Fitting Watertight NURBS Patchworks over Irregular Curve Networks with LEANURBS

Bastiaan Veelo, SARC, Trondheim/Norway, Bastiaan@SARC.nl
Herbert Koelman, SARC, Bussum/Netherlands, H.J.Koelman@SARC.nl

Abstract

This paper presents a novel method for the extraction of a patchwork of NURBS surfaces from a solid model that is represented by a network of intersecting curves of arbitrary topology. Its purpose is to produce export files in standard formats such as IGES and it specifically addresses issues that third party software have in dealing with high numbers of patches and minute gaps due to fitting tolerances. The patches produced by this method can cover large parts of the model and reduce the number of exported patches by several orders of magnitude. Touching edges of adjacent patches are mathematically equivalent so that the patchwork is free of gaps.

1. Introduction

Non-uniform rational B-spline surface patches (NURBS patches) form the basis of most geometric modellers. A NURBS patch is a mathematical method that is used to describe the shape of a sculpted surface, and its application in design software is more due to convenience for the programmer than to productivity of the designer: NURBS patches have a number of drawbacks that designers are forced to fight with if they are to use such modellers to sculpt a large smooth surface of arbitrary topology with strict geometric requirements such as a ship hull. These drawbacks are described in detail in *Veelo (2004)* and are generally acknowledged, see *Sharma et al. (2012).* Fortunately for naval architects, alternative hull form modellers exist, such as Fairway — the versatile hull design module of PIAS developed by SARC — that has been in production for decades, *Koelman and Veelo (2013).*

Fairway eliminates the drawbacks of NURBS patches by not using them for design tasks altogether. Instead, the designer works on a network of intersecting curves. The topology of this network is arbitrary and the curves are under the complete control of the designer, without unintended dependencies in their definition. Any number of curves can intersect in one point and the mesh cells in this network can have any number of sides. Based on the curve geometries, the system automatically constructs and maintains a continuous watertight surface across the entire network by filling the cells with another kind of surface patches: so-called transfinitely interpolating surfaces. NURBS aren't applicable here. This surface is used for visualisation, for interpolation of new curves and for generating faceted export files like STL.

Apart from the freedom in modelling that the designer is offered by this approach, an important aspect of Fairway is its capabilities for curve fairing. Even though a continuous surface is defined across intersection points, curves can be allowed to deviate from each other at their common intersection by a configurable degree, what we call the fairing deviation. This way, network consistency can be temporarily traded for increased curve fairness, in support of an iterative process towards a consistent and fair surface — much in the same way as lines plans have traditionally been faired by hand, see *Koelman (1997, 1999).* In practice, the fairing deviation is seldom completely reduced to zero but left at a certain production accuracy. This fact is of relevance for the method presented in this paper.

The irony, though, is that one can seek to solve challenges in a design process by inventing new methods, but designers don't live on an island: they rely on the possibility to exchange the product of their design with other systems. In earlier years, data exchange in the form of curves was sufficient, because all that was required for production were curves: stiffener profiles, plate contours and template drawings are all unambiguously defined by curves. But in modern years, where structures are analysed numerically and evaluated in virtual reality, surfaces play an increasingly important role. These third party systems expect data to be handed to them in the form of NURBS patches. And so

there is an additional problem to be solved, the problem of conversion between essentially incompatible data representations.

Until recently, Fairway had tackled that problem through the pragmatic application of a conventional NURBS surface fitting algorithm to a regular grid of surface points for each network cell individually. Each four-sided cell produced one NURBS patch and each *n*-sided cell produced *n* smaller NURBS patches. Although that method is straightforward and relatively time-efficient, a typical production-ready model can contain hundreds of curves producing thousands of very small NURBS patches. And, due to the approximating nature of fitting algorithms, there would always be minute gaps between the patches no matter how low the tolerance would be set. Perhaps not surprisingly, systems on the receiving end have problems dealing with such patch clouds — which in itself illustrates that NURBS patches are not the be all and end all of surface representation.

Fairway users desire the ability to export a hull form as a smaller set of larger NURBS patches that do not need cumbersome surface repair or stitching. By means of a considerable investment in research and development, SARC has succeeded to meet this desire with a completely new method for constructing NURBS patches over curve networks — baptized LEANURBS, an acronym for **L**owest **E**ffective **A**mount of **NURBS** — which is presented in this paper. The next section will briefly recap surface fitting theory. Section 3 presents the theoretical core of our contribution to fitting a 3D surface to irregularly structured data. Section 4 is about the measures that prevent gaps between patches, including considerations regarding the fairing deviation mentioned earlier. Section 5 discusses the definition of patch contours and partitioning of the shell. Section 6 shows that theories not always work in practice, and describes adjustments that make them work most of the time nonetheless. Words about performance are given in section 7, followed by applications and future work in section 8.

2. Basics of surface fitting theory

Fitting parametric surfaces to point clouds is a field of research on its own, and we only have space for a coarse classification of approaches here, and limit ourselves to the fitting of quadrilateral patches:

1. Fitting a 3D quadrilateral patch to a grid of points. The grid is required to be *regular,* with points evenly distributed in rows and columns.
2. Fitting to an irregularly structured point set on a rectangular domain. The fitting is done *in one dimension only.*
3. Fitting a 3D surface to an unstructured point set.

