Comparison of Parallel Solution Techniques for the Eikonal Equation

Owen Thomas[b], Philip Crowley[c], Thomas Prince[d], James Anderson[a], Jotun Hein[a], Joe Pitt-Francis[e]

[a]*Department of Statistics, University of Oxford, 1 South Parks Road, Oxford, OX1 3TG, United Kingdom*
[b]*Owen's House*
[c]*Phil's House*
[d]*Tom's House*
[e]*Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK*

Abstract

In this work, Finite Element Method (FEM) solvers and the Fast Marching Method (FMM), a graph-based implementation, are developed and applied to the Eikonal equation to model wave propagation in cardiac electrophysiology. This work builds on the methods existing in the Chaste (Cancer, Heart and Soft Tissue Environment) simulation environment. The graph-based techniques are found to be computationally faster than the FEM techniques, although FEM methods provide greater control over errors, especially in regions of high wavefront curvature. Both methods are distributed in parallel, and their scaling and timing behaviour is compared.

Keywords: Eikonal, Cardiac, Electrophysiology, Fast Marching, Chaste

1. Introduction

The heart has historically presented a challenging and complex modelling task for the computational scientist, its behaviour being defined by highly non-linear relationships between electrical, mechanical and chemical properties of the tissue. After Hodgkin and Huxleys initial work in mathematically defining the behaviour of the action potential [6], electrophysiology was expanded into the field of cardiology with the creation of the Bidomain model [8]. Given the assumption of linearly related intra- and extra-cellular anisotropies, simpler monodomain model can be obtained, defined as follows [2]:

$$\chi(\partial_t v + I_{ion}(v, \mathbf{w}) - I_{st}(t, x)) = \nabla.(\sigma_m(x)\nabla v), \quad x \in \Omega \tag{1}$$

$$\partial_t \mathbf{w} = g(v, \mathbf{w}), \quad x \in \Omega \tag{2}$$

Here, χ is the surface area to volume ratio of the cells, I_{st} and I_{ion} are the stimulation and ionic surface currents, v is the transmembrane potential, $\mathbf{w}$ is a vector containing the gating variables, $g(v, \mathbf{w})$ describes the ionic model, and $\sigma_m^{-1} = \sigma_i^{-1} + \sigma_e^{-1}$, where σ_i and σ_e are the intracellular and extracellular conductivity tensors, respectively. The model takes the following boundary conditions:

$$\nabla v.\mathbf{n} = 0, \quad \text{on} \quad \partial\Omega \tag{3}$$

$$v_0(x) = v(0, x), \quad \mathbf{w_0} = \mathbf{w}(0), \quad x \in \Omega \tag{4}$$

This model has been applied and simulated successfully in the past [11], but a full dynamical simulation is numerically intensive. It is consequently of interest to find methods to learn about cardiac wave propagation with fewer computational demands.

One such approach is to use the Eikonal equation, a first-order nonlinear differential equation describing the spatial dependence of the excitation time given the stimulus and boundary conditions [5]. It can be derived from the monodomain model, with assumptions and discretisation in both space and time, and is generally of the form:

$$K\sqrt{g}/c_m\sqrt{\nabla t^T \sigma^T \nabla t} = 1 \tag{5}$$

where t is the excitation time, σ is a conductivity matrix, g a membrane conductance per unit volume, c_m a membrane capacitance per unit volume and K a dimensionless constant. A solution to this equation represents a spatial map of the excitations time of the tissue, providing information about the wavefront behaviour without presenting the numerical domands of a full dynamical solution. It can be rewritten to concisely demonstrate its mathematical properties as:

$$v\sqrt{\nabla t^T P \nabla t} \tag{6}$$

where $v = K\sqrt{g\sigma}/c_m$ is the space-dependent propagation speed of the cardiac wave, and P is a tensor introducing anisotropy to the system. In the heterogeneous case, v varies as a function of the spatial co-ordinates, whereas in the homogeneous case it is constant across the mesh. In the simplifying case of an isotropic medium, it reduces to the "classical" Eikonal equation, of the form:

$$v|\nabla t| = 1 \tag{7}$$

The Eikonal equation lends itself to both discretised graph-theoretic methods and traditional Partial Differential Equation (PDE) solution techniques. In this paper, we implement and compare three parallel solution approaches. Firstly we use Dijkstra's method, which uses a node-based approach that systematically excites appropriate neighbouring nodes [3]. Secondly we pursue the related Fast Marching Method (FMM), which similarly uses a node-based approach but incorporates more local information when performing updates [7]. Thirdly we also apply a more general Finite Element Method (FEM) for solving PDEs to the specific Eikonal problem.

These algorithms were implemented in the Chaste (Cancer, Heart and Soft Tissue Environment) codebase, a software suite designed for general-purpose biological modelling [4]. The package already contained an application of Dijkstra's algorithm and general differential equation solvers, which were built upon to compare the algorithmic solution techniques.

2. Dijkstra, Fast Marching and Graph Theory Approaches

2.1. Dijkstra's Algorithm

Graph theoretic modelling has already been pursued in the Chaste codebase, based on Dijkstra's algorithm [3]. The implementation treats the mesh as undirected connected graph with vertices defined by the mesh's nodes, with the edges defined by the boundaries of the graph's elements. The edges are weighted with the product of their Euclidean length and the wavefront velocity. Dijkstra's algorithm relies on establishing a set of "alive" nodes in the discretised mesh and estimating the time taken for the wavefront to propagate along edges to local nodes. Dijkstra's algorithm will necessarily return a time which is the global minumum time for the wavefront to reach a given point.

However, this method is fundamentally ill-suited to electromagentic wave propagation because of the unphysical constraint that the shortest paths across the mesh travel exclusively on the edges of the graph. This requirement prevents realistic propagation of the wavefront and neglects the physical information between nodes, contributing to global error.

2.1.1. Outline of the Algorithm

For an undirected connected graph G with edges of weights $\tau[u,v]$, the following outlines an algorithm for finding the wavepropagation time $t[u]$ for a node to reach a node u using Dijkstra's algorithm.

The algorithm starts by asserting that both the set of finalised nodes S and the queue Q are empty. The queue is then populated with the source nodes with wavefront-time $t[u] = 0$. The algorithm runs by extracting the node with the lowest t from Q, and putting it in S. Adjacent nodes are then checked to see if the new information can be used to improve their estimates. This process is repeated until the queue is empty, and therefore all the nodes have been finalised. See supplementary material for pseudocode Appendix B.2.

2.2. The Fast Marching Method

The Fast Marching Method (FMM) is a well established algorithmic approach to find approximate solutions to the Eikonal equation [7]. As a one-pass global algorithm it resembles Dijkstra's method, but differs substantially in the manner in which it uses local information to estimate the wavefront travel times. Like Dijkstra's algorithm, the process begins by asserting a subset of source nodes to be zero distance. The algorithm then uses the initial condition to estimate the time at which the wavefront will reach neighbouring nodes. Each iteration of the algorithm then consists of the finalisation of one of these nodes, and use of its wavefront-time to estimate that of neighbouring nodes.

The fundamental improvement on Dijkstra's method is in how FMM always uses the maximal amount local information to estimate nodes' wavefront-time. A physical interpretation of this implementation is that where Dijkstra's algorithm constrained geodesics to the edges of the mesh, FMM allows them to pass through the volume of elements. Using more information allows for a closer approximation to an analytical solution, although it can be expected to be more computationally expensive. Pseudocode can be found in the supplementary material Appendix B.3.

