

# Fast Error-bounded Surfaces and Derivatives Computation for Volumetric Particle Data

Chandrajit Bajaj \*

Vinay Siddavanahalli †

December 6, 2005

## Abstract

Volumetric smooth particle data arise as atomic coordinates with electron density kernels for molecular structures, as well as fluid particle coordinates with a smoothing kernel in hydrodynamic flow simulations. In each case there is the need for efficiently computing approximations of relevant surfaces (molecular surfaces, material interfaces, shock waves, etc), along with surface and volume derivatives (normals, curvatures, etc.), from the irregularly spaced smooth particles. Additionally, molecular properties (charge density, polar potentials), as well as field variables from numerical simulations are often evaluated on these computed surfaces. In this paper we show how all the above problems can be reduced to a fast summation of irregularly spaced smooth kernel functions. For a scattered smooth particle system of  $M$  smooth kernels in  $R^3$ , where the Fourier coefficients have a decay of the type  $1/\omega^3$ , we present an  $O(M + n^3 \log n + N)$  time, Fourier based algorithm to compute  $N$  approximate, irregular samples of a level set surface and its derivatives within a relative  $L_2$  error norm  $\epsilon$ , where  $n$  is  $O(M^{1/3} \epsilon^{1/3})$ . Specifically, a truncated Gaussian of the form  $e^{-bx^2}$  has the above decay, and  $n$  grows as  $\sqrt{b}$ . In the case when the  $N$  output points are samples on a uniform grid, the back transform can be done exactly using a Fast Fourier transform algorithm, giving us an algorithm with  $O(M + n^3 \log n + N \log N)$  time complexity, where  $n$  is now approximately half its previously estimated value.

## 1 Introduction

Particle systems are used for modeling a number of physical world scenarios ranging from cosmological systems, vortex flows, to molecular systems [24, 13, 18, 15]. The applications are wide and varied and include chemistry, material science, and bio-engineering. A typical astrophysics computation attempts to follow the evolution of a system of cosmological boundaries, modeled by a collisionless gas particle system. An important computation performed on the volumetric particle system is that of force calculations, i.e. given  $M$  particles in three dimensions compute the effect of the gravitational or hydrodynamic force on a given particle by the other particles. These are often termed as  $M$ -body problems. The traditional model of molecules (including proteins, nucleic acids and sugars) is a collection of atomic centers with each atom modeled as a Gaussian function. Isotropic Gaussian kernels have been traditionally used to describe atoms due to their ability to approximate electron density maps and their analyticity [6, 17]. One description of molecular shape is provided by a suitable level set of the electron density of the molecule [5, 23]. Hydrophobicity is defined as the lack of affinity to water by a region of a molecule. Polar or charged regions are seen to form weak interactions with water molecules, while non polar or hydrophobic regions cluster together away from the water. This function is represented as a set of atoms with partial charges. Point set interpolation techniques [20],[2][1], are being developed to fit surfaces for spatial point cloud measurement data.

In each of the cases of volumetric particle systems there is the need for efficiently computing smooth approximations of relevant surfaces (molecular surfaces, density or temperature interfaces, vorticity thresholds, shock waves, etc), along with surface derivatives (normals, curvatures, etc.) at uniform or adaptive grids and irregular surface/volumetric meshes. Error bounded surfaces are computed using contouring techniques like marching cubes and adaptive grids. This requires evaluation of the summation function and its normal

---

\*bajaj@cs.utexas.edu

†skvinay@cs.utexas.edu

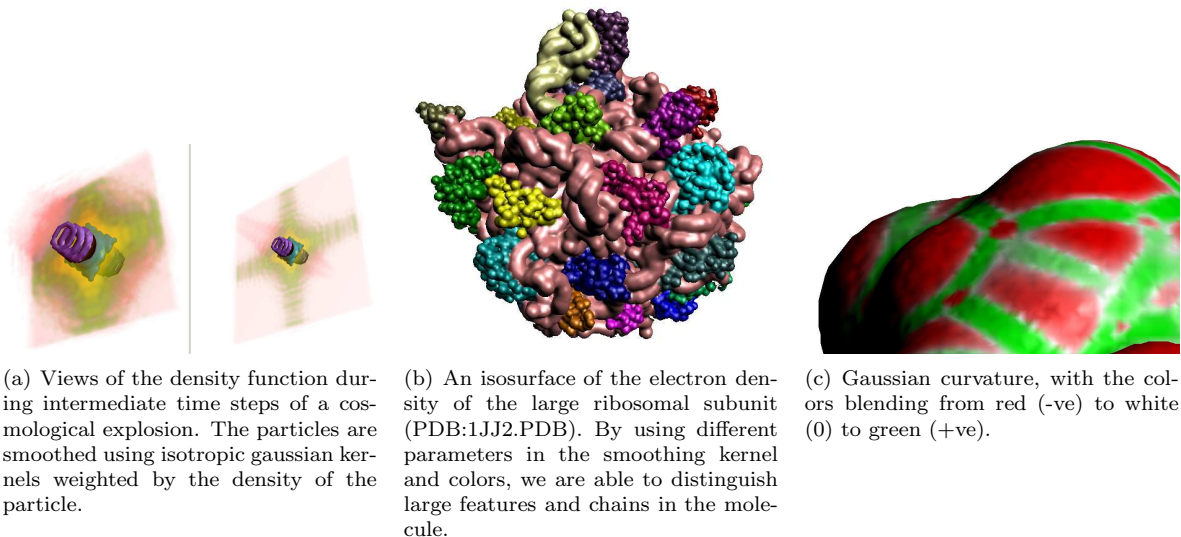


Figure 1: Smooth particle simulations and molecular electron density representations are two common examples of volumetric particle data. In the first image, we present an example of a cosmological simulation. The second image represents the molecular surface computed from a smooth particle (Gaussian) representation of atoms. In the third image, we show the Gaussian curvature function over a surface.

values at progressively refined adaptive meshes. All of these are shown to reduce to solving a set of equations of the form:  $f(\mathbf{x}) = \sum_{i=0}^{M-1} c_i k(\mathbf{x} - \mathbf{x}_i)$ . The main contributions of this paper include (1). An approximate fast summation technique for rapidly evaluating the summation of particle data with isotropic kernels onto a regular, irregular or adaptive grid. (2). Bounding the error in the function approximation and (3). An algorithm to update the function when sets of particles move relative to each other.

The rest of the paper is as follows: In section §2 we summarize some of the past work in fast summations. The core computations in several applications are shown to be fast summations of kernels in section §3. We then present our algorithm and various error bounds in the following section. Lastly, in section §5, we compare our technique with direct summations and Fast Fourier approximations. The proofs for most of our bounds are presented in the appendix.

## 2 Prior Related work

Several approaches to computing the Discrete Fourier Transform (DFT) polynomial have surfaced during the last decade [10, 11, 21, 26, 12, 3, 4, 27]. A review of many of these approaches can be found in [28]. In [27] the domain is split into subintervals and each subinterval is then projected onto a space of local Chebyshev polynomials. An alternate expansion is done in [3], Chebyshev polynomials are replaced with a Taylor expansion. For Fourier Transforms with singularities, Beylkin employs a series projections onto multi-resolution spaces [4]. Dutt et. al [10, 11] represent the DFT polynomial as a multi-pole expansion. For  $M$  non equidistant samples, the multi-pole approximate construction obtains the first  $M$  frequencies in  $\mathcal{O}(M \log M)$  computational steps and  $O(M)$  storage. The drawbacks are that such constructions require a fast multi-pole method, which leads to a complex implementation. Many of these approaches have been introduced for 1D domains. Extensions to multi-dimensions are possible through tensor products. In this paper, we follow the NFFT approach of Potts, Elbel and Steidl [21, 26, 12].