Algorithms for 1. and 2. are presented in *Dierckx (1993),* amongst other places. The first approach is used in Fairway for the fitting of NURBS patches to individual network cells and comes with the drawbacks that we have set out to eliminate. Since we want a single patch to cover a larger region of our network of curves, which has an arbitrary topology and cells are arranged irregularly, we cannot produce a regular grid of points that covers the geometry of the network in sufficient detail. So we need an approach that applies to an irregularly structured point set.

The second approach does cover the irregularity requirement but is not directly applicable because NURBS control points (which are regularly structured) can only be adjusted in one dimension to match the irregularly structured data points. This can only be used to fit to, for example, a height field in a topological map, not to shapes that are curved in all three dimensions such as ship hulls.

What we seem to need is an algorithm from the third category. However, this kind of problems is hard to solve and typically requires application of artificial intelligence. The field is still being actively researched, see *Zhang et al. (2016)* and *Marinić-Kragić et al. (2018)*. Besides, the approximative nature of the algorithms also applies to the patch borders, meaning that the problem of gaps between patches remains.

3. 3D fitted surfaces over irregularly structured data

What we have found is a way to simplify the problem, so that the straightforward one-dimensional fitting algorithm (from the second category above) can be applied to construct a three-dimensional patch that fits the irregularly structured data.

Given a rectangular region on the Fairway surface that we want to fit a NURBS patch to, we start with a simple surface patch G that coarsely approximates the internal surface shape of the region, but is an exact match along the borders. The difference between G and the Fairway surface can be seen as an offset. Computing offsets of NURBS surfaces is in general a complicated matter, see *Piegl and Tiller (1999)* and *Ravi Kumar et al. (2001)*, but we can keep things relatively simple in this case. The non-uniform offset can be expressed as a direction vector field O and a magnitude, the scalar field S. G, O and S are all bivariate "surfaces" in the mathematical sense that they are defined on a two-dimensional parameter space. This means that the fitted surface F can be expressed as the linear combination

$$F = G + O \cdot S \tag{1}$$

Since G matches the region borders, S is zero along the borders and the direction O is irrelevant along the borders. This means we can choose O as we see fit, leaving S as the only unknown. S is one- dimensional and has a rectangular domain, so it can be generated easily by an algorithm of the second category that is readily available from *Dierckx (1993)*.

Fortunately, every component on the right-hand side of equation (1) can be expressed as a B-spline surface patch, as we'll illustrate shortly. B-splines are equivalent to NURBS with uniform weight factors, making them non-rational. Standard algorithms from *Piegl and Tiller (1997)* can be applied to make these B-spline patches compatible with each other, with identical degrees and knot vectors, and equal numbers of control points. With compatibility in place, F can be computed as a (non-rational) NURBS by applying the arithmetic operations in (1) on the control points of G, O and S. We'll now take a closer look on how G, O and S can be defined.

3.1. Definition of the basis surface G

The basis surface can be defined as the bilinearly blended Coons patch expressed in B-spline form, interpolating the four boundary curves of the region. This requires the boundary curves to be defined as B-spline curves, but more on that later. As a first step, opposing boundary curves are made compatible by standard procedures of degree raising, reparameterisation and knot refinement. The three components of the Coons patch (two ruled surfaces between two pairs of opposing boundaries and one bilinear surface between the corners) are easily constructed and expressed in B-spline form. These are then made compatible again and combined linearly. This process, and the algorithms that produce compatibility, are described in detail in *Piegl and Tiller (1997)*. Because this surface is an exact match at the patch boundaries, the internal shape of the patch is a suitably coarse approximation of the internal shape of the Fairway surface region.

3.2. Definition of the vector field O

As we have hinted to earlier, we are quite free to choose O, as long as there is a unique projection for any pair if parameters (u, v) from $G(u, v)$ through $O(u, v)$ onto the Fairway surface within the bounded region. A uniform direction vector extracted from the linear component of the Coons patch would probably work in many cases, but we expect better results if the local normal vector on G can be used. However, textbooks teach that the surface normal of a B-spline patch cannot itself be expressed as a B-spline because of the square root in normalisation. Fortunately, normalisation is not important to us: As long as we have a direction vector of any non-zero length we can scale it to get from G to F. As it turns out, a field of direction vectors that is merely *orthogonal* to G can be expressed in B-spline form.