For example, if a wave propagating in the 2-D (x, y) plane forms a continuous surface in (x, y, t) space, the solution satisfies the Eikonal equation $|\nabla T|\, F(x, y) = 1$ where $F(x, y)$ is the wavespeed. FMM works by constructing the solution surface to be piecewise flat. In an n-dimensional space, n nodes and their wavefront-times will give a uniquely defined surface, given the Eikonal equation and the causality condition that the surface is always increasing. This surface then provides an estimate for the wavefront-time at the $n + 1$-th node. Defining the region bounded by these as an element, the FMM provides a method for obtaining a piecewise-flat solution for a wave propogating through space over a discretised mesh.

To ensure that the solution is physical it must satisfy both causality and an entropic condition. Causality is upheld by explicitly preventing the algorithm from taking estimates for a node's wavefront-time lower than the times at the nodes used to generate the prediction. It is also necessary to check that the new estimate corresponds to a wave propagating through the region of space associated with the element that produced it. The entropic condition is maintained by requiring that the wavefront-time of a node in the final solution must be given by the lowest estimate calculated during the algorithm. This prevents the wave from travelling over any regions repeatedly and updating any node wavefront-time more than once.

2.2.1. Simple Case: Update Procedure in 1-dimension

In one dimension the mesh is constructed of line segments bounded by a node at each end, formally described as 1-simplexes. To estimate a node wavefront-time using a source node, the algorithm constructs a line that intersects the source node with a gradient defined by the Eikonal equation. Two possible lines can be constructed and FMM uses only the one which results in a larger time for the target node. The smaller unphysical solution is discarded, this is indicated by the dot in figure Appendix E. FMM crosses the entire mesh in a single pass, constructing an element-wise solution. In the one-dimensional case, FMM and Dijkstra are confined to the edges of the mesh and hence arrive at the same solution.

2.2.2. Generalization of update procedure to n-dimensional case

An n-dimensional space divided into n-dimensional simplexes will have a corresponding $n+1$-dimensional solution space, incorporating both space and time values. Finding a piecewise-linear solution corresponds to fitting a plane to the $n+1$-dimensional space, of the form $T(\mathbf{x}) = \mathbf{x} \cdot \mathbf{a} + b$ for some unknown constant scalar b and constant vector $\mathbf{a}$. Given the wavefront-times $T_i = T(\mathbf{x}_i)$ for the n source nodes $\mathbf{x}_i$, there are n equations with known T values:

$$\begin{aligned} T_1 &= \mathbf{x}_1 \cdot \mathbf{a} + b \\ T_2 &= \mathbf{x}_2 \cdot \mathbf{a} + b \\ &\vdots \\ T_n &= \mathbf{x}_n \cdot \mathbf{a} + b \end{aligned} \tag{8}$$

and one equation for the target node labelled as 0, with unknown solution T_0:

$$T_0 = \mathbf{x}_0 \cdot \mathbf{a} + b \tag{9}$$

A system of linear equations can be obtained by subtracting Equation (9) from each of the Equations (8):

$$\begin{pmatrix} T_1 - T_0 \\ T_2 - T_0 \\ \vdots \\ T_n - T_0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_0 \\ \mathbf{x}_2 - \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_i - \mathbf{x}_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \tag{10}$$

Equation 10 becomes $\boldsymbol{\tau} = \mathbf{J}^T\mathbf{a}$, recognising the matrix of spatial co-ordinates to be the transpose of the Jacobian matrix of the simplex cell, and defining the vector of times to be $\boldsymbol{\tau}$.

The Eikonal equation $|\nabla\tau|\,F(\mathbf{x}) = 1$ imposes the constraint $|a|\,F(\mathbf{x}) = 1$. F is constant over elements for a piecewise flat mesh, such that $|a|\,F_{ele} = 1$, where F_{ele} is the wavespeed defined on each element. The Eikonal equation can hence be satisfied by inverting Equation (10) to the form $\boldsymbol{a} = \mathbf{J}^{-T}\boldsymbol{\tau}$ and taking the inner product of $\mathbf{a}$ with itself, giving:

$$F_{ele}^{-2} = |\mathbf{a}|^2 = \mathbf{a}^T\mathbf{a} = \boldsymbol{\tau}^T\mathbf{J}^{-1}\mathbf{J}^{-T}\boldsymbol{\tau} = \boldsymbol{\tau}^T\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\boldsymbol{\tau}$$

Defining $\mathbf{M} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}$ and recalling $\tau_i = T_i - T_0$, this becomes the quadratic equation:

$$F_{ele}^{-2} = \sum_{ij}^{n} M_{ij}\left(T_i - T_0\right)\left(T_j - T_0\right) \tag{11}$$

Equation 11 is a quadratic equation, and can be solved for T_0 with standard methods. However, for this to become a new estimate for the wavefront-time at $\mathbf{x}_0$, it must uphold causality and hence correspond to a wavefront propagating from the source nodes with vertices $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$.

This can be verified by evaluating the intersection between the vector pointing back along the wavefront from $\mathbf{x}_0$, and the $n-1$-dimensional surface containing the source nodes. The vector is given by $\mathbf{x}_0 + u_0\nabla T$, which by Equation (10) is equal to $\mathbf{x}_0 + u_0\mathbf{J}^{-T}\boldsymbol{\tau}$ for some scalar u_0. The source nodesurface is given by $\mathbf{x}_1 + v_2\left(\mathbf{x}_2 - \mathbf{x}_1\right) + v_3\left(\mathbf{x}_3 - \mathbf{x}_1\right) + \cdots + v_n\left(\mathbf{x}_n - \mathbf{x}_1\right)$ for some scalars v_i. The intersection between these subspaces is given by:

$$\begin{aligned}
\mathbf{x}_0 + u_0\mathbf{J}^{-T}\boldsymbol{\tau} =& \mathbf{x}_1 + v_2\left(\mathbf{x}_2 - \mathbf{x}_1\right) + v_3\left(\mathbf{x}_3 - \mathbf{x}_1\right) + \cdots + v_n\left(\mathbf{x}_n - \mathbf{x}_1\right)\\
=& \left(1 - \sum_{i=2}^{n} v_i\right)\mathbf{x}_1 + \sum_{i=2}^{n} v_i\mathbf{x}_i\\
u_0\mathbf{J}^{-T}\boldsymbol{\tau} =& \left(1 - \sum_{i=2}^{n} v_i\right)\left(\mathbf{x}_1 - \mathbf{x}_0\right) + \sum_{i=2}^{n} v_i\left(\mathbf{x}_i - \mathbf{x}_0\right)
\end{aligned}$$

which can be recast as the linear system of equations:

$$u_0\mathbf{J}^{-T}\boldsymbol{\tau} = \begin{pmatrix} \uparrow & & \uparrow \\ \mathbf{x}_1 - \mathbf{x}_0 & \cdots & \mathbf{x}_n - \mathbf{x}_0 \\ \downarrow & & \downarrow \end{pmatrix}\begin{pmatrix} 1 - \sum_{i=2}^{n} v_i \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{pmatrix} = \mathbf{J}\mathbf{v}$$

In the last step $\mathbf{v}$ has been defined as the vector containing unknown scalar coefficients v_i. The intersection of the surface $\mathbf{J}\mathbf{v}$ and the line $u_0\mathbf{J}^{-T}\boldsymbol{\tau}$, lies in the $n-1$-simplex of source nodes if $(\forall v_i \in \mathbf{v} : 0 \leq v_i \leq 1)$. However, the relationship between $\mathbf{v}_1$ and the other elements of $\mathbf{v}$ means that it is sufficient to check that all $\mathbf{v}_i$ are the same sign. This weaker criterion can be verified without having to seperate the scalar u_0 from the elements of $\mathbf{v}$. Consequently, providing that every element of the vector $u_0^{-1}\mathbf{v} = \mathbf{J}^{-1}\mathbf{J}^{-T}\boldsymbol{\tau} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\boldsymbol{\tau} = \mathbf{M}\boldsymbol{\tau}$ has the same sign, the solution corresponds to a wave that originated from the allowable source-node subspace.

