A High Quality Solver for Diffusion Curves

Jasper van de Gronde

December 20, 2010

Contents

1 Introduction **1**
- 1.1 Image representation . . . 1
- 1.2 Implementations . . . 2
- 1.3 Goal . . . 3
- 1.4 Defining diffusion curves . . . 3
- 1.5 Previous work . . . 4

2 Existing algorithms **8**
- 2.1 Gauss-Seidel . . . 8
- 2.2 Block-based Gauss-Seidel . . . 8
- 2.3 Multigrid . . . 10
- 2.4 Variable stencil size . . . 10

3 Errors and residuals **12**
- 3.1 Surface methods . . . 12
- 3.2 Boundary methods . . . 14

4 Analytic Element Method **19**
- 4.1 One dimension . . . 19
- 4.2 Two dimensions . . . 20
- 4.3 Integrating the fundamental solution . . . 22
  - 4.3.1 Near-field . . . 23
  - 4.3.2 Far-field . . . 25
  - 4.3.3 General linear segments . . . 27
- 4.4 Adding discontinuities back in . . . 28
- 4.5 Results . . . 30

5 Conclusion **33**
- 5.1 Further work . . . 33
- 5.2 Acknowledgements . . . 35

A Derivatives **36**

Bibliography **38**

Chapter 1

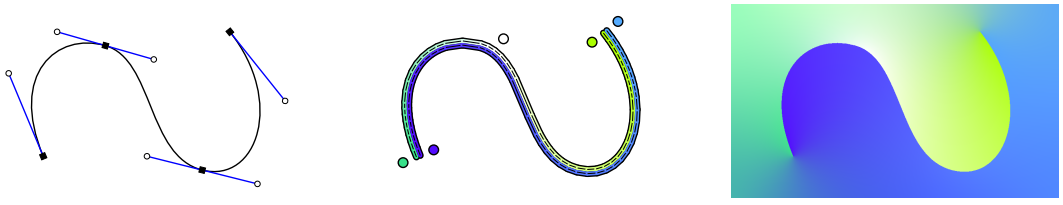
Introduction

Draw a few curves, assign color gradients to either side and the surrounding space is automatically filled in smoothly and continuously (see figure 1.1). That is the artistic promise of diffusion curves [18], an exciting new vector primitive designed to allow creation of complex gradients in a workflow that focusses on the contours in an image [17]. In contrast, gradient meshes, currently the most advanced type of gradient available to artists, require a careful subdivision of the filled region into a mesh, with colors assigned to all nodes in the mesh. With diffusion curves no mesh is needed, just the discontinuities that should be present in the image.

1.1 Image representation

Diffusion curves also offer a nice representation of existing images, as many real-world images consist of (large) smooth regions with discontinuous boundaries. In fact, Elder [6] addressed the question of completeness of edges as an image representation. Elder found edges (in combination with a scale related parameter) to indeed contain virtually all (perceptually relevant) information in an image. This makes edges an interesting general purpose image representation, and thereby diffusion curves as well, as they are in essence a vector version of the edge based representation created by Elder.

In Orzan et al. [18] and [17] a diffusion curves based representation is derived automatically, allowing further manipulation and providing an effective alternative to systems like ArDeCo [13], which automatically converts an image to a vector representation using more traditional kinds of gradients. Given the discussion in Elder [6] it may also be interesting to use automatic extraction of diffusion curves as a sparse representation for purposes of compression or in compressed sensing [31]. For example, in [18] a $946 \times 633 = 598\,818$ pixel



(a) (b) (c)

Figure 1.1: Diffusion curves can be defined using Bézier curves (a) and color values on either side (b). The result (c) is a smoothly shaded image that would be difficult to create using other methods. (Example inspired by Orzan et al. [18].)

(a) Original (b) Traced result

Figure 1.2: Result of automatic diffusion curve extraction from Orzan et al. [18]. The original image (a) has $946 \times 633 = 598\,818$ pixels, the traced image (b) uses only 16 816 data points.

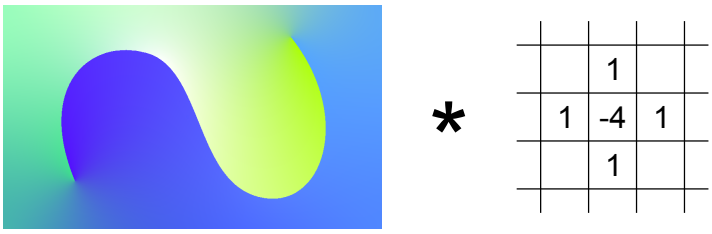


Figure 1.3: Convolution by the shown kernel is the discrete equivalent of applying the Laplace operator $\Delta = \nabla \cdot \nabla = \frac{\partial^2}{\partial x_0^2} + \frac{\partial^2}{\partial x_1^2}$.

image (figure 1.2) is well represented using a mere 16 816 geometric, color and blur control points. Although it is not known yet whether this would translate to good compression (as that would also require looking at quantization methods and such) it does suggest the ability to sparsely represent an image.

1.2 Implementations

Existing implementations for rendering diffusion curves have relied on discretizing the Laplace equation $\Delta u = \nabla \cdot \nabla u = 0$ on a regular grid (see figure 1.3). This leads to a simple, extremely sparse, system of linear equations. In Orzan [17], Orzan et al. [18], Elder [6] and Bezerra et al. [1] a multigrid solver is used to ensure fast convergence, but it requires careful handling of boundary conditions and, as noted in Jeschke et al. [10, fig. 2], can suffer from certain artifacts, especially in the presence of small image features. These problems are somewhat alleviated by the improved boundary handling and initialization in Jeschke et al. [10] (by means of a distance transform).

The algorithm developed by Jeschke et al. is similar in spirit to the multigrid solver used by others, except that instead of creating multiple grids at different scales it locally adapts the size of the Laplace kernel. This makes for an easier implementation and was found to be faster and more precise than the solver used in Orzan et al. [18], although both solvers reach interactive speeds for moderate image sizes (512×512, on a GPU). With easier boundary handling, less artifacts and better performance this solution seems perfect (and in practice it is quite nice), except that all solvers mentioned so far use a fixed number of iterations and in doing so do not even reach 8 bit precision. If instead they iterate until a certain error (or rather residual) threshold is reached they are suddenly not that fast anymore. This is shown to be linked to a fundamental problem with trying to control error in the image through minimizing the residual of the Laplacian.

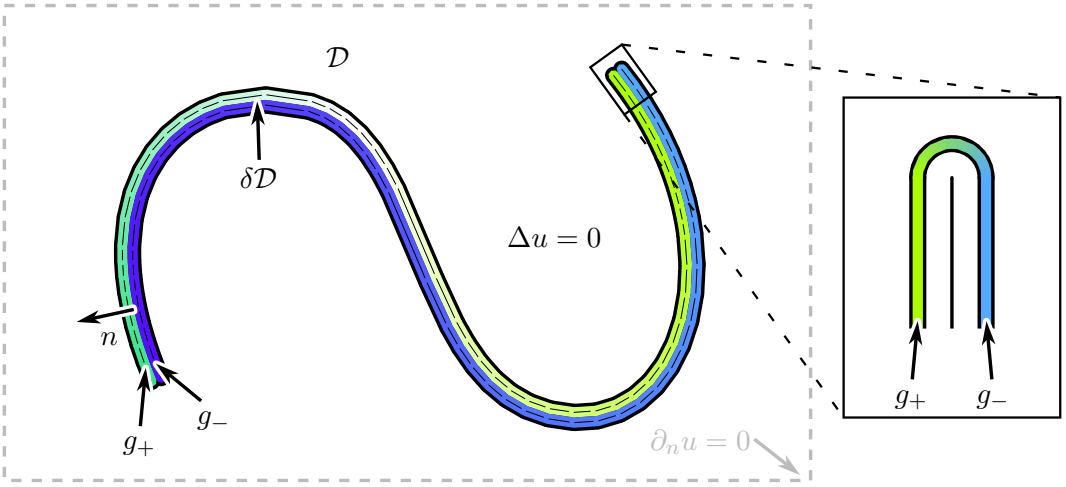


Figure 1.4: Diffusion curves give Dirichlet boundary conditions along $\delta\mathcal{D}$, in the domain $\mathcal{D}$ the Laplace equation is satisfied and we choose to follow the convention to enforce a zero normal derivate boundary condition along the image boundary (a Neumann boundary condition). Conceptually the curves are seen as the boundary of a stroked curve (with round caps) in the limit as the stroke width goes to zero, as illustrated on the right.

1.3 Goal

In this thesis I will develop a new method for rendering diffusion curves based on analytic boundary elements. This is motivated by the desire for a more precise solution, even for images with complicated boundaries and widely varying scales. Part of the newly developed solver can also be used in combination with existing grid based solvers to make coping with boundaries easier. Such developments are expected to increase the viability of diffusion curves as a commonly accepted drawing primitive and pave the way for new quality-critical applications.

1.4 Defining diffusion curves

Diffusion curves are curves from which color is diffused to the surrounding area. The resulting image u is the solution to the Laplace equation in the image domain $\mathcal{D}$ with colors defined along the diffusion curve(s) as boundary conditions:

$$\begin{aligned} \Delta u(x) &= 0 && (x \in \mathcal{D}) \\ \lim_{d \downarrow 0} u(x + d\, n(x)) &= g_+(x) && (x \in \delta\mathcal{D}) \\ \lim_{d \downarrow 0} u(x - d\, n(x)) &= g_-(x), && (x \in \delta\mathcal{D}) \end{aligned} \tag{1.1}$$

where $n(x)$ is the normal of the boundary curve at x and $g_\pm$ the color on either side of the curve (see figure 1.4). Without giving another boundary condition for what happens away from the diffusion curves the problem above is under-constrained. In the original definition, the derivative in the direction of the normal is made zero at the edges of the rendered bitmap. Such a boundary condition is called a (zero) Neumann boundary condition. Note that each color channel is handled separately.

Note that having a different color on either side of the curve gives rist to a discontinuity at the ends of the curve. To give meaning to this, a curve is imagined to be the limit of

a stroked curve (with a round cap) as the stroke width goes to zero (see the box on the right in figure 1.4). So the color depends on the angle from which you "look" at it.

As colors can be defined on either side of each diffusion curve, they effectively represent discontinuities in the image. In the original definition a spatially varying blur is applied in a post-processing step to allow for soft discontinuities. Other extensions include solving a Poisson equation instead of the homogeneous Laplace equation. These are not covered in this thesis.

1.5 Previous work

As early as the 18$^{\text{th}}$ century scientists were interested in surfaces of minimal area and this continued throughout the 19$^{\text{th}}$ and 20$^{\text{th}}$ century. Even today research is still being done in this area. In Courant [4] and in particular Trọng Thi and Fomenko [30] an overview is given of the early history of this field and it is shown how minimizing surface area is related to solving the Laplace equation. Essentially this relies on the fact that solving the Laplace equation is mathematically equivalent to minimizing the integral of the squared gradient magnitude over the problem domain, and that the latter is a good estimate for the surface area.

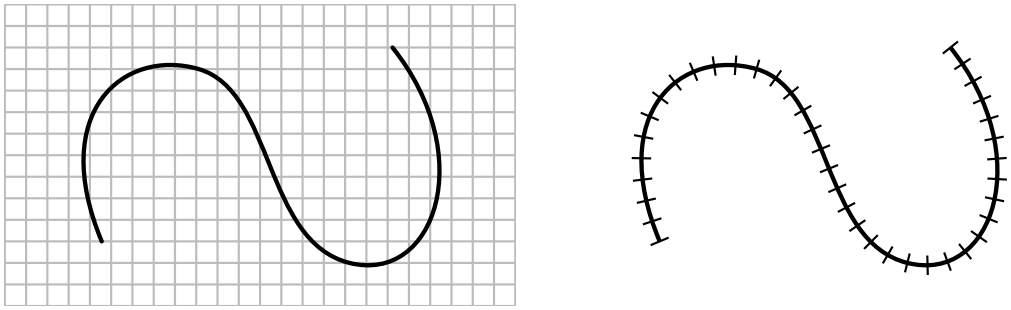
Much more recently, minimization of *curvature* was used to define "Spiro" splines [14] (the name refers to the Euler spiral used in their definition). In a sense Spiro splines are a one-dimensional higher order analog of surfaces of minimal area (which have zero mean curvature as a result of minimizing area). These curves have been integrated successfully in a number of drawing tools (e.g. Inkscape and FontForge). In section 11.5 of the original thesis [14] it was suggested that it might be interesting to look for analogues in 2D, in some sense diffusion curves are this analog[1]

In Floater [8] Mean Value Coordinates were introduced as a generalization of barycentric coordinates. Barycentric coordinates can be used to interpolate values between points in a polygon and as such can be used to serve a purpose similar to diffusion curves. They are motivated by the mean value theorem for harmonic functions (functions that satisfy the Laplace equation) and were used for image cloning in Farbman et al. [7], comparing the result directly to a result obtained by solving the Laplace equation. Interestingly, mean value coordinates can be computed much faster than a similar result based on the Laplace equation.

General solvers for the Poisson and/or Laplace equations have been studied extensively. Roughly these methods can be divided in methods that, by definition, satisfy the equations at the boundary and try to "fill in" the surface, and those methods that, again by definition, satisfy the equations everywhere but on the boundary (or outside the solution domain) and try to approximate the boundary (see figure 1.5). The first category will be called "surface methods" and the other "boundary methods". Limiting ourselves to methods that support curved boundaries as used in Diffusion Curves we can identify the following surface methods (note that I have tried to give a representative list but many more methods may still exist):

- Common (mostly iterative) solvers on a matrix-vector equation resulting from discretization on a regular grid. For exampe Gauss-Seidel iteration or the conjugate gradient method, often as part of a multigrid method [22, 23].

[1] So-called "biharmonic" functions would be an even closer 2D analog of Spiro splines, as they can take both a value and normal derivative constraint at the boundary. Biharmonic functions generalize the images generated using diffusion curves (which can be considered "harmonic" functions).



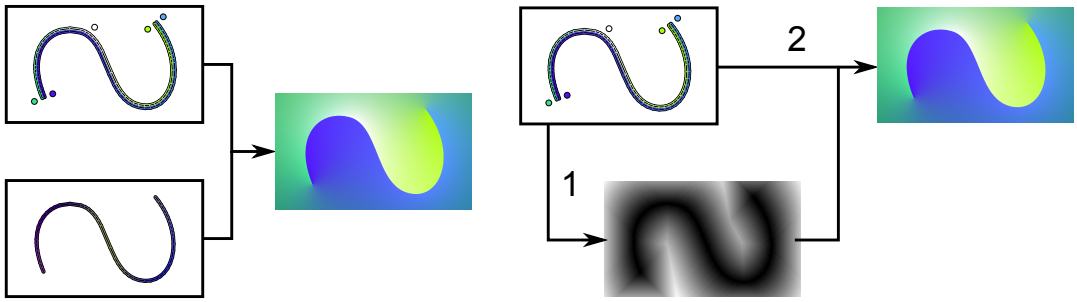
(a) Surface method (b) Boundary method

Figure 1.5: Surface methods (a) use a grid (or mesh) to cover the solution domain. In contrast, boundary methods (b) discretize the boundary, ignoring the interior of the solution domain.

