil to compute derivative estimates on a grid from values of a function sampled on the grid [23, 24]. Larger values of G provide smoother wavelet approximants at the cost of longer connection coefficient vectors (and associated increases in stencil size and computational cost). The Daubechies connection coefficient vectors are not always easy to find (and some published tables contain typographical errors), so we provide the connection coefficient vectors for genus $1 \leq G \leq 4$ in Table 1. Note that for $G < 3$ the connection coefficients coincide exactly with standard symmetric finite difference derivative approximations. The connection coefficients are obtained by solving systems of polynomial equations, and exact rational values are only available only for small values of G that lead to low degree polynomials.

Genus	$\Delta*$First Derivative Connection Coefficient Vector
1	$-\frac{1}{2}, 0, \frac{1}{2}$
2	$\frac{1}{12}, \frac{-2}{3}, 0, \frac{2}{3}, -\frac{1}{12}$
3	$-\frac{1}{2920}, -\frac{16}{1095}, \frac{53}{365}, -\frac{272}{365}, 0, \frac{272}{365}, -\frac{53}{365}, \frac{16}{1095}, \frac{1}{2920}$
4	$-\frac{1}{1189272}, \frac{128}{743295}, \frac{2645}{1189272}, -\frac{1664}{49553}, \frac{76113}{396424}, -\frac{39296}{49553}, 0, \frac{39296}{49553}, -\frac{76113}{396424}, \frac{1664}{49553}, -\frac{2645}{1189272}, -\frac{128}{743295}, \frac{1}{1189272}$

Table 1: Daubechies first derivative connection coefficient vectors for genus 1 - 4. Coefficients for genus 5-7 are available in [18]

Given values of the relevant functions (the integrand or its potential function and the functions that implicitly define the domain), numerical evaluation of any integral is a sum of contributions from the grid points. Using i, j, and k to index grid points along the coordinate directions, the general formula for an integral on an $n \in 1,2,3$-dimensional domain is simply

$$I_n = \Delta^3 * \sum (C_n)_{i,j,k} \tag{17}$$

where the relevant functions and the contributions $(C_n)_{i,j,k}$ are given in Table 2.

Subscripts x, y, z indicate derivative estimates obtained by a derivative stencil centered at the grid point. For example, if we choose Daubechies genus 1 with connection coefficient vector {-½, 0, ½}, then the derivative of the grid point with indices i, j, k is $[f_x]_{ijk} = (1/\Delta) * (\frac{1}{2} f_{i+1,j,k} - \frac{1}{2} f_{i-1,j,k})$.

Note that an optional clipping function h has been included to implicitly restrict the domain of integration, e.g. to compute an integral over the portion of an implicit surface or implicit path that lies within an implicit volume. By default, $h(x,y,z) = -1$ so that $\mathcal{X}(h) \equiv 1$ when no domain clipping is desired. Where a signum function is available, $\mathcal{X}(f)$ may be replaced with the equivalent $[1 - sgn(f(\mathbf{r}))]/2$.

In contrast to a conventional stencil computation involving sampled values of a single function [25], the stencil formulas in table 2 are generalized to include contributions from neighboring values of the defining function(s), the occupancy function(s), the integrand, and the clipping function. For a surface integral with integrand $g(\mathbf{r})$, for example, the resulting discrete formula is

$$I(\partial\Omega) = -\Delta^3 \sum_{i,j,k} \left[g \frac{\frac{\partial}{\partial x}\mathcal{X}(f)\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}\mathcal{X}(f)\frac{\partial f}{\partial y} + \frac{\partial}{\partial z}\mathcal{X}(f)\frac{\partial f}{\partial z}}{\sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2 + \frac{\partial f}{\partial z}^2}} \right]_{i,j,k} \tag{18}$$

6 Implementation

The computations specified in table 2 and in sample pseudocode in algorithms 1, 2, and 3 all fit into well-known patterns of parallel computation [26]. Evaluation of a function on a grid corresponds to mapping, the most readily parallelized pattern of computation. The remainder of the computation from each grid point involves evaluation of a stencil formula that is also readily parallelized, and summation of the contributions from the grid points corresponds to the reduction pattern. We