If the number of source nodes is lower than the dimensionality of the space, the Jacobian matrix is not initially square. In this case, the co-ordinate system can be rotated to a basis where all of the orthonormal basis vectors are either orthogonal or parallel to the plane in which the nodes lie. The vector displacements between the nodes can then be completely expressed by a vector with fewer entries. In the new basis, the Jacobian is square and the rest of the process follows as before.

2.2.3. Outline of the Algorithm

This section outlines how FMM obtains a piecewise linear approximate solution to the Eikonal equation $|\nabla T| F(\mathbf{x}) = 1$. It is performed on a space filling mesh of n-simplexes (elements) sharing vertices (nodes).

FMM starts by asserting that both the set of finalised nodes S and the queue Q are empty. The queue is then populated with the source nodes with wavefront-time $t = 0$. The algorithm runs by extracting the node u with the lowest t from Q, and placing it in S. FMM then iterates over unfinalised nodes v which are also members of the elements E of which u is a member. For each node v the algorithm attempts an estimate of the wavefront-time, initially using the maximal set of local source nodes $E \setminus S$.

If the estimate obtained this way fails to satisfy causality or entropic conditions, then FMM will try all the possible subsets L of $E \setminus S$ made by omitting one local source node. If this provides unphysical estimates then the algorithm continues to exclude an incrementally greater number of nodes until an acceptable estimate is found. This will be found before the number of source nodes falls to zero, as a physical solution always exists in the one dimensional case. If the estimate found is lower than the current best estimate, it is stored and the node v is placed in Q. The node with the next-lowest t is then extracted from Q as the new u and process restarts. This is repeated until the queue is empty and all the nodes have been finalised.

2.2.4. Error Accumulation

The approximation that the solution is a piecewise linear surface in $(\mathbf{x}, t)$ brings with it the secondary approximation that the wavefront is piecewise linear at any point in the solution. This introduces errors since the optimal source point on the real continuous wavefront will not generally correspond to that implicitly calculated by the FMM. The single pass nature of the FMM means that this geometric mismatch leads to error accumulation at a rate roughly proportional to the wavefront curvature.

This means that at a point in the solution $\mathbf{x}_j$ connected to the source by a geodesic G has an accumulated error in T_j with a value of δT_j. For a wavefront with curvature κ and characteristic element length d, the value of T can be roughly estimated as:

$$T \sim \sum_{i \in G}^{j} |\nabla T|_i \, d \tag{12}$$

The accumulated error δT can be similarly estimated as:

$$\delta T_j \sim \sum_{i \in G}^{j} \frac{|\nabla T|_i}{\kappa_i} \left(1 - \sqrt{1 - \left(\frac{1}{2} \kappa_i d \right)^2} \right) \sim \frac{1}{8} \sum_{i \in G}^{j} |\nabla T|_i \, \kappa_i d^2 = \frac{1}{8} \sum_{i \in G}^{j} F_i^{-1} \kappa_i d^2 \tag{13}$$

Consequently, it can be seen that the important constraint on d for $T_i \gg \delta T_i$ is that $\kappa d \ll 1$. In regions where the inequality $\kappa d \ll 1$ is not satisfied the error increases very quickly. A good example of this kind of behaviour is the solution over a disk, where the curvature κ diverges as $r \to 0$ at the apex of the cone as seen in Figure 2. A one-dimensional plot of this information together with a plot of δT_j is shown in Figure E.10.

There are several ways of approaching the problem of curvature causing accumulated error.

1. Refining the mesh everywhere. This will help with accuracy, but will have a high cost in terms of computational time, and will be of minor benefit in regions where curvature is not significant.
2. Intelligent mesh refinement, targeting areas where curvature of the wavefront is higher. This could be done by evaluating the curvature dynamically, which may significantly slow down the algorithm's speed. Alternatively, it could be possible target areas for refinement which are known *a priori* to have significant curvature. An example of this is the immediate neighbourhood of a point source boundary node.
3. Coupling FMM with another method. Potential partners for FMM range from relatively minor adjustments, such as an algorithm for quadratic interpolation, to complete hybridization with a scheme that does not exhibit error accumulation, such as the Finite Element Method.

3. Solving the Eikonal Equation with the Finite Element Method

It is naturally of interest to consider conventional numerical approaches to solving nonlinear Partial Differential Equations. The Eikonal equation contains the added complications of anisotropic meshes and discrete boundary conditions. Due to need for time efficiency and the unstructured nature of the mesh, the appropriate scheme is the Finite Element Method (FEM). FEM was also favorable by virtue of its high level of use in Chaste, but significant work has also been done using the Finite Volume Method, primarily in fluid dynamics [12]. The formulation used for the Eikonal equation is a Navier Stokes form [10]:

$$|\sigma \nabla \phi|^2 = 1 - \Gamma \nabla^2 \phi \tag{14}$$

Where $\sigma \in V^* \otimes V^*, V \in \mathbb{R}^3$ is the conductivity tensor, ϕ is the wavefront time and Γ is the smoothing term, necessary to ensure the convergence of the numerical methods.

In this investigation the Γ term is spatially homogeneous and iteratively reduced in order to approximately solve the unsmoothed equation. While a constant Γ produces good iterative solutions, work is needed to determine a problem-specific, spatially varying form of this smoothing term. Over the unit disc, for example, it has been shown that a Γ proportional to the radius gives very accurate solutions [10]. This radial dependence of Γ suggests that smoothing is required to regulate the gradient near the boundary and, once established, can fill the rest of the surface. However, it is naturally difficult to predict *a priori* where smoothing is most needed, particulary over different meshes, boundary conditions and conductivity tensors.

3.1. Description of FEM solutions

A thorough treatment of numerical techniques for solving nonlinear PDEs, along with a treatment of FEM for the simpler linear case, is given in [9]. For the linear case, FEM aims to reduce an infinite dimensional inversion problem to finite dimensions by laying a mesh over the domain. It then takes a vector space formed by the linear span of some basis functions. The basis functions are taken prototypically as piecewise linear over the elements, with each basis function being non-zero over a small number of elements. Consequently, the matrix inversion necessary to solve the PDE reduces to a sparse matrix which can be readily inverted.

Applying FEM to the Eikonal equation is modified by two complicating factors. Firstly, a linearisation is required, since it is still not easy to solve the nonlinear inversion problem over the approximating space. Secondly, the smoothing term must be steadily reduces such that, as the solution converges, it is forced closer to a solution of the completely unsmoothed equation. In order to reduce the Laplacian term in Equation 14 to a first-order term, the Newton Iteration requires a weak form of the problem. The weak form is then inverted, taking boundary conditions into account.