- Geometric multigrid [18, 22, 23]. This speeds up convergence by discretizing the problem on multiple grids, each on a different scale. The smaller grids allow values to diffuse through the image quickly, while the larger grids provide the necessary precision in modelling the boundary.
- Algebraic multigrid [27]. Similar in spirit to the geometric multigrid method this also uses multiple scales, but instead of simply scaling a grid it groups values based on properties of the linear system that is being solved.
- Four-point method [20]. This uses a Gauss-Seidel iteration, but processes blocks of four pixels at once. A variation is also described which does this on a grid that is scaled down by a factor of two in both the horizontal and vertical direction.
- Wavelet based [29]. This tries to solve the Laplace equation by transforming it to wavelet space. This helps because there is a very effective diagonal preconditioner for the wavelet-transformed Laplace operator.
- Variable stencil size [10]. Quite similar to multigrid methods, except that it is not the grids that vary in scale, but rather the discrete Laplace kernel.
- Finite Element Method (FEM) [17]. Instead of explicitly working with a discrete representation of the problem on a regular grid, the FEM uses a continuous representation based on a triangle mesh.

Among boundary methods we have:

- Boundary Element Method (BEM) [19, 33, 34]. Although exact methods differ a bit these all try to build a solution by placing the so-called fundamental solution along the boundary and solving a linear system to determine the correct weights.
- Method of Fundamental Solutions (MFS) [15, 16]. As the fundamental solution is singular at the origin special care must be taken in doing computations on the boundary in BEMs. The MFS tries to avoid this by placing the fundamental solution some distance away from the boundary.
- Boundary Knot Method (BKM) [3]. This method also tries to avoid having to deal with singularities, but instead of moving the fundamental solution it uses different basis functions (that actually solve a slightly different problem).



(a) Scheme from Orzan et al. [18]

(b) Scheme from Jeschke et al. [10]

Figure 1.6: The original scheme (a) put the color constraints slightly away from the boundary, using a gradient constraint on the boundary to maintain sharp transitions. The improved scheme (b) first does a distance transform, which is used to vary the stencil size, as well as to initialize each pixel with the color of the closest boundary point.

- Analytic Element Method (AEM) [9, 12, 26]. Essentially the same as the boundary element method, except that emphasis is put on analytically integrating the fundamental solution along the boundary.

Note that all current algorithms for rendering diffusion curves are based on surface methods, while I propose to use a boundary method, specifically the analytic element method. Just like most boundary methods it is in principle resolution independent and the maximum error in the image is found on the boundary and is minimized directly (assuming Dirichlet boundary conditions), as opposed to minimizing the residual of the Laplacian.

Diffusion curves were first introduced by Orzan et al. [18] and described more extensively in [17]. Curves were defined to carry both color and blur constraints, with the color constraints on either side of the curve. Both color and blur values were diffused (see figure 1.6a) and then the color image was reblurred. The solver was a multigrid solver and boundary conditions were put a few pixels away from the actual curves. Among other things this required that gradient information was diffused as well. Diffusion curves were used not only for interactive drawing purposes, but also for automatic vectorization of existing images.

Jeschke et al. [10] introduced two improvements. First of all, instead of putting the boundary colors a few pixels away from the boundary and using gradient information each pixel got the color of the correct side(!) of the closest boundary pixel as part of a distance transform. By making sure that the pixels at the boundary never change, and using an initialization based on a distance transform, the need for having a complicated positioning of boundary colors and diffusing gradient information was eliminated, leading to the scheme shown in figure 1.6b.

The second improvement in Jeschke et al. [10] was a simpler (but faster) iteration. Instead of employing multiple grids at different scales the size of the Laplace kernel was made to adapt locally to the distance to the nearest boundaries (hence the distance transform mentioned before). Apparently the type of distance transform used is not particularly important, although informal tests did indicate a small advantage in using the conservative L_∞-norm[2]. This scheme has performance properties similar to a multigrid solver (as shown later) but is less complicated to implement and has less problems in switching

[2]It is not entirely unexpected that the L_∞-norm performs slightly better, as it forces the kernel radius to be smaller near a corner compared to other (pseudo-)norms and this is precisely where the solution is usually the least smooth (making the four-point kernel less of a suitable approximation to the ideal circle).

between scales (a multigrid solver must use a carefully chosen upsampling operator).

Finally, Takayama et al. [28] extend diffusion curves to 3D and Bezerra et al. [1, 2] introduce a number of extensions to 2D diffusion curves, in particular the ability to control the diffusion in a number of ways. Instead of always having a color boundary condition on a curve Bezerra et al. also allow a gradient boundary condition for example. And instead of solving the homogenous Laplace equation they allow solving the Poisson equation, which allows for things like swirls. Also, instead of having color diffuse from all points on every curve with equal "force" they allow a weight to be assigned to each point by a mechanism analogous to homogeneous coordinates (which they call homogeneous colors).

Chapter 2

Existing algorithms

In this section it is assumed that both sides of any diffusion curve have the same color, section 4.4 shows how discontinuities can be handled for *any* solver.

2.1 Gauss-Seidel

Gauss-Seidel (or Jacobi) iteration is probably the easiest way by far to solve a discretized version of the Laplace equation with boundary conditions. Using a straightforward square grid it simply requires setting each pixel to the average of its surrounding pixels, except if the pixel belongs to the boundary, then it is left alone or the equation is modified to incorporate a zero normal derivative boundary condition. Given enough iterations this will converge. The main problem with this method is that it needs *a lot* of iterations (millions for even a modestly sized image).

2.2 Block-based Gauss-Seidel

One way to reduce the number of iterations is to take blocks of pixels as the basic unit. Instead of "solving" for one pixel at a time an entire block (generally a row or column) of pixels is solved for in the sense that locally that block of pixels will satisfy the linear system of equations. This becomes fast because there are fast methods to solve the subsystem corresponding to a sequence of non-boundary pixels. Yan and Chung [32] describe such a method based on solving a recurrence equation. They consider several cases, depending on how boundaries are handled.

Here I will show one method of deriving a solver for a sequence of pixels between boundaries. First note that in the interior of such a sequence (hereafter simply called a sequence) the relation

$$u[i-1] - \beta u[i] + u[i+1] = -(u_+[i] + u_-[i]) = b[i]$$

is satisfied. Here u_+ and u_- are taken to be equal to u if they fall outside the domain and $\beta = 4$. Equivalently u_+ and u_- can be considered zero outside the domain if we subtract one from β for each neighbour outside the domain[1].

[1] If both neighbours lie outside the domain the two zeroes of their characteristic equation become equal, and some of the recurrence relations later on would have different solutions. This case is trivial to solve though (by linear interpolation between boundary pixels) and is therefore not discussed further.

The extremal pixels of a sequence satisfy (without loss of generality we consider only the first pixel, at position 1)

$$u[2] - \beta u[1] = -(u[0] + u_+[1] + u_-[1]) = b[1] \tag{2.1}$$

if $u[0]$ belongs to a Dirichlet boundary, yielding the first case shown in Yan and Chung [32]. If $u[0]$ is outside the domain the condition $u[0] = u[1]$ is imposed and the extremal pixel satisfies

$$u[2] - (\beta - 1)u[1] = -(u_+[1] + u_-[1]) = b[1]. \tag{2.2}$$

This case is not shown by Yan and Chung, so we will derive it here. Also, it turns out that their correction can be improved by a more rigorous analysis.

First, lets introduce the basic premise of Yan and Chung's method. The recurrence relation shown above can be written as the linear system $Bu = b$, with

$$B = \begin{pmatrix} -\beta_1 & 1 & & & \\ 1 & -\beta & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -\beta & 1 \\ & & & 1 & -\beta_n \end{pmatrix}.$$

If we define $\lambda_\pm$ in such a way that $\lambda_+ + \lambda_- = \beta$ and $\lambda_+\lambda_- = 1$ this gives $\lambda_\pm = 2^{-1}(\beta \pm \sqrt{\beta^2 - 4})$[2]. Now we can find lower and upper triangular matrices

$$L = \begin{pmatrix} 1 & & & & \\ -\lambda_- & 1 & & & \\ & \ddots & \ddots & & \\ & & -\lambda_- & 1 & \\ & & & -\lambda_- & 1 \end{pmatrix} \text{ and } U = \begin{pmatrix} -\lambda_+ & 1 & & & \\ & -\lambda_+ & 1 & & \\ & & \ddots & \ddots & \\ & & & -\lambda_+ & 1 \\ & & & & -\lambda_+ \end{pmatrix},$$

such that

$$LU = \begin{pmatrix} -\lambda_+ & 1 & & & \\ 1 & -\beta & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -\beta & 1 \\ & & & 1 & -\beta \end{pmatrix} = B'.$$

This gives rise to a perturbed system $B'u' = b$, which can be solved easily using the LU decomposition of B' given above.

After the perturbed system is solved we have

$$\begin{aligned} u'[2] - \lambda_+ u'[1] &= b[1] \\ u'[i-1] - \beta u'[i] + u'[i+1] &= b[i] \\ u'[n-1] - \beta u'[n] &= b[n] \end{aligned}$$

where $\lambda_+ = 2^{-1}(\beta + \sqrt{\beta^2 - 4})$. In contrast, the recurrence relation that should be satisfied for u is (also see figure 2.1):

$$\begin{aligned} u[2] - \beta_1 u[1] &= b[1] \\ u[i-1] - \beta u[i] + u[i+1] &= b[i] \\ u[n-1] - \beta_n u[n] &= b[n]. \end{aligned}$$

[2]Compared to Yan and Chung [32] $\lambda_+ = -a$ and $\lambda_- = b$, giving a slightly more intuitive notation.

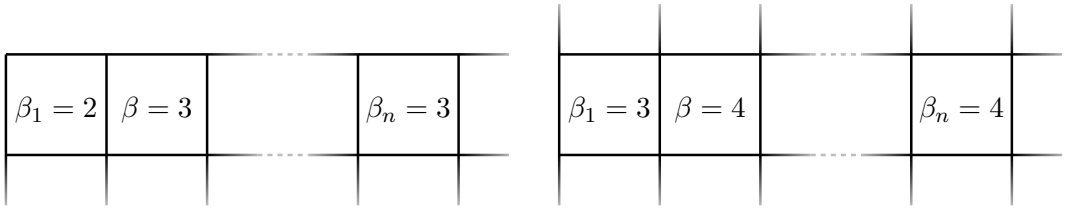


Figure 2.1: Two possible sequences, showing all possible β values. Note that the case where the image consists of a single row/column is ignored, as this is of little practical interest and trivial to solve (it results in linear interpolation between boundary pixels), while it would complicate the analysis somewhat.

So, to determine the correction vector $p = u - u'$ we only need to solve the (homogeneous) recurrence relation

$$\begin{aligned} p[2] - \beta_1 p[1] &= (\beta_1 - \lambda_+)u'[1] \\ p[i-1] - \beta p[i] + p[i+1] &= 0 \\ p[n-1] - \beta_n p[n] &= (\beta_n - \beta)u'[n]. \end{aligned}$$

This can be solved using the solutions $\lambda_\pm = 2^{-1}(\beta \pm \sqrt{\beta^2 - 4})$ to the characteristic equation $\lambda^2 - \beta\lambda + 1 = 0$ as the bases for power functions. For increased numerical stability we can avoid large exponents of λ_+ (whose magnitude is larger than one) by using $\lambda_+\lambda_- = 1$. When doing that we can express p as

$$p[i] = A\lambda_-^{n-i+1} + B\lambda_-^i.$$

Now A and B can be found by solving the following the equations:

$$\begin{aligned} A\lambda_-^n(\lambda_+ - \beta_1) + B\lambda_-(\lambda_- - \beta_1) &= (\beta_1 - \lambda_+)u'[1] \\ A\lambda_-(\lambda_- - \beta_n) + B\lambda_-^n(\lambda_+ - \beta_n) &= (\beta_n - \beta)u'[n]. \end{aligned}$$

Finally, the solution u can be found simply as $u = u' + p$.

2.3 Multigrid

In Orzan et al. [18] a simple multigrid solver is used which basically just starts with a very scaled down version of the problem, solves that, scales up the result and uses that as initialization for the next level. This is continued until the image is solved at the scale required. This probably is the simplest kind of multigrid solver, but works quite well, as it drastically reduces the time needed for colors to "reach the other side". For efficiency, it iterates a fixed number of times (3 in my implementation) at each iteration.

2.4 Variable stencil size

In Jeschke et al. [10] a solver is used which locally enlarges the size of the discrete Laplace kernel instead of scaling the solution grid like in a multigrid method (see figure 2.2). Initially the kernel is always as large as possible, in subsequent iterations the kernel is shrunk until it has reached its normal (not enlarged) size everywhere. The justification for this method is based on the smoothness of the solution, as this makes it possible to enlarge the kernel without the need for prefiltering the solution (although it could be interesting

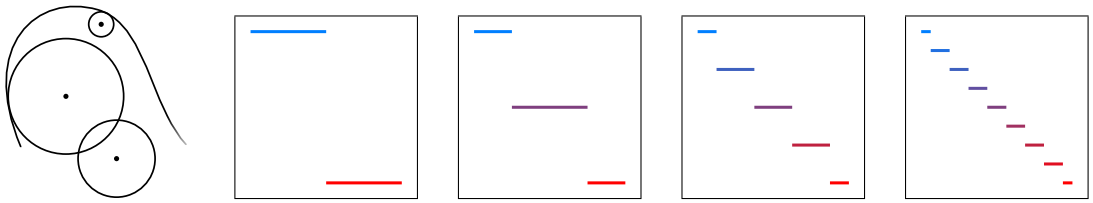


Figure 2.2: Using the variable stencil size method the size of the kernel is adjusted locally based on the distance to the boundary (l_2 or Euclidean distance is shown, but easier to compute alternatives can also be used). In combination with a proper shrinking strategy this schemes ideally doubles the precision of the solution at each iteration, as shown in 1D in the sequences of images on the right.

to explore if prefiltering would help). From a practical point of view this approach is much easier to implement than a normal multigrid method, and it appears to give better results as well, a win-win situation.

Based on Jeschke et al. [10] I have written a solver that employs a variable stencil size as well. However, in my implementation the radii r are shrunk by using $\max(r, 2^{n-i})$, with $n = \lceil \log_2(\text{max. distance}) \rceil$ being the number of iterations and $i \in [1, n]$. This gave slightly better results in my tests and makes it possible to compare the solver more directly with the multigrid solver.

