New contour reconstruction technique in template parameter space and associated placement

Fabrice Beauville, Damir Buskulic §, Raffaele Flaminio, Frédérique Marion, Louis Massonet, Benoit Mours, Julien Ramonet, Edwige Tournefier, Didier Verkindt, Olivier Veziant, Michel Yvert

LAPP/Université de Savoie, Chemin de Bellevue, B.P. 110, 74941 Annecy-le-Vieux Cedex, FRANCE

Abstract. The generation of a grid of templates and their placement in the parameter space is one of the problems that has to be addressed in the search for binary systems coalescences detectable in interferometric gravitational waves detectors. We present a technique that computes the closed contour of equal match values around a point in parameter space and a first test of a template placement algorithm using these contours. This algorithm could be used to pave the parameter space. First results about the algorithms covering efficiency are also presented.

§ Corresponding author, email: buskulic@lapp.in2p3.fr

1. Introduction

The search for gravitational wave signals coming from coalescing binary compact objects, like neutron stars or black holes, in their inspiral phase is commonly using an optimal filtering technique [1]. Given hypothesis on intrinsic parameters of the binary system, like masses or spins of the objects, a theoretical waveform of gravitational wave, called a template, is calculated and compared to the output signal of an interferometric detector, such as VIRGO [3], LIGO [4], TAMA [6] or GEO [5]. This comparison is made through an optimal filter (or Wiener filter [7]), which is essentially a weighted intercorrelation between a continuous signal $\tilde{a}(f)$ and a template $\widetilde{T}(f)$. The weight is the inverse of $S(f)$, the noise power spectral density of the detector:

$$\langle \tilde{a}, \widetilde{T} \rangle = 2 \left[\int_{f_i}^{f_s} \frac{\tilde{a}(f).\widetilde{T}^*(f)}{S(f)} \mathrm{d}f + \mathrm{c.c.} \right] \tag{1}$$

where f_i and f_s are the lower and upper limits of the antenna spectral window, f_s may be, in some circumstances, the frequency of the last stable orbit (LSO) of the binary system. The continuous signal is $\tilde{a} = \tilde{s} + \tilde{n}$, where $\tilde{n}$ is the stationary noise of the detector and $\tilde{s}$ is the physical signal buried in the noise.

Each template is represented by a point in a multidimensional parameter space. The number of parameters may be reduced by separating them between intrinsic (masses, spins, ellipticity of the trajectories...) and extrinsic ones (time of arrival of the wave, initial orbital phase of the system,...). Extrinsic parameters are taken care of by maximizing the output of the optimal filter over them, like in the case of time of arrival [2]. For intrinsic parameters, it is considered that spins of typical neutron star-neutron star binaries should be of negligible magnitude [9] and ellipticity of the compact bodies orbits should be generally negligible when the signal reaches sufficient frequency to enter the detector sensitivity range. The remaining parameters are the masses of the two bodies, leaving us with a two dimensional parameter space. If, as was proposed by a few authors, the spins misalignement and orbits ellipticity are not negligible, we may end up with a 5-dimensional parameter space.

If, while filtering with a given template, the signal which is present in the data is slightly different from the chosen template, the signal over noise ratio (SNR) found is decreased with respect to the SNR found with the exact template. In other words, in the filtering process, a template corresponding to parameters (λ_1, λ_2) is sensitive to a signal corresponding to nearby parameters $(\lambda_1+\delta\lambda_1, \lambda_2+\delta\lambda_2)$, where λ_1 and λ_2 are the intrinsic parameters of interest, for example the two star masses. Thus, for a given acceptable loss of SNR, each template covers a region in the parameter space. Following Owen [1] in a geometrical interpretation of the optimal filtering, one is able to define a distance between two templates as the ambiguity function maximized over extrinsic parameters, called "match". When filtering a signal which has the same shape as a template of parameters $(\lambda_1 + \delta\lambda_1, \lambda_2 + \delta\lambda_2)$ with a reference template of parameters (λ_1, λ_2), the match is the fraction of the optimal SNR obtained when filtering the reference template with a signal identical in shape to itself.

Given a minimal match MM, we can define the region of parameter space around a given point corresponding to a template T, the match of which, computed with any template corresponding to a point in the region, will be above MM. We will call the boundary of this region the "isomatch contour". It has been shown that, for high values of the minimal match, ($MM > 0.97$) the contour is closed and well approximated by an ellipse [1]. This is not true anymore for lower values of MM, as will be shown below. The final goal of our study is to pave the parameter space with these contours in an as optimal as possible way. This is equivalent to find the minimal set of templates whose iso-match contours pave all the parameter space, without letting any hole or unpaved region [10]. In order to achieve this goal, one has to calculate the shape of the isomatch contours at any given point of the parameter space.

