Los Alamos National Laboratory

Asymmetry Detectors for Hydrocode Simulation Outputs

Mike Cannon, Pat Fasel, Rich Fortson, John Hogden, Don Hush,
Pat Kelly, Reid Rivenburgh, and Richard Strelitz
FEEMADS Project
Computer Research and Applications Group, CIC-3
Mail Stop B265

LANL Technical Report: LA–UR–00-5203

Report Date: October 30, 2000

Abstract

This report describes the asymmetry detection problem for large simulation code outputs. Three different asymmetry detectors are developed and analyzed. These detectors are compared based on the computational resources required, sensitivity to *sampling* and *aggregation* error, and ease of use (with respect to the choice of user specified parameters).

1 Preliminaries

Let $v(x, y, z, t)$ represent the evolution of a fluid flow (i.e. the solution to a set of differential equations) at spatial location (x, y, z) (in a rectilinear coordinate system) at time t. The dependent variable v is generally multidimensional and contains such components as density, volume fraction, internal energy, and velocity. For simplicity however, we consider a single dimension (e.g. density), so that v is a scalar. For fixed z, t we obtain a two dimensional slice through the flow. In this problem we expect v to exhibit radial symmetry within a slice (around an origin x_0, y_0).

Consider a single slice and let $v(x, y)$ be the corresponding field. Converting to radial coordinates gives

$$v(r, \theta) = v(x - x_0, y - y_0)$$

where

$$r^2 = (x - x_0)^2 + (y - y_0)^2$$

$$\tan(\theta) = \left(\frac{y - y_0}{x - x_0}\right)$$

In the definitions below we consider the behavior of the field v at fixed radial distances r from the point of symmetry. For convenience we adopt the notation $v_r(\theta) = v(r, \theta)$. Note that from a practical view we may not always have measurements for the full circle of values $0 \leq \theta \leq 2\pi$ at a given radius r. In particular, some experiments consider only half of the flow so that θ is limited to the range $0 \leq \theta \leq \pi$.

Definition 1. The slice $v_r(\theta)$ is radially symmetric if, for every r,

$$v_r(\theta) = c_r, \quad 0 \leq \theta \leq 2\pi$$

where c_r is constant.

Note that c_r may vary with r, but must be constant over θ. Let the cumulative distribution of field values at radius r be

$$P_{v_r}(\alpha) = \frac{1}{2\pi} \int_0^{2\pi} I(v_r(\theta) < \alpha)\, d\theta \tag{1}$$

where $I(\cdot)$ is the indicator function that equals 1 when its argument is true and 0 otherwise. The corresponding density function is given by

$$p_{v_r}(\alpha) = \frac{\partial P_{v_r}(\alpha)}{\partial \alpha} \tag{2}$$

Note that when P_{v_r} is discontinuous, $p_{v_r}(\alpha)$ is defined in terms of impulse functions. Let $q_r(\alpha)$ be the fraction of field values concentrated at $v = \alpha$,

$$q_r(\alpha) = \lim_{\delta \to 0} \left[P_{v_r}(\alpha + \delta) - P_{v_r}(\alpha - \delta)\right] \tag{3}$$

and let $\tilde{v}_r$ be a value with the largest concentration

$$\tilde{v}_r = \arg\max_{\alpha} q_r(\alpha) \tag{4}$$

and $\tilde{p}_r$ be its concentration fraction

$$\tilde{p}_r = \max_{\alpha} q_r(\alpha) \tag{5}$$

Corollary 1. *The slice $v_r(\theta)$ is radially symmetric iff, for every r, $\tilde{p}_r = 1.0$.*

To detect an asymmetry we can simply check for a violation of the condition above, i.e. we check to see that $v_r(\theta)$ is constant along the arc $(r, [0, 2\pi])$. By checking this condition for each r we can locate the radii where asymmetries occur. But to determine the positions in θ where the asymmetry occurs requires more work. First we must define what it means for a specific point $v_r(\theta)$, $((r, \theta)$ fixed), to be asymmetric. The definition below is one possibility. It is based on the assumption that asymmetric points are those that differ from the "typical" value. The "typical" value is defined to be the value taken by more than half the points along the arc. If there is no value taken by more than half the points then no value along the arc is considered typical, and all points are considered to be asymmetric.

Definition 2. If $\tilde{p}_r \geq 0.5$ and $v_r(\theta) \neq \tilde{v}_r$ then the point (r, θ) is asymmetric. Otherwise, if $\tilde{p}_r < 0.5$ then (r, θ) is asymmetric for all $\theta \in [0, 2\pi]$.

2 Discretization

Computer simulations generate a discretized approximation to $v(x, y)$. Initially we consider discretizations on a rectangular grid where individual data points represent (approximations to) the field values $v(x_i, y_j)$ where

$$x_i = x_1 + (i-1)\Delta_x, \;\; i = 1, 2, ..., N_x$$

$$y_j = y_1 + (j-1)\Delta_y, \;\; j = 1, 2, ..., N_y$$

for some (x_1, y_1) and some fixed Δ_x and Δ_y. Since the actual value of the x and y coordinates are irrelevant to our work we adopt the notation $v(i, j) = v(x_i, y_j)$ where i, j are integral (and start at 1). Also we let (i_0, j_0) be the "indices of symmetry", i.e. they correspond to the indices of the point at location (x_0, y_0). If the discretization does not contain the point at (x_0, y_0), then we assume that it lies half way between discretized points so that the fractional part of the indices i_0 and/or j_0 is 0.5. This assumption is essential for the P-squared detector described in Section 3.

This (uniform) discretization over (x, y) maps to a (nonuniform) discretization over (r, θ). This mapping has the property that a relatively small number of field values are present at any given radius r. For example consider the discretized value at location (i, j), with $r_{ij} = \sqrt{(i-i_0)^2 + (j-j_0)^2}$. The set of locations at the same distance r_{ij} from the origin (i_0, j_0) is given by

$$S_{ij} = \{m, n : (m-i_0)^2 + (n-j_0)^2 = r_{ij}^2\}$$

It is easy to see that the following 8 locations are in this set

$$\begin{array}{ll} (i,j) & (j,i) \\ (-i,j) & (-j,i) \\ (i,-j) & (j,-i) \\ (-i,-j) & (-j,-i) \end{array} \tag{6}$$

We call these points *Pythagorean Partners*, and depending on the boundaries of the simulation there may be as few as 2 and as many as 8 available in the discretization. The set S_{ij} may contain other locations as well. For example, suppose i_0 and j_0 are integral, and consider all the addends of r_{ij}^2 (i.e. all a, b such that $a + b = r_{ij}^2$). Any pair of addends that are both perfect squares contributes another member to S_{ij}. As an example let $(i_0, j_0) = (0, 0)$ and $(i, j) = (3, 4)$. Then $r^2 = 25$, and the Pythagorean Partners are

$$\begin{array}{ll} (3,4) & (4,3) \\ (-3,4) & (-4,3) \\ (3,-4) & (4,-3) \\ (-3,-4) & (-4,-3) \end{array}$$

But $r^2 = 25$ has an addend pair other than $(3^2, 4^2)$ that contains perfect squares, namely $(0, 25)$. Thus, S_{ij} contains the four additional locations

$$\begin{array}{ll} (0,5) & (5,0) \\ (0,-5) & (-5,0) \end{array}$$

The exact number of addend pairs that form perfect squares depends on the value of r^2. The first few values of r^2 that have more than one such pair are listed in the table below.

r^2	addend pairs that are perfect squares
25	{ (9,16), (0,25) }
50	{ (1,49), (25,25) }
65	{ (1,64), (16,49) }
100	{ (36,64), (0,10) }

Even though S_{ij} may contain additional entries, only those in (6) are grouped together as Pythagorean Partners (see the illustration in Figure 1). There are two reasons for this. First, consider a slice that contains a radially symmetric field. Suppose we place a rectangular grid on this slice. This breaks it into a collection of rectangular boxes. The boxes indexed by the Pythagorean Partners are perfect rotations of one another, but the boxes indexed by the other members of S_{ij} generally are not. Thus, a detection scheme based on Pythagorean Partners should not produce false alarms when the simulation is truly symmetric. The second reason is that it is simpler to code the Pythagorean Partners than the members of S_{ij}. This is not a strong consideration however, since the the members of S_{ij} can be identified efficiently using a pre-computed look-up table. Nevertheless, a look-up is not needed to efficiently identify the Pythagorean Partners.

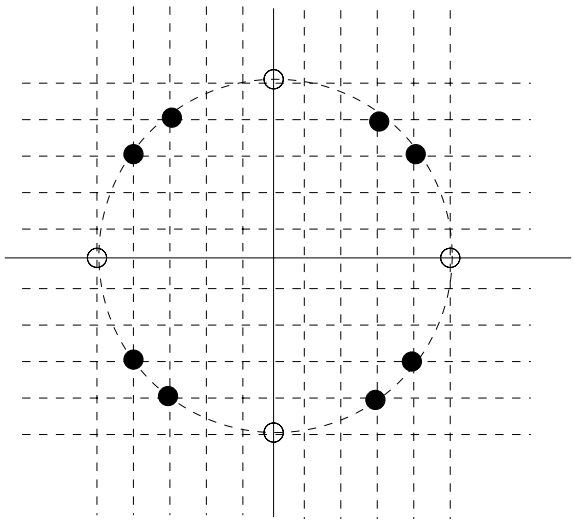

Figure 1: Example of Pythagorean Partners (indicated by dark circles) and other points that lie at the same radius (indicated by open circles).

The "P-squared" detector below determines asymmetry by comparing points with their Pythagorean Partners. However, detectors based solely on these points may be subject to "sparse sampling" error (which we discuss below). On the other hand, we often expect near-by arcs to have similar field values, suggesting that we may be able to mitigate sparse sampling error by developing asymmetry detectors that use an aggregation of samples from near-by arcs. This is accomplished by aggregating field values into "R-lists" and implementing a detection scheme on the values in an R-list as if they all belong to the same arc. An R-list is a list of field values whose radii quantize to the same value. More specifically the R-list v_R is defined as

$$v_R = \{v(x_i, y_j) : \lfloor r \rfloor = R\}$$

where $r = \sqrt{(i - i_0)^2 + (j - j_0)^2}$ and $R \in \{0, 1, ..., R_{max}\}$. Values in the R-list are arranged in no particular order and are indexed using the notation $v_R[k]$ to refer to the k^{th} entry. The aggregation performed by the R-list approach may lead to detection errors, particularly in areas of the simulation where near-by arcs *do not* have similar field values (e.g. near material boundaries).

