Solving the Material Interface Reconstruction Problem using Genetic Programming

Jeremy Meredith
Lawrence Livermore National Laboratory
PO Box 808 / MS L-98
Livermore, CA 94550
meredith6@llnl.gov
925-422-1197

Abstract: This paper develops enhanced material interface reconstruction techniques using genetic programming. Material interface reconstruction is the attempt to recreate high resolution material placement in computational meshes given only the percentages of materials at the coarser level of cells in the mesh. Previous techniques have been designed to create either accurate percentages or smooth interfaces, but not both. This paper suggests evolved modifications to known techniques as well as novel methods to combine the two goals of material interface reconstruction.

1 Introduction and Overview

Material interface reconstruction is a technique in scientific computation which attempts to recreate the shapes and placement of materials and their boundaries in cells of a computational mesh. This reconstruction is based on only the percentages of various materials in a cell of interest and in surrounding cells. Previous approaches have focused on either accurately reconstructing the percentages of materials or on reconstructing smooth, realistic looking interfaces, and there has been little success combining the two ideals. This paper examines modifications to the known techniques using genetic programming (GP) to enhance accuracy of the better looking techniques and the believability of the accurate techniques.

Section 2 explains more about material interface reconstruction and describes previous work in this field. Section 3 describes the approaches used for solving the problem. Section 4 describes methods common to all attempted solution approaches. Section 5 details the method, results, and discussion for the first approach, and Section 6 details the method, results, and discussion for the second approach. Section 7 concludes our study and Section 8 suggests directions for future work.

2 Background

Scientific simulation codes have been simulating physics on a wide variety of problems for decades. These types of simulations divide the world using a computational mesh with thousands to millions of “cells”, much like the grid on a sheet of graph paper. One thing common to many of these simulations is that every point in space must be associated with a certain type of material, be it water, air, or metal. Finite element (FE) codes, however, in some ways have an easier problem than computational fluid dynamics (CFD) codes, because every cell in an FE mesh is usually a single material. CFD problems, on the other hand, often involve hydrodynamic calculations where the materials can move faster than a mesh can keep up, for example in fluid instability simulations such as those of Rayleigh-Taylor turbulence. In this case, the mesh could tangle itself into an unsolvable problem without the ability to move materials independent of the mesh. The far extreme of this is a pure rectilinear (Eulerian) mesh, where the mesh stays constant and materials move freely through the mesh. Usually the resolution of the computational mesh is not high enough to accurately represent the material structure of the underlying physical world, so instances often exist in CFD where a cell is *mixed*: it contains two or more materials, each in a different region of the cell.

Since the materials can now necessarily be of greater resolution than the mesh proper, the question of how to keep track of this material information becomes a problem. The typical method of storage has been to store in each cell just the material *volume fractions* (VFs), or the percentage of the cell filled with each material. However, since the shape of the interface is then not stored, the difficult part of visualizing results from these simulations involves determining what the materials looked like based purely on what volume fractions are stored for each material in a cell. Methods of reconstruction and tracking of these mixed material interfaces have been researched since the

1960s. They are important for visualization, but they are also important for physical simulations themselves since an inaccurate reconstruction can give inaccurate results.

As an example, let us use a simple 3x3 cell mesh where exactly the bottom half of the mesh is water and the top half is air. This means the top three cells are completely air, the bottom three cells are completely water, and the middle three cells are half water and half air; figure 1 shows the volume fractions of air and water in each cell. These three middle cells are mixed, and it is the reconstruction method which must determine which portion of these cells is air and which is water. For this paper all fitness cases are 3x3 grids where the algorithm must determine the interface in only the center cell, and the eight surrounding cell volume fractions are used only for contextual information. In this simple example, it is clear that the center cell is "correctly" reconstructed by assigning water to the bottom half of the cell and air to the top half of the cell. Other examples are not quite as easy to solve; for example, in this case correctness should be defined not only in terms of accurately recreating an interface which covers *exactly* half the cell of interest, but that it put the water in the bottom half.

0.0	0.0	0.0
0.5	0.5	0.5
1.0	1.0	1.0

1.0	1.0	1.0
0.5	0.5	0.5
0.0	0.0	0.0

Figure 1: volume fractions for water (left) and air (right)

One of the first methods for material interface reconstruction (MIR) is a method which uses tracking particles to define the interface (Amsden, 1966). Noh and Woodward (1976) created the simple line interface calculation (SLIC) which determines the material interface using these volume fractions. It is a piecewise constant method which aligns the material interface with one of the major coordinate axes. Also originating at this time were similar piecewise constant/stair-stepped methods such as the volume-of-fluid (VOF) method (Nichols and Hirt, 1975).