### 3 Reductions of applications to summation of kernels

We present problems from a variety of domains whose main computation step can be reduced to a summation of kernel functions.

**Particle hydrodynamics simulations** Smooth particle hydrodynamics or SPH [16] is a Lagrangian numerical hydrodynamics method that considers particles as smooth functions (kernels) decaying from the particle centers. The value of a field  $f$  at a point in space is then given by the sum of the contributing smooth kernels,  $f(\mathbf{x}) = \sum_{j=1}^M \gamma_j k(\mathbf{x} - \mathbf{x}_j)$ . A typical choice is that of the *Gaussian kernel*, given by  $k(\mathbf{x} - \mathbf{x}_j) = \frac{e^{-|\mathbf{x}-\mathbf{x}_j|^2/h^2}}{\pi^{3/2}h^3}$  with width defined by the *smoothing length*  $h$ . In figure 1(a), we show a SPH simulation data of a cosmological explosion rendered using isosurface and volume rendering.

**Electron density and associated properties of molecules** The electron density of a molecule with  $M$  atoms, centered at  $\mathbf{x}_j, j \in \{1..M\}$  can be written as  $f(\mathbf{x}) = \sum_{j=1}^M \gamma_j k(\mathbf{x}-\mathbf{x}_j)$  where  $\gamma_j$  and  $k$  are typically chosen

from a quadratic exponential description of atomic electron density. We use the kernel:  $e^{-\frac{d}{r^2}((\mathbf{x}-\mathbf{x}_j)^2-r^2)} = \gamma_j k(\mathbf{x} - \mathbf{x}_j)$ , where  $\gamma_j = e^d$ , and  $k(\mathbf{x} - \mathbf{x}_j) = e^{-\frac{d}{r^2}(\mathbf{x}-\mathbf{x}_j)^2}$ . The atomic kernels are affected by the radius  $r$  of individual atoms and the decay parameter  $d$ . In figure 1(b) we show the surface of an electron density map of a ribosome, generated as a summation of atoms, where each atom was represented as a truncated Gaussian. When describing the affinity of the molecules surface to the solvent (usually water), each atom is assigned a partial charge (say  $c_i$  for the  $i^{th}$  atom) based on type, bonding and structure in the molecule. It is also common to approximate the charges on all atoms of a given residue in a protein with the same value. Thus the hydrophobicity function  $h(\mathbf{x})$  for a molecule of  $M$  atoms, using the sum of Gaussians approach is given by  $h(\mathbf{x}) = \sum_{j=0}^{M-1} c_j e^{-\frac{d}{r^2}(\mathbf{x}-\mathbf{x}_j)^2}$

**Surface modeling** Surface modeling using blobby models or RBF models are used to represent objects in a compact form. 3D objects are modeled as set of scattered points whose convolution with a kernel approximates the original object. "Blobby models" [5] are also used to reconstruct computer models of objects sampled in the real world. Objects modeled in this fashion are known as "blobby models", "Soft objects" [29] and "meta balls". Other kernel functions used include Gaussians and truncated polynomials of the form:

$$k(x) = \begin{cases} a(1 - \frac{3x^2}{b^2}) & 0 \leq x < b/3, \\ \frac{3a}{2}(1 - \frac{x}{b})^2 & b/3 \leq x < b, \\ 0 & b \leq x \end{cases}$$

$$k(x) = \begin{cases} a(1 - \frac{4x^2}{9b^2} + \frac{17x^4}{9b^4} - \frac{22x^2}{9b^2}) & 0 \leq x < b, \\ 0 & b \leq x \end{cases}$$

The use of different Radial Basis Function (RBF) as the kernel is seen to lead to different continuity guarantees, constraints and energy minimizations. Thin plate splines ( of the form  $\mathbf{r}^2 \log \mathbf{r}$ ) are shown to be especially useful as they minimize the bending energy. ( $= \int \int_{R^2} f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 dx dy$  for  $R^2$ ). The paper [20] introduces the concept of Moving Least Squares (MLS). This method is used for modeling and rendering from a set of point clouds. This concept is further expanded in [1], where Alexa *et al* describe a method for displaying surfaces from point set clouds. This error bounded scheme locally approximates the surface with polynomials using Moving Least Squares (MLS). Variants of [20] include Amenta et al ([2]), by redefining the MLS surface and minimizing a different error functional.

**Derivatives, Normals, Curvatures** Particle datasets define surfaces where derivatives including normals and curvatures could be required. We show that the derivatives are also summations of kernels. Spivak [25] provides a good exposition to Gauss' 1827 work *Disquisitiones generales circa superficies curvas* (see

also General Investigations of Curved Surfaces, Raven Press, 1965, an English translation). Differential properties including curvatures and normals for implicit surfaces is covered in [19], which the authors state is a translation of a German paper [9]. These equations are also presented in [8]. A more rigorous study is presented in [30]. Here, we derive explicit equations for the curvatures. Consider the intersection of a plane with a surface, such that the plane passes through the normal at some intersection point. At that point, the curve of intersection has some curvature  $\kappa$ . Euler showed that as the plane is rotated about its single degree of freedom, the curvature has a minimum and a maximum value, with the directions being orthogonal (assuming the surface is not flat). These two curvatures  $\kappa_1, \kappa_2$  are known as the principle curvatures. The average of the principle curvatures is called as the Mean curvature  $H = \frac{\kappa_1 + \kappa_2}{2}$ . Let  $M$  be a 2D submanifold defined as  $f(x, y, z) = 0$ . Let the Gauss field (or normal field)  $\nu : M \rightarrow S^2 \subset R^3$ . The Gaussian curvature at a point  $\mathbf{p}$  within an area  $A$  can be defined as:  $K(\mathbf{p}) = \lim_{A \rightarrow \mathbf{p}} \frac{\nu(A)}{A}$ . Let  $d\nu$  be the Weingarten map defined as  $d\nu(v_{\mathbf{p}}) = [v_{\mathbf{p}}(\nu_x), v_{\mathbf{p}}(\nu_y), v_{\mathbf{p}}(\nu_z)]$  and  $\Pi$  the second fundamental form defined as  $\Pi(\mathbf{p})(v_{\mathbf{p}}, w_{\mathbf{p}}) = - \langle d\nu(v_{\mathbf{p}}), w_{\mathbf{p}} \rangle$ . Then (see [25]):  $K(\mathbf{p}) = \det(d\nu : M_{\mathbf{p}} \rightarrow M_{\mathbf{p}}) = \det(\Pi(\gamma_1, \gamma_2))$ ,  $\gamma_1, \gamma_2 \in M_{\mathbf{p}}$ , orthonormal. The Weingarten map, and the second fundamental form on two orthonormal vector in the tangent plane is also known as the shape operator  $S = -D\mathbf{n}$ . We assume that the surface is in  $\mathbb{C}^2$  and compute the gradient vector field  $\mathbf{g}$  and the unit normal vector field  $\mathbf{n}$ . The change of the normal field defines the curvature matrix  $C$ , with the principle curvatures being its eigenvalues.