Integral class	Grid functions	Contribution formula
Volume	Integrand g Geometry f Clipping h	$C_3 = \mathcal{X}(h)g\mathcal{X}(f)$
Alternate Volume	Potential Φ Geometry f Clipping h	$C_3 = \mathcal{X}(h)\Phi \cdot \nabla(\mathcal{X}(f))$
Surface	Integrand g Geometry f Clipping h	If $\nabla(f)\cdot\nabla(f) == 0$, $C_2 = 0$, else $C_2 = {}^{\mathcal{X}(h)\nabla\mathcal{X}(f)\cdot\nabla(f)}/_{\sqrt{\nabla(f)\cdot\nabla(f)}}$
Path	Integrand g Geometry f_1, f_2 Clipping h	If $\nabla(f_1)\cdot\nabla(f_2) == 0, C_1 = 0$ else $\hat{n}_1 = {}^{\nabla(f_1)}/_{\sqrt{\nabla(f_1).\nabla(f_1)}}$ $\hat{n}_2 = {}^{\nabla(f_2)}/_{\sqrt{\nabla(f_2).\nabla(f_2)}}$ $\hat{t} = \hat{n}_1 \times \hat{n}_2$ $C_1 = -\mathcal{X}(h) * \nabla\mathcal{X}(f_1)\cdot(\mathcal{X}(f_2)\nabla\times\hat{t})g$

Table 2: Contributions from each grid point

Algorithm 1 Gradients

$df/dx = df(i+1,j,k) - df(i-1,j,k)/(2*\Delta)$
$df/dy = df(i,j+1,k) - df(i,j-1,k)/(2*\Delta)$
$df/dz = df(i,j,k+1) - df(i,j,k-1)/(2*\Delta)$
$d\mathcal{X}/dx = \mathcal{X}(df(i+1,j,k)) - \mathcal{X}(df(i-1,j,k))/(2*\Delta)$
$d\mathcal{X}/dy = \mathcal{X}(df(i,j+1,k)) - \mathcal{X}((i,j-1,k))/(2*\Delta)$
$d\mathcal{X}/dz = \mathcal{X}(df(i,j,k+1)) - \mathcal{X}(df(i,j,k-1))/(2*\Delta)$

Algorithm 2 Surface Integral: contribution at grid point i, j, k

denominator = $(df/dx)(df/dx) + (df/dy)(df/dy) + (df/dz)(df/dz)$
if denominator $< \varepsilon \approx 0$ **then**
    $C_2[i,j,k] = 0$
else
    numerator = $(df/dx)(d\mathcal{X}/dx) + (df/dy)(d\mathcal{X}/dy) + (df/dz)(d\mathcal{X}/dz)$
    $C_2[i,j,k] = -g[i,j,k] * numerator/denominator$
end if

Algorithm 3 Volume Integral using potential $z\hat{k}$: contribution at grid point i, j, k

$C_3 = -g[i,j,k] * d\mathcal{X}/dz$

created a parallel implementation of the grid-based integration formulas using CUDA, a well-known system for GPU-based parallelism based on the single-instruction multiple-thread (SIMT) model of parallel computation [27]. See [25] for the basics of CUDA including the shared memory techniques that enhance the efficiency of the stencil and reduction patterns.

Genus	Convergence Exponent	Intercept
1	-1.99	5.54
2	-2.18	4.40
3	-2.23	4.51
4	-2.29	4.88
5	-2.36	5.09
6	-2.42	5.39
7	-2.46	5.59

Table 3: Summary of convergence results for grid-based computation of the surface area of a torus with $R = 10$ and $r = 2$ using difference stencils of genus $1 \leq G \leq 7$. The convergence exponent is the slope of the best fit log-log plot for fixed genus and varying grid resolution. Note that increasing genus does steadily increase the magnitude of the convergence exponent, but that increase comes with increased computational cost associated with longer difference stencils.

7 Examples and Results

We now present results from sample computations for each class of integrals, verify convergence as the grid is refined, and quantify convergence rates including dependence of the choice of differencing stencil. We study differencing stencils corresponding to Daubechies wavelet connection coefficients for genus in the range $1 \leq G \leq 7$, and compare results with data from problems solved with alternative methods.