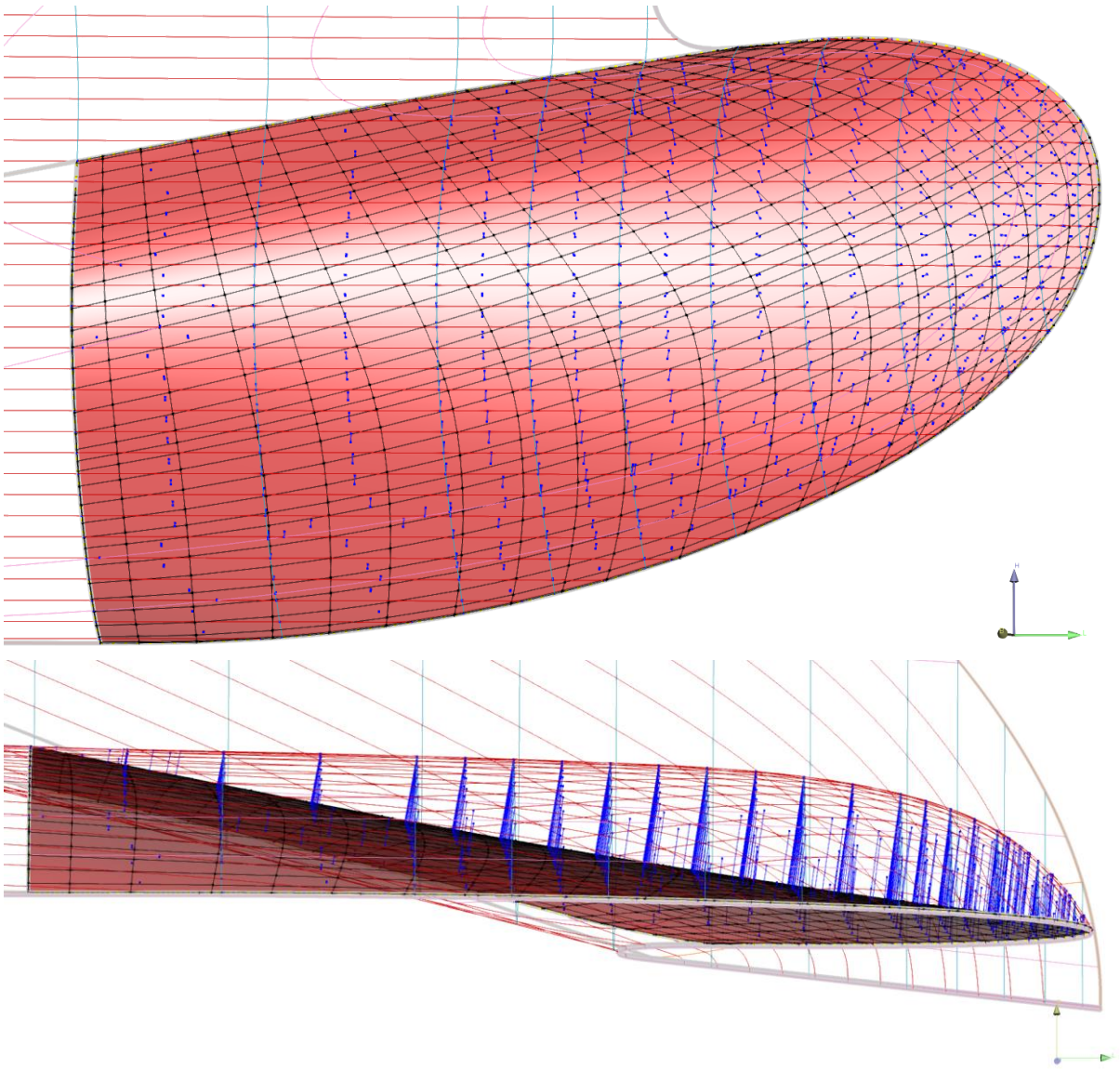

Fig.1: The basis Coons patch G with orthogonal offset vectors to the Fairway surface, viewed somewhat from the side (upper figure) and below (lower figure).

First, two B-spline representations for the first derivative of G are constructed, one in either parameter direction u and v, representing tangent vector fields in these directions. This is accomplished by standard procedures described in *Piegl and Tiller (1997)*, and results in B-spline patches of lower degree in one direction and also lacking one row or column of control vectors compared to each other. Compatibility in degrees is restored by degree raising, which also doubles (internal) knot values and adds rows or columns of control vectors. So, once again, application of knot refinement algorithms produce compatibility between the two derivative patches, so that they only differ in the values of control vectors. Taking the cross product of corresponding control vectors produces orthogonal control vectors that, when combined with the knot vectors of the compatible tangent patches, produces the B-spline representation O of the vector field that is locally orthogonal to G. It does no harm to add a normalising step after the cross product to give the control vectors equal length, and helps reduce unnecessary variation in O so that the fitting algorithm may have an easier job to find a matching S.

3.3. Determination of the scalar field S

The scalar B-spline patch S is produced by the fitting algorithm, which takes as input scale factors s_i for certain (u_i, v_i) parameter pairs that are irregularly distributed over the rectangular domain. These parameter values are produced by a search: starting from a known three-dimensional point p_i on the Fairway surface (on a curve or somewhere on a transfinite surface) we search for the (u_i, v_i) for which $O(u_i, v_i)$ passes through that point p_i and $G(u_i, v_i)$. That is, we minimise the shortest distance between the point p and a line through $G(u, v)$ in direction $O(u, v)$. Once the parameters have been determined, s_i results from the vector projection of the difference between p_i and $G(u_i, v_i)$ on $O(u_i, v_i)$:

$$s_i = \frac{(p_i - G(u_i,v_i)) \cdot O(u_i,v_i)}{|O(u_i,v_i)|^2} \tag{2}$$

Note that the vertices of G are ordinary control points, the vertices of O are three-dimensional vectors and that the vertices of S are one-dimensional scalars.

Figure 1 shows an example of what the basis Coons patch G looks like, applied to the bulb of the "Duisburg Test Case", *el Moktar (2015)*. The network of resulting NURBS control points is shown in black. Note the exact fit to the Fairway curves around the patch boundaries, and how offset vectors are locally orthogonal to G.