$$\mathbf{g} = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right], \mathbf{n} = \frac{\mathbf{g}}{\|\mathbf{g}\|}, C = \begin{bmatrix} \frac{\partial \mathbf{n}_x}{\partial x} & \frac{\partial \mathbf{n}_x}{\partial y} & \frac{\partial \mathbf{n}_x}{\partial z} \\ \frac{\partial \mathbf{n}_y}{\partial x} & \frac{\partial \mathbf{n}_y}{\partial y} & \frac{\partial \mathbf{n}_y}{\partial z} \\ \frac{\partial \mathbf{n}_z}{\partial x} & \frac{\partial \mathbf{n}_z}{\partial y} & \frac{\partial \mathbf{n}_z}{\partial z} \end{bmatrix}$$

Let the eigenvalues of the above matrix be  $[0, \kappa_1, \kappa_2]$ . The characteristic polynomial can be written as  $-\lambda(\lambda - \kappa_1)(\lambda - \kappa_2) = |C - \lambda I| = -\lambda^3 + |C|\lambda^2 + \lambda^2(C_{11} + C_{22} + C_{33}) + \lambda(C_{12}C_{21} + C_{13}C_{31} + C_{32}C_{23} - C_{11}C_{11} - C_{22}C_{22} - C_{33}C_{33})$ . Letting  $\alpha = C_{11} + C_{22} + C_{33}$ ,  $\beta = C_{12}C_{21} + C_{13}C_{31} + C_{32}C_{23} - C_{11}C_{11} - C_{22}C_{22} - C_{33}C_{33}$ . Solving for the eigenvalues, we get:  $\kappa_1 = (\alpha - \sqrt{\alpha^2 + 4\beta})/2, \kappa_2 = (\alpha + \sqrt{\alpha^2 + 4\beta})/2$ . Using  $H = \frac{\kappa_1 + \kappa_2}{2}$ ,  $K = \kappa_1\kappa_2$ , we obtain  $H = \frac{\alpha}{2}$ ,  $K = -\beta$ . As shown in [8], the elements of  $C$  can also be written in terms of the Hessian matrix  $H: C_{ij} = \frac{H_{ij}\|\mathbf{g}\|^2 - \mathbf{g}_i \text{dot}_j}{\|\mathbf{g}\|^3}$ ,  $\text{dot}_j = \mathbf{g} \cdot [H_{j0} \ H_{j1} \ H_{j2}]^T$ . Let  $f_x = \frac{\partial f}{\partial x}$  etc. Expanding the equation for  $H$  and  $K$ , we obtain:

$$H = \frac{\oplus(f_x^2(f_{yy} + f_{zz})) - 2 \oplus(f_x f_y f_{xy})}{2(\oplus(f_x^2))^{1.5}}, K = \frac{2 \oplus(f_x f_y (f_{xz} f_{yz} - f_{xy} f_{zz}))}{(\oplus(f_x^2))^2}$$

where  $\oplus$  stands for cyclic summation over  $x, y, z$ , for example,  $\oplus(f_x^2) = f_x^2 + f_y^2 + f_z^2$ . See figure 1(c) for an example rendering of the Gaussian curvature over a surface. In each case, we see that the curvature function of particle datasets are also well posed summation problems. For example, in order to compute the mean curvature, we need to obtain  $f_x, f_y, f_z, f_{xx}$  etc. All of these functions are summations of derivatives of the original kernel.

## 4 Error bounded surface and derivative computation

We present a NFFT [21] based technique to compute summations of the form:

$$f(\mathbf{x}) = \sum_{i=0}^{M-1} c_i k(\mathbf{x} - \mathbf{x}_i) \quad (4.1)$$

where  $k$  has different smoothness assumptions. We also present algorithms to compute the above for the cases of uniform, non uniform and adaptive output points.

### 4.1 NFFT based fast summation algorithms

We follow the method outlined in [22],[7] to express the summation in the Fourier series form (see Appendix §A):  $f(\mathbf{x}) = \sum_{i=0}^{M-1} c_i k(\mathbf{x} - \mathbf{x}_i) \approx \sum_{\omega \in I_n} C_{\omega} K_{\omega} e^{2\pi i \mathbf{x} \cdot \omega}$ , where  $C_{\omega}$  and  $K_{\omega}$  are the Fourier series coefficients of the function of input centers and a kernel  $k$ . Where  $I_n$  denotes a 3D grid of indices:  $\{k : [-n/2..n/2]^3, k \in I\}$ .

Most kernels used in applications are tensor products or radially symmetrical, allowing us to compute these coefficients efficiently. In [22], a FFT over a  $n^3$  grid was used to obtain the coefficients. Since  $n$  is a discrete value and varies with user defined accuracy, their method only obtains an approximation to the coefficients. In order to obtain the Fourier coefficients, we first perform a truncation of the kernel. We scale the location of  $M$  input points to lie in  $(-0.25..0.25)^3$ . This ensures that the truncation has zero error. For kernels like the cubic bspline and Gaussians, the sync and the kernel are both tensor products. Hence obtaining the Fourier coefficients in 1D is sufficient. Consider the Gaussian kernel  $g(\mathbf{x}) = e^{-\mathbf{x}^2}$ . For the truncated version, the Fourier series in 1D are  $G(w) = \int_{-0.5}^{0.5} e^{-x^2} e^{-2\pi i w x} dx$ . We use MAPLE to compute this numerically.

### Algorithm

- $n$  is computed from the error bounds and is proportional to  $M^{1/3}, \epsilon^{3/2}$ , where the  $\epsilon$  is a user defined error level, for kernels whose Fourier coefficients decay at least inversely with the frequencies.
- $K_{\vec{\omega}}$  is computed numerically.
- $C_{\vec{\omega}}$  is computed approximately the using NFFT' algorithm [21].
- $f(\vec{x})$  is computed using an Inverse Fourier Transform for uniform grids and using the NFFT algorithm for non uniformly distributed output points.

The cost of the algorithm depends on  $n$ . We first show bounds on  $n$  for different kernels.

**Lemma** For tensor product kernels with Fourier coefficients  $K_{\omega}$ , the number of coefficients  $n$  needed is at most:

$$n = \min(\hat{n}) : \sum_{\omega \in I_{\hat{n}}} (K_{\omega})^2 \geq \frac{V}{2\pi} - \frac{M \min_j (|c_j|^2) V (\frac{\epsilon}{3})^2}{(\|c\|_1)^2}$$

where  $V$  is the integral of the kernel from  $(-0.5..0.5)^3$ .

**Lemma** For tensor product kernels whose Fourier coefficients decay at least inversely with frequency, the number of coefficients  $n$  needed is  $O(M^{1/3} \epsilon^{3/2})$ .

**Lemma** The fourier coefficients of a Gaussian function  $e^{-Bx^2}$  decay as the inverse of the frequency  $\omega$ :

$$G_{\omega} \leq \max\left(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi} \operatorname{erf}\left(\frac{\pi}{\sqrt{B}}\right), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{Be^{-(1+\pi^2/B)}}{\pi^4}\right) \frac{1}{\omega}, \quad (\omega \geq 2)$$

The truncation of the Gaussian can be expressed as convolution with a sinc function in Fourier space.

Hence the Fourier series coefficients of the truncated Gaussian function can be now written as

$\int_{-\infty}^{\infty} \sqrt{\frac{\pi}{B}} e^{-\pi^2 t^2 / B} \sin(2\pi\omega t) / (2\pi\omega - t) dt$ . We then bound the sinc function with a polynomial and integrate by parts to obtain the result.

