LA-UR-16-27206
Approved for public release; distribution is unlimited.

Title: DSD2D-FLS 2010: Bdzil's 2010 DSD Code Base; Computing tb and Dn with Edits to Reduce the Noise in the Dn Field Near HE Boundaries

Author(s): Bdzil, John Bohdan

Intended for: Report

Issued: 2016-09-21

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

DSD2D-FLS 2010: Bdzil's 2010 DSD Code Base
Computing tb and Dn with Edits to
Reduce the Noise in the Dn Field Near HE Boundaries

J. Bdzil

October 9, 2014
December 19, 2014 – with updated figures & text

HISTORY

The full level-set function code, DSD3D, is fully described in LA-14336 (2007) [1]. This ASCI-supported, DSD code project was the last such LANL DSD code project that I was involved with before my retirement in 2007. My part in the project was to design and build the core DSD3D solver, which was to include a robust DSD boundary condition treatment. A robust boundary condition treatment was required, since for an important local "customer," the only description of the explosives' boundary was through volume fraction data. Given this requirement, the accuracy issues I had encountered with our "fast-tube," narrowband, DSD2D solver, and the difficulty we had building an efficient MPI-parallel version of the narrowband DSD2D, I decided DSD3D should be built as a full level-set function code, using a totally local DSD boundary condition algorithm for the level-set function, phi, which did not rely on the gradient of the level-set function being one, |grad(phi)| = 1. The narrowband DSD2D solver was built on the assumption that |grad(phi)| could be driven to one, and near the boundaries of the explosive this condition was not being satisfied. Since the narrowband is typically no more than10*dx wide, narrowband methods are discrete methods with a fixed, non-resolvable error, where the error is related to the thickness of the band: the narrower the band the larger the errors. Such a solution represents a discrete approximation to the true solution and does not limit to the solution of the underlying PDEs under grid resolution.

Given a material interface description only as detailed as that associated with the volume fraction of the materials in the computational cells, there can be no unique sub-cell definition of the location of the material interface. Working under these constraints, I built a DSD3D solver whose solution errors in the burn time field, tb, diminished as O(d(dx)^2) for boundary-free problems and whose solution error diminished as O((dx)) for problems which included an explosive's material boundaries. These convergence rates were verified by comparing the DSD3D solver's numerical solutions with exact solutions. Unlike the previous three-dimensional DSD solvers, DSD3D was built using a totally local boundary condition algorithm that required no region-wide, iterative boundary condition updates and that should have been no more difficult to "parallelize" via an MPI, domain-decomposition strategy than would be a heat-equation solver. I delivered the serial, full level-set function DSD3D solver, DSD3D-FLS, to XCP-4 (John Walter) in mid-2007.

After retiring from LANL, in 2008 I began working with the Department of Mechanical Sciences and Engineering at The University of Illinois (UIUC). Work continued at LANL on developing at MPI parallel version of DSD3D-FLS, with little to no serious progress being reported through 2009. At that time, I happened to be working with a UIUC student who was interested in scientific computing, and numerical algorithms for the solution of partial-differential equations (PDEs). Together, the student and I decided that developing and implementing an MPI, domain-decomposition solution strategy for the core DSD3D-FLS solver would be a good Masters degree thesis problem. So in early 2010, I took my standalone, one explosive, DSD3D-FLS solver and built the basic DSD2D-FLS solver, whose properties will be explored later in these notes. Together, the student and I developed a strategy and built an MPI, domain-decomposition model with which to "parallelize" DSD2D-FLS. It is worth noting that the student successfully completed this project, which then led to his being awarded a MS in Engineering from UIUC by the late fall of 2010. That work, including the basic DSD2D-FLS solution algorithm, is described in our publication on this work [2].

The single explosive, serial version of the 2010 DSD2D-FLS solver represents the code base whose properties I explore here. Later in 2010, I passed the DSD2D-FLS solver, including updates to include F90 and a multiple explosive and inert material region capabilities, to John Walter of LANL group XCP-4. John Walter then installed this DSD2D-FLS solver into the CASH-based LANL DSD solver library. At some point in early 2014, the responsibility for the DSD solver library was passed to James Quirk, also of XCP-4. Since that time, James has wrapped the DSD solver library with hooks to his Amrita environment. It will be comparisons of results from this XCP-4 twice wrapped DSD2D-FLS solver library and my 2010, DSD2D-FLS code base results that I will explore in these notes.

THE PRESENT

Late during the summer of 2014, I received questions from Mark Short, LANL WX-9 and ASCI-HE program lead, and James Quirk about some of the DSD solutions James was seeing with the CASH/Amrita wrapped, serial DSD2D-FLS solver. Before we look at these questions, I need to set down the solutions properties that we expect to see coming from the DSD2D-FLS solver.

First, I will set down some preliminaries. For problems where DSD boundary conditions are applied, the convergence properties of the 2010 code base DSD2D-FLS generated burn time field, tb, have been well-established, and show that the errors in tb diminish in an O(dx) fashion as the mesh size, dx, is decreased. Importantly, the numerical solutions do converge to the solutions of the PDEs as the mesh size is reduced. This does require that the material interface description itself be resolvable under resolution. A detailed convergence study was presented in [2]. For a number of simple geometries, here I show detailed comparisons of the TOA (time of arrival or burn table) field generated with DSD2D-FLS compared to the "exact" TOA field generated with a Maple script.

Since the normal detonation speed, dn, is directly related to tb through the relation, dn = 1/abs(grad(phi)), and given that tb comes with O(dx) errors, then clearly dn will have O(1) errors. To have a situation better than this would require that we have an algorithm for tb with errors no larger than O((dx)^2). To my knowledge, there are no such DSD boundary algorithms out there today. In addition, this would require an interface description with errors of O((dx)^2). So given that we will have O(1) errors in dn, then we need to address the questions: are these O(1) errors 0.1%, 1%, 10%, 100%, 1000%, etc., where are these errors located and what properties of the numerical solution algorithm controls these errors?

In late August to early September, 2014 I received some results and questions about the solutions James was seeing for detonation propagation in an explosive rate stick, when he solved the problems with his CASH/Amrita wrapped version of DSD2D-FLS. James Quirk's results are displayed in Figures-1), and the problem geometry is displayed in Figure-2). I believe these simulations used Dn = 1 – 0.1*kappa and were performed in 2D, slab geometry. I do not have information about either omega_s or omega_c, nor for any of the numerical parameters, such as dx, cfl, etc., which James used in his simulations. I've not seen errors in dn as large as those displayed in Figure-1), particularly on the upside, in any simple rate stick simulation that I've performed and in none of the simulations I report on here.

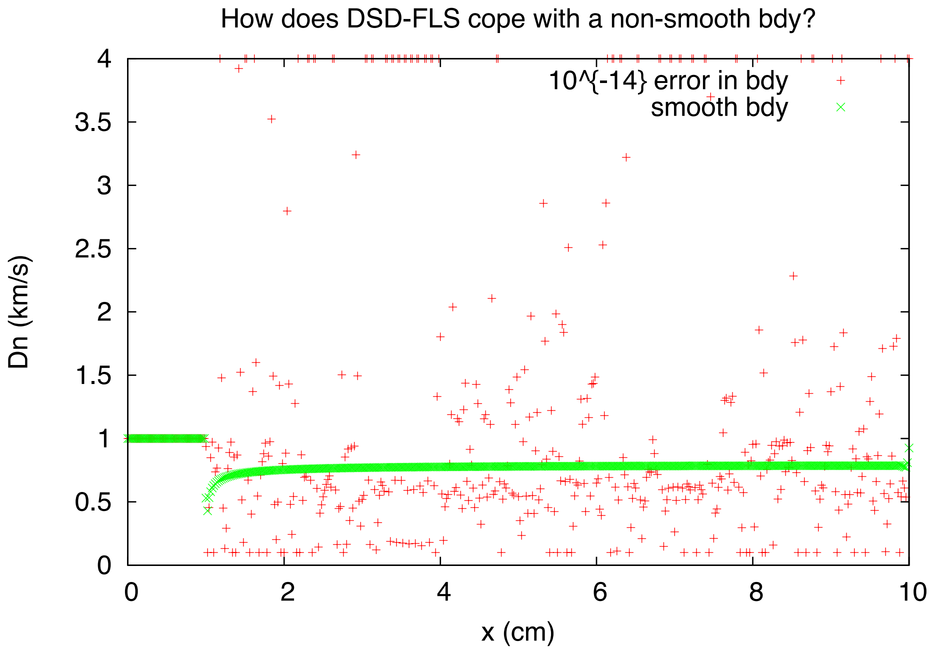

Figure-1) A comparison of the normal detonation speed, dn, measured along the upper horizontal boundary, for the rate stick geometry displayed in Figure-2. The smooth green curve is for the case where the explosive boundary is displaced from a mesh line by some fraction of the mesh spacing, dx. The red pluses are from a DSD simulation where the nominal upper explosive boundary is aligned with a horizontal mesh line, and an error is added to the geometry defining function, psi(i,j). The errors in dn for the mesh aligned simulation can be greater than 400%. These errors are much greater than anything I have seen in such a calculation. It is my understanding that for these simulations, the CFL parameter was set to a very-small value. The above results come from a simulation performed by Quirk.

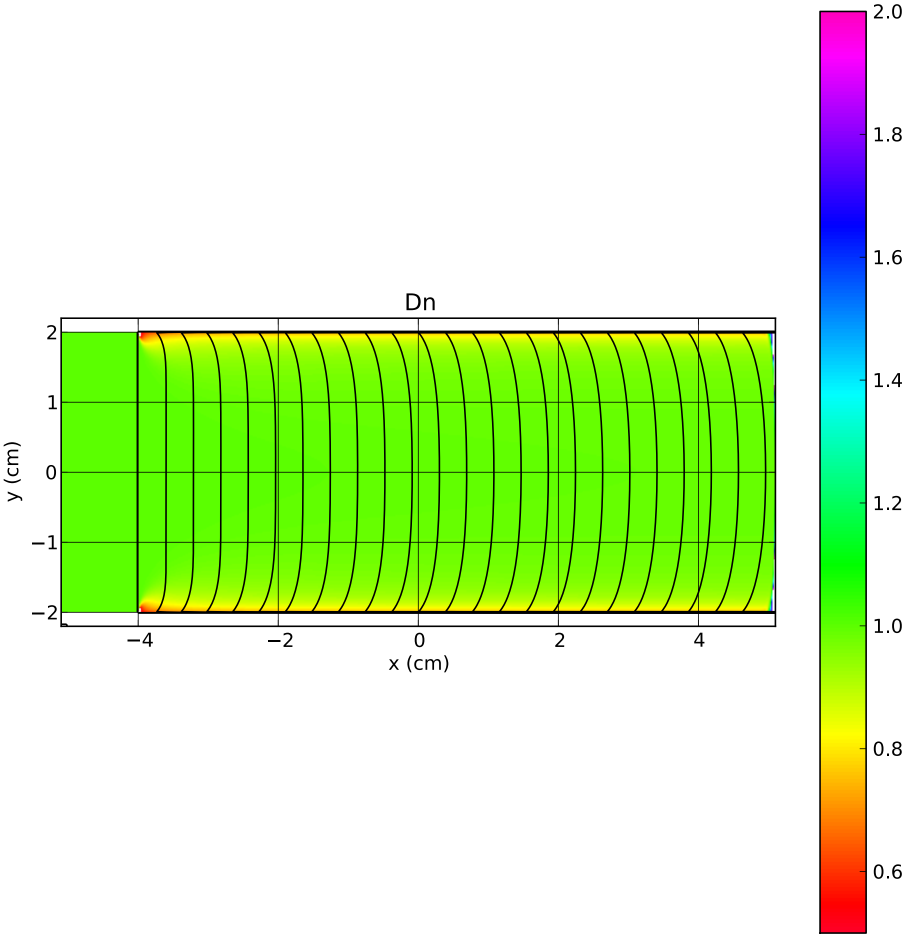

Figure-2) DSD simulation of detonation for a 2D, slab-geometry rate stick. Detonation is initiated at t = 0 along a vertical line (located at x = -4) and then proceeds to the right. The top-to-bottom curved lines are contours of the tb-field. The color palette is coordinated with the values of dn. The values of dn displayed in Figure-1) are measured along the horizontal, upper boundary of the explosive region. For this simulation, the horizontal explosive boundaries are displaced from mesh lines by a fraction of dx. The above results come from a simulation performed by Quirk.

Given the lack of specific information about the physical/numerical parameters for this rate stick problem, I've tried to guess reasonable parameters for the problem displayed in Figure-2) and then reproduce those results. My 2D, slab problem dimensions will be the same as those shown in Figure-2), which are -5 <= x <= 5 and -2 <= y <= 2, as the dimensions of the rate stick. In my simulations, I nest this rate stick in the computational domain -6 <= x <= 6 and -6 <= y <= 6. I use as DSD parameters dn = 1 – 0.1*kappa, dmin = 0.1, dmax = 9.0, omega_s = 50 degrees and omega_c = 55 degrees and select slab geometry, naxsym = 0. I use my standard set of numerical parameters, cfl = 0.9 and run with re-distancing on, where the re-distancing parameters are taken to be delta = 0.1, epsilon = 0.1. All my simulations are serial and were performed on a Macbook Pro running Mavericks-10.9.5 and Xcode-6.0.1 and using the gfortran-v4.8 compiler running mpif77 (under openMPI-1.4.5) with vanilla settings and no optimization. Two mesh sizes were used, nxpts = nypts = 600 points and nxpts = nypts = 601 points, which corresponds to the actual top boundary along the mesh line y = 2.00000 when nxpts = nypts = 600 and the numerical top boundary along the mesh line y = 1.98669 when nxpts = nypts = 601. Displayed in Figure 3a) is a plot showing dn along the numerical top boundary of the rate stick (y = 2.0 for nxpts = nypts = 600) and (y = 1.98669 for nxpts = nypts = 601). Of course, the physical top boundary is along y = 2.0. NOTE: Since the psi(i,j)= 0.0 line is exactly defined in my standalone, DSD2D-FLS code (that is, to within standard, single precision accuracy, psi(i,j) = 0.0 is along y = 2.0), then there is effectively no noise in my simple, algebraically defined psi(i,j)=0.0 function even for the case where the physical and numerical top boundaries are coincident. As is clear from Figure-3a), the two curves are smooth, noise free and appear nearly identical in the "eyeball norm."

The problem geometry displayed in Figure-2) is sufficiently simple to allow the DSD solution to be found by solving a 1D initial-value problem for the evolving shock shape as a function of the y-coordinate and time. Such a high-resolution solution of a 1D problem, developed using a Maple script, can essentially be considered to be an "exact" solution. We compare this "exact" solution (dashed curves) with the DSD2D-FLS solution (solid curves) in Figures 3b) & 3c), where the TOA-field is compared over the entire HE domain, and the Dn-field is compared along lines of constant-y, respectively. As displayed in Figure 3b), the DSD2D-FLS simulated TOA-field (computed at 0.02 resolution) and the "exact" Maple script solution overlay one another very well. As we have argued above and demonstrated in [2], the DSD2D-FLS algorithm yields a solution for TOA that is O(dx) accurate in the mesh spacing, dx, even for a HE interface description that has O(dx) errors. What the results in Figure-3c) show is that the dn-field, which can be obtained from dn = 1/abs(grad(phi)) and thus can be expected to have O(1) errors, shows rather small differences from the "exact" solution. As shown in Figure 3a), where the dn-field along the top boundary of the rate stick is displayed, the value of dn along the boundary and a fraction of dx below the boundary, has little effect on the computed result and where both results are essentially noise-free.