4. Mind the gap!

The definition of the basis surface G in the previous section assumed the boundaries to be available as B-spline curves. Fortunately, the majority of curves in a Fairway model are effectively B-splines already (NURBS with uniform weight factors). They can however extend past the patch region and therefore may be too long. The standard algorithm of knot insertion, also given by *Piegl and Tiller (1997)*, allows a B-spline curve to be trimmed at any parameter value, producing a new B-spline with a shape that is exactly identical to the original (for as far as it extends). This implies that two adjacent patches can be given borders that are mathematically equivalent, producing an absolutely watertight connection. The two patches can even run along different lengths of the curve, producing a T-joint, still without gaps because their borders are derived from the common boundary curve.

Not all curves in a Fairway model are pure B-splines: conic sections like circular arcs are rational B-splines, that is, proper NURBS. Secondly, Fairway allows so-called slave curves to be defined as the linear combination of other curves and constants. Slave curves have no NURBS representation at all. These non-B-spline curves can be converted to B-splines by sampling them densely and fairing a B-spline through them by application of Fairway's built-in fairing methods. This is an approximation, but as long as the same procedure is performed for both patches on either side of such a curve, it will produce an identical border.

So, by deriving the four bounding splines of G from the curves in the Fairway network, we principally ensure that adjacent NURBS patches possess identical geometries on their shared border, and should be free of gaps. However, there are two intricacies that require special actions to defend that principle.

Firstly, the fitting algorithm always produces an approximation according to a given tolerance, meaning that S is not guaranteed to be zero along the borders. Since G is already a perfect fit along the borders, in order to not introduce a perturbation there, we simply transfer the boundary control points of G (after it has been made compatible with O and S) straight to F and only perform the arithmetics on internal vertices.

The second intricacy is due to the fairing deviation mentioned earlier, which intentionally allows curves in a Fairway model to deviate from their shared intersection point by a configurable degree. So after boundary curves have been trimmed to the correct length, they probably do not yet connect exactly. In order to produce watertight connections at the patch corners, the trimmed curves are given an

affine transformation to stretch and tilt them so that they start and end at common intersection points with other bounding curves. Because of the affine invariance property of B-spline curves, the transformation can simply be applied to the control points of the curve. However, this is not a straightforward process: When the corner of one patch is on the edge of another patch (forming a T-joint), that corner should be shifted onto the other edge after that edge has been stretched and tilted. And because that edge may connect to a corner (or two) that itself forms a T-joint, there is a tree-like dependency graph that must be traversed of curves that should be stretched and tilted in order. We are talking about minute adjustments to the curves, but if we are to guarantee a patchwork that is free of gaps, we need to be pedantic.

To conclude this section, we should probably address a question that the observant reader may have: "Why couldn't degree raising, reparameterisation and knot refinement be used to achieve compatibility between touching NURBS patches generated by the old scheme, eliminating gaps there?" The reason is that the fitting algorithm of the first category produces a B-spline surface patch that has a parameterisation that is not necessarily linearly compatible with the network curves at its borders. A non-linear re-parameterisation is possible but complicated and approximative, and would increase the number of control points for little gain. It would not completely eliminate gaps and therefore not be worth the trouble. To the contrary, an algorithm from the second category works on input where each sample point has an accompanying pair of parameters, so that the fit is actually performed in parameter space and therefore needs no reparameterisation. That is an essential trait in the elimination of gaps.

5. Partitioning the shell

Having developed a theory for fitting NURBS surface patches to large regions over irregular curve networks, remains a practice for partitioning a network into four-sided regions. Fortunately, the Fairway graphical user interface has a tool for defining regions already, in which the user identifies corners of the region in a clockwise order. This tool is used to paint regions of the shell in a different color, or define regions of developable surfaces or define the contours of shell plate expansions. We have simply extended its use for the definition of export surfaces, if the region has exactly four corners and no knuckles in the internals of the region sides. Currently, the user is responsible for the partitioning of the shell into export regions interactively.

During this interactive process, the user is likely to experience that partitioning a ship hull, which is of arbitrary topology, into purely four-sided regions not always has an obvious solution. Often one is left with one or more "holes" with three or five or more sides. Again, this is another reminder that NURBS surface patches are not the be all and end all for surface design. Nevertheless, a working solution can usually be found with a few creative measures.

A three sided hole can in some cases be successfully filled with a four sided patch by cheatingly placing one of the corners somewhere along one of the sides of the hole. This is called a degenerate corner, and works best where the side has a bend with high curvature and the surface is ideally flat. An example would be the transom stern, bounded by the deck line, centre buttock and aft frame: the degenerate corner can be placed in the bilge of the frame. Another example can be seen in Figure 1, which has a degenerate corner in the lower front of the bulb.

If a hole has more than four sides, it can be subdivided into two or more four-sided regions by interpolating auxiliary curves, for example by intersection with an oblique plane. Such an auxiliary curve can be seen in Figure 2, in the bilge at around the middle of the aft ship, where three regions meet in a Y-formation.