3.2. Formulating the Eikonal equation for FEM

The first challenge when using FEM within the Chaste framework is in the linearization: it is computationally advantageous to compute the Jacobian rather than letting the numerical solver find and verify it with the numerical Jacobian finder. For the Eikonal equation, smoothed and with conductivity, the weak form with an arbitrary test function is given by:

$$\int_{\Omega} |\sigma \nabla u|^2 \psi dV = \int_{\Omega} \psi dV - \Gamma \int_{\Omega} \nabla u \nabla \psi dV + \Gamma \int_{\partial\Omega} g dA$$

where in the above weak form g is the value given by the Neumann boundary condition $\nabla u.\hat{\mathbf{n}}$. As discussed in [9] the weak form has the major advantage of now containing only first order terms, so the nonlinear system can be solved using only knowledge of the gradient of basis functions used in FEM.

The Jacobian is required because solutions to nonlinear PDEs need a linearisation of the problem. The following two maps were used, whose relevance may be demonstrated using the contraction mapping theorem [1]:

$$T(x) = x - sJ^{-1}F(x) \tag{15}$$

and

$$T(x) = x - rF(x) \tag{16}$$

It can be seen that solutions of the form $x = T(x)$ are desired, where F represents the PDE in weak form, and s and r are the relaxation terms for the Newton and Fixed point method respectively.

3.3. The hybridized algorithm

The solution has known singularites, corresponding to regions where the curvature of the wavefront becomes singular. This leads naturally to a hybrid solution, using both Jacobian-based and fixed point numerical methods. The slower Jacobian-based solution is run with relatively high smoothing term to guarantee convergence while the fixed point method is used to refine the solution with higher speed.

Since all matrix inversions can be reduced to $e + 1$ block matrices, where e is the dimension of the element, the time complexity of both numerical methods is linear with the number of elements in the mesh. As the problem is reducible to a matrix inversion or linear computation of the residual vector $F(x)$, its linear upper complexity bound is also geometry independant.

However, the hybrid approach gives rise to a large number of parameters, many of which need to dynamically compete. The parameters include the smoothing constant Γ, the relaxation constant in the fixed point solver s, the target level of smoothing, the amount by which to decine smoothing for each iteration, the intial condition, and where the relaxation term might be reset to a higher value. Pseudocode for the core of the algorithm can be seen in the supplementary material.

This competition comes naturally from the following costly inefficencies seen in a poorly parameterized run of the code:

1. If the smoothing drops too fast then the Newton Algorithm fails, so geometric decay with a slow rate is used with Newton, and a much faster decay is used with the fixed point algorithm.
2. The tolerance can be set too high for each smoothing iteration, with the result that a smoothed equation will be solved overly accurately.
3. Relaxation can be too small: if the relaxation constant is too low then there is a reciprocal cost against relaxation term, such that there is no benefit to small steps.
4. If the same relaxation term is too high, the solution can take longer to converge and require a further fall in this factor before convergence starts. Establishing this heuristic is the most significant speed challenge for the method in terms of numbers of iterations, especially since the appropriate value for the relaxation term is highly mesh dependant.
5. The method and amount by which to iteratively drop the smoothing term requires high levels of care, since dropping smoothing too fast can result in instant failure. Similarly, Newton's method immediately fails if applied on a too small solution. There are clear time cuts associated with being conservative with the smoothing update parameters.

The data used to support these points pertaining to the efficiency and behaviour of the Fixed point style algorithm can be found in Appendix A and figures in Appendix E: Figures E.6, E.9, E.8 and E.7. Further parametric analysis can be found in Appendix C and Appendix D.

4. Accounting for Anisotropy

Any successful model of heart tissue will successfully acount for the heavily fibrous character of the medium; much experimental and theoretical work has been done to realise and justify the physical properties of cardiac muscle [5]. The physical interpretation of this is to include a conductivitiy tensor to allow the wave to travel faster along our cardiac fibres than transverse to them. This is encapsulated in an analytically derivable general statement of the Eikonal equation, as presented in Equation 5. The wavespeed will depend on a conductivity tensor defined at each element, giving an anisotropic dispersion relation in our medium. The aim is to incorporate this into the solution methods already developed.

4.1. Fast Marching

The method applied for the Fast Marching algorithm was simply to transform each element to a basis in which the wave propogation is isotropic when considering each element. This is allowed because of the linearity of this transformation. This permitted the use of the relatively simple isotropic implementation of the algorithm to be applied beyond this step.

In a general coordinate basis the propagation of a wave travelling at a speed given by the square root of the conductivity traces out an ellispsoid $\mathbf{x}^T\boldsymbol{\sigma}^{-1}\mathbf{x} = t^2$ in a time t. Consequently, it is possible to change from $\mathbf{e}_i$ to a basis $\tilde{\mathbf{e}}_i = \mathbf{P}\mathbf{e}_i$ in which this ellipsoid is a unit sphere:

$$\mathbf{x}^T \boldsymbol{\sigma}^{-1} \mathbf{x} \Longrightarrow \tilde{\mathbf{x}}^T \mathbb{I}_3 \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T \tilde{\mathbf{x}}$$

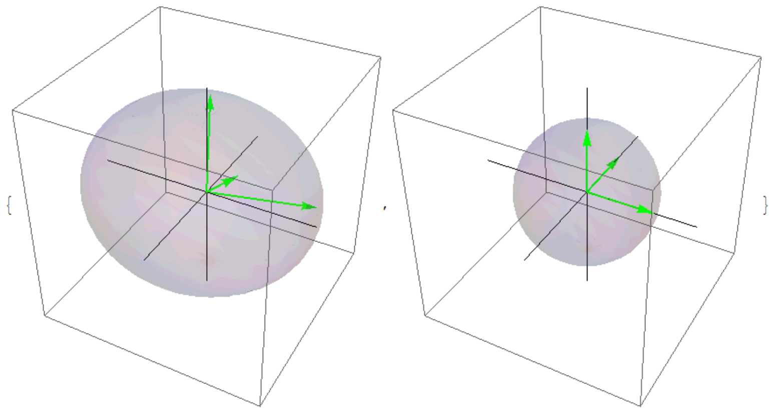


Figure 1: The left hand image shows a scalene spheroid with axis shown by green arrows, the linear transformation matrix $\mathbf{P}$ transforms this spheroid to the unit sphere shown on the right

The relatively simple algorithm developed for the isotropic case can be applied if the transformed basis is used when solving for wavefront-times on each new element. The basis transformation requires the linear transformation matrix $\tilde{\mathbf{x}} = \mathbf{P}\mathbf{x}$. The transformation is defined by the mapping of the axis of the ellipsoid to the usual orthonormal basis vectors:

$$\mathbf{P} \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ \sqrt{\sigma_1}\mathbf{e}_1 & \sqrt{\sigma_2}\mathbf{e}_2 & \sqrt{\sigma_3}\mathbf{e}_3 \\ \downarrow & \downarrow & \downarrow \end{pmatrix} = \mathbf{P}\boldsymbol{\sigma}_D^{1/2} \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ \downarrow & \downarrow & \downarrow \end{pmatrix} = \mathbb{I}$$

where $\boldsymbol{\sigma}_D$ is a diagonal matrix with entries given by the eigenvalues of $\boldsymbol{\sigma}$. Since the conductivity tensor is orthogonal in both bases, the matrix of normalised eigenvectors $\mathbf{V}$ is an orthonormal matrix whose inverse is its transpose. Thus $\mathbf{P}\boldsymbol{\sigma}_D^{1/2}\mathbf{V} = \mathbb{I}_3$ yields $\mathbf{P} = \mathbf{V}^T\boldsymbol{\sigma}_D^{-1/2}$. Consequently:

$$\mathbf{P} = \begin{pmatrix} \leftarrow & \frac{1}{\sqrt{\sigma_1}}\mathbf{v}_1 & \rightarrow \\ \leftarrow & \frac{1}{\sqrt{\sigma_2}}\mathbf{v}_2 & \rightarrow \\ \leftarrow & \frac{1}{\sqrt{\sigma_3}}\mathbf{v}_3 & \rightarrow \end{pmatrix}$$