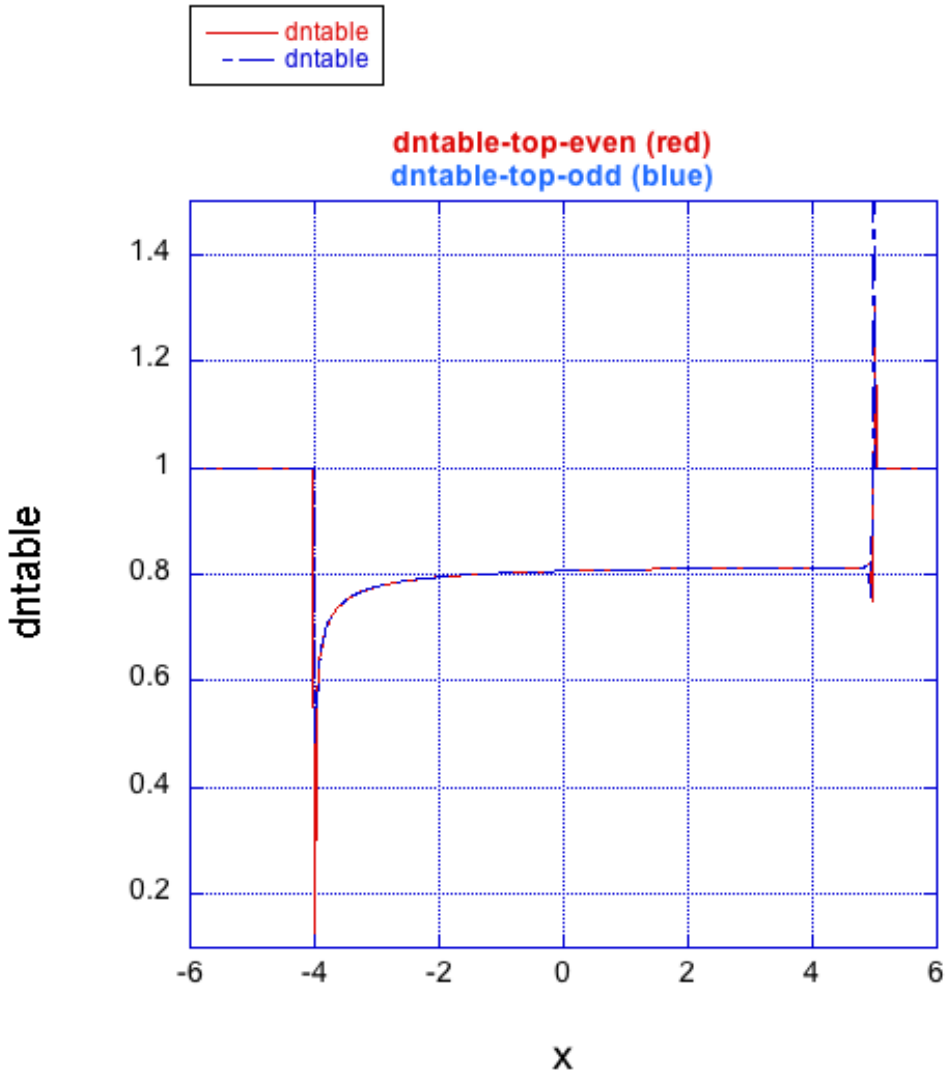

Figure-3a) dn along the physical, top horizontal boundary of the rate stick geometry displayed in Figure-2). The red curve corresponds to dn along y = 2.00000, which is the top boundary for the nxpts=nypts=600 points simulation, and the blue curve corresponds to dn along y = 1.98669, which is the top numerical boundary for the DSD2D-FLS simulation when nxpts=nypts=601 points are used. The initiating wave enters at x = -4, and the detonation exits the rate stick at x = 5. The value of dntable is computed using the curvature of the level-set function. The background value of dn stored in dntable is 1.0, and so that value is to be ignored. The value of dn = 0.1 at x = -4 is expected, while the undershoot and overshoot of dn at x = 5 is an artifact of the handover of control of the boundary condition from a horizontal, to a diagonal and then to a vertical ghost node in the corner of the explosive domain. This artifact is commonly observed in corner regions. Importantly, these solutions are noise free and smooth.

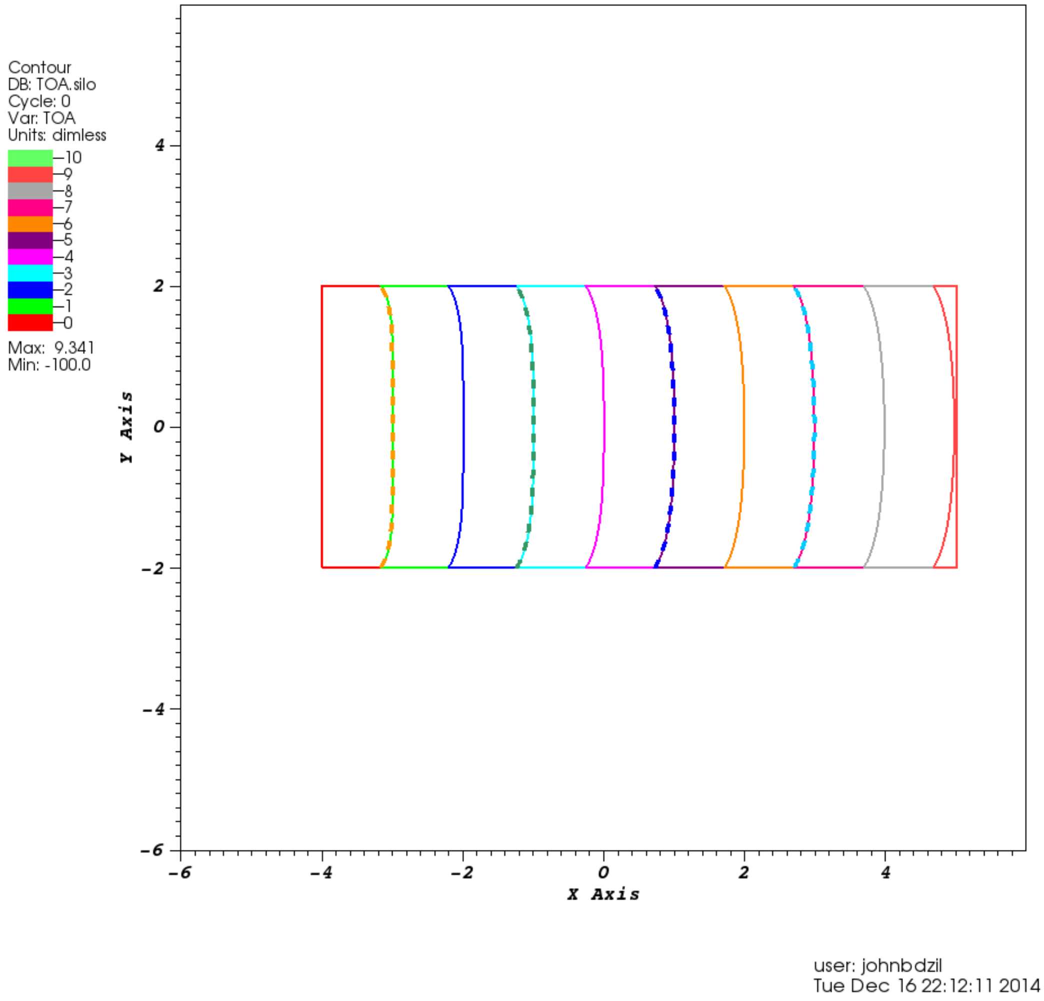

Figure-3b) My DSD2D-FLS calculation of the rate stick TOA-field for the problem displayed in Figure-2), and where nx = ny = 600. The DSD2D-FLS integrated solution curves are shown as solid, while the “exact” Maple script generated solution curves are shown as dashed. The solutions obtained with the two methods essentially overlay. In this simulation, the upper and lower boundaries of the rate stick lie along mesh lines.

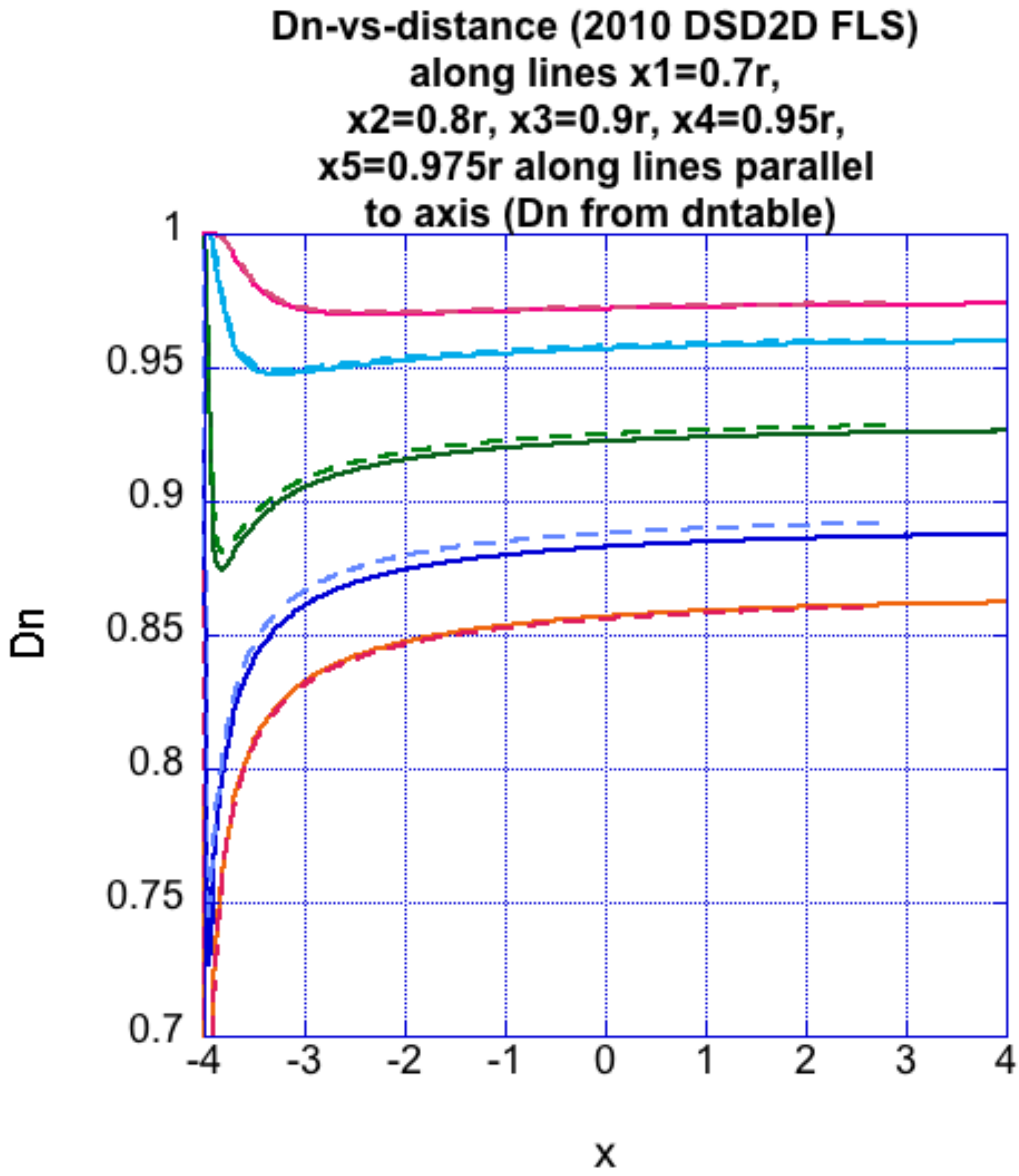

Figure-3c) My DSD2D-FLS solution for the dn-field (dntable), using the curvature of the level-set function, phi(i,j), to compute dn, are shown as the solid curves. These are compared with the "exact" Maple script generated solution, shown as dashed curves. The value of dn are displayed along five lines, which are displaced by a constant distant, y= 1.95 (lowest curve), 1.90, 1.80, 1.60 and 1.40, from the centerline of the rate stick (y = 0), for the problem shown in Figure-2). The agreement is good, with the differences between the DSD2D-FLS solution and the "exact" solution being attributable to finite resolution effects in the DSD2D-FLS solutions. The dynamics of the detonation are substantially different near the upper and lower boundaries than near the centerline of the rate stick.

Next, I study multiple cases including those where the psi(i,j) function defining the explosive interface and distance to the interface is exact and those where noise is added to the psi(i,j) distance function which defines the HE boundary geometry.

HOW NOISE ADDED TO PSI(i,j) AFFECTS THE COMPUTED DN(i,j)-FIELD

To both replicate and better understand the behavior displayed in Figure-1) and Figure-2), I performed a number of simulations where controlled noise was added to the exact psi(i,j) function that defines the rate stick geometry displayed in Figure-2). The noise I added had a sinusoidal distribution and was of the form

(1) A*sin(f*(i+j)) ,

and was added to the exact psi(i,j) function, to get

(2) psi(i,j) = psi(i,j) + A*sin(f*(i+j)) .

I ran cases where the frequency factor, f, had the values of f = 0.1 and 0.2, and the amplitudes were A = 1.0e-3, 1.0e-4, 1.0e-6 and 1.0e-7. Since the goal was not only to understand how the noise in the psi(i,j) function affects the location of psi(i,j) = 0.0 and how that passes into the simulated results but also to find a way of filtering the noise, I setup the noise filter (displayed below for the cutoff of 1.0e-7) for smearing the location of psi(i,j) = 0.0 to a band where psi(i,j) is zero, thus reducing the sensitivity of the simulation to noise in psi(i,j).

FILTER ADDED TO 2010 DSD2D-FLS driver.f, AFTER THE CALL TO setpsi.f

```
####################################################

c Set psi(i,j) = 0 if( abs(psi(i,j)).le.1.0e-7 ) psi(i,j) = 0.0

      do i = -2,nxpts+2
       x = xmin + (xmax-xmin)*real(i)/real(nxpts)
       do j = -2,nypts+2
        y = ymin + (ymax-ymin)*real(j)/real(nypts)
         if( (abs(psi(i,j)).le.1.0e-7) ) psi(i,j) = 0.0
       enddo
      enddo

#####################################################
```

To put the results on sensitivity to noise in the psi(i,j) function into some perspective, I first describe how explosive boundaries are defined in DSD2D-FLS. Explosive boundaries are defined similarly to how material boundaries are defined in fixed, Eulerian-grid, high-speed flow solvers that use the Cartesian/immersed boundary method to deal with material interfaces. Displayed in Figure-4) is a Cartesian grid on which one (interior) explosive region is shown, psi(i,j) <= 0.0, and two (exterior) ghost node regions, psi(i,j) > 0.0 are shown. The near-boundary points in the ghost node regions are used to set the DSD boundary conditions.