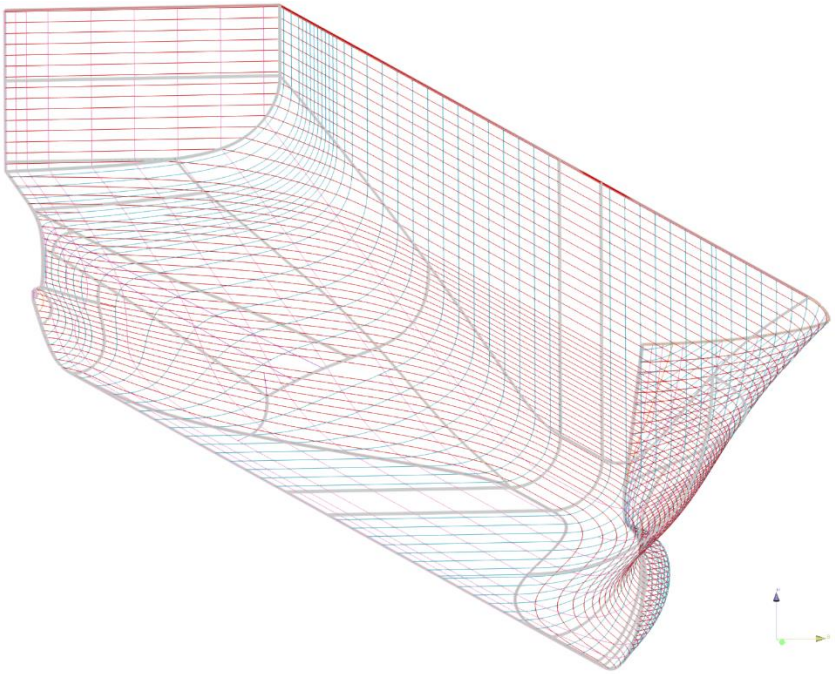

Fig.2: The "Duisburg Test Case", *el Moctar et al. (2015)*, partitioned in quadrilateral shell regions.

6. Serving the fitting algorithm right

This is the part where we discover that theories not always work in practice. When we finally completed the implementation of all algorithms and had assembled a working system, our first tests delivered results of varying quality, some of which were downright disappointing. The culprit turned out to be the fitting algorithm, which showed to be picky about the data that it is served. The symptoms were warnings about rank deficiency and severe rippling in the generated patches.

The core of the problem, actually, is that NURBS patches are built on a regular grid of control points that are distributed along rows and columns over their quadrilateral domain. Because patches often are necessarily tapered to achieve a complete patchwork arrangement over the hull, the spacing between control points can vary considerably across the patch. If sampling points (the input to the fitting algorithm) are taken from an even distribution across the region, this will lead to a scarcity of information where control points are densely packed. This causes the rank deficiency warning, which means that there are many solutions that would all produce a "valid" fit. Just increasing the density of sampling points is not a trustworthy remedy because it also increases data points in the wider areas of the patch, which motivates the algorithm to insert additional rows and columns of control points because that reduces the mean fitting error, and thereby the problem of rank deficiency is kept in place.

We have countered this problem by comparing the lengths of the borders of the region to get a measure of the concentration of control points, so that the spacing of sampling points can be adjusted accordingly across the region. This works well for typical cases where opposite sides of a region have different lengths. But should a region be narrow in the middle and wide along the sides, then this strategy will not be able to prevent rank deficiency. The pragmatic approach is then to have the user split the region in two at the narrow section, which will work around the problem.

But providing appropriate density of sampling points does not prevent all ripple effects. When sampling points happen to align more or less with rows or columns of control points, the fitting algorithm is tempted to decrease the fitting error by adding another row or column in between the sampling points. Because the fitting algorithm has no notion of aesthetics or global smoothness and only aims at minimising the mean fitting error, it is happy to place that row quite a bit off if that produces a better fit at existing points, because there are no points near the new row that would keep it in place. Typi-