2. The issues : optimality of paving and computation time

Optimality of paving is difficult to obtain. In the case of high minimal match values, several methods were developed to pave the parameter space. For minimal match values lower than 0.95, the isomatch contour shapes may strongly diverge from ellipticity. As an example, figure 1 shows, for a minimal match of 0.9, an isomatch contour calculated at the second post-newtonian order and displayed in the $(\tau_0, \tau_{1.5})$ parameter space, compared to an ellipse which has the same center and which is taken as big as possible while staying inscribed inside the contour. The question is how to optimally pave a space with complex shapes varying from point to point. Up to now, there is no known general method, and we will try to take advantage of the specificities of our analysis to overcome this problem.

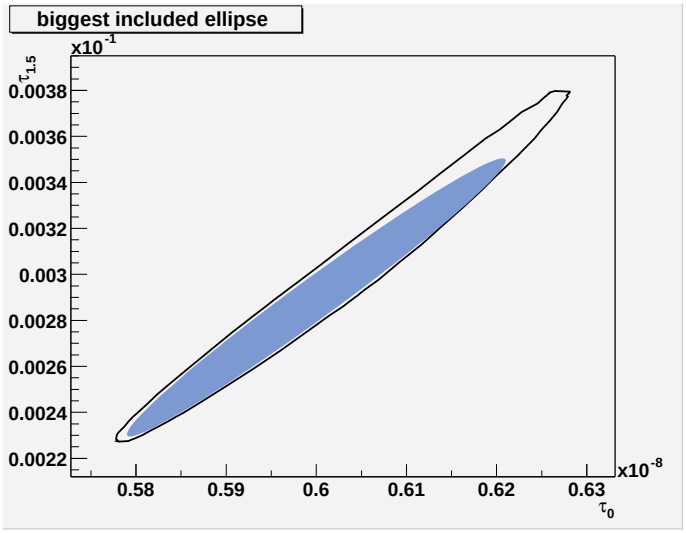

Figure 1. Example of the difference between a calculated isomatch contour (black line) and the biggest possible ellipse inscribed inside it, with the same center point (filled gray surface).

Another problem is the computation time. To calculate the value of the match between two templates typically takes from 0.1 s to 1 s on our test platform† and depends on the position of the corresponding points in parameter space. We used templates calculated by Taylor approximants at the second post-newtonian order. The most computationally intensive task is the calculation of the scalar product defined above (1). The longest step for this computation relies on Fast Fourier Transforms of the signals. One could think of calculating a 2D image of the match around a point in order to extract a given minimal match isocontour. The computing time needed to calculate an image of 100*100 points would take anything between 15 min and 3 hours on an average computer. It is impossible in these conditions to do any placement. So comes the necessity to find faster ways of extracting the contours.

3. Reconstruction of the contours at any point in parameter space

The principle of the contour reconstruction is a two step process. First, one reconstructs exact contours for a small number of points scattered among the parameter space. This exact reconstruction is the longest step in the final template placement task and it has to be as fast as possible. Then, focusing only on the geometrical shape of the contours, one is able to interpolate this shape in any parameter space point. This step, because it involves only simple geometrical operations, is very fast.

3.1. Reconstruction of exact contours

In order to be as fast as possible, it is necessary to avoid as much as possible the calculation of the match on a big number of points. One builds a "skeleton" of the contour by starting from the central point, where the match is one, and following the line of minimal steepness. As illustrated on figure 2, the algorithm is the following :

- From the central point S_0, make a small vertical step, going to point S_1
- Search, on the horizontal line passing through S_1, the point with maximum match. This point is called S'_1. The direction $\overrightarrow{S_0S'_1}$ is the progression step.
- Search, on the direction perpendicular to $\overrightarrow{S_0S'_1}$ the two points corresponding to the requested minimal match.
- Then, for each step, calculate the next step with
  - Use the previous $\overrightarrow{S'_{i-1}S'_i}$ vector to do a step starting from S'_i.
  - Search for the maximum of the match on a line perpendicular to the previous step direction. The corresponding point will be called S'_{i+1}.
  - On this line, search for the two points corresponding to the requested minimal match.

  Do this until S'_i corresponds to a match wich is lower than the requested minimal match.

