Electrostatic Fields without Singularities: Implementation and Experiments

Paolo Bientinesi* Marco Pellegrini†

B4-97-16

November 1997

Abstract

The following problems which arise in the computation of electrostatic forces and in the Boundary Element Method are considered. Given two convex interior-disjoint polyhedra in 3-space endowed with a volume charge density which is a polynomial in the Cartesian coordinates of $\mathbb{R}^3$, compute the Coulomb force acting on them. Given two interior-disjoint polygons in 3-space endowed with a surface charge density which is polynomial in the Cartesian coordinates of $\mathbb{R}^3$, compute the normal component of the Coulomb force acting on them. A new approach to these problem, based on techniques from integral geometry, computational geometry and numerical analysis, has been proposed by the second author [Pel96, Pel97b]. In this technical report several algorithms based on this approach are compared. For the second problem experimental results support the exponential asymptotic error bounds predicted for a Gaussian quadrature adaptive algorithm.

*Corso di Laurea in Scienze dell'Informazione, Pisa (Italy). E-mail: bientin@cli.di.unipi.it

†Istituto di Matematica Computazionale del CNR, Via S.Maria 46, 56126-Pisa (Italy). E-mail: pellegrini@iei.pi.cnr.it

1 Introduction

In physics it is common to find mathematical formulas which are difficult to handle from a computational point of view. In this paper, we to take in consideration the integrals of the electrostatic field. Techniques used in this paper have been useful also in other contexts such as radiosity (computation of form-factors [Pel97a]), and the integrals of the potential field (computation of potential energy [Fin96]). Close form solutions of the integrals we are going to consider are known only in special cases, therefore numerical methods are needed. The main problem encountered in numerical evaluation is the presence of singularities within the domain of integration.

In [Pel96, Pel97b] Pellegrini has introduced a geometric interpretation of the electrostatic field based on integral geometry, from which several approximation algorithms are derived in the same papers. The purpose of the present technical report is to describe implementation details of such algorithms and report on the experimental results.

Although in [Pel97b] the method has been extend to handle any polynomial distribution of charge, here we will consider mostly constant distributions of charge. There are two reasons for this choice: first of all results on constant distributions of charge are already significant with respect to precision and computing time. Secondly it is difficult to obtain reliable reference values to which compare our methods. So far we could obtain such reference values for uniform charge density.

1.1 Electrostatic Field

Let q_1 and q_2 be two particles of charge respectively in position p_1 and p_2; we have that $\vec{F}_{12}$, the force that q_1 exerts over q_2, is determined by Coulomb's law:

$$\vec{F}_{12} = \frac{q_1 q_2}{4\pi\epsilon_0 |p_1 - p_2|^2} \frac{p_1 - p_2}{|p_1 - p_2|}.$$

If we consider two bodies, B_1 and B_2, with uniform charge densities ρ_1 and ρ_2 instead of two particles of charge, the force that B_1 exerts over B_2 is obtained integrating Coulomb's law over the point of the two bodies:

$$\vec{F}_{12} = \int_{p_1 \in B_1} \int_{p_2 \in B_2} \frac{\rho_1 \, d\rho_1 \, \rho_2 \, d\rho_2}{4\pi\epsilon_0 |p_1 - p_2|^2} \frac{p_1 - p_2}{|p_1 - p_2|}. \tag{1}$$ eq:Coulomb

From here on we consider the instance of the general problem where bodies B_1 and B_2 are tetrahedra, calling them T_1 and T_2. If we want to compute the x-component of the force the above integral becomes:

$$\frac{\rho_1 \rho_2}{4\pi\epsilon_0} \iiint\limits_{T_1} \iiint\limits_{T_2} \frac{(x_1 - x_2) dx_1 \, dy_1 \, dz_1 \, dx_2 \, dy_2 \, dz_2}{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{3/2}} \tag{2}$$

Coulomb-x where $(x_1, y_1, z_1) \in T_1$ and $(x_2, y_2, z_2) \in T_2$. There are several methods for getting an approximation of this integral, but the main problem is that the integrand function has a singularity in 0; that's why it's important to try to look at the problem from a different point of view.

1.2 Volume-to-volume integrals

Now we will summarize a new interpretation of the electrostatic field $\vec{F}$, taken from [Pel96]. The basic idea behind the new definition is that electrostatic forces act along straight lines in a homogeneous medium. For each direction $\vec{L}$, the vectorial contribution to the electrostatic field is related to the polygon P obtained by the intersection of the projections of the tetrahedra over a plane orthogonal to direction $\vec{L}$, (fig. 1).

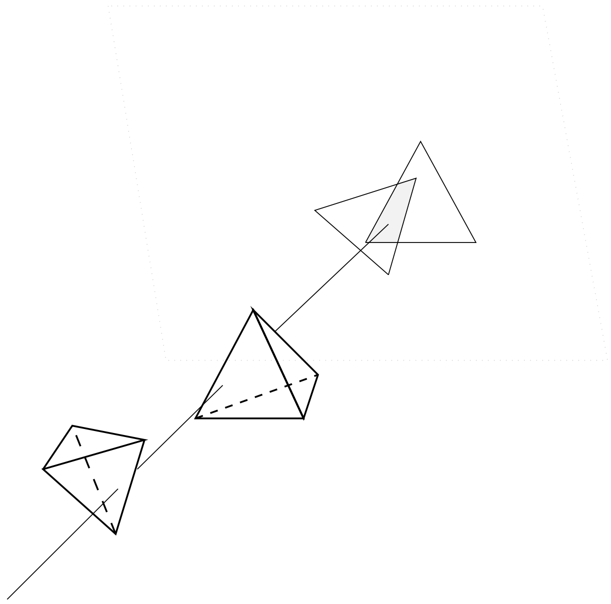

Figure 1: Intersection area of the projections of the tetrahedra fig:P

More specifically it is proved in [Pel96] the following equality:

$$\vec{E}(p) = \int_{L\cap p\neq\emptyset} l(L\cap T)\rho\vec{L}\,dL, \qquad (3)$$ eq:geomp

where p is the point, T is the tetrahedron with uniform volume charge density ρ, L is an oriented line in 3-space, $\vec{L}$ is the versor along L, l is the measure of a segment and dL is the differential measure of lines in three dimensional space. Working in the Gaussian unit system the factor $1/(4\pi\epsilon_0)$ is equal to one. The domain of integration is the set of all lines through the point p.

The force between two tetrahedra $T_1$and T_2 can be found integrating the field generated by one tetrahedron over the points of the second tetrahedron:

$$\vec{F}_{12} = \int_{p\in T_2} [\int_{L\cap p\neq\emptyset} l(L\cap T_1)\rho_1\vec{L}\,dL]\,\rho_2\,dp. \qquad (4)$$ eq:force1

In [Pel96] it is shown that the above integral can be reduced to the following integral defined over the set of lines meeting both tetrahedra:

$$\vec{F}_{12} = \int_L l(L \cap T_1)\rho_1 l(L \cap T_2)\rho_2 \vec{L}\, dL. \tag{5}$$
eq:force1bis

Now, we consider only lines with effective contribution and we write dL, using notation of exterior calculus, as $du \wedge dq$ [San67], where du is the differential measure of directions and dq is the differential measure of the area at the point q which is the intersection of L with the plane orthogonal to L and passing from the origin. In few steps we can rewrite (4) as:

$$\vec{F}_{12} = \rho_1\rho_2 \int_{u \in U} [\int_q l(L(u,q) \cap T_1)\, l(L(u,q) \cap T_2) dq] \vec{u}\, du. \tag{6}$$
eq:force2

Where U is the set of unoriented directions and $L(u,q)$ is the line whose direction is u and passing from the point q. That is the integral we have to approximate. It's important to stress that the inner scalar integral:

$$V_{12}(u) = \int_q l(L(u,q) \cap T_1)\, l(L(u,q) \cap T_2)\, dq \tag{7}$$
eq:V12

does not diverge, moreover as we will show, it will be computed exactly.

In the next subsections we will mention some general strategies for the evaluation of integral (5), namely Monte Carlo, Quasi Monte Carlo and Gaussian quadrature.

1.3 Surface-to-surface integrals

The theory in the previous subsection applies to extended bodies in three-space (volume-to-volume integrals). In applications such as the boundary element method it is important to consider also the case of flat objects endowed with surface densities of charge (surface-to-surface integrals). Considering two triangles T_1 and T_2 endowed with charge densities σ_1 and σ_2, it is proved in [Pel97b] that the component C of $\vec{F}_{12}$ along the normal to T_1 is:

$$C = \int_{u \in U} [\frac{1}{\cos\psi(u)} \int_q \sigma_1(p)\sigma_2(p')\, dq]\, du,$$

where p and p' are the points where the line $L(u,q)$ respectively meets T_1 and T_2 and $\psi(u)$ is the angle formed by $L(u,q)$ and the normal to T_2. We switch to polar coordinates for direction $u = u(\phi,\theta)$ where θ is the angle within u and the x-y plane and ϕ is the angle of the projection of u on the x-y plane and the x axis.

$$C = \int_\phi \int_\theta [\frac{\cos\theta}{\cos\psi(u)} \int_q \sigma_1(p)\sigma_2(p')\, dq]\, d\theta\, d\phi. \tag{8}$$
C

Calling $P(u)$ the polygon obtained by the intersection of the projections of T_1 and T_2 in direction u, if the functions σ_1 and σ_2 are constant, $\int_q \sigma_1(p)\sigma_2(p')\, dq$ reduces to the computation of the area of $P(u)$.

1.4 Monte Carlo Methods

The general form of the numerical integration problem is $I = \int_G f(x)\,dx$, ([HH64] for a full explanation), where the integration domain G has finite Lebesgue measure. Monte Carlo methods applied to this kind of problem yield an approximation I' of I by the following formula:

$$I = \int_G f(x)\,dx \approx \frac{\lambda(G)}{N} \sum_{n=1}^{N} f(x_n) = I',$$

where λ denotes the Lebesgue measure function and $x_1, \ldots, x_n \in G$ are N independent random samples over G. Referring to our instance, we have

$$I = \vec{F}_{12} \approx \rho_1 \rho_2 \frac{2\pi}{N} \sum_{i=1}^{N} V_{12}(u_i)\vec{u_i} = I',$$

where u_i are directions chosen uniformly at random over the unit semi-sphere.

Thanks to this method we can get an (ϵ, δ)-approximation of $\vec{F}_{12}$, an approximation that is with probability $1 - \delta$ within an absolute error ϵ from the correct value of $\vec{F}_{12}$, choosing $N = O\left(\frac{1}{\epsilon^2 \delta}\right)$ [1]. Better asymptotic results can be achieved by using a trick described in [JVV86]: instead of computing the mean value out of $N = k\frac{1}{\epsilon^2\delta}$ directions, we calculate the median value of $O(\log(1/\delta))$ experiments made with $O(4/\epsilon^2)$ directions. The total number of evaluations of $V_{12}(u)$is then $O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ that is asymptotically better than $N = O\left(\frac{1}{\epsilon^2\delta}\right)$.

Naturally in computer implementations of the Monte Carlo method we have to rely on pseudo random generators to simulate random numbers.

1.5 Quasi Monte Carlo Methods

The basic idea of Quasi Monte Carlo methods is to choose in a deterministic way the sampling points instead of trying to simulate randomness (see Niederraiter for a detailed treatment of the Quasi Monte Carlo methods, [Nie92]). So the Quasi Monte Carlo formula to approximate $I = \int_G f(x)\,dx$ is:

$$I = \int_G f(x)\,dx \approx \frac{\lambda(G)}{N} \sum_{n=1}^{N} f(x_n) = I'',$$

where $x_1, x_2, \ldots, x_n \in G$ this time are points chosen deterministically.

Error analysis in [Nie92] shows that $|I'' - I|_N$, that is the absolute error obtained approximating I with I'' using N sampling points, is directly proportional to the *discrepancy* of the set $x_1, x_2, \ldots, x_n$. Given a set of points Q, the discrepancy $D(Q)$ can be intended as a measure of the uniformity of the points of Q and, if we fix the number N of elements of the set —obtaining the so called *point set*—, then $D(Q) = O\left(\frac{1}{N}\right)$. This is another improvement compared to the $O\left(\frac{1}{\sqrt{N}}\right)$ achieved by the Monte Carlo methods. Since, in

[1] Pellegrini [Pel96, par.4] determined also the constant factor

numerical integration problems, the evaluation of function is often the costly operation, the re-utilizations of points is an important aspect of the game; we talk about *sequence* as a completely determined infinite sequence of points S, whose first N terms are considered. In this case we have that $D_N(S) = O\left(\frac{\log N}{N}\right)$. It could be argued that a *grid* would be a good choice of points; actually, the grid is the simplest Quasi Monte Carlo point set but it has a couple of drawbacks: its discrepancy is quite large, it means that there are many other sets that simulate uniformity much better, and the only way to reuse its points is by doubling the number of points.

1.6 Gaussian Formulas

Gaussian formulas are interpolatory formulas in which both nodes and weights are chosen with the purpose to maximize the precision order (a formula has precision order k if it exactly approximates all the polynomials of degree $\leq k$). The n nodes of the n-th formula are the n zeroes of the orthogonal polynomial of degree n defined over the integration interval with respect to the weight function 1. Since it is possible to transform the integration interval according to the following formula,

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{(b-a)}{2}t + \frac{(a+b)}{2})dt,$$

we can always use the Legendre orthogonal polynomials that are defined over the range $[-1, 1]$. Weights of Gaussian formulas can be expressed explicitly and they all are positive. This guarantees the theoretical convergence of formulas when the number of nodes increases. Gaussian formulas attain the maximum precision order possible for an interpolatory formula ([BBCM92, DR84]).

We had to generate sampling points in a plane, so we used the Cartesian product of two Gaussian formulas of the same order.

1.7 Organization of the Technical Report

In section 2 we describe common features of all the implementations for the body-to-body integral. In section 3 we describe a set of experiments with volume-to-volume integral, in particular we describe the inputs, how to compute the reference value and several sampling strategies. We give tables, plots and we discuss the results. In section 4 we describe the set of experiments with surface to surface integrals, following the blueprint of the previous section.

2 Implementation

`Implementation`

We have implemented all of the methods discussed so far: Monte Carlo, Monte Carlo with median, grid, Quasi Monte Carlo and Gauss. The inputs are the coordinates of the tetrahedra vertices, expressed as triple of reals and the number of sampling directions (`N`) required; the output is an approximated value of the x-projections of the force vector acting between those tetrahedra, assuming that the volume density charges are constantly equal to 1. The general scheme consists of a `for` loop like this one:

```
for (i=0; i<N; i++)
    {
    u = generate_direction;
    if intersection(u,T1,T2)
      then compute_contribution(u,T1,T2);
    }
```

Thus it is straightforward to divide the program in three main sections: the generation of a sampling direction, the computation of the intersection area (if any) of the projections of the tetrahedra, (fig. 1), and the computation of the inner integral $V_{12}(u)$, (7).