Inputs to the asymmetry detectors below are slices from a computer simulation of the flow, discretized on a rectangular grid. Outputs are slices on the same discretized grid with a '1' at each location where an asymmetry is detected and a '0' otherwise. To keep track of the correspondence between R-list entries and their (i, j) coordinates on the rectangular grid we build a look-up table. Algorithm 1 builds a look-up table that maps from R-list entries (R, k) to the rectangular coordinates (i, j), and Algorithm 2 uses this table to perform the mapping. The run time of the `Build-Rlist2Rect-LUT` is $T \leq \sqrt{2}\max(N_x, N_y) + N_x N_y$. Note that this same look-up table is used to process *every* slice from *every* time step of the simulation, and so its run time is inconsequential to the overall run time of the detectors described in the next section.

Algorithm 1 `Build-Rlist2Rect-LUT`: Build Rlist-to-rectangular look-up table.

`INPUTS:`
N_x, N_y: size of discretized slice
i_0: index of symmetry for x coordinate
j_0: index of symmetry for y coordinate

`OUTPUTS:`
$ix[R,k]$: x coordinate index i of k^{th} sample in R-list R.
$iy[R,k]$: y coordinate index j of k^{th} sample in R-list R.
$Size[R]$: number of samples in R-list R.

$R_{max} \leftarrow \max_{i,j} \left\{ \lfloor 0.5 + \sqrt{(i-i_0)^2 + (j-j_0)^2} \rfloor \right\}$

{Initialize list sizes.}
for ($R = 0$ to R_{max}) **do**
  $Size[R] \leftarrow 0$
end for

{For each sample: determine R-list and insert indices.}
for ($i = 1$ to N_x) **do**
  for ($j = 1$ to N_y) **do**
    $R \leftarrow \lfloor 0.5 + \sqrt{(i-i_0)^2 + (j-j_0)^2} \rfloor$
    $ix[R, Size[R]] \leftarrow i$
    $iy[R, Size[R]] \leftarrow j$
    $Size[R] \leftarrow Size[R] + 1$
  end for
end for

Algorithm 2 `Rlist2Rect`: Convert from R-list to rectangular coordinates.

`INPUTS:`
R, k: indices of the k^{th} entry in R-List R

`OUTPUTS:`
(i, j): the rectangular coordinate indices

$i \leftarrow ix[R,k]$
$j \leftarrow iy[R,k]$
return(i, j)

To perform asymmetry detection on the entire 3d simulation we run one of these detectors over each slice in turn. Since the computations from one slice to the next are decoupled, the detection process can be parallelized quite naturally by placing different slices (or parts of slices) on different processors.

3 Asymmetry Detectors

In this section we describe three asymmetry detectors. The primary sources of errors for these detectors are *sampling error* and *aggregation error*. Sampling error is the error due to incomplete knowledge about the field values along an arc (i.e. we have only a finite number of samples along any given arc). Aggregation error refers to the error that results from the aggregation of values from different arcs into a single R-list.

The three detectors will be compared using the following criteria.

1. computational cost
2. user specified parameters, (how many, how easy are they to choose, and how robust is the algorithm to them)
3. sensitivity to sampling error
4. sensitivity to aggregation error
5. ability to tune their sensitivity to the size/magnitude of the asymmetry

In addition, we will use specific (synthetic) examples of field values along an arc, and their distributions, to make comparisons. These examples are provided in Figures 2-9, and will be used to illustrate strengths and weaknesses of the three detectors. According to Definition 2, Figures 2, 3 and 5 represent situations where the arc has a symmetric component. In the other figures the arcs are entirely asymmetric. In some simulations however, small asymmetries may be unavoidable and we may want detectors that can be made insensitive to them. For example, the situation in Figure 9 may be acceptable if the width (or variance) of the distribution is small compared to the range of values taken by the data in the full simulation. The same can be said about Figures 6-8. In the same vein, we may wish to classify the values corresponding to the dominant mode in Figure 4 as symmetric. Finally, note that Figures 6-8 show how three different arcs can lead to the same uniform density.

Pythagorean Partners Detector (also called the P-squared method) This detector makes a decision at each discretized point using only points that lie at *exactly* the same radius from the point of symmetry. These points are called *Pythagorean partners* and are defined in equation (6). Recall that there are at most 8 such points at any given radius (see Figure 1). Note that these points are symmetric with respect to lines passing through the origin x_0, y_0 at angles 0, $\pi/4$, $\pi/2$ and $3\pi/4$. That is, if we were to fold the slice at these angles then P-squared partners will be aligned in pairs on top of one another. The advantage of this detector is that it has *zero* aggregation error. That is, errors made by this detector are not due to aggregation. Consequently, the accuracy of this detector is not degraded near material boundaries in the simulation.

The detector works as follows. Suppose there are $2 \leq q \leq 8$ Pythagorean partners. Then for each point v we compute the absolute difference between v and its $q-1$ partners. Definition 2 suggests that if v is a symmetric point then at least half of these differences will be zero. On the other hand, if v is an asymmetric point then at least half of these

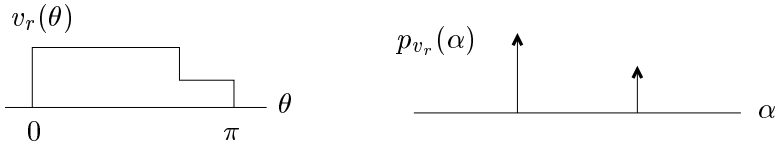

Figure 2: Bimodal A.

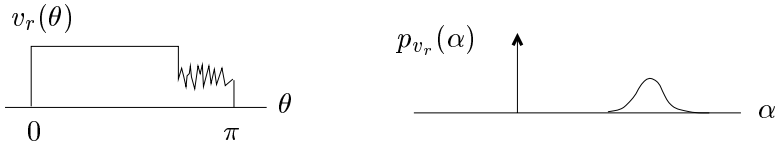

Figure 3: Bimodal B.

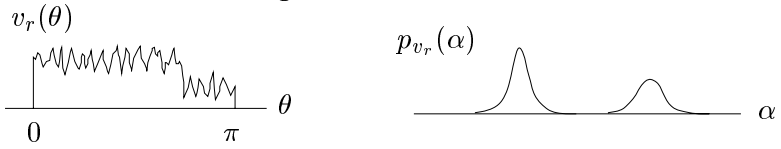

Figure 4: Bimodal C.

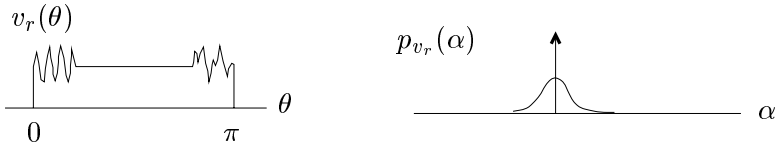

Figure 5: Bimodal D.

differences will be nonzero. So we use the median of the difference values as a measure of asymmetry. Pseudo code for this procedure is shown in Algorithm 3. The double loop indexes through each sample in one octant of the slice, and gathers the $q-1$ (up to 7) partners for each sample in the formation of V. Then, all $q(q-1)/2$ pairwise absolute differences are computed, each of which will be referenced q times in the "for all" loop that follows. This loop sorts the distance values for each value $v(m,n)$ in V, computes the median, and compares it to a threshold. The run time of this procedure is

$$\begin{aligned} T_P &\leq c(1/q_{max})N_xN_y\left[0.5q_{max}(q_{max}-1)+q_{max}^2\log(q_{max})\right] \\ &= cN_xN_y\left(0.5(q_{max}-1)+q_{max}\log(q_{max})\right) \end{aligned}$$

where q_{max} is the maximum value of q in the simulation and c is a small positive constant.

This detector has a single user defined parameter, the asymmetry threshold. This value should be chosen in the range $0 \leq AsymThold \leq (\max_{i,j}\{v[i,j]\} - \min_{i,j}\{v[i,j]\})$. Smaller thresholds make the detector more sensitive. For example, if the goal is to detect asymmetries of *all* magnitudes then $AsymThold$ should be set to a value that is on the order of the machine precision. On the other hand, if the goal is to detect asymmetries that are large in magnitude, say roughly 10% of the range of the data values, then $AsymThold$ should be set to $0.1 \cdot (\max_{i,j}\{v[i,j]\} - \min_{i,j}\{v[i,j]\})$.

The advantages of the Pythagorean Partners (referred to as "P-squared") method are that it is simple, computationally efficient, and has zero aggregation error. Its disadvantage

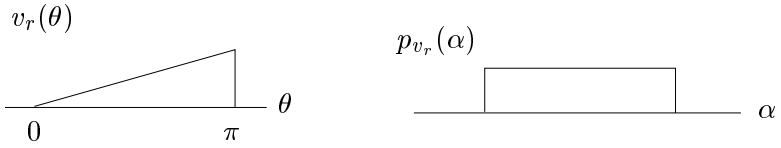

Figure 6: Uniform A.

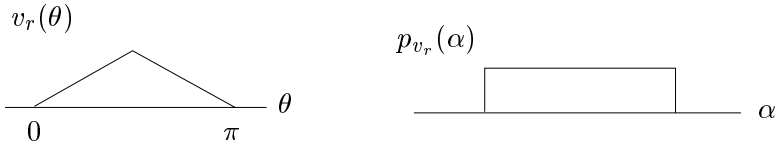

Figure 7: Uniform B.

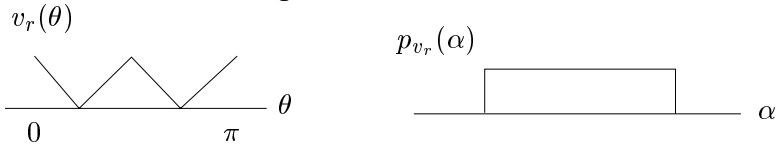

Figure 8: Uniform C.

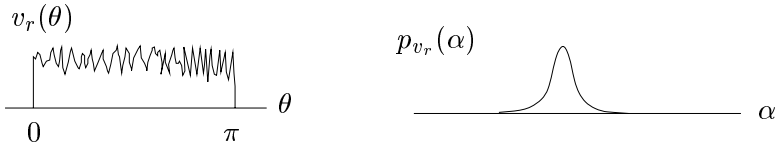

Figure 9: Unimodal.

is that it can make decisions are not always consistent with Definition 2. Errors come in two forms; missed detections and false alarms. This method is more likely to produce false alarms than missed detections. Missed detections seem unlikely since they require a *radially asymmetric* arc to exhibit *local symmetries* with respect to points at the folding angles. For example, the arc in Figure 8, which is locally symmetric about the points at angles $\pi/4$, $\pi/2$ and $3\pi/4$, represents a worst-case scenario for the P-squared detector in that it will label the entire arc as symmetric, even though it should be labeled asymmetric. At the same time however, the P-squared detector will perform quite well on the arc in Figure 6 which has essentially the same distribution. Generally speaking the special symmetries that "fool" the P-squared detector are expected to be rare.