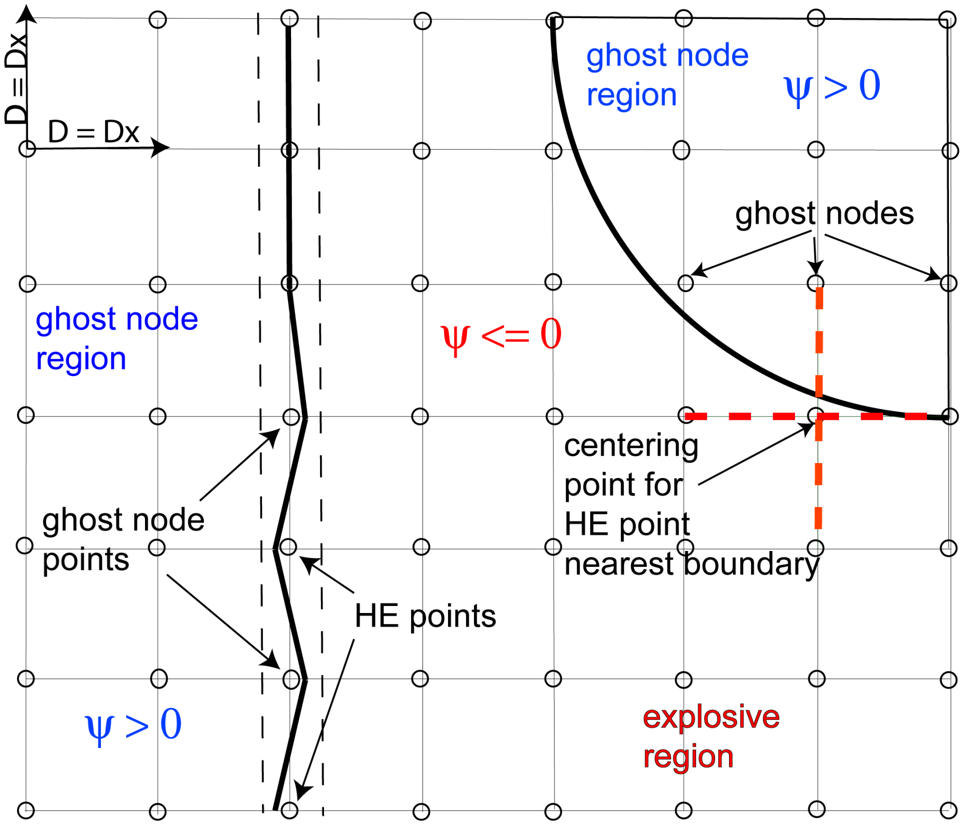

Figure-4) The Eulerian grid used in DSD2D-FLS simulations is displayed, showing the explosive (HE) region, psi(i,j) <= 0.0, and the ghost node regions, psi(i,j) > 0.0, which are used to set the DSD boundary conditions. A four-point central-difference stencil is used to compute gradients of both psi(i,j), the distance function to the explosive boundary, and phi(i,j), the level-set function in which the propagating detonation front is embedded. The DSD front curvature is computed with a nine-point stencil involving phi(i,j) and centered at (i,j). The heavy circular line in the upper right hand side of the figure and the heavy vertical line breaking into a zigzag line represent the HE interfaces, psi(i,j) = 0.0, where the zigzag section is meant to represent the effect of numerical noise on the definition of an HE interface that is coincident with a mesh line. The two dashed vertical lines denote the expanded region where psi(i,j) = 0.0, which is how I propose to eliminate the effect of numerical noise on mesh-line coincident HE interfaces.

In our first-order (in the mesh spacing, dx) boundary treatment, there is no sub-cell resolution of the HE material interface. Thus, points with psi(i,j) <= 0.0 are considered to be in the HE, and points with psi(i,j) > 0.0 are considered as outside the HE and are ghost node points. For the HE boundary represented by the section of a circular arc in the upper right hand side of Figure-4), noise added to the circular boundary would not represent any particular problem, since the definition of the circle in DSD2D-FLS lacks sub-cell resolution and thus comes with an O(dx) uncertainty in the definition of the boundary. With a noise-free psi(i,j) representation of the boundary, the circular arc would occasionally pass very near, but to either side of the boundary. So occasionally, an (i,j)-point would be just inside or outside the real HE material boundary, which would lead to O(dx) noise in the HE material interface. Such exceptional points represent no particular problem, and adding a low-level noise to the psi(i,j) function will have very little overall effect on how DSD2D-FLS interprets the circular boundary. However, for the case shown in Figure-4) where a vertical HE interface is aligned with a mesh line, adding noise (which is schematically represented by the zigzag section trailing off downwards) would have a significant effect. This would produce an HE interface with a zigzag shape on the O(dx) mesh scale and not the actual smooth vertical HE boundary. This roughened interface could then affect the DSD simulation results. This issue would only arise in the circumstance when either a truly vertical or horizontal HE boundary was coincident with a mesh line.

In the next few figures, I display some of my results for James Quirk's rate stick problem displayed in Figure-2), where I purposely add noise to the psi(i,j) function defining this rate stick. For these examples, I use the nxpts=nypts=600 mesh with a noisy psi(i,j) function given by

(3) psi(i,j) = psi(i,j) + 1.0e-7*sin(0.2*(i+j)) .

Here I'm adding a very small amplitude noise to psi(i,j). The advantage of this noise function is that the noise only occasionally moves the interface so that a mesh point moves from the HE to the ghost node regions. Displayed in Figure-5a) is a comparison of the DSD2D-FLS simulated TOA-field with the "exact" Maple script generated solution. Comparing the HE boundary displayed in Figure-5a) with the HE boundary in Figure-3b), reveals the O(dx) noise we are seeing in the DSD2D-FLS boundary location. The slight oscillation on the boundary location has essentially no effect on the comparison of the DSD2D-FLS TOA-field with the "exact" TOA-field.

Displayed in Figure-5b) are the DSD2D-FLS computed values of dn along y = 2.0 (red) and y = 1.98 (blue). The plotted points correspond to the simulation where the psi(i,j)-function has added noise, according to Eq. (3). The solid curves correspond to the results from the noise-free simulation. The noise in dn is substantial, although considerably less than what Quirk's results, displayed in Figure-1), show. The dn values from the simulation containing noise in psi(i,j), mostly cluster around the curves for the noise-free simulations. The plateaus consisting of points (and displayed in red) correspond to the default value of dn that is initialized into the ghost-node region. Shown in Figure-5c) is dn, computed with DSD2D-FLS for the noise containing psi(i,j)-field simulations, compared with the "exact" solutions for the noise-free problem, in both cases displayed along lines of constant-y as in Figure-3c). What these results show is that the noise in dn at the HE boundary can be substantial (although much less than that displayed if Figure-1)), but that the noise in dn decreases in amplitude the further one moves into the HE region.

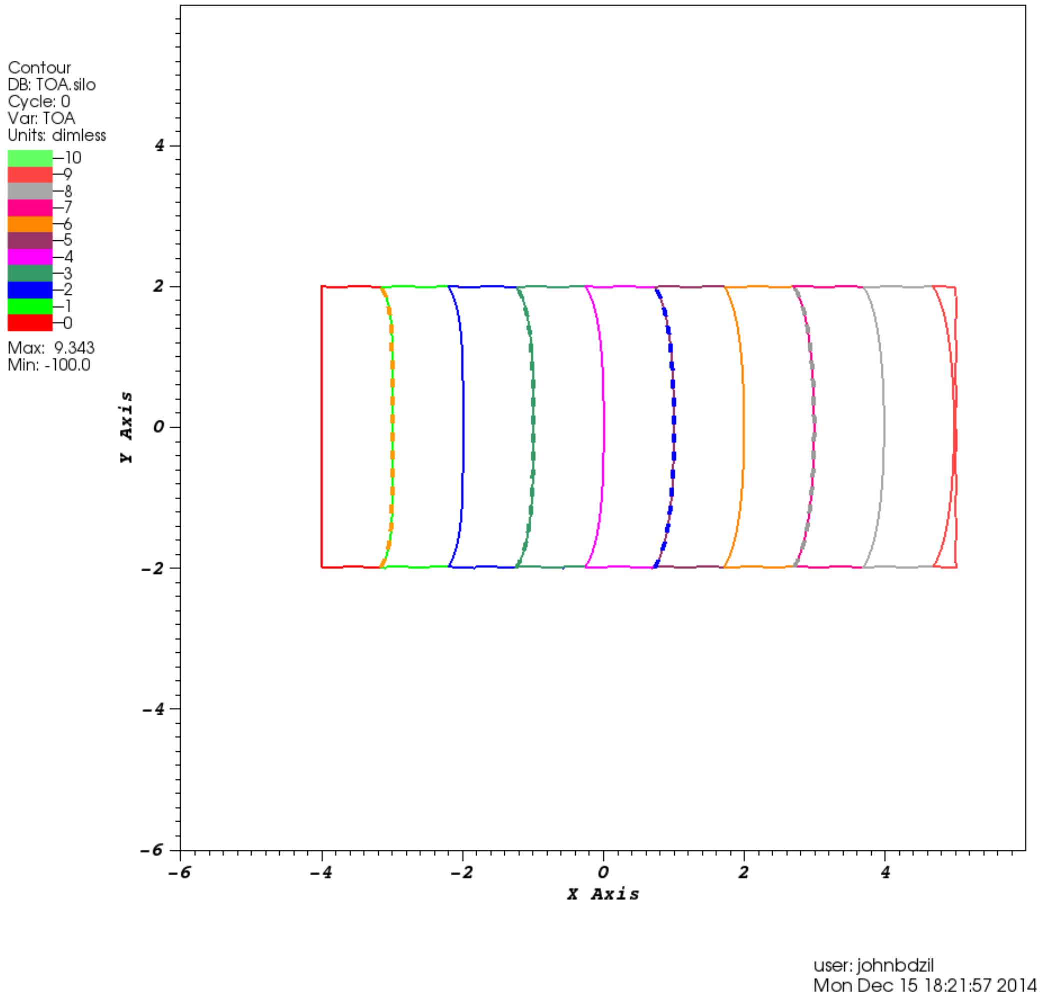

Figure-5a) My DSD2D-FLS calculation of the rate stick TOA-field for the problem displayed in Figure-2), where nx = ny = 600, and with noise added to psi(i,j) as described by Eq. (3). This noise leads to the slight oscillation that is visible in the HE boundary. The DSD2D-FLS integrated solution curves are shown as solid curves, while the "exact" Maple script generated solution curves are shown as dashed curves. The solutions obtained with the two methods essentially overlay. No noise is apparent in the DSD2D-FLS computed TOA-field, which is the expected result.

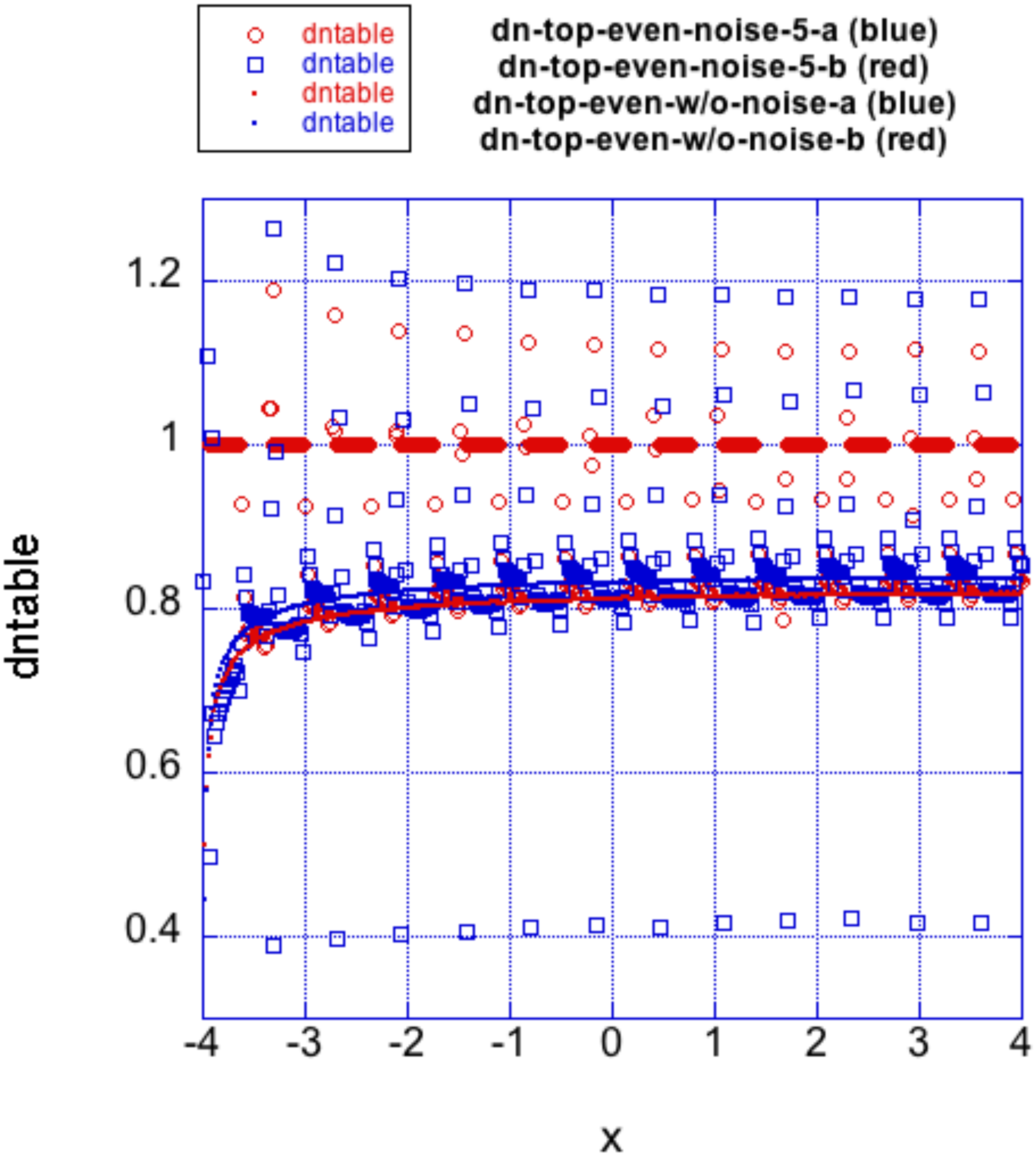

Figure-5b) My DSD2D-FLS calculation of the rate stick dn-field (dntable), using the curvature of the level-set function, phi(i,j), to compute dn. The data displayed in red corresponds to dn along y = 2.00000, which is the top boundary for the nx=ny=600 points simulation. The points are from the DSD2D-FLS simulation with noise, while the solid line comes from the simulation without noise added to psi(i,j). The data displayed in blue corresponds to dn along y = 1.98, which is at a distance of dx into the HE, again with and without noise. The problem geometry is that displayed in Figure-2), where nx = ny = 600, and with noise added to psi(i,j) as described by Eq. (3). Substantial noise is evident. The dn = 1.0 plateaus correspond to the background value in the ghost-node region.

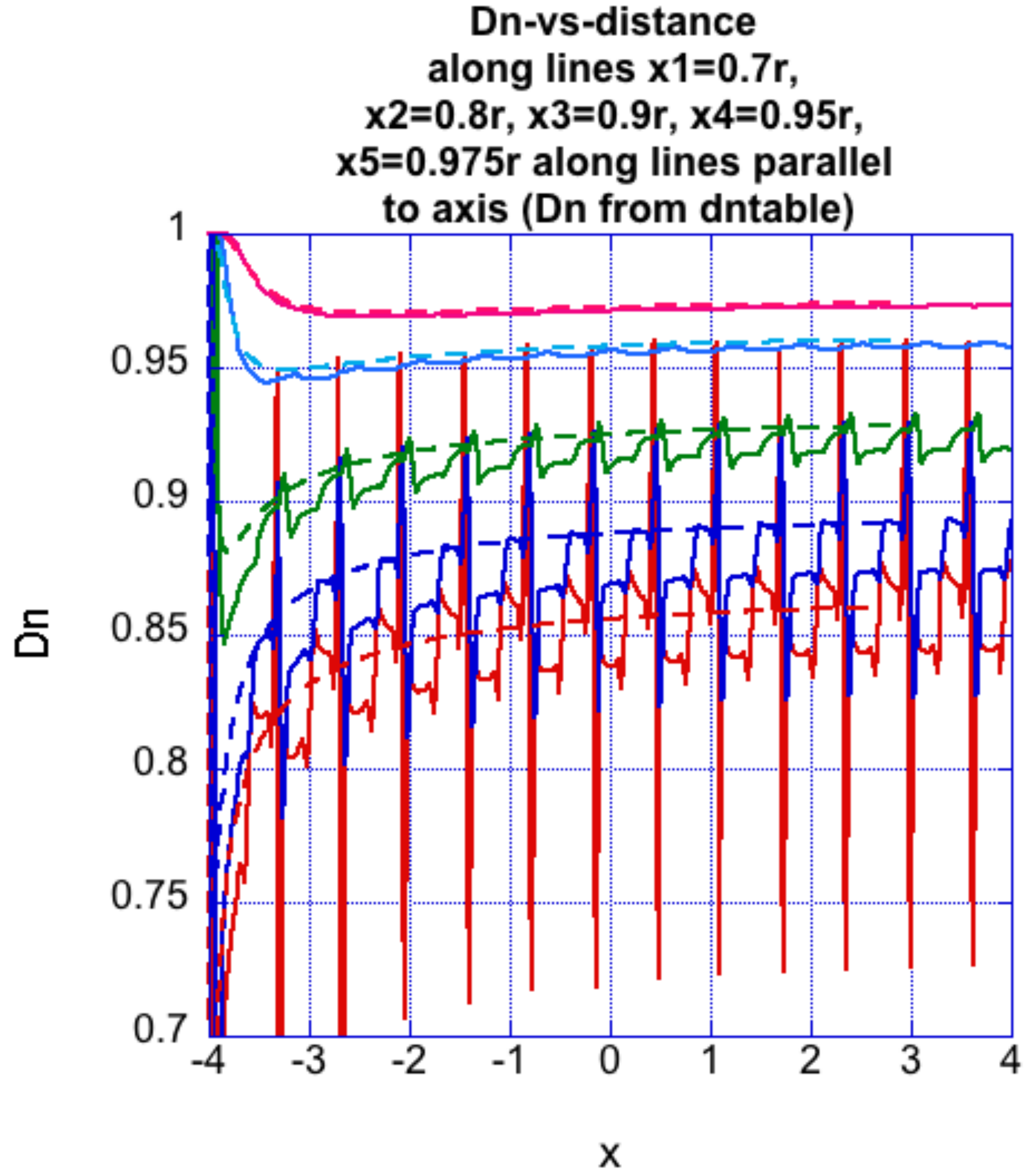

Figure-5c) My DSD2D-FLS solution for the dn-field (dntable) for the case when the noise described in Eq. (3) is added to psi(i,j). Substantial noise is now visible in the dn-field, displayed as the solid curves. These are compared with the “exact” Maple script generated solution, shown as dashed curves. The values of dn are displayed along five lines, which are displaced by a constant distant, y= 1.95 (red, lowest curve), 1.90 (blue), 1.80 (green), 1.60 (light blue) and 1.40 (light red), from the centerline of the rate stick (y = 0), for the problem shown in Figure-2). This noise in the dn-field is seen to diminish as one moves into the explosive, and away from the boundaries.