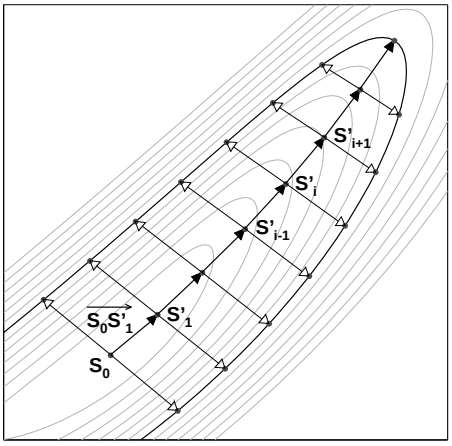

Figure 2. Reconstruction of an isomatch contour by following the line of minimal steepness. At each step, a search is made perpendicularly to the line of progression to find the point corresponding to the required minimal match.

Each point is searched with a very simple binary method, which ensures that a small number of points are calculated, ensuring the overall speed of the method. Of course, there is a trade-off to be found between the number of calculated points and the computational problems coming from the irregularities of the match function or numerical limitations. Furthermore, the steps are not equidistant. Smaller steps are made at the start near the maximum of the match function and near the end, where all the points converge.

The reconstruction time of a single contour ranges in our tests from a few tens of seconds to a few tens of minutes, depending on the position of the point in the parameter space. These tests were made, using a library called "Inspiral", with the following typical values of the signal parameters: mass range of [1 $M_\odot$;30 $M_\odot$], minimal frequency $f_{min} = 30$ Hz, maximal frequency f_{max} corresponding to the last stable orbit (LSO) of the binary system or half of the sampling frequency, whichever is the lowest. The noise power spectral density is the standard Virgo noise.

3.2. Geometrical shape interpolation

Once a set of exact contours has been calculated on points scattered on the parameter space, we will call these contours "seed contours", it is possible to obtain an approximate shape of an isomatch contour at any point by doing a geometrical interpolation. Focusing only on the geometrical shape of the contours, one needs three exact contour shapes (in a two dimensional space) to interpolate linearly a contour at any given point in the triangle defined by the centers of the exact contours. Before doing so, one has to be sure that any point in the parameter space belongs to one and only one such triangle. Given a set of points, in our case the centers of the exactly calculated contours, the procedure

† An Alpha XP1000 667 MHz workstation

which determines all the three-points sets forming all the possible triangles such that the previous condition is fulfilled is called triangulation in computational geometry.

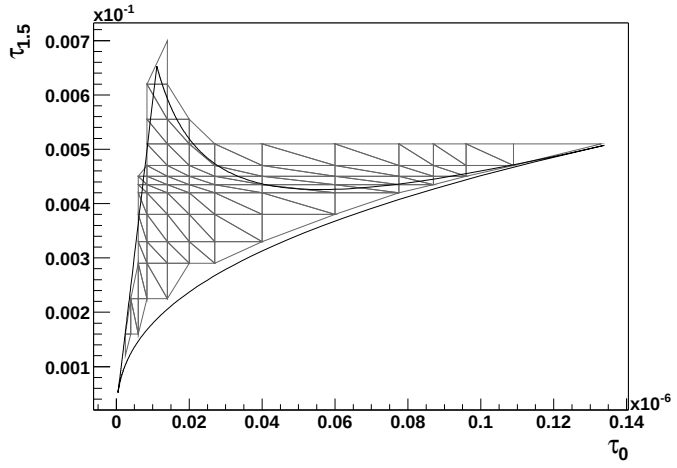

Figure 3. Pseudo-triangulation of the parameter space given a set of points scattered on a regular square grid.

For the first tests we have made, the triangulation was obtained semi automatically, the position of the points being decided a priori on a rectangular grid. The result is illustrated on figure 3. It is not strictly speaking a triangulation since parts of the parameter space were not covered, but it is enough for our initial purpose.

Figure 4 illustrates the interpolation step. The contour in the middle of the triangle is linearly interpolated between the seed contours the centers of which are at the corners of the triangle.