Errors from the P-squared algorithm are more likely to come in the form of false alarms, i.e. detections at points where no asymmetry is actually present. The worst case scenario is when q is even, and half the points are sampled from the symmetric part of the arc while the other half are from the asymmetric part. In this case *all* samples will be declared asymmetric. This situation is more likely when the asymmetry is concentrated around one (or more) of the "folding angles", $0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2, 7\pi/4$. For example, the arc in Figure 7, where the asymmetry is folded around $\theta = \pi/2$, represents a worst-case scenario in this regard.

The P-squared detector should perform quite well on the other examples in Figures 2-9. Note that we can tune the sensitivity of this detector to the desired magnitude of the

Algorithm 3 `PathagoreanPartners`: This routine implements the P-squared asymmetry detector that compares the $4 \leq q \leq 8$ samples from a rectangular grid that lie at precisely the same distance from the point of radial symmetry. The function `H()` below is the heaviside function that returns a '1' when its argument is ≥ 0 and a '0' otherwise.

`INPUTS:`
$\{v[i,j]\}$: a discretized slice of field values, $i = 1, 2, ..., N_x$, $j = 1, 2, ..., N_y$
i_0: index of symmetry for x coordinate
j_0: index of symmetry for y coordinate
$AsymThold$: Asymmetry threshold, $0 \leq AsymThold \leq \max_{i,j} |v[i,j]|$

`OUTPUT:`
$h[i,j]$: detection array with a '1' where an asymmetry is detected and a '0' otherwise

for ($i = 0$ to $\max(\lceil i_0 \rceil, N_x - \lceil i_0 \rceil)$) **do**
  for ($j = 0$ to i) **do**

    {Form the set of pathagorean partners.}
    $V = \{v(m,n) : m, n \in \{\lceil i_0 \rceil \pm i, \lceil j_0 \rceil \pm j\}, 1 \leq m \leq N_x, 1 \leq n \leq N_y\}$

    {Compute absolute differences.}
    $D \leftarrow$ set containing all $p(p-1)/2$ pairwise absolute differences of values in V

    {Compute detection value for each point.}
    for all $v(m,n)$ in V **do**
      $D_{m,n} \leftarrow$ subset of D containing all $(p-1)$ absolute differences involving $v(m,n)$
      $M \leftarrow median(D_{m,n})$
      $h[m,n] \leftarrow$ `H`$(M - AsymThold)$
    end for
  end for
end for

asymmetry through our choice of $AsymThold$.

Finally, since the P-squared algorithm makes decisions using a relatively small number of samples (i.e. $2 \leq q \leq 8$) it is subject to sampling error. Arcs that are actually asymmetric may be sampled in such a way that the samples appear to be symmetric. Obviously this type of error is less frequent for the larger values of q.

Outlier Detector This approach is motivated by the intuition that asymmetries should appear as outliers in the distribution of points along an arc. Outliers are points that are unlikely to have been drawn from the "model" distribution. To perform outlier detection from sample data alone we need a way to produce a reliable estimate of the sample distribution. There are many possibilities. For example, if the model distribution is Gaussian, then a robust estimate of its mean and standard deviation can often be determined from a sample data set (which contains outliers) by computing a *trimmed* mean and standard deviation. Trimming discards a fraction τ of the largest and smallest samples (i.e. the

samples most likely to be outliers). Once the trimmed estimates are in hand, outlier detection can be performed by computing $(v - \mu)/\sigma$ and thresholding this value.

In our problem the intuition is that the trimmed mean will be a good representative of the value $\tilde{v}_r$. With appropriate choice of τ this will certainly be the case for the arcs in Figures 2, 3 and 5. Figure 4 can be interpreted as a "noisy" version of Figure 3, where there is no $\tilde{v}_r$ with $\tilde{p}_r \geq 0.5$, but the trimmed mean continues to play the role of the "typical" or "most likely" value along the arc.

Pseudo code for the outlier detector is shown in Algorithm 4. Values in the R-list are first sorted so that trimming can be accomplished by simply skipping values at the beginning and end of the list. Next the (trimmed) mean and standard deviation are computed, and then the detection formula is applied for each value in the R-list. The run time of this procedure is

$$T_O \leq c \sum_{R=0}^{R_{max}} Size(R) + Size(R) \cdot \log(Size(R))$$

for some small positive constant c. Using the approximation $Size[R] \approx 2\pi R$ and $R_{max} \approx N = \sqrt{N_x^2 + N_y^2}$ gives

$$T_O \leq 2\pi c \sum_{R=1}^{N} R + R \cdot \log(2\pi R)$$

Simplification yields

$$T_O \leq 7cN^2(1 + \log(7N))$$

This detector has two user defined parameters, the trim fraction τ and the asymmetry threshold $AsymThold$. Since over half the samples take a value $\tilde{v}_r$ when symmetry is present, τ can be chosen quite large (i.e. very close to 0.5) without significantly degrading performance, and so this is recommended. In the ideal case, when the trimmed mean represents $\tilde{v}_r$, the asymmetry threshold should be very close to zero (since any value that differs from the mean is asymmetric by definition). Larger values of $AsymThold$ can be used if we want to be insensitive to small asymmetries in the simulation. In any case however, it seems unlikely that $AsymThold$ should ever be chosen larger than 1.0.

This detector will make errors any time the mean does not represent the value where more than half the arc is concentrated. For example, regardless of the choices for τ and $AsymThold$, this detector will make errors (missed detections) on the arcs in Figures 4-9. On the other hand, in situations where we want to be insensitive to small asymmetries in the simulation itself, this detector can be quite effective. In this case, with the proper choices of τ and $AsymThold$, this detector can perform quite well on the arcs in Figures 2-5, 9.

Mode Concentration Detector (clustering)

This detector represents an approximation to the direct implementation of Definition 2. Note that a direct implementation would require us to compare points along the arc with

Algorithm 4 `OutlierDetection`: This routine detects asymmetric locations along an arc (R-list) by comparing values in the list to the trimmed mean. It assumes that the `Build-Radial2Rect-LUT` routine has already been used to initialize the Rlist-to-recangular look-up table. The function `H()` below is the heaviside function that returns a '1' when its argument is ≥ 0 and a '0' otherwise.

`INPUTS:`
$\{v[i,j]\}$: a discretized slice of field values, $i = 1, 2, ..., N_x$, $j = 1, 2, ..., N_y$
i_0: index of symmetry for x coordinate
j_0: index of symmetry for y coordinate
τ: the trim fraction (from each side of the distribution) $(0 \leq \tau < 0.5)$
$AsymThold$: the standard deviation threshold $(AsymThold > 0)$

`OUTPUT:`
$h[i,j]$: detection array with a '1' where an asymmetry is detected and a '0' otherwise

for ($R = 0$ to R_{max}) **do**

{Form the R-list and sort.}
$v_R \leftarrow \{v[i,j] : \lfloor 0.5 + \sqrt{(i-i_0)^2 + (j-j_0)^2} \rfloor = R\}$
$\tilde{v}_R \leftarrow \texttt{Sort}(v_R)$

{Compute trimmed mean and standard deviation.}
$sum \leftarrow 0$
$sum2 \leftarrow 0$
for ($i = \max(1, \lfloor \tau \cdot Size[R] \rfloor)$ to $\lfloor Size[R] \cdot (1-\tau) \rfloor$) **do**
$sum \leftarrow sum + \tilde{v}_R[i]$
$sum2 \leftarrow sum2 + \tilde{v}_R^2[i]$
end for
$N \leftarrow \max(1, \lfloor Size[R] \cdot (1-2\tau) \rfloor)$
$\mu_R \leftarrow sum/N$
$\sigma_R \leftarrow \sqrt{(sum2/N) - \mu_R^2}$

{Compute detection values.}
for ($k = 1$ to $Size[R]$) **do**
$(i,j) \leftarrow \texttt{Radial2Rect}(R,k)$
$h[i,j] \leftarrow \texttt{H}(|v_R[k] - \mu_R| - AsymThold \cdot \sigma_R)$
end for
end for

$\tilde{v}_r$, the value where at least half of the arc is concentrated. We employ a clustering routine to determine value(s) where the data is concentrated. More specifically, we cluster the data from an R-list using a routine that automatically determines the number of clusters and the cluster membership with a single pass through the data. The cluster with the largest number of entries determines our approximation to $\tilde{v}_r$ (the mode). If this cluster

contains less than half the total number of points then all points in the R-list are declared asymmetric, otherwise points in the clusters with less than half the data are declared asymmetric.

Pseudo code for this procedure is shown in Algorithm 5. The first value in the R-list forms the representative for the first cluster. For each successive sample we determine the closest cluster, and then assign the sample to that cluster if it is within $BinSize$ of the cluster representative. Otherwise we create a new cluster and use the current sample as its representative. Once clustering is finished it is a simple matter to determine which samples belong to clusters with less than half the points.

The run time of this procedure is determined as follows. The number of steps required to cluster a single R-list is no less than $Size[R]$ and no more than $Size[R] \cdot C[R]$, where $C[R]$ is the final number of clusters for R-list R. Then it takes $Size[R]$ steps to compute the detection values, so the total run time T_C for one slice of data is bounded by

$$c_1 \sum_{R=0}^{R_{max}} Size[R] \leq T_C \leq c_2 \sum_{R=0}^{R_{max}} (1 + C[R]) \cdot Size[R]$$

for small positive constants c_1 and c_2. The lower bound simplifies to

$$c_1 \sum_{R=0}^{R_{max}} Size[R] = c_1 N_x N_y$$

For the upper bound let

$$N_C = \max_{0 \leq R \leq R_{max}} (1 + C[R])$$

be the maximum number of clusters formed over all the R-lists. With this we have

$$T_C \leq c_2 \sum_{R=0}^{R_{max}} N_C \cdot Size[R] = N_C N_x N_y$$

Note that we can bound N_C in terms of N_x and N_y as follows,

$$N_C \leq \max_R Size[R] \approx \lfloor 2\pi \min(N_x, N_y) \rfloor$$

so that T_C is clearly less than $7c_2 N_x N_y \min(N_x, N_y)$. In summary we have

$$c_1 N_x N_y \leq T_C \leq 7c_2 N_x N_y \min(N_x, N_y)$$

This detector has only one parameter called $BinSize$. $BinSize$ controls the sensitivity of the method. A sample must be closer to the cluster representative than $BinSize$ to be treated as part of that cluster. Ideally, in the absence of noise and numerical imprecision, individual clusters would represent individual values and we would set $BinSize = 0$. In practice however, if we wish to be robust to errors introduced by numerical imprecision we should set $BinSize$ equal to the magnitude of the maximum simulation error we expect to see due to this imprecision (this magnitude usually increases as the simulation moves forward). In addition, we may wish to be insensitive to small amounts of noise due to aggregation, or small asymmetries that occur naturally in the simulation. In this case we would chose an even larger $BinSize$. With appropriate choice of $BinSize$ the "Mode Concentration" detector will perform quite well on all the examples in Figures 2-9.