Chapter 3

Errors and residuals

When looking at the approximate solution $\tilde{u}$ found by an algorithm there are two objects that can be used to say something about the objective quality of the result. These are the error $e = \tilde{u} - u$ (with u the reference solution) and the residual in solving whatever system of equations the algorithm was trying to solve. This chapter looks at the link between these two for both surface methods and boundary methods. I will show how boundary methods have a natural advantage over surface methods for solving the diffusion curve problem stated in equation (1.1). Specifically, for boundary methods it is shown that mostly the residual gives the error at the boundary, and that the maximum error occurs on the boundary. In contrast, for surface methods it is shown that the factor between the error and the residual can grow quadratically in the size of the domain (this is made more precise below).

3.1 Surface methods

All of the above grid based surface methods suffer from an issue that is both interesting and detrimental to their usefulness. As with any numerical method an exact solution will never be obtained. So instead of the Laplace equation the approximate solution $\tilde{u}$ will satisfy the non-homogeneous Poisson equation[1]

$$\Delta\tilde{u}(x) = f(x), \qquad (x \in \mathcal{D})$$

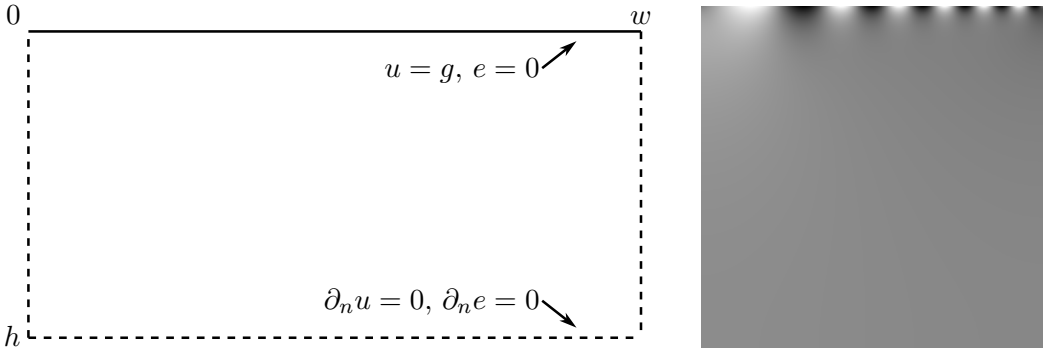
where $f(x)$ is the residual in solving the Laplace equation. Since the true solution u satifies the homogeneous Laplace equation

$$\Delta e(x) = \Delta(\tilde{u}(x) - u(x)) = f(x). \qquad (x \in \mathcal{D}) \tag{3.1}$$

In practice there are often areas where the residual has the same sign. Assume for the moment that such an area is circular, that the error on the boundary of the area is zero and that the residual is everywhere equal to ϵ. This will yield a *radially symmetric* error function that satisfies (using polar coordinates around the center of the area)

$$\begin{aligned}\epsilon &= \frac{\partial^2}{\partial\rho^2}e_\epsilon + \frac{1}{\rho}\frac{\partial}{\partial\rho}e_\epsilon + \frac{1}{\rho^2}\frac{\partial^2}{\partial\rho^2}e_\epsilon \\ &= \frac{\partial^2}{\partial\rho^2}e_\epsilon + \frac{1}{\rho}\frac{\partial}{\partial\rho}e_\epsilon.\end{aligned}$$

[1] For simplicity the continuous PDE is used instead of the discrete difference equation, it is expected that similar results will hold for the discrete case.



(a) Rectangular domain (b) Reference output

Figure 3.1: The rectangular domain with one Dirichlet boundary and three Neumann boundaries and the reference output. Along the top of the image the boundary condition $g(x_0) = \frac{1}{2}\left[1 + \sin(2\pi(1 + 5\frac{x_0}{w})\frac{x_0}{w}\right]$.

Solving this equation (allowing only smooth solutions) with $e(\pm R) = 0$ yields

$$e_\epsilon(\rho) = \frac{(\rho^2 - R^2)\epsilon}{4}.$$

Evidently e_ϵ reaches its maximum absolute value $\frac{R^2|\epsilon|}{4}$ at the center of the disc ($\rho = 0$). This shows that the maximum absolute error satisfies a quadratic dependency on the radius of the area, as well as growing quadratically from the boundary. This follows naturally from constraining the second derivative to some non-zero constant value and making the value zero at the boundaries.

Now suppose that the residual varies over the region but is still bounded from zero by a (without loss of generality) positive ϵ (that is $0 < \epsilon \leq f(x, y)$). Then $e - e_\epsilon$, the difference between the error functions, satisfies $\Delta e = f - \epsilon$. Since $f - \epsilon$ has the same sign as f and ϵ (or it is zero) the difference function also has the same sign (or is zero) as e_ϵ and $\max|e| \geq \max|e_\epsilon|$, only making things worse.

In general the components of f are amplified in e by a factor that is quadratic in their "period" $T = 1/\|\xi\|$, with ξ representing frequency coordinates. This can be seen by Fourier transforming equation (3.1):

$$\begin{aligned}
\mathcal{F}\{\Delta e(x)\} &= \mathcal{F}\{f(x)\} \\
\left[(2\pi \mathrm{i}\xi_0)^2 + (2\pi \mathrm{i}\xi_1)^2\right] E(\xi) &= F(\xi) \\
-4\pi^2\|\xi\|^2 E(\xi) &= F(\xi) \\
E(\xi) &= -\frac{1}{4\pi^2\|\xi\|^2} F(\xi). \\
E(\xi) &= -\frac{T^2}{4\pi^2} F(\xi).
\end{aligned}$$

Note that this does not uniquely identify the solution $e(x)$, as no boundary conditions are given. If boundary conditions are given a harmonic function can be added to make the solution satisfy them.

To see this effect in practice, consider the rectangular domain shown in figure 3.1. If the residual $f(x)$ is equal to the constant ϵ over the entire domain, then it is clear that the error is

$$e(x) = \frac{1}{2} x_1(x_1 - 2h)\epsilon. \tag{3.2}$$

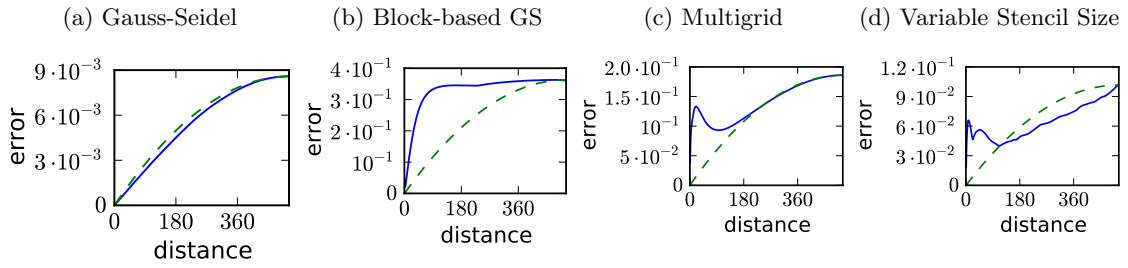


Figure 3.2: Plots of maximum absolute error against distance to the closest boundary. The solid blue line shows the measured value, the dashed green line shows equation (3.2), with ϵ chosen so that the error at the largest distance agrees. The test image described in figure 3.1 is used. To give an indication of the ratio between the maximum error and the residual, the RMS residuals were approximately $5.3 \cdot 10^{-8}$, $2.4 \cdot 10^{-4}$, $1.1 \cdot 10^{-4}$ and $2.3 \cdot 10^{-5}$, respectively.

This error's absolute value reaches its maximum $\frac{1}{2}h^2|\epsilon|$ at $x_1 = h$. As can be seen in figure 3.2 the measured error is almost always roughly equal to or larger than an estimate based on fitting equation (3.2), demonstrating that a small residual does not necessarily give a small error everywhere in the image. The main exception is the graph for the variable stencil size method. This one exception is likely due to a problem with the algorithm (or its specific implementation) though, as its error does not appear to satisfy the Neumann boundary condition (which it should do by definition), and because the graph does behave as expected when the algorithm is modified to iterate to a certain residual threshold before shrinking.

Note that all of the methods shown in figure 3.2 exhibit errors that are much larger than the smallest representable value for 8 bit images ($3.9 \cdot 10^{-3}$), except for the Gauss-Seidel method, which is way too slow. I have tried improving the results of the multigrid and variable stencil size methods by letting them iterate at each scale until the root-mean-square value of the residual was below some low value (10^{-7}). This resulted in a maximum absolute error that would just be acceptable for 8 bit images, but solving a 512×512 image took roughly 500 times as long, making this not very interesting in practice.

3.2 Boundary methods

In the face of Dirichlet boundaries it is obvious that boundary methods perform quite well, as the error will necessarily be a harmonic function (i.e., it satisfies the Laplace equation) in $\mathcal{D}$, and thus obtains its maximum value on the boundary. As it is the error on the boundary that is minimized in a boundary method this leads to a very direct relation between the maximum value of the residual in the solution of the boundary problem and the maximum error in the image: they are essentially equal.

In the face of Neumann boundaries, however, it is not so obvious how boundary methods perform. A useful tool to illustrate this is the 2π-periodic strip domain (figure 3.3), with a Dirichlet boundary on one side and a Neumann boundary on the other side. This geometry is particularly relevant to many real-world scenarios because the 2π-periodic strip domain can be mapped to any doubly-connected (annulus-like) domain while preserving a solution to the Laplace equation, using the theory of conformal mapping[2]. In particular this enables application of the analysis here to the case where all diffusion curves lie within the bounds of the image and a zero normal derivative is enforced across the image edges,

[2]Courant [4, §1.7] discusses how to construct conformal maps between doubly-connected domains.

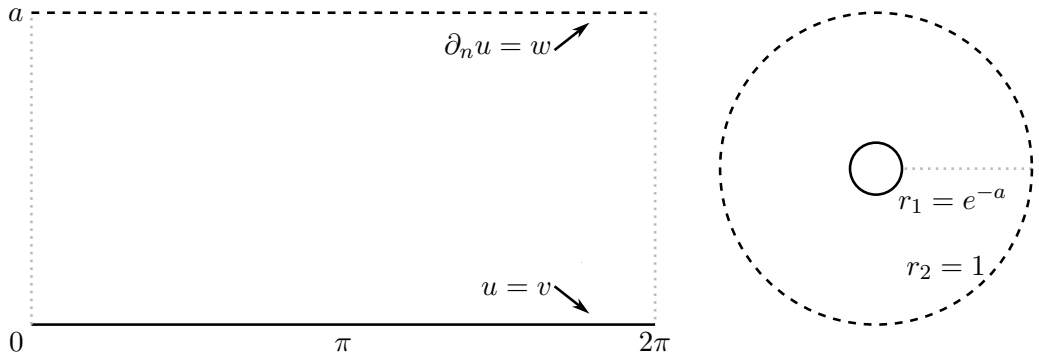


Figure 3.3: The 2π-periodic domain with one Dirichlet boundary and one Neumann boundary, and the corresponding annulus-shaped domain constructed with the conformal map $x' = e^{y-a}\cos(x), y' = e^{y-a}\sin(x)$.

by looking at the space between the image edges and any hull (not necessarily convex) around the diffusion curves.

The value of a resulting from a conformal map from a doubly-connected domain to the strip domain depends on how "thin" the original domain is, which can be made precise using the concept of extremal length [4, §A.3.6]. For example, the extremal length between the boundaries is $\ell_A = (2\pi)^{-1}\log(r2/r1)$ for an annulus[3] and $\ell_S = (2\pi)^{-1}a$ for the 2π-periodic strip domain. As extremal length is invariant under a conformal map these values can be equated to find the value of a such that the 2π-periodic strip domain is conformally homeomorphic to an annulus with radii r_1, r_2.

The Laplace equation can be conveniently solved on the 2π-periodic strip domain by using the separation of variables method for solving PDEs [21, ch.5 and §7.7], which works by *trying* to represent $u(x)$ as the product of $u_0(x_0)$ and $u_1(x_1)$. If $u = u_0 \cdot u_1$ is substituted in $\Delta u(x) = 0$ we obtain

$$\begin{aligned}
\Delta u(x) &= 0 \\
\Delta(u_0(x_0)u_1(x_1)) &= 0 \\
u_0''(x_0)u_1(x_1) + u_0(x_0)u_1''(x_1) &= 0 \\
u_0''(x_0)u_1(x_1) &= -u_0(x_0)u_1''(x_1) \\
\frac{u_0''(x_0)}{u_0(x_0)} &= -\frac{u_1''(x_1)}{u_1(x_1)}.
\end{aligned}$$

The last step gives the method its name. Now, since x_0 and x_1 are independent there has to be a constant λ such that

$$\frac{u_0''(x_0)}{u_0(x_0)} = -\frac{u_1''(x_1)}{u_1(x_1)} = \lambda.$$

This leads to

$$u_0'' - \lambda u_0 = 0 \tag{3.3}$$
$$u_1'' + \lambda u_1 = 0. \tag{3.4}$$

Now suppose that we represent $u_0(x_0)$ by the Fourier series

$$u_0(x_0) = \sum_{n=-\infty}^{\infty} U_0[n]e^{inx_0}.$$

[3] I use the square of the definition used in Courant [4], for convenience.