By making this transformation on each element as the algorithm looks at it, the computation is slowed down slightly. The effect is not huge but it can be completely circumvented, because the information on the transformed nodes is a property of the environment and not of the solution. Consequently, it is possible to store the pre-transformed coordinates of the nodes in the mesh, and simply call them rather than the usual spatial coordinates when the algorithm looks at an element.

4.2. Dijkstra's algorithm

A similar approach was taken when including anisotropy into Dijkstra's algorithm: using an appropriate co–ordinate transformation, the problem can be solved in a basis in which the anisotropic conductivity is incorporated into the distance metric.

This approach becomes problematic, however, when dealing with heterogeneous distributions of conductivity tensors. It makes both physical and computational sense to associate conductivities with volume elements, but since Dijkstra contrains the signal to move along edges rather than geodesic paths, then there is an ambiguity as to which neighbouring element's conducitivty to use.

Two options are immediately available; either select a local element at random, making a smoothness assumption about the condutivity distribution, or take an average of all local elements and hence lose locality of information. The former was implemented, representing a fast but noisy method to generate anisotropic solutions.

4.3. Analysis of Anisotropy on the disk for FEM

With two different sets of parameters, the hybrid algorithm was applied to various conductivity tensors of the form $\sigma = (\lambda - 1)\, e_1^* \otimes e_1^* + I_d$, with d the dimension of the mesh as a topological manifold with a boundary.

5. Comparison of techniques

5.1. Errors

The chief advantage of the Finite Element Method is that the tolerance threshold may be set (the lowest accepted value of the residual $F(U)$): a valuable condition if the solver can approach the best solution that the mesh discretization will allow. Consequently, the user can demand specific levels of solution accuracy.

The FEM's ability to specifiy error tolerance stands in contrast to Dijkstra and FMM code, each of which iteratively accumulates errors inherent to the algorithm. Dijkstra's method generates fairly globally noisy data, whereas FMM exhibits greatest errors in regions of high wavefront curvature, as seen in Figure 2.

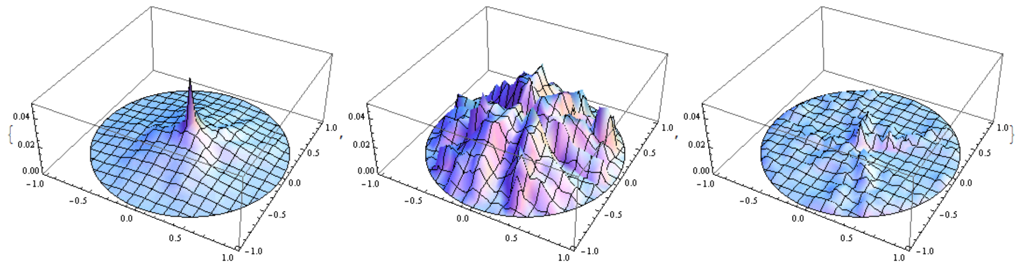


Figure 2: The errors on a wave propogated from the boundary of the circle into the centre in (from left to right) FMM, Dijkstra, FEM.

5.2. Timing

It is clear that since the FEM requires a large number of iterations to solve effectively then it will not be able to compete with FMM in terms of computation time; there in fact may be competition per iteration. It may remain appropriate in certain circumstances where precise control of error thresholds is necessary.

Due to the triviality of the calculations in a run of Dijkstra's algorithm, in comparison to the relatively complex update procedure of FMM, it is unsurprising that there was a significant difference in computation time. The timing difference is seen in Figure 5.2, in which Dijkstra's

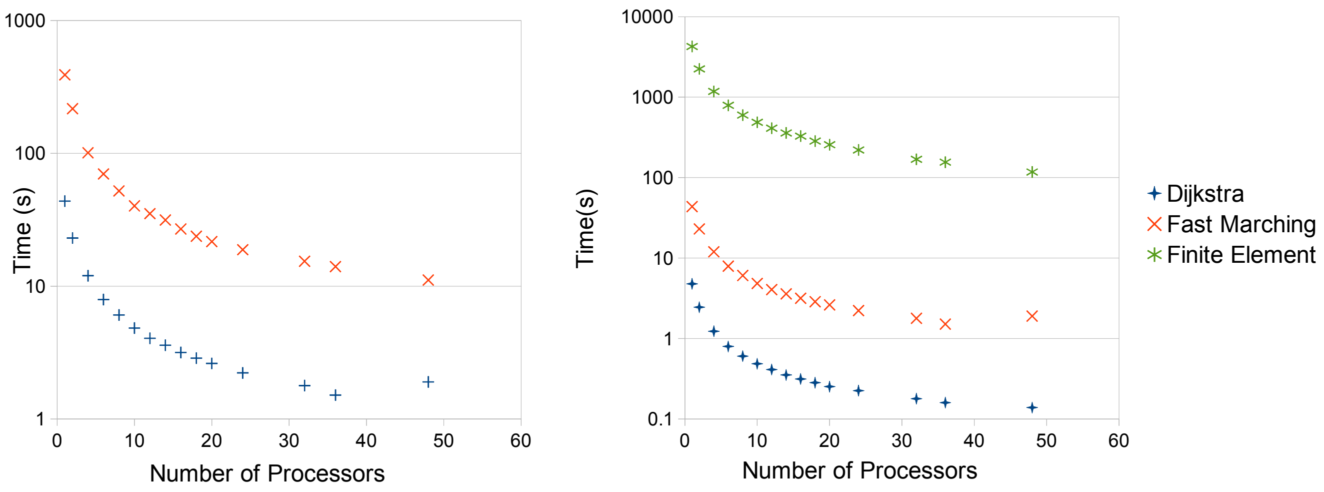


Figure 3: The solution times for the algorithms evaluated on the Oxford heart mesh with 4 million nodes (left) and the UCSD heart mesh with 63885 nodes, showing how solution time scales with number of processors averaged over 5 runs. The FEM algorithm was not applied to the Oxford heart mesh, since the size of the mesh made it prohibitively difficult to find convergent parameters.

algorithm is roughly an order of magnitude faster than FMM. It can similarly be seen that the FEM is a further order of magnitude slower than FMM, although the convergence time will be dependant on the error tolerance provided. Data can be found in Table Appendix A.

5.3. Parallel Scalability

The algorithms parallelised in a relatively straightforward way, with most challenges arising from data handling and output. Given the structural similarity of Dijkstra's algorithm and FMM, it is to be expected that they will speed up in similar ways for large clusters. Similarly, the FEM also scales in a successful fashion, designed as it is to be handled by large clusters of processors. This behaviour can be seen in figure 5.2.