An improvement to these methods came with piecewise linear interface calculations (PLIC) such as the one from Youngs (1992). These methods involve two steps for the reconstruction. First, calculate the slope of the interface using a window around the cell of interest, often using the gradient of volume fractions. Second, calculate the intercept of that interface line such that it intersects the exact volume fraction in the cell of interest. The process is repeated for each material. Using the above example, the PLIC method would first decide that the water-air boundary in the cell of interest should be a horizontal line with water below and air above, and it would then decide that the boundary should be placed exactly 50% between the bottom and top of the cell.

However, these methods are all very concerned not just with the interface reconstruction, but more so with the interface tracking using this interface. In other words, these algorithms are not designed to look correct, but to perform correctly *within* a CFD or hydrodynamic code. This makes visualization of material interfaces using these methods unappealing. Their primary drawback is that the interface is never designed to be continuous across cells; visualization would clearly show the stair stepping and discontinuities inherent in these algorithms (see figure 2 for an example). In fact, it is impossible using only a single linear interface to obtain an interface which is both accurate and continuous across cell boundaries, because it can only approximate what should in many cases be a curved line.

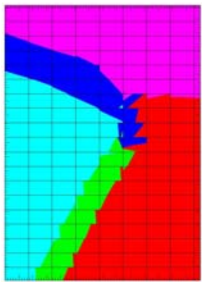
Figure 2: PLIC method

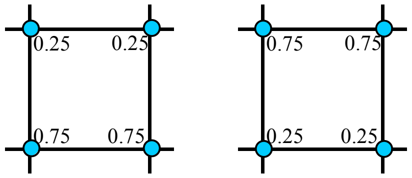

Figure 3: interpolated values for the cell of interest for water (left) and air (right)

The MeshTV scientific visualization code of Lawrence Livermore National Laboratory uses a different method to reconstruct and visualize mixed material meshes. The basic premise of this algorithm is the following. First, create a linearly interpolated variable over the mesh using the volume fractions for each material. This is done by averaging the volume fractions of each material in all four cells surrounding a mesh node to that node. See figure 3 for an example in the central cell from the water/air mesh example from above. Second, using an interpolation function such as bilinear interpolation, evaluate the function for each material at every point within the cell. Where the value for water is greater, it is determined that the point contains water, and where air is greater, it is determined that the point contains air. For this example, the values for water are greater than 0.5 underneath the horizontal center line, and the values for air are below 0.5 underneath that same line. Thus the bottom half of the cell becomes water and the top half becomes air.

This approach has one distinct advantage, which is that it is guaranteed to generate continuous interfaces from one mixed cell to the next (see figure 4 for an example); the values obtained by averaging the four cell values to the node are the same no matter which

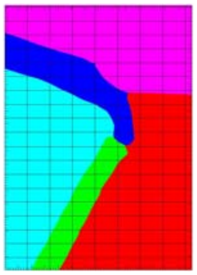
Figure 4: MeshTV method

cell of the four we are considering. This means the interpolant generates the same values at any edge no matter which of the two cells along the edge we are considering. In addition, it generates boundaries which look subjectively “good” – the water in a mixed cell will be near the water in neighboring cells, and the air in a mixed cell will be near the air in neighboring cells. However, unlike the historical techniques, it makes no guarantees about volume fractions. In the above example it happened to create a boundary which gave exactly half the area to water and half to air, and the more water/air that is supposed to be in the cell, the more it will generate. However, this technique makes no effort to be exactly accurate and will in general not be so; as mentioned above, with so few degrees of freedom it may be impossible to ensure accuracy if it ensures continuity.

There is one enhancement to this technique worthy of note: the central cell of interest may be subdivided into a 2x2 arrangement, requiring nine values instead of just four for a single cell (see the circles in figure 5). The values at the corners are determined as above, the values at the edges are determined by averaging the volume fractions from two neighboring cells, and the value at the center of the cell is simply the original cell volume fraction. This subdivision empirically works well to allow extra curvature when needed within a cell to improve accuracy, although still no accuracy guarantees are made.