Equation (3.3) then becomes

$$\sum_{n=-\infty}^{\infty} \left[-n^2 U_0[n] - \lambda U_0[n]\right] e^{inx_0} = 0.$$

Clearly, it is in general impossible to select a single λ that makes this true. However, if λ is made to depend on n we have

$$\begin{aligned} n^2 U_0[n] + \lambda[n] U_0[n] &= 0 \\ \lambda[n] &= -n^2. \end{aligned}$$

Substituting this in equation (3.4) for a frequency dependent $U_1[n](x_1)$ we get

$$\begin{aligned} U_1[n]''(x_1) - n^2 U_1[n](x_1) &= 0 \\ U_1[n](x_1) &= A[n]e^{nx_1} + B[n]e^{-nx_1}. \end{aligned}$$

Finally leading to the general solution

$$\begin{aligned} u(x) &= \sum_{n=-\infty}^{\infty} U_0[n] U_1[n](x_1) e^{inx_0} \\ &= \sum_{n=-\infty}^{\infty} U_0[n] \left(A[n]e^{nx_1} + B[n]e^{-nx_1}\right) e^{inx_0}. \end{aligned} \tag{3.5}$$

To enforce a non-zero Dirichlet boundary condition at $x_1 = 0$ and a zero Neumann boundary condition at $x_1 = a$, $u(x_0, 0) = g(x_0)$ and $(\partial/\partial x_1)u(x_0, a) = 0$, we have

$$\begin{aligned} U_1[n](0) &= 1 = A[n] + B[n] \\ U_1[n]'(a) &= 0 = nA[n]e^{na} - nB[n]e^{-na} \\ A[n] &= 1 - B[n] \\ 0 &= n(1 - B[n])e^{na} - nB[n]e^{-na} \\ 0 &= e^{na} - B[n](e^{na} + e^{-na}) \\ B[n] &= e^{na}/(e^{na} + e^{-na}) \\ A[n] &= e^{-na}/(e^{na} + e^{-na}). \end{aligned}$$

Similarly a solution with a zero Dirichlet b.c. and a non-zero Neumann boundary condition, $u(x_0, 0) = 0$ and $(\partial/\partial x_1)u(x_0, a) = h(x_0)$, can be found by setting

$$\begin{aligned} U_1[n](0) &= 0 = A[n] + B[n] \\ U_1[n]'(a) &= 1 = nA[n]e^{na} - nB[n]e^{-na} \\ A[n] &= -B[n] \\ 1 &= n(-B[n])e^{na} - nB[n]e^{-na} \\ 1 &= -B[n]n(e^{na} + e^{-na}) \\ B[n] &= -1/\left[n(e^{na} + e^{-na})\right] \\ A[n] &= 1/\left[n(e^{na} + e^{-na})\right]. \end{aligned}$$

Putting the above all together a solution to the Laplace equation with $u(x_0, 0) = g(x_0)$ and $(\partial/\partial x_1)u(x_0, 1) = h(x_0)$ can be found. Denote the coefficients of the Fourier series for g and h by $G[n]$ and $H[n]$, respectively. Insert these in equation (3.5) using the

corresponding definitions for $A[n]$ and $B[n]$ derived above, and add the two solutions that arise, we then have

$$\begin{aligned} u(x) &= \sum_{n=-\infty}^{\infty} \left[G[n] \left(e^{n(x_1-a)} + e^{n(a-x_1)} \right) + H[n] \left(e^{nx_1} - e^{-nx_1} \right) / n \right] \frac{e^{inx_0}}{e^{na} + e^{-na}} \\ &= \sum_{n=-\infty}^{\infty} \left[G[n] \cosh(n(x_1 - a)) + H[n] \sinh(nx_1)/n \right] \operatorname{sech}(na) \cdot e^{inx_0}. \end{aligned} \tag{3.6}$$

This equation makes it possible to analyze the propagation of error through the domain by realizing that the error function e is of exactly the same form, ideally with $g = h = 0$, but with non-zero g and h in practice. Also, it is important to note that, as this is a harmonic function, the extrema are always at either $x_1 = 0$ or $x_1 = a$. As can be seen from the first part of equation (3.6) the part of the error involving G is reasonably benign, at $x_1 = a$ its frequency components are scaled by

$$\cosh(n(a - a)) \operatorname{sech}(na) = 2/(e^{na} + e^{-na}),$$

which is clearly positive and ≤ 1, and in fact exponentially decreasing in $|n|$. In contrast, for the second part, involving H, the components are scaled by

$$\sinh(na) \operatorname{sech}(na)/n = \tanh(na)/n.$$

For $|n| \geq 1$ the maximum absolute value over all a is $1/|n| \leq 1$, which is fine. Unfortunately things get a little hairier for $n = 0$, for which we have, by l'Hôpital's rule,

$$\lim_{n \to 0} \tanh(na)/n = \lim_{n \to 0} a \operatorname{sech}(na)^2 = a.$$

This shows that the error caused by the residual on the Neumann boundary can grow (at most) linearly in a. It is important to note that (non-zero) Neumann boundary conditions are not necessarily preserved by a conformal map, so one might question the relevancy of this result when looking at real world situations. Clearly the most important component of H that is important in this respect is $H[0]$, so below I will show that this component does remain unchanged after applying a conformal map.

It is useful to state a few properties of conformal maps now. Given the conformal map $F = [U, V]$ from a general doubly-connected domain $\mathcal{D}^*$ to the 2π-periodic strip domain we have (with $\delta\mathcal{D}_N^*$ the Neumann boundary):

P1. $U(x)$ maps $\delta\mathcal{D}_N^*$ bijectively to $[0, 2\pi)$.

P2. $x \in \delta\mathcal{D}_N^* \Rightarrow V(x) = a$.

P3. The Cauchy-Riemann equations $(\partial/\partial x_0)U = (\partial/\partial x_1)V$ and $(\partial/\partial x_1)U = -(\partial/\partial x_0)V$.

Notice that the Cauchy-Riemann equations imply that the Jacobian of F is a scalar times a rotation matrix, and in particular $\nabla U \perp \nabla V$ and $\|\nabla U\| = \|\nabla V\|$. Using these properties,

and denoting the solution in the general doubly-connected domain by u^*,

$$\begin{aligned}
u^*(s) &= u(F(s)) \\
\frac{\partial u^*}{\partial n}(s) &= J_{u^*}(s)\, n(s) \\
&= J_u(F(s))\, J_F(s)\, n(s) && \text{(chain rule)} \\
&= \frac{\partial u}{\partial x_0} J_U(s)\, n(s) + \frac{\partial u}{\partial x_1} J_V(s)\, n(s) \\
&= \frac{\partial u}{\partial x_1}(F(s))\|\nabla V(s)\| && (\mathrm{P3}, s \in \delta\mathcal{D} \Rightarrow \nabla V \parallel n) \\
&= \frac{\partial u}{\partial x_1}(U(s), a)\|\nabla U(s)\| && (\mathrm{P2}, \mathrm{P3}) \\
&= h(U(s))\|\nabla U(s)\|.
\end{aligned}$$

Which can be used to show that

$$\begin{aligned}
H^*[0] &= \int_{\delta\mathcal{D}_N} \frac{\partial u^*}{\partial n}(s)\, ds \\
&= \int_{\delta\mathcal{D}_N} h(U(s))\|\nabla U(s)\|\, ds \\
&= \int_0^{2\pi} h(x_0)\, dx_0 && (P1, \|\nabla U(s)\|\, ds = dx_0) \\
&= H[0].
\end{aligned}$$

This establishes that the maximum factor between the residual on the Neumann boundary and the resulting error in the image is linear in ℓ (note that since we used a conformal map $\ell = \ell_S = (2\pi)^{-1}a$).

In summary, whereas with surface methods the error can have a quadratic dependency on the diameter of a region[4], with boundary methods the maximum error is either independent of the size of the domain or depends linearly on the extremal length ℓ between the boundaries (and only for the constant component of the residual on the Neumann boundary).

[4] A quadratic dependency on the diameter for the error was shown for surface methods in a circular domain with Dirichlet boundaries and in the rectangular domain with one Dirichlet boundary and three Neumann boundaries, but something similar can be shown for the 2π-periodic strip domain with one Neumann boundary.

Chapter 4

Analytic Element Method

In contrast to all of the methods above, the AEM does not consider the interior of the domain, it only considers the boundary of the domain. This makes it easy to handle domains of arbitrary size (even infinite) and can provide a distinct performance advantage whenever it takes (much) less elements to discretize the boundary than it takes to discretize the domain (which is often the case). In addition, as shown earlier, the AEM has the potential to provide more control over errors in the image. Instead of having an error that can grow quadratically with the size of the image, the largest error in the image is mostly bounded by the largest residual in trying to match the boundary conditions. Whenever the largest error is larger than the largest residual, this is due to the zero frequency component of the error on the Neumann boundary and it will still be only linear in the extremal length between the Dirichlet and Neumann boundaries.

4.1 One dimension

As an introduction to the concept behind the analytic element method we will first examine the (trivial) one-dimensional case. Suppose we have a solution domain that consists of a line segment, with a few points on this segment where we have a Dirichlet boundary condition (on either side of the point) and zero Neumann boundary conditions on the endpoints of the segment. It is clear that a solution to the one-dimensional equivalent of the Laplace equation ($u''(x) = 0$) will be linear, so in our case the complete solution will be piecewise-linear, with all discontinuities at boundary points.

To apply the AEM we first need to find suitable basis functions. A simple linear function does not suffice, as any linear combination of linear functions will yield another linear function, making it impossible to satisfy more than two constraints. Instead a function $\varphi(x)$ is used which satisfies $\varphi''(x) = \delta(x)$[1]: $\frac{1}{2}|x|$. Obviously, to still satisfy $u''(x) = 0$ between boundary points, this so-called *fundamental solution* can only be centered on boundary points.

The fundamental solution $\frac{1}{2}|x|$ can be used to build up any *continuous* piecewise-linear solution. This can be done using a sum of translated and scaled copies, with each copy centered on a boundary point. To also allow for discontinuities at boundary points we need a basis function whose *first* derivative is $\delta(x)$ (see section 4.4 for how to handle discontinuities in the 2D case). This is clearly satisfied by the first derivative of the fundamental solution, as $(d/dx)(\varphi'(x)) = \varphi''(x) = \delta(x)$. As illustrated in figure 4.1 this

[1]The Dirac Delta $\delta(x)$ is defined as satisfying $\int_{\mathbb{R}} f(\tau)\delta(t-\tau)\,d\tau = f(t)$ for any function $f(x)$. One construction is to take the limit of the Normal distribution $e^{-x^2/(2\sigma^2)}/\sqrt{2\pi\sigma^2}$ as σ goes to zero.

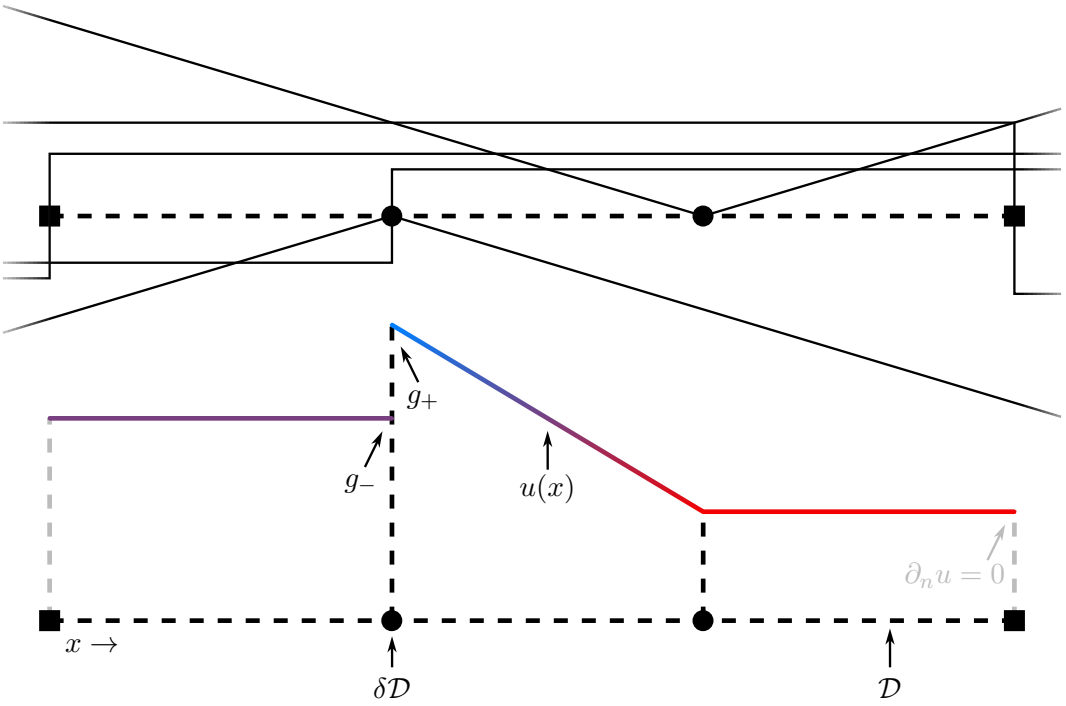


Figure 4.1: A one-dimensional problem, showing the weighted basis functions (top) and the result (bottom). Note that as no constraint is put on the values outside the domain there is some choice in how to choose the weights for the two outer boundary points.

can be used to write any solution as

$$u(x) = \sum_{y \in \delta\mathcal{D}} \omega_N(y) \frac{1}{2} |x - y| + \omega_D(y) \frac{1}{2} \operatorname{sign}(x - y). \tag{4.1}$$

The analytic element method tries to solve equation (4.1) by trying to find $\omega_N(y)$ and $\omega_D(y)$ such that the boundary conditions are met.

4.2 Two dimensions

As we have seen, the fundamental solution to Laplace's equation is the basis of the AEM and related methods. In two dimensions it is defined as ([21, ch. 8], assuming $x \in \mathbb{R}^2$)

$$\varphi(x) = \frac{1}{2\pi} \ln \|x\|.$$

It can be checked that $\Delta\varphi(x) = \delta(x)$.