We apply the grid-based surface integral formulation to calculate the surface area of a torus, a problem with a well-known solution. The torus of major radius R and minor radius r can be implicitly defined by the function

$$f_{torus}(x,y,z) = \left(R - \sqrt{x^2+y^2}\right)^2 + z^2 - r^2 \tag{19}$$

and has surface area $A = 4\pi^2 Rr$.

We sampled the function f_{torus} of Eq. (19) on grids over a wide range of refinement quantified by R/Δ and computed the surface area of a torus of major radius $R = 10$ and minor radius $r = 2$ using the implementation described in Section 6 of the formula derived in Section 2. Figure 1 shows a log-log plot of relative error in the computed value as a function of the grid refinement. Figure 1a shows the data points and the best-fit line for derivative estimates of genus 1 and genus 2. Figure 1b shows the best fit lines for genus 1 through genus 7, and the slopes which quantify convergence rates are given in Table 3.

Figure 1a illustrates that, for this example, increasing genus (and stencil radius) from 1 to 2 produces an increase in the magnitude of the convergence exponent from 1.99 to 2.18 and a significant reduction of relative error over a broad range of grid refinement. Figure 1b shows that the relative error plots are tightly bunched for genus 2 through genus 7. Note that achieving a further 10% change in the convergence exponent from genus 2 requires using genus 6, increasing the stencil radius from 2 to 10, and accounting for function values at 5 times as many neighboring points.

The calculation of surface area involves the trivial integrand $g(\mathbf{r}) = 1$. However, the same method can be applied with an arbitrary integrand. For example, the moment of inertia of a thin toroidal shell about the z-axis can be calculated with the integrand $g(\mathbf{r}) = x^2 + y^2$ and the same function f_{torus}. Assuming unit mass per area, the exact solution for a torus centered around the z-axis is

$$I_z = \frac{1}{2} m\left(2R^2 + 3r^2\right) = 2\pi^2 rR\left(2R^2 + 3r^2\right). \tag{20}$$

Data from the convergence study for the genus 1 moment of inertia calculation is shown in figure 2. The convergence rate found as the magnitude of the slope of the best linear fit to the log-log data is 1.49. This convergence rate is somewhat slower than the convergence for the surface area calculation, and is likely associated with an integrand that is less well-behaved (compared to area computation where the integrand $g = 1$ has a trivial Lipschitz bound).

The analogous quantities, volume and moment of inertia, are also known for the solid torus [28]:

$$V = 2\pi^2 r^2 R \tag{21}$$

$$I_z = \frac{1}{4}m\left(4R^2+3r^2\right) = \frac{1}{2}\pi^2 r^2 R\left(4R^2+3r^2\right). \tag{22}$$

We choose the following non-unique potential function Φ (where $\nabla\cdot\Phi) = 1$) to calculate the volume:

$$\Phi = \frac{x\hat{i}+y\hat{j}+z\hat{k}}{3} \tag{23}$$

Using this potential function, we obtain the convergence results for grid-based evaluation of the volume of the torus defined by f_{torus} shown in Fig. 3. Raw data and best fit line are shown for the log-log plot of relative error vs. grid refinement. The fit indicates a a convergence exponent of 1.907.

To compute the moment of inertia of the solid torus using the method outlined in section 4, we chose the potential function

$$\Phi = \frac{x^3\hat{i}+y^3\hat{j}}{3} \tag{24}$$

so that $\nabla\cdot\Phi$ corresponds with the desired integrand $g = x^2+y^2$. Figure 4 shows the log-log plot of relative error as a function of grid refinement. The best-fit line indicates a convergence exponent of 1.51.

Direct integration can also be applied to path integral calculations. A simple case is the length of the intersection between two intersecting spheres. We take the example of two spheres of radius 15 and 13 centered at $\mathbf{r_1} = -7,0,0$ and $\mathbf{r_2} = 7,0,0$ respectively. This construction gives a circular intersection of radius 12 and an intersection arc length of 24π. Results using a genus 1 stencil are plotted in fig. 5, with a convergence exponent of 1.71.

The previous examples all have known exact results which make them useful for establishing and quantifying convergence. We now present a practical application to a current research problem. The data set consists of a grid of fuel to oxydizer mixture fractions resulting from direct numerical simulation of non-premixed turbulent combustion [29, 30]. An important aspect of this research involves characterizing the flame surface that is implicitly defined as an isosurface of the mixture fraction whose sampled values are produced by the numerical simulation.