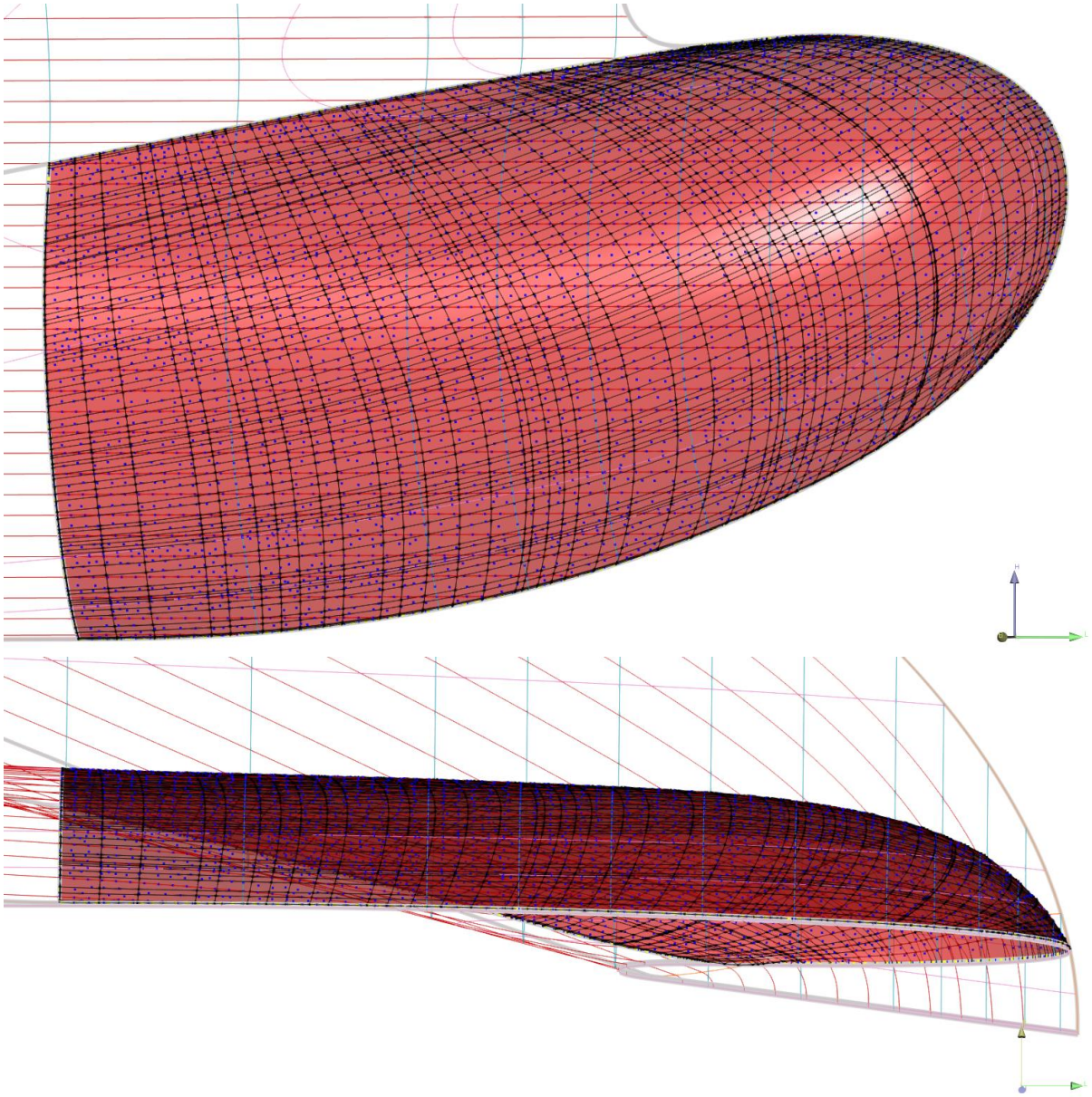

Fig.3: The fitted NURBS patch with sample points,
viewed somewhat from the side (upper figure) and below (lower figure)

cally, this does not produce a single ridge but several ripples running parallel, probably because undulation causes additional rows of control points to be added. Again, increasing the sampling density does not mitigate the effect, it just causes finer and more ripples.

We have found that ripples can be prevented in most cases by adding a random jitter to the parameters at which points are sampled from transfinite patches, so that alignment with control points is greatly reduced. Of course, sample points taken from curves can still align, so it may be necessary for the user to increase sample point density or maybe remove a curve or two from the model if they are not essential for the definition of geometry and are not relevant for construction.

To conclude this section about input to the fitting algorithm, we should mention sampling point weighting. The algorithm allows different weights to be given to individual points, to make some points more important than others. Since curves are the authoritative definition of geometry in Fairway, and the transfinite surface patches that fill the network cells derive their geometry from the curves, it makes sense to give points that are sampled from curves a greater weight than the points that are sampled from patches. That works well if many curves run skewly across the region, which often

is the case, but since only samples from patches can be jittered, giving emphasis to curves can, again, cause ripples to emerge. The same counts for choosing a higher sampling density on curves than in cells. Tweaking these values does affect the balance between success and failure.

Figure 3 shows the same region as Figure 1, with the sample point separation decreased to 0.4m and fit completed. Note the jitter of sample point positions, the increased density of the NURBS control network (in black), and the smoothness of the resulting surface.

7. Performance

There are many steps in the construction of LEANURBS, but the ones that require most time are the search for parameters and the fitting step. These depend heavily on the number of sample points, so that the time it takes to produce a NURBS surface patch grows roughly quadratically with increasing sampling density. This means that, for example, fitting with a separation of 0.01m can take 100 times longer than fitting with a separation of 0.1m. Therefore, we have made this density configurable in the GUI, per region, and the NURBS patch can be generated and previewed per region in the GUI. It is recommend to start out with a large separation, like a tenth or fifth of the longest side of the region, which will produce a patch in less than a second, then assess the result and decrease the separation only when necessary. The largest separation that still produces a good fit depends of course on the internal shape of the region: a completely planar surface produces a perfect fit with a minimum of sampling points. The patch in Figure 3 was produced in less than nine seconds. By the way, the issue of performance speed is not of prime importance, because the LEANURBS method is not applied in the hull design process itself, but only occasionally, when the hull shape is ready to be exported to other software.

