Comparing Shape in Biological Objects

The University of Manchester

Department of Mathematics

Rudi Agnew

2022

Contents

Introduction

The aim of this project was to reformulate an interpolation technique called the thin-plate spline (TPS) and use it to analyse change in biological forms via Cartesian grid transformations as seen in D'arcy Thompson's On Growth and Form [10]. We shall rederive the systems of equations that generate the TPS from the physical idea of a thin sheet of metal being bent by point forces at certain locations with specific magnitudes. In this project we will use TPS to interpolate homologous points or landmarks between two biological forms as a deformation, we shall apply ideas from continuum mechanics to characterise this deformation and try to explain why certain areas become more or less deformed under the transformation given by the TPS. We shall also compare D'arcy Thompson's Cartesian grid transformations to our own TPS generated grid transformations and highlight similarities and differences.

Chapter 1

Biological background and motivations

1.1 The history of morphometrics

One of the most fundamental and interesting properties of biological objects, whether that be singular cells or whole organisms, is the ability to change shape; allowing for a more complex and efficient structure. The study in which the difference of these forms are quantified is called morphometrics. One of the earliest and most influential ideas in this area was written by D'Arcy Wentworth Thompson in his seminal 1917 book *On Growth and Form* [10]. The book focused on different aspects than typical Darwinian approaches of the time. Thompson, annoyed at the 'Just so' explanations of morphology given by Darwinians [1], argued that physical forces were the main driving factor in the shape of objects in nature. He was more focused on this physical aspect and explaining how change in shape happened as opposed to evolution approach and explaining why it happened.

The most famous section of the book is the last chapter, *On the Theory of Transformations, or the Comparison of Related Forms*, where Thompson explored how the forms of organisms can be explained by geometrically transformations. Inspired by Albrecht Dürer's work on human proportion [5], Thompson compares different organisms via Cartesian transformations, such as a simple shear seen in Figure 1.1, which resulted in the iconic images that made the book famous. This article [9] goes into more detail on the validity of Thompson's methods and conclusions whereas we shall generate our own Cartesian grid transformations via a different method and compare our grids with Thompson's directly.

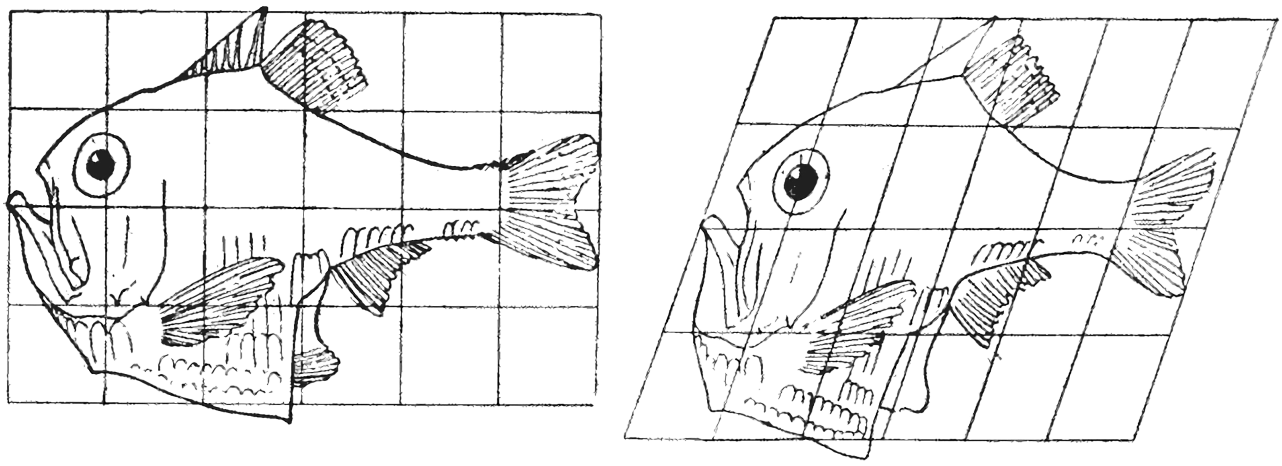


Figure 1.1: An example of D'arcy Thompson's Cartesian grid transformations.

1.2 Landmark based morphometrics

To be specific, morphometrics is the study of difference in shape of related forms. These relations are defined by homologies. A homologous feature shared between organisms is a similarity which is due to a shared ancestry, for example most insects have two pairs of wings. To compare such morphologies there are three distinct styles of morphometrics. Those are traditional, landmark based and outline based morphometrics, the latter two come under the general guise of geometric morphometrics. Traditional morphometrics which deals with physical linear measurements of specimens such as bone shaft diameters and wing span [7]. This technique is generally useful when considering the specimen on a whole, for example comparing growth rates over time. A big advantage of this method is its simplicity, however its major disadvantage is that these linear measurements of distance are often correlated to one another, which gives less data to analyse.

Landmark based morphometrics is what we shall discuss in this paper. A landmark is a part of a biological object that can be considered biologically meaningful. Landmarks represent homologies, that is, points or areas or curves that are unequivocally similar in the specimens being compared. Examples of these are the dorsal fins which can be found on most marine animals and the wings of insects as discussed earlier. Combining multiple landmarks on one species can build a picture of that particular specimen. This can then be compared to other similar species with the same landmarks, seeing how these landmarks have moved around, altering shape. To implement this landmark analysis we need create a mapping between forms whilst preserving landmarks. This is done by interpolation as the landmarks act as the points we wish to interpolate and then the mapping between forms is the interpolant. It is impor-

tant to note that change in form does not just occur from species to species via evolution but through developmental biology. For example, this could be going from a juvenile to an adult or a cell developing through morphogensis. These processes can also be described and helped understood by the techniques discussed in this project.

Chapter 2

Thin plate spline interpolation

We want a way to understand and quantify the difference in form of biological objects using landmark based morphometrics. The focus of this paper will be on the specific interpolation technique used in this quantification. The aim of our interpolant is to act as a mapping between landmarks of specimens. We want this mapping to mimic evolution and be biologically viable, that is, be gradually changing. In mathematical terms we want it to be smooth and continuous with the change in shape of the specimens being minimal. We don't want to restrict ourselves to a small number of landmarks to compare, so our data set will be large. Therefore We will discuss a particular variant of spline interpolation over polynomial to avoid large oscillations, given by Runge's phenomenon, when dealing with high numbers of landmarks.

2.1 Basic spline interpolation

Spline interpolation is specific type of interpolation which uses a concatenation of polynomials to interpolate data points or 'knots'. These piecewise polynomials are called splines. This is different to polynomial interpolation which uses one high degree polynomial that fits all the data points. The basics of spline interpolation are touched on briefly below. This builds up to the particular spline we shall use, the thin plate spline (TPS).

2.1.1 Linear and cubic Splines

The most simple continuous form of interpolating data points is 1D linear interpolation. For two data points (x_0, y_0) and (x_1, y_1) the linear interpolant is just the straight line between them $y = y_0 + (x - x_0)(y_1 - y_0)/(x_1 - x_0)$. This process is then repeated for all the data points and

the collection of straight line splines are then joined together to form a curve going through all the data points. This curve is continuous but not in general differentiable, specifically at the data points. To introduce differentiability higher degree polynomials are used instead of linear functions, namely cubic splines.

For n data points (x_i, y_i), where $i \in [0, n-1]$, we define a cubic polynomial $g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$. The collection of such polynomials will form the continuous curve that passes through all the data points is differentiable everywhere and this time we can impose continuous derivatives at the data points, so the curve is smooth. The problem is to find the constant unknowns which amounts to $4n$ parameters to determine. We can find $2n$ of these by requiring the splines actually pass through the data points, that is

$$g_i(x_i) = y_i, \quad g_i(x_{i+1}) = y_{i+1}, \quad 0 \leq i \leq n-1$$

and the continuity condition in the first and second derivatives is

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1}), \quad g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}), \quad 0 \leq i \leq n-2$$

which gives $4n - 2$ total conditions. There are a few options for the final two conditions. A few popular choices are the 'not a knot' spline which involves demanding continuity at x_0 and x_{n-1} in the third derivative, the 'natural' spline which is setting the end point second derivatives to zero and the 'clamped' spline where the first derivative is set to a specific value at the end points. Thus the cubic spline problem has a closed form solution, $4n$ unknowns and $4n$ equations. See [8] for full system of equations.

Figure 2.1 shows a comparison between the two types of spline interpolation discussed above. The smoothness property of the cubic spline is apparent and this is an important element of what we want for our landmark mappings. We can consider a simplification of specimens we want to analyse and imagine them as laying flat in the plane and ignore the third spatial dimension. Hence we want an interpolant that depends on both the x and y coordinates. The TPS is a natural extension of the cubic into 2 dimensions which is what we require.

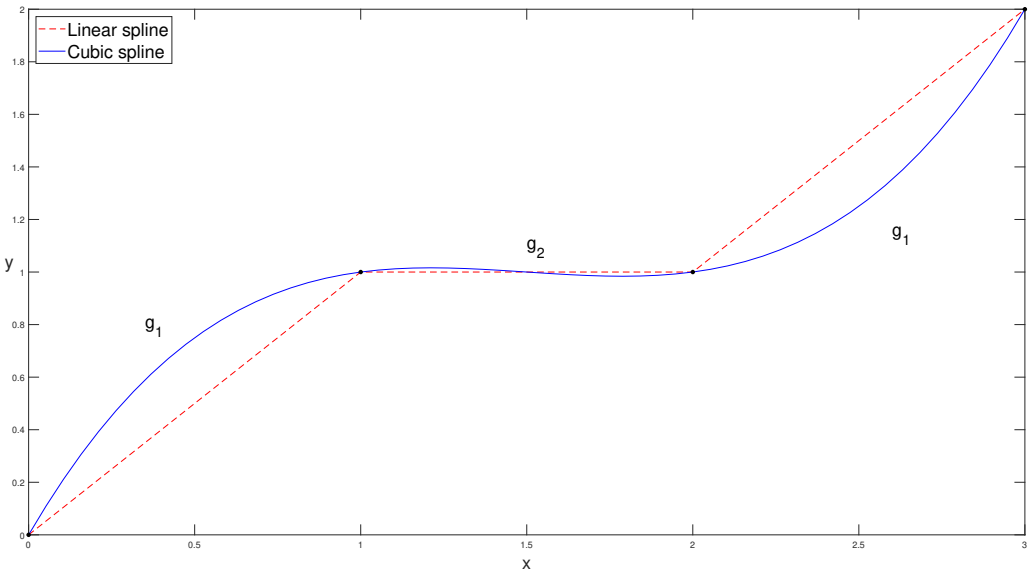


Figure 2.1: Comparison between linear and not a knot cubic splines for a set of data points (0,0), (1,1), (2,1), (3,2).

2.2 Physical intuition behind the TPS

The physical motivation driving the idea of the TPS is that of a flat, thin, infinite, isotropic and homogeneous sheet of metal embedded in $\mathbb{R}^3$ but lying flat in the plane $(x, y) \in \mathbb{R}^2$. A thin sheet of metal has an aspect of rigidness and resistance to deformation, so when such a deformation occurs there is no extreme change to the structure of the sheet. Instead it deforms gradually and in a smooth manner. These are exactly the properties we require for our interpolant. To see why the plate could act as an interpolant we can imagine these deformations occurring normal to the plane in the z direction. If we can control the exact magnitude of these bends we then have the plate passing through specific points and thus interpolating them whilst retaining all the previous desired properties. These specific points will be the landmarks discussed in Chapter 1. We can also interpret this idea as a minimisation problem, specifically minimising the bending energy of the plate. This gives further physical motivation for the TPS and is discussed further in section 2.4. To turn this physical picture into mathematics we need a few key ideas.

2.3 Mathematical framework

2.3.1 The Dirac delta function

To deform a metal sheet we need a way of applying a force to it. To do this let us introduce the Dirac delta function. The Dirac delta function $\delta(x)$ is a so called generalised function whose value is 0 everywhere except for at $x = 0$ where its value is, loosely speaking, infinite. The delta function, in spatial terms, models a surface force that occurs at an area that is taken to be infinitely small, a point force. For example, if we punch a metal plate we can think of the area this punch was applied to as getting smaller and smaller. The point force given by this punch is Dirac delta function. The infinite part of the delta function is where this impact occurs, it is zero everywhere else representing the idea that the force occurs at one specific point. Hence we define the delta function as follows

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{2.1}$$

and we also require that the delta function integrates to one when crossing the point force, note that the delta function integrates to zero if the domain does not contain this point.

$$\int_{-\infty}^{\infty} \delta(x)dx = 1. \tag{2.2}$$

Using this property we can think of the delta function as a distribution with standard deviation approaching 0 as seen in figure 2.2.

The delta function also has the algebraic property that it can be shifted, and in doing so picks out a certain value of a function. Note that this is a more general definition of the delta function. The delta function is also the derivative of the Heaviside function.

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0). \tag{2.3}$$

2.3.2 The biharmonic equation

In continuum mechanics, specifically plate theory, the biharmonic is used to model thin structures that react elastically to 'pure' bending, that is, a bending moment applied without a simultaneous shear force. This is equation is ideal for modelling our thin metal sheet idea. For a displacement field $u(x, y)$, the biharmonic satisfies

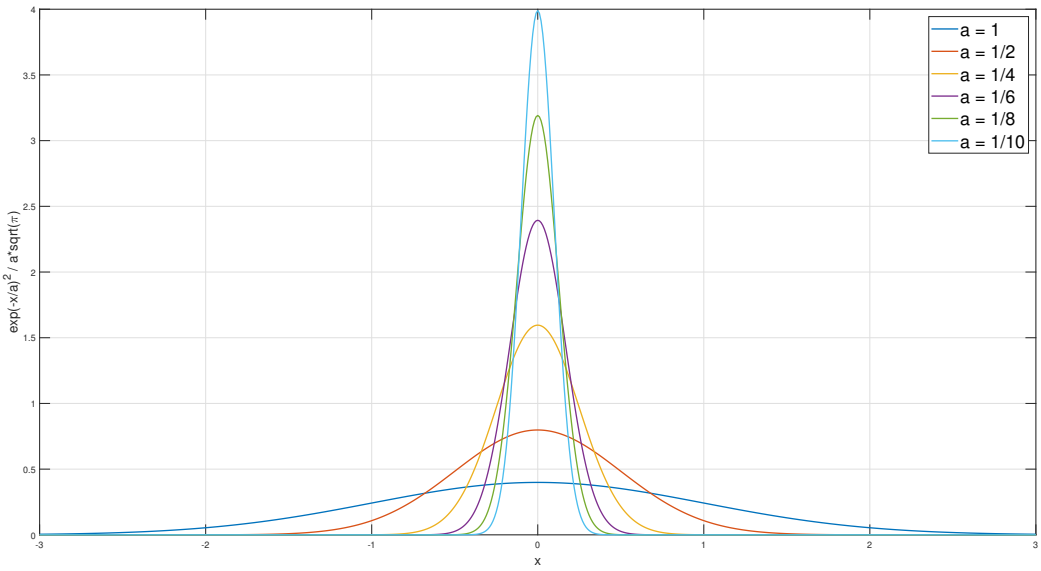


Figure 2.2: Graph showing a comparison of normal distribution probability density functions of mean zero with decreasing standard deviation a. In the limit as $a \to 0$ we obtain a Dirac delta function.

$$\nabla^4 u(x, y) = 0 \tag{2.4}$$

and upon expanding we get

$$\frac{\partial^4 u}{\partial x^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} + \frac{\partial^4 u}{\partial y^4} = 0. \tag{2.5}$$

To complete our thin plate spline picture, we must now solve for a Greens function $u(x, y)$ that satisfies

$$\nabla^4 u(x, y) = \delta(\boldsymbol{x} - \boldsymbol{x_0}), \quad \boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \boldsymbol{x_0} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{2.6}$$

where the domain is $\boldsymbol{x} \in \mathbb{R}^2$ and $\boldsymbol{x_0}$ is the location our point force is being applied. This is the so called fundamental solution of the biharmonic operator or Greens function, and represents the deflection of a plate due to a point force applied at (x_0, y_0).