Summarizing, we find that noise in the description of the HE boundary coming from the psi(i,j)-function has little affect on the computed TOA-field (tb-field) near the boundary but has a substantial affect on the dn-field near the boundary. This result was anticipated, given that dn = 1/abs(grad(TOA(i,j))) and knowing that TOA(i,j) is first-order accurate in the mesh spacing, dx. That is what my testing, some of which is described above, shows. Testing also shows that the noise observed in the results does not depend strongly on the level of noise in the geometry describing, psi(i,j)-function. Since DSD2D-FLS sees the HE boundary as the collection of interior HE nodes that are closest to psi(i,j) = 0, then the HE boundary will appear as a zigzag boundary, on the O(dx) scale, due to small variations in the location of psi(i,j) = 0. Thus, a low-level of noise in psi(i,j) = 0 can lead to O(dx) scale variations in the boundary, which in turn will lead to the detonation front being sequentially accelerated and decelerated as the front interacts with what it sees as a zigzag boundary. This is displayed in Figure-5d) below. Such artificial variations in the HE boundary shape are what DSD2D-FLS sees as the HE boundary. Thus, one might consider how such O(dx), artificial variations in the boundary might be smoothed.

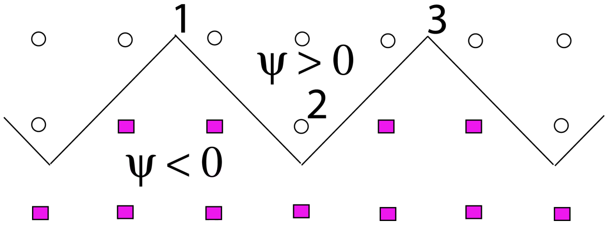

Figure-5d) Noise in psi(i,j) = 0, and also a slowly changing HE boundary, can generate a rough HE boundary within the DSD2D-FLS solver, as the boundary weaves between HE nodes (displayed as purple squares) and ghost nodes (displayed as open circles). In the vicinity of #1, this can lead to 4-points in the 9-point curvature stencil being ghost nodes, which can lead to dn being poorly calculated near such a point on the HE boundary.

Given that near such a zigzag boundary the 9-point curvature stencil can require 4-points from the ghost node region, and given that the ghost nodes are populated mostly such that the DSD angle boundary condition is satisfied, then we might consider how to make the curvature calculation near the boundary less dependent on ghost node values. In the next section we consider how to reduce the sensitivity of the curvature calculation near the boundary to ghost node values. To do this, we utilize the fact that in our boundary treatment, we have considered that the phi(i,j) function can be assumed to be locally planar near the boundary.

NEW WORK: THE REDUCTION OF NOISE IN THE DN FIELD NEAR BOUNDARIES

Over the years, the "customer" base has reported that the computed dn fields were noisy near the boundaries of the explosive. This was never directly reported to me until recently, when Mark Short brought this to my attention. Now, as I explained at the beginning of these notes, the method I have is first-order in the mesh spacing, dx, for the burn time field, tb(i,j) (the TOA(i,j) field). Given that information and given that dn = 1/abs(grad(tb)), then it follows that order one errors can be expected in the computed dn(i,j) field. Now, in applying the DSD boundary conditions, we use a layer of first-nearest neighbor ghost nodes to apply the DSD angle boundary conditions. The DSD front curvature involves second derivatives of the level-set function, which in turn requires data from a nine-point stencil. Near the boundary, this nine-point stencil can require not only data from first-nearest neighbor ghost nodes but also from second-nearest neighbor ghost nodes. To help alleviate the noise in dn(i,j) near the explosive boundaries, I recently changed my method for populating these second-nearest neighbor ghost nodes. To maintain consistency with the assumption that as far as applying DSD boundary conditions is concerned, the level-set function is assumed to be planar near the boundary, I've replaced my extrapolation method for populating second-nearest neighbor ghost nodes with simple linear extrapolation. Expressed in DSD2D-FLS notation,

(4)   phi(i,j) = -p2 + 2.*p1 .

This set of consistent assumptions then brings with it a reduced dependency of the curvature of the level-set function, phi(i,j), on second-nearest neighbor ghost nodes, leading to the cross second derivative being identically zero

(5) phixy(i,j) = [phi(i+1,j+1) + phi(i-1,j-1) – phi(i-1,j+1) – phi(i+1,j+1)]/4*dx*dy ,
      phixy(i,j) = 0.0 .

Thus, the values of the second-nearest ghost nodes neither enter the curvature calculation nor the dn(i,j) calculation near the boundary. Changes are required in the DSD2D-FLS subroutines ibextra.f and ibupdate.f at the point where the second-nearest neighbor ghost nodes are populated. Those one-line changes are detailed immediately below.

```
#########################################################

c       phi(i,j) = p2 - 2.*dxl*(di*phix +dj*phiy)

c Let the second derivative along the given 45-degree
c line be zero, thus staying with the planarity of phi
c locally. It is important to note that doing this keeps
c the IB2 nodes from entering into the curvature calculation.
c This is consistent with what re-distancing does in
c the interior regions. So here we are assuming that
c re-distancing is on, with linear extrapolation then
c being the consistent thing to do. Even with re-distancing
c off, performing linear extrapolation keeps the IB2 nodes
c from entering into the curvature calculation near the
c HE boundaries.

        phi(i,j) = -p2 + 2.*p1
```

###

From this point forward in these notes, I will use these modified ibextra.f and ibupdate.f subroutines in the DSD2D-FLS simulations that I report on. [NOTE: I also will be using the change to driver.f that implements the noise-reduction strategy reported earlier in these notes.]

Here I return to the Quirk rate stick, displayed in Figure-2) and for which my DSD2D-FLS results are shown in Figures-3) and Figures-5a)-5c), and repeat the DSD2D-FLS simulations so as to include the changes detailed directly above for populating ghost nodes. For these simulations (and these simulations only), I will not be applying the noise filtering that is now a part of driver.f, and the psi(i,j) function will contain noise as given by Eq. (3).

As expected, the TOA-field, displayed in Figure-6a), appears unchanged when compared with Figure-5a) and shows good agreement with the “exact” Maple script solution. Displayed in Figure-6b) are the DSD2D-FLS computed values of dn along y = 2.0 (red) and y = 1.98 (blue). Again, the plotted points correspond to the simulation where the psi(i,j)-function has added noise, according to Eq. (3). The solid curves correspond to the results from the noise-free simulation. The noise in dn is roughly half of that observed in Figure-5b). The dn values from the simulation containing noise in psi(i,j), mostly cluster around the curves for the noise-free simulations. Again, the plateaus consisting of points (and displayed in red) correspond to the default value of dn that is initialized into the ghost-node region. Shown in Figure-6c) is dn, computed with DSD2D-FLS for the noise containing psi(i,j)-field simulations, compare with the “exact” solutions for the noise-free problem, in both cases displayed along lines of constant-y as in Figure-3c). What these results show is that the noise in dn is roughly half of what is observed in Figure-5c), and where again the noise is seen to decrease as one moves further into the HE region.

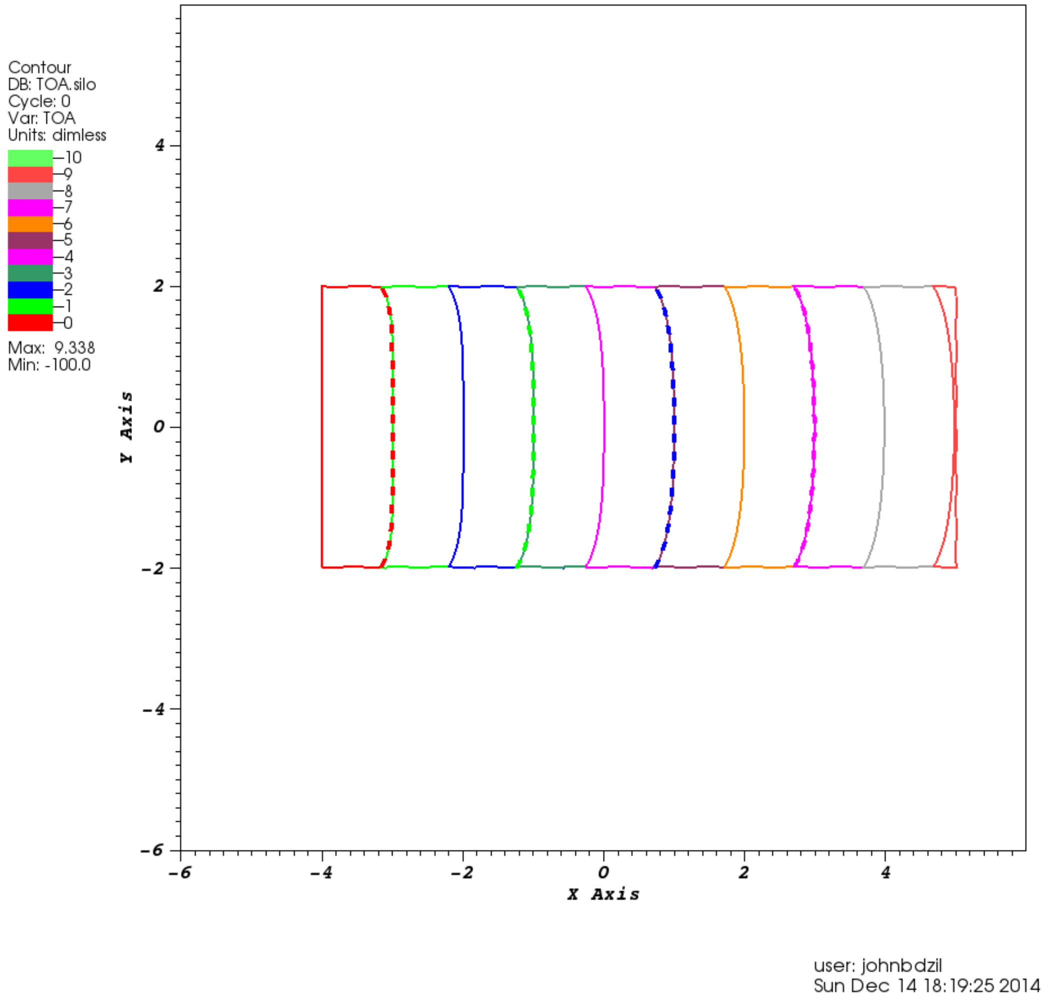

Figure-6a) My DSD2D-FLS calculation of the rate stick TOA-field for the problem displayed in Figure-2), where nx = ny = 600, and with noise added to psi(i,j) as described by Eq. (3). This noise leads to the slight oscillation that is visible in the HE boundary. The DSD2D-FLS integrated solution curves are shown as solid, while the "exact" Maple script generated solution curves are shown as dashed. The solutions obtained with the two methods essentially overlay. As before, no noise is apparent in the DSD2D-FLS computed TOA-field, which is the expected result and which is similar to what is displayed in Figure-5a).

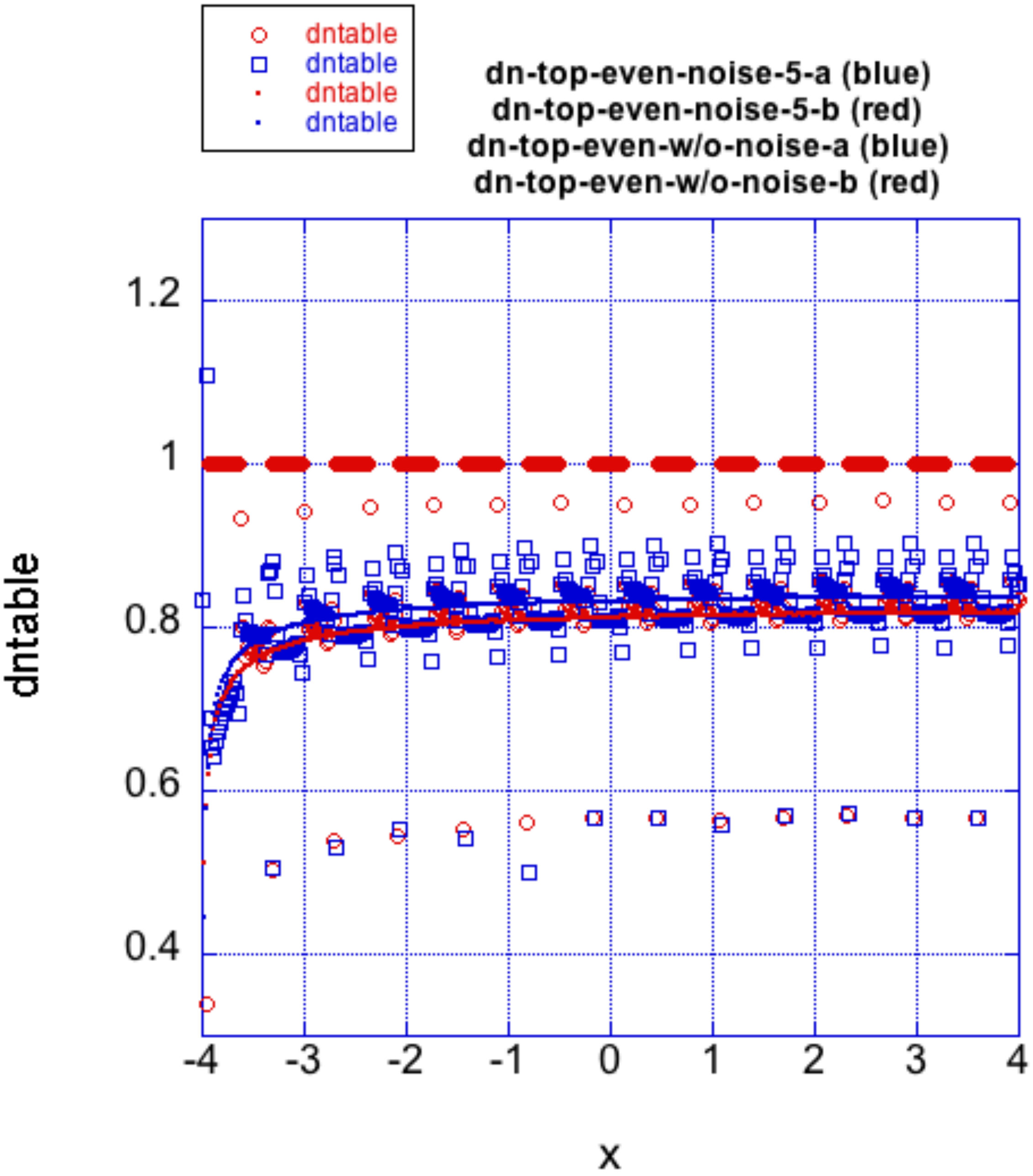

Figure-6b) My DSD2D-FLS calculation of the rate stick dn-field (dntable), using the curvature of the level-set function, phi(i,j), to compute dn. The data displayed in red corresponds to dn along y = 2.00000, which is the top boundary for the nx=ny=600 points simulation. The points are from the DSD2D-FLS simulation with noise, while the solid line comes from the simulation without noise added to psi(i,j). The data displayed in blue corresponds to dn along y = 1.98, which is at a distance of dx into the HE, again with and without noise. The problem geometry is that displayed in Figure-2), where nx = ny = 600, and with noise added to psi(i,j) as described by Eq. (3). The noise in Dn is reduced substantially from what is displayed in Figure 5b). Again, the dn = 1.0 plateaus correspond to the background value in the ghost-node region.