For analytical purposes it can be useful to use $\ln(z) = \ln|z| + \arg(z)\mathrm{i}$ with $z = x_0 + x_1\mathrm{i}$ and rewrite the fundamental solution as

$$\varphi(z) = \operatorname{Re}\left[\frac{1}{2\pi} \ln(z)\right].$$

This leads to much simpler algebraic manipulations. In particular it makes it possible to easily write down closed form expressions for integrals involving the fundamental solution,

for example (taking the full, complex, solution and ignoring the constant factor $(2\pi)^{-1}$):

$$\begin{aligned}\int_{-1}^{1} \ln(z-t)\,dt &= \int_{z-1}^{z+1} \ln(\zeta)\,d\zeta \quad , \zeta = z-t \\ &= [\zeta\ln(\zeta)-\zeta]_{z-1}^{z+1} \\ &= (z+1)\ln(z+1)-(z+1)-(z-1)\ln(z-1)+(z-1) \\ &= (1+z)\ln(z+1)+(1-z)\ln(z-1)-2.\end{aligned}$$

Just like in the one dimensional case the AEM for diffusion curves tries to find a solution u that satisfies the diffusion curve problem (1.1) by placing the fundamental solution everywhere along the boundary. Here the boundary is one dimensional (a curve) though, so the sum becomes an integral. In this scheme the solution is given by

$$u(x) = \int_{\delta\mathcal{D}} w(s)\varphi(x-s)\,ds.$$

Assume the boundary is parameterized by a function $l : [0, N] \to \mathbb{R}^2$, consisting of N continuous segments traced out by $l_j(t) = l(j+t)$ (with $t \in [0,1]$), then

$$\begin{aligned}u(x) &= \int_0^N w(l(\tau))\varphi(x-l(\tau))\|\nabla l(\tau)\|\,d\tau \\ &= \sum_{j=1}^{N}\int_0^1 w(l_j(\tau))\varphi(x-l_j(\tau))\|\nabla l_j(\tau)\|\,d\tau.\end{aligned}$$

When evaluated on the boundary this gives a continuous system of linear equations (using a suggestive matrix notation)

$$u(l(t)) = (Mw)_t. \qquad\qquad (M_{t,\tau} = \varphi(l(t)-l(\tau))\|\nabla l(\tau)\|)$$

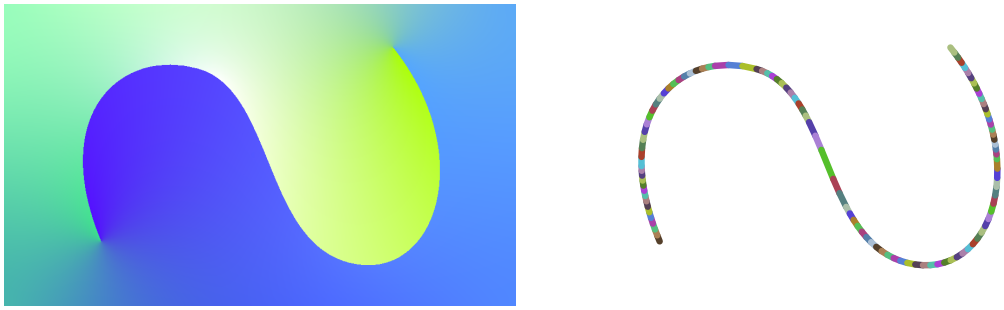
The "columns" of M can be discretized by using polynomials p_n upto order M as basis functions for w[2]:

$$\begin{aligned}u(l(t)) &= \sum_{j=1}^{N}\int_0^1 w_j(l(\tau))\varphi(l(t)-l_j(\tau))\|\nabla l_j(\tau)\|\,d\tau \\ &= \sum_{j=1}^{N}\int_0^1 \left[\sum_{n=0}^{M}\omega_{j,n}\,p_n(\tau)\right]\varphi(l(t)-l_j(\tau))\|\nabla l_j(\tau)\|\,d\tau \\ &= \sum_{j=1}^{N}\sum_{n=0}^{M}\omega_{j,n}\int_0^1 \varphi(l(t)-l_j(\tau))\|\nabla l_j(\tau)\|p_n(\tau)\,d\tau.\end{aligned}$$

The "rows" of M can be discretized by one of two methods: the collocation method, or the Galerkin method[3]. This consists of taking either a finite number of samples of Mw along each line segment (collocation) or the dot products between Mw and some finite

[2]In principle the "columns" of M can also be discretized using a finite number of Dirac deltas (pulses), but this is not done in the analytic element method, one advantage being that no spurious singularities are introduced.

[3]The Galerkin method can be seen as a generalization of the collocation method if you consider the Dirac delta as possible test "function".



(a) AEM solution (b) Boundary segments

Figure 4.2: Just like in the grid-based methods a Neumann boundary condition was enforced at the four sides of the image.

family of appropriate test functions (Galerkin). The Galerkin method can give rise to a nice, symmetric, system of linear equations, given by the matrix

$$M^G_{(i,m),(j,n)} = \int_0^1 \int_0^1 p_m(t) \, \|\nabla l_i(t)\| \, \varphi(l_i(t) - l_j(\tau)) \, \|\nabla l_j(\tau)\| \, p_n(\tau) \, d\tau \, dt. \tag{4.2}$$

Note that for clarity the row and column indices are given as pairs, so (i, m) specifies the row, and (j, n) the column, of M^G.

In its simplest form, the collocation method gives rise to the (non-symmetric) matrix

$$M^C_{(i,m),(j,n)} = \|\nabla l_i(t_m)\| \int_0^1 \varphi(l_i(t_m) - l_j(\tau)) \, \|\nabla l_j(\tau)\| \, p_n(\tau) \, d\tau. \tag{4.3}$$

The factor $\|\nabla l_i(t_m)\|$ gives more weight to samples representing a larger portion of the boundary[4].

In figure 4.2 I used the collocation method as given above. Each segment is straight and the polynomials $p_k(\tau)$ are the monomials transformed so that $p_k(0) = (-1)^k$ and $p_k(1) = 1$. The number of samples taken is greater than the number of unknowns (known as the overspecification principle [9]) and the resulting system is solved in a least-squares sense. For simplicity QR decomposition is used, although in practice methods like the nested super-block method [5] would give better performance by using a hierarchical datastructure to compute long-range interactions.

4.3 Integrating the fundamental solution

Both equations (4.2) and (4.3) contain (definite) integrals of the fundamental solution. These integrals are not that easy to evaluate numerically, as the common methods for doing so depend on being able to approximate the integrand using a polynomial, which is not possible if the integrand can be singular in the integration interval. Here we will work around this issue by performing the integration analytically, leading to a simple, precise and efficient implementation. For simplicity only the integrals necessary for the collocation method are investigated, those for the Galerkin method can likely be dealt with in a similar fashion.

Please note that we will only consider linear segments, so $\|\nabla l_j(\tau)\|$ is constant and we can ignore it in most of the computations. Also, without loss of generality, we will only

[4]The factor $\|\nabla l_i(t_m)\|$ also follows from the definition for M^G if $p_m(t) = \delta(t - t_m)$.

consider the line segment that goes from $(-1,0)$ to $(1,0)$ as t goes from -1 to 1 in this section. This is in contrast to the previous section, where t went from 0 to 1 for a single line segment. This difference in notation does not truely matter for the argumentation, but does make some equations more natural. It should also be noted that from now on only the complex fundamental solution $\ln(z)$ will be used, as this greatly simplifies the results.

Below we will first look at deriving an explicit representation of the required integrals, which will prove to be mostly useful for the near-field (although in practice it does quite well in the far-field as well, at least for low order polynomials). Next a series representation will be derived that can be useful for far-field computations. Finally, I will discuss two approaches for handling arbitrary linear segments, and not just the segment that goes from $(-1,0)$ to $(1,0)$. Appendix A has details on how to compute the derivatives of the integrals derived here.

4.3.1 Near-field

Previously, we found

$$\begin{aligned}\Phi_0(z) &= \int_{-1}^{1} \ln(z-t)\,dt \\ &= (1+z)\ln(z+1) + (1-z)\ln(z-1) - 2.\end{aligned}$$

In general we have

$$\Phi_n(z) = \int_{-1}^{1} p_n(t)\ln(z-t)\,dt.$$

As it is not difficult to integrate a polynomial we could try integration by parts:

$$\int_{-1}^{1} p_n(t)\ln(z-t)\,dt = \left[\left(\int p_n(t)\,dt\right)\ln(z-t)\right]_{-1}^{1} - \int_{-1}^{1}\left(\int p_n(t)\,dt\right)\frac{1}{t-z}\,dt.$$

This does not seem to get us anywhere though, as $p_n(t)$ is not divisible by $t-z$ and there is no other obvious way to integrate the last term. But what if $p_n(t)$ had been divisible by $t-z$? Then we would no longer have anything other than a polynomial as integrand, making integration trivial. Although $p_n(t)$ cannot be divisible by $t-z$, the following manipulations do lead to a formula that only requires integrating polynomials:

$$\begin{aligned}\int p_n(t)\ln(z-t)\,dt &= \left(\int p_n(t)\,dt\right)\ln(z-t) - \int\left(\int p_n(t)\,dt\right)\frac{1}{t-z}\,dt \\ &= \left(\int p_n(t)\,dt\right)\ln(z-t) - \left(\int p_n(z)\,dz\right)\ln(z-t) \\ &\quad - \int\left(\int p_n(t)\,dt\right)\frac{1}{t-z}\,dt + \left(\int p_n(z)\,dz\right)\ln(z-t) \\ &= \left(\int p_n(t)\,dt - \int p_n(z)\,dz\right)\ln(z-t) \\ &\quad - \int\left(\int p_n(t)\,dt\right)\frac{1}{t-z}\,dt + \left(\int p_n(z)\,dz\right)\left(\int \frac{1}{t-z}\,dt\right) \\ &= \left(\int p_n(t)\,dt - \int p_n(z)\,dz\right)\ln(z-t) \\ &\quad - \int\left(\int p_n(t)\,dt - \int p_n(z)\,dz\right)\frac{1}{t-z}\,dt.\end{aligned}$$

That $\int p_n(t)\,dt - \int p_n(z)\,dz$ is indeed divisible by $(t-z)$ can be seen as follows:

$$\begin{aligned}\int p_n(t)\,dt - \int p_n(z)\,dz &= \int \sum_i c_i^{(n)} t^i\,dt - \int \sum_i c_i^{(n)} z^i\,dz \\ &= \sum_i \frac{1}{i+1} c_i^{(n)} \left[t^{i+1} - z^{i+1}\right] \\ &= \sum_i \frac{1}{i+1} c_i^{(n)} (t-z) Q_i(t,z),\end{aligned}$$

with (note that the last step is only valid if $i \geq 0$)

$$\begin{aligned}Q_i(t,z) &= \frac{t^{i+1} - z^{i+1}}{t-z} \\ &= zQ_{i-1}(t,z) + t^i \\ &= \left[t^i + t^{i-1}z + \cdots + z^i\right].\end{aligned} \tag{4.4}$$

For the monomials we can now find

$$\begin{aligned}\Phi_n^M(z) &= \int_{-1}^{1} t^n \ln(z-t)\,dt \\ &= \left[\left(\int t^n\,dt - \int z^n\,dz\right)\ln(z-t)\right]_{t=-1}^{t=1} \\ &\quad - \int_{-1}^{1} \left(\int t^n\,dt - \int z^n\,dz\right)\frac{1}{t-z}\,dt \\ &= \left[\left(\frac{1}{n+1}t^{n+1} - \frac{1}{n+1}z^{n+1}\right)\ln(z-t)\right]_{t=-1}^{t=1} \\ &\quad - \int_{-1}^{1} \left(\frac{1}{n+1}t^{n+1} - \frac{1}{n+1}z^{n+1}\right)\frac{1}{t-z}\,dt \\ &= \frac{1}{n+1}\left[(1-z)Q_n(1,z)\ln(z-1) - (-1-z)Q_n(-1,z)\ln(z+1) - \int_{-1}^{1} Q_n(t,z)\,dt\right] \\ &= \frac{1}{n+1}\left[(1-z)Q_n(1,z)\ln(z-1) + (1+z)Q_n(-1,z)\ln(z+1) - \int_{-1}^{1} Q_n(t,z)\,dt\right].\end{aligned}$$

This can be evaluated using the recurrence relation

$$\begin{aligned}\Phi_0^M(z) &= (1-z)\ln(z-1) - (-1-z)\ln(z+1) - \int_{-1}^{1} 1\,dt \\ &= (1-z)\ln(z-1) + (1+z)\ln(z+1) - 2 \\ \Phi_n^M(z) &= \frac{1}{n+1}\left[(1-z)Q_n(1,z)\ln(z-1) + (1+z)Q_n(-1,z)\ln(z+1) - \int_{-1}^{1} Q_n(t,z)\,dt\right] \\ &= \frac{1}{n+1}\left[(1-z)(zQ_{n-1}(1,z) + 1^n)\ln(z-1)\right. \\ &\qquad \left. +(1+z)(zQ_{n-1}(-1,z) + (-1)^n)\ln(z+1) - \int_{-1}^{1} zQ_{n-1}(t,z) + t^n\,dt\right] \\ &= \frac{1}{n+1}\left[nz\Phi_{n-1}^M(z) + (1-z)\ln(z-1) + (-1)^n(1+z)\ln(z+1) - \frac{1+(-1)^n}{n+1}\right].\end{aligned}$$

Conveniently, this allows efficient evaluation of Φ_n for a range of n, making these computations scale quite well when using high order elements.

Alternatively, all terms except the integral of Q_n can be computed explicitly and the integral of Q_n can be computed using the recurrence relation

$$\begin{aligned}\int_{-1}^{1} Q_0(t,z)\,dt &= \int_{-1}^{1} 1\,dt = 2\\ \int_{-1}^{1} Q_n(t,z)\,dt &= \int_{-1}^{1} zQ_{n-1}(t,z) + t^n\,dt\\ &= z\int_{-1}^{1} Q_{n-1}(t,z)\,dt + \frac{1+(-1)^n}{n+1}.\end{aligned}$$

4.3.2 Far-field

The above can prove problematic when $|z| \gg 1$. Instead we can try again to apply the product rule repeatedly to the original integral. Just looking at the antiderivative this gives

$$\begin{aligned}\int p_n(t)\ln(z-t)\,dt &= \left(\int p_n(t)\,dt\right)\ln(z-t)\\ &\quad - \int\left(\int p_n(t)\,dt\right)\frac{1}{t-z}\,dt\\ &= \left(\int p_n(t)\,dt\right)\ln(z-t)\\ &\quad - \left(\iint p_n(t)\,dt^2\right)\frac{1}{t-z}\\ &\quad - \int\left(\iint p_n(t)\,dt^2\right)\frac{1}{(t-z)^2}\,dt\\ &= \left(\int p_n(t)\,dt\right)\ln(z-t)\\ &\quad - \left(\iint p_n(t)\,dt^2\right)\frac{1}{t-z}\\ &\quad - \left(\int\cdots\int p_n(t)\,dt^3\right)\frac{1}{(t-z)^2}\\ &\quad - \int\left(\int\cdots\int p_n(t)\,dt^3\right)\frac{2}{(t-z)^3}\,dt\\ &= \left(\int p_n(t)\,dt\right)\ln(z-t)\\ &\quad - \sum_{i=1}^{\infty}\left(\int\cdots\int p_n(t)\,dt^{i+1}\right)\frac{(i-1)!}{(t-z)^i}.\end{aligned}$$

Using the Cauchy formula for repeated integration (where $\int_a^t f(t)\,dt$ should be read as $\left[\int f(t)\,dt\right]_a^t$, effectively giving an antiderivative which is zero at a)

$$\int_a^t\cdots\int_a^t p_n(t)\,dt^{i+1} = \frac{1}{i!}\int_a^t (t-s)^i p_n(s)\,ds$$

this becomes

$$\int_a^t p_n(t)\ln(z-t)\,dt = \left(\int_a^t p_n(s)\,ds\right)\ln(z-t) - \sum_{i=1}^{\infty}\frac{1}{i}\left(\int_a^t (t-s)^i p_n(s)\,ds\right)\frac{1}{(t-z)^i}.$$

Now the definite integral

$$\begin{aligned}\Phi_n(z) &= \left(\int_a^1 p_n(s)\,ds\right)\ln(z-1) - \left(\int_a^{-1} p_n(s)\,ds\right)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\left(\int_a^1 (1-s)^i p_n(s)\,ds\right)\frac{1}{(1-z)^i} - \left(\int_a^{-1}(-1-s)^i p_n(s)\,ds\right)\frac{1}{(-1-z)^i}\right] \\ &= \left(\int_a^1 p_n(s)\,ds\right)\ln(z-1) + \left(\int_{-1}^a p_n(s)\,ds\right)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\left(\int_a^1 (1-s)^i p_n(s)\,ds\right)\frac{1}{(1-z)^i} + \left(\int_{-1}^a (1+s)^i p_n(s)\,ds\right)\frac{1}{(1+z)^i}\right].\end{aligned}$$

Different choices of a lead to different approximations. Good choices are $a = 0$ or *both* $a = -1$ and $a = +1$ and averaging the result. The latter has the advantage that for the Legendre polynomials the integral $\int_{-1}^1 (1 \pm s)^i p_n(s)\,ds$ is zero for $i < n$. This integral becomes zero for small i, because the n-th Legendre polynomial is orthogonal to all polynomials of degree less than n. In particular this avoids needing to compute a log term for $n > 0$.