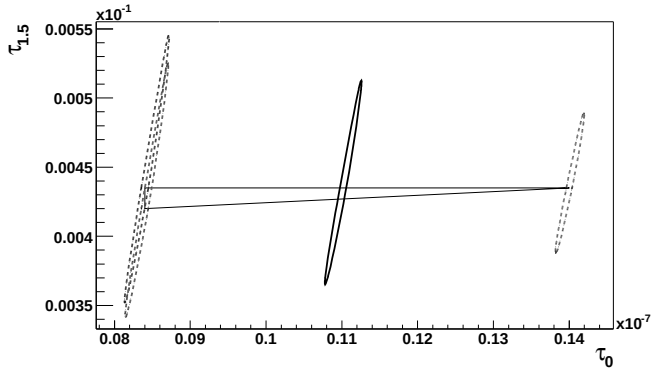

Figure 4. Linear interpolation between exactly calculated contours at an arbitrary point in a triangle.

Though the interpolation calculation is very fast, it remains to be seen how well the interpolated contour approximates an exact contour that may be calculated at the same point. On figure 5, we show one of the worst cases we have seen. The difference may

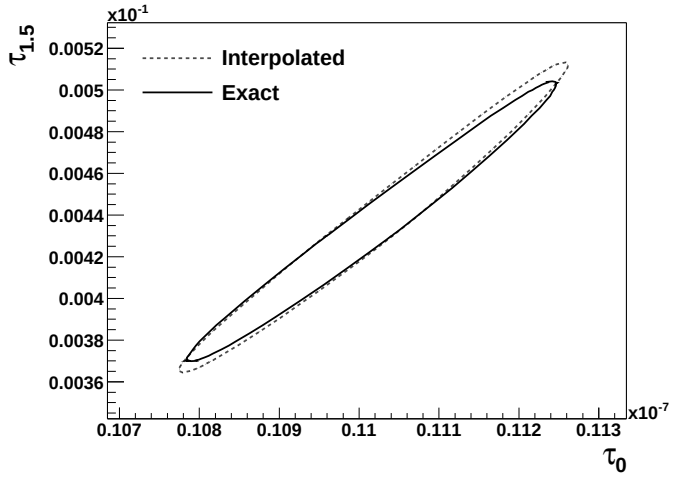

Figure 5. Example of a difference between an interpolated and exact contour. The exact contour is in plain while the dashed one is interpolated. The masses of the binary are $m_1 = 14.7$ M$_\odot$, $m_2 = 0.76$ M$_\odot$.

be explained by several factors, among which the fact that we use a linear interpolation and the manual, therefore not optimal choice of the grid of seed contours are the most important. It has to be noted that it may be interesting to use some kind of spline-based interpolation and to improve the choice of the seed contours position, but we will see that the kind of discrepancies noted above do not seem to influence fundamentally the final result, which is the full coverage of the parameter space by the template placement.

4. Template placement

4.1. Principle in the simple case of circles

In order to have clear ideas, it is interesting to look at the very simple case of paving a plane with circles. As was already noted [10], because of the rotational symmetry, the centers of the circles should sit at the vertices of regular polygons which make a regular tiling of the plane. This is only possible for triangles, squares or hexagons. In the first case, the centers of the circles are placed on the corners of an equilateral triangle, as shown in figure 6 A). It is desirable to have the sparsest possible circles, which means that three circles touch at one single point P. The surface region consisting of the points whose closest circle center is C is shown in gray. This is also the surface covered on average by one circle. In the triangular case, it is a hexagon. The set of points which belong to this region is called the Voronoi set of C. As illustrated on figure 6, in the case of a square tiling, the Voronoi set has a square shape and in the case of a hexagonal tiling, the Voronoi set has a triangular shape. It has been shown [10], as one would intuitively expect, that the most efficient tiling in the case of circles placement is the triangular one.

It is impossible to readily extend this simple case to the one where contour shapes

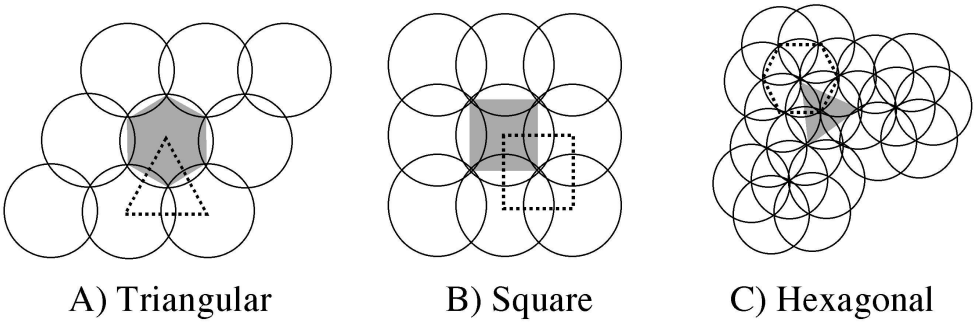

Figure 6. Paving of a plane with different elementary cells. The relevant Voronoi sets are shown in gray.

are complex and changing along the position in parameter space. But it is possible to exploit some specificities of isomatch contours, as we will now show.