Previous efforts to compute properties of the flame surface such as area or area density involved use of a marching cubes algorithm [15, 31] to obtain a surface triangulation on which to compute desired properties. By applying the grid-based integration method to a DNS data set we obtained a flame surface area value within 0.8% of the value obtained using triangulation, but without having to compute a triangulation. Moreover, flame surface area density corresponds directly to the contributions from the grid points (i.e., area density is obtained by performing the stencil computation but omitting the steps of multiplying by the voxel volume and summing).

Finally, we compare the accuracy of our results with a common alternative method. A Monte-Carlo approach is typical for volumetric integration with a smooth integrand [32]. We implemented a Monte Carlo integration for the solid torus and, since Monte Carlo methods do not involve a grid resolution parameter, we compared convergence rates based on the number of points where the relevant functions are evaluated. The grid-based comparison result is based on the alternative volume integral formulation with potential $\Phi(x,y,z) = z\hat{k}$ corresponding to the desired integrand $g = 1$.

Figure 7 compares the results of the convergence study of computational error as a function of the number of evaluation points. In contrast to the Monte-Carlo method which is known to be probablistic and relatively slow to converge, the grid-based method is deterministic and converges to the exact value to more than 4 decimal places of accuracy based on fewer than 1 million sample points, while the Monte-Carlo results can still show noticeable errors with more than 2 million evaluation points.

While timing is not the primary focus of this paper (and would require comparisons among execution times from different algorithms running on different hardware), it is appropriate to present a representative timing result. Figure 8 shows execution time as a function of number of points in the grid for computing the volume of the torus. The easy comparison to make (because the results are identical) is between the serial and parallel implementations of grid-based integration. Even with a relatively simple implicit defining function, serial execution takes on the order of $50\times$ longer than the parallel implementation running on a modern GPU. Comparisons with Monte Carlo methods are somewhat less straightforward, but 2 results can be clearly stated. For a fixed number of evaluation points, the parallel grid-based computation runs about $10\times$ faster than the Monte Carlo computation. For comparisons involving a specified accuracy, the number of Monte Carlo points

considered in the comparison may need to be increased significantly. For example, computing the torus volume to within 0.1% requires about 200k Monte Carlo evaluation points and 50k direct integral grid evaluation points, and the resulting speed advantage offered by the grid-based method is a factor of approximately $9\times$.

8 Conclusions

We have presented a unified approach to evaluation of volume, surface, and path integrals on implicitly defined domains. We present formulas for converting each type of integral to a volume integral over a box that can be readily subdivided leading to computations on a regular grid of points. The contribution from each grid point involves gradients of implicit functions and/or occupancy functions, and we compute the gradients in Cartesian coordinates using differencing stencils corresponding to Daubechies wavelet connection coefficients (which, for small genus, coincide with traditional central difference formulas).

We presented the results obtained by applying a CUDA implementation that leverages GPU-parallelism to enhance computational throughput to the point where interactive systems with real-time integral computation have been achieved. Results are presented for examples of known volume integrals (volume and moment of inertia of a solid torus), surface integrals (surface area of a torus and moment of inertia of a thin toroidal shell), and path integrals (length of the intersection curves between 2 spheres). We also present, as a point of comparison to other techniques, results for the area of a flame surface based on direct numerical simulation data and Monte-Carlo results for the volume calculation of a solid torus.

Wherever possible, results are confirmed to agree with available analytical values and results of a serial implementation. In all cases where data can be obtained for refined grids, grid refinement leads to convergence. Convergence exponents are observed in the range of 1.49 to 2.46 and, with higher genus differencing stencils producing faster convergence. For the examples presented, increasing genus from 1 to 2 generally produced significant reduction in relative error and seemed worth the computational cost of increasing the stencil radius from 1 to 2. The examples presented do not provide an example where further increase in genus (and stencil size) is obviously worthwhile. Our early experience with these methods suggests great potential for applications and for significant enhancement of integral property evaluation tools for computer-aided math systems.

References

[1] Edwards, H. M., 1993. *Advanced Calculus: A Differential Forms Approach*. Birkhäuser.

[2] Sarraga, R., 1982. "Computation of surface areas in gmsolid". *IEEE Computer Graphics and Applications 2,* **7**, pp. 65–72.