2.3.3 The fundamental solution of the biharmonic

To solve this first note that for $u = u(x, y)$ we have $\nabla^4 u = \nabla^2(\nabla^2 u)$. Using the fact that the Dirac delta function integrates to 1, equation (2.6) reduces to

$$1 = \iint_D \nabla^2 \left(\nabla^2 u\right) dA = \iint_D \nabla \cdot \left(\nabla \left(\nabla^2 u\right)\right) dA \tag{2.7}$$

where we are integrating over some region $D \subset \mathbb{R}^2$. We can now apply the divergence theorem to transform this from a surface to a line integral

$$1 = \iint_D \nabla \cdot \left(\nabla \left(\nabla^2 u\right)\right) dA = \oint_{\partial D} \nabla \left(\nabla^2 u\right) \cdot \boldsymbol{n} dS. \tag{2.8}$$

We now assume that u is radially symmetric and transform to planar polar coordinates. This results in

$$\oint_{\partial D} \nabla \left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right) \cdot \boldsymbol{n} dS = 1 \tag{2.9}$$

where r is the radial coordinate and θ would be the polar angle but it doesn't appear in (2.9) as we assumed u was dependent on r only. Restricting this integral to a small circle $S = \varepsilon\theta$, ε representing the radius of S, implies that $dS = \varepsilon d\theta$. Since u is a function of r alone, the Laplacian simplifies to the partial derivative with respect to r. We can now directly integrate which results in the following derivation

$$\Rightarrow \quad \int_0^{2\pi} \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right) \varepsilon d\theta = 1.$$

$$\Rightarrow \quad \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right)\Bigg|_{r=\varepsilon} \int_0^{2\pi} \varepsilon d\theta = 1.$$

$$\Rightarrow \quad \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right)\Bigg|_{r=\varepsilon} = \frac{1}{2\pi\varepsilon}$$

$$\Rightarrow \quad \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right) = \frac{1}{2\pi r}$$

$$\Rightarrow \quad \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) = \frac{1}{2\pi}\ln(r) + k_0$$

$$\Rightarrow \quad \frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) = \frac{r}{2\pi}\ln(r) + k_0 r, \quad \text{then by parts}$$

$$\Rightarrow \quad r\frac{\partial u}{\partial r} = \frac{r^2}{4\pi}\ln(r) - \frac{r^2}{8\pi} + \frac{k_0 r^2}{2} + k_1$$

$$\Rightarrow \quad \frac{\partial u}{\partial r} = \frac{r}{4\pi}\ln(r) - \frac{r}{8\pi} + \frac{k_0 r}{2} + \frac{k_1}{r}$$

$$\Rightarrow \quad u = \frac{r^2}{8\pi}\ln(r) - \frac{r^2}{16\pi} - \frac{r^2}{16\pi} + \frac{k_0 r^2}{4} + k_1\ln(r) + k_2$$

$$\Rightarrow \quad u = \frac{r^2}{8\pi}(\ln(r) - 1) + \frac{k_0 r^2}{4} + k_1\ln(r) + k_2$$

where k_0, k_1 and k_2 are real constants. We want our solution to have no singularities so we set k_1 to be 0. As u tends to infinity it is the $r^2 \ln r$ term that dominates so we set k_0 to $1/4\pi$ so it absorbs the other constant term, we shall also set k_2 to be zero and disregard the $1/8\pi$ out the front. The justification for doing this, and whether it is the correct thing to do or not, shall be discussed in a later section. Hence we obtain the fundamental solution of the biharmonic equation

$$u(r) = r^2 \ln r = \frac{1}{2} r^2 \ln r^2. \tag{2.10}$$

The negative of this solution, $-u(x, y) = -r^2 \ln r$, is displayed in 2.3. This represents the thin metal sheet being bent by a point force applied at (0,0).

2.3.4 Linear combinations of $u(x, y)$

We want the thin metal sheet to pass through specific points so that it will act as an interpolant between these points. Due to our construction and the governing equation of the plate, this interpolation will be as smooth and continuous and discussed earlier. This is done by constructing n point forces on our thin metal sheet using linear combinations of functions like $u(x, y)$. These point forces will cause the plate to pass through the points we want to interpolate. To control

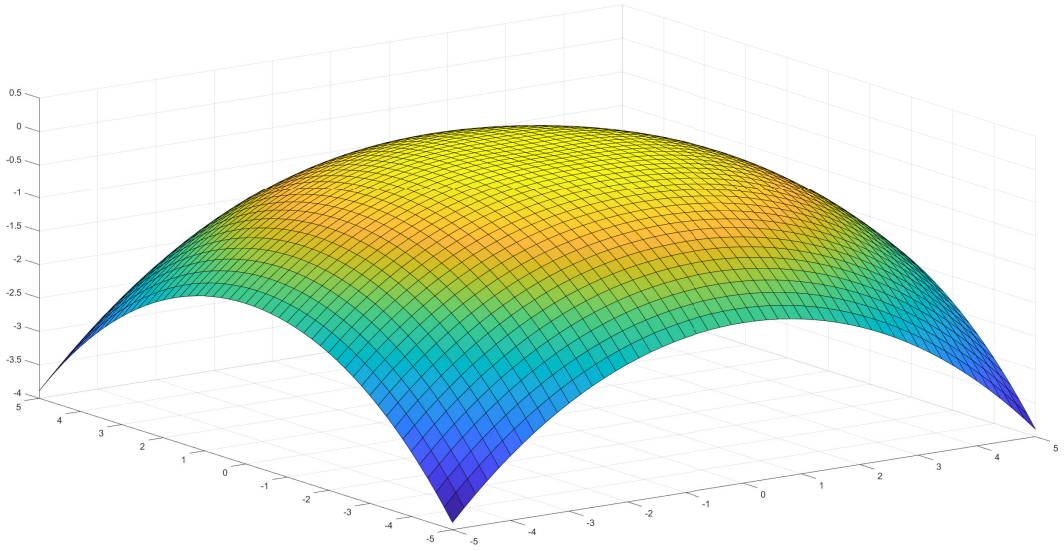


Figure 2.3: The fundamental solution of the biharmonic.

where the points are we adjust the magnitudes of the point forces by adjusting the constants in front of the delta functions. Here we see why we could ignore the $1/8\pi$ in our fundamental solution and is just gets absorbed into the weights. These are referred to as weights. Starting simple with $n = 2$ we have the following setup. We want to solve the equation

$$\nabla^4 z(x, y) = w_1 \delta(\boldsymbol{x} - \boldsymbol{x_1}) + w_2 \delta(\boldsymbol{x} - \boldsymbol{x_2}) \tag{2.11}$$

where $\boldsymbol{x} \in \mathbb{R}^2$ are points in the plane, $\boldsymbol{x_i} \in \mathbb{R}^2$ are the locations where the point forces are being applied and $z(x, y)$ is the displacement orthogonal to the plane induced by these forces. We want to adjust the weights w_1 and w_2 such that the plate passes through the points we wish to interpolate, namely z_1 and z_2. The picture to imagine is displayed in figure 2.4.

We already have the solution to 2.11, namely 2.10, only now we are using a linear combination of this solution. Define $u_i(x, y) = |(\boldsymbol{x} - \boldsymbol{x_i})|^2 \ln |(\boldsymbol{x} - \boldsymbol{x_i})|^2$ and the equation of the thin plate is

$$z(x, y) = w_1 u_1(x, y) + w_2 u_2(x, y). \tag{2.12}$$

We also have the conditions given by the delta functions which allows us to solve for the weights

$$\begin{cases} z_1 = z(x_1, y_1) = w_1 u_1(x_1, y_1) + w_2 u_2(x_1, y_1), \\ z_2 = z(x_2, y_2) = w_1 u_1(x_2, y_2) + w_2 u_2(x_2, y_2). \end{cases} \tag{2.13}$$

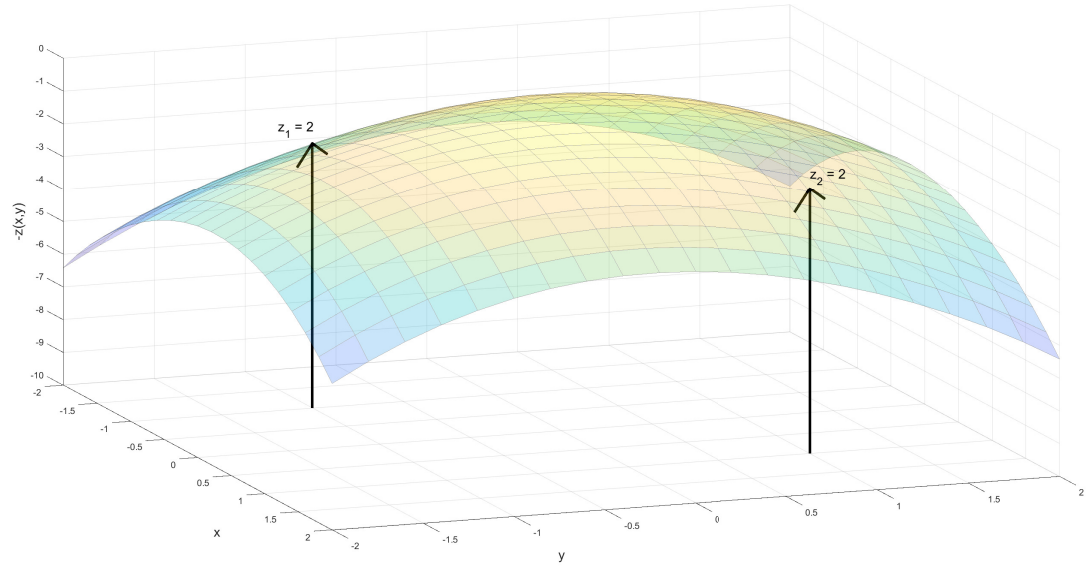


Figure 2.4: Figure illustrating two point forces acting on a thin metal sheet.

This is a linear system of equations and can be written in matrix form. Notice that $u_i(x_i, y_i) = |(\boldsymbol{x_i} - \boldsymbol{x_i})| \ln |(\boldsymbol{x_i} - \boldsymbol{x_i})| = 0$, note that this technically a limiting value as $\ln(0)$ is undefined. We are looking at the limit as $x \to x_i$ here, not simply evaluating the function. Hence we obtain

$$\begin{pmatrix} 0 & u_2(x_1, y_1) \\ u_1(x_2, y_2) & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}. \tag{2.14}$$

Also notice that $u_1(x_2, y_2) = |(\boldsymbol{x_2} - \boldsymbol{x_1})|^2 \ln |(\boldsymbol{x_2} - \boldsymbol{x_1})|^2 = |(\boldsymbol{x_1} - \boldsymbol{x_2})|^2 \ln |(\boldsymbol{x_1} - \boldsymbol{x_2})|^2 = u_2(x_1, y_1)$ so our matrix is hollow (diagonal values are all zero) and symmetric. Due to the simplicity of the system, it is easily solvable by hand and we obtain the solution to our plane

$$z(x, y) = \frac{z_1}{u_2(x_1, y_1)} + \frac{z_2}{u_1(x_2, y_2)} \tag{2.15}$$

Using the values $(x_1, y_1) = (-1, -1)$, $z_1 = 2$ and $(x_2, y_2) = (1, 1)$, $z_2 = 2$ we obtain the surface seen in figure 2.4. Of course, we want to generalise and interpolate between n points. The equation of the plane becomes

$$z(x, y) = w_1 u_1(x, y) + w_2 u_2(x, y) + \cdots + w_n u_n(x, y) = \sum_{i=1}^{n} w_i u_i(x, y). \tag{2.16}$$

For every new point force we add we gain another condition for the weights

$$\begin{cases} z_1 = z(x_1, y_1) = w_1 u_1(x_1, y_1) + w_2 u_2(x_1, y_1) + \cdots + w_n u_n(x_1, y_1), \\ z_2 = z(x_2, y_2) = w_1 u_1(x_2, y_2) + w_2 u_2(x_2, y_2) + \cdots + w_n u_n(x_2, y_2), \\ \vdots \\ z_n = z(x_n, y_n) = w_1 u_1(x_n, y_n) + w_2 u_2(x_n, y_n) + \cdots + w_n u_n(x_n, y_n). \end{cases}$$

This can be written in matrix form as

$$\begin{pmatrix} u_1(x_1, y_1) & u_2(x_1, y_1) & \cdots & u_n(x_1, y_1) \\ u_1(x_2, y_2) & u_2(x_2, y_2) & \cdots & u_n(x_2, y_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(x_n, y_n) & u_2(x_n, y_n) & \cdots & u_n(x_n, y_n) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \tag{2.17}$$

Note that we have the generalisation of earlier noticed properties

$$u_i(x_i, y_i) = |(\boldsymbol{x_i} - \boldsymbol{x_i})|^2 \ln |(\boldsymbol{x_i} - \boldsymbol{x_i})|^2 = 0 \tag{2.18}$$

$$u_i(x_j, y_j) = |(\boldsymbol{x_i} - \boldsymbol{x_j})|^2 \ln |(\boldsymbol{x_i} - \boldsymbol{x_j})|^2 = |(\boldsymbol{x_j} - \boldsymbol{x_i})|^2 \ln |(\boldsymbol{x_j} - \boldsymbol{x_i})|^2 = u_j(x_i, y_i) \tag{2.19}$$

So our matrix is symmetric and hollow as seen before. To recap, so far we have a method for determining the equation of a thin metal plate that acts as a smooth interpolant for n points in 3 dimensional space.

2.3.5 The TPS as a natural extension of the cubic spline

Here we discuss why the TPS is the natural extension of the 'natural' cubic spline, an idea briefly mentioned in section 2.1.1. Looking at the example in figure Figure 2.1, but this time using a natural spline instead, at the 'knots' we require the two meeting splines are continuous functions and are continuous in their first and second derivatives. Since the splines are cubics their second derivatives will be straight lines, so at the knot we have two straight lines meeting. If we then take the 3rd derivative we have two constant lines with a jump discontinuity at the knot, this is an example of the Heaviside function. If we then take the fourth derivative we obtain delta functions at each of the knots. So another formulation of the cubic spline is to solve $\nabla^4 f(x) = \sum_i^n \delta(x - x_i)$. Upon extending $f(x)$ to a function of both x and y we obtain the approach taken in the formulation of the TPS.

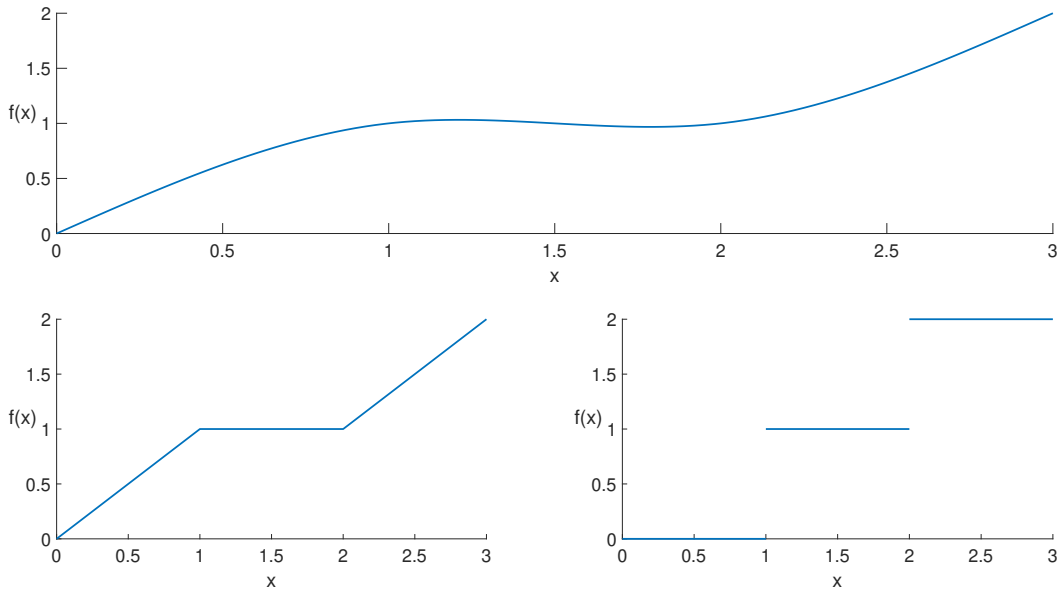


Figure 2.5: Showing the natural cubic spline (1) with its first (2) and second (3) derivatives.

2.4 Bending Energy

A natural question to ask is why does the biharmonic represent a thin plate and how does it give it its characteristic bending resistance. This is due to (2.10) minimising the so called bending energy of a thin plate which is defined to be

$$E = \int_{\mathbb{R}^2} \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y} \right)^2 \right) dxdy \tag{2.20}$$

where f is an arbitrary function of (x, y). To show that (2.16) does indeed minimise this bending energy we need to find the particular f which makes (2.20) stationary, this will turn out to be (2.10). To prove this we need to use the calculus of variations.

2.4.1 Calculus of variations and the bending energy functional

We introduce the bending energy as a functional

$$I[f] = \int_{\mathbb{R}^2} F\left(f_{xx}(x,y), f_{xy}(x,y), f_{yy}(x,y)\right) dxdy \tag{2.21}$$

where F is the integrand from 2.20, note that this functional is a purely a function of the second derivatives of $f(x, y)$. We want to minimise this functional which is equivalent to finding stationary points, hence we can apply the appropriate form of the Euler-Lagrange equation,

see [4] for details. We have

$$\frac{\partial F}{\partial f} - \frac{\partial}{\partial x}\left(\frac{\partial F}{\partial f_x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial F}{\partial f_y}\right) + \frac{\partial^2}{\partial x^2}\left(\frac{\partial F}{\partial f_{xx}}\right) + \frac{\partial^2}{\partial x \partial y}\left(\frac{\partial F}{\partial f_{xy}}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial F}{\partial f_{yy}}\right) = 0$$

and using the fact that (2.21) is a function of second derivatives only we get

$$\frac{\partial^2}{\partial x^2}\left(2\frac{\partial^2 f}{\partial x^2}\right) + \frac{\partial^2}{\partial x \partial y}\left(4\frac{\partial^2 f}{\partial x \partial y}\right) + \frac{\partial^2}{\partial y^2}\left(2\frac{\partial^2 f}{\partial y^2}\right) = 0.$$

Dividing out by the constant factor of 2 this reduces to

$$\Delta^2 f = \nabla^4 f = 0$$

which is the biharmonic equation. So we have seen solving this bending energy minimisation problem is equivalent to solving the biharmonic which adds to the physical motivations behind modelling the thin plate in this way.