2.1 Direction generation

sec:DirGen We have to create sample directions over $\mathbb{R}^3$; they must be random for the Monte Carlo methods while they are completely predetermined for the Quasi Monte Carlo and grid methods. We obtain random directions generating random points over the unit sphere as in [Knu69]. Other algorithms exist, but all of them are essentially the same. If the tetrahedra are spaced far from one another, then most directions will have null contribution in computation of the field $\vec{E}$. In this case it is helpful to implement an additional tool to determine a spherical polygon P over the unit sphere such that the points inside P represent the only directions with effective contribution. Then we will just have to generate sampling points over that polygon; we can do it triangulating P and using the algorithm suggested in [Arv95].

With the Quasi Monte Carlo and Grid methods, since we use the transformation to polar coordinates, we have to generate sampling points (ϕ, θ) over the rectangle $[0, 2\pi] \times [0, \pi/2]$; for the former method we choose the the *N-element Hammersley point set in base three* [Nie92] that is:

$$(x_n, y_n) = \left(\frac{n}{N}, \phi_3(n)\right) \text{ for } n = 0, 1, \ldots, N-1$$

where $\phi_b(n)$ is the *radical inverse function in base b*. $(x_n, y_n) \in [0,1]^2$, so we get a point inside the rectangle by $(2\pi\ x_n, 2\ y_n/\pi)$.

Given the number N of desired sampling directions, we determine the points for the grid method forcing that the number of points on the ϕ axes are four times the number of points on the θ axes. That is $(\phi_i, \theta_j) = (\frac{2i\pi}{\lceil\sqrt{N}/2\rceil}, \frac{j\pi}{2\lceil\sqrt{N}/2\rceil})$, with $i = 0, \ldots, 4\lceil\frac{sqrtN}{2}\rceil - 1$ and $j = 0, \ldots, \lceil\frac{sqrtN}{2}\rceil - 1$.

2.2 Intersection

Given a direction $\vec{L}$ and the vertices of the tetrahedra T_1 and T_2 we have to determine whether their projections over a plane orthogonal to $\vec{L}$ intersect or not. We use Euler's rotations matrices to determine the projections. We use standard techniques in computational geometry [PS85, O'R94]. More specifically, for each tetrahedron we compute the planar-coordinates of all the vertices projected over a plane orthogonal to $\vec{L}$ and the convex hull of these 4 points by the Jarvis march algorithm. These two polygons P_1 and P_2 (they can be triangles or quadrilaterals) can be considered as made up by 3 or 4 sub-faces corresponding facets of the tetrahedron. We obtain the intersection polygon P from P_1 and P_2 as the union the pairwise intersection of the sub-faces of P_1 with those of P_2.

2.3 Contribute in a fixed direction

Once we have established that given a direction $\vec{L}$ the polygon P exists, we have to exactly compute $V_{12}(u)$, (7), where u is the point on the unit sphere corresponding to the direction $\vec{L}$. Instead of using a numerical integration routine, we make a symbolic calculation. We divide P in k sub-polygons P_k such that all the lines over $\mathbb{R}^3$ parallel to L and intersecting P_k always intercept the same faces of T_j (henceforth called the *covering* faces of P_k).

Let us denote with X, Y, Z the Cartesian reference frame where Z is the coordinate along the u direction and X and Y are coordinates on the plane orthogonal to u. For each sub-polygon P_k let $Z_j^0(X,Y)$ and $Z_j^1(X,Y)$, $(j \in 1,2)$, the equations of the planes spanning the facets covering P_k. Note that $l(L \cap T_j)$ corresponds to $|Z_j^0 - Z_j^1|$. So we can rewrite (7) as:

$$V_{12}(u) = \sum_k \int_{P_k} |(Z_1^0 - Z_1^1)(Z_2^0 - Z_2^1)| \, dX \, dY. \qquad (9)$$ eq:symbolic

The general form of Z_j^i is $aX + bY + c$, where a, b, c are determined from the vertices of the faces intercepted by L; polygons P_k can have at most 6 vertices, so they can be easily triangulated, creating s triangles T_s; so expand (9):

$$V_{12}(u) = \sum_k \sum_s \int_{T_s} |[(a_1^0 - a_1^1)X + (b_1^0 - b_1^1)Y + (c_1^0 - c_1^1)][(a_2^0 - a_2^1)X + (b_2^0 - b_2^1)Y + (c_2^0 - c_2^1)]| \, dX \, dY.$$

Grouping similar monomials and calling f, g, h, i, l, m the resulting coefficients, we obtain the following formula:

$$V_{12}(u) = \sum_k \sum_s \int_{T_s} |f\,X^2 + g\,XY + h\,X + i\,Y^2 + l\,Y + m| \, dX \, dY.$$

It's now immediate to compute the primitive and express the inner integral as function of the vertices of the tetrahedra and of the triangle over which we are integrating.

3 Experiments: Volume-to-volume integral

ExpVtoV In this section we compare the computation of the component of $\vec{F}_{12}$ (6) in direction $\vec{x}$, setting ρ_1 and ρ_2 to 1, considering the factor $\frac{1}{4\pi\epsilon_0} = 1$, obtained by several methods.

3.1 Computing equipment

All the experiments have been made with a Pentium 100 processor with 32Mb of Ram; the methods derived from the Geometric Field —Monte Carlo, Monte Carlo with median, Grid, Gauss and Quasi Monte Carlo, have been implemented in C-language and run under the X-Windows graphical user interface of the Linux operating system.

3.2 Computation of reference values

The reference values for the tests have been calculated by the Mathematica software (version 2.2.2), run under the Microsoft Windows '95 operating system. We started from the integral based on Coulomb's law:

$$\iiint_{T_1}\iiint_{T_2} \frac{dx_1\,dy_1\,dz_1\,dx_2\,dy_2\,dz_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}\cos\phi\,\cos\theta, \tag{10}$$

eq:integrale

where $p_1=(x_1,y_1,z_1)\in T_1, p_2=(x_2,y_2,z_2)\in T_2$, θ is the angle that the line $\overline{p_1p_2}$ creates with the plane x-y, and ϕ is the angle formed by the plane x-z and the line $\overline{p_1'p_2'}$, being $\overline{p_1'p_2'}$ the projection of $\overline{p_1p_2}$ on the x-y plane.

Then it's straightforward to write (10) as in 1.1:

$$\iiint_{T_1}\iiint_{T_2} \frac{|x_2-x_1|}{[(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2]^{\frac{3}{2}}}dx_1\,dy_1\,dz_1\,dx_2\,dy_2\,dz_2.$$

In our experiments we split the domain in two subdomains where the sign of x_2-x_1 is constant, therefore we can drop the absolute value. In order to produce random points in the unitary cube in $\mathbb{R}^6$, we used the standard Mathematica random point generator to get a pair of 3-coordinates. Then each triple of coordinates represents a point is tested for inclusion in the tetrahedra. A 6-coordinates point is useful for the computation if both the triples represent points inside the tetrahedra. The final value is the mean of 60 computations of $150,000$ useful 6-dimensional points (in total about 36 hours of computing time).

The same algorithm have been implemented in C code for purpose of comparing the running time with C implementation of other algorithms on the same number of directions in the range $n_0=10,\ldots,n_{50}=10,000$. The 6-dimensional points are chosen uniformly at random in the Monte Carlo Coulomb method (MCC), and using the Hammersley Point set of dimension six in Quasi Monte Carlo Coulomb method (QMCC) according to the following formula:

$$(x_n,y_n)=\left(\frac{n}{N},\phi_2(n),\phi_3(n),\phi_5(n),\phi_7(n),\phi_{11}(n)\right)\text{ for } n=0,1,\ldots,N-1,$$

where $\phi_b(n)$ has been defined in section (2.1).

3.3 Inputs

We made three sets of experiments, considering critical tetrahedra positions: T_1 will always be the 'unit' tetrahedron: $\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$; in the first set of tests T_2 is symmetrical to T_1 respect to the origin, (fig. 2).

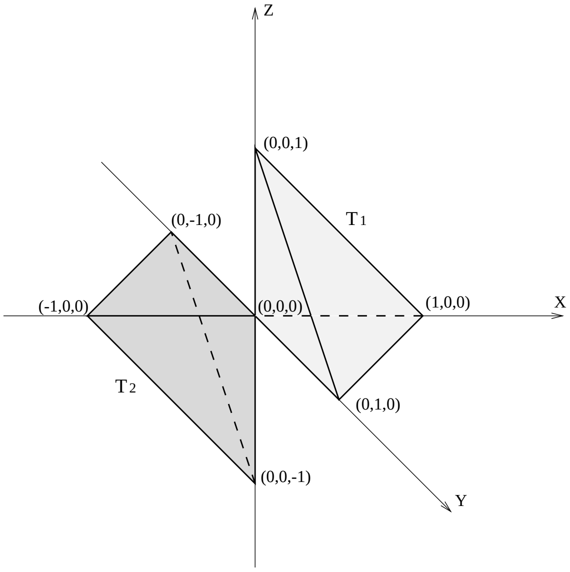

Figure 2: T_1 and T_2 share a vertex fig:tet1

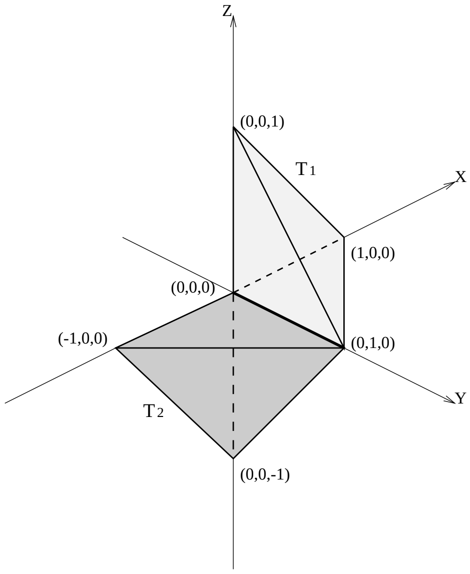

Figure 3: T_1 and T_2 share one edge fig:tet2

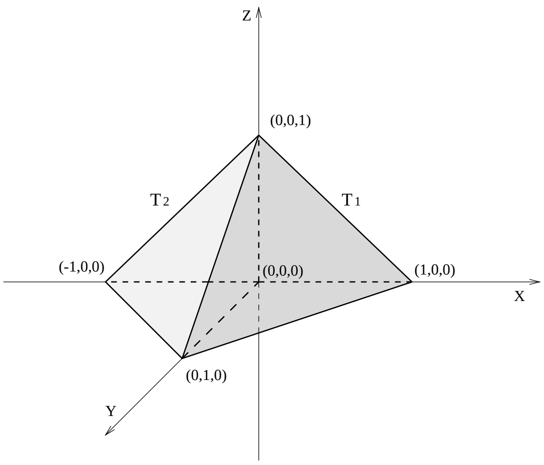

Figure 4: T_1 and T_2 share one face fig:tet3

Thus T_1 and T_2 g share the $(0,0,0)$ vertex. In the second experiments, T_2 is: $\{(0,0,0),(-1,0,0),(0,1,0),(0,0,-1)\}$, so the tetrahedra share the edge $\{(0,0,0),(0,1,0)\}$, (fig. 3). In the third set of experiments we take in consideration the case in which T_1 and T_2 share a face: T_2 is the symmetric of T_1 respect to the face $\{(0,0,0),(0,1,0),(0,0,1)\}$, (fig. 4).

The test consists in running the algorithms with $n_0 = 10, \ldots, n_i, \ldots, n_{50} = 10,000$ directions where n_i, with $i = 0, \ldots, 50$, are such that they are equally spaced in the logarithmic scale. We have that:

$$n_{i+1} = \lceil n_i\, e^{\Delta} \rceil \,\text{, where } \Delta = \frac{\log 10,000 - \log 10}{50},$$

so $n_0 = 10, n_1 = 12, n_2 = 14, \ldots$. Because of the ceiling function the last step $n_{49} - n_{50}$ doesn't match the definition, since n_{50} would be greater than $10,000$.

3.4 Algorithms

3.4.1 Monte Carlo (MC)

This is the basic algorithm, the outcome is just the mean value of the contributions of N directions; its quality, especially for big values of N, depends greatly on the random number generator. For the sake of the uniformity of randomness, we didn't keep trace of the former results, calculating n_i new directions for each i.

3.4.2 Monte Carlo with Median (Med)

Instead of a plain mean out of N directions, we take as result the median of 10 experiments run with $\lfloor \frac{N}{10} \rfloor$ directions.

3.4.3 Grid

As said before the grid is built upon the constriction that the points on the ϕ edge of the grid must be four times the number of the points on θ edge. This affects the number of sampling points, often greater than the fixed N, so if N is small ($N < 250$), we can get the same set of grid points (and consequentially the same results) for different values of N. We also implemented experiments with square grids and the results are substantially the same.

3.4.4 Gauss non adaptive (Gauss)

Here we didn't fix N, we just used Gaussian square grids over the rectangle $[0, 2\pi] \times [0, \pi/2]$. A Gaussian grid is the product of two Gaussian formulas; they are squared because we took the same number of nodes for both axes. The nodes and the weights are pre-computed, this means that the algorithm itself is faster but it needs a pre-processing time. The biggest grid is $45 * 45 = 2025$ directions because of problems of numerical instability that arise starting from the $43 * 43$ grid.

3.4.5 Quasi Monte Carlo (QMC)

Since the values of N are relatively small, and we are not interested in reusing the evaluations of the integrand functions, we used the Hammersley point set to produce sampling points; it is possible to gain something in terms of speed, and probably reducing a bit the likelihood of error, implementing sequences.

3.5 Tables

In this section we show in tabular form experimental results for tetrahedra sharing one face (subsection 3.5.1, fig. 4) , one edge (subsection 3.5.2, fig. 3) and one vertex (subsection 3.5.3, fig. 2). In each subsection the first table gives the relative error as a function of the number of directions for the algorithms based on formula (6). The second table gives the error as function of the number of points in the six-dimensional space for algorithms based on formula (10). The third table gives the precision obtained after a certain as elapsed from the beginning of the computation for a selected number of methods. Data of this section is a significant sample of the date used to produce plots shown in section (3.6).