[3] Lee, Y. T., and Requicha, A. A., 1982. "Algorithms for computing the volume and other integral properties of solids. i. known methods and open issues". *Communications of the ACM,* **25**(9), pp. 635–641.

[4] Lee, Y. T., and Requicha, A. A., 1982. "Algorithms for computing the volume and other integral properties of solids. ii. a family of algorithms based on representation conversion and cellular approximation". *Communications of the ACM,* **25**(9), pp. 642–650.

[5] Ellis, J. L., Kedem, G., Lyerly, T., Thielman, D., Marisa, R. J., Menon, J., and Voelcker, H. B., 1991. "The ray casting engine and ray representatives". In Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications, ACM, pp. 255–267.

[6] Wang, C. C., Leung, Y.-S., and Chen, Y., 2010. "Solid modeling of polyhedral objects by layered depth-normal images on the gpu". *Computer-Aided Design,* **42**(6), pp. 535–544.

[7] Wu, J., Kramer, L., and Westermann, R., 2016. "Shape interior modeling and mass property optimization using ray-reps". *Computers & Graphics,* **58**, pp. 66–72.

[8] Gonzalez-Ochoa, C., McCammon, S., and Peters, J., 1998. "Computing moments of objects enclosed by piecewise polynomial surfaces". *ACM Trans. Graph.,* **17**(3), July, pp. 143–157.

[9] Krishnamurthy, A., and McMains, S., 2010. "Accurate moment computation using the gpu". In Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, SPM '10, ACM, pp. 81–90.

[10] Krishnamurthy, A., and Mcmains, S., 2011. "Accurate gpu-accelerated surface integrals for moment computation". *Comput. Aided Des.,* **43**(10), Oct., pp. 1284–1295.

[11] Luft, B., Shapiro, V., and Tsukanov, I., 2008. "Geometrically adaptive numerical integration". In Proceedings of the 2008 ACM symposium on Solid and physical modeling, ACM, pp. 147–157.

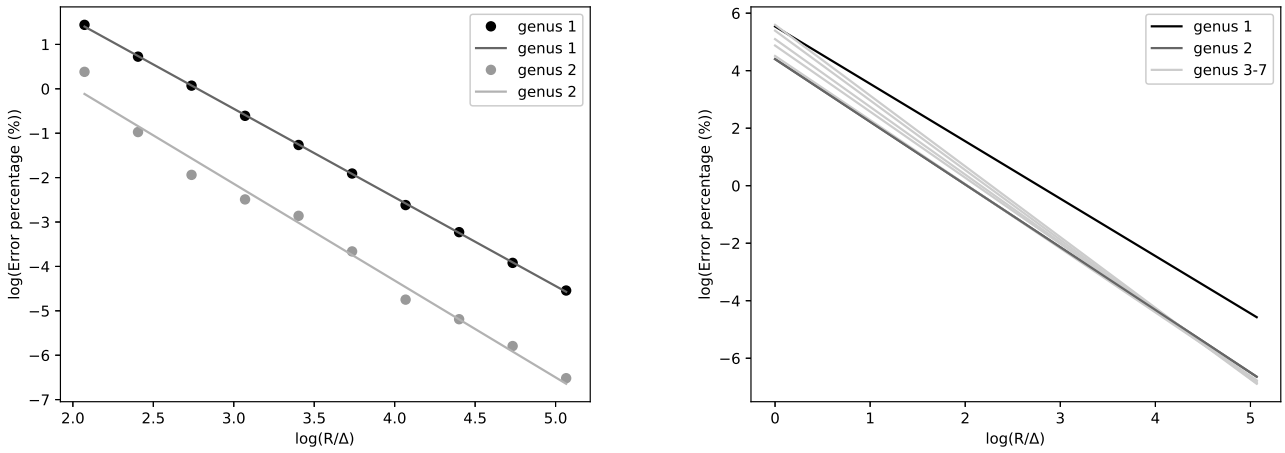
[12] ThatsMaths, 2014. The high-power hypar. online, May.

[13] Shapiro, V., Tsukanov, I., and Grishin, A., 2011. "Geometric issues in computer aided design/computer aided engineering integration". *Journal of Computing and Information Science in Engineering,* **11**(2), p. 021005.