2.5 2-D Interpolation using the TPS

We can look at the power of this interpolation method by observing some of the surfaces it generates. As usual, the physical interpretation here is a thin infinite metal plate with point forces being applied to it. These point forces keep the plate in place at the points we want to interpolate. This is all encapsulated in (2.16) and (2.17). Figures Figure 2.6 and Figure 2.7 show the side-on and birds-eye view respectively of a surface created by the process described above. In this example we are still only interpolating 6 points so the surface is still rather simple. We can see the smoothness property of $z(x, y)$, which is given by the bending resistance of the thin plate, by looking at the two closest points which have been pushed in opposite directions. The gradient at this point is large in magnitude and hence we get a peak and a trough forming. The next points the surface needs to reach are again in opposite in magnitude to the peak and trough respectively but due to smoothness we get a slow decrease from the peak and a slow increase from the trough as the derivatives change as slow as possible. If we didn't have this smoothness property we would see rapid change in derivatives and jagged edges as a result.

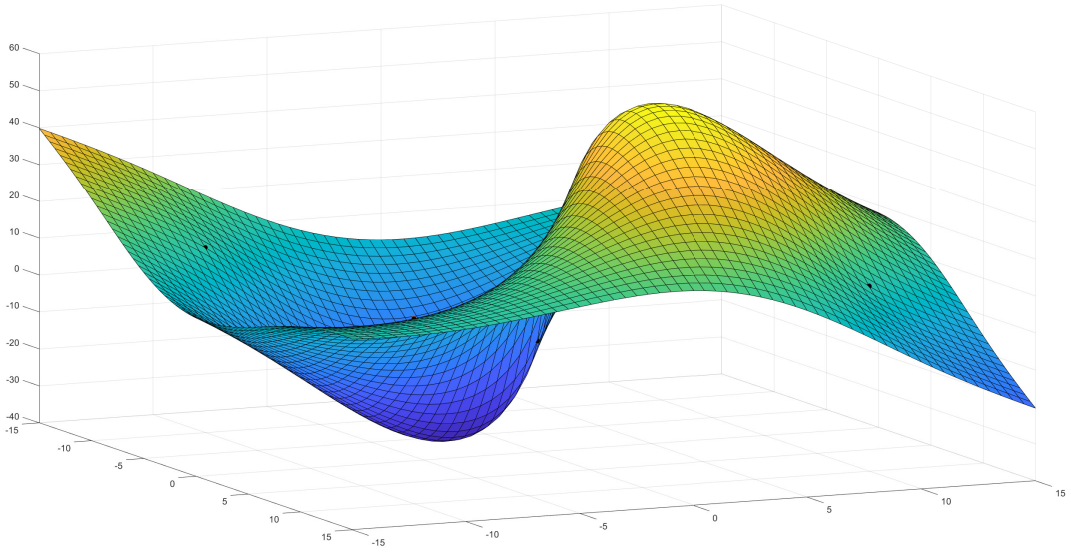


Figure 2.6: Side on view of a thin metal plate.

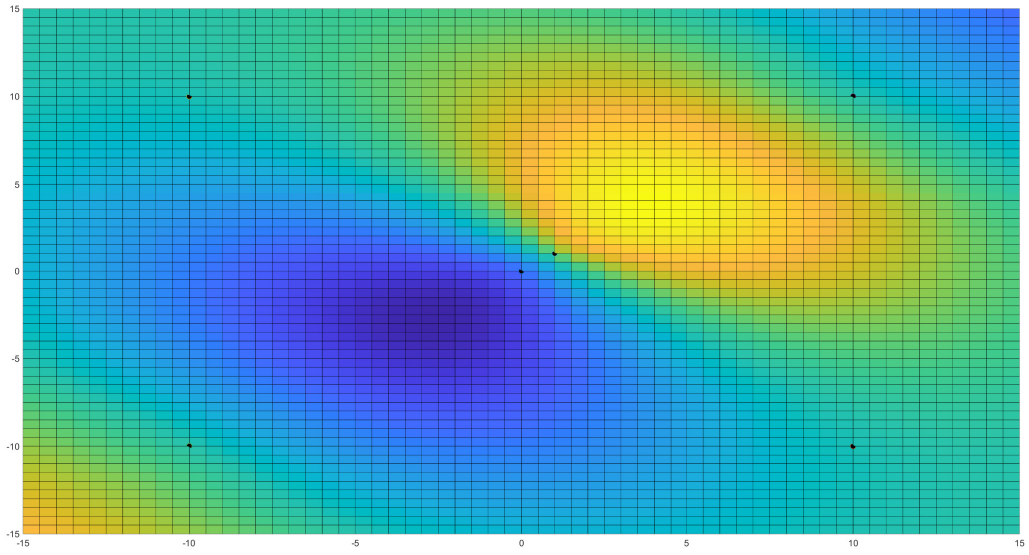


Figure 2.7: Birds-eye view of 2.6.

If we now look at 100 points, with the (x, y) values and desired heights sampled from a normal distribution we obtain a height map generated by the TPS shown in Figure 2.8.

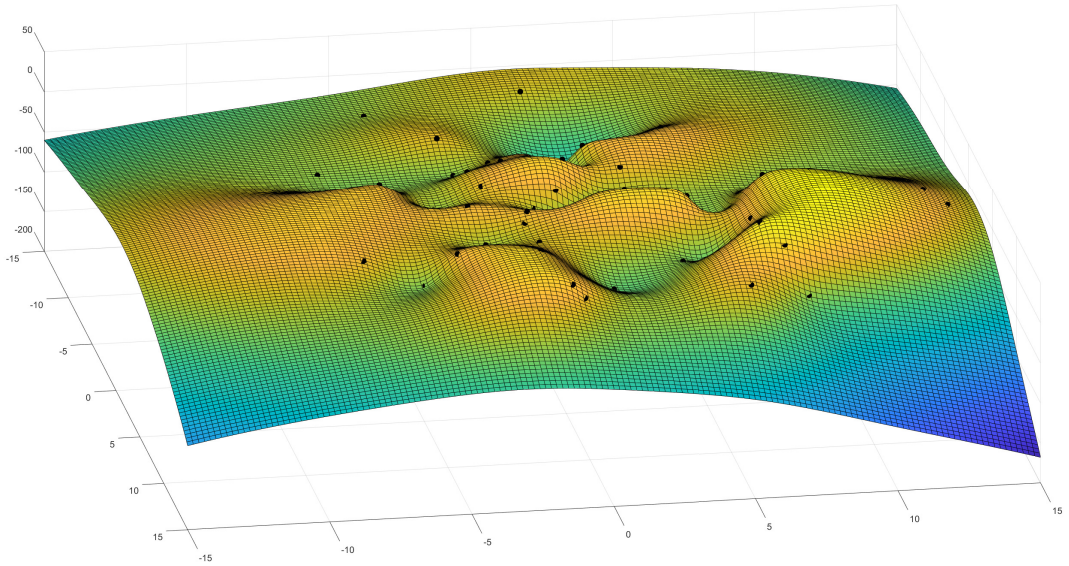


Figure 2.8: Height map.

As we increase the number of interpolated points the matrix (2.17) increases as well. To calculate the equation of the surface (2.16) we need to invert this matrix. Hence the spline becomes more costly to compute as a function of the number of interpolants. This will be discussed more in later chapters. If we try one set of (x, y) coordinates to different heights the matrix becomes singular which is to be expected as this is equivalent to requiring 2 different values for the point force at one specific location which makes no physical sense.

Chapter 3

Thin plate spline as a deformation map

3.1 Deformation mappings

Up until now we have just seen the TPS as a method to generate a smooth interpolating surface between points in $\mathbb{R}^3$. The next step is to interpret the displacements $z(x, y)$ such that they lie in the plane and apply to the coordinates x and y individually. For clarity we are using two TPS solutions here, one for the x coordinates and one for the y. This allows us to transform known points (x_i, y_i) to other known points (X_i, Y_i) whilst keeping the surrounding deformation smooth and continuous. As before we will begin with a finitely many points and then generalise.

3.1.1 Finite point example

For four points we have the following picture:

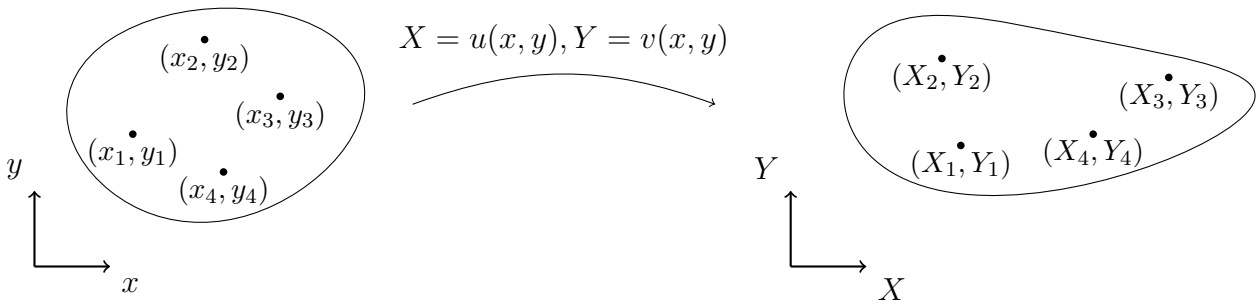


Figure 3.1: Figure depicting a deformation map where material points $\boldsymbol{x}_i$ are mapped to points $\boldsymbol{X}_i$.

where $(x_i, y_i) \in \mathbb{R}^2$, $(X_i, Y_i) \in \mathbb{R}^2$ are the undeformed and deformed coordinates respectively. First looking at x, by (2.16) we have the mapping

$$X(x,y) = w_1 u_1(x,y) + w_2 u_2(x,y) + w_3 u_3(x,y) + w_4 u_4(x,y) \tag{3.1}$$

with the conditions on the interpolated points being such that

$$\begin{cases} X_1 = w_1 u_1(x_1,y_1) + w_2 u_2(x_1,y_1) + w_3 u_3(x_1,y_1) + w_4 u_4(x_1,y_1) \\ X_2 = w_1 u_1(x_2,y_2) + w_2 u_2(x_2,y_2) + w_3 u_3(x_2,y_2) + w_4 u_4(x_2,y_2) \\ X_3 = w_1 u_1(x_3,y_3) + w_2 u_2(x_3,y_3) + w_3 u_3(x_3,y_3) + w_4 u_4(x_3,y_3) \\ X_4 = w_1 u_1(x_4,y_4) + w_2 u_2(x_4,y_4) + w_3 u_3(x_4,y_4) + w_4 u_4(x_4,y_4) \end{cases}$$

which again is a system of linear equations. The matrix of this system is

$$\begin{pmatrix} u_1(x_1,y_1) & u_2(x_1,y_1) & u_3(x_1,y_1) & u_4(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & u_3(x_2,y_2) & u_4(x_2,y_2) \\ u_1(x_3,y_3) & u_2(x_3,y_3) & u_3(x_3,y_3) & u_4(x_3,y_3) \\ u_1(x_4,y_4) & u_2(x_4,y_4) & u_3(x_4,y_4) & u_4(x_4,y_4) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} \tag{3.2}$$

This is of the form of (2.17) which is to be expected as we have done exactly the same thing except apply the deformation in the x direction rather than perpendicular to the plane

Switching attention to the y coordinates we obtain a similar coordinate transformation and system of equations

$$Y(x,y) = w_1' u_1(x,y) + w_2' u_2(x,y) + w_3' u_3(x,y) + w_4' u_4(x,y) \tag{3.3}$$

$$\begin{pmatrix} u_1(x_1,y_1) & u_2(x_1,y_1) & u_3(x_1,y_1) & u_4(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & u_3(x_2,y_2) & u_4(x_2,y_2) \\ u_1(x_3,y_3) & u_2(x_3,y_3) & u_3(x_3,y_3) & u_4(x_3,y_3) \\ u_1(x_4,y_4) & u_2(x_4,y_4) & u_3(x_4,y_4) & u_4(x_4,y_4) \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} \tag{3.4}$$

We are now equipped to look at deforming a unit square rotated 45° into a kite and observing

what kind of a deformation grid our TPS gives. Choosing $(-1, 0)$, $(0, 1)$, $(1, 0)$, $(0, -1)$ as the left, top, right and bottom corners of the square respectively we displace the bottom coordinate down one unit while keeping the rest the same. This gives a kite.

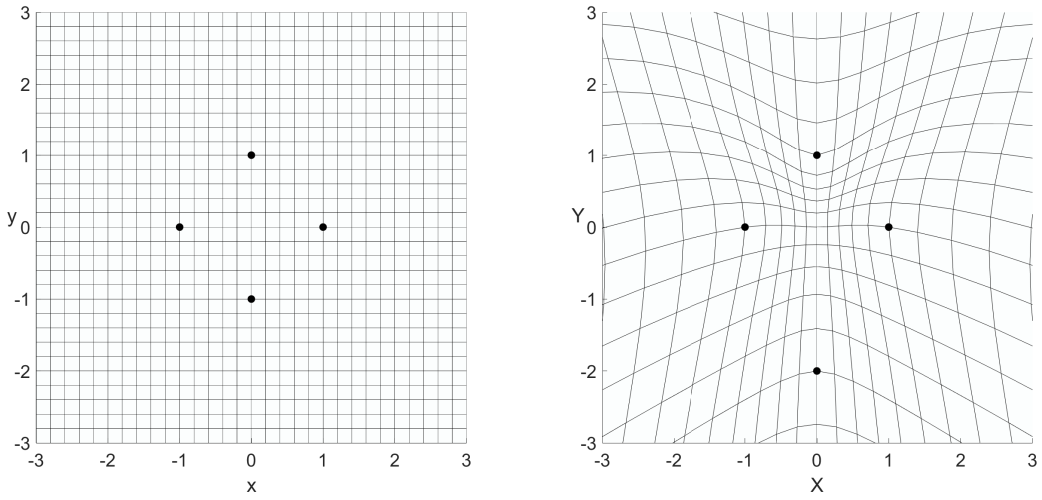


Figure 3.2: Looking at the interpolation map generated by the TPS when deforming a unit square into a kite.

This image is concerning as the grid does not look particularly smooth considering this is a very simple transformation. If we look at the image grid squares all of them have been stretched as they are larger than the source grid squares. This means the TPS by itself is not doing a good job at minimising the change in area elements which is what we expect due to the thin plate being resistant to bending. This stretching is increasing as x and y get larger in magnitude. This implies we need extra conditions to rein in the distortion due to the TPS as x and y tend to infinity. We will also re-evaluate our choices in regards to disregarding terms in the fundamental solution of the biharmonic and see how they relate to looking far afield in x and y. First we will just touch on generalising what we have achieved so far to n points for the sake of completeness.