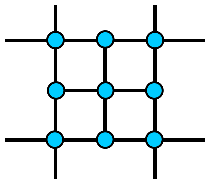
Figure 5: positions for values when subdividing

3 Statement of the Problem

This paper describes in detail two different approaches to improving upon existing techniques. The first takes the MeshTV algorithm with subdivision and attempts to change the method where values are assigned to the nodes in an attempt to improve the accuracy while maintaining the continuous boundaries and the subjective visual correctness.

The second approach takes the PLIC algorithm, which creates linear boundaries within a cell that ensure accurate volume fractions, and extends it to use conic sections instead of purely linear interfaces. It is also more general in that it attempts to get the accurate volume fractions using the same set of functions as the ones which determine the shape of the interface.

4 Common Methods

This study used the LIL-GP genetic programming software from Michigan State University.

Both approaches indirectly create a function for each material which is then evaluated at every point in the cell. Where the value of this function is greater when given the first material’s volume fractions as input, this determines that the first material exists at that point. Where the value of this function is greater for the second material, it determines that the second material exists at that point.

The difference between the problem setups is that the first approach creates function trees which create values used for an interpolation function, while the second approach creates function trees which create constants to a quadric function.

Fitness cases are a set of 3x3 cell volume fraction information for two materials. Some of these fitness cases are used to evaluate only the accuracy of the solution in terms of the generated volume fraction. Others are used to determine not only total amount of each material, but its correct placement within the cell as well.

To evaluate fitness for accuracy of volume fraction, the total percentage of points within the cell of interest of each material determines its experimental volume fraction. This is compared with the original target volume fraction listed for that cell to determine the error.

To evaluate fitness for correct material placement within the cell, every point in the cell is predetermined to be one of the two materials, and error is calculated as the percentage of points which have the wrong experimental material compared to the target material for that point.

5 Extending the MeshTV Algorithm

5.1 Methods

Table 1 summarizes the preparatory steps for the extended MeshTV algorithm with 2x2 subdivision. This algorithm produces nine values used for control points of an interpolation function.

There are three trees for each individual. One is the NODE tree. This has as terminals functions of all four cells surrounding the point at each corner of the cell. It is evaluated four times, once at each corner, with the four appropriate cells used as the input each time. The second is the CELL tree. It has as terminals functions of all nine cells in the problem, and it is evaluated once for the point in the center of the cell. The third is the EDGE tree. It has as terminals functions of the two or six cells surrounding each edge of the cell. It is evaluated four times, once at the point in the center of each edge, with the two and six appropriate cells used as the input each time.

The if0any functions take two input subtrees, and evaluate the first if any of the input cells is 0 and the second otherwise. The if1any functions are similar but check for a 1 in any of the input cells.

For example, the Nif0any function, when called on the upper-right-hand node evaluates the first input if any of the upper-right-hand four cells' VFs are 0 and the second input otherwise. The Eminall6 terminal, when called on the node at the center of the right-hand edge, returns the minimum value of all 6 cell VFs on the right-hand side of the 9 cell window.

Fitness was evaluated as the root-mean-square average of all fitness cases' error in volume fraction.

A population of 2000 was used. With an initial depth limit of three, the search space is roughly 10 million per tree initially, and more when individuals may expand their depth. All structures of the search space are likely initially, and all structures of the expanded search space may be created.

Table 1: Tableau for the Enhanced MeshTV Algorithm

Objective:	Find a set of functions in symbolic form that take a set of volume fractions for a cell of interest and the eight surrounding cells and return a set of nine values for the material that, when bilinearly interpolated, is greater than the corresponding interpolant for other materials in the cell for points covering the exact volume fraction of that material for the cell of interest.
Terminal set:	All trees: ERC (50% are –1.0 to 1.0, 50% are 2^n with n = -2..+2) Tree NODE: Nminall, Nmaxall, Nsumall, Navgall, Nmaxdiffall Tree CELL: Cminall, Cmaxall, Csumall, Cavgall, Cmaxdiffall, Ccentervf Tree EDGE: Eminall2, Emaxall2, Esumall2, Eavgall2, Emaxdiffall2, Eminall6, Emaxall6, Esumall6, Eavgall6, Emaxdiffall6
Function set:	All trees: +, -, / (safe), *, ^ Tree NODE: Nif0any, Nif1any Tree CELL: Cif0any, Cif1any Tree EDGE: Eif0any2, Eif1any2, Eif0any6, Eif1any6
Fitness cases:	15 arrangements of a set of 3x3 volume fractions (VFs) for two materials, used to measure volume fraction accuracy. See figure 6
Fitness:	Hits: the number of fitness cases where the reconstructed VF is within 5% of the target Standardized and Raw Fitness: the RMS average of the error across all fitness cases
Wrapper:	None
Parameters:	M=2000, G=1000; Operations: crossover=90% reproduction=10%
Termination::	The first function set that scores 15 hits is selected as the best-of-run and breeding halts.