[14] Storti, D., Ganter, M., Ledoux, W., Ching, R., Hu, Y., and Haynor, D., 2009. "Wavelet sdf-reps: Solid modeling with volumetric scans". *ASME. J. Comput. Inf. Sci. Eng,* **9**(3).

[15] Lorensen, W. E., and Cline, H. E., 1987. "Marching cubes: A high resolution 3d surface construction algorithm". In ACM siggraph computer graphics, Vol. 21, ACM, pp. 163–169.

[16] Zames, F., 1977. "Surface area and the cylinder area paradox". *The Two-Year College Mathematics Journal,* **8**(4), pp. 207–211.
[17] Koenderink, J. J., and Rabins, J. M., 1991. "Solid shape". *Applied Optics,* **30**, p. 714.
[18] Yurtoglu, M., 2017. "Gpu-based parallel computation of integral properties of volumetrically digitized objects". PhD thesis, University of Washington.
[19] Resnikoff, H. L., Raymond Jr, O., et al., 2012. *Wavelet analysis: the scalable structure of information*. Springer Science & Business Media.
[20] Storti, D., 2010. "Using lattice data to compute surface integral properties of digitized objects". *Proceedings of IDMME—Virtual Concept, Bordeaux, France*.
[21] Romine, C., and Peyton, B., 1997. "Computing connection coefficients of compactly supported wavelets on bounded intervals". PhD thesis, Oak Ridge National Laboratory.
[22] Bulut, F., 2015. "An alternative approach to compute wavelet connection coefficients". *Applied Mathematics Letters,* **53**.
[23] Latto, A., Resnikoff, H., and Tenenbaum, E., 1991. "The evaluation of connection coefficients of compactly supported wavelets". French-USA Workshop on Wavelets and Turbulance.
[24] Latto, A., and Tenenbaum, E., 1990. "Les ondelletes à support compact et la solution numérique de l'équation de burgers". *C. R. Acad. Sci. France,* **311**(903).
[25] Storti, D., and Yurtoglu, M., 2015. *CUDA for Engineers: An Introduction to High-performance Parallel Computing*. Addison-Wesley Professional.
[26] McCool, M. D., Robison, A. D., and Reinders, J., 2012. *Structured parallel programming: patterns for efficient computation*. Elsevier.
[27] NVIDIA, 2015. Cuda C programming guide, v7.5.
[28] Tatum, J., 2017. *Classical mechanics*.
[29] Wang, W., 2013. "Kinematic study of the evolution and properties of flame surfaces in turbulent nonpremixed combustion with local extinction and reignition". PhD thesis, University of Washington.
[30] Blakeley, B. C., Riley, J. J., Storti, D. W., and Wang, W., 2017. "On the kinematics of scalar iso-surfaces in turbulent flow". In American Physical Society Conference (Division of Fluid Dynamics).
[31] Newman, T. S., and Yi, H., 2006. "A survey of the marching cubes algorithm". *Computers and Graphics,* **30**(5), pp. 854–879.
[32] Liu, Y.-S., Yi, J., Zhang, H., Zheng, G.-Q., and Paul, J.-C., 2010. "Surface area estimation of digitized 3d objects using quasi-monte carlo methods". *Pattern Recognition,* **43**(11), pp. 3900–3909.



(a) Genus 1 and 2 stencils.

(b) Genus 1-7

Fig. 1: Data from convergence study for grid-based evaluation of surface area of a torus. (a) Raw data and best fit log-log plots of relative error versus grid refinement are shown for genus 1 and 2. The genus 2 stencil (with radius 2) produces a significant reduction in relative error across the full range of refinement. (b) Best-fit lines for log-log plots of relative error vs. refinement for genus 1 through 7. Increasing genus beyond $G = 2$ does not consistently provide decreases in relative error to make up for increased computational cost associated longer stencils.