3.1.2 Generalising to n points

Using (2.16) and (2.17) we arrive at the method for determining an n point TPS deformation map. Looking at the x coordinates first, we have

$$X(x,y) = w_1 u_1(x,y) + w_2 u_2(x,y) + \cdots + w_n u_n(x,y) \tag{3.5}$$

$$\begin{pmatrix} u_1(x_1,y_1) & u_2(x_1,y_1) & \cdots & u_n(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & \cdots & u_n(x_2,y_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(x_n,y_n) & u_2(x_n,y_n) & \cdots & u_n(x_n,y_n) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} \tag{3.6}$$

and then for the y coordinates

$$Y(x,y) = w'_1 u_1(x,y) + w'_2 u_2(x,y) + \cdots + w'_n u_n(x,y) \tag{3.7}$$

$$\begin{pmatrix} u_1(x_1,y_1) & u_2(x_1,y_1) & \cdots & u_n(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & \cdots & u_n(x_2,y_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(x_n,y_n) & u_2(x_n,y_n) & \cdots & u_n(x_n,y_n) \end{pmatrix} \begin{pmatrix} w'_1 \\ w'_2 \\ \vdots \\ w'_n \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} \tag{3.8}$$

We can combine these two separate systems of equations into one to obtain a full solution of the spline

$$\begin{pmatrix} u_1(x_1,y_1) & u_2(x_1,y_1) & \cdots & u_n(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & \cdots & u_n(x_2,y_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(x_n,y_n) & u_2(x_n,y_n) & \cdots & u_n(x_n,y_n) \\ u_1(x_1,y_1) & u_2(x_1,y_1) & \cdots & u_n(x_1,y_1) \\ u_1(x_2,y_2) & u_2(x_2,y_2) & \cdots & u_n(x_2,y_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(x_n,y_n) & u_2(x_n,y_n) & \cdots & u_n(x_n,y_n) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w'_1 \\ w'_2 \\ \vdots \\ w'_n \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \\ Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}. \tag{3.9}$$

3.1.3 Looking at limits

We want to modify our equations (3.5), (3.7) and (3.9) to be asymptotically affine for large x and y to counter the distortion. Define a new function F such that

$$F(x,y) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{n} w_i u_i(x,y), \quad a_0, a_1, a_2 \in \mathbb{R}^2 \tag{3.10}$$

Notice this is just (2.16) with an additional affine part. We want to look at the limit as (x, y) tend to infinity and introduce more constraints on the weights w_i such that we can solve for the constants a_0, a_1 and a_2. Let us fix y, looking in the limit as $x \to \pm\infty$ and expanding out $z(x, y)$ we obtain

$$\begin{aligned} z(x,y) &= \frac{1}{2}\sum_{i=1}^{n} w_i \left[(x-x_i)^2 + (y-y_i)^2\right] \ln\left[(x-x_i)^2 + (y-y_i)^2\right] \\ &= \frac{1}{2}\sum_{i=1}^{n} w_i \left[x^2 - 2xx_i + x_i^2 + (y-y_i)^2\right] \ln\left[x^2 - 2xx_i + x_i^2 + (y-y_i)^2\right] \\ &= \frac{1}{2}\sum_{i=1}^{n} w_i \left[x^2 - 2xx_i + x_i^2 + (y-y_i)^2\right] \ln\left[x^2\left(1 - \frac{2x_i}{x} + \frac{x_i^2}{x^2} + \left(\frac{y-y_i}{x}\right)^2\right)\right]. \end{aligned}$$

The dominating term from the natural log part of $u_i(x, y)$ in the limit $x \to \pm\infty$ is $\ln x^2 = 2 \ln x$, hence

$$\begin{aligned} F(x,y) &\xrightarrow[x\to\infty]{} a_0 + a_1 x + a_2 y + \ln x \sum_{i=1}^{n} w_i \left[x^2 - 2xx_i + x_i^2 + (y-y_i)^2\right] \\ &= a_0 + a_1 x + a_2 y + x^2 \ln x \sum_{i=1}^{n} w_i - 2x \ln x \sum_{i=1}^{n} w_i x_i + \cdots \end{aligned}$$

We want F to be affine in this limit i.e. $a_1 x$ is the dominant term so we set

$$\sum_{i=1}^{n} w_i = 0 \quad \text{and} \quad \sum_{i=1}^{n} w_i x_i = 0.$$

Doing the same analysis for $y \to \infty$ with x being fixed and due to the symmetry of $z(x, y)$ we can easily obtain another condition

$$F(x,y) \xrightarrow[y\to\infty]{} a_0 + a_1 x + a_2 y + \ln y \sum_{i=1}^{n} w_i \left[y^2 - 2yy_i + y_i^2 + (x-x_i)^2\right]$$

$$= a_0 + a_1 x + a_2 y + y^2 \ln y \sum_{i=1}^{n} w_i - 2y \ln y \sum_{i=1}^{n} w_i y_i + \cdots$$

$$\Rightarrow \sum_{i=1}^{n} w_i y_i = 0.$$

So we have three more conditions on the weights which ensure the affine nature of $F(x, y)$ as x and y become large.

$$\begin{aligned} \sum_{i=1}^{n} w_i &= 0 \\ \sum_{i=1}^{n} w_i x_i &= 0 \\ \sum_{i=1}^{n} w_i y_i &= 0. \end{aligned} \tag{3.11}$$

The requirement that $\sum_{i=1}^{n} w_i = 0$ can be physically interpreted as all the point forces given by the delta functions on the thin metal plate have to be balanced i.e. there is no net force. The other two conditions ensure that the overall bending moment is zero.

Now if we take a look at the fundamental solution for the biharmonic before we disregarded terms and let a new constant $k = k_0 - 1/2\pi$ we obtain

$$u(r) = \frac{r^2}{16\pi} \ln r^2 + \frac{kr^2}{8\pi} + k_1.$$

If we then use this as our new definition for u_i in (2.16) and look far afield as above we retain all our previous conditions but additionally see that the $1/8\pi$ constant gets absorbed into the weights so we are justified in ignoring it. We also see that we have a constant term involving k_1 and then linear terms involving k. These terms get absorbed into the affine part of our equation and can be thought of as the origin to why we needed an affine part. This shows that our original assumption about ignoring terms wasn't completely correct and we needed to introduce the affine part which we have done so now.

3.1.4 Adjusting the TPS system of equations

Now we have our extra terms we need to encompass them into (3.6) and officially redefine (2.16) as $F(x, y)$. Let (3.10) be our new equation defining the TPS and let $\boldsymbol{A}$ denote the matrix in (3.6). Our system of equations for the x coordinates are now

$$
\begin{aligned}
X_1 &= w_1u_1(x_1,y_1) + w_2u_2(x_1,y_1) + \cdots + w_nu_n(x_1,y_1) + a_0 + a_1x_1 + a_2y_1,\\
X_2 &= w_1u_1(x_2,y_2) + w_2u_2(x_2,y_2) + \cdots + w_nu_n(x_2,y_2) + a_0 + a_1x_2 + a_2y_2,\\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \quad \vdots \quad \vdots\\
X_n &= w_1u_1(x_n,y_n) + w_2u_2(x_n,y_n) + \cdots + w_nu_n(x_n,y_n) + a_0 + a_1x_n + a_2y_n\\
0 &= w_1 + w_2 + \cdots + w_n + 0 + 0 + 0\\
0 &= w_1x_1 + w_2x_2 + \cdots + w_nx_n + 0 + 0 + 0\\
0 &= w_1y_1 + w_2y_2 + \cdots + w_ny_n + 0 + 0 + 0.
\end{aligned}
$$

We can write this system in matrix form as follows

$$
\left(\begin{array}{cccc|ccc}
u_1(x_1,y_1) & u_2(x_1,y_1) & \cdots & u_n(x_1,y_1) & 1 & x_1 & y_1\\
u_1(x_2,y_2) & u_2(x_2,y_2) & \cdots & u_n(x_2,y_2) & 1 & x_2 & y_2\\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\\
u_1(x_n,y_n) & u_2(x_n,y_n) & \cdots & u_n(x_n,y_n) & 1 & x_n & y_n\\
\hline
1 & 1 & \cdots & 1 & 0 & 0 & 0\\
x_1 & x_2 & \cdots & x_n & 0 & 0 & 0\\
y_1 & y_2 & \cdots & y_n & 0 & 0 & 0
\end{array}\right)
\left(\begin{array}{c} w_1\\ w_2\\ \vdots\\ w_n\\ \hline a_0\\ a_1\\ a_2 \end{array}\right)
=
\left(\begin{array}{c} X_1\\ X_2\\ \vdots\\ X_n\\ \hline 0\\ 0\\ 0 \end{array}\right). \tag{3.12}
$$

Denoting the top right quadrant of the matrix as $\boldsymbol{B}$ and noticing the bottom left is just $\boldsymbol{B}^T$ we can write this in a more compact form

$$
\left(\begin{array}{c|c} \boldsymbol{A}_{n\times n} & \boldsymbol{B}_{n\times 3}\\ \hline \boldsymbol{B}^T_{3\times n} & \boldsymbol{0}_{3\times 3} \end{array}\right)
\left(\begin{array}{c} \boldsymbol{W}_{n\times 1}\\ \hline \boldsymbol{a}_{3\times 1} \end{array}\right)
=
\left(\begin{array}{c} \boldsymbol{X}_{n\times 1}\\ \hline \boldsymbol{0}_{3\times 1} \end{array}\right). \tag{3.13}
$$

Similarly for the y coordinates we have

$$
G(x,y) = a'_0 + a'_1x + a'_2y + \sum_{i=1}^{n} w'_iu_i(x,y) \tag{3.14}
$$

$$
\left(\begin{array}{c|c} \boldsymbol{A}_{n\times n} & \boldsymbol{B}_{n\times 3}\\ \hline \boldsymbol{B}^T_{3\times n} & \boldsymbol{0}_{3\times 3} \end{array}\right)
\left(\begin{array}{c} \boldsymbol{W}'_{n\times 1}\\ \hline \boldsymbol{a}'_{3\times 1} \end{array}\right)
=
\left(\begin{array}{c} \boldsymbol{Y}_{n\times 1}\\ \hline \boldsymbol{0}_{3\times 1} \end{array}\right). \tag{3.15}
$$

We have derived the TPS which was discussed in [2]. Revisiting the square example we have an updated picture of the deformation map, shown in figure 3.3

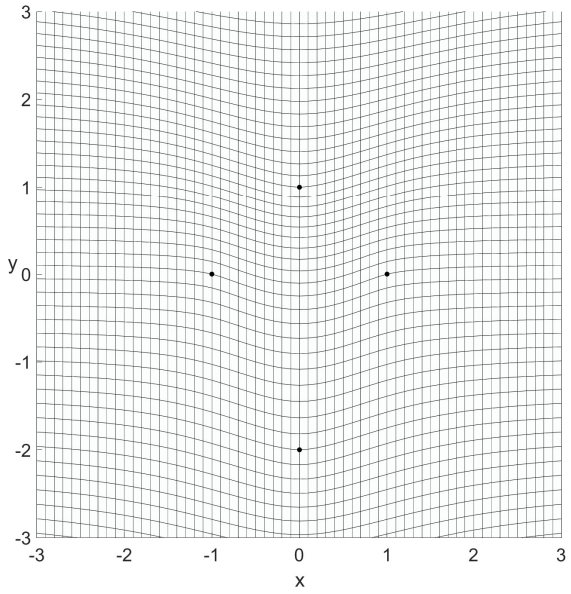


Figure 3.3: Deformation of a square into a kite generated by the TPS

This looks much better as the size of the grid squares has varied only slightly which is more in line with the gradual interpolation we want to see.

3.1.5 Bending energy revisited

Now we have the full formulation of the TPS we can relate the system of equations to the plate bending energy (2.20). We can use Bookstein's formula, see [2] and [3], for the TPS bending energy which in our notation is

$$E_F = \frac{1}{8\pi}\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{W} \tag{3.16}$$

$$E_G = \frac{1}{8\pi}\boldsymbol{W}'^T\boldsymbol{A}\boldsymbol{W}' \tag{3.17}$$

where E_F and E_G represent the bending energies associated with the spline applied to the x coordinates and y coordinates respectively. To calculate the total bending energy we sum E_F and E_G. The bending energy for the kite seen in Figure 3.3 is 0.0708 with no contribution from the E_F term which is to be expected as the x coordinates are unchanged during mapping. We

can visualise these bending energies by calculating surfaces for the affine free part of the splines applied to the x and y coordinates respectively. These surfaces do not represent any particular interpolation, they are merely a representation of how the thin plate would bend under the conditions given by the displacements of the coordinates.

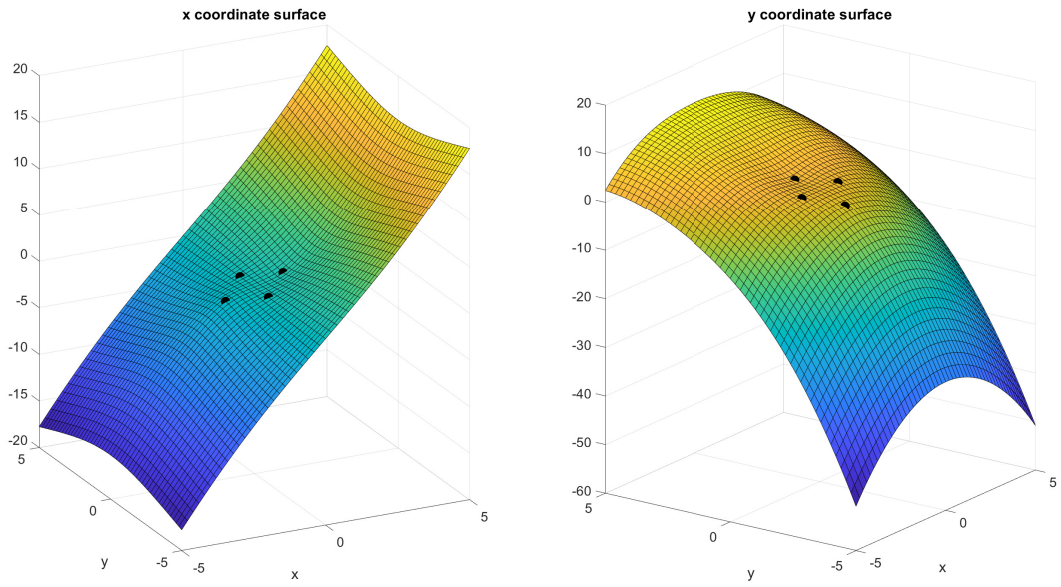


Figure 3.4: Bending energy visualised, E_F (left) and E_G (right).

Figure 3.4 allows us to immediately see why there is no contribution from the E_F term as the bent plate is symmetrical about the middle of the kite and we have two bends applied to the top and bottom of the kite in opposite directions which cancel each other out. Whereas for the y displacements there seems to be a general displacement downwards into the plate which is to be expected as we do have contributions to the bending energy from the E_G term.

3.2 Deformation analysis of the TPS

We want to be able to measure exactly how much the grid squares are changing in our deformation map, to do this we will take ideas for continuum mechanics and develop the deformation gradient tensor and strain tensors to characterise the amount of deformation for the TPS.

3.2.1 Finite strain theory, a brief overview

We have seen the picture of a deformation map in figure Figure 3.1.1, this showed known material points being mapped to known material points whilst the continuum deformed around

them according to the TPS model. We now want to look at line elements that connect two different material points in the undeformed region and quantify how this line element changes as we apply the deformation. Note that we are working in the Lagrangian framework here as everything is a function of the undeformed coordinates, we are mapping from an 'old' biological object to 'new' one where the maps are a function of the 'old' object. Consider the mapping

$$\begin{aligned}(X, Y) &= (F(x, y), G(x, y)) \\ \Rightarrow (X + dX, Y + dY) &= (F(x + dx, y + dy), G(x + dx, y + dy))\end{aligned}$$

where F and G are as seen in (3.5) and (3.7) respectively. Using Taylor's theorem we find

$$\begin{aligned}F(x + dx, y + dy) &= F(x, y) + \frac{\partial F}{\partial x}(x, y)dx + \frac{\partial F}{\partial y}(x, y)dy + \mathcal{O}\big(dx^2, dy^2\big) \\ G(x + dx, y + dy) &= G(x, y) + \frac{\partial G}{\partial x}(x, y)dx + \frac{\partial G}{\partial y}(x, y)dy + \mathcal{O}\big(dx^2, dy^2\big)\end{aligned}$$

hence in the limit as $x \to \infty$ this reduces to

$$\begin{aligned}X + dX &= F(x, y) + \frac{\partial F}{\partial x}(x, y)dx + \frac{\partial F}{\partial y}(x, y)dy \\ Y + dY &= G(x, y) + \frac{\partial G}{\partial x}(x, y)dx + \frac{\partial G}{\partial y}(x, y)dy\end{aligned}$$

but $X = F(x, y)$ and $Y = G(x, y)$ by construction. Hence we obtain the following expressions for dX and dY

$$\begin{aligned}dX &= \frac{\partial F}{\partial x}(x, y)dx + \frac{\partial F}{\partial y}(x, y)dy \\ dY &= \frac{\partial G}{\partial x}(x, y)dx + \frac{\partial G}{\partial y}(x, y)dy.\end{aligned}$$

Writing this in matrix notation we have

$$\begin{pmatrix} dX \\ dY \end{pmatrix} = \begin{pmatrix} F_x & F_y \\ G_x & G_y \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \tag{3.18}$$

where we recognise the Jacobian matrix, $\mathbf{J}$, of the function $\boldsymbol{F} = (F(x, y), G(x, y))$. The Jacobian matrix is a representation of the deformation gradient tensor and it describes how line elements change under the transformation $\mathbf{F}$. $\mathbf{J}$ has important properties, one being that $\det(\mathbf{J})$ gives the scale factor by which $\mathbf{F}$ is expanding or shrinking area elements. This is because when we transform our grid all the square area elements transform into parallelograms which have areas $dX \times dY$ where $\times$ is the cross product. Hence, using the antisymmetry property of the cross product and the fact $dx \times dx = dy \times dy = 0$, we have

$$
\begin{aligned}
dX \times dY &= \left(\frac{\partial F}{\partial x}dx + \frac{\partial F}{\partial y}dy\right) \times \left(\frac{\partial G}{\partial x}dx + \frac{\partial G}{\partial y}dy\right) \\
&= \frac{\partial F}{\partial x}\frac{\partial G}{\partial x}dx \times dx + \frac{\partial F}{\partial x}\frac{\partial G}{\partial y}dx \times dy + \frac{\partial F}{\partial y}\frac{\partial G}{\partial x}dy \times dx + \frac{\partial F}{\partial y}\frac{\partial G}{\partial y}dy \times dy \\
&= \frac{\partial F}{\partial x}\frac{\partial G}{\partial y}dx \times dy - \frac{\partial F}{\partial y}\frac{\partial G}{\partial x}dx \times dy \\
&= \left(\frac{\partial F}{\partial x}\frac{\partial G}{\partial y} - \frac{\partial F}{\partial y}\frac{\partial G}{\partial x}\right) dx \times dy \\
&= \det\left(\mathbf{J}\right)\, dx \times dy.
\end{aligned}
$$

For physically feasible deformations, we require $\det\left(\boldsymbol{J}\right) > 0$. This ensures that the mapped areas don't fold in on themselves.