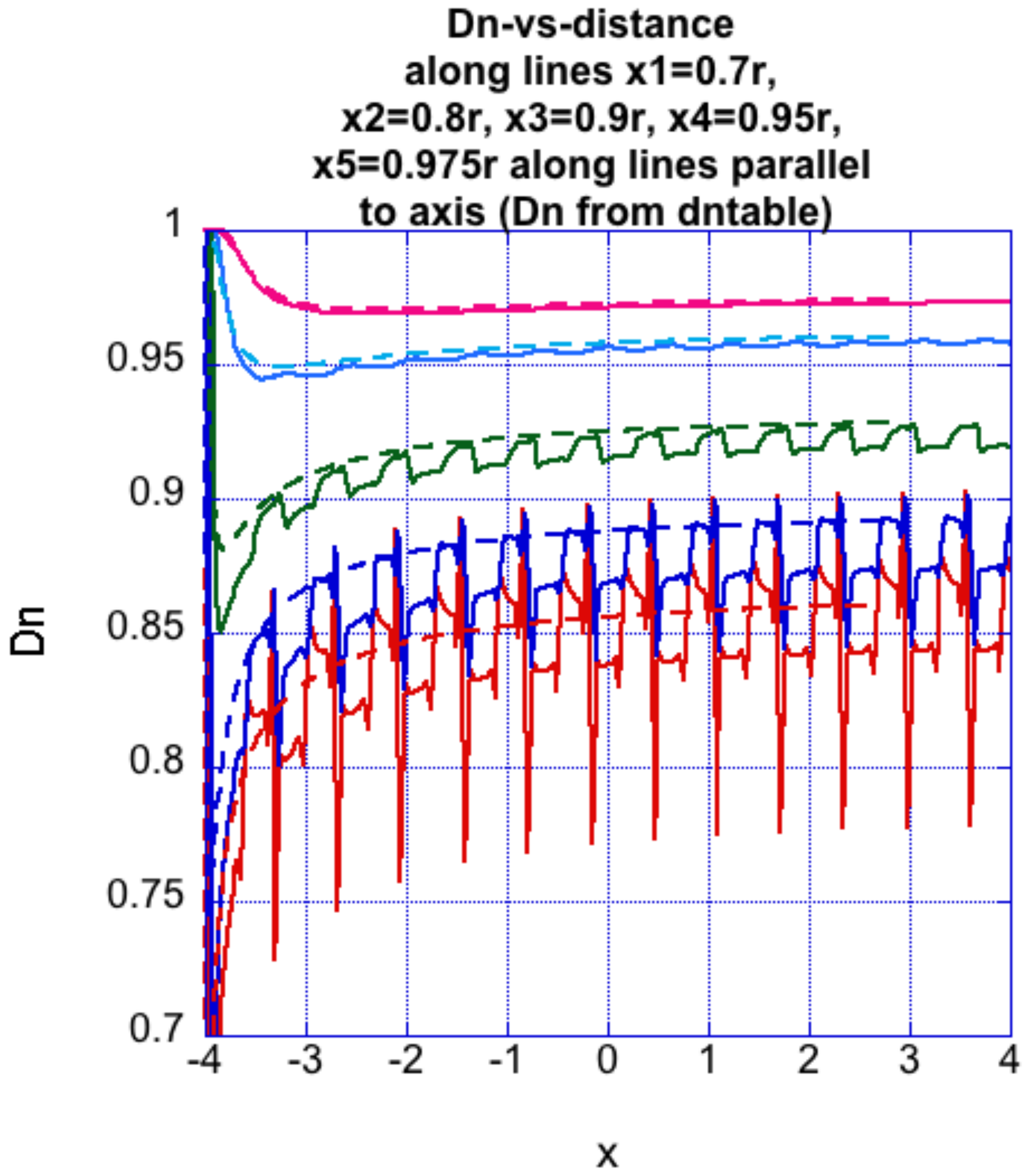

Figure-6c) My DSD2D-FLS solution for the dn-field (dntable) for the case when the noise described in Eq. (3) is added to psi(i,j). Although substantial noise is still evident in the dn-field, displayed as the solid curves, it is roughly cut in half from what is displayed in Figure-5c). As before, these are compared with the "exact" Maple script generated solution, shown as dashed curves. The values of dn are displayed along five lines, which are displaced by a constant distant, y= 1.95 (red, lowest curve), 1.90 (blue), 1.80 (green), 1.60 (light blue) and 1.40 (light red), from the centerline of the rate stick (y = 0), for the problem shown in Figure-2). Again, this noise in the dn-field is seen to diminish as one moves into the explosive, and away from the boundaries.

COMPUTING DN FROM 1/abs(grad(toa)) USING SMOOTHING STENCILS

The values I've displayed for the dn-field (dntable) were obtained directly from the curvature of the level-set function, phi(i,j), that is, dn = 1 – alpha*kappa, where kappa = div(grad(phi)/abs(grad(phi))). Once the TOA-field (tb-field) is obtained over the entire solution space, dn can also be obtained in a post-processing step through the gradient of the tb-field, dn = 1/abs(grad(tb)). Here I consider how higher-order, smoothing stencils for derivatives can be applied to the tb-field to obtain another rendition of the dn-field, referred to as dn_tb_ho, that is less noisy near the HE boundaries. My results show that dntable and dn_tb_ho are essentially coincident in regions away from the HE boundaries. As part of my study, I considered a variety of different post-processing strategies, most of which can be found coded in driver.f. Here I focus on using higher-order stencils for computing derivatives, using one-sided versions of those stencils near boundaries. Displayed below is the driver.f code section that I use to compute dn_tb_ho.

```
########################################################
c direct evaluation of dn from gradient of tb field, using higher-order
c smoothing difference formulas from Pavel Holoborodko

      tbx = (2.*(tb(i+1,j)-tb(i-1,j))+tb(i+2,j)-tb(i-2,j))/(8.*dx)
      if( (tb(i+1,j).eq.-100.0).or.(tb(i+2,j).eq.-100.0) ) then
c        tbx = (3.*tb(i,j)- 4.*tb(i-1,j)+tb(i-2,j))/(2.*dx)
       tbx = (tb(i,j)+tb(i-1,j)-tb(i-2,j)-tb(i-3,j))/(4.*dx)
      endif
      if( (tb(i-1,j).eq.-100.0).or.(tb(i-2,j).eq.-100.0) ) then
c        tbx = (3.*tb(i,j)- 4.*tb(i+1,j)+tb(i+2,j))/(2.*dx)
       tbx = (tb(i,j)+tb(i+1,j)-tb(i+2,j)-tb(i+3,j))/(4.*dx)
      endif
      tby = (2.*(tb(i,j+1)-tb(i,j-1))+tb(i,j+2)-tb(i,j-2))/(8.*dx)
      if( (tb(i,j+1).eq.-100.0).or.(tb(i,j+2).eq.-100.0) ) then
c        tby = (3.*tb(i,j)- 4.*tb(i,j-1)+tb(i,j-2))/(2.*dx)
       tby = (tb(i,j)+tb(i,j-1)-tb(i,j-2)-tb(i,j-3))/(4.*dx)
      endif
      if( (tb(i,j-1).eq.-100.0).or.(tb(i,j-2).eq.-100.0) ) then
c        tby = (3.*tb(i,j)- 4.*tb(i,j+1)+tb(i,j+2))/(2.*dx)
       tby = (tb(i,j)+tb(i,j+1)-tb(i,j+2)-tb(i,j+3))/(4.*dx)
      endif
      dn_tb_ho(i,j) =                              &
     &   max(dmin,min(dmax,1./sqrt(tbx**2+tby**2+zeps2)))

########################################################
```

Displayed below in Figures 6d) & 6e) are the dn_tb_ho results for dn corresponding to what is displayed for dntable in Figures 6b) & 6c). What the comparison shows is that the noise near the HE boundaries is further reduced by applying the smoothing stencils to compute dn (i.e., dn_tb_ho) from the tb-field, which itself uses the modifications to the procedure for populating ghost nodes that I've described.

In the next section of this report, I go on to consider other problem geometries, using all the improvements I've detailed above to compute the dn-field. In all the examples that are to follow, I do not add noise to the HE boundary definition, psi(i,j). Roughness now enters what DSD2D-FLS sees as the HE boundary due to the fact that the HE boundary crosses the mesh either obliquely to the mesh or as a circular boundary, as displayed in the upper right hand side of Figure-4).

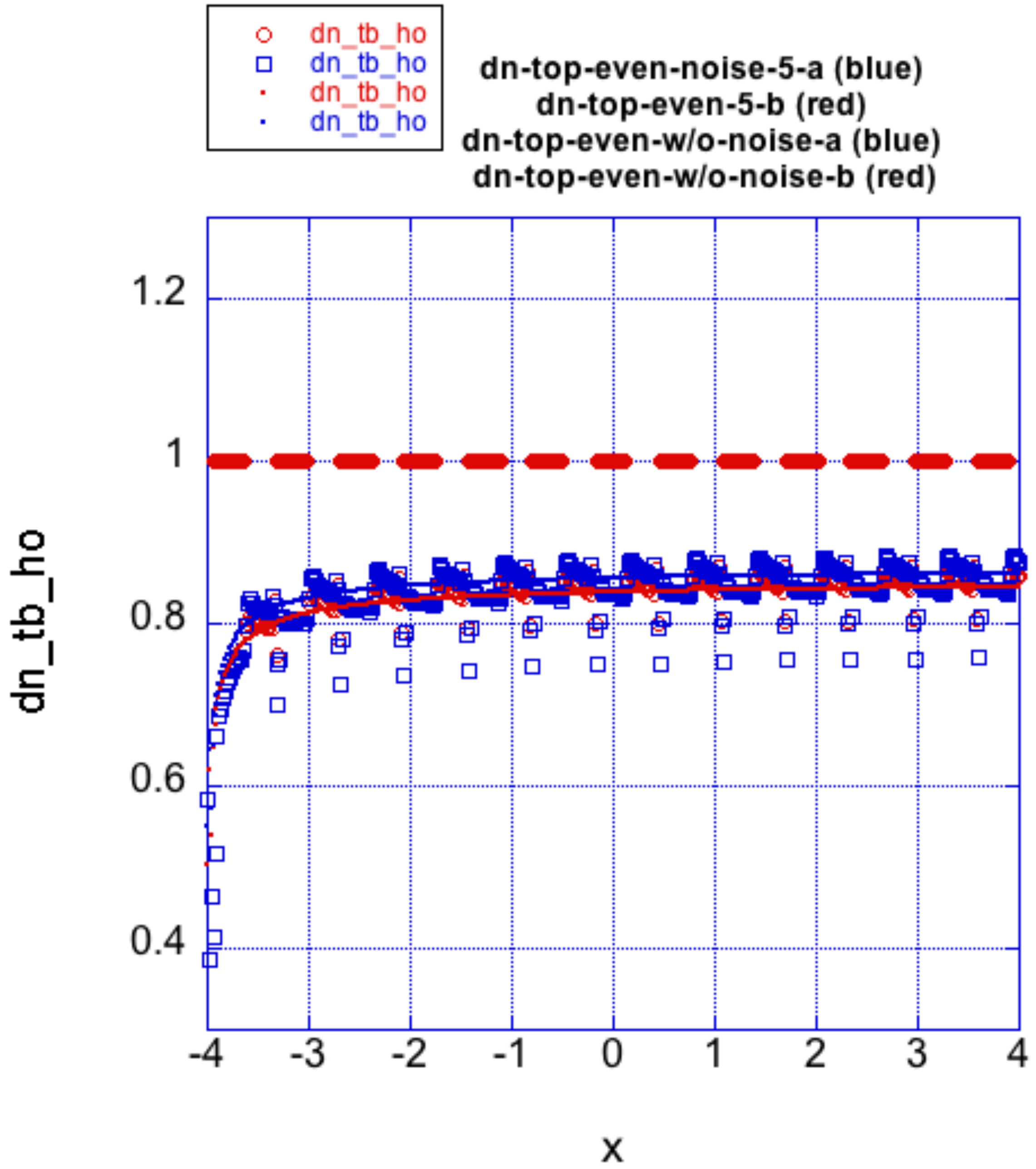

Figure-6d) My DSD2D-FLS calculation of the rate stick dn-field (dn_tb_ho), using the smoothing stencils on tb(i,j)=TOA(i,j) to compute dn. The data displayed in red corresponds to dn along y = 2.00000, which is the top boundary for the nx=ny=600 points simulation. The points are from the DSD2D-FLS simulation with noise, while the solid line comes from the simulation without noise added to psi(i,j). The data displayed in blue corresponds to dn along y = 1.98, which is at a distance of dx into the HE, again with and without noise. The problem geometry is that displayed in Figure-2), where nx = ny = 600, and with noise added to psi(i,j) as described by Eq. (3). The noise in Dn is reduced once again, now from what is already the reduced noise displayed in Figure 6b). Again, the dn = 1.0 plateaus correspond to the background value in the ghost-node region.