Algorithm 5 `ModeConcentrationDetector`: This routine detects asymmetric locations along an arc by clustering points from an R-list and then declaring all points that belong to a cluster with less than half the points to be asymmetric. It assumes that the `Build-Rlist2Rect-LUT` routine has already been used to initialize the Rlist-to-recangular look-up table.

`INPUTS:`
$\{v[i,j]\}$: a discretized slice of field values, $i = 1, 2, ..., N_x$, $j = 1, 2, ..., N_y$
i_0: index of symmetry for x coordinate
j_0: index of symmetry for y coordinate
$BinSize$: size of the 'bin' (i.e. interval) for a cluster

`OUTPUT:`
$h[i,j]$: detection array with a '1' where an asymmetry is detected and a '0' otherwise

for ($R = 0$ to R_{max}) **do**
  {Form the R-list.}
  $v_R \leftarrow \{v[i,j] : \lfloor 0.5 + \sqrt{(i-i_0)^2 + (j-j_0)^2} \rfloor = R\}$
  {Proceed only if the R-list has more than 1 entry.}
  if ($Size[R] > 1$) **then**
    {Cluster the samples in the R-list.}
    $(i,j) \leftarrow$ `Radial2Rect`$(R, 1)$
    Create a new cluster with $v[i,j]$ as its representative
    for ($k = 2$ to $Size[R]$) **do**
      $(i,j) \leftarrow$ `Radial2Rect`(R,k)
      $c \leftarrow$ index of cluster that is closest to $v[i,j]$
      if ($|v[i,j] - Representative[c]| \leq BinSize$) **then**
        Add $v[i,j]$ to cluster c
      else
        Create a new cluster with $v[i,j]$ as its representative
      end if
    end for
    {Compute detection values.}
    for ($k = 1$ to $Size[R]$) **do**
      $(i,j) \leftarrow$ `Radial2Rect`(R,k)
      $c \leftarrow$ index of cluster to which $v[i,j]$ is assigned
      $N \leftarrow$ number of points in cluster c
      if ($N < Size[R]/2$) **then**
        $h[i,j] \leftarrow 1$
      else
        $h[i,j] \leftarrow 0$
      end if
    end for
  end if
end for

4 Run Time Summary

The table below summarizes the run time bounds for the three asymmetry detectors.

Detector	**Run Time**
Pythagorean Partners	$T_P \leq cN_xN_y\,(0.5(q_{max}-1)+q_{max}\log(q_{max}))$
Outlier	$T_O \leq 7c(N_x^2+N_y^2)(1+\log(7\sqrt{N_x^2+N_y^2}))$
Concentration	$c_1N_xN_y \leq T_C \leq 7c_2N_xN_y\min(N_x,N_y)$

A plot of these bounds is shown in Figure 10. To obtain the curves in this plot we have set the parameters as follows:

$$c = c_1 = c_2 = 1$$
$$q_{max} = 8$$
$$N_x = N_y$$

The horizontal axis in Figure 10 corresponds to the size N of a 2d slice (where $N = N_xN_y$), and the vertical axis is the run time bound. The run time of the Mode Concentration detector depends on the number of asymmetries present in the simulation, so Figure 10 shows two curves for this detector; a best case and a worst case (note that the best case curve C has such a small slope relative to the others that it appears to fall on top of the horizontal axis). While the best case for the Mode Concentration detector is superior to the other two, its worst case is significantly slower. Overall, the run times scale like N, $N\log(\sqrt{N})$, and $[N, N^{3/2}]$ for the Pythagorean Partners, Outlier and Mode Concentration detectors respectively.

The analysis above provides bounds on the number of steps required to process a single slice. Bounds for the full 3d simulation (at one time step) can be obtained by multiplying by N_z (the number of slices). A single time step from a 3d simulation can be processed in parallel by distributing the slices uniformly across processors. We have employed this strategy to develop parallel algorithms for all three detectors. To determine their actual run times and to test scalability of the code the following experiments were performed. Artificial data was generated in a cube of size $N_x \times N_y \times N_z$, where $N_x = N_y = N_z$, and N_x ranges from 100 to 600. The data was generated with random field values drawn from a uniform distribution (this creates a scenario that leads to a near worst case run time for the mode concentration detector). The results of these experiments are shown in Figures 11 and 12. Figure 11 shows the actual run time verses the problem size. These run times are in close agreement with the bounds in Figure 10 (note that the horizontal axis in Figure 10 is N_xN_y while in Figure 11 it is N_x). The P-squared method is clearly the fastest of the three, and the mode concentration method is the slowest (on this problem). More specifically, the P-squared method is roughly an order of magnitude faster than either of the alternatives for all problem sizes in this study. Note also that it takes only *3 minutes* to run the largest problem ($600 \times 600 \times 600$). Figure 12 shows the speed-up verses the number of processors for the $600 \times 600 \times 600$ problem. The formula for speed-up in this plot is

$$\text{speed up for P processors} = \frac{\text{run time on 20 processors}}{\text{run time on P processors}}$$

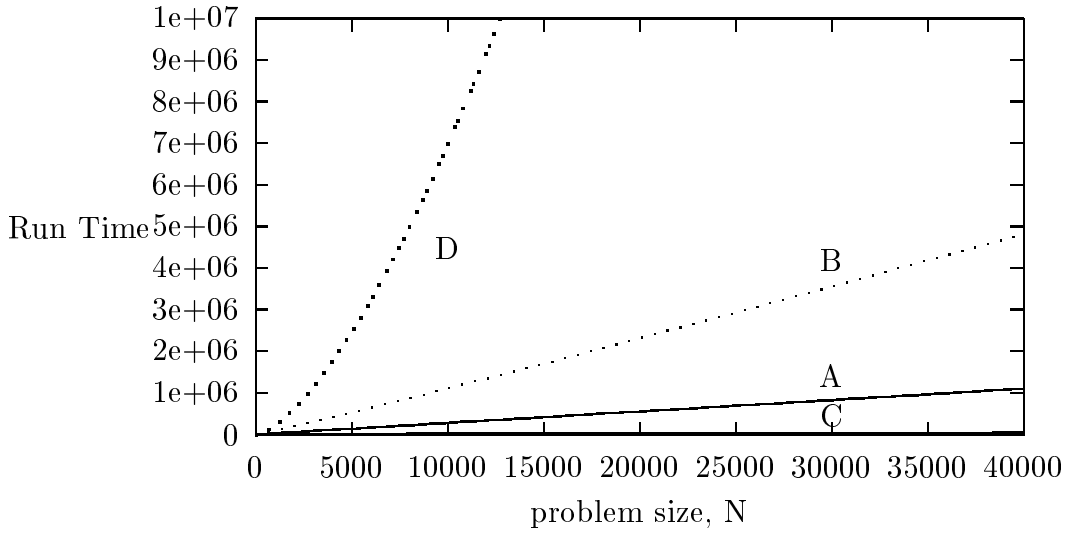

Figure 10: Run Time Comparison: (A) Pythagorean Partners, (B) Outlier, (C) Mode Concentration (lower bound), and (D) Mode Concentration (upper bound). Note that C has such a small relative slope that is appears to fall on top of the horizontal axis.

Note that we have used a nonstandard definition of speed-up in that the numerator corresponds to the run time on 20 processors instead of 1. This definition is used here because we were unable to run this problem on a single processor (because of its size). The results in Figure 12 suggest that all three algorithms scale reasonably well. They fall short of the "ideal" linear speed-up primarily because of the overhead associated with distributing the data onto the processors up front, and collecting the results at the end. In fact, since the P-squared algorithm has the fastest run time it also has the worst speed-up simply because the process of distributing the data up front and collecting the results at the end represents a larger fraction of the overall run time.

5 Empirical Results

In this section we provide detection results for a simple 3d simulation involving a cylindrical puck and a rectangular plate. The simulation is computed in a rectangular cube with physical coordinates (in centimeters)

$$\begin{aligned} -10 \leq x \leq 10 \\ 0 \leq y \leq 10 \\ -10 \leq z \leq 10 \end{aligned}$$

A rectangular plate of size $20 \times 10 \times 1$ cm is placed in the middle of the cube (i.e. the center of the plate is at location $z = 0$), and a cylindrical puck with diameter 2.5 cm and thickness 1 cm is centered 2 cm above the plate. This experiment is discretized to a $200 \times 100 \times 100$ grid (listed in the order x, y, z) for simulation.

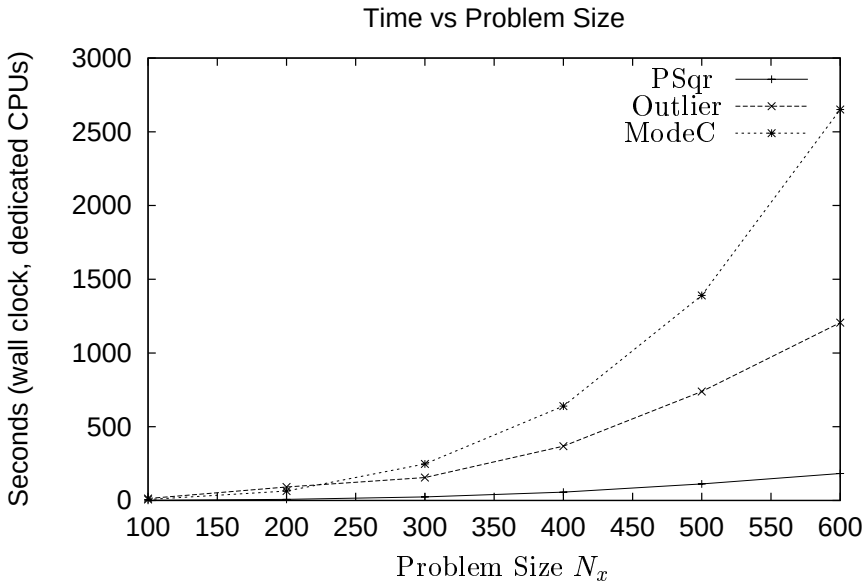

Figure 11: Run time verses problem size for the three detection methods using 120 CPUs.