We construct further useful quantities such as the Cauchy-Green deformation tensor [6], this helps us quantify the change in line elements when applying the TPS. Switching to index notation such that $(x_1, x_2) = (x, y)$, $(X_1, X_2) = (X, Y)$ and using the Einstein summation convention, we define the length of an undeformed line element as $ds^2 = dx_i dx_i$ and likewise $dS^2 = dX_i dX_i$ as the deformed line element length. We can then define a measure of the change in line element length by

$$
dS^2 - ds^2 = dX_i dX_i - dx_i dx_i = J_{ki} dx_i J_{kj} dx_j - dx_k dx_k = \left(J_{ki} J_{kj} - \delta_{ki}\delta_{kj}\right) dx_k dx_k
$$

The quantity $J_{ki}J_{kj} = \boldsymbol{J}^T\boldsymbol{J}$ is the (right) Cauchy-Green deformation tensor, $\boldsymbol{C}$, whose entries give us the local square change in size of the line elements. The eigenvectors of $\boldsymbol{C}$ give the maximal and minimal stretches of the deformation and the associated eigenvalues give the squared principal stretches which are the squares of the stretches in the direction of the eigenvectors. The Green-Lagrange strain tensor is then defined to be $\boldsymbol{E} = 1/2(\boldsymbol{C} - \boldsymbol{I})$ which tells us how much the size of the line elements have changed under the deformation, however we have all we need in $\boldsymbol{C}$ and $\boldsymbol{J}$ so we will not be calculating $\boldsymbol{E}$ for the TPS.

Most importantly we can construct the polar decomposition of the deformation gradient tensor $\mathbf{J}$. Since we are demanding $\det\left(\boldsymbol{J}\right) > 0$, $\mathbf{J}$ is invertible and hence we can decompose it into an orthogonal tensor, $\mathbf{R}$ and a positive definite symmetric tensor, $\mathbf{U}$, as follows

$$
\boldsymbol{J} = \boldsymbol{R}\boldsymbol{U} \Rightarrow \boldsymbol{J}^T\boldsymbol{J} = \left(\boldsymbol{R}\boldsymbol{U}\right)^T \boldsymbol{R}\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}^2 \tag{3.19}
$$

Hence $\boldsymbol{U}^2 = \boldsymbol{J}^T\boldsymbol{J} = \boldsymbol{C}$, the Cauchy-Green deformation tensor. So $\boldsymbol{U} = \sqrt{\boldsymbol{C}}$ represents the stretch in the deformation and $\boldsymbol{R}$ represents the rotation. $\boldsymbol{R}$ can be easily found by applying $\boldsymbol{J}$ to $\boldsymbol{U}^{-1}$. This decomposition shows that any deformation given by the TPS can be seen locally as a rotation and two mutually orthogonal stretches, the principal stretches, which are the eigenvalues of $\boldsymbol{U}$.

3.2.2 Deformation gradient tensor of the TPS

We can now apply all this theory to the TPS. From (2.10) we have

$$\begin{aligned}
\frac{\partial u(x,y)}{\partial x} &= \frac{\partial}{\partial x}\left(\left[(x-x_i)^2+(y-y_i)^2\right]\ln\left[(x-x_i)^2+(y-y_i)^2\right]\right) \\
&= 2(x-x_i)\ln\left[(x-x_i)^2+(y-y_i)^2\right] + 2(x-x_i)\frac{\left[(x-x_i)^2+(y-y_i)^2\right]}{\left[(x-x_i)^2+(y-y_i)^2\right]} \\
&= 2(x-x_i)\left(\ln\left[(x-x_i)^2+(y-y_i)^2\right]+1\right) \\
\frac{\partial u(x,y)}{\partial y} &= 2(x-x_i)\left(\ln\left[(x-x_i)^2+(y-y_i)^2\right]+1\right).
\end{aligned}$$

Hence the deformation gradient tensor or Jacobian matrix is, for $r_i = \sqrt{(x-x_i)^2+(y-y_i)^2}$,

$$\boldsymbol{J} = \begin{pmatrix} F_x & F_y \\ F'_x & F'_y \end{pmatrix} = \begin{pmatrix} a_1 + \sum_i^n w_i(x-x_i)\ln(r_i^2) & a_2 + \sum_i^n w_i(y-y_i)\ln(r_i^2) \\ a'_1 + \sum_i^n w'_i(x-x_i)\ln(r_i^2) & a'_2 + \sum_i^n w'_i(y-y_i)\ln(r_i^2) \end{pmatrix} \tag{3.20}$$

This allows us to use 3.18 to obtain expressions for the line elements under the mapping given by the TPS

$$\begin{pmatrix} dX \\ dY \end{pmatrix} = \begin{pmatrix} \left(a_1 + \sum_i^n w_i(x-x_i)\ln(r_i^2)\right)dx + \left(a_2 + \sum_i^n w_i(y-y_i)\ln(r_i^2)\right)dy \\ \left(a'_1 + \sum_i^n w'_i(x-x_i)\ln(r_i^2)\right)dx + \left(a'_2 + \sum_i^n w'_i(y-y_i)\ln(r_i^2)\right)dy \end{pmatrix}. \tag{3.21}$$

Since we have calculated the deformation gradient tensor $\boldsymbol{J}$, we can easily calculate the Cauchy-Green deformation tensor $\boldsymbol{C}$ for the TPS. We have

$$\boldsymbol{C} = \boldsymbol{J}^T\boldsymbol{J} = \begin{pmatrix} F_x & G_x \\ F_y & G_y \end{pmatrix}\begin{pmatrix} F_x & F_y \\ G_x & G_y \end{pmatrix} = \begin{pmatrix} (F_x)^2+(G_x)^2 & F_xF_y+G_xG_y \\ F_xF_y+G_xG_y & (F_y)^2+(G_y)^2 \end{pmatrix} \tag{3.22}$$

To find the polar decomposition of $\boldsymbol{J}$ we need to take the matrix square root of $\boldsymbol{C}$. Since $\boldsymbol{C}$ is real and symmetric, it is diagonalizable and hence we can use the spectral theorem take its

eigendecomposition $\boldsymbol{C} = \boldsymbol{Q}^T\boldsymbol{D}\boldsymbol{Q}$ where $\boldsymbol{D} = diag(\lambda_1, \lambda_2)$ and $\boldsymbol{Q}$ is an orthogonal matrix who's columns are the eigenvectors of $\boldsymbol{C}$ with associated eigenvectors λ_1, λ_2. Hence we find that $\sqrt{\boldsymbol{C}} = \boldsymbol{Q}^T\sqrt{\boldsymbol{D}}\boldsymbol{Q}$ where $\sqrt{\boldsymbol{D}} = diag(\sqrt{\lambda_1}, \sqrt{\lambda_2})$. Indeed, $(\boldsymbol{Q}^T\sqrt{\boldsymbol{D}}\boldsymbol{Q})(\boldsymbol{Q}^T\sqrt{\boldsymbol{D}}\boldsymbol{Q}) = (\boldsymbol{Q}^T\sqrt{\boldsymbol{D}}\sqrt{\boldsymbol{D}}\boldsymbol{Q}) = \boldsymbol{Q}^T\boldsymbol{D}\boldsymbol{Q} = \boldsymbol{C}$. All our deformation measures rely purely on the derivatives of F and G which we have equations for (3.20) and so we are now well equipped to analyse deformation given by the TPS.

3.2.3 Deformation of a square into a kite

We can now revisit our kite example in full. We can visualise the change in line elements by constructing a line element vector $\binom{dx}{dy}$ at the bottom left corner of each of the area elements of the spline which are represented by a grid as seen in 3.3. Before the transformation, these vectors are purely horizontal going from the bottom left corner of the area element to the bottom right. We can see how the area elements are rotated by the direction in which these vectors are pointing once the transformation has been applied. We construct these deformed line element vectors by applying the rotation matrix of this particular deformation, which is obtained from the polar decomposition of (3.20), to the original line elements.

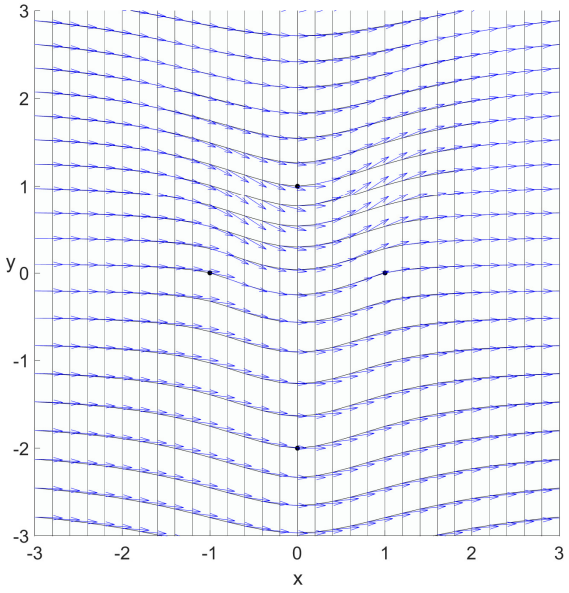


Figure 3.5: Figure highlighting the rotational element of a transformation by the TPS by applying the rotation matrix to line elements. Note that arrow length is not an indication of magnitude.

In figure 3.5 we can really start to see the effect the TPS has on the grid. As expected there is clear symmetry down the middle of the kite. To the left of this symmetry line we see line

elements rotating clockwise into the stretch given by displacement. Then to the right of the symmetry line we see the opposite, line elements rotating anticlockwise to counter balance the clockwise rotation. It is clear that the rotation is most extreme near the top of the kite, this is due to the area elements remaining roughly the same size so most of the deformation comes from the rotation whereas the area elements near the bottom of the kite the area elements are being elongated so most of the deformation comes from stretches rather than rotation. To quantify the change in size of area elements we can calculate the determinant of the deformation gradient tensor using (3.20).

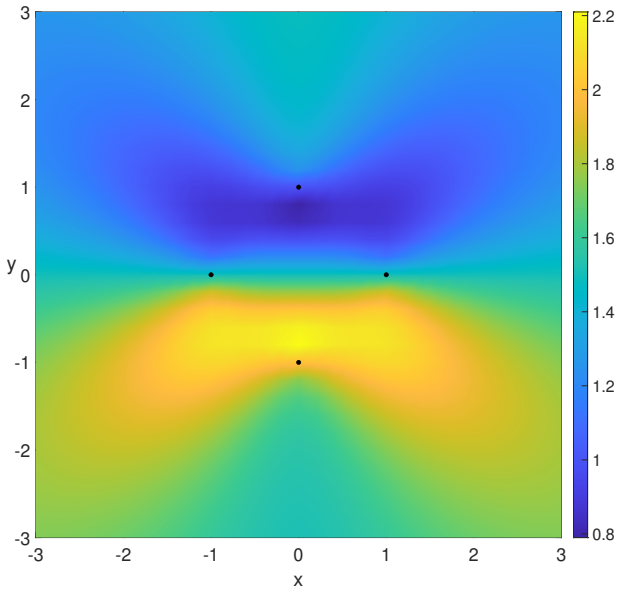


Figure 3.6: Heatmap highlighting the change in area element size which is given by $\det(\boldsymbol{J})$.

The heatmap show in figure (3.6) is obtained by taking the Jacobian determinant at each individual area element in the transformation grid. There is obvious symmetry here above and below the middle of the kite. Note how the rotations had symmetry down the middle of the kite and now the line of symmetry is along the middle. We can see the areas of greatest deformation are those below the two middle vertices of the kite which is to be expected as we can imagine the square being stretched out from below to deform into the kite. To compensate for this dilation, the top area of the kite and surrounding grid is compressed. For this compression to occur we do also see further dilation of area elements above the kite albeit a lot smaller in magnitude. Comparing the deformation gradient of this grid transformation to the one seen in Figure 3.2, we can see the impact of the new conditions we have introduced. For the original mapping, the area element that was scaled the most was done so by a factor of around 100, whereas for our most recent iteration of the TPS the most scaled area element was only done

so by a factor of around 2. This shows significant improvement which is due to the addition of the affine part of the map.

Chapter 4

D'arcy Thompson and the TPS

In this chapter we shall take a look at D'arcy Thompson's famous cartesian grid transformations [10] and see how they compare to the TPS.

4.1 Mapping homologous points on fish via the TPS

The first of Thompsons transformations we will look at is the comparison of two fish, a Haddock and a Plaice.

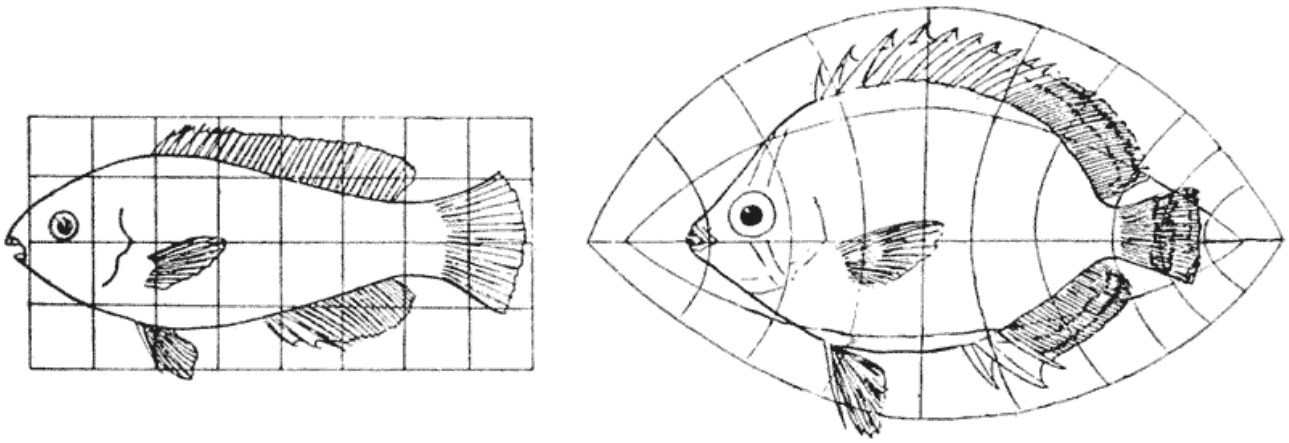


Figure 4.1: An example D'arcy Thompson's cartesian grid transformations applied to two different species of fish, a Haddock (left) and Plaice (right). [10].

We can immediately see here that this transformation is not physically possible and is not the sort we shall see using the TPS. If we look at the grid squares above and below the mouth of the Plaice we see that a square grid element has been transformed into a triangular shape. The only way this could have occurred is if two corners of the square welded into one, this would

imply a zero determinant of the deformation gradient tensor as two distinct points are being collapsed into one. This is then mirrored down the middle of the fish with the with exact same transformation occurring at the tail. This implies that some biological matter has been welded together, this is physically possible but however violates our idea of a smooth map.

For the first attempt of using the TPS to transform this Haddock into a Plaice we shall just use landmarks based on the location of the fins of the fish and then we will add in more landmarks to see what effect the increase has on the accuracy of the spline. As before we shall visually interpret these transformations by looking at before and after grid squares.

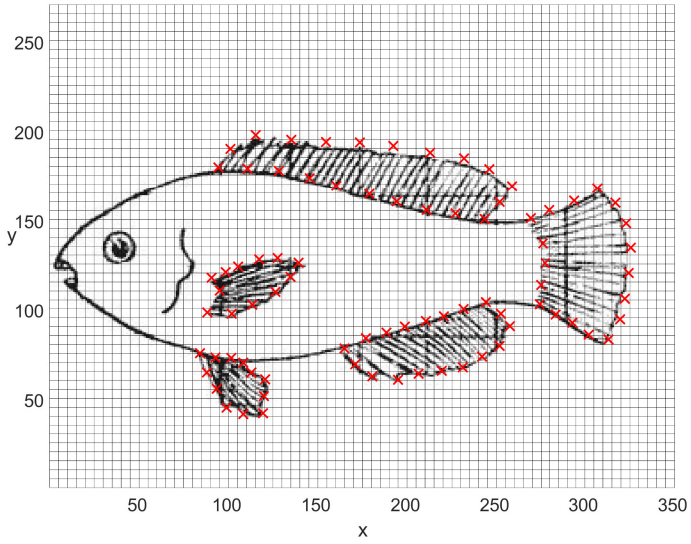


Figure 4.2: Displaying the Haddock from On Growth and Form [10] superimposed onto a new grid with landmarks highlighted as red crosses.