5.2 Results

The best-of-generation individual from generation 0 scored 5 hits with a fitness of 0.1001.

```
NODETREE:
  (/ (+ nsumall nmaxall) (+ navgall nmaxdiffall))
CELLTREE:
  ccentervf
EDGETREE:
  eminall2
```

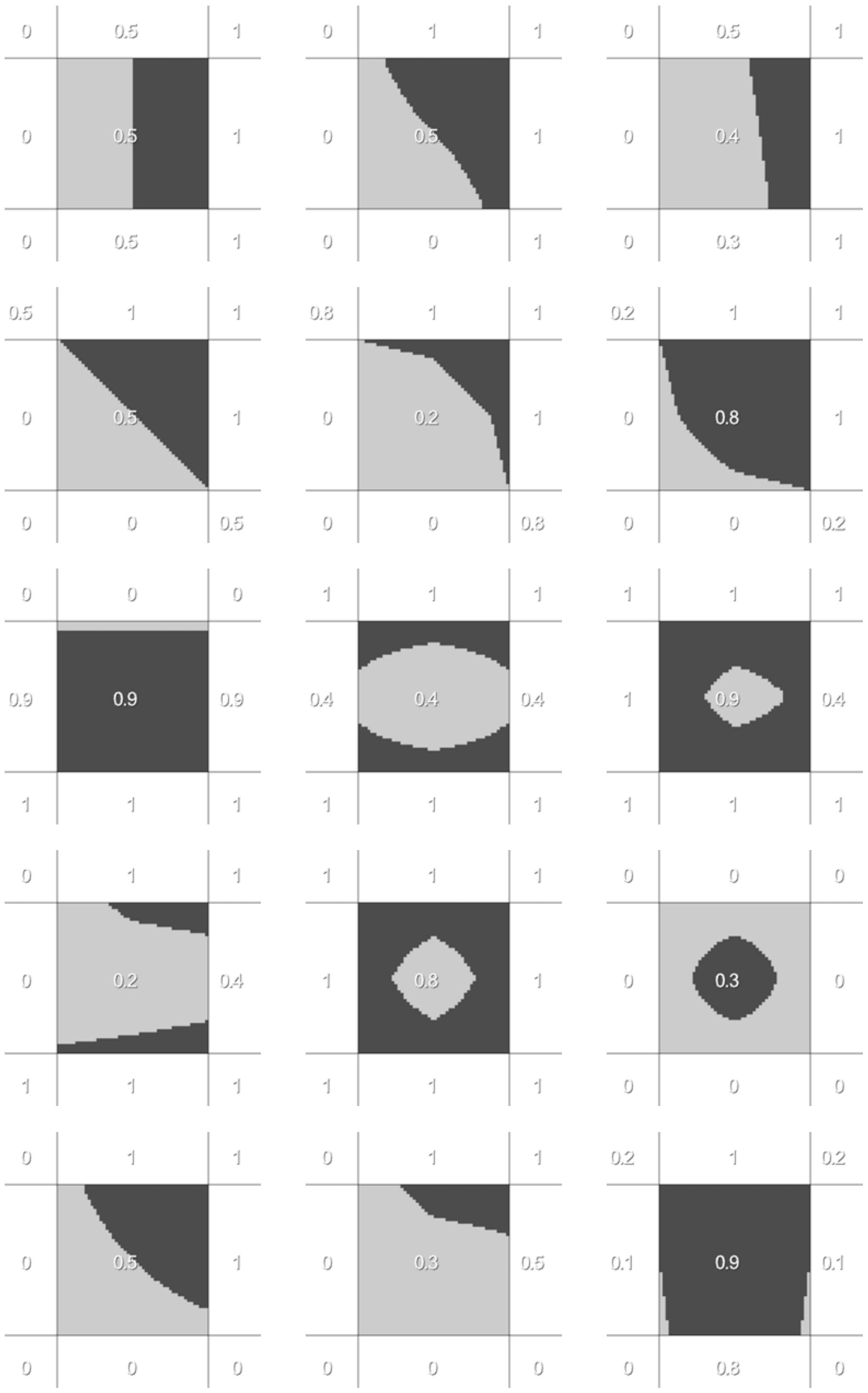

Figure 6: 15 fitness cases and results for best of run individual for the extended MeshTV algorithm (given volume fractions are for the dark material)