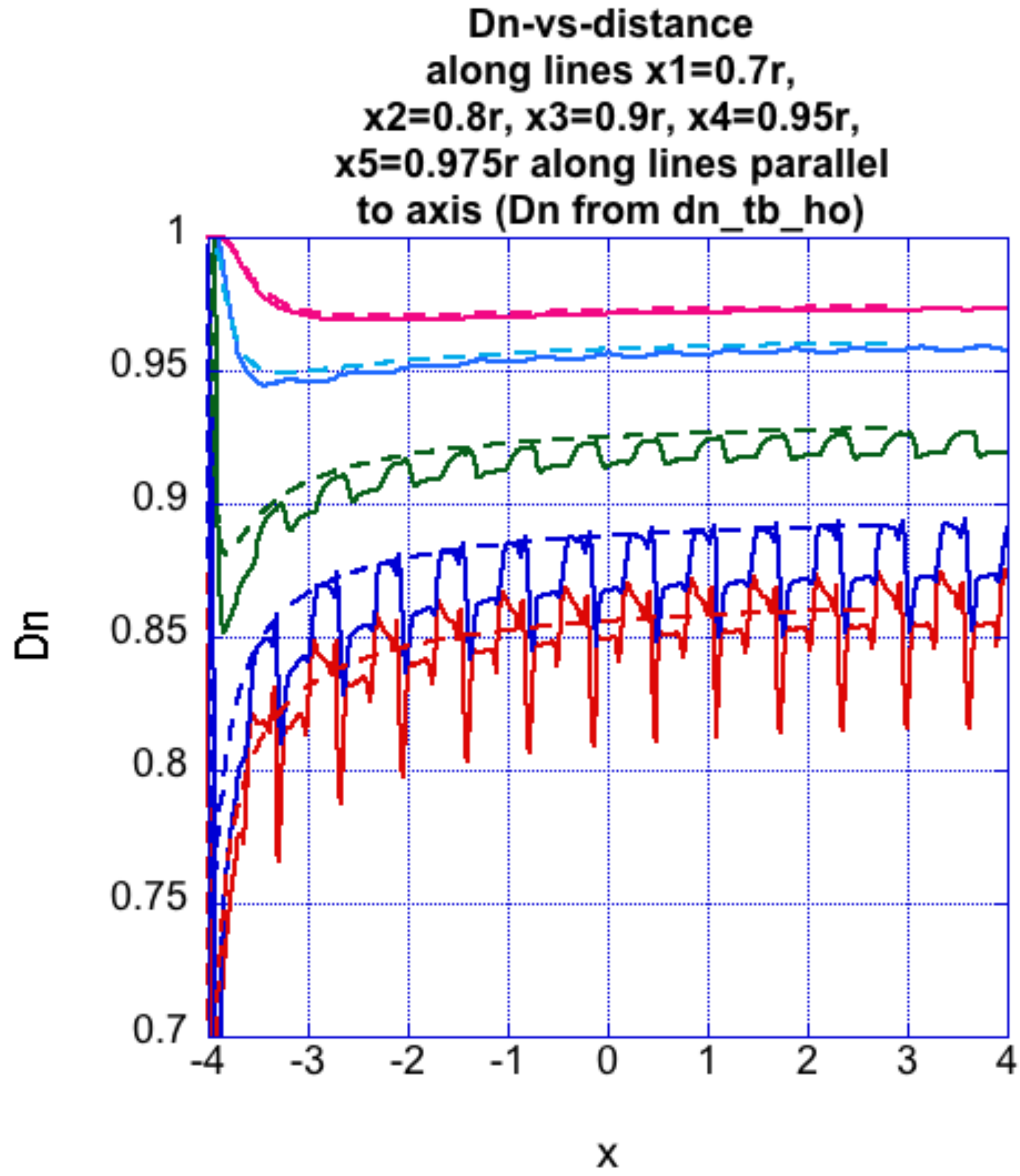

Figure-6e) My DSD2D-FLS solution for the dn-field (dn_tb_ho) for the case when the noise described in Eq. (3) is added to psi(i,j). Although noise is still evident in the dn-field, displayed as the solid curves, it is reduced again from what is displayed in Figure-6c). As before, these are compared with the "exact" Maple script generated solution, shown as dashed curves. The value of dn are displayed along five lines, which are displaced by a constant distant, y= 1.95 (red, lowest curve), 1.90 (blue), 1.80 (green), 1.60 (light blue) and 1.40 (light red), from the centerline of the rate stick (y = 0), for the problem shown in Figure-2). Again, this noise in the dn-field is seen to diminish as one moves into the explosive, away from the boundaries.

PROBLEMS WITH HE BOUNDARIES OBLIQUE TO MESH: DSD2D-FLS SOLUTIONS

All the solutions we display in this and the following sections use the modifications I've described that have been made to driver.f, ibextra.f and ibupdate.f. No noise is explicitly added to the geometry definition function, psi(i,j). The roughness that DSD2D-FLS sees in the boundary results from the physical HE boundary crosscutting over the mesh lines/points used in the DSD2D-FLS solution.

To serve as a further validation of the above, simple procedure for reducing the near-boundary noise in the dn-field, I next show the results for a rate stick that makes an angle of 30-degrees with the direction of the vertical mesh lines. Figure-7a) shows a sub-region of the computational mesh (for a problem with coarse to moderate resolution) superimposed on which is the psi = 0.0 line describing the HE boundary. The purple-shaded squares are the interior HE region points that are closest to the boundary. DSD2D-FLS sees the boundary through these points. Little to no symmetry exists between the mesh and the HE boundary, although there is a long wavelength repeat pattern. This example, where the boundary crosscuts the mesh, is the typical way in which noise is introduced into the dn(i,j)-field near HE boundaries in a DSD2D-FLS simulation.

I solve this problem using the same DSD parameters that I set down earlier (omega_s = 50 degrees, omega_c = 55 degrees, dmin = 0.1, dmax = 9.0, cfl = 0.9 and re-distancing on). The resolution used for this example is nxpts = nypts = 1408. The results displayed in Figures 7b)-7d) show the tb(i,j) and dn(i,j) fields from both dntable and dn_tb_ho. Although the dn(i,j)-field formally can have O(1) errors, one finds the field is smooth and relatively noise free, even near the boundaries. As before, the tb(i,j)-field has O(dx) errors, which vanish under mesh resolution. As displayed in Figure-7b), the DSD2D-FLS tb(i,j)-field and the "exact" Maple script solution are coincident. Figures 7c) & 7d) show that the noise, at comparable fractional radii to those consider for the noise-peppered Quirk rate stick, is smaller than what we saw for the Quirk rate stick. Some part of this reduction is due to the higher mesh resolution used in this run. It should be noted, and I do so here, that the near-boundary noise we observe is confined to a narrower and narrower region in physical space as the resolution is increased.

Generally, the noise imparted to the near-boundary dn(i,j)-fields by the the psi = 0 line crosscutting the mesh is more random and appears to be smaller in magnitude (at least for the dn_tb_ho field) than what we saw for the Quirk rate stick, where noise was purposely added to the psi(i,j)-field. My next example problem considers detonation propagation in an arc of explosive.

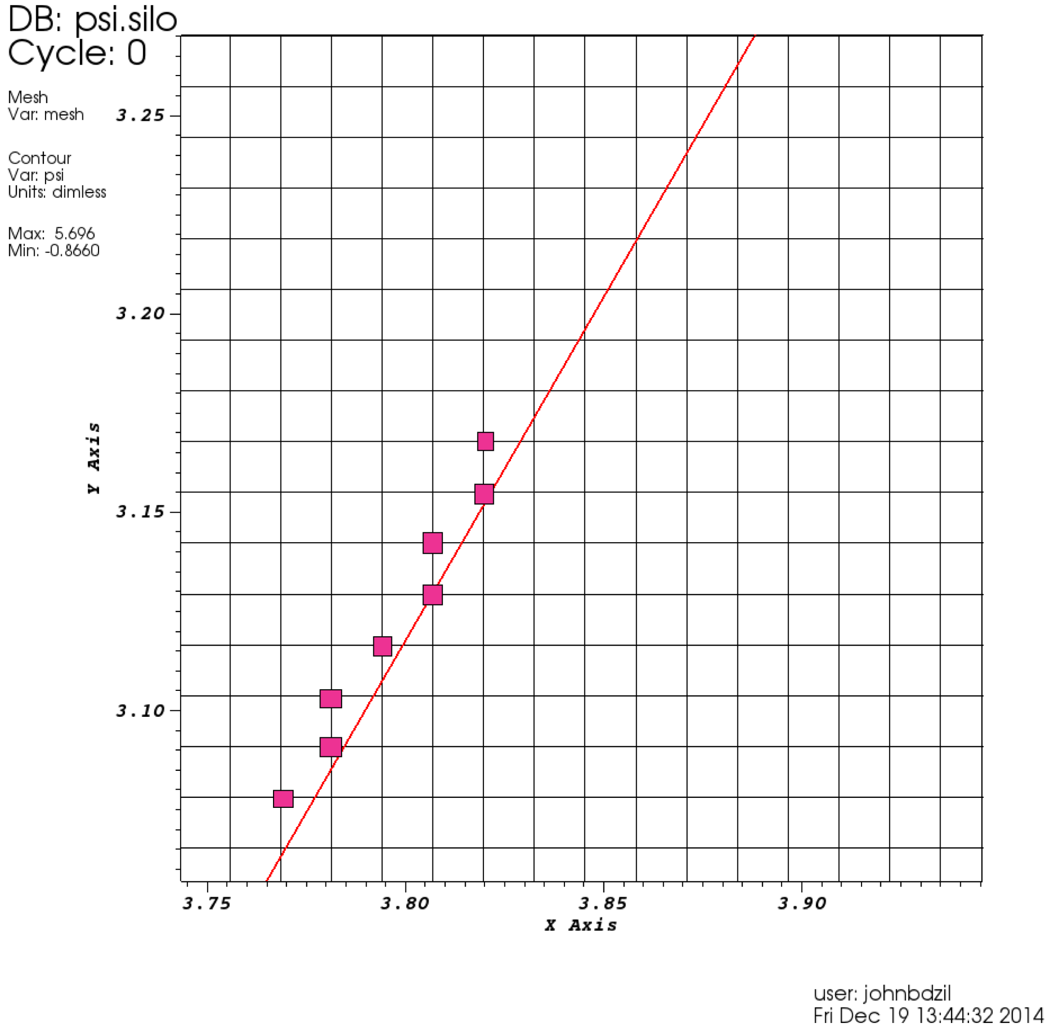

Figure-7a) A highly expanded view of the mesh and psi = 0.0 line for the 30-degree rate stick problem example. The HE region is above and to the left of the psi = 0.0 line. The solid-purple squares, which make a zigzag pattern, denote the interior HE points closest to the boundary. These points are how DSD2D-FLS sees the HE boundary.

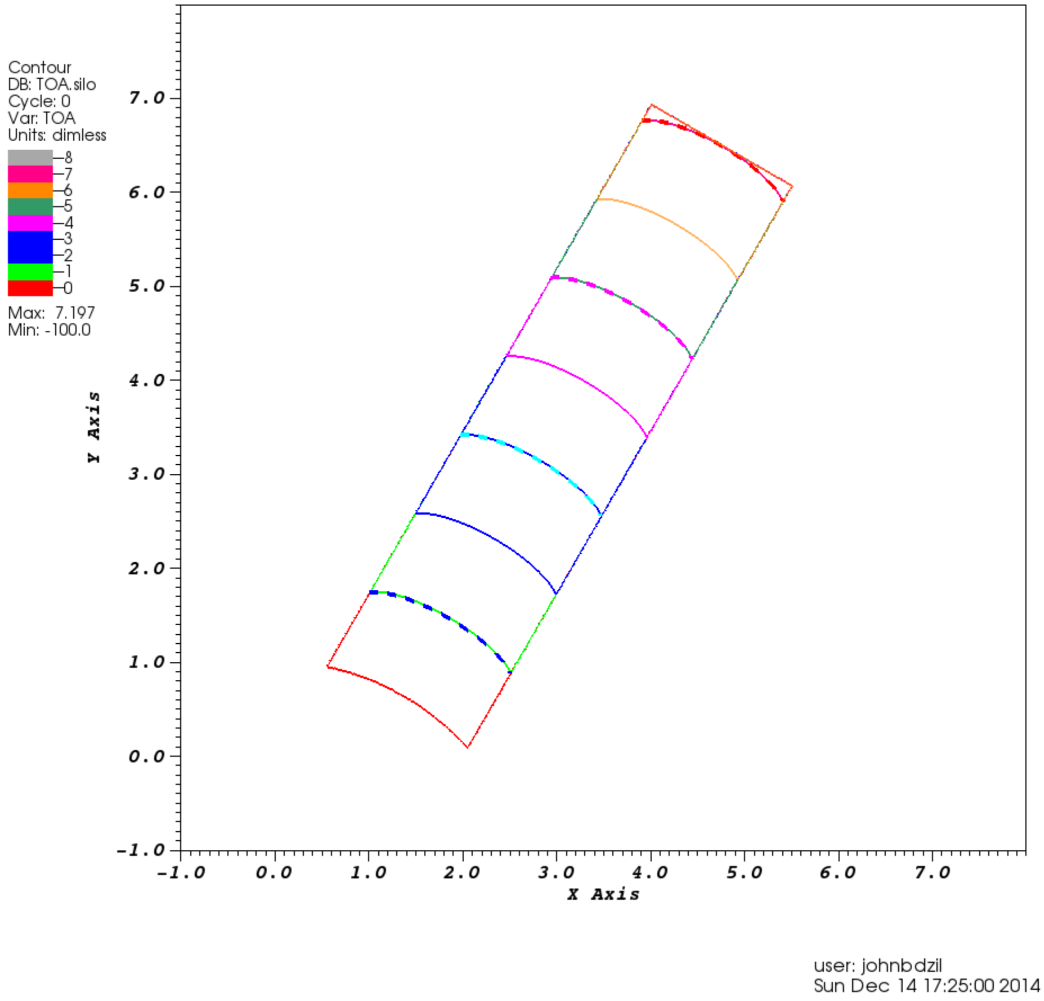

Figure-7b) My DSD2D-FLS calculation of the TOA-field for the 30-degree rate stick problem. This figure also serves to define the problem geometry. Here, nxpts = nypts = 1408. The DSD2D-FLS integrated solution curves are shown as solid, while the “exact” Maple script generated solution curves are shown as dashed. The solutions obtained with the two methods essentially overlay. As before, no noise is apparent in the DSD2D-FLS computed TOA-field, which is the expected result. I note that whatever noise is generated in the DSD2D-FLS solution is due to the crosscutting of the HE boundary across the mesh, as displayed in Figure-7a).

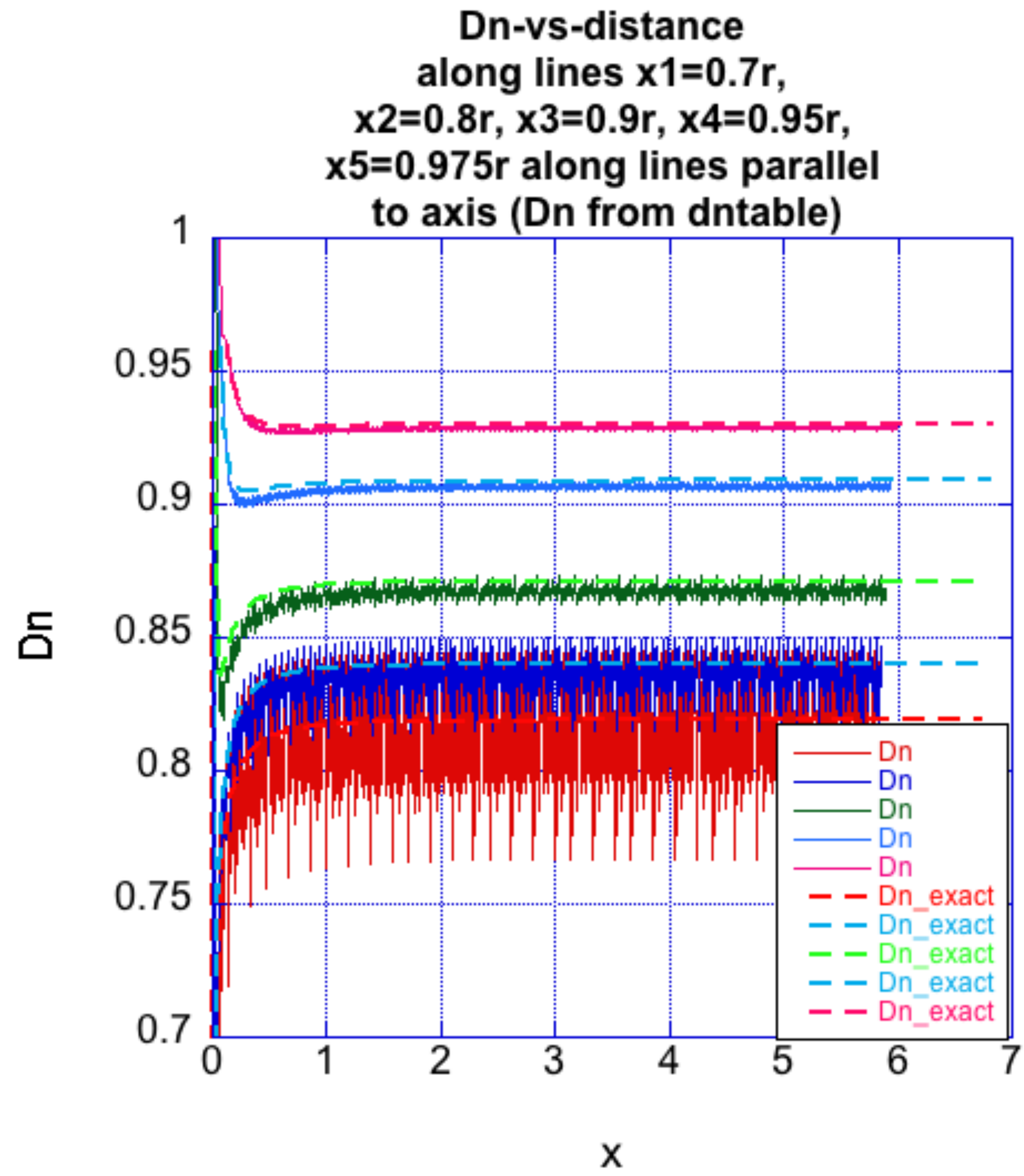

Figure-7c) My DSD2D-FLS solution for the dn-field (dntable) for the 30-degree rate stick problem. The noise in dntable for the outer-most radius is comparable to that displayed in Figure-6c). As before, these are compared with the "exact" Maple script generated solution, shown as dashed curves. The values of dn are displayed along five lines, which are displaced by a constant radial distant, r= 1.975r (red, lowest curve), 1.95r (blue), 1.90r (green), 1.80r (light blue) and 1.70r (light red), from the centerline of the rate stick (r = 0). Again, this noise in the dn-field is seen to diminish as one moves into the explosive, and away from the boundaries.