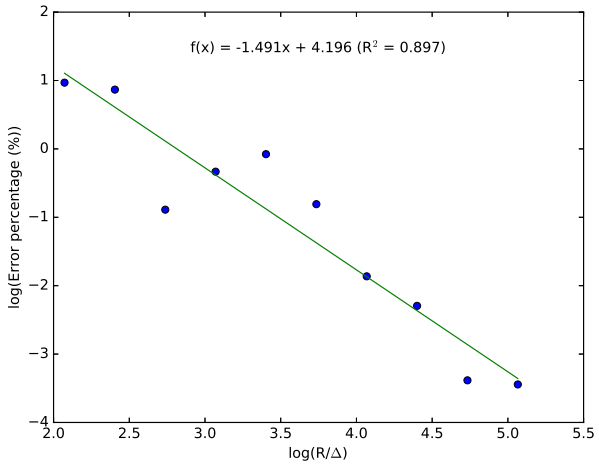


Fig. 2: Results of convergence study for genus 1 grid-based evaluation of moment of inertia of a toroidal shell.

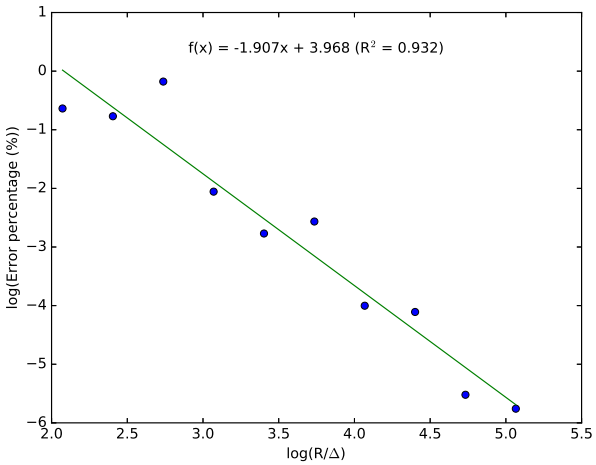


Fig. 3: Results of convergence study for genus 1 grid-based evaluation of the volume of a solid torus. Log-log plot of relative error vs. grid refinement shows raw data and best-fit line.

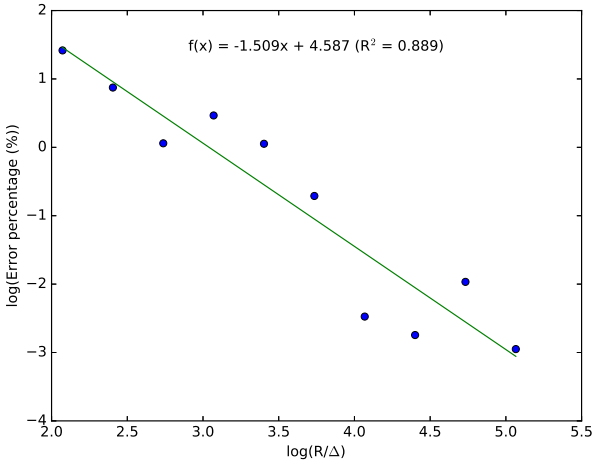
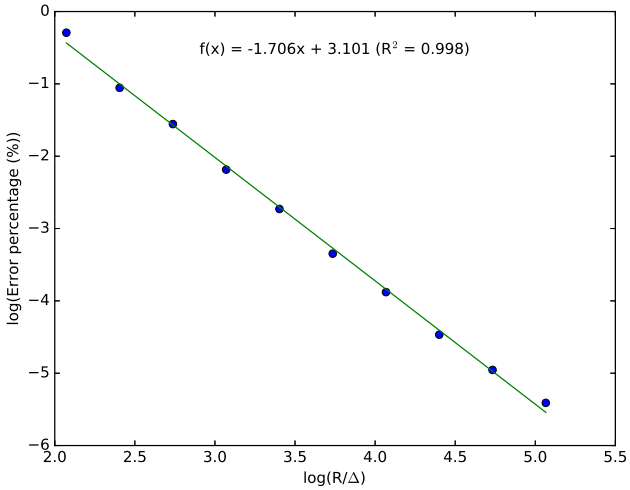


Fig. 4: Results of convergence study for genus 1 grid-based evaluation of the moment of inertia of a solid torus. Log-log plot of relative error vs. grid refinement shows raw data and best-fit line.



(a)

Fig. 5: Results of convergence study for genus 1 grid-based evaluation of the path length of a sphere-sphere intersection curve. Log-log plot of relative error vs. grid refinement shows raw data and best-fit line.