When choosing $a = 0$ and working with polynomials for which $p_n(-t) = (-1)^n p_n(t)$ (such as the monomials, Legendre polynomials or Chebyshev polynomials), then it is convenient to make the substitutions $s = 1 - s$ and $s = s - 1$ in the left and right integrals, respectively, to get

$$\begin{aligned}\Phi_n(z) &= \left(\int_0^{1-a} p_n(1-s)\,ds\right)\ln(z-1) + \left(\int_0^{1+a} p_n(s-1)\,ds\right)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\left(\int_0^{1-a} s^i p_n(1-s)\,ds\right)\frac{1}{(1-z)^i} + \left(\int_0^{1+a} s^i p_n(s-1)\,ds\right)\frac{1}{(1+z)^i}\right] \\ &= \left(\int_0^{1-a} p_n(1-s)\,ds\right)\ln(z-1) + (-1)^n\left(\int_0^{1+a} p_n(1-s)\,ds\right)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\left(\int_0^{1-a} s^i p_n(1-s)\,ds\right)\frac{1}{(1-z)^i} + \left(\int_0^{1+a} s^i p_n(1-s)\,ds\right)\frac{(-1)^n}{(1+z)^i}\right].\end{aligned}$$

Specifically, for the monomials, we have

$$\begin{aligned}\Phi_n^M(z) &= \left(\int_0^{1-a}(1-s)^n\,ds\right)\ln(z-1) + (-1)^n\left(\int_0^{1+a}(1-s)^n\,ds\right)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\left(\int_0^{1-a} s^i(1-s)^n\,ds\right)\frac{1}{(1-z)^i} + \left(\int_0^{1+a} s^i(1-s)^n\,ds\right)\frac{(-1)^n}{(1+z)^i}\right] \\ &= \mathrm{B}(1-a;1,n+1)\ln(z-1) + (-1)^n\mathrm{B}(1+a;1,n+1)\ln(z+1) \\ &\quad - \sum_{i=1}^{\infty}\frac{1}{i}\left[\mathrm{B}(1-a;i+1,n+1)\frac{1}{(1-z)^i} + \mathrm{B}(1+a;i+1,n+1)\frac{(-1)^n}{(1+z)^i}\right].\end{aligned}$$

If we then indeed choose $a = 0$ this reduces to

$$\begin{aligned}\Phi_n^M(z) &= \mathrm{B}(1, n+1)\left[\ln(z-1) + (-1)^n \ln(z+1)\right] \\ &\quad - \sum_{i=1}^{\infty} \frac{1}{i}\mathrm{B}(i+1, n+1)\left[\frac{1}{(1-z)^i} + (-1)^n \frac{1}{(1+z)^i}\right] \\ &= \mathrm{B}(1, n+1)\left[\ln(z-1) + (-1)^n \ln(z+1)\right] \\ &\quad + \sum_{i=1}^{\infty} \frac{1}{i}\mathrm{B}(i+1, n+1)\left[(-1)^{n+1}\frac{1}{(1+z)^i} - \frac{1}{(1-z)^i}\right].\end{aligned} \tag{4.5}$$

Here $\mathrm{B}(a, b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ is the beta function defined as $\int_0^1 t^{a-1}(1-t)^{b-1}\,dt$, and $\mathrm{B}(x; a, b)$ is the incomplete beta function defined as $\int_0^x t^{a-1}(1-t)^{b-1}\,dt$. As $\Gamma(i+1) = i!$ for any integer i, $\mathrm{B}(i+1, n+1) = (i/(n+i+1))\mathrm{B}(i, n+1)$, this can be used to quickly compute the coefficients.

The series in equation (4.5) is suitable for use with large z, and similar to series used in Strack [25, e.g. ch. 3 and ch. 6], Le Grand [12] and Steward et al. [24]. To evaluate it efficiently, it is useful to note that

$$\begin{aligned}Z_{i+1} &= (-1)^{n+1}\frac{1}{(1+z)^{i+1}} - \frac{1}{(1-z)^{i+1}} \\ &= (-1)^{n+1}\frac{(1-z)}{(1-z)(1+z)(1+z)^i} - \frac{(1+z)}{(1-z)(1+z)(1-z)^i} \\ &= \frac{1}{(1-z)(1+z)}Z_i - (-1)^{n+1}\frac{z}{(1-z)(1+z)(1+z)^i} - \frac{z}{(1-z)(1+z)(1-z)^i} \\ &= \frac{1}{(1-z)(1+z)}2Z_i - (-1)^{n+1}\frac{(z+1)}{(1-z)(1+z)(1+z)^i} - \frac{(z-1)}{(1-z)(1+z)(1-z)^i} \\ &= \frac{1}{(1-z)(1+z)}2Z_i - (-1)^{n+1}\frac{1}{(1-z)(1+z)^i} + \frac{1}{(1+z)(1-z)^i} \\ &= \frac{1}{1-z^2}\left[2Z_i - Z_{i-1}\right].\end{aligned}$$

4.3.3 General linear segments

The previous subsections considered a linear segment starting at $(-1, 0)$ and ending at $(1, 0)$. In practice this is of course rarely the case, so we need a mapping from a general segment to this standard segment. The most straightforward way to do this is to simply scale, translate and rotate the coordinate system in such a way that the segment in question becomes the standard segment. If the segment goes from p to q (using complex "coordinates"), then the transformed coordinate

$$z' = \frac{(z-p)-(q-z)}{q-p} = \frac{2z-p-q}{q-p}.$$

This can then be used to compute $\Phi_n(z')$. As this mapping is conformal, $\Phi_n(z')$ still satisfies the Laplace equation away from the boundary.

Performing the above coordinate transformation is a valid method to compute a basis function for the analytic element method, but it does not (in general) give the same answer as integrating the fundamental solution along an arbitrary linear segment. A small modification is needed if exact equivalence is necessary, as it is for using the technique

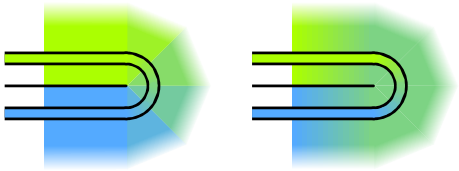


Figure 4.3: Left: An angle dependent color at endpoints in a FEM reproduces the discontinuity along the curve, but also introduces discontinuities between adjacent triangles at the endpoint. Right: Assigning all nodes at the endpoint the same (average) color (as in Orzan [17]) causes color bleeding.

described in section 4.4:

$$\begin{aligned}
\Phi_n(z) &= \frac{1}{4\pi}|q-p| \int_{-1}^{1} p_n(t) \ln\left(z - \left[\frac{1}{2}(q-p)t + \frac{1}{2}(p+q)\right]\right) dt \\
&= \frac{1}{4\pi}|q-p| \int_{-1}^{1} p_n(t) \ln\left(\left[\frac{1}{2}(q-p)z' + \frac{1}{2}(p+q)\right] - \left[\frac{1}{2}(q-p)t + \frac{1}{2}(p+q)\right]\right) dt \\
&= \frac{1}{4\pi}|q-p| \int_{-1}^{1} p_n(t) \ln\left(\frac{1}{2}(q-p)(z'-t)\right) dt \\
&= \frac{1}{4\pi}|q-p| \left[\int_{-1}^{1} p_n(t) \ln\left(z'-t\right)\, dt + \ln\left(\frac{1}{2}(q-p)\right) \int_{-1}^{1} p_n(t)\, dt\right].
\end{aligned}$$

The factor in front of the integral accounts for the length of the segment and the factor of $(2\pi)^{-1}$ that should be in front of the fundamental solution, also compare to $\|\nabla l_j(\tau)\|$ in equation (4.3).

4.4 Adding discontinuities back in

Discontinuities in the resulting image are, frankly, the raison d'être for diffusion curves. They do come with some problems though. First of all, it is not completely clear how to properly handle endpoints of curves. What equations should hold there? In figure 1.4 I gave one possibility for the continuous case which could be used with a FEM or any other grid-based method, except that it would introduce discontinuities where they are *not* supposed to be, as shown in figure 4.3. Orzan [17] suggests averaging the colors at the endpoints, but as figure 4.3 shows, this can lead to color bleeding.

Secondly, the fact that both sides of the curve can have a different color complicates discretization of the boundary *between* the endpoints as well. So much so in fact that Orzan et al. [18] deemed it necessary to take special precautions to prevent both sides of the boundary from interfering with each other. Also, the FEM-like method used in Orzan [17, §3.3] needs to explicitly double the curve edges. The discretization shown in Jeschke et al. [10] alleviates these issues, but still requires some extra bookkeeping and a "heavy", two-pixel wide, boundary. It would be great if we could somehow make the solver work without discontinuities, while still having jumps in color in the final image.

Luckily we can eliminate any discontinuities from the image *beforehand*, and this can be done directly, without explicitly solving a system of equations, only requiring an adjustment of the boundary conditions. This is based on separating the solution u into a continuous part v that uses the average color of both sides as boundary condition (with a slight correction, read on) and a discontinuous part w that purely represents the "jumps" across the boundaries (see figure 4.4). Any normal derivative (Neumann) boundary conditions also need some adjustment. These solutions satisfy

$$\begin{aligned}
\Delta v(x) &= 0 && (x \in \mathcal{D}) \\
v(x) &= \frac{1}{2}(g_+(x) + g_-(x)) - w(x) && (x \in \delta\mathcal{D})
\end{aligned}$$

(a) Final solution $u = v + w$ (b) Continuous solution v (c) Discontinuous solution w

Figure 4.4: The final solution u (a) is built up of a continuous solution v (b) and a discontinuous solution w (c). In the latter, zero is represented by 50% full-scale gray (no other color transformations have been applied). Note that the discontinuous solution is trivial to compute and that the continuous solution can be found using any of the mentioned algorithms to solve the Laplace equation.

and

$$\begin{aligned}
\Delta w(x) &= 0 && (x \in \mathcal{D}) \\
\lim_{d \downarrow 0} w(x + dn(x)) &= w(x) + \frac{1}{2}(g_+(x) - g_-(x)) && (x \in \delta\mathcal{D}) \\
\lim_{d \downarrow 0} w(x - dn(x)) &= w(x) + \frac{1}{2}(g_-(x) - g_+(x)), && (x \in \delta\mathcal{D})
\end{aligned}$$

leading to

$$\begin{aligned}
u(x) &= v(x) + w(x) \\
\Delta u(x) &= \Delta(v(x) + w(x)) = 0 && (x \in \mathcal{D}) \\
\lim_{d \downarrow 0} u(x + dn(x)) &= \frac{1}{2}(g_+(x) + g_-(x)) + \frac{1}{2}(g_+(x) - g_-(x)) = g_+(x) && (x \in \delta\mathcal{D}) \\
\lim_{d \downarrow 0} u(x - dn(x)) &= \frac{1}{2}(g_+(x) + g_-(x)) + \frac{1}{2}(g_-(x) - g_+(x)) = g_-(x). && (x \in \delta\mathcal{D})
\end{aligned}$$

To form w I will now show that the above procedure corresponds to splitting the solution into the two components that result from Green's representation formula, modified slightly to take into account that two-sided boundaries are used. Using $\frac{\partial}{\partial n}$ to denote the normal derivative, Green's formula can be stated as

$$u(x) = \int_{\delta\mathcal{D}} \left[\varphi(x - s) \frac{\partial u}{\partial n}(s) - u(s) \frac{\partial \varphi}{\partial n}(x - s) \right] ds.$$

Viewing the boundary curves as stroked curves in the limit as the stroke width goes to zero, it becomes obvious that this formula can be rephrased for diffusion curves as follows:

$$\begin{aligned}
u(x) = &\int_{\delta\mathcal{D}} \left[\varphi(x - s) (\frac{\partial u}{\partial n}(s))_+ - g_+(s) \frac{\partial \varphi}{\partial n}(x - s) \right] ds \\
&- \int_{\delta\mathcal{D}} \left[\varphi(x - s) (\frac{\partial u}{\partial n}(s))_- - g_-(s) \frac{\partial \varphi}{\partial n}(x - s) \right] ds \\
= &\int_{\delta\mathcal{D}} \left[\varphi(x - s) \left((\frac{\partial u}{\partial n}(s))_+ - (\frac{\partial u}{\partial n}(s))_- \right) - (g_+(s) - g_-(s)) \frac{\partial \varphi}{\partial n}(x - s) \right] ds.
\end{aligned}$$

As the $\varphi(x - s)(\cdots)$ part cannot introduce any discontinuities we naturally have

$$w(x) = -\int_{\delta\mathcal{D}} (g_+(s) - g_-(s)) \frac{\partial \varphi}{\partial n}(x - s)\, ds.$$

Note that some care must be taken in computing $w(x)$ for $x \in \delta\mathcal{D}$, as the normal derivative of the fundamental solution centered on x is (defined to be) zero at x.

A separation based on Green's representation formula is also used in Zieniuk [33, 34], but not with the explicit intention to remove discontinuities, nor to use this principle in combination with any kind of solver. Rather, Green's formula is used directly to lead to a system of linear equations in which both the solution and the normal derivative of the solution along the boundary appear. This approach is useful when having to deal with Neumann boundary conditions for example.

Once w is determined v can be solved using any of the algorithms discussed (and more) and then, finally, u can be formed simply by adding w to v. This technique is used in my implementation of the AEM for diffusion curves.