See appendix §B.1 for proofs of the above lemmas.

### Cost

- $G_{\vec{\omega}}$  are computed numerically to the desired accuracy. Since for most applications (like electron density computation) the kernel size is known, these coefficients can be precomputed once.
- Given NFFT' sampling parameters  $\alpha \approx 2, m \approx 3$ , the cost of computing  $C_{\vec{\omega}}$  is  $O(Mm^3 + n^3 \log n)$  and takes  $O(M + (\alpha n)^3)$  space.
- The Inverse Fourier Transform takes  $O(N^3 \log N)$  time and  $O(N^3)$  memory. To expand  $p$  output points only, the computational cost is  $O((\alpha n)^3 \log(\alpha n)) + mp$  and takes  $O((\alpha n)^3 \log(\alpha n) + p)$  space.

## 4.2 Summation to different types of output point distributions

Depending on the application, we need to compute summations and their derivatives on either adaptive grids, uniform grids or non-uniformly distributed points.

### 4.2.1 Summation to a non-uniformly distributed set of points

Using the NFFT algorithm, we can compute the summation from the frequencies at  $N$  non-uniformly distributed points by approximating the inverse step as a sum of compactly supported functions  $\phi$  such as bsplines or truncated Gaussians:

$$f(\mathbf{x}) = k \otimes \sum_{i=0}^{M-1} \delta(\mathbf{x} - \mathbf{c}_i) \approx \phi \otimes \sum_{\mathbf{i} \in I_{\tilde{n}}} h_{[\mathbf{x}-\mathbf{i}]} \delta([\mathbf{x} - \mathbf{i}]) \quad (4.2)$$

where  $\phi$  is a locally supported function,  $h$  is the uniform set of coefficients of  $\phi$ .  $I_{\tilde{n}}$  is a cubic set of indices:  $\{\{i, j, k\} : i, j, k \in -n/2 .. n/2 - 1\}$ . The truncated function has support of size  $2m + 1$  and  $m \approx 3$ . The new number of frequencies required are  $\tilde{n} = \alpha n$ ,  $\alpha \approx 2$ . The algorithm uses  $O(Mm^3 + (\alpha n)^3 \log(\alpha n) + Nm^3)$  time and  $O((\alpha n)^3 + N + M)$  memory.

### 4.2.2 Summation to a uniform grid

We can eliminate the errors in the inverse step (from the Fourier coefficients to the summations) by using a 3D IFFT. The algorithm uses  $O(Mm^3 + n^3 \log(n) + N \log(N))$  time and  $O(n^3 + N + M)$  memory.

### 4.2.3 Summation from non-uniform centers to a uniform sub-grid

Consider solving equation (4.1) on a sub-grid of size  $L \times L \times L$  embedded in  $\Pi^3$ . This is especially useful for AMR data visualization. As shown in (4.2), we can represent particle data as a grid  $h$  of compactly supported functions  $\phi$  with a certain desired precision. The output to any subgrid is a convolution of  $\phi$  with the corresponding subgrid in  $h$ . This convolution can be computed efficiently using a Fast Fourier transform in  $O(L^3 \log L)$ . The precomputation cost for obtaining  $h$  is  $O(Mm^3 + n^3 \log(n))$ . Each update for a uniform subgrid costs  $O(L^3 \log L)$ .

## 4.3 Derivatives computation

We have already expressed derivatives as summations of kernel functions. The Fourier transform derivative theorem can be used to obtain these derivatives at uniform grid positions. In equation (4.2), we have a set of compactly supported functions  $\phi$  contributing to each point  $k$ . The derivative of a convolution is the convolution of the derivative of either function with the other. Hence we can express the derivative of the function as:  $f_x(\mathbf{x}) = \sum_{\mathbf{k} \in I_{n,m}(\mathbf{x})} h_k \phi_x(\mathbf{x} - \frac{\mathbf{k}}{n})$ , where  $I_{n,m}(x)$  is the set  $\{l \in I_n : nx - m \leq l \leq nx + m\}$ . The order of the memory and computation costs is the same as that of the fast summation algorithm.

## 4.4 Updating moving sets of particles

In the case of flexible objects, like biomolecules, it is required to update functions and their derivatives due to the displacement of the centers. Hence it is useful to update function  $f$  defined in equation (4.1) when the  $M$  centers move relative to one another over time. If we are given a representation of the object such that centers which move together are grouped, then we can take advantage of that distribution. Otherwise, we can break up the centers spatially in to a collection of sets  $S$ , and consider which of them changed positions over time. In the case of biomolecules, flexible structures naturally give rise to a disjoint union  $S$  of the centers  $M$  (this set could also be hierarchically refined). In biochemical terms, these groups are called domains. Given a set of  $M_d$  points a moving domain, we can compute the new function in  $O(m^3 M_d + \alpha^3 M_d \log(\alpha^3 M_d) + m^3 N)$  time. The result follows directly from the linearity of the summation operation and the fact that  $n_d$  is proportional to  $M_d$  for our application.

## 5 Implementation and Results

We used the FFTW package [14] to perform the fast Fourier transforms. The code was implemented and tested on Linux and SGI. The results were obtained from a single 400MHz processor of a SGI Onyx 2. Since ours is an approximation method, we have used three different error threshold: 1%, 5% and 10%. We provide both  $L^2$  error norms and images in each case. Since we can perform function computation of smoothed particle data to both uniformly and non uniformly distributed output points, we present two different cases: 1). Summation to uniformly distributed output points, 2). Summation to non uniformly distributed output points.

### 5.1 Summation of particle data from Non-uniform centers to uniform output grid points

The fast surface and volume extraction algorithm was used on both SPH and molecular data sets. In order to compare our approximation algorithm with existing techniques, we provide timing and error results with two data sets. In these experiments, we vary: 1). Number of centers  $M$ , 2). Number of output points  $N$ , 3). The drop off parameter of the Gaussian/exponential. The different methods we compare are 1). Truncated kernels brute force summation. The kernels are truncated to lengths depending on the error accepted in a truncation. A simple quadratic cost algorithm is used to compute the summation at the output points. 2). The discretized regular fast Fourier transform method. In this method, each center is discretized to a grid point. The Fourier transform of the entire grid is taken and the summation required at an output point is taken to be the result at the closest grid point. 3). The algorithm described in section § 4 is used. Two different test cases are considered: The myoglobin, a small molecule containing 1221 atoms, blurred to a resolution of  $128^3$  and the large ribosomal subunit, containing 90403 atoms, blurred to a larger resolution of  $512^3$ . Table 5.2.2 shows the timing and memory results for obtaining the volumetric summation of the smoothed atoms of myoglobin. We have chosen a very fine resolution of 1Å and a coarser resolution of 4Å. The 1Å resolution shows very fine atomistic details and the gaussian has a sharp decay, leading to a broader frequency response. In this case, it is seen that the direct summation always performs better than a convolution. Our approximation technique outperforms the FFT in all three error categories. The errors show that even for 1%  $L^2$  error, our method consumes less memory than the FFT. At the 4Å resolution, chosen to show the secondary structure features of the molecule, our approximation algorithm clearly outperforms both the direct and the FFT methods. It should also be noted that as we go to coarser resolutions, our algorithms performance in both time and memory improves while the contrary is true for time in the direct summation case. In Table 5.2.3, in the case of the large ribosomal subunit, we see similar results as above. For fine resolutions, a direct summation is better than our method, while at coarser resolutions, our algorithm vastly outperforms other techniques.