4.2. Placement of a set of three contours

The first step of a placement procedure is to place correctly three neighbouring contours C_1, C_2 and C_3 associated to templates $\widetilde{T}_1$, $\widetilde{T}_2$ and $\widetilde{T}_3$, assuming that a local triangular tiling is still the most efficient. The properties of isomatch contours that we can use are the following :

- Whatever the closed shapes we use, we will make the reasonable assumption that the property seen with circles, namely that in the most efficient case three contours touch at one single point, remains true. This implies that the contour shapes are not too irregular which will limit the lowest minimal match we wish to consider. The value of this lowest minimal match is not obvious though.
- One very important property coming from (1) is the match symmetry. If $\widetilde{T}_1$ and $\widetilde{T}_2$ are two templates, one has :

  $$\langle \widetilde{T}_1, \widetilde{T}_2 \rangle = \langle \widetilde{T}_2, \widetilde{T}_1 \rangle \tag{2}$$

  On an isomatch contour C_1 around a point corresponding to a template $\widetilde{T}_1$, all the points have by definition the same match M with respect to $\widetilde{T}_1$. Let $\widetilde{T}_2$ be a template associated to a point on C_1. Let C_2 be the isomatch contour around $\widetilde{T}_2$ of value M. Then, because of the symmetry property (2), the point associated to $\widetilde{T}_1$ will be located on C_2 (figure 7).

The combination of the two previous properties leads to the idea of a guiding contour. A base cell of the placement is formed by three contours crossing at a single point P. This point may be used to build a contour C_g. Because P is located on each of the three contours C_1, C_2 and C_3, and because of (2), the three centers (points correponding to central templates $\widetilde{T}_1$, $\widetilde{T}_2$ and $\widetilde{T}_3$) are located on the guiding contour C_g.

Taking the opposite stance, one can start from a guiding contour, search for the best location of three points on it and decide that these are the positions of three

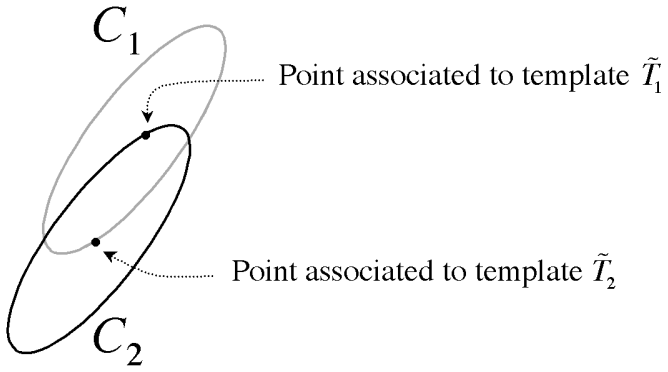

Figure 7. Illustration of match symmetry between two templates. C_1 and C_2 are isomatch contours corresponding to the same match value M around points corresponding to templates $\widetilde{T}_1$ and $\widetilde{T}_2$. Each point is located on the contour associated to the other template.

contour centers (figure 8). One is assured that those three contours will cross at only one point, which is the center of the guiding contour. The optimal position of the points

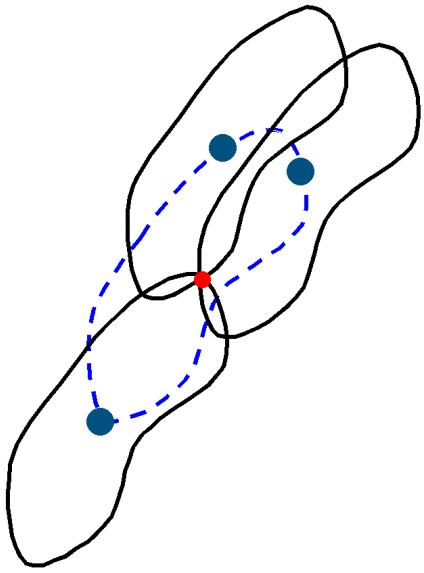

Figure 8. Role of a guiding contour (dashed line) in the placement of a three contour set.

is searched by maximizing the surface of the triangle formed by these points. In the case of concavities of the shapes, holes may appear in the coverage of the space. In that case, one has to modify the position of the points, still staying on the guiding contour [10].