3.5.1 Tetrahedra sharing one face

N	MC	Med	Grid	QMC	Gauss
10	0.5652615774	0.0928976356	1.1388556434	0.4350593236	0.7608771551
50	0.0021370464	0.3820793616	0.6625503084	0.0591742188	0.1397661223
100	0.1115409971	0.0574428973	0.3580519776	0.0251656549	0.0341402263
500	0.0349494872	0.0233822627	0.1509617357	0.0044767558	0.0076053915
1000	0.0102702892	0.0701072382	0.1020022204	0.0023132319	0.0035086255
5000	0.0149894773	0.0269422073	0.0445899522	0.0007431816	-
10000	0.0057333530	0.0100745621	0.0320705985	0.0005778573	-

Table 1: One face shared: Number of directions/Relative error

N	MCC	QMCC
10	0.4925783479	0.0020888978
50	0.1828346891	0.7419726416
100	0.2753769242	0.0980607779
500	0.5196132812	0.0356476756
1000	0.0373248485	0.1621276607
5000	0.0710196972	0.0127689165
10000	0.0182647919	0.0038469034

Table 2: One face shared: Number of 6-dimensional points/Relative error

sec.	QMCC	Gauss	QMC
0.1	0.1454315113	0.2536975336	0.1536621516
0.5	0.0356476756	0.0341402263	0.0251656549
1	0.0439055117	0.0195223527	0.0109789399
2	0.0512893272	0.0094273259	0.0051045192
5	0.0341901838	0.0087351927	0.0023132319
10	0.0230681231	0.0033960990	0.0013137913
15	0.0038469034	-	0.0009976352

Table 3: One face shared: Time/Relative error

3.5.2 Tetrahedra sharing one edge

N	MC	Med	Grid	QMC	Gauss
10	0.3122420878	0.3837482787	0.2172313869	0.5626566890	0.0244223926
50	0.2514247500	0.3021220980	0.0803293451	0.0854972999	0.0159342981
100	0.1793142522	0.4668144707	0.0274327455	0.0406854481	0.0057290342
500	0.0080051122	0.0324433004	0.0057558990	0.0042811140	0.0006895191
1000	0.0057906254	0.0127611581	0.0143108836	0.0017446695	0.0004992071
5000	0.0217495062	0.0559701770	0.0063938419	0.0003233382	-
10000	0.0128011429	0.0388325702	0.0042378283	0.0002580829	-

Table 4: One edge shared: Number of directions/Relative error

N	MCC	QMCC
10	0.1876232925	0.0952700954
50	0.0306540475	0.1388792072
100	0.1414152941	0.0986771464
500	0.0015847394	0.0556014040
1000	0.1551862698	0.0227489575
5000	0.0188877688	0.0123905844
10000	0.0076069421	0.0166686442

Table 5: One edge shared: Number of 6-dimensional points/Relative error

sec.	QMCC	Gauss	QMC
0.1	0.0724711226	0.0751396871	0.2394331843
0.5	0.0556014040	0.0007218235	0.0406854481
1	0.0244958984	0.0036420661	0.0141363719
2	0.0062692177	0.0010638932	0.0054389963
5	0.0078114892	0.0001165337	0.0017446695
10	0.0160310402	0.0000440116	0.0007049249
15	0.0166686442	-	0.0004797220

Table 6: One edge shared: Time/Relative error

3.5.3 Tetrahedra sharing one vertex

N	MC	Med	Grid	QMC	Gauss
10	0.1197184863	0.7211609368	0.5371114771	0.3334888986	0.7501879952
50	0.3064578046	0.4164028777	0.2849726752	0.0922418581	0.0558888289
100	0.0909187085	0.2003124224	0.1133529464	0.0475938392	0.0262061191
500	0.0434842399	0.0361647134	0.0245930808	0.0049142097	0.0041004917
1000	0.0026719821	0.0536498740	0.0193777814	0.0019280281	0.0014513033
5000	0.0148705394	0.0916958334	0.0069645584	0.0001239842	-
10000	0.0033061279	0.0064090753	0.0041268376	0.0000378761	-

Table 7: One vertex shared: Number of directions/Relative errors

N	MCC	QMCC
10	0.6421534942	0.0407420136
50	0.0846158187	0.0593743052
100	0.0597215232	0.0965819923
500	0.1101481928	0.0641282729
1000	0.0129919454	0.0051843874
5000	0.0233731057	0.0052875782
10000	0.0166144408	0.0019393418

Table 8: One vertex shared: Number of 6-dimensional points/Relative error

sec.	QMCC	Gauss	QMC
0.1	0.0254018880	0.2214823012	0.2148558957
0.5	0.0641282729	0.0262061191	0.0475938392
1	0.0006547718	0.0174635800	0.0192403594
2	0.0013972022	0.0060861040	0.0080699829
5	0.0001124349	0.0016123472	0.0019280281
10	0.0039748868	0.0131755276	0.0006127605
15	0.0019393418	-	0.0003323616

Table 9: One vertex shared: Time/Relative error

3.6 Plots

sec:Plots

All the plots are drawn in log-log scale (base 10). We have a subsection for each of the three cases of tetrahedra sharing a face, an edge, a vertex. For each experiment we plot the relative error against the number of directions for the geometrical-field methods; the relative error against the number of $\mathbb{R}^6$-points for the Coulomb methods and the relative error against number of seconds for a comparison between Quasi Monte Carlo methods. Times for the Gaussian methods don't include the pre-processing time for the computation of nodes and weight.