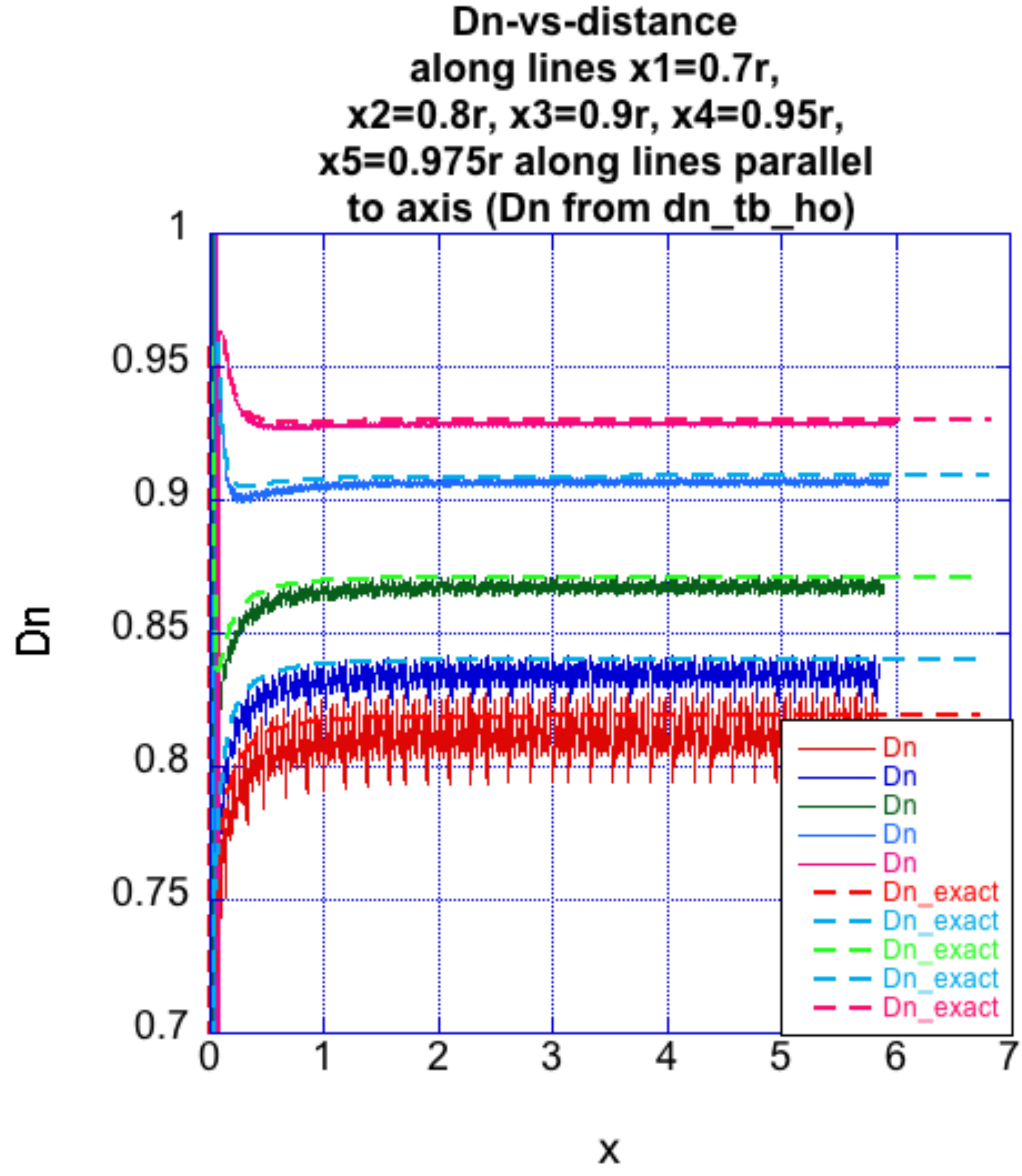

Figure-7d) My DSD2D-FLS solution for the dn-field (dn_tb_ho) for the 30-degree rate stick problem. The noise in dn_tb_ho for the outer-most radius is reduced to that displayed in both Figure-6d) and Figure-7c). As before, these are compared with the "exact" Maple script generated solution, shown as dashed curves. The values of dn are displayed along five lines, which are displaced by a constant radial distant, r= 1.975r (red, lowest curve), 1.95r (blue), 1.90r (green), 1.80r (light blue) and 1.70r (light red), from the centerline of the rate stick (r = 0). Again, this noise in the dn-field is seen to diminish as one moves into the explosive, and away from the boundaries.

DSD2D-FLS SIMULATIONS OF DETONATION IN AN ARC OF EXPLOSIVE

As part of the ASC-PEM-HE Program FY14 project management review of progress on their L2 Milestone, James Quirk presented a DSD simulation, performed with the August 2014 version of his CASH/Amrita wrapped 2010-code base DSD2D-FLS solver, for the problem of detonation in a 180-degree arc of explosive. The results from his simulation are displayed in Figure-8a). These results do not show any obvious problems or inconsistencies.

Lacking information on any V&V that was performed on the CASH/Amrita wrapped 2010-code base DSD2D-FLS solver, here I'll present results for a 270 degree explosive arc that has the same inner and outer radii, r_inner = 2.0 and r_outer = 4.0, as those shown in Figure-8a). All my results were generated with the DSD parameters I listed earlier for the Dn law and for dmin and dmax. Although, I've done a resolution study of my results, here I'll present a cross section of my results for nxpts=nypts=704.

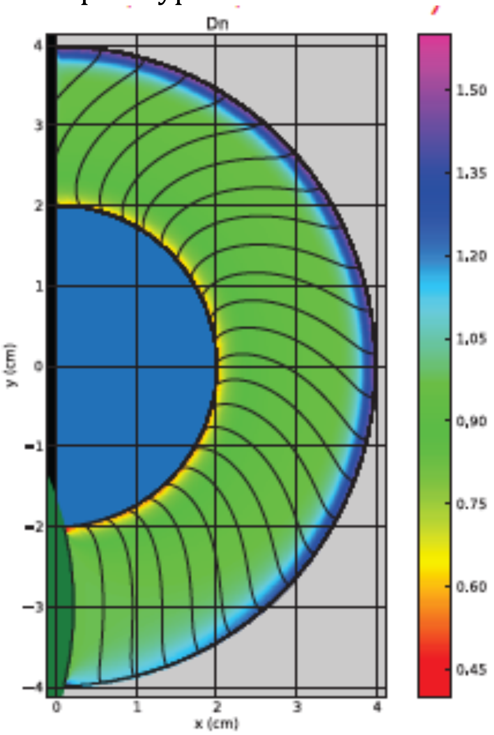

Figure-8a) A DSD simulation, performed using the CASH/Amrita wrapped 2010-code base DSD2D-FLS solver, of detonation in a 180-degree arc of explosive as discussed in LA-14277. The fixed boundary condition (omega_c = 90 degrees) is applied on the outside of the arc, and the free boundary condition (omega_c = omega_s) is applied on the inside of the arc. Contours of the tb(i,j)-field are displayed over a color palette plot of the dn(i,j)-field. Localized large departures of dn(i,j) values off of the mean would not be noticeable in such a plot. Although most details of the DSD parameters used in this simulation are not know to me, Mark Short did reveal that the CFL parameter was set at a very low value, perhaps with CFL = 0.09. This indicates that there is some degree of stiffness to the problem, which is due to either the problem not being temporally or spatially resolved, and raises questions about the quality of these results and/or the implementation.

In my simulations of the arc problem, I extend the arc by 90 degrees to a 270-degree arc and consider a wide range of values for omega_s and omega_c (for the results for wide-ranging values of omega_s and omega_c, the reader should consult an Appendix). To display these results in a more quantitative manner, I plot contours of the tb(i,j)-field as before and plot the dn(i,j)-field along curves of constant radius, r. As will become apparent, the dn(i,j) fields show noise near the inner boundary of the arc with the noise diminishing as one moves further into the arc. My results appear in the few figures that follow, and where the values of omega_s = 50-degrees and omega_c = 55-degrees are used. NOTE: In my standalone DSD2D-FLS research code, omega_s and omega_c have the same values on all boundaries of the explosive region. I use my standalone DSD2D-FLS code which I used for the 30-degree rate stick problem discussed in the previous section. I display only the dn_tb_ho field for the dn(i,j)-field from this point forward.

As before, I compare the DSD2D-FLS solution of the 270-degree arc problem with the “exact” solution generated with a Maple script. The solution developed with the Maple script solves the front propagation problem. That is, first the PDE for the detonation-front normal angle, phi(r,t), is solved as a function of the radial coordinate, r, and time, t, with a high-resolution, error controlled PDE solver available in Maple. Then in a second step, the motion of the front over a Cartesian grid is solved for with an ODE solver. Displayed in Figure-9a) is a comparison of the DSD2D-FLS simulation of the TOA(i,j)-field for the case omega_s = 50-degree, omega = 55-degrees and run at nxpts = nypts = 1408 points, with the “exact” solution of the problem. The agreement of the DSD2D-FLS TOA-field solution with the “exact” Maple script solution is generally good. One can however see some finite-resolution effects, showing the DSD2D-FLS wave front at t = 12 microseconds being slightly behind the “exact” solution.

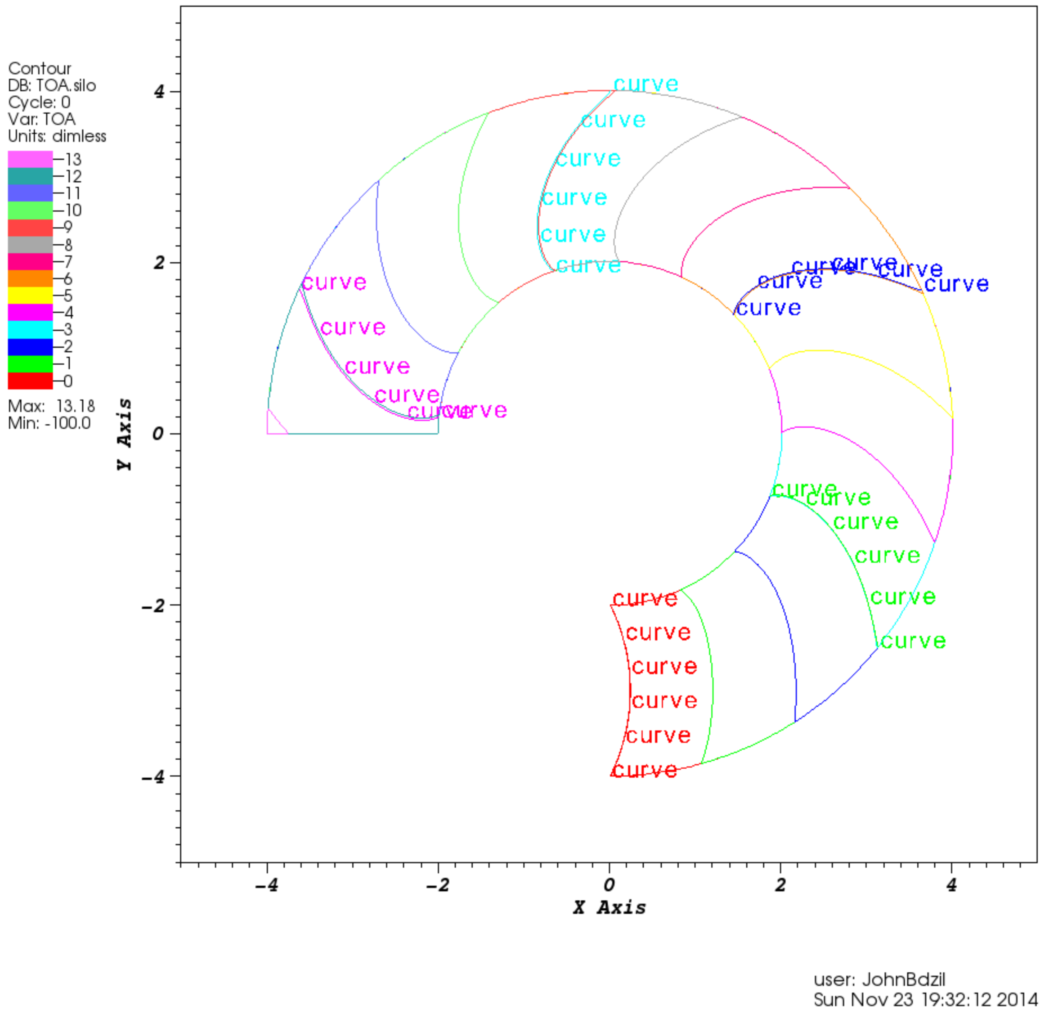

Figure-9a) My DSD2D-FLS calculation of the TOA-field for the 270 degree, explosive-arc problem. This figure also serves to define the problem geometry. The detonation begins at the 6 o'clock position and runs counterclockwise, exiting the arc at t = 13.18 microseconds. Here, nxpts = nypts = 1408. The DSD2D-FLS integrated solution curves are shown as solid, while the "exact" Maple script generated solution curves are shown as the labeled "curves." The solutions obtained with the two methods essentially overlay. As before, no noise is apparent in the DSD2D-FLS computed TOA-field, which is the expected result. I note that whatever noise is generated in the DSD2D-FLS solution is due to the crosscutting of the HE boundary across the mesh, similar to what is displayed in Figures 4) & 7a).