Figure 9 shows a real case where one sees that the shapes of the three contours are not constant but they still cross at the center of the guiding contour.

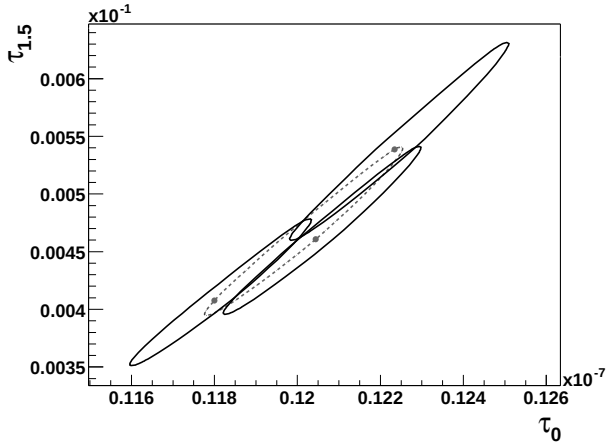

Figure 9. Placement of three contours in the (τ_0,$\tau_{1.5}$) plane

4.3. Placement for the whole parameter space

The principle of placement for the whole parameter space should take advantage of the described symmetries. After the placement of a set of three contours C_1, C_2 and C_3 on positions P_1, P_2 and P_3 using a guide C_g^0, each pair of contours has two intersections (actually, if it has more, one should change the positions on the guiding contour). Let us consider the (C_2, C_3) pair. The first intersection I_c is the one which is common to all three contours, the second I_c' may be used to position a new guiding contour C_g^1. This new guiding contour will have two points already defined, the centers P_2 and P_3 of the (C_2, C_3) pair. If one searches for a third point P_4 such that the triangle (P_2,P_3,P_4) has a maximal surface area, this point will be the center of a contour C_4 which will be correctly placed (figure 10).

Using iteratively a guiding contour on each intersection of a newly placed contour with previously placed ones, it should be possible to pave the parameter space. Unfortunately, the imperfections of interpolated contours, as well as some numerical imprecisions make this simple algorithm fail.

Searching for a suboptimal solution, we ended up with the following algorithm to obtain a coverage of the space line by line (figure 13):

- Start by placing a set of three contours.
- Using the previously described simple algorithm, place iteratively contours such that they form a two lines set crossing the parameter space (figure 11).
- Crossing points between different contours are present on the two edges of the previous two lines set. Take one such point and use it as a starting point to build a second set of two lines independently of the first set, except for this point.
- One of the lines of the second set will be by construction approximately superimposed to a line of the first set (figure 12). Keep only the external line.
- Execute iteratively the two last steps until the space is covered.

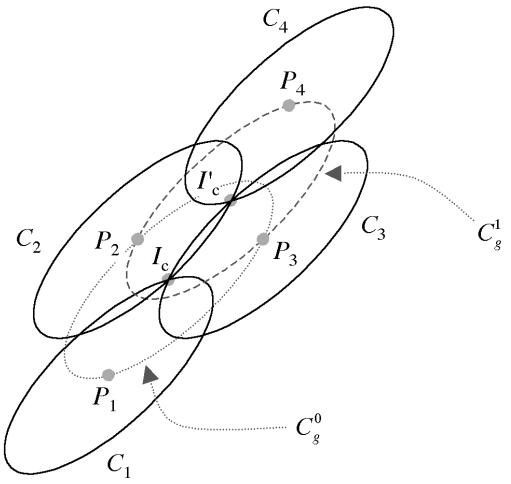

Figure 10. Starting from a set of three contours C_1, C_2 and C_3, placement of a fourth one, C_4, with the help of a guiding contour C_g^1 which center is at a free intersection of C_2 and C_3, I'_c

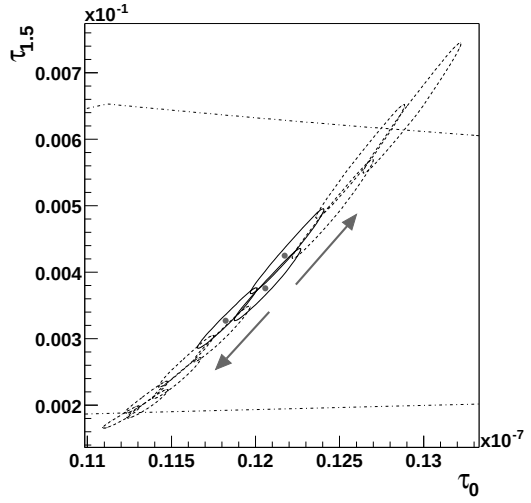

Figure 11. Starting from a three contour cell, building of a set of two lines of contours crossing the parameter space. The two arrows indicate the directions of progression while building the lines.