6. Conclusions and Future Work

It is clear that for most practical applications, Fast Marching methods provide a relatively fast solution with predictable errors. Dijkstras algorithm provides a method for fast solutions, when accuracy is less of a priority. Finite element methods provide solutions with well-defined errors, although will remain prohibitively slow for most problems. It is possible that either improved PDE solvers or a hybridisation of Fast Marching solutions with the Finite Element techniques will make this a valuable computational approach in the future.

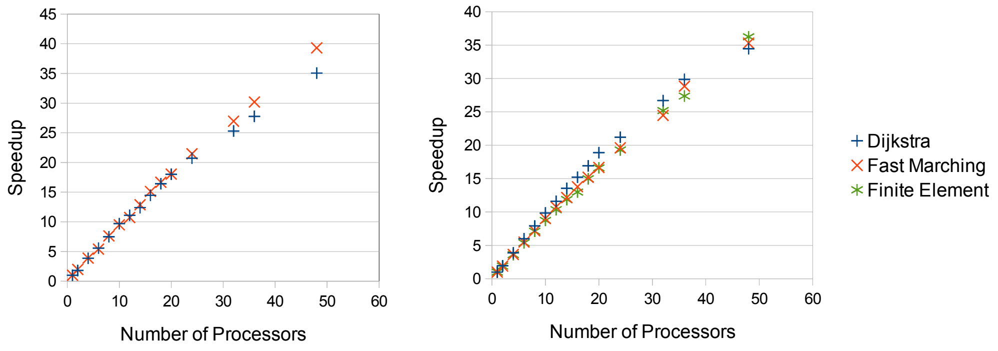


Figure 4: The speedup rates for the algorithms evaluated on the Oxford heart mesh with 4 million nodes (left) and the UCSD heart mesh with 63885 nodes, where "Speedup" is the time taken for one processor divided by the time taken for p processors, averaged over 5 runs and plotted against p. Sublinear scaling with p demonstrates efficient parallelisation.

Appendix A. Data

Fixed Point Tolerance	1.00×10^{-6}	1.00×10^{-6}	1.00×10^{-6}	3.00×10^{-6}	3.50×10^{-6}
Newton Tolerance	1.00×10^{-5}	1.00×10^{-5}	1.00×10^{-5}	1.00×10^{-5}	1.00×10^{-5}
Relaxation	-1.4	-1.2	-2	-1.4	-1.4
Initial Smoothing	0.5	0.2	0.2	0.1	0.1
Final Error	0.062	0.063	0.0626234	0.048	0.0503959
Residual	9.90×10^{-7}	9.90×10^{-7}	9.90×10^{-7}	2.90×10^{-6}	3.37×10^{-6}
Newton Timing	40.08	3.12	13.5	4.24	4.14
Newton Iterations	38	3	13	4	4
Fixed point Timing	39.78	30.56	76.32	11.02	10.49
Fixed point iterations	614	475	1166	163	155
Total time	79.86	33.68	89.82	15.26	14.63
Average Newton Iteration	1.054736842	1.04	1.038461538	1.06	1.035
Average Fixed point Iteration	0.064788274	0.064336842	0.065454545	0.067607362	0.067677419

Table A.1: Data produced by the FEM on a small irregular mesh given various starting parameters

Appendix B. Code Structure

Appendix B.1. FEM Hybrid

The algorithms described for the hybrid application of nonlinear iterative techniques has potentially broader application and so some care has been taken over the design of the structure of this part of the solver. The UML diagram attached shows this structure and illustates its application. There are remarks below:

Fixed Point Tolerance	3.50E-006	1.00E-007	1.00E-007	5.00E-007
Newton Tolerance	1.00E-005	1.00E-006	1.00E-006	1.00E-006
Relaxation	-1.4	-1.5	-1.5	-1.5
Initial Smoothing	0.1	0.1	0.5	0.08
Final Error	0.154	0.00907	0.0124	0.0091
Residual	3.40E-006	9.90E-008	9.90E-008	4.90E-007
Newton Timing	17.42	17.11	396	13.06
Newton Iterations	4	4	92	3
Fixed point Timing	62.24	399.74	473.63	101.49
Fixed point iterations	226	1524	1781	371
Total time	79.66	416.85	869.63	114.55
Average Newton Iteration	4.355	4.2775	4.304347826	4.353333333
Average Fixed point iteration	0.27539823	0.262296588	0.265934868	0.273557951

Table A.2: Data produced by the FEM on a 4096 element square mesh given various starting parameters

	Big disk			UCSD Heart			Oxford Heart	
Processors	Dijkstra	FMM	FEM	Dijkstra	FMM	FEM	Dijkstra	FMM
1	0.0066	0.039	16.8528	4.789	43.6776	4260.8	390.0506	3386.64
2	0.0034	0.0208	8.848	2.4542	23.0406	2244.5	216.8358	1710.5832
4	0.0022	0.0122	4.8586	1.2346	11.991	1177.8716	101.0048	870.986
6	0.004	0.0208	4.9706	0.7984	7.934	792.0466	69.9682	627.3468
8	0.0018	0.0076	2.7798	0.6054	6.0732	598.8184	52.1258	443.8192
10	0.003	0.015	3.6342	0.4864	4.8402	487.4424	40.1988	356.8224
12	0.003	0.0126	2.6942	0.4126	4.0576	411.8136	35.1748	315.9332
14	0.003	0.0128	2.0286	0.3536	3.5928	359.2468	31.4698	263.2326
16	0.002	0.0092	1.6966	0.3148	3.1696	328.2122	26.9514	224.0288
18	0.003	0.0118	1.815	0.2832	2.8726	284.6292	23.7686	203.3602
20	0.0028	0.0102	1.7944	0.2534	2.6204	256.4826	21.6642	187.7552
24	0.004	0.0116	1.7258	0.2258	2.2254	220.7676	18.82	157.6692
32	0.0038	0.009	1.3044	0.1794	1.7848	169.1536	15.4142	125.5864
36	0.0058	0.0108	1.3136	0.1604	1.5144	155.6778	14.0478	112.1416
48	0.0066	0.012	1.6188	0.139	1.9032	117.5576	11.1214	86.147

Table A.3: Timing data for the three algorithms in seconds, taken on a 2-D Big Disk, the UCSD heart mesh and the Oxford heart mesh, averaged over 5 repeats with the boundary conditions of all boundary nodes starting excited.

```
procedure SOLVE
    POPULATESOLVERS
    return RUNSOLVER(0)
end procedure
procedure RUNSOLVER($n$)
    $fail := 0$
    while $s >$ target or $res >$ TOL[$n$] do
        if $res <$ TOL[$n$] then
            CUTSMOOTHING
        end if
        $fail =$ SOLVERMETHOD(n)
        if fail then
            break
        end if
    end while
    n := GETNEWLEVEL($n$)
    return RUNSOLVER($n$)
end procedure
```

1. *AbstractHybridSolver* - this class forms the engine, into which all the methods are atteched, it is in this source file that the template pattern algorithm described above is implemented, since when subclassed the heuristics and parameters for determining the behaviour of the solver are implemented.
2. *AbstractHybridMethod* - the class to be inherited from to use this solver to in other contexts
3. *EikonalHybridMethod* - the actual implementation, here one will find the implemention of functions such as '*cutSmoothingTerm*' or '*getLevel*', to dictate the solver's behaviour.
4. *EikonalSolver* - hard coded residual and jacobian finding functions, it is from these that the FEM problem is assembled and dealt with by the solver, either to solve a linearized equation or to pass directly to a fixed point style method.