3.6.1 Monte Carlo versus Monte Carlo with Median

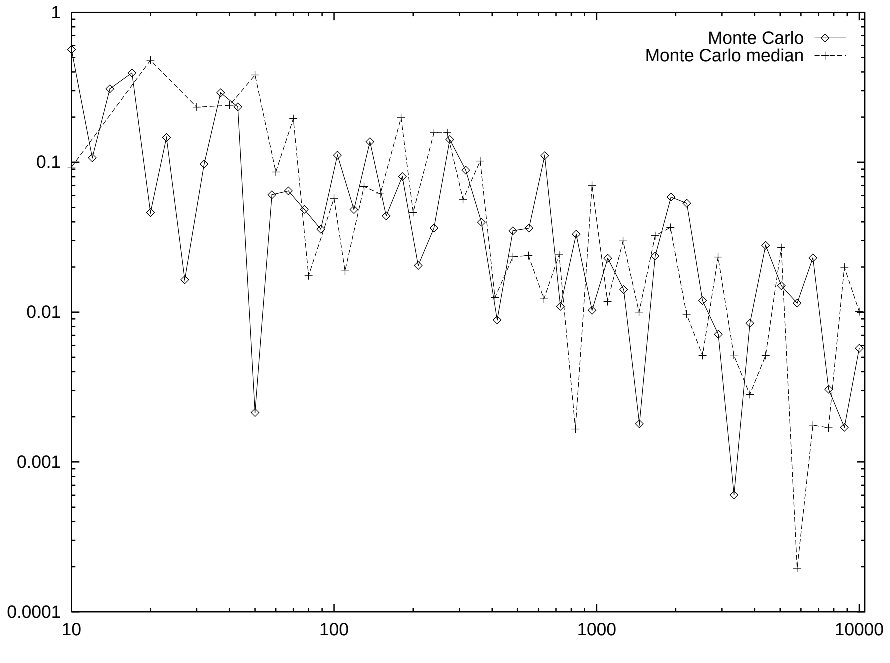

Figure 5: One face shared: Number of directions/relative error

figura5

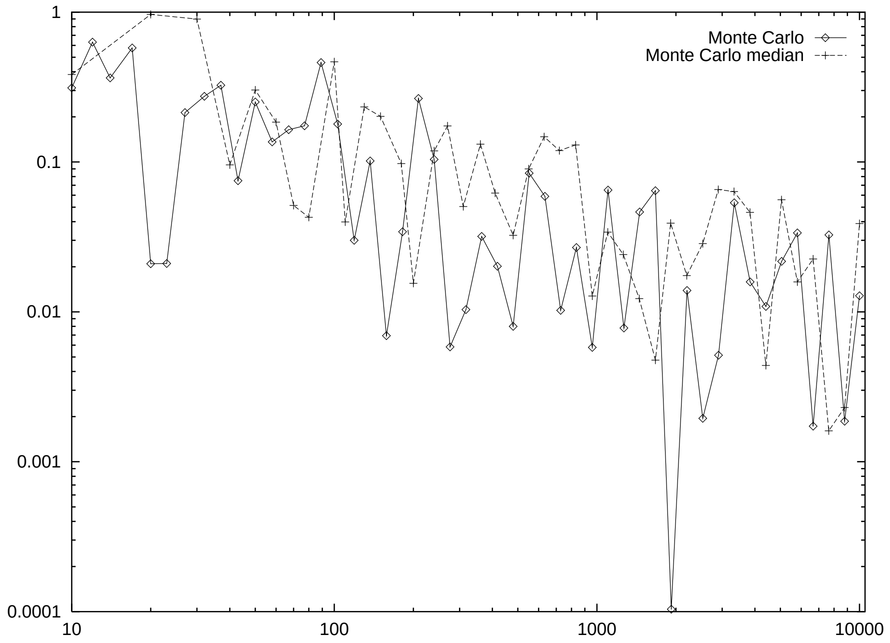

Figure 6: One edge shared: Number of directions/relative error

figura10

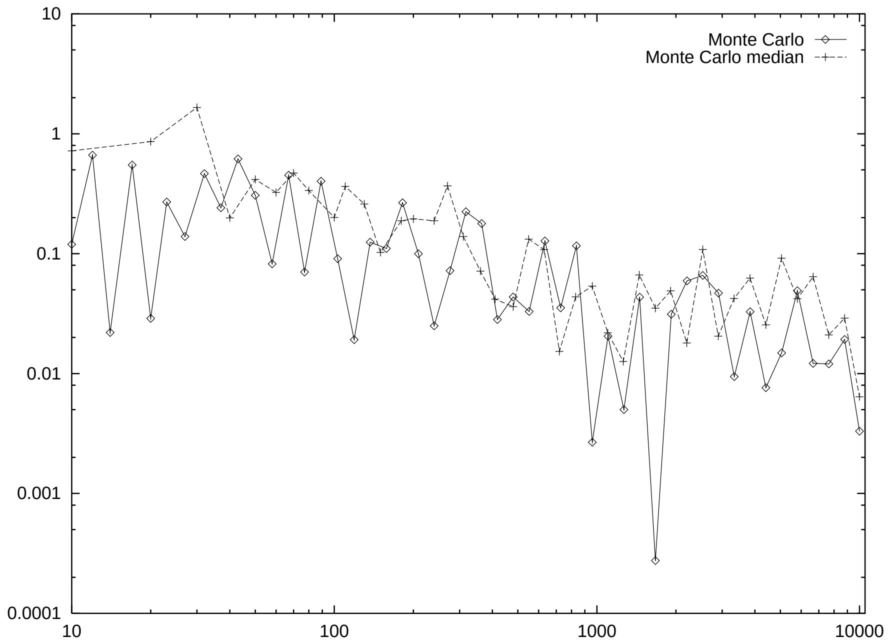

Figure 7: One vertex shared: Number of directions/relative error

figura15

3.6.2 Monte Carlo versus Quasi Monte Carlo

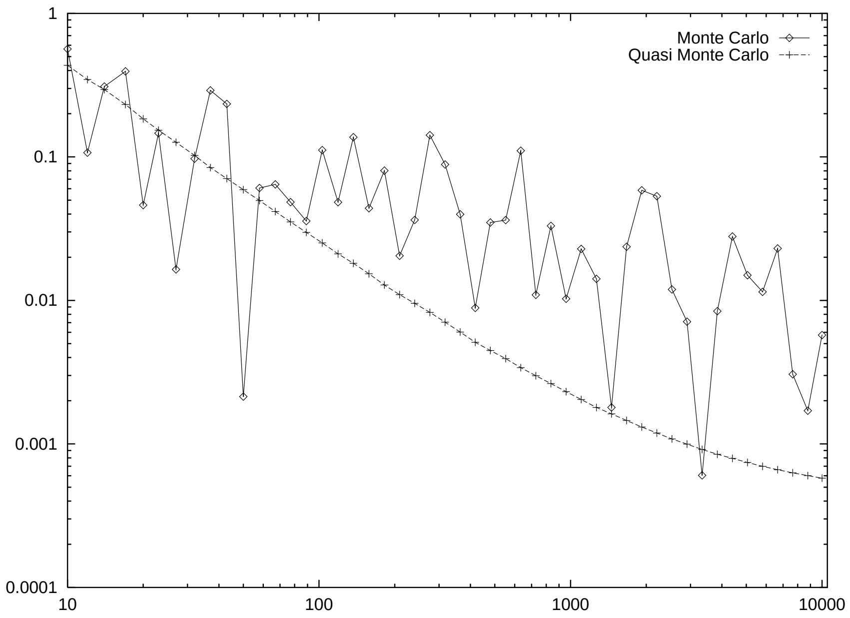

Figure 8: One face shared: Number of directions/relative error

figura6

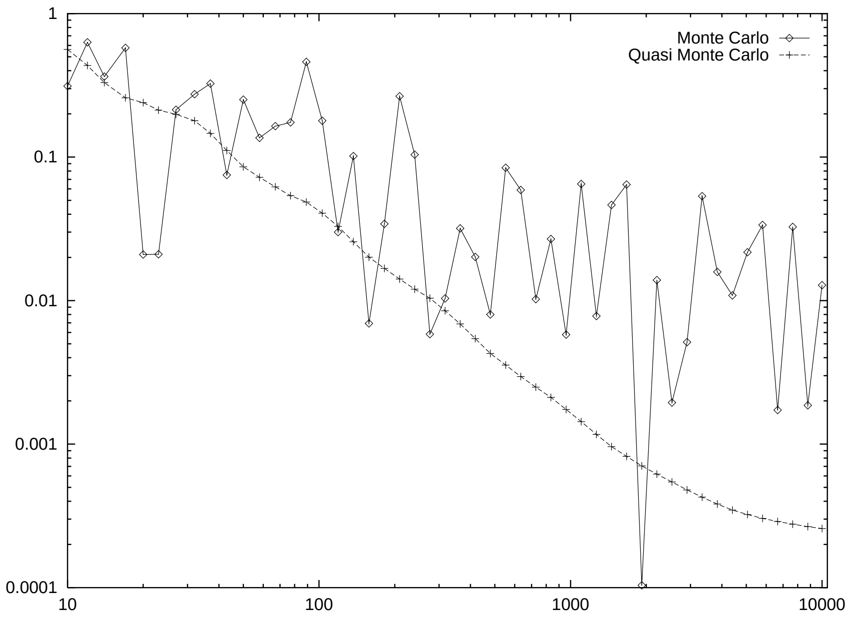

Figure 9: One edge shared:Number of directions/relative error

figura11

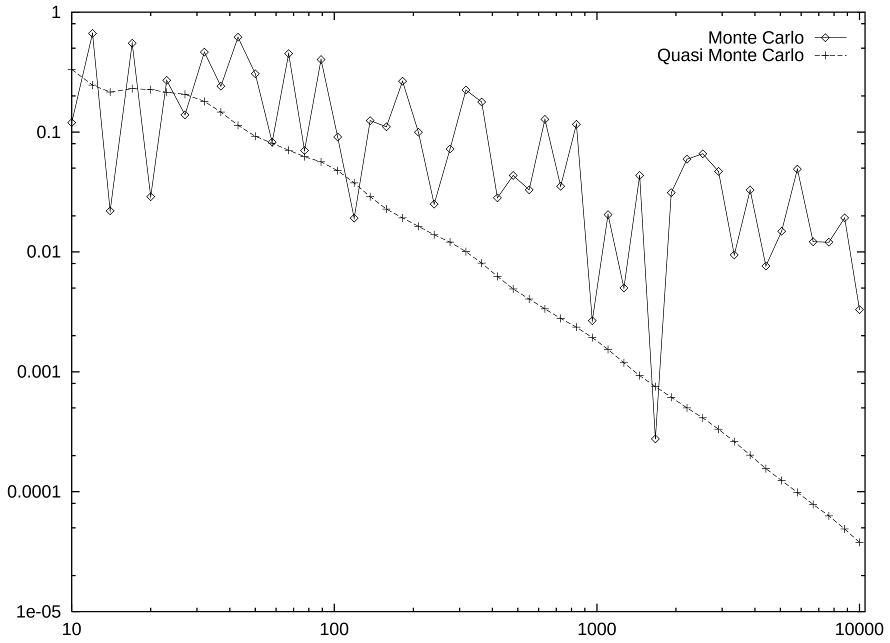

Figure 10: One vertex shared: Number of directions/relative error

figura16

3.6.3 All methods

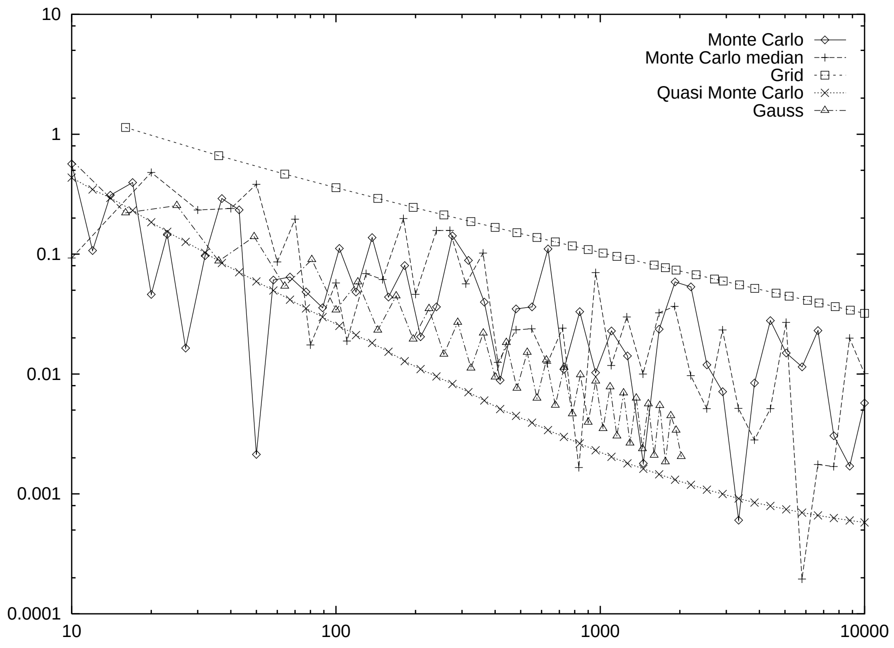

Figure 11: One face shared: Number of directions/relative error

figura7

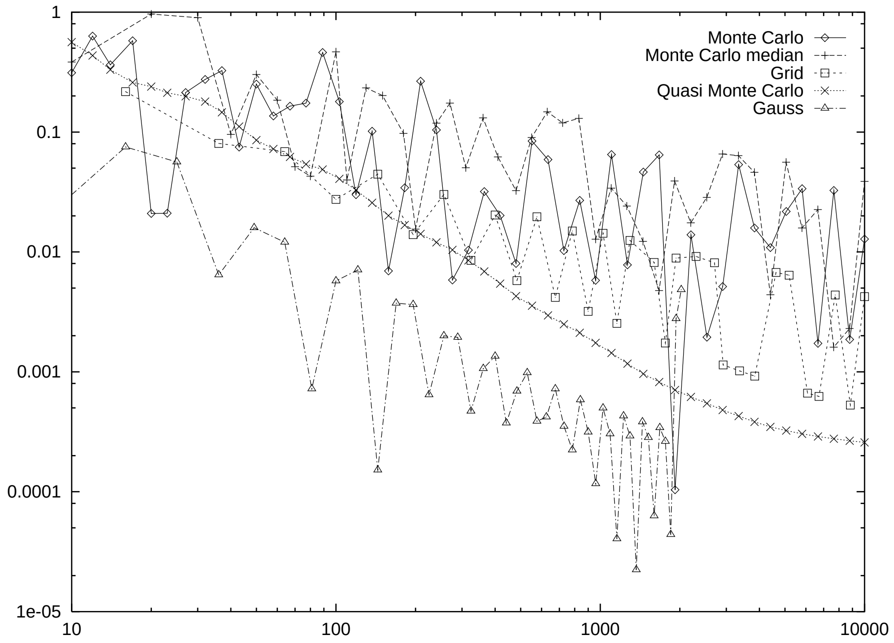

Figure 12: One edge shared: Number of directions/relative error figura12

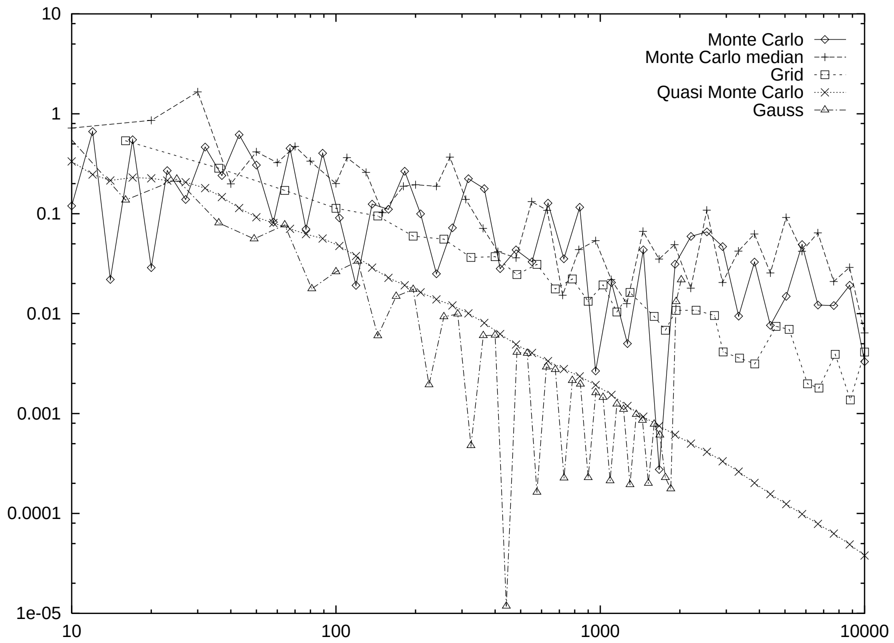

Figure 13: One vertex shared: Number of directions/relative error figura17

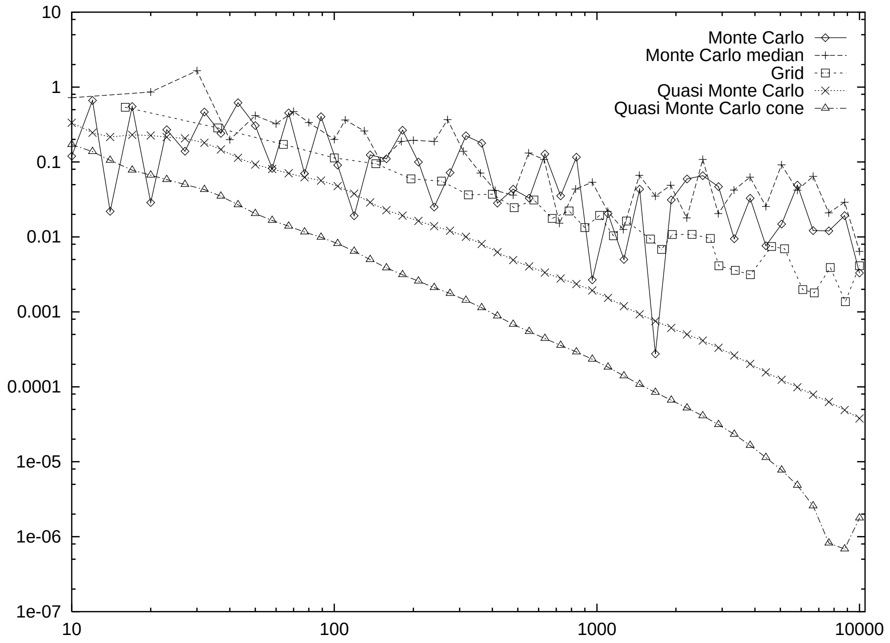

Figure 14: One vertex shared: Number of directions/relative error

figura18

3.6.4 Monte Carlo Coulomb versus Quasi Monte Carlo Coulomb

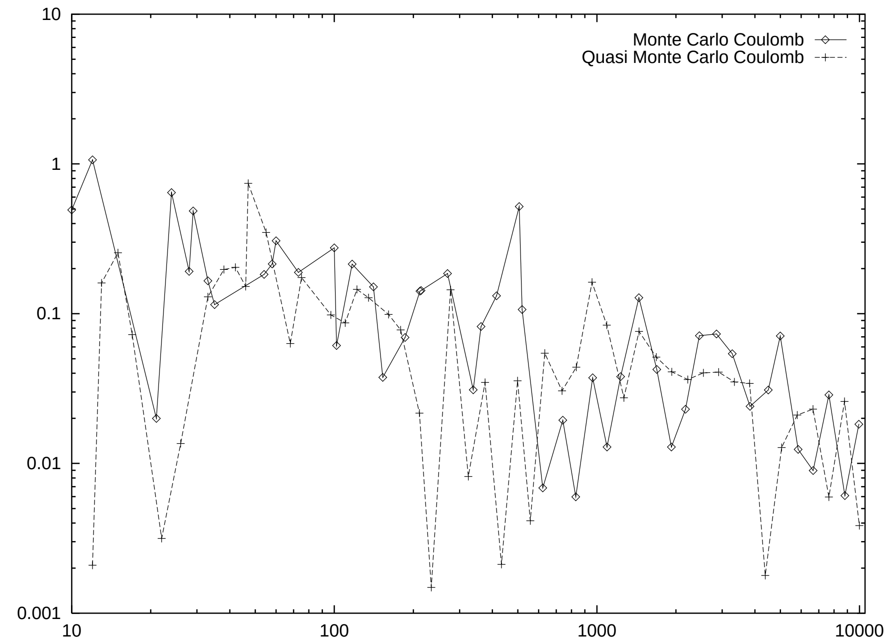

Figure 15: One face shared: Number of 6-dimensional points/relative error

figura8

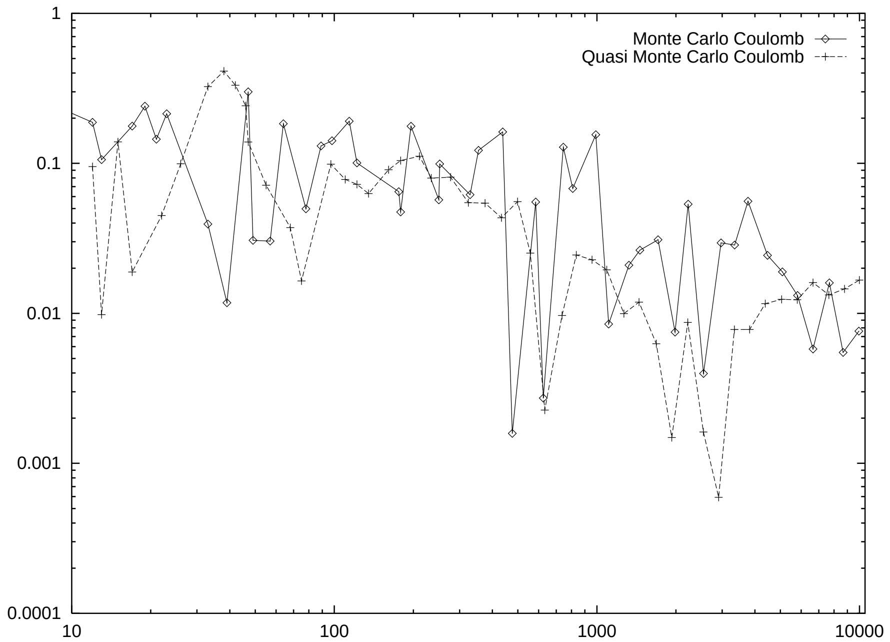

Figure 16: One edge shared: Number of 6-dimensional points/relative error `figura13`

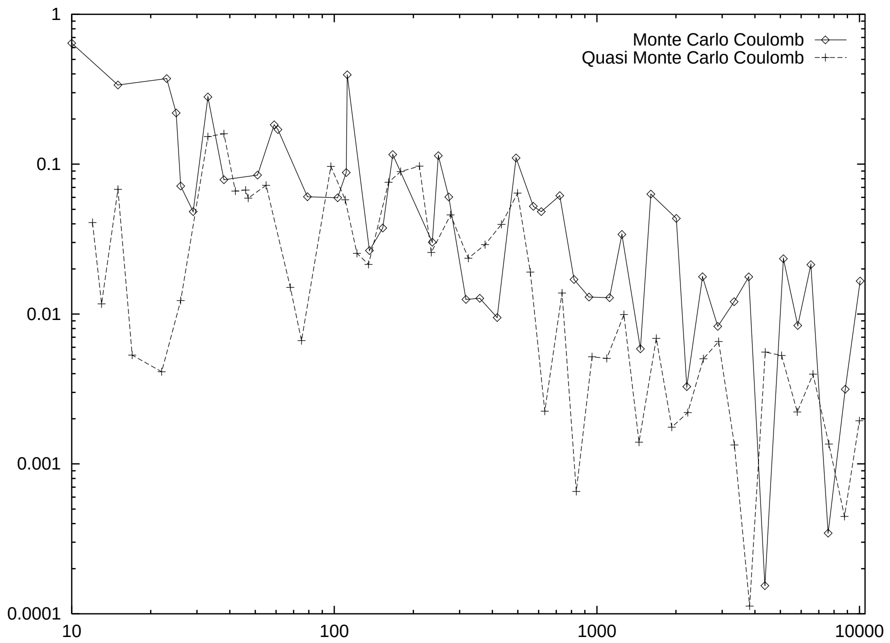

Figure 17: One vertex shared: Number of 6-dimensional points/relative error `figura19`