### 5.2 Summation of particle data from non-uniform centers to non-uniform output grid points

Smoothed volumetric particle data functions at some predefined surfaces can be estimated using our approximation algorithm provided in section 4. Also, given an isosurface, it is useful to obtain gradients at the vertices of the mesh. In table 5.2.1, we provide the times taken, in seconds, to perform the summation of 90403 atoms of the large ribosomal subunit. We see an almost linear increase in the time required to compute the output function with respect to the number of output points. Here, we consider building the grid of compactly supported functions representation from the volumetric particle data as a preprocessing step. If we consider the smooth mesh of the large ribosomal subunit contains approximately 10000 vertices, then the estimation of functions like curvatures on it is seen to be in the order of a second. Also, the size of the higher order grid is  $130^3$ , which is compact compared to the  $512^3$  grid required for the whole volume.

## 6 Conclusion

In this paper, we provide a fast, approximate method to compute summations of volumetric smoothed particles and their gradients. Surface extraction and gradient computation from particle data can be performed

| Number of output centers | time in seconds |
|--------------------------|-----------------|
| 1000                     | 0.078514        |
| 10000                    | 0.787210        |
| 100000                   | 7.867189        |
| 1000000                  | 77.809066       |

Table 5.2.1: Timing and errors for fast summation of Gaussian kernels at different number of non-uniformly spaced output points, using  $m = 3$  order B-spline grid obtained from the NFFT. The time reported is the time it takes to compute the function given the sum of b-splines grid representation. We used the large ribosomal subunit electron density blurring at  $1\text{\AA}$  resolution and 10% error. Please refer to table 5.2.3 for the time taken to precompute the grid at different errors and resolution.

| Myoglobin (101M.pdb), 1221 atoms, $48\text{\AA} \times 40\text{\AA} \times 40\text{\AA}$ , opt. grid size = $128^3$<br>FFT method time: 16.509123 |          |       |      |    |           |      |       |    |
|---|----------|-------|------|----|-----------|------|-------|----|
|   | Res=1    |       |      |    | Res=4     |      |       |    |
| Direct  | 6.730830 |       |      |    | 14.733616 |      |       |    |
|   | a        | b     | c    | d  | a         | b    | c     | d  |
| $F.B., \epsilon \approx 1\%$  | 4.08     | 12.26 | 1.08 | 96 | 1.33      | 8.76 | 1.13  | 44 |
| $F.B., \epsilon \approx 5\%$  | 3.01     | 11.49 | 4.74 | 76 | 1.17      | 8.54 | 5.08  | 32 |
| $F.B., \epsilon \approx 10\%$   | 1.69     | 9.87  | 9.30 | 64 | 1.13      | 8.50 | 10.37 | 24 |

Table 5.2.2: Timing (in seconds) and errors ( $L^2$  percent) for fast summation of 1221 Gaussian kernels to a  $128 \times 128 \times 128$  uniform grid. The entries contain the timing in seconds and the actual error for different resolutions.  $m = 3$  was used for the interpolating functions in each case. The notations are: a: Forward time in seconds, b: Full time in seconds, c: Actual error, d:  $\alpha M$ . Since we perform the blurring in frequency space, our method would be much faster than the others at such low frequencies. The quadratic algorithms Gaussian was clamped when it reduced to  $10^{-3}$  of its peak.

| Large ribosomal subunit (1JJ2.pdb), 90403 atoms, $221\text{\AA} \times 221\text{\AA} \times 175\text{\AA}$ , opt. grid size = $512^3$<br>FFT method time: 1425.239929 |            |        |       |     |             |        |       |    |
|---|------------|--------|-------|-----|-------------|--------|-------|----|
|   | Res=1      |        |       |     | Res=6       |        |       |    |
| Direct  | 128.577673 |        |       |     | 5844.995279 |        |       |    |
|   | a          | b      | c     | d   | a           | b      | c     | d  |
| $F.B., \epsilon \approx 1\%$  | 245.53     | 983.21 | 0.98  | 200 | 7.849469    | 760.38 | 0.94  | 60 |
| $F.B., \epsilon \approx 5\%$  | 98.22      | 803.31 | 4.84  | 150 | 4.05421     | 743.42 | 4.78  | 40 |
| $F.B., \epsilon \approx 10\%$   | 71.79      | 790.50 | 10.30 | 130 | 3.211702    | 730.82 | 10.22 | 32 |

Table 5.2.3: Timing (in seconds) and errors ( $L^2$  percent) for fast summation of 90403 Gaussian kernels to a  $512 \times 512 \times 512$  uniform grid. The entries contain the timing in seconds and the actual error for different resolutions.  $m = 3$  was used for the interpolating functions in each case. The notations are: a: Forward time in seconds, b: Full time in seconds, c: Actual error, d:  $\alpha M$ . Since we perform the blurring in frequency space, our method would be much faster than the others at such low frequencies. The quadratic algorithms Gaussian was clamped when it reduced to  $10^{-3}$  of its peak. We used a larger resolution range as this is a relatively large molecule.



using our approximate Fourier transform based method. We provide a compact representation in the form of a grid of compactly supported functions for smooth maps. A fast method for updating moving sets of particles is also presented. All of the algorithms for smooth kernels are shown to be error bounded. Our results (theoretical and experimental) show that the algorithms presented are faster and more memory efficient than traditional techniques for a range of errors. It should be noted that apart from truncated gaussian and bsplines, our algorithms are suited to general smooth RBFs like the commonly used thin-plate spline. As part of future work, we would like to perform adaptive isocontouring directly from the grid of bsplines representation.

A 3D numerical gas dynamics simulation, which uses a anisotropic version of SPH, was developed by Paul Shapiro and Hugo Martel of the Galaxy Formation and the Intergalactic Medium Research Group, The University of Texas at Austin. The data set was used in the generation of the cosmological explosion images shown in this paper. The myoglobin and large ribosomal subunit data sets are available at the Protein Data Bank ([www.pdb.org](http://www.pdb.org), PDB:101M.PDB, PDB:1JJ2.PDB).