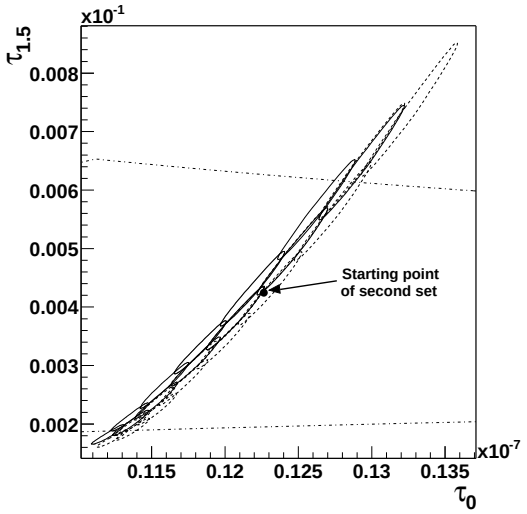

Figure 12. Independent placement of the third line starting from a crossing point on the edge of a first two lines set.

As can be seen on figure 13, the space is not completely covered in our preliminary tests, and this is due to the limits of the algorithm when reaching places where the size of the contours becomes bigger than the cross-section of the parameter space. We expect to improve this in future releases of the algorithm.

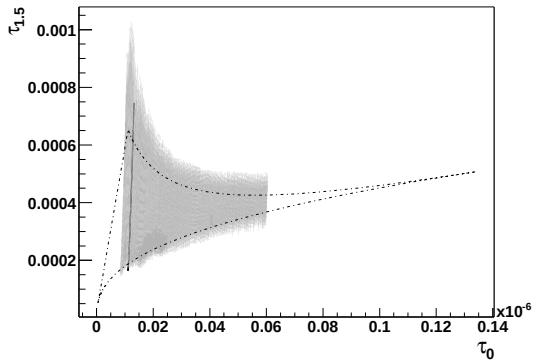

Figure 13. Placement on the major part of parameter space following the algorithm described in the text, line by line. The arrow shows the direction of progression.

4.4. Computational cost

The estimated computation time is still in evaluation but is of the order of a few hours on the test platform for the generation of 80 seed contours and between 10 and 20 minutes for the placement. We obtained around 6300 placed templates for a requested minimal match of .9 in the limited region shown on figure 13. Reminder, the mass range in our tests is [1 $M_\odot$;30 $M_\odot$], and the minimal frequency $f_{min} = 30$ Hz. This result is clearly very preliminary and our aim is to improve it.

4.5. Test of the coverage of parameter space

In our algorithm, each line of contours is built independently of the others, except for one point, which implies that holes may appear between the lines. In our understanding, this is due to the small differences that may appear between interpolated and exact contours. We tried to estimate the effect of such holes on the coverage of the parameter space. Test points were generated randomly and uniformly distributed in a limited region of parameter space ($\tau_0 \in [10^{-8}; 1.8\ 10^{-8}]$ and $\tau_{1.5} > 2\ 10^{-4}$ in geometrized units). The match with the closest template was calculated. Figure 14 shows the position of points having a match lower than the required minimal match of 0.9 in this example. It is obvious on this figure that most of the points lie in holes left by the placement algorithm.

The distribution of the match with the closest template for all generated points is shown on figure 15. The requested minimal match was 0.9. Figure 16 shows the proportion of surface that is not covered by the placement algorithm versus an effective minimal match different from the one used for the placement. In other words, this shows the inefficiency of the placement algorithm corresponding to a requested minimal match of 0.9, with respect to an even lower test minimal match. If one sticks to a minimal match of 0.9, the inefficiency is around 4%. The quoted errors are only statistical.

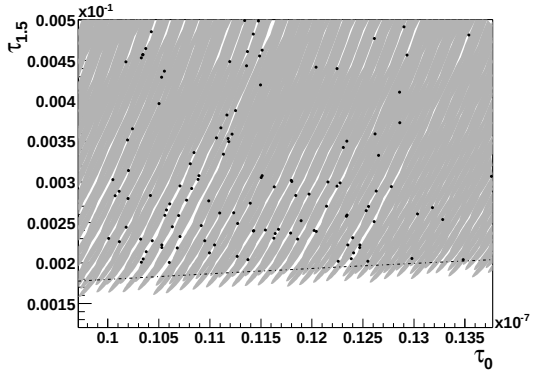

Figure 14. Location of points having a match lower than 0.9 with the closest template. A Monte-Carlo technique was used to perform the test.