The best-of-run individual was from generation 30 scoring 15 hits with a fitness of 0.0258.

```
NODETREE:
  navgall
CELLTREE:
  (/ (- (+ (cif0any (- 2.00000 ccentervf)
                    (- (+ ccentervf cminall) cminall)) ccentervf) cminall)
     (+ (/ cminall 0.25000) cmaxall))
EDGETREE:
  eavgall2
```

5.3 Discussion

Using the terminals such as Navgall or Eminall2 instead of individual cell volume fractions ensures that the same value is used for each interpolation point no matter which cell of interest is being considered. This by itself guarantees continuity of the resulting interface across mixed cells.

Note the best individual from generation 0. The CELL tree happened to correspond exactly with the naïve choice from the original MeshTV algorithm. The EDGE tree chose the minimum of the two edge values instead of the average. The NODE tree was something more complex, involving the sum, average, maximum, and maximum difference of the four cells around each node.

Now examine the best individual of the run, from generation 30. The NODE and EDGE trees *both* corresponded exactly with the naïve choice from the original MeshTV algorithm. The CELL tree was something more complex this time. Because this modification is so simple, it could be dropped straight in to the production MeshTV code as a replacement for the function used to find values for the point at the center of the cell and instantly improve the accuracy of an algorithm which already produces visually appealing results. In addition, since the genetic program was prepared in such a way as to maintain continuity, no quality should be lost as a result of this replacement.

Plots of the best-of-run individual for the 15 fitness cases are seen in figure 6 on the previous page. In these plots, the numbers are the volume fractions of the dark material. It is apparent in this figure that this individual results in an approximately correct volume fraction in the central cell of interest, and that it places the dark material in the cell of interest in a realistic placement relative to the larger amounts of dark material in surrounding cells.

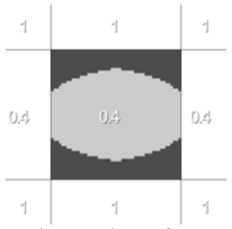

Figure 7: best-of-run individual's results on fitness case number 8

Success obtained here left time for further study, so the decision was made to extend the project with extra experimentation. One extension involved an attempt to improve the realism of the results further. For the motivation, take the best of run individual's result for fitness case 8 as seen in figure 7. Note that although there is a constant value horizontally across all volume fractions, the resultant shape will have an undulating interface. It will be continuous across mixed zones, but it will not be a straight line.

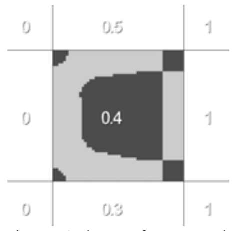

Figure 8: best-of-run result from 4x4 subdivision on fitness case number 3

Since one can argue that this should be a straight line interface (or two of them, more specifically), it was decided that more degrees of freedom were needed. The next experiments changed the subdivision from 2x2 to 3x3, then 4x4. However, it turns out that this was too much freedom with not enough constraints. See figure 8 for an example – it was free to choose so many parameters now that it lost the need for any semblance of realistic looking interfaces. Attempts to evaluate the "smoothness" of the interface and account for it in the fitness value proved too difficult to balance the smoothness and accuracy.

The decision was made to try a different, far more ambitious approach altogether – one with even more freedom and at the same time more constraints. This was the choice to attempt to extend the PLIC algorithm to deal with quadric interfaces, and at the same time add fitness cases specifically designed to ensure that individuals must "look good" before they can breed.

6 Extending the PLIC Algorithm

6.1 Methods

Table 2 summarizes the preparatory steps for the PLIC algorithm extended to quadric instead of linear interfaces – PQIC if you will. This algorithm will construct a function of the form v=Ax^2+By^2+Cx+Dy+Exy+F.

There are four result producing trees for each individual. One results in the coefficients A and B of the squared terms (x^2 and y^2), with B evaluated by flipping the x/y inputs. One results in the coefficients C and D of the linear terms (x and y), with D evaluated by flipping the x/y inputs. One results in the coefficient E for the xy term, and one results in the constant coefficient F. There are also five automatically defined function trees, with varying amounts of arguments, that can be used in the RPB trees.