3.6.5 Running Times versus Relative Error

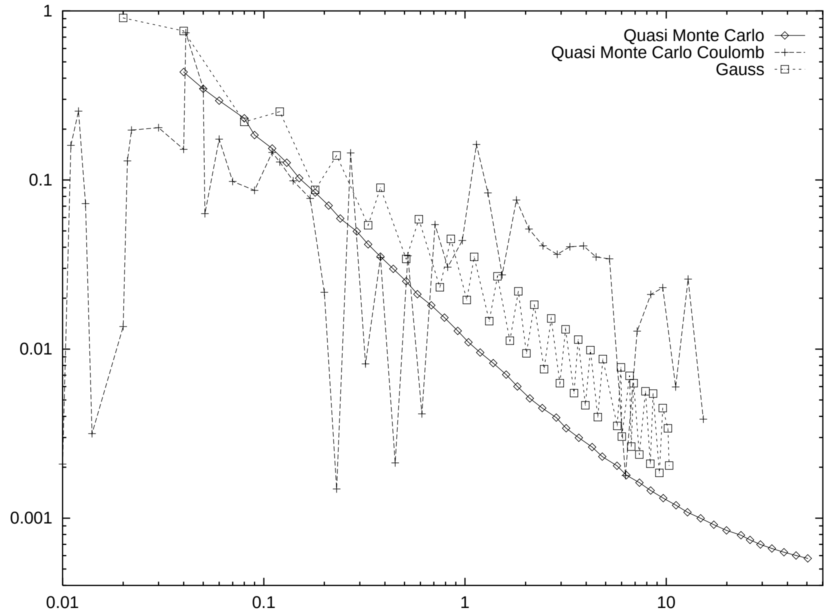

Figure 18: One face shared: Seconds/Relative error

figura9

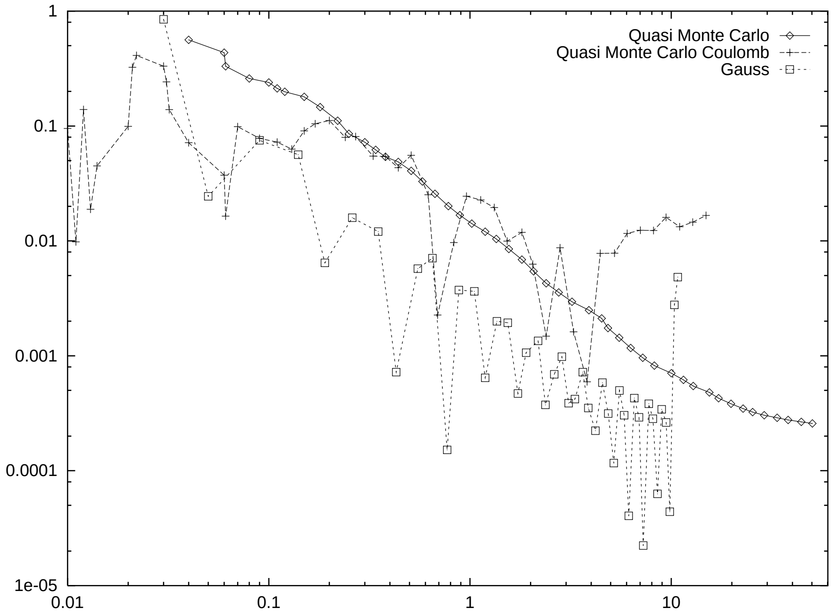

Figure 19: One edge shared: Seconds/Relative error

figura14

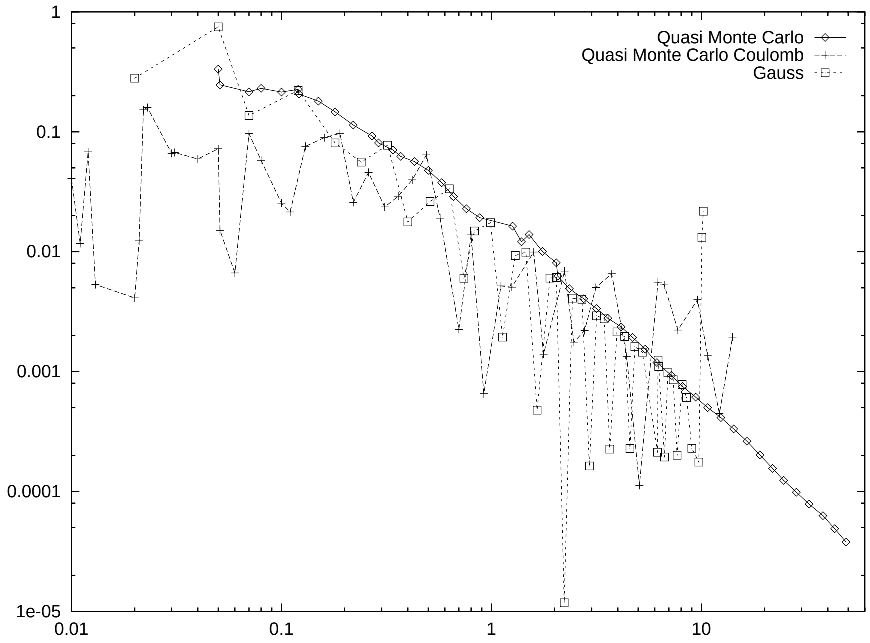

Figure 20: One vertex shared: Seconds/Relative error

figura20

3.7 Discussion of results

Figures 5, 6 and 7 show that there is little difference between results obtained with a straight Monte Carlo algorithm and a Monte Carlo with median. Figures 8 , 9 and 10 show that the curve obtained by a Quasi Monte Carlo approach is a much smoother than those obtained by the Monte Carlo approach, and it is consistently lower. Figures 11, 12, 13 show that of all the methods tested, the Quasi Monte Carlo approach gives the best reliable results over all the inputs. Figure 14 shows improvements due to a more refined sampling strategy. Figures 15 16 and 17 show that in the case of integrals based on Coulomb's law the Monte Carlo and Quasi Monte Carlo have a similar highly non-smooth behaviour. Figures 18, 19 and 20 show that the good performance of Quasi Monte Carlo based on the relative error as a function of the number of directions is confirmed when tabulating the relative error in function of the computing time.

Note that in the case of volume-to-volume integrals the Gaussian adaptive algorithm in [Pel97b] has not been considered in this round of tests. In the tests for surface-to-surface we have found that the reference value obtained by a long Monte Carlo process was rapidly matched by the Gaussian adaptive method for the first 4 digits. Afterwards The Gaussian adaptive method was converging smoothly towards a value different from the reference value. This behaviour can be explained by considering such reference value reliable only for the first 4 digits. In the surface-to-surface case we could overcome this obstacle by computing the reference value with a different process.

4 Experiments: Surface-to-surface integral

4.1 Implementation

The main steps of the algorithm are the same as the case of tetrahedra: the generation of the directions, the determination of the intersection-polygon P (if it exists), and the contribute computation. Regarding the generation of directions, we implemented the Quasi Monte Carlo and Gauss methods (no differences from the implementation for tetrahedra), also with the reduction of useless directions, and a special Gaussian adaptive method. The intersection-polygon is determined exactly in the same way as in the former algorithm.

The contribute computation can not be derived from (8) when $\cos\psi(u)$ is zero or almost zero, but we can set the range ϵ for this special case as thin as desired. We set $\epsilon = 10^{-6}$ and special cases don't influence the result.

4.2 Algorithms

4.2.1 Quasi Monte Carlo Arvo

In the first set of experiments it is possible to easily avoid all the directions that give null contribution to the computation of the electrostatic field. For this we generate sampling points on the ϕ-θ plane inside the square $[-\Pi/2, \Pi/2] \times [-\Pi/2, \Pi/2]$ rather than inside the rectangle $[0, 2\Pi] \times [0, \Pi/2]$. Thus the fact that a direction has (have?) null contribution depends only on the value of ϕ. Precisely, if α is the dihedral angle, all directions with $-\Pi/2 \leq \phi \leq -\Pi/2 + \alpha$ has null contribution to the computation of the electrostatic field, while all directions with $-\Pi/2 + \alpha \leq \phi \leq \Pi/2$ give effective contributions. The bigger is the value of α, the more convenient is to produce sampling points over the rectangle $[-\Pi/2 + \alpha, \Pi/2] \times [-\Pi/2, \Pi/2]$.

4.2.2 Gauss Arvo

As described in the former section, the algorithm consists in generating Gaussian points over a reduced rectangular zone of the ϕ-θ plane, so that all the null contribution directions are avoided.

4.2.3 Gauss adaptive

This algorithm is described in detail in [Pel97b]. The basic idea is to locate the zones of the ϕ-θ plane in which the names of the vertices of the intersection polygon P are constant. For instance, in the case of two triangles ABC and ABD sharing the edge AB, the ϕ-θ plane is divided in four zones, according to four possible kinds of intersection polygon (fig. 21). It is possible to explicitly write the expressions for the curves that delimit the zones and for the area of P as functions of ϕ and θ. The Gauss adaptive method consists in the application of bidimensional Gaussian formulas to each zone.

4.3 Triangles sharing one edge: Inputs

We made two sets of experiments: $T_1 = \{\{0,0,-1\},\{0,0,1\},\{0,-1,0\}\}$, $T_2 = \{\{0,0,-1\},\{0,0,1\},\{\sin A, -\cos A, 0\}\}$ (fig. 22), the angle A assuming the values

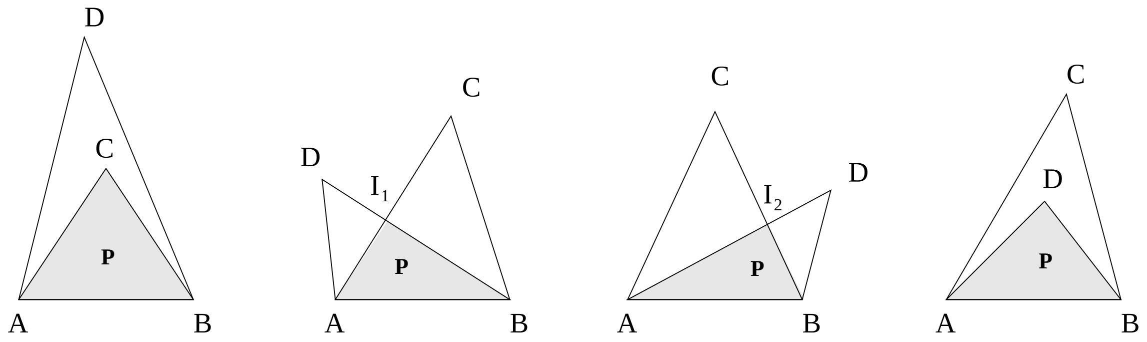

Figure 21: Triangles sharing one edge: there are four kinds of intersection polygon P fig:zones

$\frac{3}{4}\Pi, \Pi/2, \Pi/4, \Pi/8, \Pi/16, \Pi/32$, and $T_1 = \{\{0,0,-1\},\{0,0,1\},\{0,-1,-1\}\}$, $T_2 = \{\{0,0,-1\},\{0,0,1\},\{-\sin B, 0, \cos B\}\}$ (fig. 23), with the angle B that is $\frac{i}{6}\Pi$ for $i = 1 \ldots 5$.

The first set of experiments is intended to study the dependency of the algorithms on the dihedral angle between the input triangles. The second set of experiments is intended to study the dependency on the aspect ratio of one of the two triangles. We supposed the functions σ_1 and σ_2 to be constant and equal to 1. We compared Quasi Monte Carlo and Gaussian methods. The approximations of C are obtained using 10000 directions for the former methods and with product Gaussian formulas for the latter ones, stopping computations when round-off errors arose.