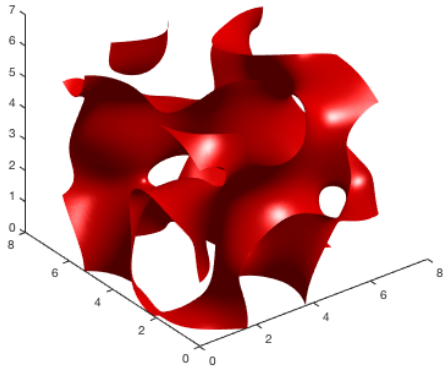
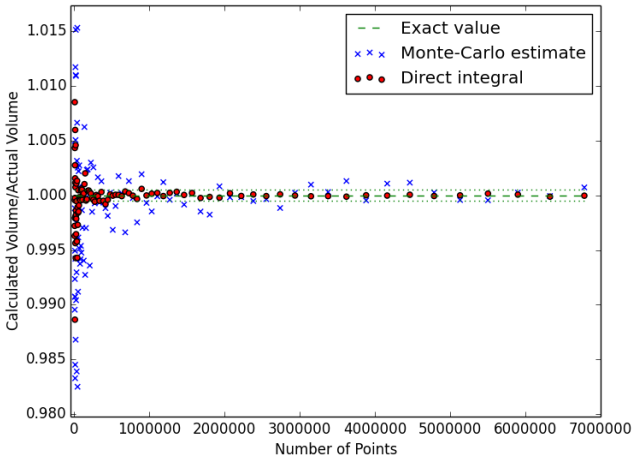
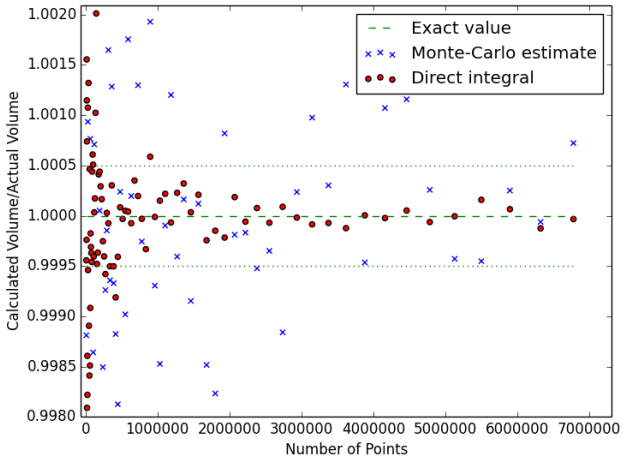


Fig. 6: Visualization of stoichiometric surface implicitly defined by a 512x512 grid of DNS simulation data. Area and area density were successfully computed using the grid-based surface integral method.

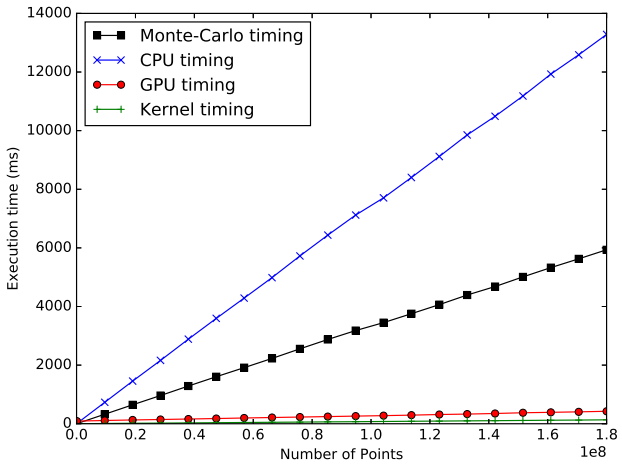


(a) Volume error vs. Number of points



(b) Detail of (a)

Fig. 7: Computed Volume/Actual Volume vs. Number of points for a the calculation of the volume of a torus, comparing a Monte-Carlo method with direct integration. Direct integration results are within $\pm.0005$ in under 1 million points, where some Monte-Carlo runs remain outside that range at 7 million points



(a)

Fig. 8: Timing comparison of calculation of torus volume with wavelet genus 1 in number of points evaluated vs. execution time (ms). Indicated are timings for direct integration on the CPU and GPU (including full execution time and kernel time) as well as Monte-Carlo timing