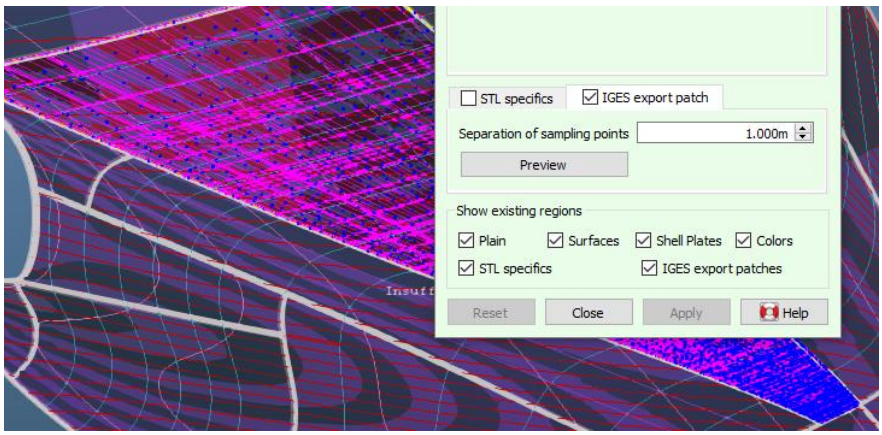

Fig.4: Graphical user interface (GUI) showing sampling density and preview.

LEANURBS typically have a highly non-uniform knot vector, and unevenly spaced rows and columns of control points, which are also high in number. This stems primarily from the many knot refinement steps necessary to obtain compatibility between opposing boundary curves and patch components, which is inevitable for the elimination of gaps. The number of control points is additionally increased by the fitting algorithm. We have not found that receiving systems have a problem with these patches, but it does mean that they are badly suitable for interactive design by control point manipulation — but that should hardly come as a surprise.

Users should be aware that the fitting algorithm can only add whole rows and columns of control points that run across the entire patch, even when their added control is only required in one area of the region. So a sensible partition of the shell that takes this mechanism into account can help to prevent excessive numbers of control points, which of course also affects the performance of the fitting algorithm both in speed and quality.

There are two other problems that can arise from an unfortunate surface partition. Firstly, there may be local minima in the search for parameters (u, v) if the surface shape varies wildly within the region. This means that a solution cannot be found. This is of course detectable so that the corresponding sample point can be discarded, and a satisfactory fit may still be obtainable. Secondly, problems will arise if the orthogonal vectors originating from the base surface have more than one intersection with the Fairway surface. This is not very likely but if it occurs it will cause a discontinuous parameterisation which will definitely produce a distorted NURBS patch shape. Usually, these problems do not arise in typical shell partitions, assuming partitioning by a diligent user, and are easily detected visually in the preview.

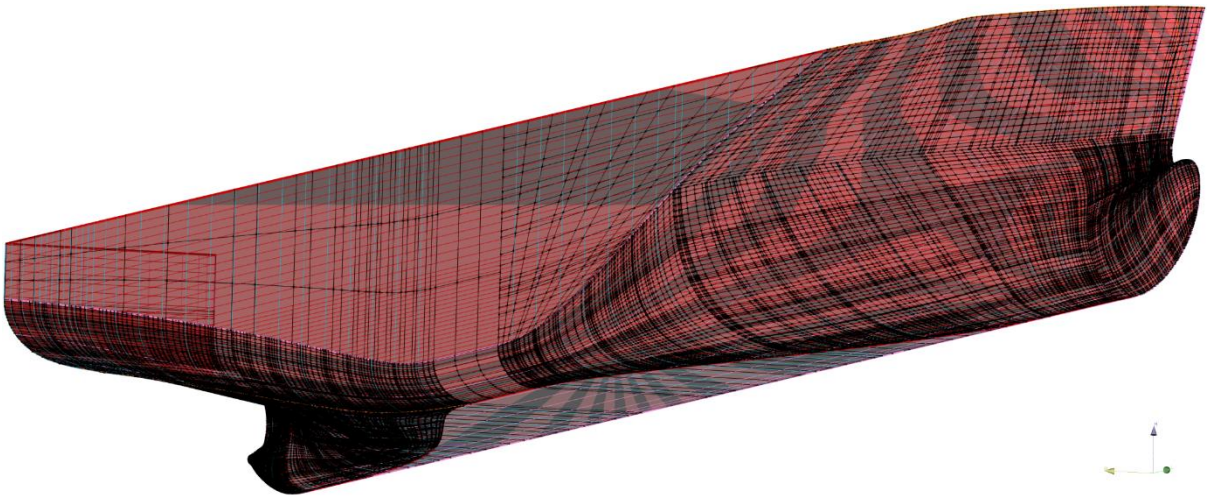

Fig.5: The "Duisburg Test Case" completely described by LEANURBS patches.

8. Applications and future work

Let it be clear: even with this considerable investment in NURBS surface patches, we are not going to implement NURBS surface editing functionality in Fairway. What we have illustrated is that one can trade a high number of small patches for a low number of large patches, but at the price of a high number of control points, making editing impractical. Whenever a designer has the desire to control a Fairway surface by means of control points, we offer a better method, developed in *Veelo (2004),* called spatial deformation based on radial functions, that even gives control over the extent of the surface on which the manipulation has effect. That is a flexibility that cannot be offered by pure NURBS modellers or subdivision surface modellers.

Obviously, the main application of LEANURBS is export of Fairway geometry in standard exchange formats such as IGES. In addition, we are currently in collaboration with Friendship Systems to develop an exchange protocol utilising LEANURBS to establish a lossless exchange of hull geometry back and forth between Fairway and CAESES, which is a flexible CAD and optimization platform for fast and comprehensive design studies with simulation tools. Figure 6 shows the generated LEANURBS as imported by CAESES. Due to the absence of gaps, CAESES had no problem generating a watertight mesh.