The cellXY terminals are the volume fractions of the nine cells, where X and Y are integers from 0 to 2. The vd1 and vd2 terminals are the centered difference approximations to the vertical first and second derivatives. For example, vd1 is (cell21–cell01), and vd2 is (cell21 – 2*cell11 + cell01)/2. The hd1 and hd2 terminals are similar but for horizontal derivatives, and d1d1,d1d2,d2d1,d2d2 are similar but for the two sets of diagonal derivatives. These are likely to be useful shortcuts based on domain-specific knowledge.

The min2/3, max2/3, and avg2/3/4 functions are min, max, and averages of multiple input trees. The if< and if> functions take four inputs and evaluate either the third or fourth argument depending on the comparison between the first two. The if1, if0, if<1 and if>0 take three inputs and evaluate either the second or third depending on the comparison of the first input with 0 or 1. The ifmix function is similar, but the second input is evaluated if the first is between 0 and 1 exclusive, and the third input is evaluated otherwise. These are also likely to be useful.

Fitness was evaluated as the RMS average of the errors across all 29 cases. The first 16, however, were weighted to contain 90% of the error metric so that individuals were coerced to look good first, then become more accurate later. These fitness cases were simple cases with a known accurate solution, just as the air/water example from Section 2.

For this approach, a much larger population of 50,000 individuals was used because of the immensity of the search space. (Assuming only a few choices of ERC and an initial depth limit of three, this is still a search space of roughly 3 billion. With the tree allowed to increase its depth, this number certainly grows.) All structures of the search space are likely initially, and all structures of the expanded search space may be created.

Table 2: Tableau for the PQIC Algorithm

Objective:	Find a function in symbolic form that take a set of volume fractions for a material in a cell of interest and the eight surrounding cells, as well as an x/y pair (x and y in the range –0.5 to +0.5) and returns a value that, when evaluated at all x/y points in the cell, is greater than the corresponding function for other materials in the cell for points covering either the correct region or just the correct volume fraction of that material for the cell of interest.
Terminal set:	All trees and ADFs: ERC (50% are –1.0 to 1.0, 50% are 2^n with n = -2..+2), cell00 through cell22, vd1,vd2, hd1,hd2, d1d1,d1d2, d2d1,d2d2
Function set:	All trees: +,-,/ (safe),*, min2, max2, avg2, avg3, avg4, abs, sqr, if<, if>, if>0, if<1, ifmix, if0, if1 RPB trees: ADF0 through ADF5 ADF1: ADF0 ADF2: ADF0 and ARG0 ADF3: ADF0, ADF1, ADF2 and ARG0 ADF4: ADF0, ADF1, ADF2, ADF3 and ARG0, ARG1 ADF5: ADF0, ADF1, ADF2, ADF3, ADF4
Fitness cases:	16 arrangements of a set of 3x3 volume fractions (VFs) for two materials used to measure physical accuracy, and 13 arrangements of a set of 3x3 volume fractions (VFs) for two materials used to measure volume fraction accuracy
Fitness:	Hits: the number of fitness cases where the reconstructed VF is within 5% of the target Standardized and Raw Fitness: the RMS average of the error across all fitness cases, with a 90% weighting on the first 16
Wrapper:	None
Parameters:	M=50000, G=10000; Operations: crossover=85% reproduction=10% mutation 5%
Termination:	The first function set that scores 29 hits is selected as the best-of-run and breeding halts.