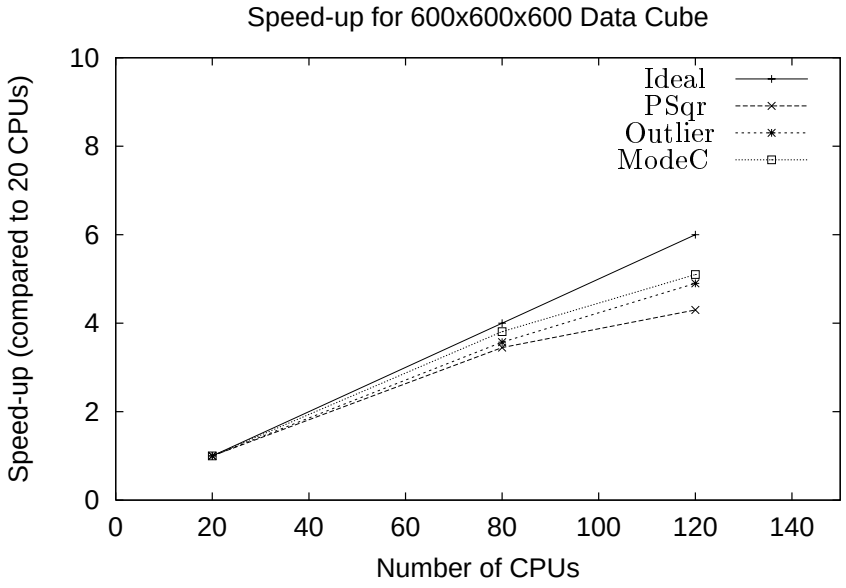

Figure 12: Speed-up (relative to the run time on 20 CPUs) for the three detection methods.

In the simulation the puck moves towards the plate at a very high velocity until it slams into the plate. The simulation is terminated once the puck has produced a large indentation in the plate. Although the simulation code produces several fields, the results here are for the density field only. In addition, the simulation code provides output data files for the puck and the plate separately. The results below are for the plate. We provide results for two different simulations;

one that is intended to be symmetric and one that is purposefully asymmetric. In the symmetric simulation we simply slam the puck squarely into the plate, and in the asymmetric simulation we tilt the puck slightly so that the indentation is formed asymmetrically.

As we shall see, all three of our detectors discovered asymmetries in the symmetric simulation! This suggested that there was an error in the simulation code. Indeed, the error was traced to a bug in the code, which has now been fixed. It turns out that the bug was in the formation of the cylindrical puck. That is, the puck was not perfectly symmetric at the start of the simulation and so once it made contact with the plate it introduced unexpected asymmetries. Although these asymmetries started out small, they grew rapidly and ended up as large as the asymmetries produced by the asymmetric simulation. We compare the results from this "symmetric" simulation and those for the asymmetric simulation below.

Our first set of results are for the symmetric simulation, and we start by examining two global statistics; the *scale* and the *size* of asymmetry at each time step. The scale statistic is represented by the maximum *asymmetry value* at each time step, where the asymmetry value a for each boxel is defined as follows for each of the three detectors:

1. P-Squared Detector: $a = median(D)$, where D is the set of absolute differences between the field value of this boxel and the field values of its pythagorean partners. This value falls in the range $0 < a < (\max_{i,j}\{v[i,j]\} - \min_{i,j}\{v[i,j]\})$.

2. Mode Concentration Detector: $a = \left(\frac{|R|-N_c}{|R|}\right)$ where $|R|$ is the number of entries in the R-list, and N_c is the number of entries that end up in the same cluster as the field value for the current boxel. Note that $0 \leq a < 1$, where $a = 0$ for a perfectly symmetric R-list and a is close to 1 for an asymmetric R-list with uniformly distributed field values.

3. Outlier Detector: $a = \left(\frac{v-\mu}{\sigma}\right)$ where v is the field value for the boxel and μ and σ are the trimmed mean and standard deviation. This asymmetry measure satisfies $a \geq 0$ and can be interpreted as the number of standard deviations from the mean.

The maximum value of a at each time step for each of the three detectors is shown in Figures 13-15. The parameters for these detectors where set as follows,

- P-Squared Detector: no parameter required
- Mode Concentration Detector: $Binsize = 10^{-10}$
- Outlier Detector: $\tau = 0.4$ (trim fraction)

Note that asymmetry was detected at the same time step by all three methods, and remains at an easily detectable level for the remainder of the simulation. It is no surprise that the initial detections occurred at the precise time step when the puck first contacts the plate. The maximum asymmetry value for the P-squared detector in Figure 13 exhibits a curious behavior in that it dips dramatically for a few time steps after its initial rise. This behavior is absent in the plots for the other two methods, and was not attainable by simply changing the parameters of these methods (e.g. $BinSize$ and τ). This curious behavior is currently unexplained, and worthy of further investigation. The plot for the Mode Concentration detector in Figure 14

shows a maximum asymmetry that remains very close to 1 once the asymmetry is introduced. This suggests that the distribution of field values for the maximum asymmetry are roughly uniform (i.e. the asymmetry is probably not due to a single (small) isolated phenomenon). The plot for the Outlier detector in Figure 15 shows extremely large asymmetry values, i.e. it has field values that are several hundred standard deviations from their mean. This is due to the fact that some of the standard deviations are very small (near zero), which is more likely when the trim factor is large (i.e. close to 0.5).

The maximum asymmetry values provides a narrow view of the simulation in that this asymmetry value may not be representative of the rest of the simulation. In Figure 16 we show the distribution of asymmetry values verses time for the P-squared method. This figure tells us that while the maximum is on the order of 10^1, most of the asymmetries fall in the range $[10^{-4}, 10^{-2}]$.

To obtain a global *size* statistic we run the three asymmetry detection algorithms and count the number of detected boxels. The parameters for these algorithms were set as follows,

- P-Squared Detector: $AsymThold = 10^{-10}$
- Mode Concentration Detector: $Binsize = 10^{-10}$
- Outlier Detector: $\tau = 0.4$ (trim fraction), $AsymThold = 10^{-10}$

Figures 17-19 show the fraction of boxels that are detected at each time step. These plots show that the size of the asymmetry grows monotonically with time, and ends up occupying over 1% of the simulation boxels. It is interesting that such a large asymmetry can result from a very small error in the setup code. The plots for the P-squared and Mode Concentration detector are virtually identical, but the Outlier detector appears to be missing some detections picked up by the other two. This can be explained in part by the fact that asymmetric boxels will go undetected along any asymmetric arc where the trimmed mean is equal to one or more of the field values. That is, the outlier detector will often miss detections when the mean value along an arc corresponds to an asymmetric point.

Finally, to get a better sense of how the three detection methods compare we view their asymmetry values from one slice of the rectangular cube at several time steps from the simulation. Each slice from each time step is displayed as an image, and the asymmetry values from each slice are scaled so that the relevant features are visible. The slice we have chosen is $z = 100$, which is (roughly) in the middle of the plate. The time steps we have chosen correspond to cycles 52, 65 and 96. Cycle 52 is the first cycle where an asymmetry appears in this slice. Cycle 65 is a few time steps later, and 96 is towards the end of the simulation. The results are shown for Cycle 52 in Figures 20-25, Cycle 65 in Figures 26-31, and Cycle 96 in Figures 32-37. Each set of results shows 6 images. The first two are the density field at two different scales; the first is the scale at which the simulation would be viewed when displayed as a movie (i.e. it is scaled so that the density values fall in the range 0-255 over the course of the simulation), and the second is a scale (and offset) that is chosen to amplify the differences at the current time step so that the effect of the puck on the plate can clearly be seen. The third and fourth images show asymmetry values for the Outlier method and the Mode Concentration method, each scaled so that detections can be seen (note that the Mode Concentration values

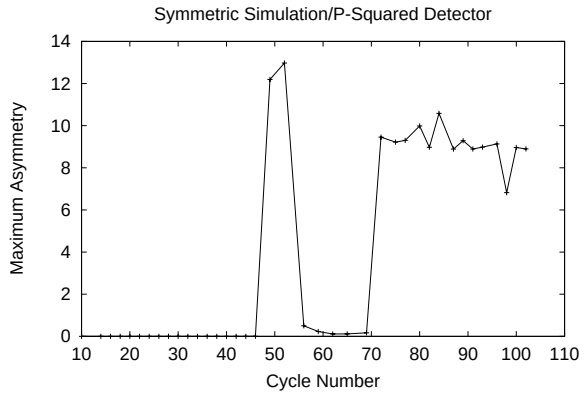

Figure 13: Maximum P-Squared Asymmetry vs Time for Symmetric Simulation.

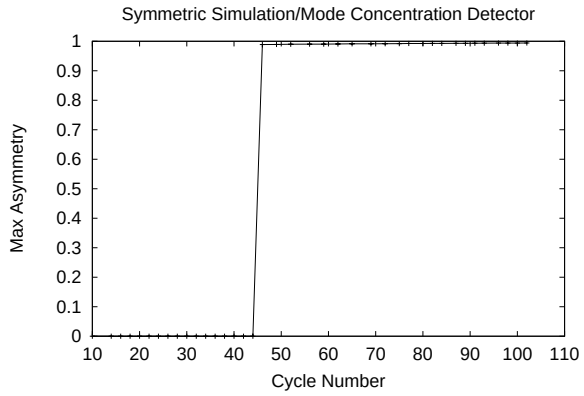

Figure 14: Maximum Mode Concentration Asymmetry vs Time for Symmetric Simulation.