4.5 Results

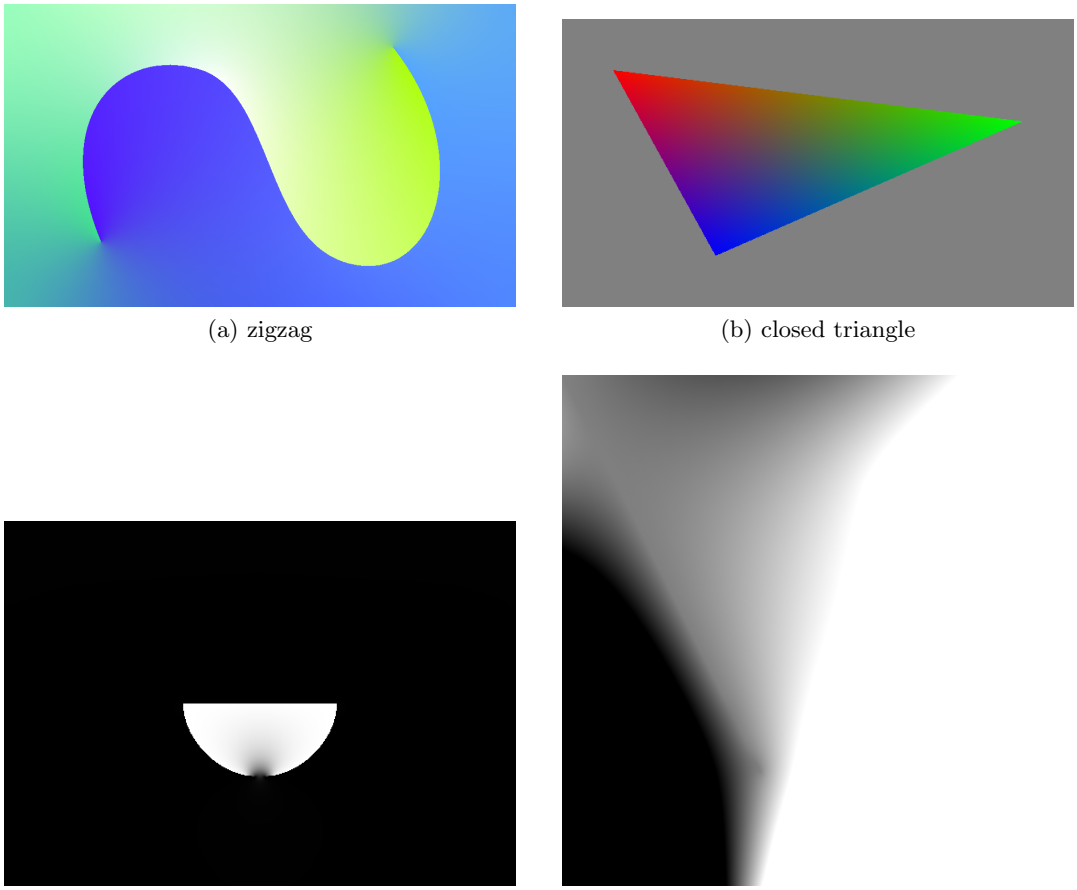
In this section I will show some results obtained with the analytic element method for diffusion curves. The four images used for testing are shown in figure 4.5, the segments used in discretizing the boundaries can be seen in figure 4.6 and figure 4.7 shows the residuals in trying to match the boundary conditions, to give an impression of the precision of the results.

The zigzag image is used throughout this thesis to demonstrate diffusion curves and has a lot of curves. The discretization of the boundary was made to depend on the final resolution, ensuring that the difference between the discretized boundary and the actual boundary is negligible at the rendered resolution. In this case a final resolution of 948×560 was used, yielding 108 segments (+4 for the Neumann boundaries).

The closed triangle image has a very simply analytic solution based on barycentric coordinates (the colors are interpolated linearly along each of the three sides of the triangle). Although one of the nice properties of the analytic element method is that it affords an easy error estimate, this does not necessarily work when Neumann boundaries are involved and should of course always be double-checked. Figure 4.7 shows only greens for this image, indicating that the residual doesn't go above 10^{-6}, and indeed the root-mean-square error of the shown result for the closed triangle image is approximately $4.76 \cdot 10^{-9}$ (the maximum error is $2.87 \cdot 10^{-8}$). This indicates that (in this case) the analytic element method lives up to its name, giving an essentially exact result (even though the result is not trivial in the sense that v is zero or something like that). In comparison, for the variable stencil size method the RMS error is $1.37 \cdot 10^{-2}$.

The "loop" image (so named because the curves sort of resemble a loop) has non-continuous boundary conditions and tries to fill an almost closed space. Its boundaries (apart from the Neumann boundaries at the edges of the image) consist of two horizontal segments at either side of an opening at the bottom of half a circle (which is closed at the top by a linear segment). The half-circle (and its closing segment) has a white color on the inside, all other boundaries are black. As might be expected the AEM does have some difficulty in matching the boundary conditions near the opening, but other than that it seems to cope quite well.

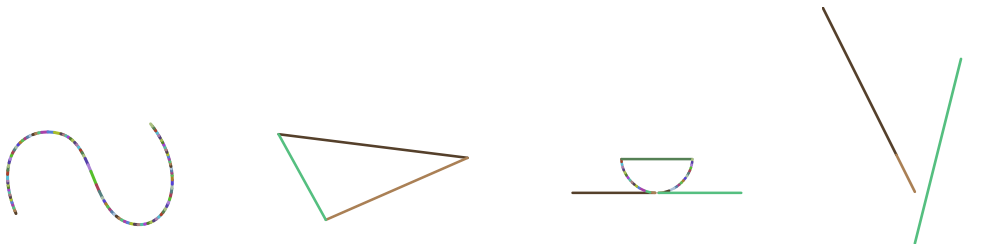
The lines image has two sloping lines as boundary curves, both incident on the boundary and is an example of when the AEM as currently implemented does *not* yield a good result. This is most likely due to a large constant component of the error on the Neumann boundary. In the other images the absolute value of the constant component of the error on the Neumann boundary is roughly 10^{-3} or less, for the lines image it is approximately 0.1.



(a) zigzag (b) closed triangle

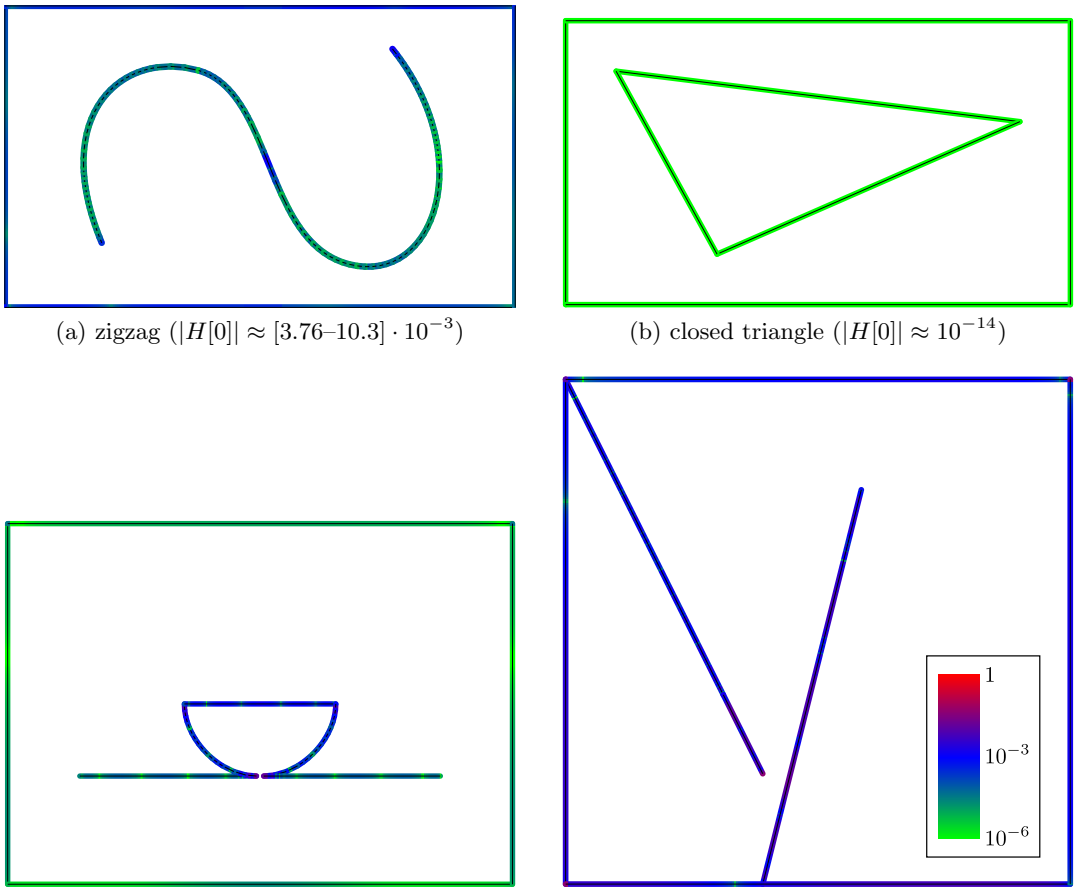
(c) loop (d) lines

Figure 4.5: Images rendered using the analytic element method for diffusion curves. Where applicable curves were subdivided until they were approximately linear (see figure 4.6). In all cases the weights along elements were defined using third-order polynomials.



(a) zigzag (108) (b) closed triangle (3) (c) loop (33) (d) lines (3)

Figure 4.6: Segments used for rendering with the analytic element method for diffusion curves (the number of segments is given between parentheses). None of the segment counts include the four Neumann boundaries at the edges of the images. Note that the extra segment in the lines image was added manually, to aid in being able to match the boundary condition near the tip of the upper-left line.



(a) zigzag ($|H[0]| \approx [3.76\text{–}10.3] \cdot 10^{-3}$)

(b) closed triangle ($|H[0]| \approx 10^{-14}$)

(c) loop ($|H[0]| \approx 8.94 \cdot 10^{-3}$)

(d) lines ($|H[0]| \approx 0.135$)

Figure 4.7: Residuals in matching the boundary conditions for the images shown in figure 4.5. Assuming the extremal length ℓ is not too large (which would increase the influence of $H[0]$) most of the results are more than adequate for 8-bit (per channel) images, which have a smallest representable value of $3.92 \cdot 10^{-3}$. Note that for the top two images the root-mean-square value over the three color channels is used. Also, for the top two images the maximum and minimum value over the three color channels is given for $H[0]$. Here $H[0]$ is the constant component of the error on the Neumann boundary.

Chapter 5

Conclusion

In this thesis I have shown that surface methods for solving the Laplace equation have to deal with a problematic relation between the residual Δu and the error in the solution. Specifically, with surface methods the maximum error in the image can grow quadratically with the size of the image. Existing fast solvers for diffusion curves are not able to overcome this problem sufficiently to be accurate enough for even an 8 bit (per channel) image. In contrast, boundary methods are inherently immune to such problems. It is only in the face of Neumann boundary conditions that similar problems occur with boundary methods, and then only with respect to the residual in satisfying the Neumann boundary conditions, and even then with only a linear dependence on the "thickness" of the region (expressed by the extremal length ℓ).

Motivated by a need for high quality renderings I have developed a boundary method for solving diffusion curves based on the analytic element method. This method has the theoretical advantages of boundary methods and is shown to give good results in practice as well. Furthermore, based on methods developed in the literature it has the potential to be fast as well. For example, in Craig et al. [5] the problem domain is subdivided hierarchically to allow far-field contributions of elements to be combined as much as possible. Having the potential to give both good and fast solutions makes the method developed in this thesis a good addition to the arsenal of solvers for diffusion curves.

Finally, part of the newly developed solver can be reused in combination with other solvers as well to eliminate problems with discontinuities. Most surface methods (if not all) have trouble coping with the endpoints of curves, as either the endpoint should be assigned a single color or the color should depend on the angle from which you are looking. This leads to color bleeding and/or discontinuities were there should be none. By separating the solution into an easily obtained discontinuous part and a continuous part that can be solved as usual, all problems with handling discontinuities across diffusion curves can be avoided.

5.1 Further work

The solver as developed in this thesis could potentially be made faster by using the nested superblock approach developed in Craig et al. [5]. It would be interesting to see if this could lead to a version of the AEM for diffusion curves that is fast enough for interactive use. Especially since, with the current code, the bottle-neck is rendering the image after the solution has been found, at least for images with relatively few elements (including all of the test images shown in this thesis).

Also, in preliminary tests it worked quite well to dynamically split elements so as to

divide the current residual equally over both halves of the element. Ultimately I did not include this in the final implementation, as the performance costs were relatively high and it did not help demonstrate the main point of this thesis. It would, however, be interesting to experiment further with splitting schemes to see how this compares with simply using higher order elements.

It would also be interesting to explore the full benefit of computing the discontinuous part of the solution separately in surface methods. This would be especially interesting for a mesh based solver, like the one developed in Orzan [17], as this would simplify the implementation (no need to split the curves) and could potentially improve the quality of the result, not in the least because there is no need to average the color at endpoints (which can cause color bleeding). But for other grid based solvers it is also interesting, as it allows for a much more straightforward discretization of the boundary.

When looking at grid solvers again it would also be interesting to explore full multigrid methods, or to try and do something similar with the variable stencil size method. Multigrid methods are meant to be able to solve this kind of problem, and I would expect a more elaborate implementation that can "come back" to a coarser scale after having iterated at a finer scale to fare better. But obviously this would have to be tried.

For handling curves it would be interesting to look at alternatives to subdividing. This could include using a conformal map to "bend" the fundamental solution, see for example Kadi and Rockwood [11], Le Grand [12], Steward et al. [24]. One advantage of using curved elements is that you can avoid having corners where they do not need to be, this can look nicer, but certain terms from neighbouring elements can also cancel each other out.

In conclusion, there are a number of things which could improve the quality of the method presented here. For one thing, rasterization artifacts can be quite noticeable along curves in the final rendered image and it would be desirable to use some form of anti-aliasing to combat this. In principle all of the solution should be anti-aliased, but as it is generally quite smooth away from the boundaries it could be advantageous to only anti-alias along the boundary curves. The cleanest way to anti-alias the solution would be to analytically convolve Φ_n with some anti-aliasing filter (like a box filter).

Given the importance of minimizing the constant component of the error on the Neumann boundary, $H[0]$, it could prove beneficial to employ the Galerkin method instead of the collocation method. Also, handling of Neumann boundaries in general could be improved. In some cases, having Neumann boundaries can perhaps even be avoided, but they could potentially also be useful to have explicitly. For example, in Bezerra et al. [1, 2] Neumann boundaries are used to have a kind of "free" boundary which does stop color diffusion but does *not* give any explicit color constraints.

One way to get rid of Neumann boundaries might be to use the AEM to construct conformal maps between the problem domain and some other easy to solve domain (similar to what I did with the 2π-periodic strip domain) and to then use an analytical solution in the easy domain. The advantage would be that the AEM does not have to deal with Neumann boundaries and as such has no problems in constraining the error. Another advantage could be that it might make it possible to deal with very general boundaries (as the geometry of the boundary is captured in the conformal map from the easy domain to the original problem domain).

5.2 Acknowledgements

I would like to thank the authors of Orzan et al. [18], and in particular Alexandrina Orzan, Adrien Bousseau and Holger Winnemöller for being most supportive and providing me with details of their GPU based solver, as well as some excellent literature suggestions. I would also like to thank Randal Barnes and Philippe Le Grand for their swift and useful replies to my questions on how to (efficiently) evaluate analytic elements. Finally I would like to thank my supervisor Jos Roerdink for being most patient with me.