Appendix B.2. Dijkstra

```
1 $S \leftarrow \emptyset$
2 $Q \leftarrow \emptyset$
3 for $v \in$ Sources
4       t[$v$] $\leftarrow 0$
5       $Q \leftarrow Q \cup \{v\}$
6 while $Q \neq \emptyset$
7       do $u \leftarrow$ Extract Minimum(Q)
8           $S \leftarrow S \cup \{u\}$
9           for each vertex $v \in$ Adjecent to($u$)
10              if t[$v$] $>$ t[$u$] $+ \tau[u,v]$
11                  t[$v$] $\leftarrow$ t[$u$] $+ \tau[u,v]$
12                  $Q \leftarrow Q \cup \{v\}$
```

Appendix B.3. Fast Marching Method

```
1 $S \leftarrow \emptyset$
2 $Q \leftarrow \emptyset$
3 for $v \in$ Sources
4       t[$v$] $\leftarrow 0$
5       $Q \leftarrow Q \cup \{v\}$
6 while $Q \neq \emptyset$
7       do $u \leftarrow$ Extract Minimum($Q$)
8           $S \leftarrow S \cup \{u\}$
9           for each element $E : u \in E$
10              for each node $v \in E \setminus S$
11                  while no acceptable estimates
12                      for each $L \subset E \setminus S, |S| = n$
13                          try $n-$dimensional fit($L$)
14                      $n \leftarrow n - 1$
15                  if t[$v$] $>$ best acceptable estimate
15                      t[$v$] $\leftarrow$ best acceptable estimate
16                      $Q \leftarrow Q \cup \{v\}$
```

Appendix C. Smoothing Term

So far the study of the smoothing term has been restricted to analysis on the disk model, where the Eikonal equation: $|\sigma\nabla\phi|^2 = 1 - \Gamma\nabla^2\phi$ becomes:

$$y'^2 + \Gamma\left(y'' + \frac{1}{x}y'\right) = 1 \tag{C.1}$$

One can more easily investigate the possible role of Γ: in Figures E.11 and E.12, we see that for a constant smoothing term with boundary conditions forced at $r = 1$, there is a pronouned spike at the origin to account for the effect of the smoothing on the gradient, although this declines as Γ falls. However, if we allow smoothing to vary, it is beneficial to smooth less in the centre. This is somewhat frustrating since it is the centre that requires higher smoothing in order for weaker methods to converge to a solution at all.

Appendix D. Relaxation Factor

Here variation in the relaxation factor is discussed: the s present in: $U_{k+1} = U_k + sF_d(U_k)$ on a discretized map F_d obtained by finite element approximation from a PDE: $F(u) = 0$. The investigation is performed from data in Table A.1 and Figures E.6, E.9, E.8 and E.7. Note that in the graphs the two lines represent different meshes, blue relating to a 984 element disk geometry and the orange a 128 element square one.

Again considering the disk, we present the data in Table A.1 based on the timing and number of iterations. Figures E.6 and E.7 demonstrate that for relatively high tolerance results there is a high correlation between tolerance and global error. Conversely, for the smaller values increased tolerance can actually *lower* the global error from the analytic solution. This is quite a disturbing phenomenon; it shows that high accuracy solutions can be obtained in fewer iterations than expected but also confirms that tolerance is not a tight bound on the error from an analytic solution.

It should be noted that the effect of the initial smoothing term can be overstated: while it appears that lowering this term greatly lowers the time to achieve a solution, this is solely due to the failure of the Newton solver to converge after a few steps and passing to the much faster fixed point iterator. Since faster iterators may bring these two in competition this will remove the effect of this term.

A potential extension of this work is to give these parameters and the data collected for a variety of meshes. To indicate mesh dependancy, consider the equivalent Table A.2 for the 4096 element square mesh.

Appendix E. Figures

References

[1] Robert M Brooks and Klaus Schmitt, *The contraction mapping principle and some applications*, Department of Mathematics, Texas State University, 2009.

[2] P Colli Franzone, LF Pavarino, and B Taccardi, *Simulating patterns of excitation, repolarization and action potential duration with cardiac bidomain and monodomain models*, Mathematical biosciences **197** (2005), no. 1, 35–66.

[3] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik **1** (1959), 269–271.

[4] Joe Pitt-Francis et al., *Chaste: A test-driven approach to software development for biological modelling*, Computer Physics Communications **180** (2009), 24522471.

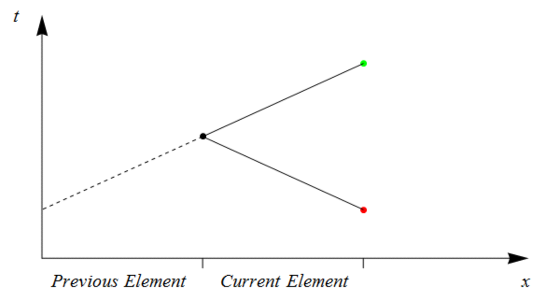


Figure E.5: In the simple case of 1D the two possible solutions found that satisfy the gradient condition are increasing or decreasing. Causality requires that we discard the decreasing solution. The correct solution is indicated by the green dot