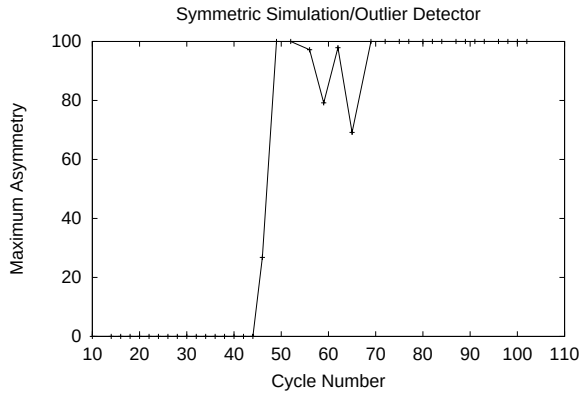

Figure 15: Maximum Outlier Asymmetry vs Time for Symmetric Simulation.

are always scaled by 25, so their image values fall in the range $[0, 250]$). The last two images show the asymmetry values for the P-squared method; the first is shown at the same scale and

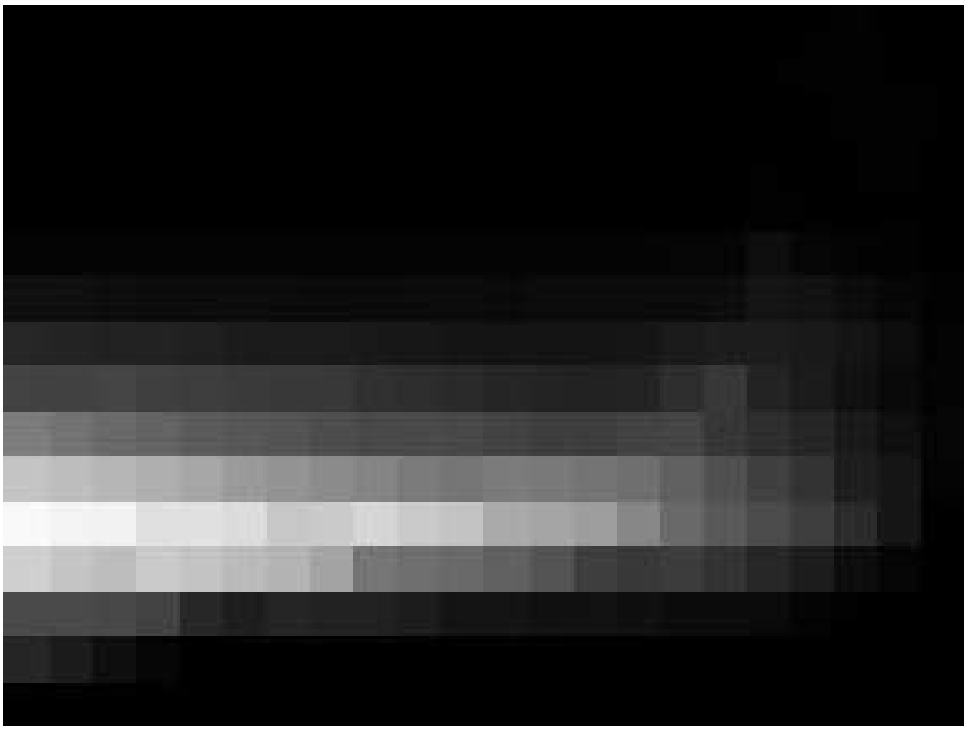

Figure 16: Asymmetry Scale Distribution vs Time for the P-squared method on the Symmetric Simulation. The horizontal axis represents time, where simulation time increases from left to right. The vertical axis represents bins into which the asymmetries at different scales are placed. More specifically it is indexed from top to bottom by $i = -14, -13, ..., 1, 0, 1$ where the scale of the asymmetry is 10^i. The intensity of the image represents the relative number of asymmetry values that occur at a given scale and time step.

offset as the first density image (i.e. the scale that would be used when viewing these values as a movie), and the second is scaled so that the detections can be seen.

Cycle 52 is the first time step at which the puck penetrates this slice, and although its effect on the density field cannot be seen at a normal scale, the half circle corresponding to the indentation made by the puck is clearly present when the density is properly scaled. The Outlier detector does not detect an asymmetry in this slice at this time step, but the other two detectors make it clear that an asymmetry is present. The P-squared values reveal three regions where the asymmetry is largest. The region at the tip of the half circle corresponds to the position on the puck where the asymmetry was introduced by the setup code. The two regions near the edge are (probably) false detections that occur because the asymmetry at the tip falls squarely on one of the folds (recall that an asymmetry that falls directly on one of the fold angles can lead to false detections as discussed earlier). The Mode Concentration method is very sensitive in this simulation (since the bin size is so small), and its output suggests that the asymmetries have spread across a significant range of radii. Although some of these asymmetries may be very small, this detector does not distinguish small from the large (its output is a function of the field value *distribution* along an arc, not the scale of these values). The asymmetries detected by the Mode Concentration detector are also present in the

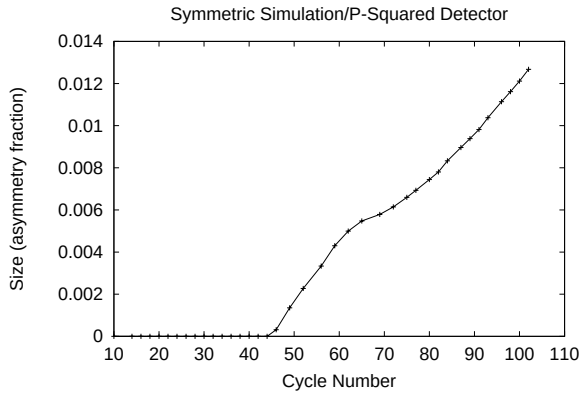

Figure 17: P-squared Asymmetry Size vs Time for Symmetric Simulation.

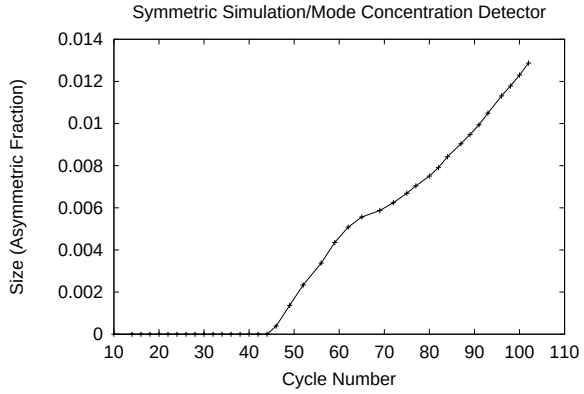

Figure 18: Mode Concentration Asymmetry Size vs Time for Symmetric Simulation.

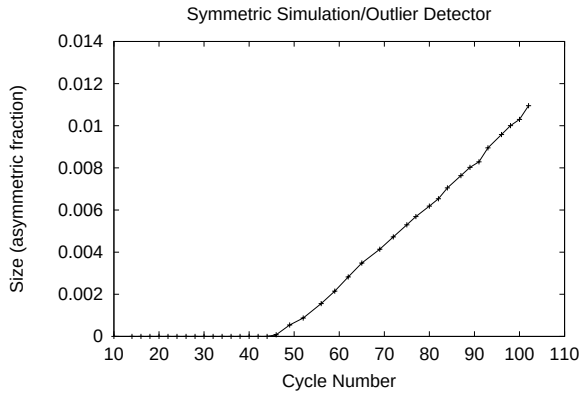

Figure 19: Outlier Detector Asymmetry Size vs Time for Symmetric Simulation.

P-squared output, but cannot be seen at this scale.

At Cycle 65 (Figures 26-31) the puck has penetrated the plate to the point where its effect

on the density can be seen at a normal scale. Asymmetries are now visible in all three methods. Again, the Mode Concentration method clearly reveals the full extent of asymmetry *locations*, while the P-squared method gives a better feel for the *scale* of the asymmetries and how they are spatially distributed. The results from the Outlier method are harder to interpret, but it has clearly detected asymmetries in the same region as the other two.

The results at Cycle 96 (Figures 32-37) are qualitatively similar to those at Cycle 65. The density pattern is more varied by this point, and the asymmetry locations have continued to spread. The results from the P-squared detector suggest that the largest asymmetries remain in the neighborhood of the arc where they originated.

Figure 20: Symmetric Simulation: Density at Cycle 0052 (offset 0,scale 25).

Figure 21: Symmetric Simulation: Density at Cycle 0052 (offset -8.93,scale 100000).

Figure 22: Outlier value at Cycle 0052 (offset 0,scale 4000000,τ=.4).

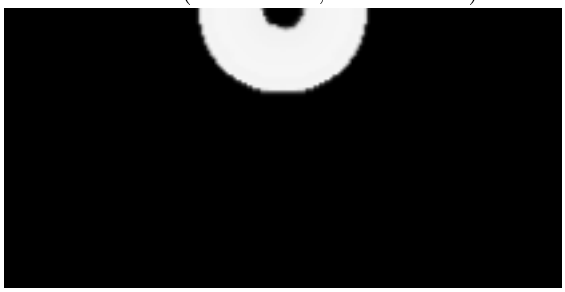

Figure 23: Mode Concentration value at Cycle 0052 (offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 24: P-squared value at Cycle 0052 (offset 0,scale 25).

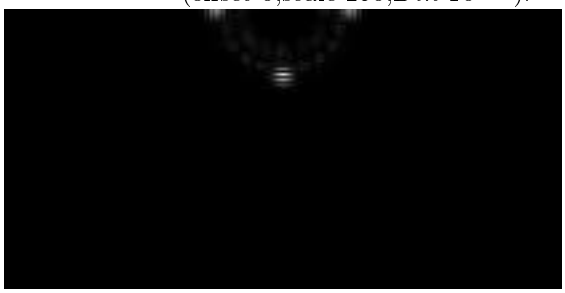

Figure 25: P-squared value at Cycle 0052 (offset 0,scale 4000000).

Figure 26: Symmetric Simulation: Density at Cycle 0065 (offset 0,scale 23).

Figure 27: Symmetric Simulation: Density at Cycle 0065 (offset -8.93,scale 150).

Figure 28: Outlier value at Cycle 0065 (offset 0,scale 15,$\tau = .4$).

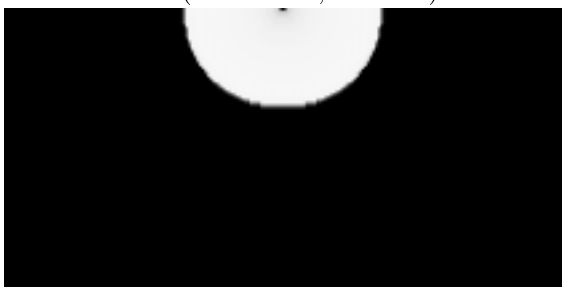

Figure 29: Mode Concentration value at Cycle 0065 (offset 0,scale 250,$Bin\ 10_{-10}$).

Figure 30: P-squared value at Cycle 0065 (offset 0,scale 25).