Figure 4.2 shows the haddock from On Growth and Form with a new superimposed grid which will demonstrate how the fish transforms under the TPS. Figure 4.3 shows where the landmarks in 4.2 will be mapped to. We are trying to see the effect this transformation will have on the rest of the fish's structure. We can see this effect in Figure 4.5 which shows the image of the haddock under the transformation given by the TPS. We can see most of the deformation is focused around the tail which is as expected. If we compare the original images of the fish we see that the Plaice has a much smaller tail compared to the rest of its body than the haddock and is much closer to the other fins. Hence the haddock's tail is going to be drawn into the body and reduced in size. We can use the deformation gradient to quantify the degree to which the grid squares are changing area. The deformation gradient map highlights areas that have

been stretched as well, in particular the dorsal fin. Once again comparing the two original fish we see this is expected as the Plaice has more curved and elongated dorsal fin, so the Plaice is stretched upwards towards the front of its dorsal fin as this is where the peak height of the Plaice's dorsal fin is.

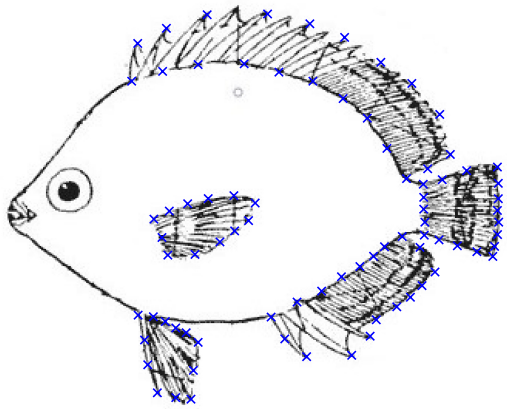


Figure 4.3: Displaying the Plaice from On Growth and Form [10] with landmarks highlighted as blue crosses.

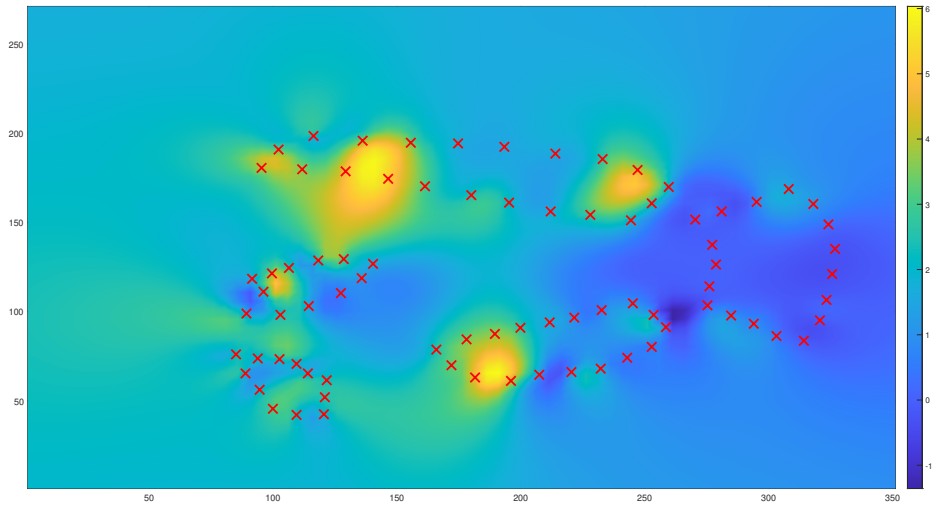


Figure 4.4: Jacobian applied to each area element of the transformation seen in 4.5.

Figure 4.5 shows the original image of the haddock wrapped around the grid generated by the TPS to give a picture of how the haddock would look under the given transformation. We

are aiming for the transformed haddock to look as similar to the plaice as possible and indeed even with just looking at homologous points on the fins we have quite a close match. In theory one could choose a set number of points per line in the image of the haddock and them match them to the same number of points per line in the plaice, essentially redrawing the haddock as a plaice. A very close match under the transformation given by the TPS would be obtained but it could not be known for sure if this was accurately simulating evolution as we are no longer using homologous points as landmarks. Even with using fins as landmarks it is difficult to tell what part of the fin on the haddock is mapped to what part of the fin on the plaice, only an educated guess can be made. What we have assumed in our mappings so far for the fins is that corners are mapped to corners and the intermediate points remain equally spaced apart. Nevertheless this is an issue with quantifying what determines a landmark rather than the TPS itself. One could argue that the bending energy of the TPS is an indicator to whether certain points are homologous or not, if the bending energy is very high that means an extreme deformation is required to interpolate the points via a TPS which could imply that it is unlikely the points should be interpolated at all i.e. they are not homologous. However extreme deformations do occur in nature so a high bending energy could naturally arise.

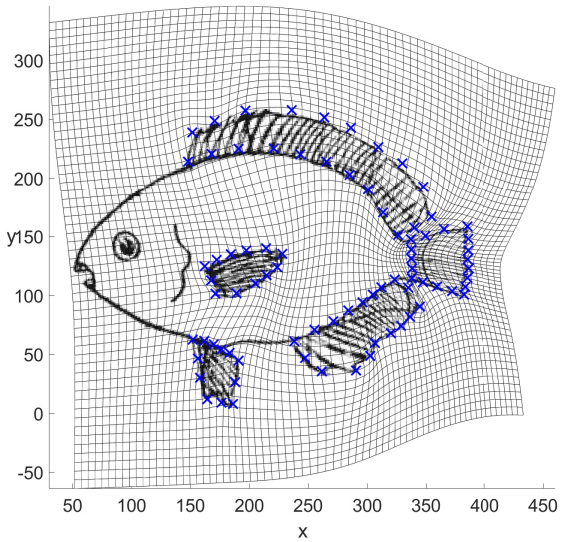
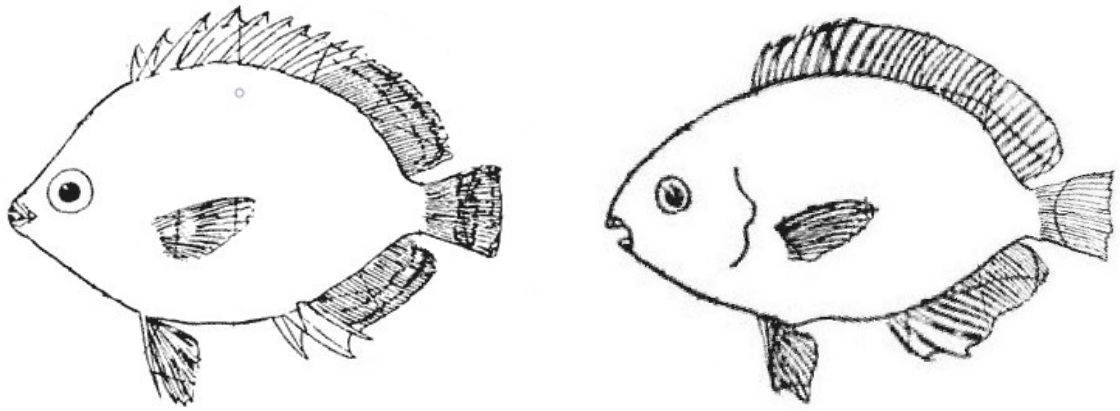


Figure 4.5: Displaying the Haddock from On Growth and Form [10] wrapped over a new grid transformed under the TPS with landmarks highlighted as blue crosses.



(a) Plaice (b) Deformed Haddock

Figure 4.6: Comparing deformed Haddock (without the grid) to the desired Plaice.

Up until now we haven't actually replicated D'arcy Thompson's approach in On Growth and Form, that is, creating a grid but then superimposing it onto the desired image which in our case is the plaice. What we have done is deform the original image by wrapping it around the grid generated by the TPS and then directly comparing it to the desired image. Figure 4.7 shows our TPS version of D'arcy Thompson's superimpositions and there are some obvious similarities. Around the tail area we see curves of a similar shape to Thompson's albeit without the welding element. This is slightly mirrored at the mouth but not to the same degree, however this might be due to the fact that we have a lot more landmarks around the tail and next to none at the head. If we map the mouth and the eye we might see more curving like Thompson's but as discussed earlier it is difficult to choose any extra landmarks around that area as it can be hard to justify that a certain line on one fish is homologous to a certain line on the other. In essence our mapping does seem to agree with Thompson's around the area of the tail.

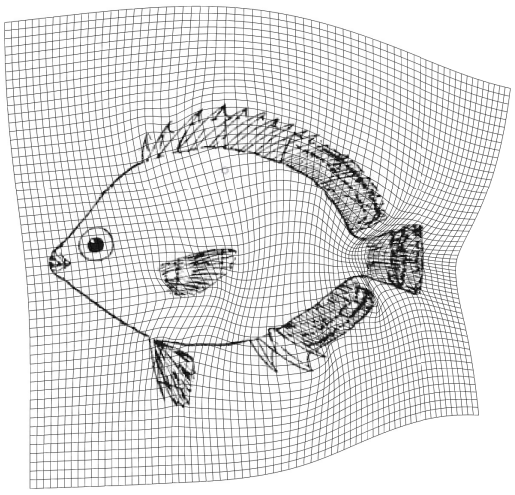


Figure 4.7: Shows a D'arcy Thompson inspired TPS generated grid superimposition.

4.2 Tangent lines under the TPS

For our fish we can introduce three more landmarks, one for the centre of the eye and two for the top and bottom of the mouth. Introducing these landmarks of course increases the similarity between the deformed haddock and the plaice, as seen in Figure 4.8, but also causes the line connecting the mouth to the top fin to straighten out which is in agreement with the shape of the plaice. This introduces an interesting question which is are tangent lines preserved under deformation given by the TPS, this question was explored by Bookstein in a later paper [3] by introducing the idea of an edgel. Edgels incorporate edge data by taking a pair of landmarks on a line and seperating them by an infinitesimally small distance. Bookstein noticed that as the landmarks approached each other the line segment between them became less steep, this was undesirable as we want to retain the original gradient of the line so a correction must be introduced. These corrections lead to a new algebraic development of the TPS that is beyond the scope of this paper, see [3] for details. One key point was that Bookstein was trying to bring in new data in edges as well as keeping landmark data. As the number of landmarks increase the block matrix seen in 3.6 becomes larger too as it is an $(n+3)\times(n+3)$ matrix with n being the number of landmarks and this matrix must be inverted to calculate the weights for the TPS. In fact we must invert two of these matrices, one for the x coordinates and one for the y, this becomes increasingly costly computationally wise. One could argue that with introducing edge data you could forgo some landmarks in exchange for edgels as opposed to introduce edgels

in addition to already predetermined landmarks and hence reducing the computational cost significantly.

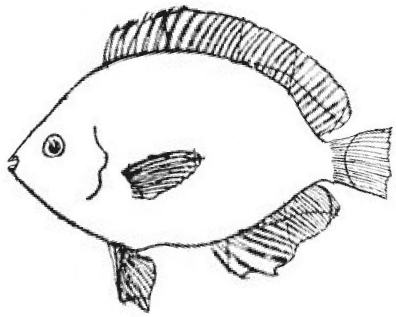


Figure 4.8: Transformed haddock with additional landmarks introduced around the mouth and eye.

4.3 Time varying TPS deformations

So far we have used the TPS to generate a mapping between two objects, the original object we wish to deform and the desired object we wish to deform the original into. We did this by interpolating predetermined landmarks, one set belonging to the original object and the other to the desired object, and looking at how the area deformed around them under the image of the TPS. We now want to look at interpolating the landmarks themselves using some interpolation technique and then treating the set of intermediate landmarks as a set of new desired objects. We then apply the TPS to these new sets of landmarks such that the original object is being mapped to each of the new objects. This creates multiple deformation maps which help build a picture of what the deformation looks like over time.

To interpolate the landmarks we can fist treat the y coordinates as functions of the x coordinates and then interpolate each individual original landmark and desired landmark pair in the (x, y) plane. This gives us a range of new interpolated y coordinates which are the y coordinates of our intermediate landmarks. We then repeat this process with treating the x coordinates as functions of y in the (y, x) plane to get the intermediate x landmark coordinates. Since

we are interpolating For the example in figure (4.1) we only have two sets of landmarks to interpret so we can use linear interpolation to find the intermediate landmarks. To interpolate the y coordinates we set up the equation $y = y_0 + (x - x_0)(y_1 - y_0)/(x_1 - x_0)$, as seen in chapter 1, where (x_0, y_0) and (x_1, y_1) are the corresponding landmarks for the original and desired image respectively and $x \in [x_0, x_1]$. The output y gives the y coordinates of the new intermediate landmarks. We then repeat this process swapping x and y. If we had 3 or more sets of landmarks we could use another TPS only in 1D i.e. a cubic spline.

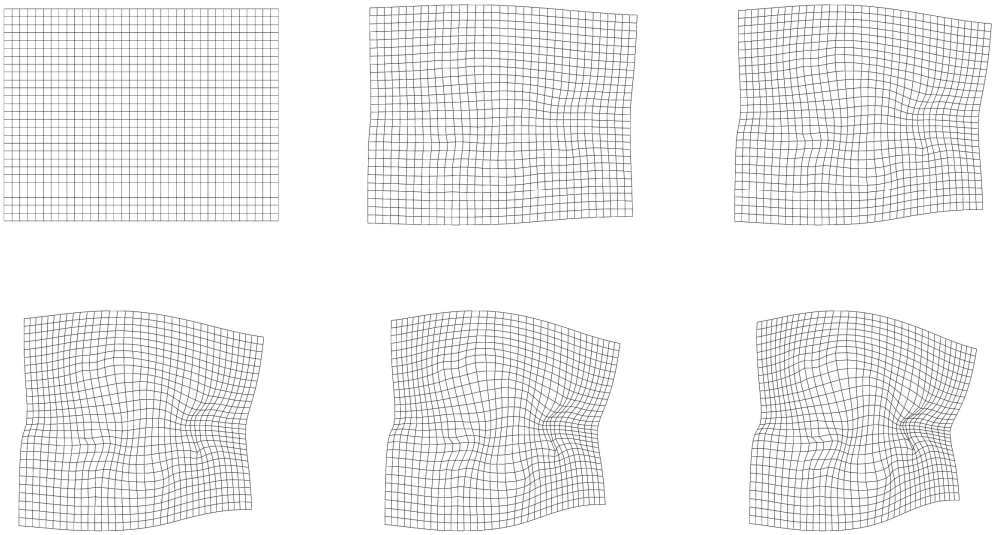


Figure 4.9: Time interpolated TPS transformation grids.

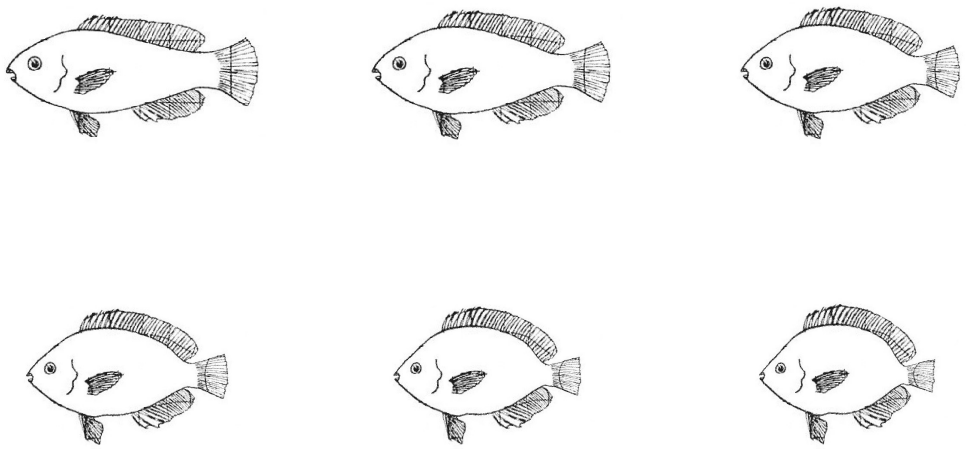


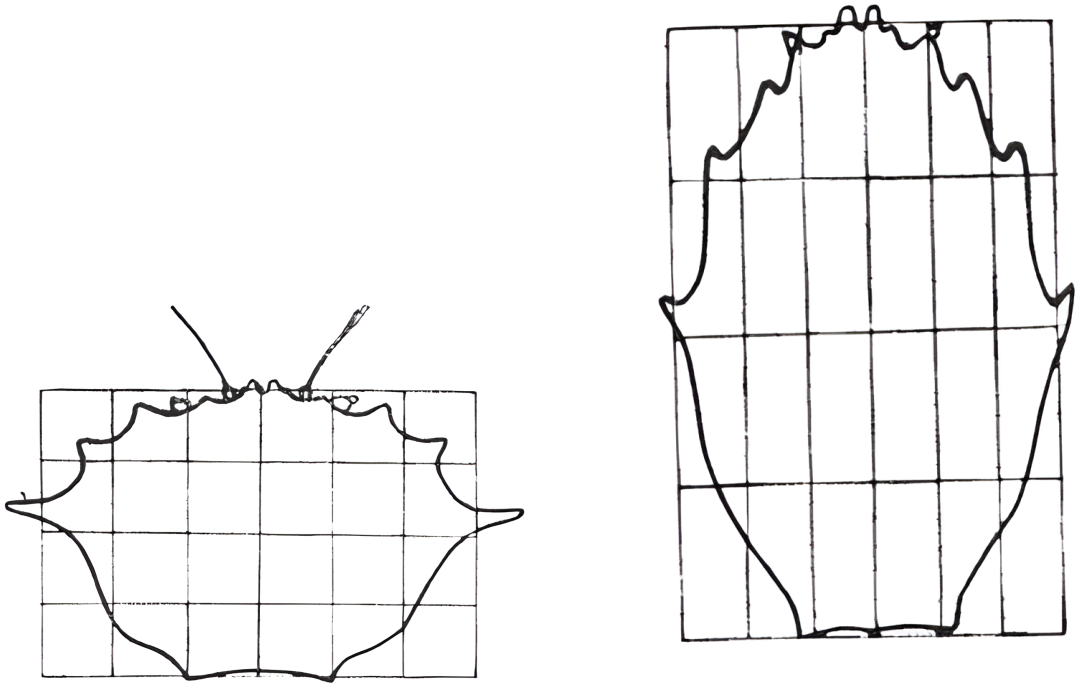
Figure 4.10: The Haddock from on Growth and Form [10] being wrapped over the time interpolated grids from Figure 4.9 with the grid lines removed.