Figure E.6: Residual against relaxation factor

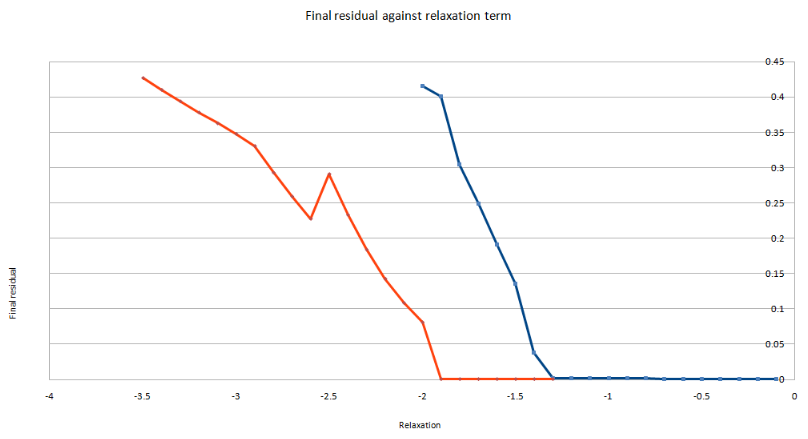


Figure E.7: Residual against relaxation factor on smaller values

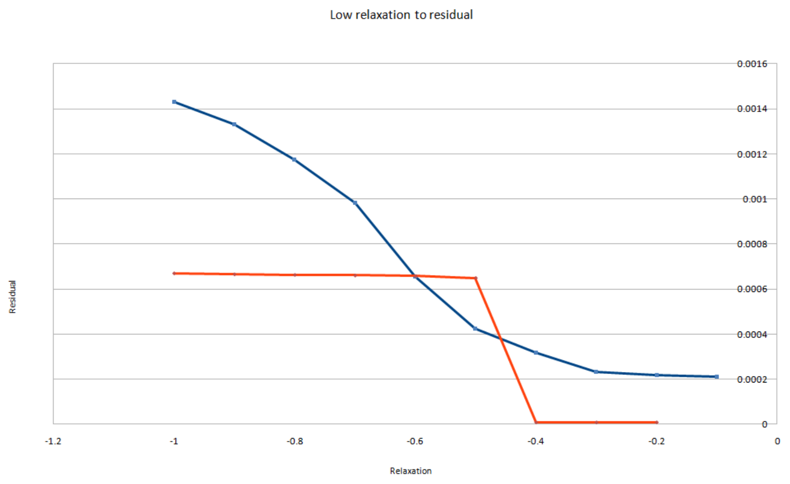


Figure E.8: Determining the growth rate of iterations to relaxation factor

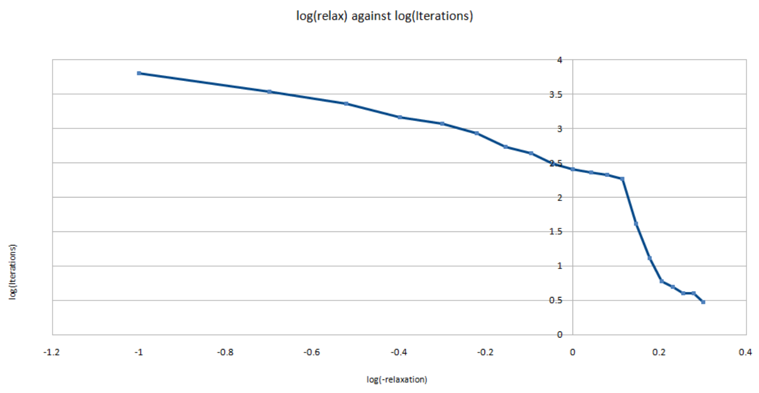
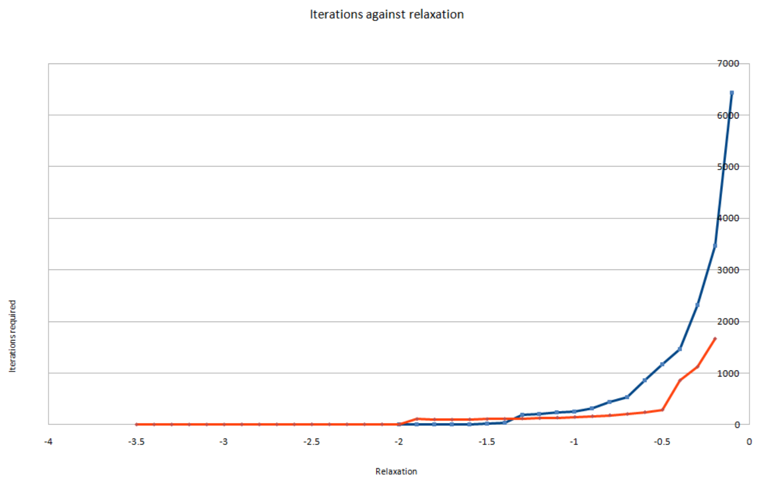


Figure E.9: Iterations taken over different relaxation terms (Fixed point method only)



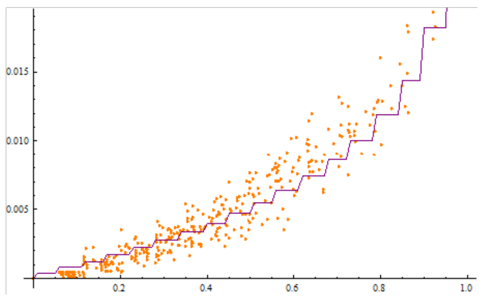


Figure E.10: A plot of the deviation of the fast marching method on the 2D disk against distance from the source nodes. The line is a plot of the approximate error as given by equation (13).

Figure E.11: 1D case with gradient lowered and

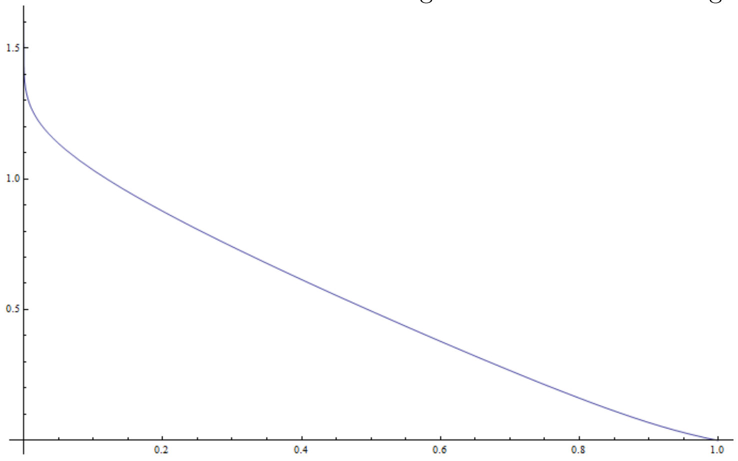
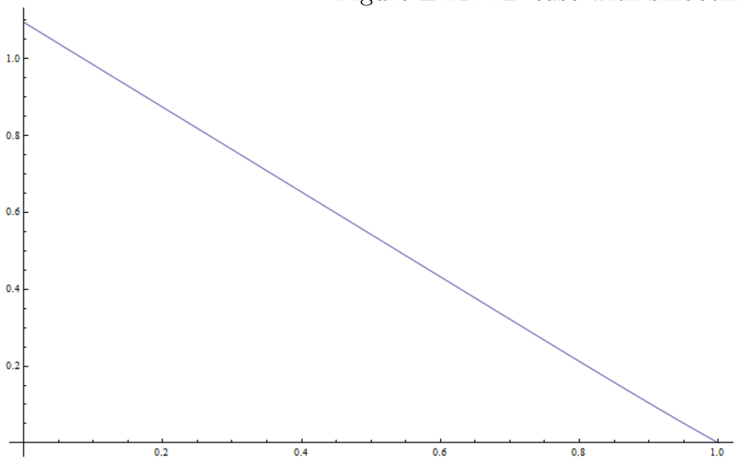


Figure E.12: 1D case with smoothing factor $\Gamma = \epsilon r, \epsilon = 0.2$



[5] P.C. Franzone and L. Guerri, *Spreading of excitation in 3-d models of the anisotropic cardiac tissue*, Mathematical Biosciences **113** (1993), 145–209.

[6] AL Hodgkin and AF Huxley, *Propagation of electrical signals along giant nerve fibres*, Proceedings of the Royal Society of London. Series B, Biological Sciences **140** (1952), no. 899, 177–183.

[7] R. Kimmel and J.A.Sethian, *Computing geodesic paths on manifolds*, Proc. Natl. Acad. Sci. USA **95** (1997), no. 15, 8431–8435.

[8] AL Muler and VS Markin, *Electrical properties of anisotropic neuromuscular syncytia.*, Biofizika **22** (1977).

[9] Pras Pathmanathan, 2011. Numerical Methods and Object Oriented Design.

[10] P.G. Tucker, *Differential equation-based wall distance computation for des and rans*, Journal of Computational Physics **190** (2003), 229–248.

[11] M. Wallman, N.P. Smith, and B. Rodriguez, *A comparative study of graph-based, eikonal, and monodomain simulations for the estimation of cardiac activation times*, Biomedical Engineering, IEEE Transactions on **59** (2012), no. 6, 1739–1748.

[12] Hao Xia and Paul G. Tucker, *Finite volume distance field and its application to medial axis transforms*, Int. J. Numer. Meth. Engng (2009).