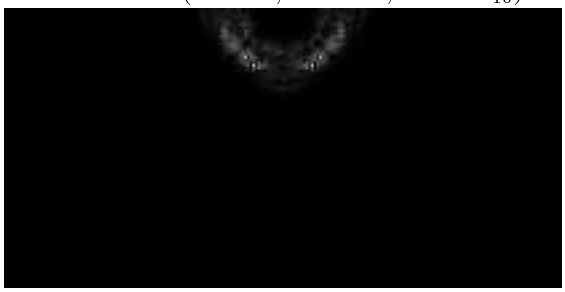

Figure 31: P-squared value at Cycle 0065 (offset 0,scale 40000).

Now we present results for the asymmetric simulation. The maximum asymmetry values verses time for the three detectors are shown in Figures 38-40. Once again we see that all three methods detected the asymmetry as soon as the puck hit the plate, and that the asymmetry remains at an easily detectable level for the remainder of the simulation. Note that we do not see the dip in the maximum asymmetry value for the P-squared detector that we saw in the previous simulation.

Figure 41 shows the distribution of asymmetry values verses time for the P-squared method. The asymmetries are more tightly concentrated than the previous simulation, and most fall in the range $[10^{-2}, 10^{0}]$, which is roughly two orders of magnitude larger than the previous

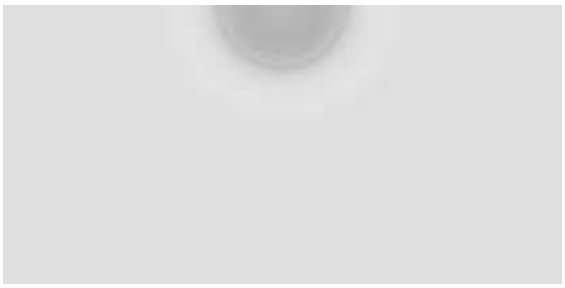

Figure 32: Symmetric Simulation: Density at Cycle 0096 (offset 0,scale 25).

Figure 33: Symmetric Simulation: Density at Cycle 0096 (offset -8.00,scale 200).

Figure 34: Outlier value at Cycle 0096 (offset 0,scale 5,$\tau = .4$).

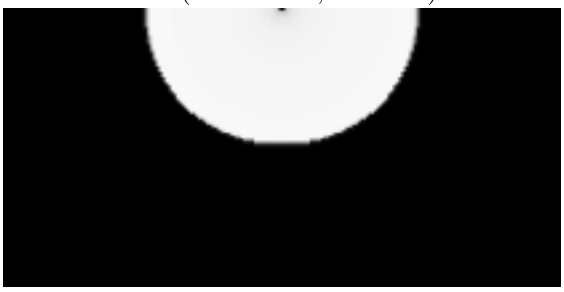

Figure 35: Mode Concentration value at Cycle 0096 (offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 36: P-squared value at Cycle 0096 (offset 0,scale 25).

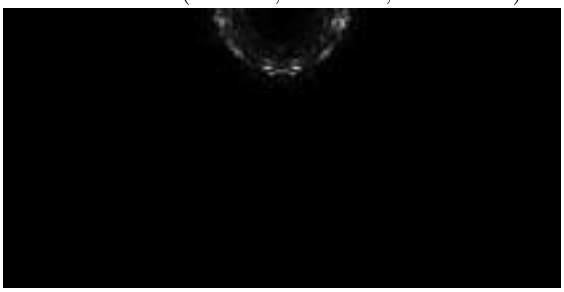

Figure 37: P-squared value at Cycle 0096 (offset 0,scale 1200).

simulation.

The global *size* statistic is shown in Figures 42-44. Although the shape of these curves is the same for all three detectors, the outlier detector once again appears to be missing some of the boxels detected by the other two methods. Note also that the shape of these curves bears a striking resemblance to those for the previous simulation, but the actual size is roughly 4 times as large as the previous simulation.

Finally, the asymmetry values from slice $z = 100$ at cycles 48, 51, 61 and 120 are shown in Figure sets 45-50, 51-56, 57-62, and 63-68 respectively. Cycle 48 is the first cycle where the

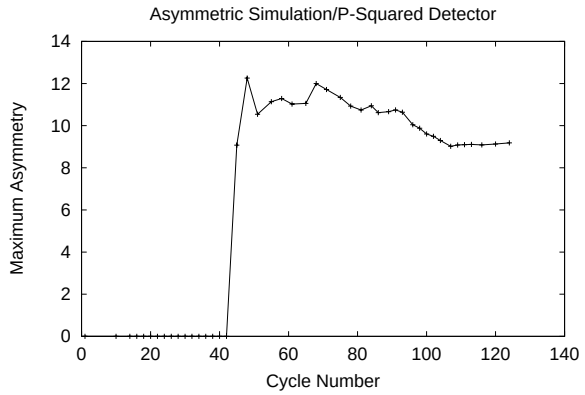

Figure 38: Maximum P-squared Asymmetry vs Time for Asymmetric Simulation.

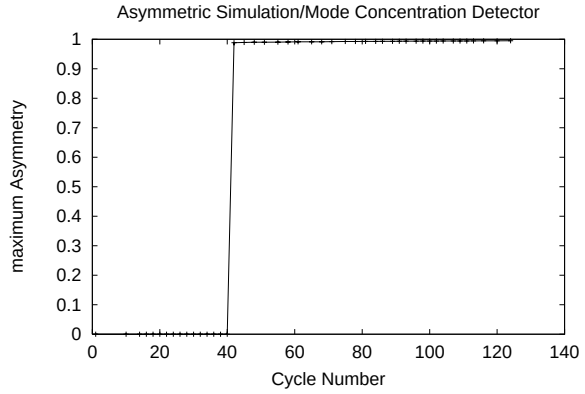

Figure 39: Maximum Mode Concentration Asymmetry vs Time for Asymmetric Simulation.

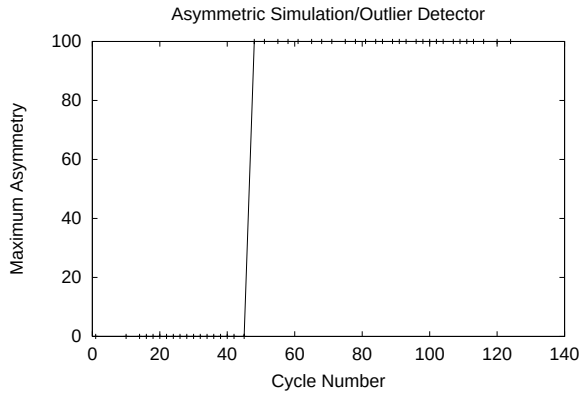

Figure 40: Maximum Outlier Asymmetry vs Time for Asymmetric Simulation.

puck's penetration affects the density values in this slice, and it is obvious that the puck is tilted so the the right tip makes first contact. The asymmetry introduced by this penetration is

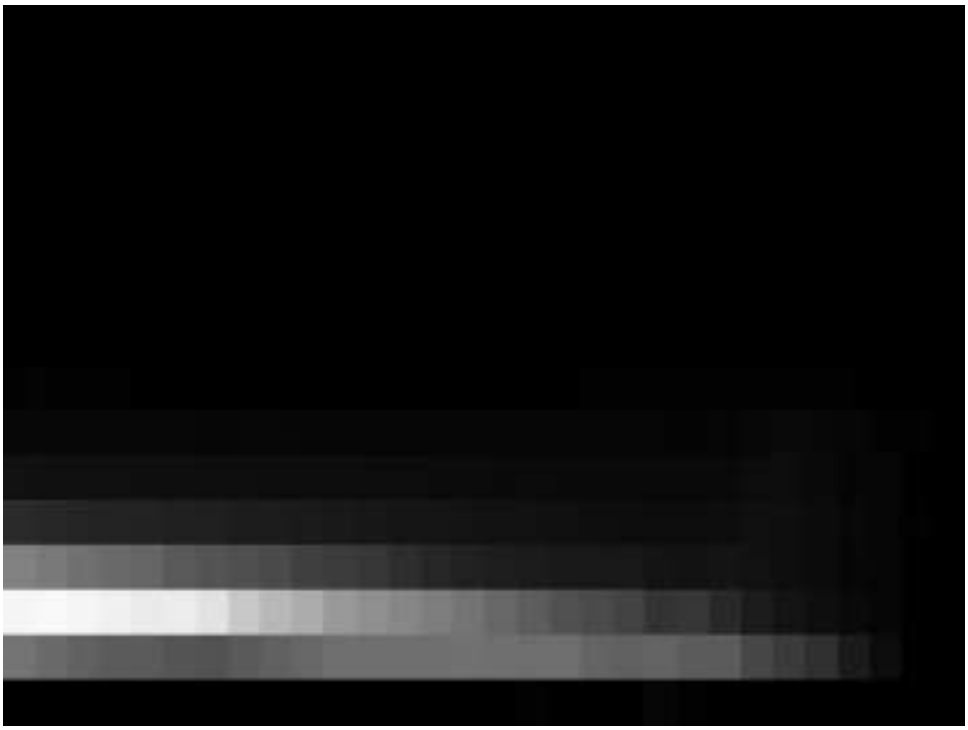

Figure 41: Asymmetry Scale Distribution vs Time for the P-squared method on the Asymmetric Simulation. The horizontal axis represents time, where simulation time increases from left to right. The vertical axis represents bins into which the asymmetries at different scales are placed. More specifically it is indexed from top to bottom by $i = -14, -13, ..., 1, 0, 1$ where the scale of the asymmetry is 10^i. The intensity of the image represents the relative number of asymmetry values that occur at a given scale and time step.

small, but easily detected by the Mode Concentration and P-squared methods. Note that the Mode Concentration method produces small (but non-zero) values at *symmetric positions* along the asymmetric arcs. These values are all less than 0.5, and would not be labeled asymmetric by this method. They are non-zero simply because the asymmetries at one location along the arc make the quantity $a = \left(\frac{|R|-N_c}{|R|}\right)$ non-zero at the other locations. Three time steps later, at Cycle 51, the asymmetry has grown, but is still undetected by the Outlier method. At this point the Mode Concentration values suggest that the asymmetry has grown in size to the point where it occupies nearly half of the positions along the arcs. So (based on our definition) these arcs will very soon become entirely asymmetric. This is made clear by the Mode Concentration values at Cycle 61. The density images at Cycle 61 also make this clear. In addition, the Outlier method has detected asymmetries at Cycle 61, with the largest values falling along the right side of the arcs located near the radius where the tip of the puck first contacted the plate. The P-squared values present a different picture, and one that is more closely aligned with the Mode Concentration method. The pattern displayed by the P-squared method at this cycle is somewhat curious. This last set of images are for Cycle 120, which is near the end of this simulation. The density images suggest that the puck has punched all the way through this slice, leaving a hole in the middle with zero density. No signs of

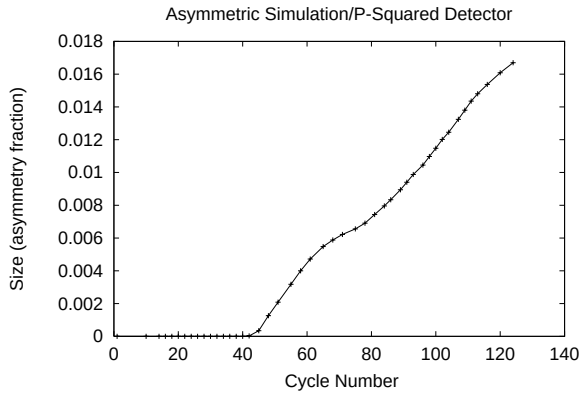

Figure 42: P-squared Asymmetry Size vs Time for Asymmetric Simulation.