6.2 Results

The best-of-run individual was from generation 221, with 23 hits and a fitness of 0.19658.

```
[[x^2 || y^2]]:
 (/ hd2 (max2 (+ (- (- (sqrt (max2 (sqrt (- (- (/ d1d2 (- (sqrt (avg2 (min2 hd2 ADF0) (if>0 d1d2 ADF5 d1d1))) (avg2 (avg2 d1d2
(avg2 (min2 hd2 ADF0) (min2 cell21 d2d2))) (avg3 (ADF2 cell10) (- (max2 (/ d2d1 ADF5) (sqr ADF5)) d2d2) (- (sqrt (sqrt (if0 (+ (/
d2d1 ADF5) (max3 d1d2 hd2 (- cell20 cell00))) cell11 ADF1))) (min2 cell21 d2d2)))))) (min2 cell21 d2d2)) d2d2)) cell12)) (min2
cell21 d2d2)) (sqrt ADF1)) d2d2) (max2 (/ d2d1 ADF5) (sqr ADF5))))
[[xy]]:
 (if0 (sqrt (avg2 (sqrt ADF1) (- (/ d1d2 hd1) (- (sqrt ADF1) (avg2 d1d2 (avg2 d1d2 (- (/ (+ (avg2 d1d1 cell21) d2d2) hd1) (/ hd2
(max2 (min2 cell21 d2d2) (sqr ADF5)))))))))) cell11 ADF1)
[[x || y]]:
 (max3 (- (if0 cell20 cell22 vd2) (ADF3 (sqrt ADF1))) hd2 (max2 (max3 (- cell20 cell00) hd2 (/ d1d2 hd1)) cell12))
[[1]]:
 (sqrt (- (sqrt (- (sqrt ADF1) (avg2 d1d2 (- (/ d1d2 hd1) (min2 cell21 d2d2))))) (min2 cell21 d2d2)))
ADF0:
 (avg3 hd1 (abs cell00) vd1)
ADF1:
 (/ cell01 (if< (if1 ADF0 hd1 d2d2) (if> cell11 (/ cell01 (if< (if> cell11 ADF0 d2d2 (/ cell01 (if< (if> cell11 ADF0 d2d2 cell22)
(avg2 cell22 ADF0) (abs d1d2) (/ cell22 cell12)))) (ifmix d2d1 hd2 cell01) hd1 (avg2 cell12 (if< (/ cell01 (avg2 cell22 ADF0))
(avg2 cell22 ADF0) (if1 ADF0 hd1 d2d2) (/ cell22 cell12))))) d2d2 cell22) (/ (/ cell01 (avg2 cell22 ADF0)) (abs cell02)) cell11))
ADF2:
 (avg2 (avg2 (- ARG0 vd2) (avg2 d1d2 cell00)) (sqr d1d2))
ADF3:
 (/ (/ hd2 (/ (/ hd2 (/ (/ hd2 cell11) d2d2)) d2d2)) d2d2)
ADF4:
 cell11
ADF5:
 (- (sqrt cell21) (min3 (min3 (max2 d2d2 cell12) (- cell02 (min3 d2d2 (- cell22 (avg3 (- (sqrt cell21) (min3 d2d2 (min3 d2d2 (-
(min3 (max2 d2d2 cell12) (ADF3 vd2) (min3 d2d2 (min3 (min3 (min3 (max2 d2d2 cell12) (ADF3 vd2) (min3 d2d2 (min3 d2d2 (min3 d2d2
cell10 cell02) (min3 d2d2 (- cell22 (min3 d2d2 (- cell22 (min3 d2d2 cell20 cell02)) cell02)) cell02)) cell02)) cell20 (max2 d2d2
cell12)) cell20 (min3 (min3 d2d2 (min3 (- cell02 (min3 d2d2 (- cell22 (avg3 vd1 (sqrt cell21) cell21)) cell21)) cell20 cell02)
cell02) (- cell22 (min3 d2d2 cell20 cell02)) cell02)) cell02)) (min3 (ADF4 (sqrt (- cell22 (min3 d2d2 cell20 cell02))) (min3
(avg3 hd1 d1d1 cell10) (avg3 cell20 cell11 cell10) (ifmix d1d2 cell21 cell10))) cell20 (min3 d2d2 cell20 cell02))) cell02)
cell02)) cell20 (min3 d2d2 (min3 d2d2 (sqrt cell21) cell02) cell21))) cell21)) (min3 d2d2 (- cell22 (min3 d2d2 cell20 cell02))
cell02)) cell20 (min3 d2d2 (min3 d2d2 (min3 d2d2 cell10 cell02) (min3 d2d2 (- cell22 (min3 d2d2 (- cell22 (min3 d2d2 cell20
cell02)) cell02)) cell02)) cell02)))
```

6.3 Discussion

Figure 9 shows progress of the best-of-generation individual from generations 0 through 3 on four of the fitness cases. The first two fitness cases are from the "looks good" category and were highly pressured to match the known placement of materials within the cell. The other two were pressured much less, and to match only the known volume fraction of the materials in the cell. It is worthwhile to note that the need for the looks-good fitness cases was derived from previous failed solutions that gave accurate volume fractions but unphysical placement. It appears that their existence improves the physical realism of the solution even on the normal fitness cases.