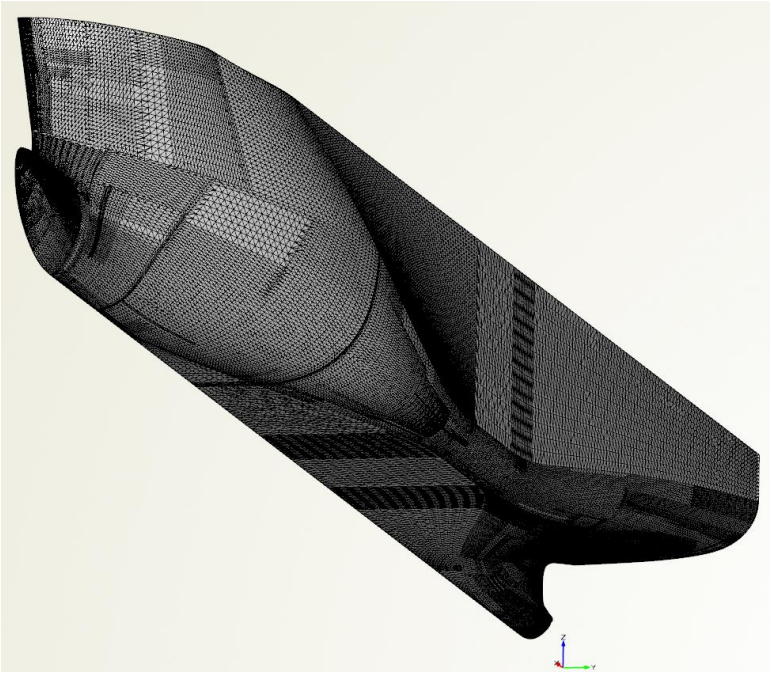

Fig.6: LEANURBS imported in CAESES. Courtesy Friendship Systems.

Whereas we currently rely on manual partitioning by the user, *Ardavani (2017)* has shown that this task can largely be automated by application of a genetic algorithm. Proof-of-concept code exists, but has yet to be brought in production. One possible application could be to have the algorithm propose a partitioning, that subsequently can be adjusted or perfected by the user.

9. Conclusion

Although it is widely acknowledged that the NURBS surface method is less suitable for hull shape design and modelling, it keeps on haunting us: With PIAS/Fairway a better, non-NURBS surface method is available, but still other software suites, such as for CFD, FEM or engineering, require their import data to be in NURBS format. Simple conversion from the Fairway surface patches to NURBS surface patches has been available for years, but induced some problems, such as the resulting large number of NURBS patches. In order to provide a solution — and hence solve a problem that was introduced by inappropriate choices of others in the first place — a cunning framework of geometric modelling tools has been constructed, which converts a Fairway hull representation to a set of NURBS surfaces that are watertight, and limited in number.

References

ARDAVANI, M. (2017), *Modeling and Simulation of Ship Shape Topology*, Master thesis UvA-VU

DIERCKX, P. (1993), *Curve and Surface Fitting with Splines*, Monographs on Numerical Analysis, Oxford University Press (Fortran code: `www.netlib.org/dierckx`)

KOELMAN, H.J. (1997), *Hull form design and fairing: tradition restored*, 6th Int. Marine Design Conf., Newcastle upon Tyne, pp.421–428

KOELMAN, H.J. (1999), *Computer Support for the Design, Engineering and Prototyping of the Shape of Ship Hulls*, Doctoral thesis TUDelft

KOELMAN, H.J.; VEELO, B.N. (2013), *A technical note on the geometric representation of a ship hull form*, Computer-Aided Design 45/11, Elsevier, pp.1378–1381

MARINIĆ-KRAGIĆ I.; ĆURKOVIĆ, M.; VUČINA, D. (2018), *Adaptive re-parameterization based on arbitrary scalar fields for shape optimization and surface fitting*, Eng. Appl. of Artificial Intelligence 67, Elsevier, pp.39–51

MOKTAR, EL, A.; SHIGUNOV, V.; ZORN, T. (2012) *Duisburg Test Case: Post-Panamax Container Ship for Benchmarking*, J. Ship Technology Research 59/3, pp.50–64

RAVI KUMAR, G.V.V.; SHASTRY, K.G.; PRAKASH, B.G. (2002), *Computing non-self-intersecting offsets of NURBS surfaces*, Computer-Aided Design 34, Elsevier, pp.209–228

PIEGL, L.; TILLER, W. (1997), *The NURBS Book*, Monographs in Visual Communication, 2nd ed., Springer

PIEGL, L.; TILLER, W. (1999), *Computing offsets of NURBS curves and surfaces*, Computer-Aided Design 31, Elsevier, pp.147–156

SHARMA, R.; KIM, T.-W.; STORCH, R.L.; HOPMAN, J.J.; ERIKSTAD, S.O. (2012), *Challenges in computer applications for ship and floating structure design and analysis*, Computer-Aided Design 44/3, Elsevier, pp.166–185

VEELO, B.N. (2004), *Variations of Shape in Industrial Geometric Models*, Doctoral thesis NTNU 2004:106, hdl.handle.net/11250/241067

ZHANG, Y.; CAO, J.; CHEN, Z.; LI, X.; ZENG, X.-M. (2016), *B-spline surface fitting with knot position optimization*, Computers & Graphics 58, Elsevier, pp.73–83