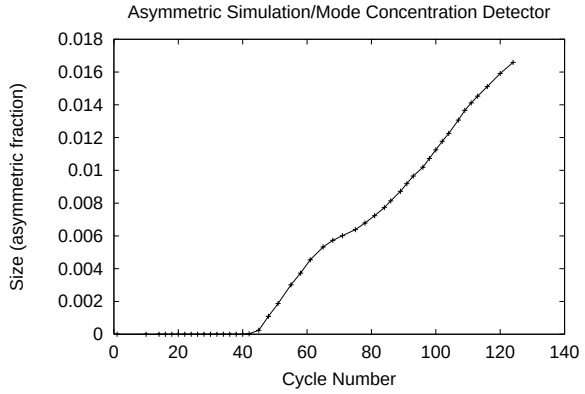

Figure 43: Mode Concentration Asymmetry Size vs Time for Asymmetric Simulation.

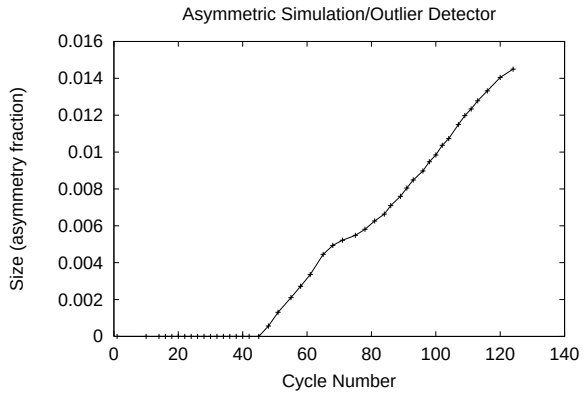

Figure 44: Outlier Asymmetry Size vs Time for Asymmetric Simulation.

asymmetry are present from any of the detectors in this middle region. Once again the Mode Concentration method clearly identifies all regions of asymmetry, and the P-squared method

reveals their scale. The Outlier method provides results similar to the P-squared method, but has larger values near the boundaries. This may be due to the aggregation process that is used to form the R-lists.

Figure 45: Asymmetric Simulation: Density at Cycle 0048 (offset 0,scale 25).

Figure 46: Asymmetric Simulation: Density at Cycle 0048 (offset -8.93,scale 2000000).

Figure 47: Outlier value at Cycle 0048 (offset 0,scale 2000000,$\tau = .4$).

Figure 48: Mode Concentration value at Cycle 0048 (offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 49: P-squared value at Cycle 0048 (offset 0,scale 25).

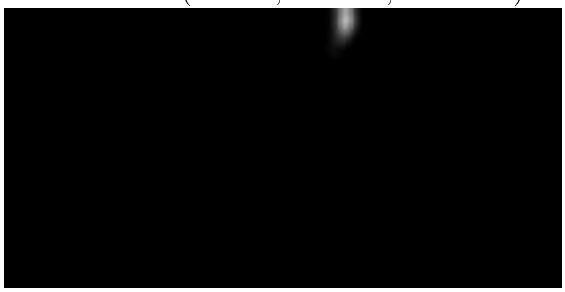

Figure 50: P-squared value at Cycle 0048 (offset 0,scale 2000000).

In summary, all three methods demonstrated an ability to detect asymmetries, although it appears that the Outlier method may often miss asymmetries detected by the other two. All three methods detected asymmetries that appear at such a small scale (at least initially) that they would go undetected by a simple visual inspection (i.e. a movie). In the first simulation (the "symmetric" simulation) the asymmetries remained at a scale that was small enough that they may never have been detected by visual inspection. The Mode Concentration method is good at locating asymmetries, but provides little information about their scale. On the other hand, the P-squared method is good at both location and scale.

Figure 51: Asymmetric Simulation: Density at Cycle 0051
(offset 0,scale 25).

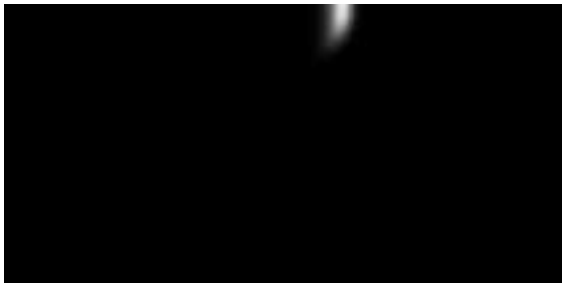

Figure 52: Asymmetric Simulation: Density at Cycle 0051
(offset -8.93,scale 9000).

Figure 53: Outlier value at Cycle 0051
(offset 0,scale 10000,$\tau = .4$).

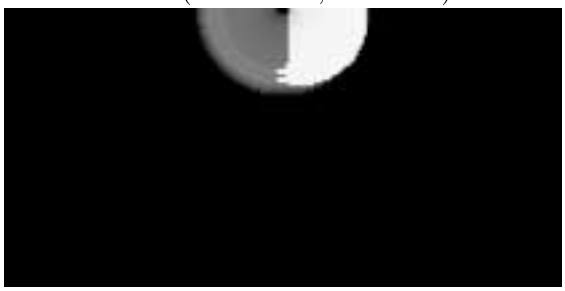

Figure 54: Mode Concentration value at Cycle 0051
(offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 55: P-squared value at Cycle 0051
(offset 0,scale 25).

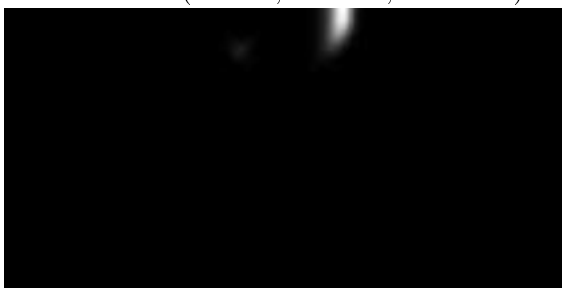

Figure 56: P-squared value at Cycle 0051
(offset 0,scale 10000).

Figure 57: Asymmetric Simulation: Density at Cycle 0061
(offset 0,scale 25).

Figure 58: Asymmetric Simulation: Density at Cycle 0061
(offset -8.93,scale 150) .

Figure 59: Outlier value at Cycle 0061
(offset 0,scale 10,$\tau = .4$).

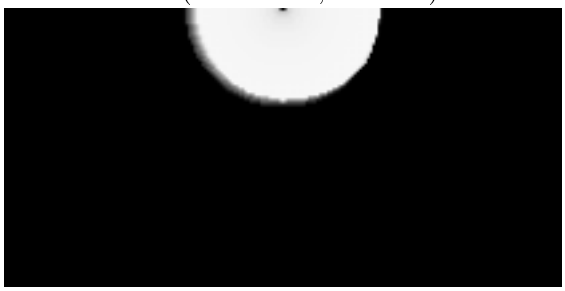

Figure 60: Mode Concentration value at Cycle 0061
(offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 61: P-squared value at Cycle 0061
(offset 0,scale 25).

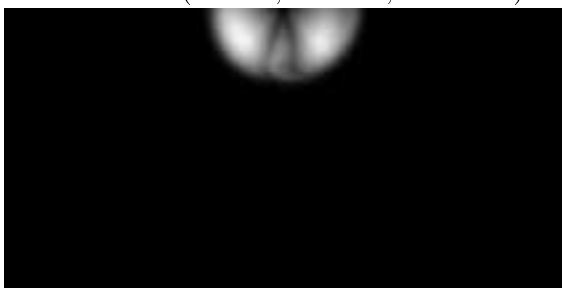

Figure 62: P-squared value at Cycle 0061
(offset 0,scale 200).

Figure 63: Asymmetric Simulation: Density at Cycle 0120
(offset 0,scale 25).

Figure 64: Asymmetric Simulation: Density at Cycle 0120
(offset 0,scale 25).

Figure 65: Outlier value at Cycle 0120
(offset 0,scale 10,$\tau = .4$).

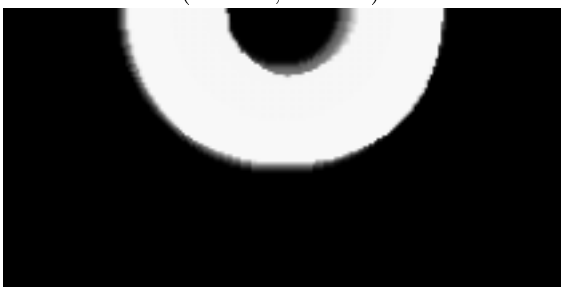

Figure 66: Mode Concentration value at Cycle 0120
(offset 0,scale 250,$Bin\ 10^{-10}$).

Figure 67: P-squared value at Cycle 0120
(offset 0,scale 25).

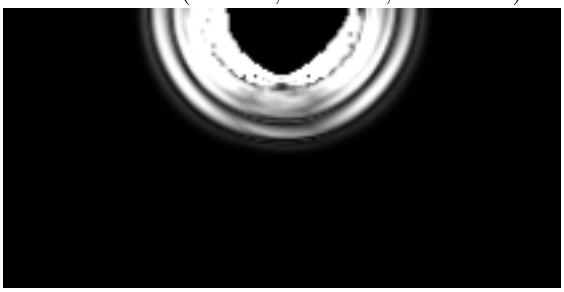

Figure 68: P-squared value at Cycle 0120
(offset 0,scale 2500).