The best-of-run individual is fairly complex, with a depth of 23. This run clearly obtained only a partial solution, since the best-of-run individual scored only 23 out of 29 hits. However, it demonstrated good progress towards a solution. The limits on time and available compute power let this run use 80 hours and 850 MB of RAM on a 1GHz Pentium III workstation, such that it was killed by the experimenter before reaching generation 250. One of the few observations that can be made about the best-of-run

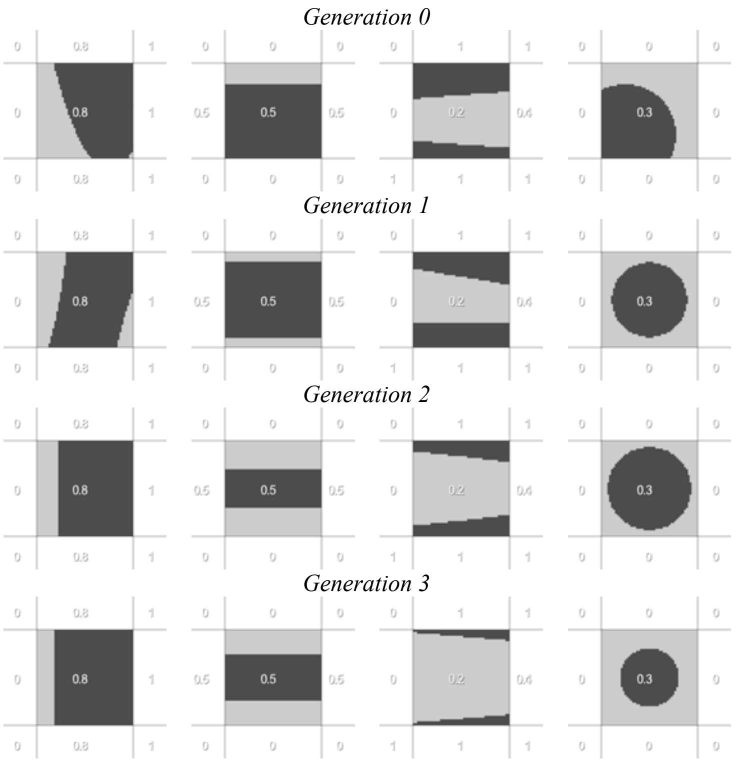

Figure 9: results from the best-of-generation individual for four fitness cases of the PQIC algorithm (given volume fractions are for the dark material)

individual is that it did make use of the ADFs, and there seems to be a lot of repetition of subexpressions. This implies that a better structured problem may have had a better chance of running to completion.

As such, although promising, it does not yet seem a solution made for practical application. In addition, although the use of higher-order curves than the normal PLIC algorithm in combination with fitness cases designed to make physically believable results implies greater connectivity across mixed cells, it needs to be tested on a wider array of cases and on more real-world problems to confirm this in a more general sense.

7 Conclusions

We have presented an extension to the MeshTV material interface reconstruction algorithm which is easy to implement and replace in the tools which use this algorithm. It continues to provide believable, continuous interfaces but greatly improves the accuracy of the generated volume fractions.

In addition, we present good progress toward a usable version of a piecewise quadric interface construction (PQIC) algorithm which generates very smooth, believable interfaces within a cell, as well as accurate volume fractions.

8 Future Work

The PQIC algorithm would make a good subject of further experimentation and research, as it has a great amount of freedom to solve the problem well. Using different function and terminal sets, a larger population and a parallel computer on which to run the evolution would be an excellent start. In addition, a larger window of cell volume fractions, a larger set of fitness cases, and possibly a higher order function than a quadric, would likely lead to a more manageable, more easily implemented result with smoother and more physically realistic interfaces than any known solution.

References

Amsden, A. A. (1966). “The Particle-in Cell Method for the Calculation of the Dynamics of Compressible Fluids”, *Los Alamos Scientific Laboratory Report LA-3466.*

Koza, John (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: The MIT Press 1992.

Nichols, B. D., and Hirt, C. W. (1975). “Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies”, *First International Conference on Numerical Ship Hydrodynamics, Gaithersburg, MD.*

Noh, W. F., and Woodward, P. (1976). “SLIC (Simple Line Interface Calculation)”, *Lecture Notes in Physics*, 59, Springer Verlag.

Parker, B. J., and Youngs, D. L. (1992). “Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces”, *Third Zababakhin Scientific Talks, Kyshtim, USSR*