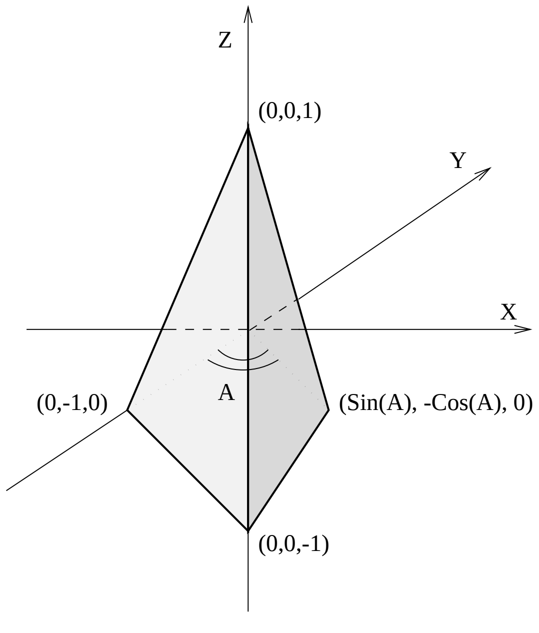

Figure 22: First set of experiments fig:tri1

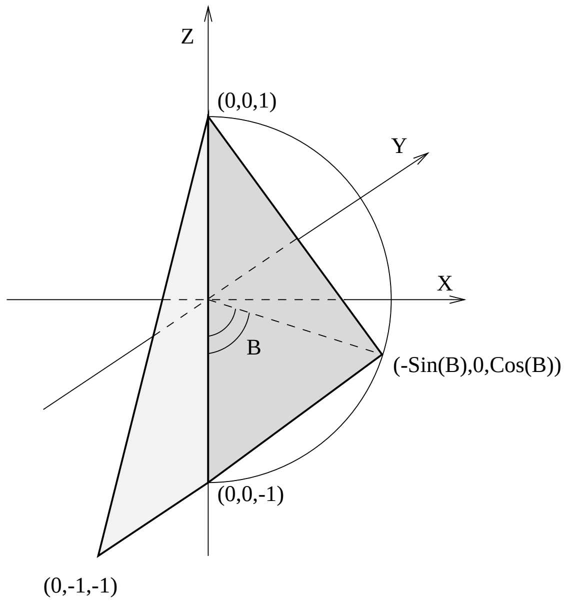

Figure 23: Second set of experiments fig:tri2

4.3.1 Computation of the reference value

In order to obtain reference values, after having divided the ϕ-θ plane, as for the Gauss Adaptive method, in zones so that the names of the vertices of the intersection polygon P are constant, we explicitly wrote the formulas for the area of P as function of ϕ and θ. Here are the expressions in the case $A = \Pi/2$:

- zone1 $= \cos\theta$
- zone2 $= \cos(\theta)\ \cot(\phi)$
- zone3 $= \dfrac{2\ \cos(\phi)\cos(\theta)^2}{\cos(\phi)\ (\cos(\theta) - \cos(\phi)\ \sin(\theta)) - \sin(\phi)\ (-\cos(\theta) + \sin(\phi)\ \sin(\theta))}$
- zone4 $= \dfrac{2\ \cos(\phi)\cos(\theta)^2}{-(\cos(\phi)\ (-\cos(\theta) - \cos(\phi)\ \sin(\theta))) + \sin(\phi)\ (\cos(\theta) + \sin(\phi)\ \sin(\theta))}$

Then we symbolically integrated these functions on the θ variable (this step requires the subdivision of zones 3 and 4 in 2 subzones), and finally we numerically integrated on the ϕ variable by the internal Mathematica routine. This guarantees the achievement of at least sixteen digits of precision. It must be stressed that the first step of symbolical integration has been possible when some constraints are used to generate input triangles, but not in general. Here we show the results of the symbolical integration step:

- $SymbF1(\phi)$= $\dfrac{2\,(\cos(\phi) - \sin(\phi))}{\sqrt{2 - \sin(2\,\phi)}}$
- $SymbF2(\phi)$= $\dfrac{-2\ \cot(\phi)\ (\cos(\phi) - \sin(\phi))}{\sqrt{2 - \sin(2\,\phi)}}$
- $SymbF3\mathrm{a}(\phi)$=

$$\frac{-4\,i\ \arctan\left(\frac{-i\,\sqrt{2}+(-1)^{\frac{1}{4}}\ \cos(\phi)+(-1)^{\frac{3}{4}}\ \cos(\phi)+(-1)^{\frac{1}{4}}\ \sin(\phi)+(-1)^{\frac{3}{4}}\ \sin(\phi)}{\sqrt{2}\,\sqrt{-i+\cos(\phi)+\sin(\phi)}\,\sqrt{i+\cos(\phi)+\sin(\phi)}}\right)\ \cos(\phi)}{(-i + \cos(\phi) + \sin(\phi))^{\frac{3}{2}}\ (i + \cos(\phi) + \sin(\phi))^{\frac{3}{2}}}$$

$$+\frac{4\,i\ \arctan\left(\frac{\sec K1\,(i\ \cos(\phi-K1)-i\ \cos(\phi+K1)-2\,i\ \cos K1-i\ \sin(\phi-K1)+i\ \sin(\phi+K1))}{2\,\sqrt{-i+\cos(\phi)+\sin(\phi)}\,\sqrt{i+\cos(\phi)+\sin(\phi)}}\right)\ \cos(\phi)}{(-i + \cos(\phi) + \sin(\phi))^{\frac{3}{2}}\ (i + \cos(\phi) + \sin(\phi))^{\frac{3}{2}}}$$

$$+\frac{1 + \cos(2\,\phi) + \sin(2\,\phi)}{2 + \sin(2\,\phi)}$$

$$+\frac{-\cos(\phi - 2\,K1) - 0.5\ \cos(2\,\phi - 2\,K1) - \cos(\phi + 2\,K1) + 0.5\ \cos(2\,\phi + 2\,K1)}{2 + \sin(2\,\phi)}$$

$$+\frac{-\frac{(\cos(\phi)-\sin(\phi))}{\sqrt{2-\sin(2\,\phi)}} + 0.5\ \sin(2\,\phi - 2\,K1) - 0.5\ \sin(2\,\phi + 2\,K1)}{2 + \sin(2\,\phi)},$$

where $K1 = (\frac{\arctan(\cos(\phi)-\sin(\phi))}{2})$.

- $SymbF3\mathrm{b}(\phi)$=

$$\frac{-4\,i\,\arctan\left(\frac{-i\,\sqrt{2}+(-1)^{\frac{1}{4}}\,\cos(\phi)+(-1)^{\frac{3}{4}}\,\cos(\phi)+(-1)^{\frac{1}{4}}\,\sin(\phi)+(-1)^{\frac{3}{4}}\,\sin(\phi)}{\sqrt{2}\,\sqrt{-i+\cos(\phi)+\sin(\phi)}\,\sqrt{i+\cos(\phi)+\sin(\phi)}}\right)\,\cos(\phi)}{(-i+\cos(\phi)+\sin(\phi))^{\frac{3}{2}}\,(i+\cos(\phi)+\sin(\phi))^{\frac{3}{2}}}$$

$$+\frac{4\,i\,\arctan(\frac{\sec K1\,(-i\,\cos(\phi-K1)+i\,\cos(\phi+K1)-2\,i\,\cos K1+i\,\sin(\phi-K1)-i\,\sin(\phi+K1))}{2\,\sqrt{-i+\cos(\phi)+\sin(\phi)}\,\sqrt{i+\cos(\phi)+\sin(\phi)}})\,\cos(\phi)}{(-i+\cos(\phi)+\sin(\phi))^{\frac{3}{2}}\,(i+\cos(\phi)+\sin(\phi))^{\frac{3}{2}}}$$

$$+\frac{1+\cos(2\,\phi)+\sin(2\,\phi)}{2+\sin(2\,\phi)}$$

$$+\frac{-\cos(\phi-2\,K1)+0.5\,\cos(2\,\phi-2\,K1)-\cos(\phi+2\,K1)-0.5\,\cos(2\,\phi+2\,K1)}{2+\sin(2\,\phi)}$$

$$+\frac{\frac{(\cos(\phi)-\sin(\phi))}{\sqrt{2-\sin(2\,\phi)}}-0.5\,\sin(2\,\phi-2\,K1)+0.5\,\sin(2\,\phi+2\,K1)}{2+\sin(2\,\phi)},$$

where $K1=(\frac{\arctan(\cos(\phi)-\sin(\phi))}{2})$.

Functions $SymbF4\mathrm{a}(\phi)$ and $SymbF4\mathrm{b}(\phi)$ are similar to $SymbF3\mathrm{a}(\phi)$ and $SymbF3\mathrm{b}(\phi)$.

4.4 Triangles sharing one edge: Tables

N	QMC	QMC Arvo	Gauss	Gauss Arvo	Gauss adaptive
10	0.0061644837	0.0022704817	0.0210938350	0.0099399448	-
50	0.0098639752	0.0059076679	0.0077730135	0.0013574675	0.004827410731152
100	0.0027106227	0.0019818569	0.0035424189	0.0004783674	0.001741267285381
500	0.0014004229	0.0011901843	0.0000789556	0.0000617531	0.000008450932804
1000	0.0003817477	0.0003598230	0.0004450780	0.0000232381	0.000000029977179
5000	0.0001105339	0.0001038090	-	-	-
10000	0.0000595452	0.0000577228	-	-	-

Table 10: First set of experiments, $A = \frac{\Pi}{32}$, Directions/Relative Error

Sec	QMC Arvo	Gauss Arvo	Gauss Adaptive
0.01	0.0067820330	0.0004680806	0.012406136272012
0.05	0.0019818569	0.0003015102	0.001741267285381
0.1	0.0008050237	0.0001302307	0.000217525382339
0.5	0.0004879939	0.0000181147	0.000000008928422
1	0.0001912208	-	0.000000001578904
2	0.0001448100	-	0.000000000038610
5	0.0000577228	-	-

Table 11: First set of experiment, $A = \frac{\Pi}{32}$, Seconds/Relative Error

N	QMC	Gauss	Gauss Adaptive
10	0.0061422310	0.0034567108	-
50	0.0149990514	0.0080967296	0.001910107156030
100	0.0038533881	0.0043477814	0.000401146587702
500	0.0016252574	0.0004539652	0.000000468939641
1000	0.0003893272	0.0004082786	0.000000012224489
2000	0.0002936440	0.0004224322	0.000000000032109
5000	0.0001228376	-	-
10000	0.0000672983	-	-

Table 12: First set of experiments, $A = \frac{\Pi}{16}$, Directions/Relative Error

N	QMC	Gauss	Gauss Adaptive
10	0.0306591377	0.0691388584	-
50	0.0185114963	0.0129659379	0.000006422735849
100	0.0041568603	0.0036278347	0.000056420242361
500	0.0019162651	0.0011201276	0.000000080335962
1000	0.0005235731	0.0001720985	0.000000000258235
2000	0.0003600855	0.0002232347	0.000000000003420
5000	0.0001442366	-	-
10000	0.0000753721	-	-

Table 13: First set of experiments, $A = \frac{\Pi}{8}$, Directions/Relative Error

N	QMC	QMC Arvo	Gauss	Gauss Arvo	Gauss Adaptive
10	0.0535728383	0.0128789871	0.0126393418	0.0214120734	-
50	0.0205920701	0.0191165477	0.0039487063	0.0051507019	0.000377148381460
100	0.0048356470	0.0037153534	0.0030789788	0.0015784829	0.000051574350422
500	0.0026548897	0.0020125312	0.0004981855	0.0001036810	0.000000002691458
1000	0.0007270418	0.0005830705	0.0008049655	0.0000274621	0.000000000000166
2000	0.0004190069	0.0003220956	0.0003039623	0.0000064155	0.000000000000998
5000	0.0001937239	0.0001419660	-	-	-
10000	0.0001018837	0.0000809973	-	-	-

Table 14: First set of experiments, $A = \frac{\Pi}{4}$, Directions/Relative Error

N	QMC	QMC Arvo	Gauss	Gauss Arvo	Gauss Adaptive
10	0.0655086154	0.0297884032	0.2165219575	0.0664795933	-
50	0.0512291174	0.0237291482	0.0387407106	0.0093797082	0.000208637303841
100	0.0083601961	0.0045762836	0.0103432812	0.0004240828	0.000015489387908
500	0.0047339359	0.0024859659	0.0022276241	0.0000037783	0.000000000060139
1000	0.0011279527	0.0006169545	0.0007542757	0.0000792836	0.000000000000015
5000	0.0003203950	0.0001619560	-	-	-
10000	0.0001561082	0.0000752719	-	-	-

Table 15: First set of experiments, $A = \frac{\Pi}{2}$, Directions/Relative Error

Sec	QMC	QMC Arvo	Gauss	Gauss Arvo	Gauss Adaptive
0.01	0.0420138102	0.0170879244	0.0304167504	0.0130667482	0.002283139023758
0.05	0.0156178834	0.0082368579	0.0109412861	0.0007172048	0.000015489387908
0.1	0.0068845075	0.0039086681	0.0041953194	0.0003179505	0.000000102017690
0.5	0.0017144039	0.0009041885	0.0007648762	0.0000192332	0.000000000000011
1	0.0006168770	0.0003813926	0.0012118725	0.0003657898	-
2	0.0004834223	0.0002361355	-	-	-
4.5	0.0001561082	0.0000752719	-	-	-

Table 16: First set of experiments, $A = \frac{\Pi}{2}$, Seconds/Relative Error

N	QMC	QMC Arvo	Gauss	Gauss Arvo	Gauss Adaptive
10	0.5280948993	0.0397382223	0.5693915997	0.0973367496	-
50	0.1012786276	0.0254972998	0.0560743988	0.0126320763	0.000151416854249
100	0.0104194970	0.0048628793	0.0052809078	0.0004245439	0.000009612858490
500	0.0081320945	0.0027506408	0.0003899622	0.0002324747	0.000000000012638
1000	0.0024061698	0.0006579719	0.0019503726	0.0000793079	0.000000000000001
5000	0.0006788748	0.0001740919	-	-	-
10000	0.0004166019	0.0000764521	-	-	-

Table 17: First set of experiments, $A = \frac{3}{4}\Pi$, Directions/Relative Error

Sec	QMC Arvo	Gauss Arvo	Gauss Adaptive
0.01	0.0188796748	0.0191949988	0.001894388562174
0.05	0.0048628793	0.0014686414	0.000009612858490
0.1	0.0041920162	0.0005585416	0.000000041795875
0.5	0.0009415366	0.0000414397	0.000000000000001
1	0.0003889558	0.0002812471	-
2	0.0002390120	-	-
5	0.0000764521	-	-

Table 18: First set of experiments, $A = \frac{3}{4}\Pi$, Seconds/Relative Error

4.5 Triangles sharing one edge: Plots

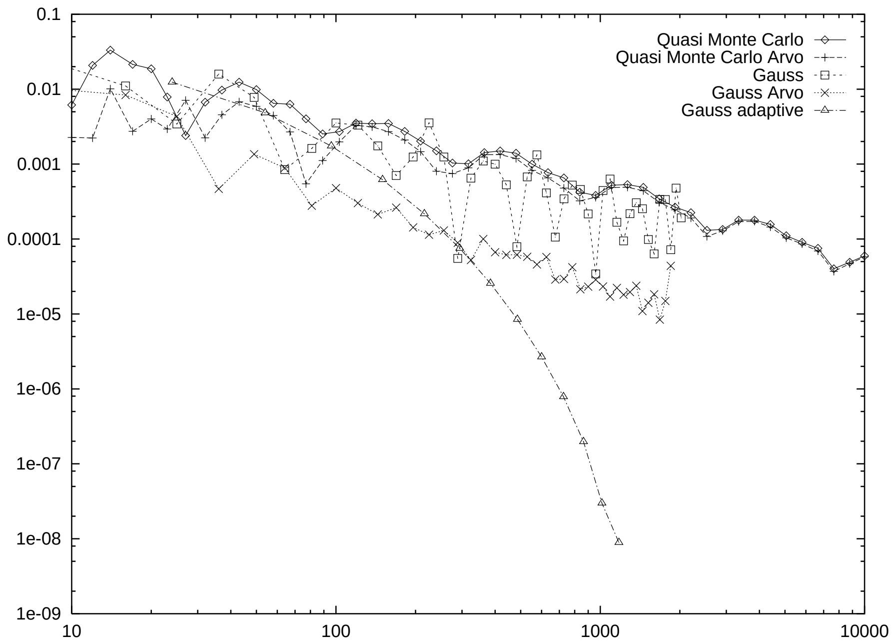

Figure 24: First set of experiments, $A = \frac{\Pi}{32}$; Directions/relative error