We could expand this technique to interpolate different types of landmarks at different rates, for example if the fish's top fin evolved faster than its bottom fin. If this data was known we could increase the accuracy of the time varying TPS via this method.

4.4 A new example

Another interesting example of Thompson's to look at is the crab, specifically the pure stretch seen in Figure 4.11. These two crab shells are extremely similar and Thompson has laid out a Cartesian mapping that is purely a stretch in the y direction. The shells are so similar that in fact this might not represent any sort of evolutionary mapping between two different species and might be purely an example in deformation. Hence this is a good example to test our TPS out on and see if we obtain similar results.

The landmarks chosen are shown in Figure 4.12 and the resultant D'arcy Thompson style TPS grid superimposition is shown in Figure 4.13. Eyeballing it, we can already see within the crab the stretches implied in Thompson's images with most of the non pure vertical stretch deformation occurring at the top of the crab shell where the highest density of landmarks is. The TPS at this area is trying to fit every single landmark so we expect a higher level of deformation.



(a) Undeformed crab shell

(b) Deformed crab shell

Figure 4.11: An example of D'arcy Thompson's Cartesian grid transformations of crab shells.

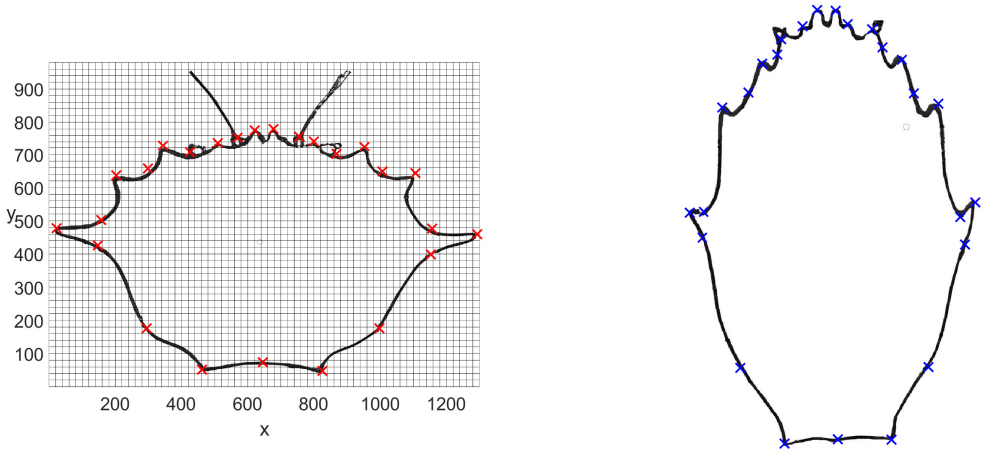


Figure 4.12: The landmarks on the undeformed crab (left) and deformed crab (right) represented by red and blue crosses respectively.

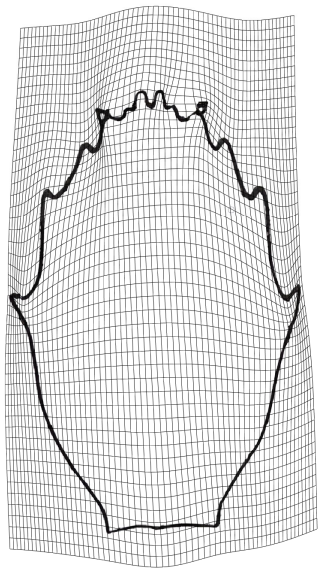


Figure 4.13: D'arcy Thompson style Cartesian grid superimposition with the grid generated by the TPS.

We can make use of the deformation gradient here to see if we do indeed have mostly uniform stretch. We can see from Figure 4.14 that we do have a fairly uniform stretch within the body of the crab as the area elements are roughly the same size. We can also see that there is a significant reduction of area element size near the top left of the shell which is an outlier in regards to the rest of the deformation. This is an indication that the landmarks chosen at this point were not homologous.

If we reduce the grid spacing for our deformation map to make it more in line with Thompson's images we have a very similar grid transformation to his as seen in Figure 4.15. So once again we have used the TPS to confirm that Thompson had the right general idea with his Cartesian grid transformations.

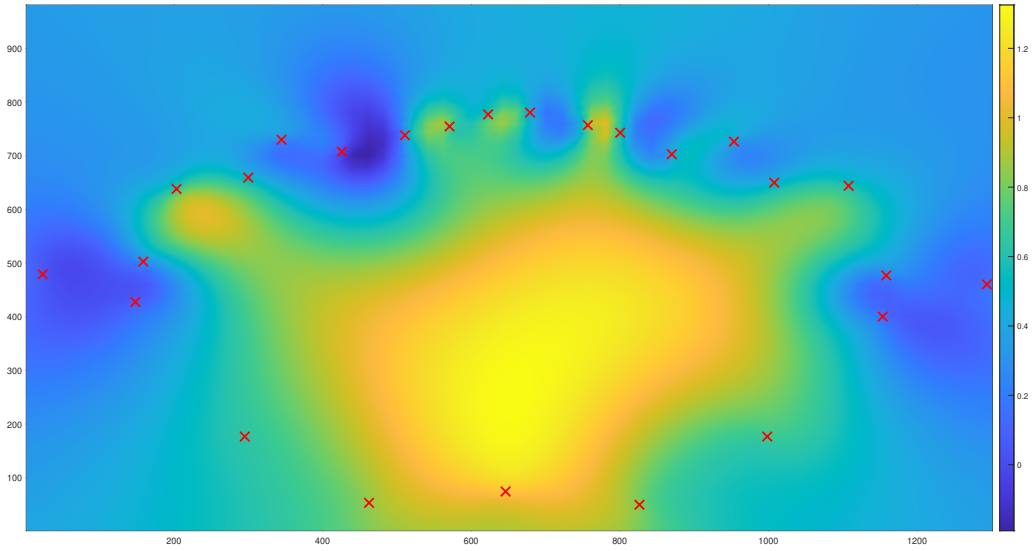


Figure 4.14: Deformation gradient of the mapping given by the TPS seen in Figure 4.13 where the landmarks of the undeformed crab are shown as red crosses.

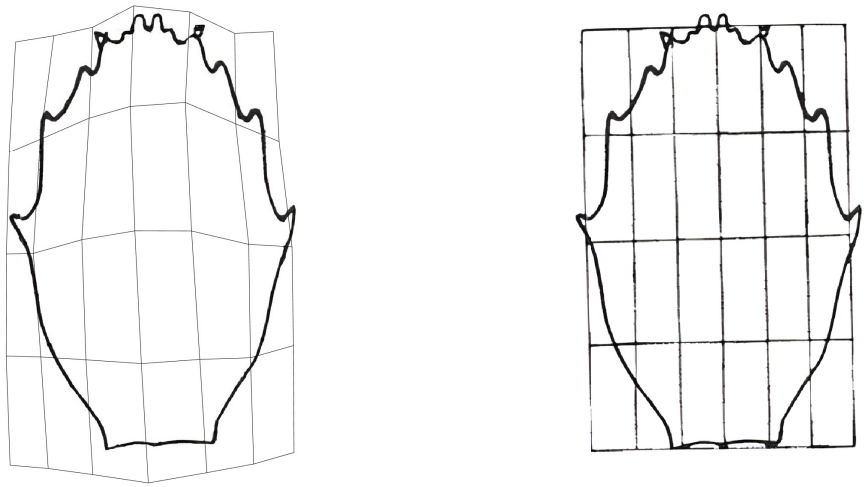


Figure 4.15: Final comparison between Thompson's Cartesian grid transformation between two crabs and our TPS generated version.

Conclusion

In summary we have fully rederived the TPS system of equations that allows us to create a mapping between biological forms whilst interpolating homologous points. This mapping acts as a deformation and we have calculated the deformation gradient and its polar decomposition to quantify how areas and line elements change under the map. This was useful in seeing where and how much deformation was occurring when comparing related forms. These mappings were then applied to D'arcy Thompson's examples and compared to his original Cartesian grid transformations. We conclude that he had the correct general idea about his mappings (at leas the ones we looked at) as we created very similar grids. His mappings were not perfect however and some aspects not possible under our assumption of a smooth map as discussed earlier but this is to be expected as he was essentially just eyeballing it.

Appendix A

Important MATLAB Code

Some of these listings are quite long but are nevertheless important as they show exactly how I implemented the TPS and calculated all the necessary deformation measures using MATLAB.

Equation 2.10.Equation 2.10.

Listing A.1: The function for the fundamental solution of the biharmonic equation

```
function output = gk(x,y,xk,yk)
% Solution for the biharmonic
% Parameters are coordinates (x,y) and point of impulse (xk,yk)
output = (((x - xk).^2 + (y - yk).^2) .* 1/2 .* log((x - xk).^2 +
    (y - yk).^2));
output(isnan(output)) = 0; % accounts for variable type erros
end
```

Listing A.2: The function for the partial derivative with respect to x of Equation 2.10

```
function output = gkprimex(x,y,xk,yk)
% Solution for the biharmonic
% Parameters are coordinates (x,y) and point of impulse (xk,yk)
output =  2 .* (x - xk) .* (log((x - xk).^2 + (y - yk).^2) + 1);
output(isnan(output)) = 0; % accounts for variable type erros
end
```

Listing A.3: The function for the partial derivative with respect to y of Equation 2.10

```
function output = gkprimey(x,y,xk,yk)
```

```
% Solution for the biharmonic
% Parameters are coordinates (x,y) and point of impulse (xk,yk)
output = 2 .* (y - yk) .* (log((x - xk).^2 + (y - yk).^2) + 1);
output(isnan(output)) = 0; % accounts for variable type erros
end
```

Listing A.4: Code generating the TPS mapping along with all its deformation measures of the kite example seen in subsection 3.2.3.

```
function KiteMappings
m = 4; % number of points mapped
A = zeros(m);
fineness = 0.2;
% starting coords
X1 = [0,1]; % top
X2 = [1,0]; % right
X3 = [-1,0]; % left
X4 = [0,-1]; % bottom

% mapped to coords
x1 = [0,1]; % top
x2 = [1,0]; % right
x3 = [-1,0]; % left
x4 = [0,-2]; % bottom

Xi = [X1;X2;X3;X4];
xi = [x1;x2;x3;x4];

% generating matrix of eqns
for i = 1:m
    for j = 1:m
        A(i,j) = gk(Xi(i,1),Xi(i,2),Xi(j,1),Xi(j,2));
    end
end

```

```
[coord1,coord2] = meshgrid(-3:fineness:3,-3:fineness:3);
[coord1surf,coord2surf] = meshgrid(-5:fineness:5,-5:fineness:5);

F1 = linsolve(A,xi(:,1));
H1 = linsolve(A,xi(:,2));

% ----CALCULATING SURFACES----
u_surf = 0;
for k = 1:m
    u_surf = u_surf + F1(k)*gk(coord1surf,coord2surf,Xi(k,1),Xi(k
       ,2));
end

v_surf = 0;
for k = 1:m
    v_surf = v_surf + H1(k)*gk(coord1surf,coord2surf,Xi(k,1),Xi(k
       ,2));
end

% plotting surfaces
figure
subplot(1,2,1)
surf(coord1surf,coord2surf,u_surf)
hold on
for i = 1:m
    plot3(Xi(i,1),Xi(i,2),xi(i,1),"o","MarkerFaceColor","k",'
       MarkerEdgeColor','k',"MarkerSize",10)
end
hold off
title('x coordinate surface')
xlabel('x')
ylabel('y')
set(gca,'fontsize',15)
```

```
57
58
59  subplot(1,2,2)
60  surf(coord1surf,coord2surf,v_surf)
61  hold on
62  for i = 1:m
63      plot3(Xi(i,1),Xi(i,2),xi(i,2),"o","MarkerFaceColor","k",'
        MarkerEdgeColor','k',"MarkerSize",10)
64  end
65  hold off
66  title('y coordinate surface')
67  xlabel('x')
68  ylabel('y')
69  set(gca,'fontsize',15)
70  % ----CALCULATING MAPPINGS----
71  B = ones(m,3);
72
73  for i = 1:m
74      B(i,2) = Xi(i,1);
75      B(i,3) = Xi(i,2);
76  end
77
78  C = [A,B;transpose(B),zeros(3)];
79
80  F = linsolve(C,[xi(1:m,1);0;0;0]); % xi(1:m) x coords
81  H = linsolve(C,[xi(1:m,2);0;0;0]); % xi(1:m) y coords
82
83  u = 0;
84  for k = 1:m
85      u = u + F(k)*gk(coord1,coord2,Xi(k,1),Xi(k,2));
86  end
87
88  % adding affine part
```

```
u = u + F(m+1) + F(m+2)*coord1 + F(m+3)*coord2;

v = 0;
for k = 1:m
    v = v + H(k)*gk(coord1,coord2,Xi(k,1),Xi(k,2));
end

v = v + H(m+1) + H(m+2)*coord1 + H(m+3)*coord2;

% ----BENDING ENERGY----
bending_energy_u = transpose(F(1:m))*A*F(1:m)
bending_energy_v = transpose(H(1:m))*A*H(1:m)


% ----CALCULATING LINE ELEMENTS----
uprimex = 0;
for k = 1:m
    uprimex = uprimex + F(k)*gkprimex(coord1,coord2,Xi(k,1),Xi(k
       ,2));
end
uprimex = uprimex + F(m+2);

uprimey = 0;
for k = 1:m
    uprimey = uprimey + F(k)*gkprimey(coord1,coord2,Xi(k,1),Xi(k
       ,2));
end
uprimey = uprimey + F(m+3);

vprimex = 0;
for k = 1:m
    vprimex = vprimex + H(k)*gkprimex(coord1,coord2,Xi(k,1),Xi(k
       ,2));
```

```
119 end
120 vprimex = vprimex + H(m+2);
121
122 vprimey = 0;
123 for k = 1:m
124     vprimey = vprimey + H(k)*gkprimey(coord1,coord2,Xi(k,1),Xi(k
        ,2));
125 end
126 vprimey = vprimey + H(m+3);
127
128 % defining how long dX and dY are
129 dX = ones(size(coord1)) * fineness;
130 dY = zeros(size(coord2)) * fineness;
131
132 % line element transformations
133 dx = (uprimex .* dX) + (uprimey .* dY);
134 dy = (vprimex .* dX) + (vprimey .* dY);
135
136 % the jacobian determinant
137 J = uprimex.*vprimey - uprimey.*vprimex;
138
139 dx_stretch = zeros(size(dX));
140 dy_stretch = zeros(size(dY));
141 dx_rotate = zeros(size(dX));
142 dy_rotate = zeros(size(dY));
143 x_principal_stretch = zeros(size(dX));
144 y_principal_stretch = zeros(size(dY));
145 x1_principal_stretch_vec = zeros(size(dX));
146 y1_principal_stretch_vec = zeros(size(dY));
147 x2_principal_stretch_vec = zeros(size(dX));
148 y2_principal_stretch_vec = zeros(size(dY));
149 volume_change = zeros(size(dX));
150 % Jacobian matrix at a certain coordinate (i.e. certain area
```

```
    element)
for i = 1:size(dX)
    for j = 1:size(dY)
        JM = [uprimex(i,j),uprimey(i,j);vprimex(i,j),vprimey(i,j)
            ]; % Jacobian matrix
        c = transpose(JM)*JM; % cauchy-green deformation gradient
            tensor
        [eigvecs,eigvals] = eig(c);
        x_principal_stretch(i,j) = sqrt(eigvals(1,1));
        x1_principal_stretch_vec(i,j) = eigvecs(1,1);
        y1_principal_stretch_vec(i,j) = eigvecs(1,2);
        x2_principal_stretch_vec(i,j) = eigvecs(2,1);
        y1_principal_stretch_vec(i,j) = eigvecs(2,2);
        y_principal_stretch(i,j) = sqrt(eigvals(2,2));
        volume_change(i,j) = x_principal_stretch(i,j)*
            y_principal_stretch(i,j); % volume change using princp
            stretches
        U = sqrtm(c); % stretch matrix
        R = U\JM; % rotation matrix
        dx_stretch(i,j) = U(1,1)*dX(i,j) + U(1,2)*dY(i,j); %
            stretching part of line elements
        dy_stretch(i,j) = U(2,1)*dX(i,j) + U(2,2)*dY(i,j);
        dx_rotate(i,j) = R(1,1)*dX(i,j) + R(1,2)*dY(i,j); %
            rotation part of line elements
        dy_rotate(i,j) = R(2,1)*dX(i,j) + R(2,2)*dY(i,j);
    end
end


% ----UNTRANSFORMED PLOTS----
% plotting kite
figure
surf(coord1,coord2,zeros(size(u)),'FaceAlpha',0.01)
```

```
grid off
hold on
for k = 1:m
    plot3(Xi(k,1),Xi(k,2),0,"o","MarkerFaceColor","k",'
       MarkerEdgeColor','k',"MarkerSize",5);
end
% plotting line elements
%quiver(coord1,coord2,dX,dY,1,"Color",'b')
hold off

% visuals and axis
view(0,90)
set(gca,'FontSize',20)
pbaspect([1,1,1])
xlim([-3,3])
ylim([-3,3])


% ----TRANSFORMED PLOTS----
% plotting transformed kite
figure
surf(u,v,zeros(size(u)),'FaceAlpha',0.01)
hold on
for k = 1:m
    plot3(xi(k,1),xi(k,2),0,"o","MarkerFaceColor",'k','
       MarkerEdgeColor',"k","MarkerSize",5);
end
% plotting line element changes
%quiver(u,v,dX,dY,'Color','k') % undeformed line element
%quiver(u,v,dx,dy,"Color",'r') % plotting overall deformed line
   element change
quiver(u,v,dx_rotate,dy_rotate,"Color",'b') % plotting deformed
   line element rotations
```

```
206 %quiver(u,v,dx_stretch,dy_stretch,'Color','g') % plotting
       deformed line element stretches
207
208 % plotting principal stretches
209 %quiver(u,v,x1_principal_stretch_vec,y1_principal_stretch_vec,"
       Color",'b')
210 %quiver(u,v,x2_principal_stretch_vec,y2_principal_stretch_vec,"
       Color",'r')
211
212 hold off
213
214 % visuals and axis
215 view(0,90)
216 set(gca,'FontSize',20)
217 pbaspect([1,1,1])
218 xlabel('x')
219 ylabel('y')
220 set(get(gca,'ylabel'),'rotation',0)
221 grid off
222 xlim([-3,3])
223 ylim([-3,3])
224
225 % plotting volume change via jacobian
226 figure
227 pcolor(-3:fineness:3,-3:fineness:3,J);
228 hold on
229 %quiver(u,v,dx,dy,1,"Color",'r')
230 for k = 1:m
231     plot3(Xi(k,1),Xi(k,2),0,"o","MarkerFaceColor","k",'
          MarkerEdgeColor','k',"MarkerSize",5);
232 end
233 hold off
234 shading interp;
```

```
colorbar
grid off
pbaspect([1,1,1])
xlabel('x')
ylabel('y')
set(get(gca,'ylabel'),'rotation',0)
set(gca,'FontSize',20)

% plotting volume change via principal stretches (indentical to
   the jacobian of course)
figure
pcolor(-3:fineness:3,-3:fineness:3,volume_change);
hold on
for k = 1:m
    plot3(Xi(k,1),Xi(k,2),0,"o","MarkerFaceColor","k",'
       MarkerEdgeColor','k',"MarkerSize",5);
end
hold off
shading interp;
colorbar
grid off
pbaspect([1,1,1])
xlabel('x')
ylabel('y')
set(get(gca,'ylabel'),'rotation',0)
set(gca,'FontSize',20)

end
```