## References

- [1] Marc Alexa, Johannes Behr, Daniel Cohen, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
- [2] Nina Amenta and Yong Kil. Defining point-set surfaces. *SIGGRAPH 2004*, pages 264–270, 2004.
- [3] C. Anderson and M.D. Dahleh. Rapid computation of the discrete fourier transform. *SIAM J. Sci. Computing*, 17:913–919, 1996.
- [4] G. Beylkin. On the fast fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363–381, 1995.
- [5] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.
- [6] S. F. Boys. Electronic wave functions, i: A general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 200(1063):542–554, February 1950.
- [7] Julio Castrillon-Candas, Vinay Siddavanahalli, and Chandrajit Bajaj. Nonequispaced fourier transforms for protein-protein docking. ICES Report 05-44, The University of Texas Austin, 2005.
- [8] Bruno Rodrigues de Araujo and Joaquim Armando Pires Jorge. Curvature dependent polygonization of implicit surfaces. *Computer Graphics and Image Processing, XVII Brazilian Symposium on (SIBGRAP'04)*, pages 266–273, 2004.
- [9] Peter Dombrowski. Krümmungsgroben gleichungsdefinierter untermannigfaltigkeiten riemannscher mannigfaltigkeiten. *Mathematische Nachrichten*, 38(3/4):133–180, 1968.
- [10] A. Dutt and V. Rokhlin. Fast fourier transform for nonequispaced data. *SIAM J. Sci. Computing*, 14:1368–1393, 1993.
- [11] A. Dutt and V. Rokhlin. Fast fourier transform for nonequispaced data ii. *Appl. Comput. Harmon. Anal.*, 2:85–100, 1995.
- [12] B. Elbel. Fast fourier transform for non equispaced data. *Approximation theory, C.K. Chui and L.L. Schumaker (eds.), Vanderbilt University Press*, 1998.
- [13] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphics Interface '96*, pages 204–212, 1996.
- [14] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 3, pages 1381–1384, Seattle, WA, May 1998.
- [15] Razif R. Gabdoulhine and Rebecca C. Wade. Analytically defined surfaces to analyze molecular interaction properties. *Journal of Molecular Graphics*, 14(6):341–353, December 1996.
- [16] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Royal Astronomical Society*, 181:375–389, 1977.
- [17] J.A. Grant and B.T. Pickup. A gaussian description of molecular shape. *Journal of Physical Chemistry*, 99:3503–3510, 1995.
- [18] H. Green and W. Lane. Particulate clouds: Dusts. D. Van Nostrand Company, Inc. 1964, 1964.
- [19] John F. Hughes. Differential geometry of implicit surfaces in 3-space – a primer. Technical Report CS-03-05, Computer Science Dept., Brown University, 2003.

- [20] D. Levin. Mesh-independent surface interpolation. In G. Brunnett, B. Hamann, K. Mueller, and L. Linsen, editors, *Geometric Modeling for Scientific Visualization*. Springer-Verlag, 2003.
- [21] Daniel Potts, G. Steidl, and M. Tasche. *Fast Fourier transforms for nonequispaced data: A tutorial in Modern Sampling Theory: Mathematics and Applications*, chapter 12, pages 249–274. 2000.
- [22] Daniel Potts, Gabriele Steidl, and Arthur Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numer. Math.*, 98(2):329–351, 2004.
- [23] George D. Purvis and Chris Culberson. On the graphical display of molecular electrostatic force-fields and gradients of the electron density. *Journal of Molecular Graphics*, 4:88–92, 1986.
- [24] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983.
- [25] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 2. Publish or Perish Inc, Berkeley, USA, 1979.
- [26] G. Steidl. A note on fast fourier transforms for nonequispaced grids. *Advances in Computational Mathematics*, 9:337–352, 1998.
- [27] E. Suli and A. Ware. A spectral method of characteristics for hyperbolic problems. *SIAM J. Numer. Anal.*, 28:423–445, 1991.
- [28] Anthony Ware. Fast approximate fourier transforms for irregularly spaced data. *SIAM Rev.*, 40:838.
- [29] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, August 1986.
- [30] G. Xu and C. Bajaj. Curvature computations of 2-manifolds in  $r^k$ . *Journal of Computational Mathematics*, 21(5):681 – 688, 2003.

## Appendix

### A Fourier series expansion for summations of kernels

Any periodic bounded function can be expanded as a Fourier series. For example, a periodic function in  $[-1/2, 1/2]$  can be expressed as:

$$f(x) = \sum_{j=-\infty}^{\infty} \omega_j e^{2\pi i j x}$$

where the coefficients  $\omega_j = \int_{-1/2}^{1/2} f(x) e^{-2\pi i j x} dx$ . Let  $k$  be the truncated kernel function located at each center. The truncation does not change the problem being solved as we are interested in a finite domain. Let  $I_n$  denote a 3D grid of indices:  $\{k : [-n/2..n/2]^3, k \in I\}$ . Let us expand the kernel function in its Fourier series form:

$$k(\mathbf{x} - \mathbf{x}_j) = \sum_{\omega \in I_\infty} K_\omega e^{2\pi i (\mathbf{x} - \mathbf{x}_j) \cdot \omega}$$

Substituting the expansion for the Kernel in (4.1), we get

$$f(\mathbf{x}) = \sum_{i=1}^M c_j \left( \sum_{\omega \in I_\infty} K_\omega e^{2\pi i (\mathbf{x} - \mathbf{x}_j) \cdot \omega} \right) \approx \sum_{j=1}^M c_j \left( \sum_{\omega \in I_n} K_\omega e^{2\pi i (\mathbf{x} - \mathbf{x}_j) \cdot \omega} \right)$$

We limit the number of required frequencies in  $R^3$  to just  $n^3$ , where  $n^3 = O(M)$  is computed given the error allowed by the user. Now,

$$f(\mathbf{x}) \approx \sum_{\omega \in I_n} K_\omega e^{2\pi i \mathbf{x} \cdot \omega} \sum_{j=1}^M c_j e^{-2\pi i \mathbf{x}_j \cdot \omega}$$

The second sum is the Discrete Fourier transform of the sum of atom centers. Let us denote these coefficients by  $C_\omega$ .

$$f(\mathbf{x}) \approx \sum_{\omega \in I_n} K_\omega C_\omega e^{2\pi i \mathbf{x} \cdot \omega} = \sum_{\omega \in I_n} F_\omega e^{2\pi i \mathbf{x} \cdot \omega} \quad (\text{A.1})$$

**Approximation made in the algorithm** Let us denote by  $f, \hat{f}$  the exact and computed summation. Let the exact and approximate Fourier series coefficients of the function of centers and the truncated kernel be  $C_\omega, \hat{C}_\omega, K_\omega, \hat{K}_\omega$ .

$$\begin{aligned} \|\hat{f} - f\|_2 &= \left\| \sum_{\omega \in I_\infty} C_\omega K_\omega e^{2\pi i \omega \cdot \mathbf{x}} - \sum_{\omega \in I_n} \hat{C}_\omega \hat{K}_\omega e^{2\pi i \omega \cdot \mathbf{x}} + \epsilon_1 \right\|_2 \\ &= \left\| \sum_{\omega \in I_n} (C_\omega K_\omega - \hat{C}_\omega \hat{K}_\omega) e^{2\pi i \omega \cdot \bar{\mathbf{x}}} + \sum_{\omega \in I_\infty \setminus I_n} C_\omega K_\omega e^{2\pi i \omega \cdot \bar{\mathbf{x}}} + \epsilon_1 \right\|_2 \\ &\leq \left\| \sum_{\omega \in I_n} (C_\omega K_\omega - \hat{C}_\omega \hat{K}_\omega) e^{2\pi i \omega \cdot \bar{\mathbf{x}}} \right\|_2 + \left\| \sum_{\omega \in I_\infty \setminus I_n} C_\omega K_\omega e^{2\pi i \omega \cdot \bar{\mathbf{x}}} \right\|_2 + \|\epsilon_1\|_2 \\ &= \sqrt{\left( \sum_{\omega \in I_n} (C_\omega K_\omega - \hat{C}_\omega \hat{K}_\omega)^2 \right)} + \sqrt{\left( \sum_{\omega \in I_\infty \setminus I_n} (C_\omega K_\omega)^2 \right)} + \|\epsilon_1\|_2 \\ &= e_1 + e_2 + e_3 \end{aligned}$$

$e_1$  and  $e_3$  are errors due to NFFT', NFFT. We choose  $m, \alpha$  such that these errors are both less than one third the user allowed error. We are left to choose  $n$  such that the second term is also less than  $\epsilon/3$ . For error bounds for  $m, \alpha$ , please look at [21]. In practise,  $\alpha \approx 2, m \approx 3$  gives us errors less than 1% which is sufficient for our applications.