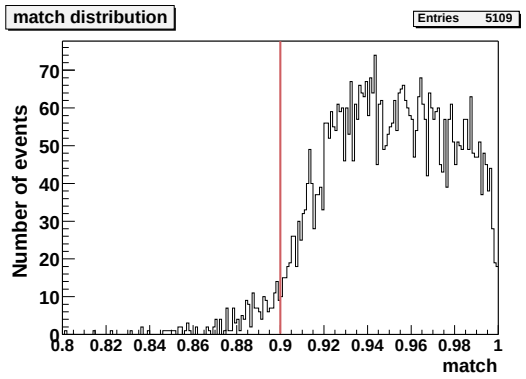

Figure 15. Match distribution of randomly generated test points. The vertical line shows the requeted minimal match for the placement, which was 0.9.

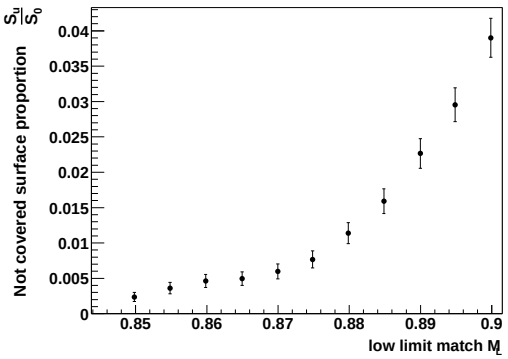

Figure 16. Surface proportion not covered by the placement algorithm if one uses an effective minimal match different from the one used in the placement. Only statistical errors are shown.

5. Conclusions and perspectives

We have presented an isomatch contour reconstruction technique in template parameter space that allows to build quasi-exact isomatch contours after a first pass of exact reconstruction on a fixed set of points giving so called seed contours. A placement technique was also described.

After having generated a test grid of templates with this technique, a set of test points were randomly scattered in a portion of the parameter space in order to test the coverage efficiency by the grid. Only 4% of the points had a match lower than the requested minimal match, falling into holes left by the placement.

Though the placement method is computationally intensive, there is a lot of room for improvement. Some more general improvements are also to be done, like design a

better placement to remove some remaining holes, a better triangulation using Delaunay methods [11], which should help the placement, and finally do a better interpolation between seed contours, using for instance splines.

6. Acknowledgments

We would like to thank all our VIRGO colleagues who participated to the development of the Inspiral library, and particularly Andrea Viceré for his insightful comments and dedicated work.

[1] *Search templates for gravitational waves from inspiraling binaries: Choice of template spacing*, B.J.Owen, Physical Review D, **53**(1996) 6749-6761

[2] B.F. Schutz, in *The Detection of Gravitational Radiation*, Cambridge University Press, Cambridge, England, 1989

[3] VIRGO Coll., Final Design Report, 1997, see also http://www.virgo.infn.it/

[4] http://www.ligo.caltech.edu/

[5] *The GEO600 gravitational wave detector*, Class. Quantum Grav., **19**(2002) 1377-1387, see also http://www.geo600.uni-hannover.de/

[6] *Current status of TAMA*, Class. Quantum Grav., **19**(2002) 1409-1419, see also http://tamago.mtk.nao.ac.jp/tama.html

[7] See, e.g., N. Wiener,*The Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications* (Wiley, New York, 1949)

[8] *Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement*, B.J.Owen, B.S. Sathyaprakash, Physical Review D, **60**(1999) 022002

[9] *Gravitational-Radiation Damping of Compact Binary Systems to Second Post-Newtonian Order* L. Blanchet, T. Damour, B.R.Iyer, C.M. Will and A.G. Wiseman, Phys. Rev. Lett. **74**(1995), 3515

[10] *Optimum Placement of Post-1PN GW Chirp Templates Made Simple at any Match Level via Tanaka-Tagoshi Coordinates*, R.P. Croce, Th Demma, V. Pierro and I.M. Pinto, gr-qc/0110024

[11] See, e.g., J. O'Rourke,*Computational geometry in C* (Cambridge University Press, Cambridge, 1998)