Appendix A

Derivatives

When implementing Neumann boundaries in the analytic element method not only $\Phi_n(z)$ has to be evaluated, but also its (normal) derivative. For this it suffices to compute the complex derivative of $\Phi_n(z)$. To see that this is true, consider the partial derivatives of z to x_0 and x_1: $\partial z/\partial x_0 = (\partial/\partial x_0)(x_0 + x_1\mathrm{i}) = 1$ and $\partial z/\partial x_1 = \mathrm{i}$. Using these identities $(\partial/\partial z)\Phi_n(z)$ gives all the information needed to compute the derivative in any direction.[1]

Also, as later on we will use the normal derivative of $\Phi_n(z)$ as part of the solution as well, we will also need second derivatives. Here, again, it suffices to simply compute the second derivative to z and use identities like

$$\frac{\partial^2\Phi_n(z)}{\partial z^2}\frac{\partial z^2}{\partial x_0^2} = \frac{\partial^2\Phi_n(z)}{\partial z^2}\left(\frac{\partial z}{\partial x_0}\right)^2 = \frac{\partial^2\Phi_n(z)}{\partial z^2}.$$

Note the subtlety in placing the square, $\partial z^2/\partial x_0^2$ is in general not the same as $\partial^2 z/\partial x_0^2$.

For the near-field formulation

$$\begin{aligned}
\frac{\partial}{\partial z}\Phi_n^M(z) &= \frac{\partial}{\partial z}\frac{1}{n+1}\Bigg[(1-z)Q_n(1,z)\ln(z-1) + (1+z)Q_n(-1,z)\ln(z+1) \\
&\qquad - \int_{-1}^{1} Q_n(t,z)\,dt\Bigg] \\
&= \frac{1}{n+1}\Bigg[\frac{\partial}{\partial z}\left(1-z^{n+1}\right)\ln(z-1) - \frac{\partial}{\partial z}\left((-1)^{n+1}-z^{n+1}\right)\ln(z+1) \\
&\qquad + (1-z)Q_n(1,z)\frac{1}{z-1} + (1+z)Q_n(-1,z)\frac{1}{z+1} \\
&\qquad - \int_{-1}^{1} Q_n^{(0,1)}(t,z)\,dt\Bigg] \\
&= \frac{1}{n+1}\Bigg[\left(-(n+1)z^n\right)\ln(z-1) - (n+1)\left(-z^n\right)\ln(z+1) \\
&\qquad + Q_n(-1,z) - Q_n(1,z) - \int_{-1}^{1} Q_n^{(0,1)}(t,z)\,dt\Bigg] \\
&= z^n\left[\ln(z+1) - \ln(z-1)\right] - \frac{1}{n+1}\left[\int_{-1}^{1} Q_n^{(1,0)}(t,z)\,dt + \int_{-1}^{1} Q_n^{(0,1)}(t,z)\,dt\right] \\
&= z^n\left[\ln(z+1) - \ln(z-1)\right] - \int_{-1}^{1} Q_{n-1}(t,z)\,dt.
\end{aligned}$$

[1] This can also be seen in the light of the Cauchy-Riemann equations (see section 3.2), which are indeed satisfied by $\Phi_n(z)$.

Again, this can be evaluated efficiently and easily using a recurrence equation. Similarly, the second derivative can now be found as (note that for $n = 0$ the terms multiplied with n should indeed be considered zero)

$$\begin{aligned}
\frac{\partial^2}{\partial z^2}\Phi_n^M(z) &= \frac{\partial}{\partial z}\left[z^n\left[\ln(z+1)-\ln(z-1)\right]-\int_{-1}^{1} Q_{n-1}(t,z)\,dt\right] \\
&= nz^{n-1}\left[\ln(z+1)-\ln(z-1)\right]+\frac{z^n}{z+1}-\frac{z^n}{z-1}-\int_{-1}^{1} Q_{n-1}^{(0,1)}(t,z)\,dt \\
&= nz^{n-1}\left[\ln(z+1)-\ln(z-1)\right]+\frac{z^n}{z+1}-\frac{z^n}{z-1} \\
&\quad -\int_{-1}^{1} nQ_{n-2}(t,z)-Q_{n-1}^{(1,0)}(t,z)\,dt \\
&= nz^{n-1}\left[\ln(z+1)-\ln(z-1)\right]+\frac{z^n}{z+1}-\frac{z^n}{z-1} \\
&\quad -n\int_{-1}^{1} Q_{n-2}(t,z)\,dt+Q_{n-1}(1,z)-Q_{n-1}(-1,z) \\
&= nz^{n-1}\left[\ln(z+1)-\ln(z-1)\right]+\frac{z^n}{1+z}+\frac{z^n}{1-z} \\
&\quad -n\int_{-1}^{1} Q_{n-2}(t,z)\,dt+\frac{1-z^n}{1-z}+\frac{(-1)^n-z^n}{1+z} \\
&= nz^{n-1}\left[\ln(z+1)-\ln(z-1)\right]+\frac{(-1)^n}{1+z}+\frac{1}{1-z}-n\int_{-1}^{1} Q_{n-2}(t,z)\,dt.
\end{aligned}$$

For the far-field formulation

$$\begin{aligned}
\frac{\partial}{\partial z}\int_{-1}^{1} t^n\ln(z-t)\,dt &= \mathrm{B}(1,n+1)\left[\frac{\partial}{\partial z}\ln(z-1)+(-1)^n\frac{\partial}{\partial z}\ln(z+1)\right] \\
&\quad +\sum_{i=1}^{\infty}\frac{1}{i}\mathrm{B}(i+1,n+1)\left[(-1)^{n+1}\frac{\partial}{\partial z}\frac{1}{(1+z)^i}-\frac{\partial}{\partial z}\frac{1}{(1-z)^i}\right] \\
&= \mathrm{B}(1,n+1)\left[\frac{1}{z-1}+(-1)^n\frac{1}{z+1}\right] \\
&\quad +\sum_{i=1}^{\infty}\mathrm{B}(i+1,n+1)\left[(-1)^{n+2}\frac{1}{(1+z)^{i+1}}-\frac{1}{(1-z)^{i+1}}\right] \\
&= \sum_{i=0}^{\infty}\mathrm{B}(i+1,n+1)\left[(-1)^{n+2}\frac{1}{(1+z)^{i+1}}-\frac{1}{(1-z)^{i+1}}\right]
\end{aligned}$$

and

$$\frac{\partial^2}{\partial z^2}\int_{-1}^{1} t^n\ln(z-t)\,dt = \sum_{i=0}^{\infty}(i+1)\mathrm{B}(i+1,n+1)\left[(-1)^{n+3}\frac{1}{(1+z)^{i+2}}-\frac{1}{(1-z)^{i+2}}\right].$$

Bibliography

[1] Hedlena Bezerra, Elmar Eisemann, Doug DeCarlo, and Joëlle Thollot. Diffusion Constraints for Vector Graphics. In *NPAR 2010: Proceedings of the 8th International Symposium on Non-photorealistic Animation and Rendering*, pages 35–42, New York, NY, USA, 2010. ACM Press. ISBN 978-1-4503-0125-1. doi: 10.1145/1809939.1809944.

[2] Hedlena Bezerra, Elmar Eisemann, Doug Decarlo, and Joëlle Thollot. Controllable Diffusion Curves. Technical report, Institut National de Recherche en Informatique et en Automatique, January 2010.

[3] W. Chen, L. J. Shen, Z. J. Shen, and G. W. Yuan. Boundary knot method for Poisson equations. *Engineering Analysis with Boundary Elements*, 29(8):756–760, August 2005. ISSN 0955-7997. doi: 10.1016/j.enganabound.2005.04.001.

[4] Richard Courant. *Dirichlet's Principle, Conformal Mapping, and Minimal Surfaces*. Interscience Publishers, November 1950. ISBN 0486445526.

[5] James R. Craig, Igor Janković, and Randal Barnes. The Nested Superblock Approach for Regional-Scale Analytic Element Models. *Ground Water*, 44(1):76–80, 2006. doi: 10.1111/j.1745-6584.2005.00081.x.

[6] James H. Elder. Are Edges Incomplete? *International Journal of Computer Vision*, 34(2):97–122, August 1999. ISSN 09205691. doi: 10.1023/A:1008183703117.

[7] Zeev Farbman, Gil Hoffer, Yaron Lipman, Daniel Cohen-Or, and Dani Lischinski. Coordinates for Instant Image Cloning. *ACM Transactions on Graphics*, 28(3):1–9, August 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531373.

[8] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20 (1):19–27, March 2003. doi: 10.1016/S0167-8396(03)00002-5.

[9] Igor Janković and Randal J. Barnes. High-order line elements in modeling two-dimensional groundwater flow. *Journal of Hydrology*, 226(3-4):211–223, December 1999. ISSN 00221694. doi: 10.1016/S0022-1694(99)00140-7.

[10] Stefan Jeschke, David Cline, and Peter Wonka. A GPU Laplacian Solver for Diffusion Curves and Poisson Image Editing. *ACM Transaction on Graphics*, 28(5):116:1–116:8, December 2009. ISSN 0730-0301. doi: 10.1145/1618452.1618462.

[11] Zafer Kadi and Alyn Rockwood. Conformal maps defined about polynomial curves. *Computer Aided Geometric Design*, 15(4):323–337, April 1998. ISSN 01678396. doi: 10.1016/S0167-8396(97)00035-6.

[12] Philippe Le Grand. *Advanced curvilinear shapes for object centered modeling of groundwater flow with the analytic element method*. PhD thesis, l'Ecole Nationale Supérieure des Mines de Saint-Etienne, l'Université Jean Monnet, 2003.

[13] Gregory Lecot and Bruno Lévy. ARDECO: Automatic Region DEtection and COnversion. In *Eurographics Symposium on Rendering*, 2006. doi: 10.2312/EGWR/EGSR06/349-360.

[14] Raphael L. Levien. *From Spiral to Spline: Optimal Techniques in Interactive Curve Design*. PhD thesis, University of California, Berkeley, December 2009.

[15] Zi-Cai Li. Combinations of method of fundamental solutions for Laplace's equation with singularities. *Engineering Analysis with Boundary Elements*, 32(10):856–869, October 2008. ISSN 0955-7997. doi: 10.1016/j.enganabound.2008.01.002.

[16] Zi-Cai Li. The method of fundamental solutions for annular shaped domains. *Journal of Computational and Applied Mathematics*, 228(1):355–372, June 2009. ISSN 0377-0427. doi: 10.1016/j.cam.2008.09.027.

[17] Alexandrina Orzan. *Contour-based Images: Representation, Creation and Manipulation*. PhD thesis, INPG, June 2009.

[18] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. In *SIGGRAPH 2008*, volume 27, pages 1–8, New York, NY, USA, 2008. ACM. doi: 10.1145/1360612.1360691.

[19] Jörg Ostrowski, Zoran Andjelic, Mario Bebendorf, Bogdan Cranganu-Cretu, and Jasmin Smajic. Fast BEM-solution of Laplace problems with H-matrices and ACA. *IEEE Transactions on Magnetics*, 42(4):627–630, April 2006. ISSN 0018-9464. doi: 10.1109/TMAG.2006.871642.

[20] M. Othman and A. R. Abdullah. An efficient four points modified explicit group poisson solver. *International Journal of Computer Mathematics*, 76(2):203–217, 2000. doi: 10.1080/00207160008805020.

[21] Yehuda Pinchover and Jacob Rubinstein. *An Introduction to Partial Differential Equations*. McGraw-Hill series in electrical engineering. McGraw-Hill, 1989. ISBN 0-521-61323-1.

[22] Marco Saraniti, Achim Rein, Günther Zandler, Peter Vogl, and Paolo Lugli. An efficient multigrid Poisson solver for device simulations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(2):141–150, February 1996. ISSN 02780070. doi: 10.1109/43.486661.

[23] G. Speyer, D. Vasileska, and S. M. Goodnick. Efficient Poisson Solver for Semiconductor Device Modeling Using the Multi-Grid Preconditioned BiCGSTAB Method. *Journal of Computational Electronics*, 1(3):359–363–363, October 2002. ISSN 15698025. doi: 10.1023/A:1020747508122.

[24] David R. Steward, Philippe Le Grand, Igor Janković, and Otto D. L. Strack. Analytic formulation of Cauchy integrals for boundaries with curvilinear geometry. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 464(2089): 223–248, January 2008. doi: 10.1098/rspa.2007.0138.

[25] Otto D. L. Strack. *Groundwater Mechanics*. 1989.

[26] Otto D. L. Strack. Principles of the analytic element method. *Journal of Hydrology*, 226(3-4):128–138, December 1999. ISSN 00221694. doi: 10.1016/S0022-1694(99)00144-4.

[27] Prasad S. Sumant and Andreas C. Cangellaris. Algebraic multigrid Laplace solver for the extraction of capacitances of conductors in multi-layer dielectrics. *Int. J. Numer. Model.*, 20(5):253–269, 2007. doi: 10.1002/jnm.650.

[28] Kenshi Takayama, Olga Sorkine, Andrew Nealen, and Takeo Igarashi. Volumetric Modeling with Diffusion Surfaces. *ACM Transactions on Graphics*, 29(5), 2010. doi: 10.1145/1882261.1866202.

[29] Balša Terzić and Ilya V. Pogorelov. Wavelet-based Poisson solver for use in particle-in-cell simulations. *Annals of the New York Academy of Sciences*, 1045:55–67, June 2005. ISSN 0077-8923. doi: 10.1196/annals.1350.006.

[30] Đào Trọng Thi and A. T. Fomenko. *Minimal Surfaces, Stratified Multivarifolds, and the Plateau Problem*, volume 84 of *Translations of Mathematical Monographs*. American Mathematical Society, February 1991. ISBN 0821845365.

[31] Yaakov Tsaig and David L. Donoho. Extensions of compressed sensing. *Signal Processing*, 86(3):549–571, 2006. doi: 10.1016/j.sigpro.2005.05.029.

[32] Wen-Ming Yan and Kuo-Liang Chung. A Fast Algorithm for Solving Special Tridiagonal Systems. *Computing*, 52(2):203–211, June 1994. ISSN 0010-485X. doi: 10.1007/BF02238076.

[33] Eugeniusz Zieniuk. Potential problems with polygonal boundaries by a BEM with parametric linear functions. *Engineering Analysis with Boundary Elements*, 25(3): 185–190, March 2001. ISSN 0955-7997. doi: 10.1016/S0955-7997(01)00035-2.

[34] Eugeniusz Zieniuk. Bézier curves in the modification of boundary integral equations (BIE) for potential boundary-values problems. *International Journal of Solids and Structures*, 40(9):2301–2320, May 2003. ISSN 00207683. doi: 10.1016/S0020-7683(03)00050-7.