## B Estimating number of Fourier coefficients required for given precision

In this section, we derive values for  $n, m, \alpha$  to keep the theoretical error within the user defined error threshold  $\epsilon$ . This analysis thus provides us with both a theoretical an upper bound on the computational and storage costs of the algorithm and variables. We need to find the minimum value for  $n$  which satisfies

$$\frac{\|\hat{f} - f\|_2}{\|f\|_2} \leq \frac{\epsilon}{3}$$

**Lemma** For tensor product kernels with Fourier coefficients  $K_\omega$ , the number of coefficients  $n$  needed is atmost:

$$n = \min(\hat{n}) : \sum_{\omega \in I_{\hat{n}}} (K_\omega)^2 \geq \frac{V}{2\pi} - \frac{M \min_j (|c_j|^2) V (\frac{\epsilon}{3})^2}{(\|c\|_1)^2}$$

where  $V$  is the integral of the kernel from  $(-0.5..0.5]^3$ .

**Proof**

We first bound the denominator  $\|f\|_2$  as follows:

$$\begin{aligned} \|f\|_2 &= \sqrt{\int_{I_1} \left( \sum_{j=1}^M c_j K(\mathbf{x} - \mathbf{x}_j) \right)^2 d\mathbf{x}} \\ &\geq \sqrt{\int_{I_1} \sum_{j=1}^M (c_j)^2 (K(\mathbf{x} - \mathbf{x}_j))^2 d\mathbf{x}} \\ &\geq \sqrt{\min_j (|c_j|^2) \int_{I_1} \sum_{j=1}^M (K(\mathbf{x} - \mathbf{x}_j))^2 d\mathbf{x}} \\ &= \sqrt{M \min_j (|c_j|^2) V}, \quad V = \int_{I_1} K(\mathbf{x})^2 d\mathbf{x} \end{aligned}$$

$$|C_\omega| \leq \sum_{j=1}^M |c_j| |e^{-2\pi i \mathbf{x}_j \cdot \omega}| = \sum_{j=1}^M |c_j| = \|c\|_1$$

$$\sum_{\omega \in I_\infty \setminus I_n} (C_\omega K_\omega)^2 \leq \max |C_\omega|^2 \sum_{\omega \in I_\infty \setminus I_n} (K_\omega)^2 = (\|c\|_1)^2 \sum_{\omega \in I_\infty \setminus I_n} (K_\omega)^2$$

Let  $D = M \min_j (|c_j|^2) V$ , error due to term  $e_2 = \epsilon/3$

$$\begin{aligned}
(\|c\|_1)^2 \sum_{\omega \in I_\infty \setminus I_n} (K_\omega)^2 &\leq D\left(\frac{\epsilon}{3}\right)^2 \\
(\|c\|_1)^2 \left( \sum_{\omega \in I_\infty} (K_\omega)^2 - \sum_{\omega \in I_n} (K_\omega)^2 \right) &\leq D\left(\frac{\epsilon}{3}\right)^2 \\
(\|c\|_1)^2 \left( \frac{V}{2\pi} - \sum_{\omega \in I_n} (K_\omega)^2 \right) &\leq D\left(\frac{\epsilon}{3}\right)^2 \\
\sum_{\omega \in I_n} (K_\omega)^2 &\geq \frac{V}{2\pi} - \frac{D}{(\|c\|_1)^2} \left(\frac{\epsilon}{3}\right)^2
\end{aligned}$$

Therefore:  $n = \min(\hat{n}) : \sum_{\omega \in I_{\hat{n}}} (K_\omega)^2 \geq \frac{V}{2\pi} - \frac{D}{(\|c\|_1)^2} \left(\frac{\epsilon}{3}\right)^2$ .

□

**Lemma** For tensor product kernels whose Fourier coefficients decay at least inversely with frequency, the number of coefficients  $n$  needed is  $O(M^{1/3}\epsilon^{3/2})$ .

**Proof**

$$\begin{aligned}
\frac{\sqrt{\sum_{\omega \in I_\infty \setminus I_n} (C_\omega K_\omega)^2}}{\|f\|_2} &\leq \frac{\epsilon}{3} \\
\sum_{\omega \in I_\infty \setminus I_n} (C_\omega K_\omega)^2 &\leq \left(\frac{\epsilon\|f\|_2}{3}\right)^2 \\
\sum_{\omega \in I_\infty \setminus I_n} (K_\omega)^2 &\leq \left(\frac{\epsilon\|f\|_2}{3\|c\|_1}\right)^2 \\
2 \sum_{\omega=n/2+1}^{\infty} (K_\omega)^2 + (K_{n/2})^2 &\leq \left(\frac{\epsilon\|f\|_2}{3\|c\|_1}\right)^{2/3}
\end{aligned}$$

For kernels where  $K_\omega$  decays as  $1/n$ , we have:

$$2 \sum_{\omega=n/2+1}^{\infty} \frac{1}{(K_\omega)^2} + \frac{4}{n^2} \leq \frac{\pi^2 e^2}{32} \left(\frac{\epsilon\|f\|_2}{3\|c\|_1}\right)^{2/3}$$

Using  $\sum_{\omega=n/2+1}^{\infty} 1/(K_\omega)^2 \leq \int_{\omega=n/2}^{\infty} 1/(K_\omega)^2 = 2/n$ ,

$$\frac{1}{n} + \frac{1}{n^2} \leq \frac{\pi^2 e^2}{128} \left(\frac{\epsilon\|f\|_2}{3\|c\|_1}\right)^{2/3}$$

Hence, we would like to obtain the minimum  $n$  which satisfies the above inequality. Since  $\|c\|_1$  is proportional to  $M$  and  $\|f\|_2$  is proportional to  $\sqrt{M}$ , we see that  $n^3$  is proportional to  $M$  as  $n$  grows. □

## B.1 The Fourier coefficients of a truncated Gaussian

We consider the Gaussian kernel  $g(x)$  as we use it extensively in our applications. In general, we can consider any smooth kernel for the analysis. The truncated Gaussian functions Fourier series coefficients  $G_\omega$  are the convolution of a Gaussian with a sinc. We present the decay of  $G_\omega$  and show that it is upper bounded by  $1/\omega$ .

**Lemma** The fourier coefficients of a Gaussian function  $e^{-Bx^2}$  decay as the inverse of the frequency  $\omega$ :

$$G_\omega \leq \max\left(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi} \operatorname{erf}\left(\frac{\pi}{\sqrt{B}}\right), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{Be^{-(1+\pi^2/B)}}{\pi^4}\right) \frac{1}{\omega}, \quad (\omega \geq 2)$$

**Proof**

The truncated gaussian is the product of a gaussian and a box function. In the fourier domain, this can be expressed as a convolution between a sinc and a gaussian. the sinc itself can be upper bounded by a function of type  $1/x$ . Since the second function goes to infinity at 0, we replace the  $1/x$  with a flat function near the origin. Specifically,

$$FT(\text{Box}) = \frac{2 \sin(2\pi t)}{2\pi t} \leq \begin{cases} 2 & -1 \leq t \leq 1 \\ \frac{1}{\pi t} & t > 1 \\ -\frac{1}{\pi t} & t < -1 \end{cases}$$

Also let

$$\hat{K}(t) = FT(K)(t) = \sqrt{\frac{\pi}{B}} e^{-\frac{\pi^2 t^2}{B}}$$

thus we get  $G_\omega, \omega > 1$  as

$$G_\omega = \int_{-\infty}^{\omega-1} \frac{\hat{K}(t)}{\pi(\omega-t)} dt + \int_{\omega-1}^{\omega+1} 2\hat{K}(t) dt + \int_{\omega+1}^{\infty} \frac{\hat{K}(t)}{\pi(t-\omega)} dt$$

We show that this function has a decay of  $O(1/\omega)$ . Also, due to symmetry,  $G_\omega = G_{-\omega}$ .