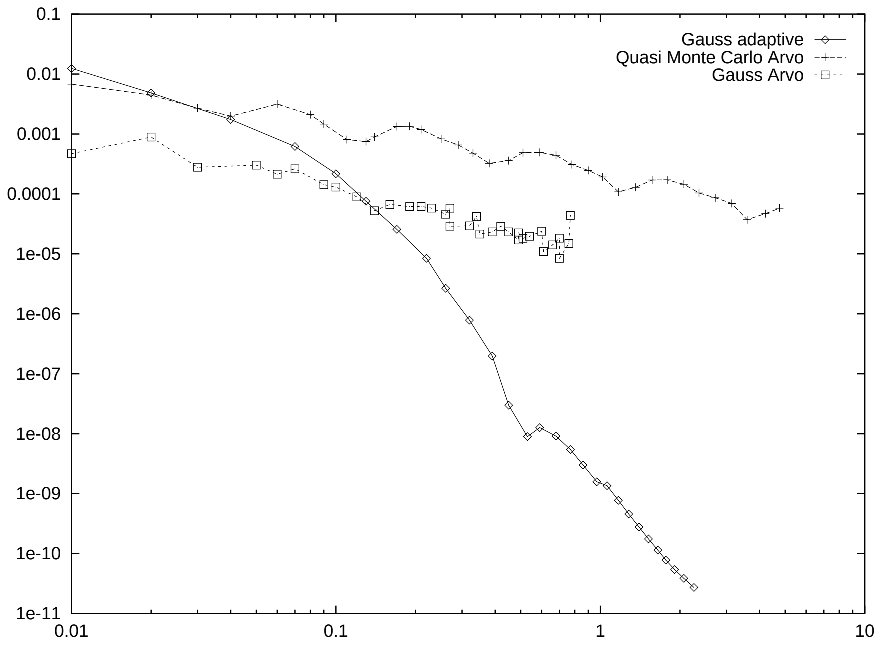

Figure 25: First set of experiments, $A = \frac{\Pi}{32}$; Seconds/relative error

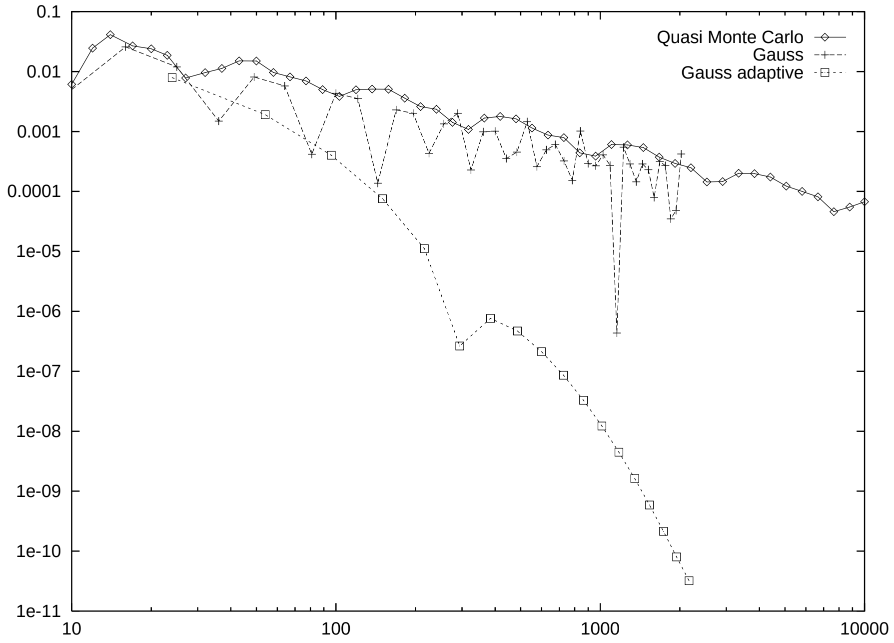

Figure 26: First set of experiments, $A = \frac{\Pi}{16}$; Directions/relative error

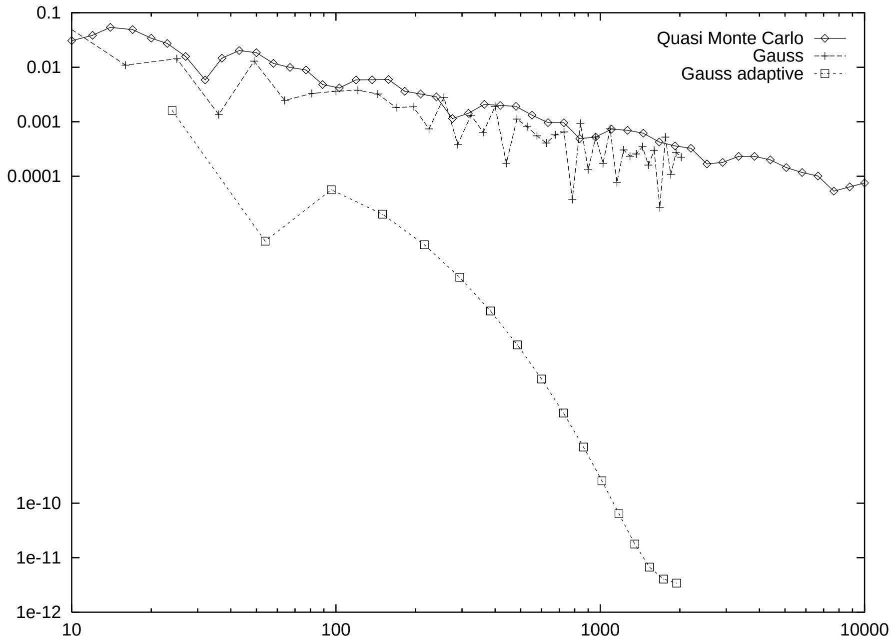

Figure 27: First set of experiments, $A = \frac{\Pi}{8}$; Directions/relative error

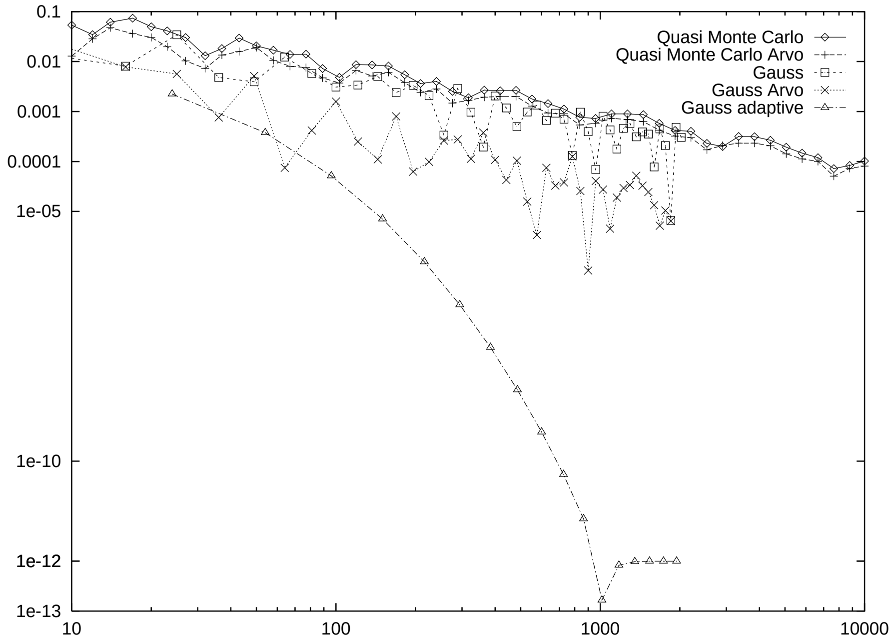

Figure 28: First set of experiments, $A = \frac{\Pi}{4}$; Directions/relative error

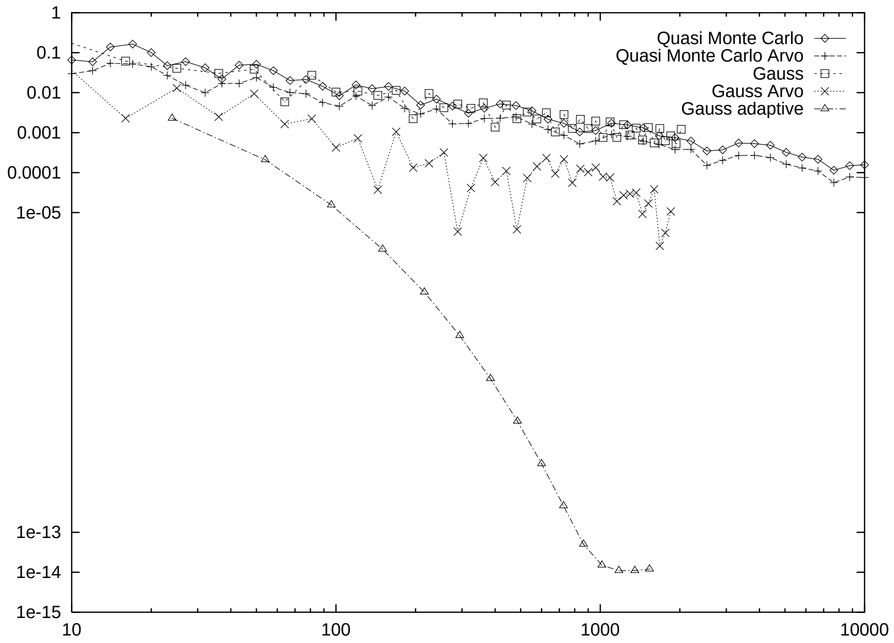

Figure 29: First set of experiments, $A = \frac{\Pi}{2}$; Directions/relative error

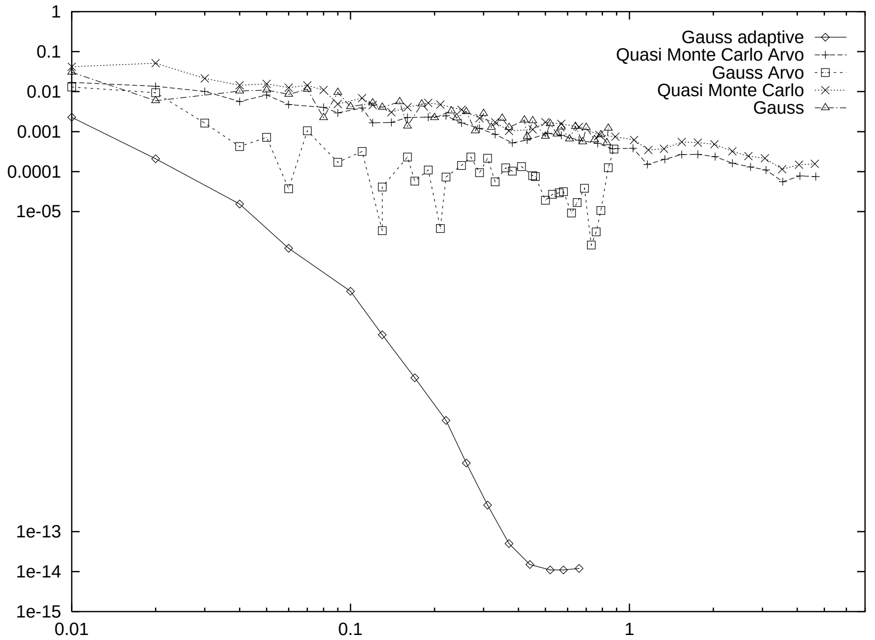

Figure 30: First set of experiments, $A = \frac{\Pi}{2}$; Seconds/relative error

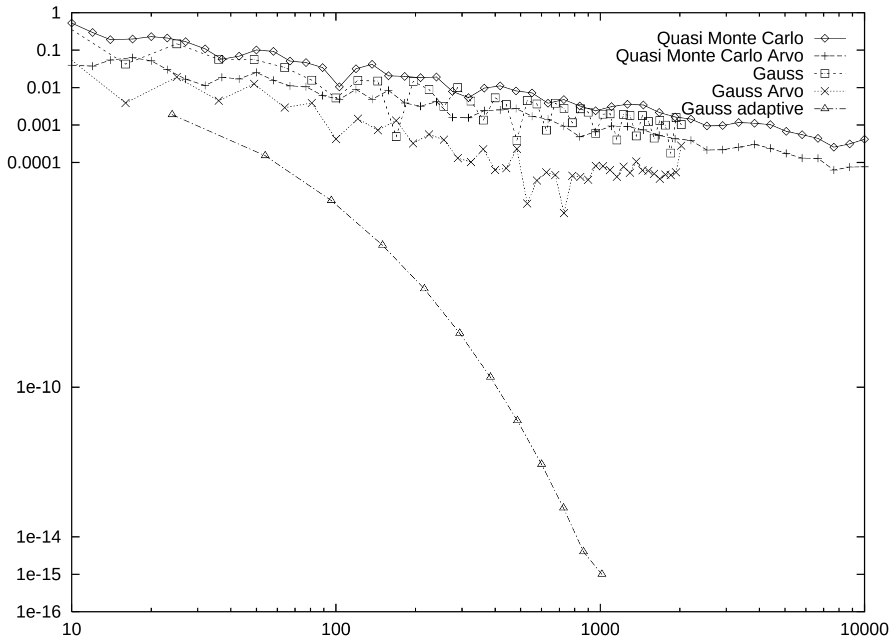

Figure 31: First set of experiments, $A = \frac{3}{4}\Pi$; Directions/relative error

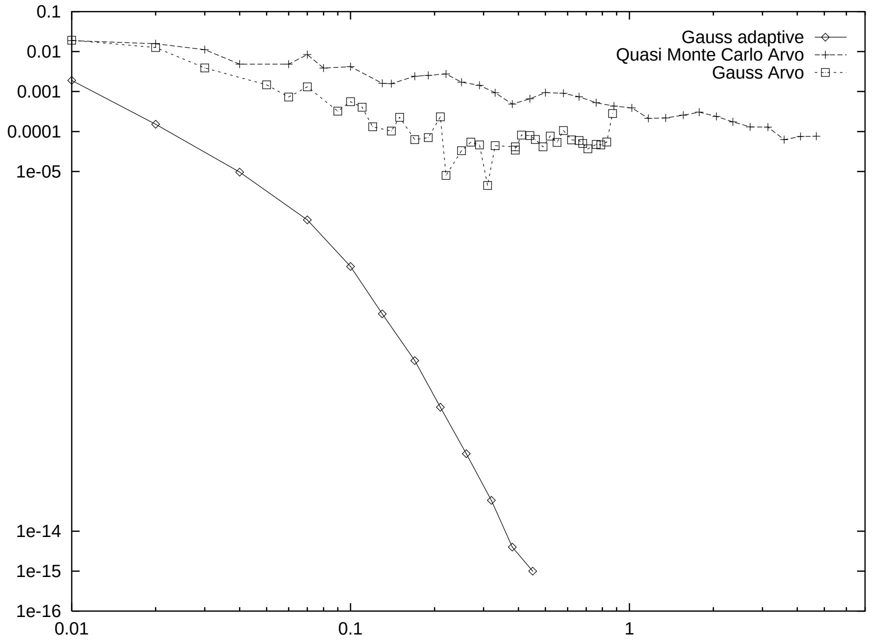

Figure 32: First set of experiments, $A = \frac{3}{4}\Pi$; Seconds/relative error