Listing A.5: Code generating the TPS mapping along with all its deformation measures of the D'arcy Thompson haddock and plaice.

```
function HaddockAndPlaiceInterpolation
% loading landmark data and declaring landmark variables
img1 = imread("fish01edit.jpg");
```

```
4  img2 = imread("fish02edit.jpg");
5
6  load("fin1data.mat");
7  load("fin2data.mat");
8  load("fin3data.mat");
9  load("fin4data.mat");
10 load("fin5data.mat");
11 load("fishmouthdata.mat")
12 load("fisheyedata.mat")
13 xi = [fin1x;fin2x;fin3x;fin4x;fin5x];%;mouthx;eyex];
14 yi = [fin1y;fin2y;fin3y;fin4y;fin5y];%;mouthy;eyey];
15
16 Xi = [fin1X;fin2X;fin3X;fin4X;fin5X];%;mouthX;eyeX];
17 Yi = [fin1Y;fin2Y;fin3Y;fin4Y;fin5Y];%;mouthY;eyeY];
18
19 fineness = 5; % grid spacing
20
21 % ----GENERATING MATRIX EQAUTIONS----
22 m = length(xi); % number of points mapped
23 A = zeros(m);
24
25 for i = 1:m
26     for j = 1:m
27         A(i,j) = gk(xi(i,1),yi(i,1),xi(j,1),yi(j,1));
28     end
29 end
30
31 F1 = linsolve(A,Xi(:,1));
32 H1 = linsolve(A,Yi(:,1));
33
34 [coord1,coord2] = meshgrid(1:fineness:352,1:fineness:274);
35
36 % ----CALCULATING SURFACES----
```

```
u_surf = 0;
for k = 1:m
    u_surf = u_surf + F1(k)*gk(coord1,coord2,xi(k,1),yi(k,1));
end

v_surf = 0;
for k = 1:m
    v_surf = v_surf + H1(k)*gk(coord1,coord2,xi(k,1),yi(k,1));
end

% plotting mappings as surfaces in their own right
surf(coord1,coord2,u_surf)
hold on
for i = 1:m
    plot3(xi(i,1),yi(i,1),Xi(i,1),"o","MarkerFaceColor","k",'
       MarkerEdgeColor','k',"MarkerSize",5)
end
hold off
title('x coords')

surf(coord1,coord2,v_surf)
hold on
for i = 1:m
    plot3(xi(i,1),yi(i,1),Yi(i,1),"o","MarkerFaceColor","k",'
       MarkerEdgeColor','k',"MarkerSize",5)
end
hold off
title('y coords')

% ----BENDING ENERGY----
bending_energy_u = transpose(F1(1:m))*A*F1(1:m)
bending_energy_v = transpose(H1(1:m))*A*H1(1:m)

```

```
% ----CALCULATING MAPPINGS----
B = ones(m,3);

for i = 1:m
    B(i,2) = xi(i,1);
    B(i,3) = yi(i,1);
end

C = [A,B;transpose(B),zeros(3)];

F = linsolve(C,[Xi(1:m,1);0;0;0]);
H = linsolve(C,[Yi(1:m,1);0;0;0]);

u = 0;
for k = 1:m
    u = u + F(k)*gk(coord1,coord2,xi(k),yi(k));
end


% adding affine part
u = u + F(m+1) + F(m+2)*coord1 + F(m+3)*coord2;

v = 0;
for k = 1:m
    v = v + H(k)*gk(coord1,coord2,xi(k),yi(k));
end

v = v + H(m+1) + H(m+2)*coord1 + H(m+3)*coord2;

% ----CALCULATING LINE ELEMENTS----
uprimex = 0;
for k = 1:m
    uprimex = uprimex + F(k)*gkprimex(coord1,coord2,xi(k),yi(k));
```

```
end
uprimex = uprimex + F(m+2);

uprimey = 0;
for k = 1:m
    uprimey = uprimey + F(k)*gkprimey(coord1,coord2,xi(k),yi(k));
end
uprimey = uprimey + F(m+3);

vprimex = 0;
for k = 1:m
    vprimex = vprimex + H(k)*gkprimex(coord1,coord2,xi(k),yi(k));
end
vprimex = vprimex + H(m+2);

vprimey = 0;
for k = 1:m
    vprimey = vprimey + H(k)*gkprimey(coord1,coord2,xi(k),yi(k));
end
vprimey = vprimey + H(m+3);

%JM = [uprimex,uprimey;vprimex,vprimey]; % jacobian matrix

J = uprimex.*vprimey - uprimey.*vprimex; % the jacobian
   determinant

dX = ones(size(coord1)) * fineness; % defining how long dX and dY
    are
dY = ones(size(coord2)) * fineness;

dx = (uprimex .* dX) + (uprimey .* dY); % applying jacobian
   component wise to all dX and dY
dy = (vprimex .* dX) + (vprimey .* dY);
```

```
131
132 % ----UNTRANSFORMED PLOTS----
133 % Plotting untransformed fish and grid
134 warp(coord1,coord2,zeros(size(u)),img1)
135 hold on
136 surf(coord1,coord2,zeros(size(u)),'FaceAlpha',0)
137 hold off
138
139 % plotting landmarks
140 hold on
141 for k = 1:m
142     plot3(xi(k,1),yi(k,1),0,"x","MarkerSize",15,"MarkerEdgeColor
        ",'r',"LineWidth",2);
143 end
144 hold off
145
146 %visuals
147 view(0,90)
148 set(gca,'FontSize',20)
149 pbaspect([1,1,1])
150 set(gca,'YDir','normal');
151 set(get(gca,'ylabel'),'rotation',0)
152 axis equal tight
153 xlabel('x')
154 ylabel('y')
155
156 imshow(img2)
157 % plotting landmarks
158 hold on
159 for k = 1:m
160     plot3(Xi(k,1),Yi(k,1),0,"x","MarkerSize",15,"MarkerEdgeColor
        ",'b',"LineWidth",2);
161 end
```

```
hold off

%visuals
view(0,90)
set(gca,'FontSize',20)
pbaspect([1,1,1])
set(gca,'YDir','normal');
set(get(gca,'ylabel'),'rotation',0)
axis equal tight

% ----TRANSFORMED PLOTS----
% Plotting transformed fish and grid
warp(u,v,zeros(size(u)),img1)
hold on
surf(u,v,zeros(size(u)),'FaceAlpha',0)
hold off

% plotting landmarks
hold on
for k = 1:m
    plot3(Xi(k,1),Yi(k,1),0,"x","MarkerSize",15,"MarkerEdgeColor
       ','b',"LineWidth",2);
end
hold off

% plotting line elements
hold on
for k = 1:m
    %quiver(u,v,dX,dY,0.5,"Color",'b')
end
hold off

%visuals
```

```
194 view(0,90)
195 set(gca,'FontSize',20)
196 pbaspect([1,1,1])
197 set(gca,'YDir','normal');
198 set(get(gca,'ylabel'),'rotation',0)
199 axis equal tight
200 xlabel('x')
201 ylabel('y')
202
203 %----D'ARCY THOMPSON STYLE SUPERIMPOSITIONS----
204 % desired fish with image grid
205 imshow(img2)
206 hold on
207 surf(u,v,zeros(size(u)),'FaceAlpha',0)
208 hold off
209
210 %visuals
211 view(0,90)
212 set(gca,'FontSize',20)
213 pbaspect([1,1,1])
214 set(gca,'YDir','normal');
215 set(get(gca,'ylabel'),'rotation',0)
216 axis equal tight
217
218 %----DEFORMATION ANALYSIS----
219 pcolor([1:fineness:352],[1:fineness:274],J)
220 shading interp;
221 colorbar;
222 hold on
223 for k = 1:m
224     plot3(xi(k,1),yi(k,1),0,"x","MarkerSize",15,"MarkerEdgeColor
        ",'r',"LineWidth",2);
225 end
```

```
hold off

%----SIDE BY SIDE----
warp(u,v,zeros(size(u)),img1)
%visuals
view(0,90)
set(gca,'FontSize',20)
pbaspect([1,1,1])
set(gca,'YDir','normal');
set(get(gca,'ylabel'),'rotation',0)
axis equal tight
set(gca,'visible','off')

imshow(img2)
%visuals
view(0,90)
set(gca,'FontSize',20)
pbaspect([1,1,1])
set(gca,'YDir','normal');
set(get(gca,'ylabel'),'rotation',0)
axis equal tight
end
```

Listing A.6: Code generating a general TPS mapping given input landmark data and source image.

```
function [u,v] = TPS_NO_LINE_ELEMENTS(xi,yi,Xi,Yi,fineness,img)
% Creates a TPS interpolation (with no landmark crosses) given
   landmark data
% ----GENERATING MATRIX EQAUTIONS----
m = length(xi); % number of points mapped
A = zeros(m);

for i = 1:m
    for j = 1:m
```

```
9           A(i,j) = gk(xi(i,1),yi(i,1),xi(j,1),yi(j,1));
10      end
11  end
12
13  B = ones(m,3);
14
15  for i = 1:m
16      B(i,2) = xi(i,1);
17      B(i,3) = yi(i,1);
18  end
19
20  C = [A,B;transpose(B),zeros(3)];
21
22  F = linsolve(C,[Xi(1:m,1);0;0;0]);
23  H = linsolve(C,[Yi(1:m,1);0;0;0]);
24
25  [coord1,coord2] = meshgrid(1:fineness:length(img),1:fineness:
        height(img));
26
27  % ----CALCULATING MAPPINGS----
28  u = 0;
29  for k = 1:m
30      u = u + F(k)*gk(coord1,coord2,xi(k),yi(k));
31  end
32
33  u = u + F(m+1) + F(m+2)*coord1 + F(m+3)*coord2;
34
35  v = 0;
36  for k = 1:m
37      v = v + H(k)*gk(coord1,coord2,xi(k),yi(k));
38  end
39
40  v = v + H(m+1) + H(m+2)*coord1 + H(m+3)*coord2;
```

```
41
42 end
```

Listing A.7: Code for the time varying TPS seen in section 4.3.

```
1  function [u,v] = TPS_NO_LINE_ELEMENTS(xi,yi,Xi,Yi,fineness,img)
2  % Creates a TPS interpolation (with no landmark crosses) given
       landmark data
3  % ----GENERATING MATRIX EQAUTIONS----
4  m = length(xi); % number of points mapped
5  A = zeros(m);
6
7  for i = 1:m
8      for j = 1:m
9          A(i,j) = gk(xi(i,1),yi(i,1),xi(j,1),yi(j,1));
10     end
11 end
12
13 B = ones(m,3);
14
15 for i = 1:m
16     B(i,2) = xi(i,1);
17     B(i,3) = yi(i,1);
18 end
19
20 C = [A,B;transpose(B),zeros(3)];
21
22 F = linsolve(C,[Xi(1:m,1);0;0;0]);
23 H = linsolve(C,[Yi(1:m,1);0;0;0]);
24
25 [coord1,coord2] = meshgrid(1:fineness:length(img),1:fineness:
       height(img));
26
27 % ----CALCULATING MAPPINGS----
28 u = 0;
```

```
for k = 1:m
    u = u + F(k)*gk(coord1,coord2,xi(k),yi(k));
end

u = u + F(m+1) + F(m+2)*coord1 + F(m+3)*coord2;

v = 0;
for k = 1:m
    v = v + H(k)*gk(coord1,coord2,xi(k),yi(k));
end

v = v + H(m+1) + H(m+2)*coord1 + H(m+3)*coord2;

end
```

Bibliography

[1] P Ball. In retrospect: On growth and form. *Nature*, (494):32–33, 2013.

[2] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:567 – 585, 1989.

[3] Fred L Bookstein and William DK Green. A feature space for edgels in images with landmarks. *Journal of Mathematical imaging and vision*, 3(3):231–261, 1993.

[4] D Courant, R; Hilbert. *Methods of Mathematical Physics. Vol. I*, page 192. New York: Interscience Publishers, Inc., 1953.

[5] Albrecht Dürer. Hierinn sind begriffen vier bucher von menschlicher proportion durch albrechten durer von nurerberg. 1528.

[6] Fridtjov Irgens. *Continuum Mechanics*, page 661. Springer Berlin, Heidelberg, 1982.

[7] Leslie F. Marcus. Proceedings of the michigan morphometrics workshop, chapter 4, traditional morphometrics. 1990.

[8] Ryan G. McClarren. Chapter 10 - interpolation. In Ryan G. McClarren, editor, *Computational Nuclear Engineering and Radiological Science Using Python*, pages 173–192. Academic Press, 2018.

[9] Knötel D. Baum D. Scholtz, G. D'arcy w. thompson's cartesian transformations: a critical evaluation. *Zoomorphology*, (139):293–308, 2020.

[10] D'Arcy Wentworth Thompson. On growth and form. 1917.