**I<sub>1</sub> :**

Let

$$I_1 = \int_{-\infty}^0 \frac{\hat{K}(t)}{\pi(\omega-t)} dt, I_2 = \int_0^{\omega-1} \frac{\hat{K}(t)}{\pi(\omega-t)} dt, I_3 = \int_{\omega-1}^{\omega+1} 2\hat{K}(t) dt, I_4 = \int_{\omega+1}^{\infty} \frac{\hat{K}(t)}{\pi(t-\omega)} dt$$

$$I_1 = \int_{-\infty}^0 \frac{\hat{K}(t)}{\pi(\omega-t)} dt \leq \int_{-\infty}^0 \frac{\hat{K}(t)}{\pi\omega} dt$$

Since

$$\int_{-\infty}^0 \hat{K}(t) dt = \int_{-\infty}^0 \sqrt{\frac{\pi}{B}} e^{-\frac{\pi^2 t^2}{B}} dt = \frac{1}{2\sqrt{\pi}}$$

$$I_1 \leq \frac{1}{2\pi\sqrt{\pi}} \frac{1}{\omega}$$

**I<sub>2</sub> :**

$$I_2 = \int_0^{\omega^{-1}} \frac{\hat{K}}{\pi(\omega-t)} dt = \int_0^1 \frac{\hat{K}}{\pi(\omega-t)} dt + \int_1^{\omega^{-1}} \frac{\hat{K}}{\pi(\omega-t)} dt = I_{2,1} + I_{2,2}$$

Bounding  $I_{2,1}$ :

$$I_{2,1} = \int_0^1 \frac{\hat{K}}{\pi(\omega-t)} dt \leq \int_0^1 \frac{\hat{K}}{\pi\omega} dt = \frac{1}{\omega\sqrt{\pi B}} \int_0^1 e^{-\pi^2 t^2/B} dt$$

Let  $y = \pi t/\sqrt{B}$ . Hence,  $y^2 = \pi^2 t^2/B$ ,  $dt = \sqrt{B}/\pi dy$ ,  $t = 0 \rightarrow y = 0$  and  $t = 1 \rightarrow y = \pi/\sqrt{B}$ .

therefore,

$$I_{2,1} \leq \frac{1}{\omega\pi\sqrt{\pi}} \int_0^{\pi/\sqrt{B}} e^{-y^2} dy = \frac{1}{2\pi\omega} \left(\frac{2}{\sqrt{\pi}}\right) \int_0^{\pi/\sqrt{B}} e^{-y^2} dy = \frac{1}{2\pi\omega} \operatorname{erf}\left(\frac{\pi}{\sqrt{B}}\right)$$

Bounding  $I_{2,2}$ :

Let

$$f_1 = \sqrt{\frac{\pi}{B}} e^{-\pi^2 t^2/B}, \quad f_2 = \frac{c}{t^2}$$

$$\operatorname{Roots}(f_1 - f_2) = \left( \frac{\sqrt{-LW\left(\frac{-\pi^2 c}{\sqrt{\pi B}}\right)B}}{\pi}, \frac{\sqrt{LW\left(\frac{-\pi^2 c}{\sqrt{\pi B}}\right)B}}{\pi} \right)$$

We used Maple to compute the roots. LW stands for the LambertW function. Since roots of complex numbers are complex, and using the properties of the LambertW function, there are no real roots (letting  $f_2$  bound  $f_1$ ) if:

$$c \geq \frac{\sqrt{B}}{e\pi^{3/2}}$$

$$\int_1^{\omega^{-1}} \frac{1}{\pi(\omega-t)} \frac{1}{t^2} dt = \frac{1}{\omega} \left( \frac{\log(\omega-1)}{\omega} - \frac{1}{\omega-1} - \frac{\log(2\omega-1)}{\omega} + 1 + \frac{\log(\omega+1)}{\omega} \right) \leq \frac{3}{\omega}$$

$$I_{2,2} \leq 3 \frac{\sqrt{B}}{e\pi^{3/2}} \frac{1}{\omega}$$

**I<sub>3</sub>** :

$$I_3 = \int_{\omega^{-1}}^{\omega+1} 2\hat{K} dt \leq 4\sqrt{\frac{\pi}{B}} e^{-\pi^2(\omega-1)^2/B}$$

Let

$$I_3 \leq \frac{c}{\omega}, \hat{\omega} = \omega - 1, f_1 = \frac{d}{\hat{\omega}}, f_2 = 4\sqrt{\frac{\pi}{B}}e^{-\pi^2\hat{\omega}^2/B}$$

$$\text{Roots}(f_1 - f_2) = \frac{d}{\sqrt{\pi/B}}\sqrt{\frac{-LW(-1/8\pi d^2)}{2\pi d^2}}$$

there is no intersection when there is no real root. this implies that  $f_1$  is an upper bound as required. For no real roots we get:

$$d \geq \sqrt{\frac{8}{\pi e}} \rightarrow c > 4\sqrt{2\pi e}$$

$$I_3 \leq 4\sqrt{\frac{2}{\pi e}} \frac{1}{\omega}$$

**I<sub>4</sub>** :

$$I_4 = \int_{\omega+1}^{\infty} \frac{\hat{K}}{\pi(t-\omega)} dt \leq \int_{\omega+1}^{\infty} \frac{\hat{K}}{\pi} dt \leq \int_{\omega+1}^{\infty} \frac{\pi}{B} \frac{e^{-\pi^2 t/B}}{\pi} dt = \frac{e^{-\pi^2(\omega+1)/B}}{\pi^2}$$

Let

$$I_4 \leq f_1 = \frac{c}{\omega}, f_2 = \frac{e^{-\pi^2(\omega+1)/B}}{\pi^2}$$

$$\text{Roots}(f_1 - f_2) = \frac{-B LW\left(\frac{-\pi^4 c e^{\pi^2/B}}{B}\right)}{\pi^2}$$

there is no intersection when there is no real root. this implies that  $f_1$  is an upper bound as required. For no real roots we get:

$$\frac{-\pi^2 c e^{\pi^2/B}}{B} < -e^{-1} \rightarrow I_4 \leq \frac{B e^{-(1+\pi^2/B)}}{\pi^4} \frac{1}{\omega}$$

$$\therefore G_\omega \leq \max\left(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi} \text{erf}\left(\frac{\pi}{\sqrt{B}}\right), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{B e^{-(1+\pi^2/B)}}{\pi^4}\right) \frac{1}{\omega}, \quad (\omega \geq 2)$$

□

For our applications, for all practical values of  $B$ :

$$G_\omega \leq 4\sqrt{\frac{2}{\pi e}} \frac{1}{\omega} \tag{B.1}$$