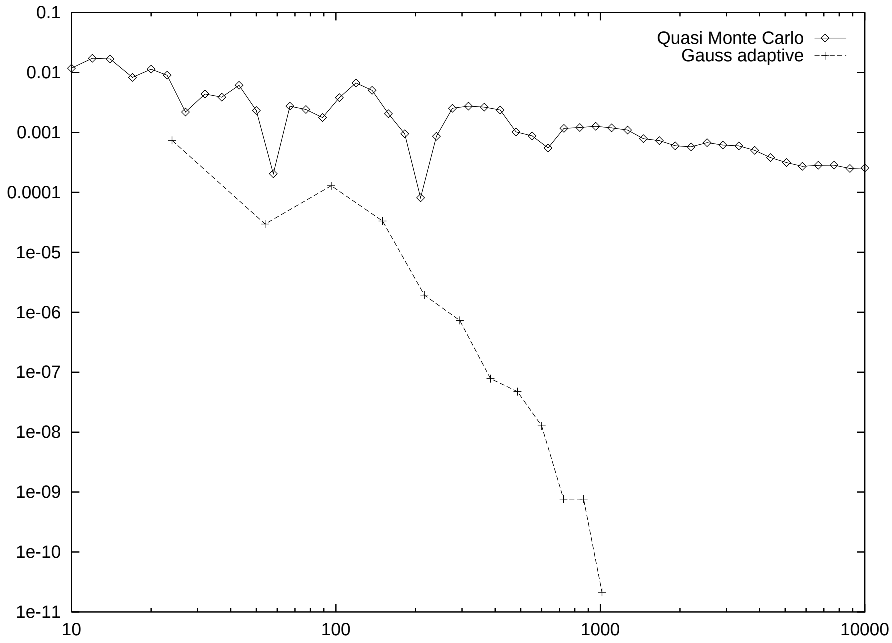

Figure 33: Second set of experiments, $B = \frac{1}{6}\Pi$; Directions/relative error

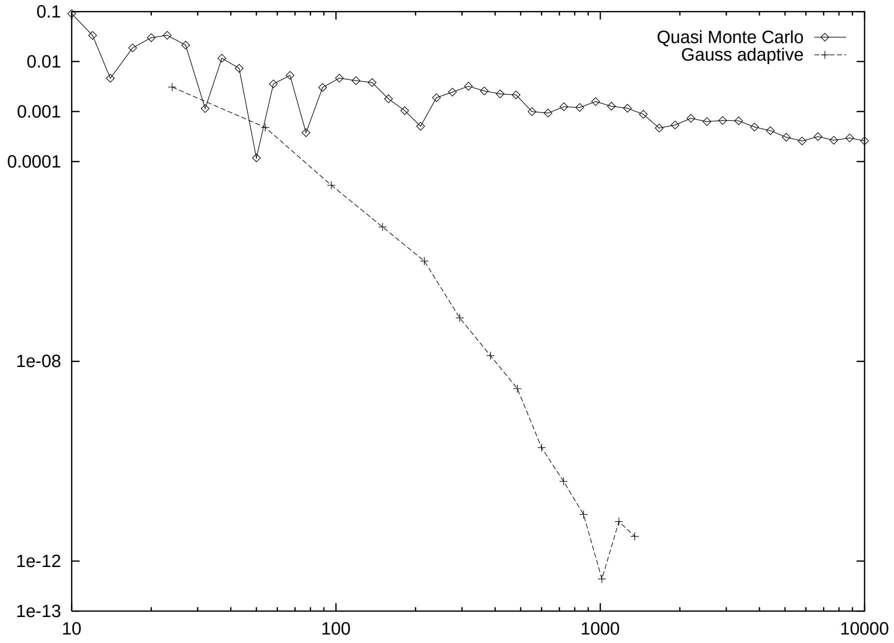

Figure 34: Second set of experiments, $B = \frac{2}{6}\Pi$; Directions/relative error

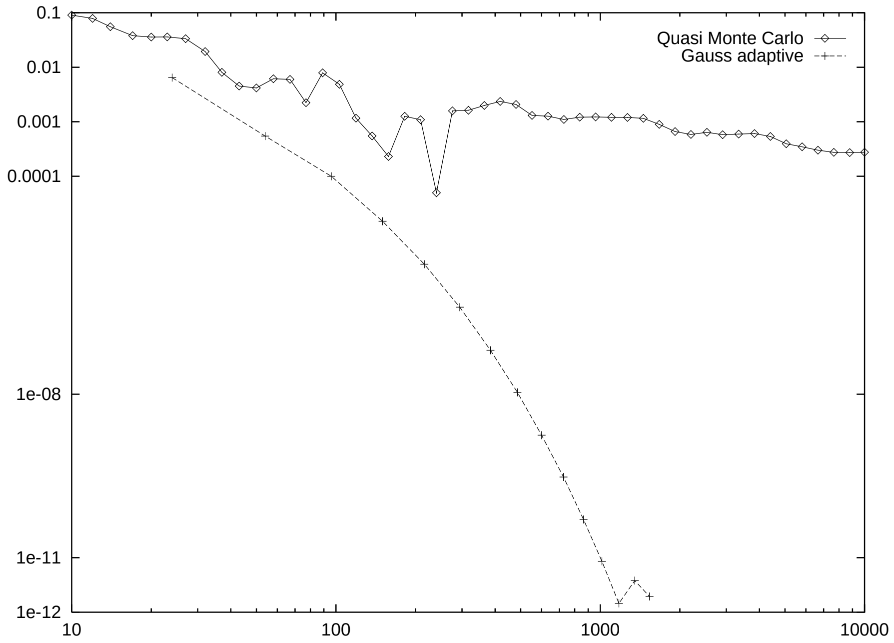

Figure 35: Second set of experiments, $B = \frac{3}{6}\Pi$; Directions/relative error

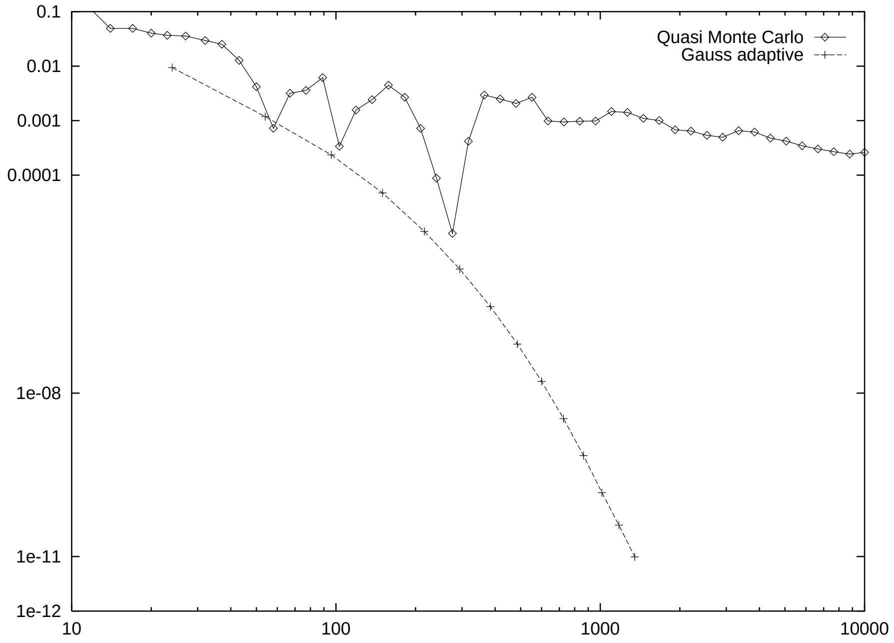

Figure 36: Second set of experiments, $B = \frac{4}{6}\Pi$; Directions/relative error

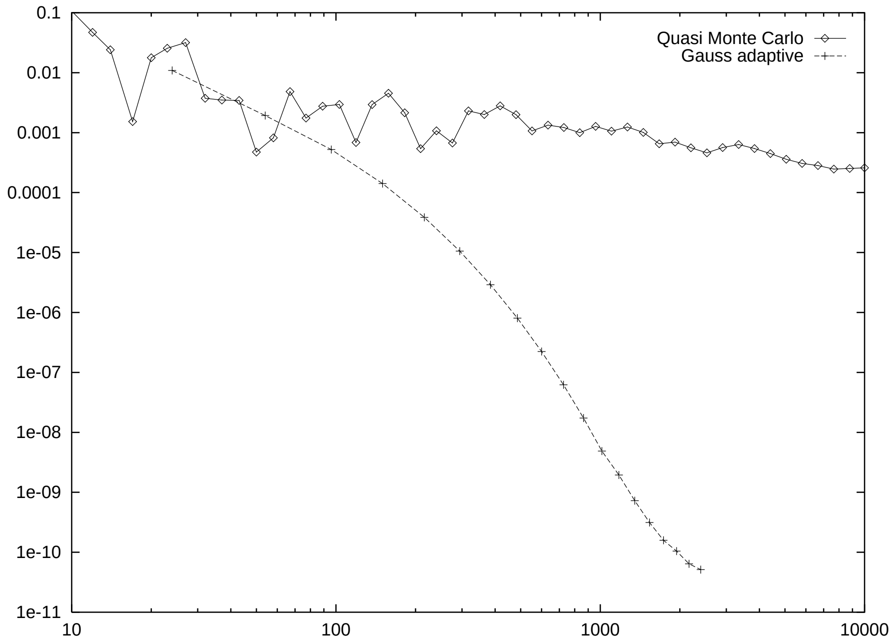

Figure 37: Second set of experiments, $B = \frac{5}{6}\Pi$; Directions/relative error

4.6 Discussion of results

Figures relative to experiment A show that all methods tested, except the Gaussian adaptive, in the range of directions $[10, 10000]$ achieve at best error 10^{-4}. The Gauss adaptive method already for 1000 directions attains errors between 10^{-8} and 10^{-14}. The Gauss adaptive approach produces a rather smooth plot in the range $[10, 1000]$, afterwards it suffer from numerical instabilities probably due to round-off. The lower end of the error range is attained for small dihedral angles ($A = \pi/32$), while the high end is attained for angles close to $\pi/2$.

Experiment B shows errors in the range between 10^{-10} and 10^{-11} for the angles B between $\pi/6$ and $5\pi/6$. Dependency of the error on the angles A and B is weak in the middle of the range, with slightly worse performance towards the extremes of the angle range. The computing time needed to attain the best precision in each experiment is in the range 0.5 to 2 seconds. These times should be considered considered indicative only since at the moment no effort has been put in optimizing the codes. Experimental evidence is consistent with the exponential error bound predicted in [Pel97b] for the Gauss adaptive method.

At the moment we could not find in literature other comparable experimental data. Available data on alternative methods for computing surface-to-surface integrals [HS93, SS96b, SS96a] is relative to the solution of the linearized system with respect to the initial integral equation. This approach allows to use as reference value an analytic solution to the integral equation which can be found in the examples considered in [HS93, SS96b, SS96a]. On the other hand the outcome of the experiment is influenced by the choice of meshing strategy, by the choice of the functional basis and truncation of the functional expansion. Finally, if an iterative method is used, also the convergence of the iterative method influences the result. The computation of the entries of the stiffness matrix is just one of many critical steps. For the above reasons, although the preliminary experimental results are encouraging, further research is needed to asses its practical value.

References

Arvo [Arv95] J. Arvo. Stratified sampling of spherical triangles. *Computer Graphics*, 29(Annual Conference Series):437–438, 1995.

Metodi [BBCM92] Roberto Bevilacqua, Dario Bini, Milvio Capovani, and Ornella Menchi. *Metodi Numerici.* Zanichelli, Bologna, 1992.

Davis [DR84] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration.* Academic Press, London, second edition, 1984.

Finocchiaro [Fin96] Daniele Finocchiaro. Monte carlo evaluation of electrostatic potential and potential energy. Technical Report TR-IMC B4-96-12, Istituto Matematica Computazionale, CNR, Pisa, Italy, June 1996.

Hammer [HH64] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Methuen, London, 1964.

hs-eugm-93 [HS93] W. Hackbusch and S.A. Sauter. On the efficient use of the galerkin-method to solve fredholm integral equations. *Applications of mathematics*, (38):301–322, 1993.

Jerrum [JVV86] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

Knuth [Knu69] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, Reading, MA, 1969.

Niede [Nie92] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *CBMS-NSF regional conference series in Appl. Math.* SIAM, Philadelphia, 1992.

ORourke [O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

Pelle [Pel96] Marco Pellegrini. Electrostatic fields without singularities: Theory and algorithms. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 184–191, 1996. Extended version in Tech. Report IMC B4-96-02, Istituto Matematica Computazionale, CNR, Pisa, Italy.

PelfromF [Pel97a] Marco Pellegrini. Monte carlo approximation of form factors with error bounded a priori. *Discrete & Computational Geometry*, 17:319–337, 1997.

Pell97-TR9715 [Pel97b] Marco Pellegrini. Electrostatic field without singularities: Theory, alghoritms and error analisys. Technical Report IMC B4-97-15, Istituto Matematica Computazionale, CNR, Pisa, Italy, November 1997.

Preparata [PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

Santalo [San67] L.A. Santalo. Integral geometry. In S.S. Chern, editor, *Studies in Global Geometry and Analysis*. The Mathematical Association of America, 1967.

es-eaqi3d-96 [SS96a] S.A. Sauter and C. Schwab. Efficient automatic quadrature in 3-d galerkin bem. Technical Report TR-96-15, Berichtsreihe des Mathematischen Seminars, Christian-Albrechts-Universitt zu Kiel Ludewig-Meyn-Str. D-24098 Kiel., 1996.

ss-qfhpg-96 [SS96b] S.A. Sauter and C. Schwab. Quadrature for hp-galerkin bem in 3-d. Technical Report TR-96-02, Seminar for Applied mathematics, Department of Mathematics, ETH, Zurich, 1996.