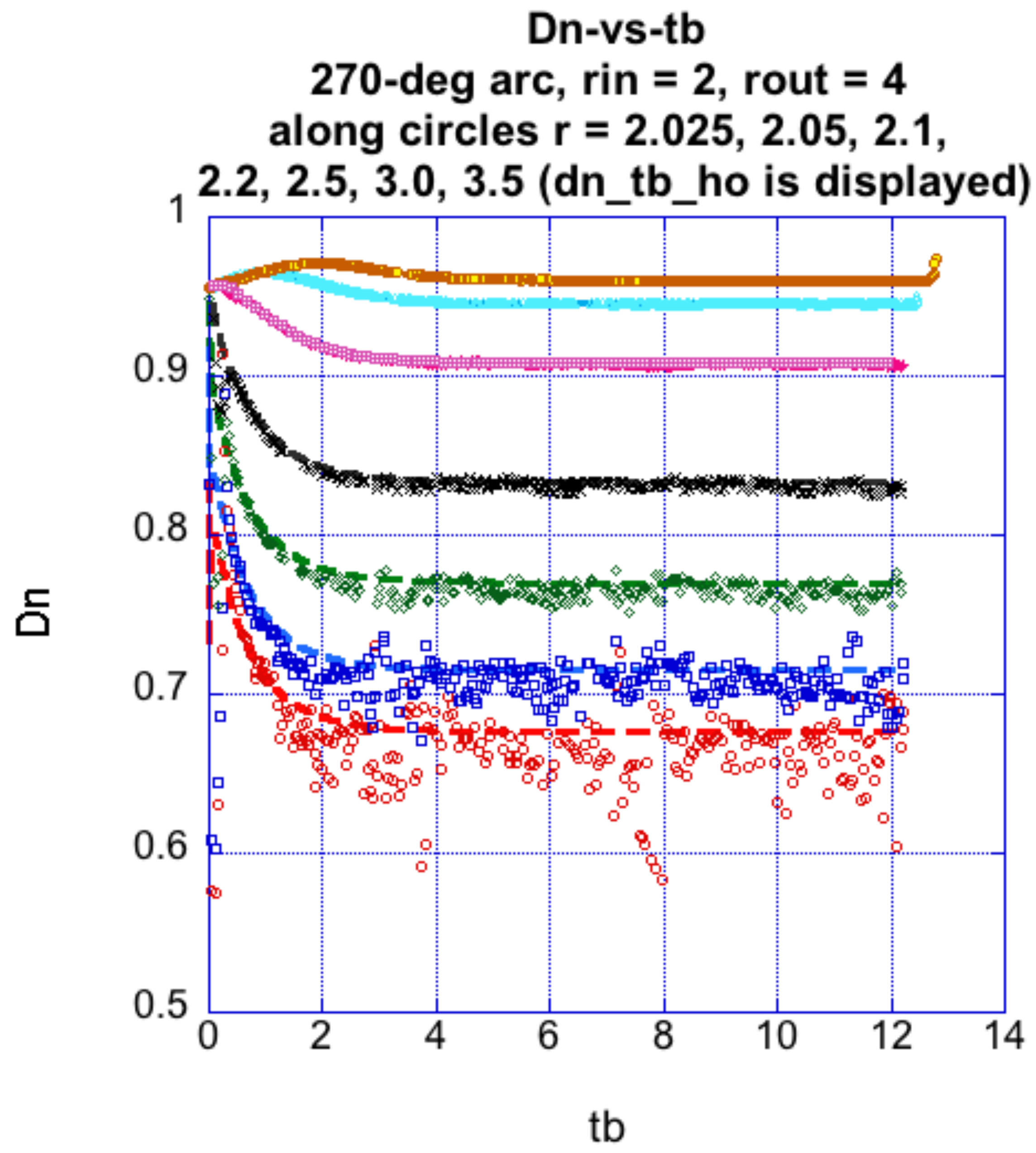

Figure-9b) My DSD2D-FLS solution for the dn-field (dn_tb_ho) for the 270 degree explosive-arc problem. The noise in dn_tb_ho is greatest for the inner-most radius, being of O(10%). The noise is seen to diminish as one moves into the explosive arc. As before, the DSD2D-FLS solutions (displayed as 20% of the data points) are compared with the "exact" Maple script generated solution, shown here as dashed curves. The value of dn are displayed along five circular arcs, which are located at the fixed radial distances of, r = 2.025 (red, lowest curve), 2.05 (blue), 2.1 (green), 2.2 (black), 2.5 (pink), 3.0 (light blue) and 3.5 (light brown). Again, this noise in the dn-field is seen to diminish as one moves into the explosive, and away from the boundary. The noise is at its greatest at the 3, 12 and 9 o'clock locations on the arc.

SUMMARY

The DSD2D-FLS V9, which was used to perform the simulations appearing in the last few sections, adds only a few minor changes to my serial, 2010 DSD2D-FLS code base. The two changes, which are detailed in these notes, improve the computation of the normal detonation speed near the explosive's boundaries. The most significant of these changes concerns how second-nearest neighbor ghost nodes are populated in the subroutines, ibextra.f and ibupdate.f. I now use linear extrapolation along 45-degree lines to set the second-nearest neighbor nodes. This change then leads to the cross derivative, phi_xy, being zero, and thus the second-nearest neighbors are not contributing to the curvature calculation near the boundary. In that way, my longstanding questions about how second-nearest neighbor ghost nodes values should be populated becomes a moot point, since now the values at these nodes do not influence the curvature calculation near the boundary. The other minor changes are in driver. A few statements are added to reduce the sensitivity of the geometry defining function, psi(i,j), to random numerical noise, and higher-order, smoothing derivative stencils are used to compute dn_tb_ho, via the expression, dn = 1/abs(grad(tb)).

I suggest that the few changes to DSD2D-FLS V9 described here be implemented in Quirk's CASH/Amrita wrapped DSD2D-FLS solver. In addition, the problems and results described in this report would provide data V&V for Quirk's CASH/Amrita wrapped DSD2D-FLS solver.

REFERENCES

[1] Bdzil, J.B., Aida, T, Henninger, R.J., Walter, J.W., "Test Problems for DSD3D," LA-14336 (2007).

[2] Hernandez, A., Bdzil, J.B., Stewart, D.S., "An MPI parallel level-set algorithm for propagating front curvature dependent detonation shock fronts in complex geometries," Combust. Theory and Modelling, Vol 17, No. 1, 109-141 (2013).

APPENDIX-A: ASSORTED OTHER SOLUTIONS OF THE 30-DEGREE RATE STICK

In this appendix, I display a collection of other solutions constructed in this study. All these DSD2D-FLS solutions used the modified driver.f, ibextra.f and ibupdate.f that I described above. The DSD model is the same as used above, with the exception that some of the omega_s and omega_c values are different.

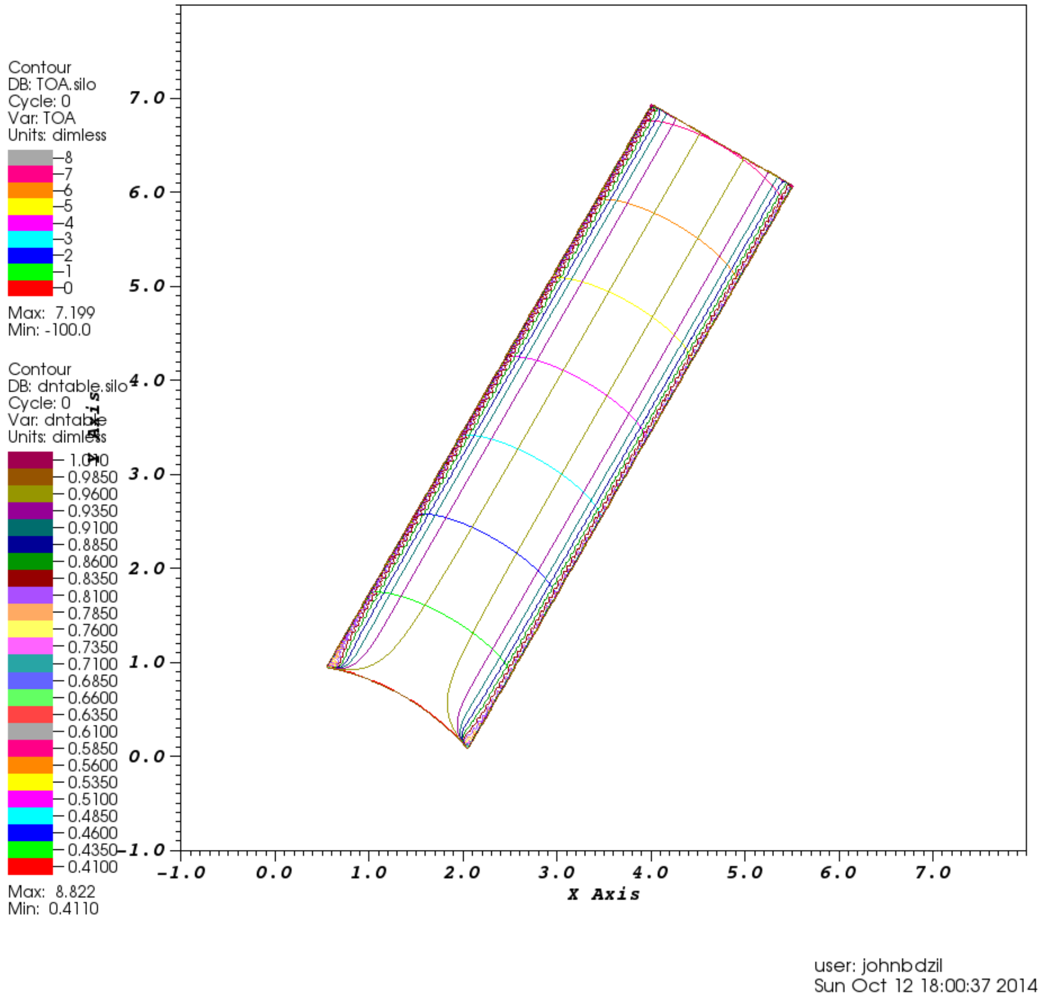

Figure-10a) Contours of the burn time field, tb(i,j), and the dntable field, dn(i,j), for the problem of detonation in a rate stick. To explore how the orientation of the rate stick relative to the mesh affects the results, the rate stick's axis of symmetry is tilted by 30 degrees off of the vertical direction. Only minor noise in the dn(i,j) field is observed near the side boundaries of the rate stick. Omega_s = 50-degrees and omega_c = 55-degrees. The numerical resolution is nxpts = nypts = 704.

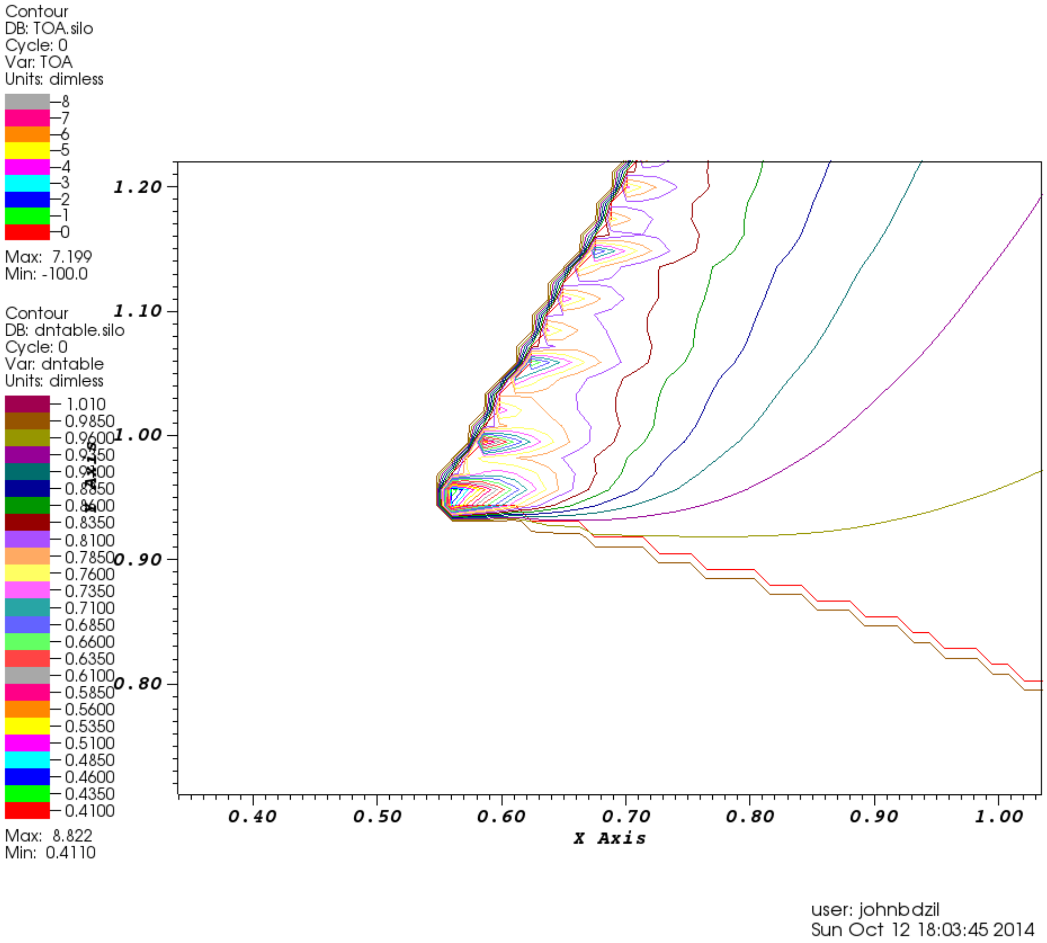

Figure-10b) Expanded view in the lower left hand corner showing contours of the burn time field, tb(i,j), and the dntable field, dn(i,j), for the problem of detonation in a rate stick. To explore how the orientation of the rate stick relative to the mesh affects the results, the rate stick's axis of symmetry is tilted by 30 degrees off of the vertical direction. Only minor noise in the dn(i,j) field is observed near the side boundaries of the rate stick. Omega_s = 50-degrees, and omega_c = 55-degrees. The numerical resolution is nxpts = nypts = 704.

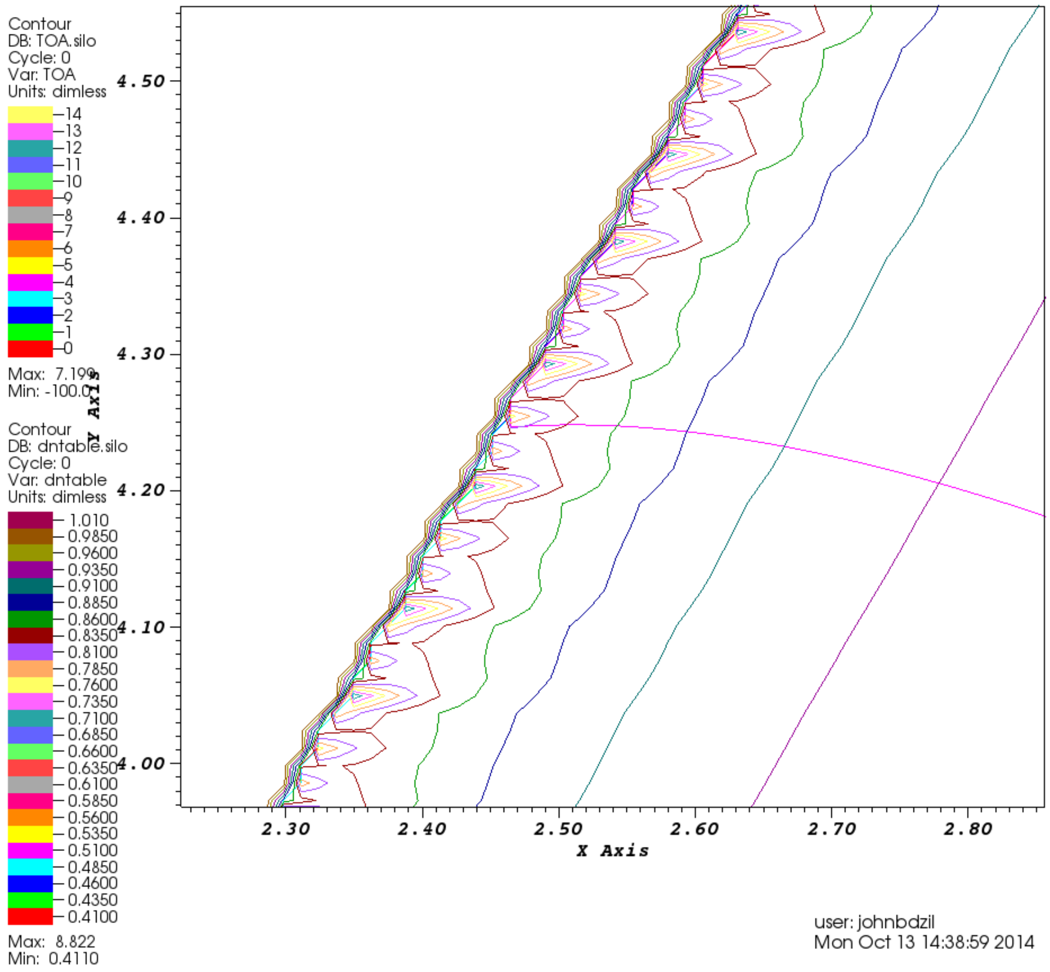

Figure-10c) Expanded view along the lateral boundary showing contours of the burn time field, tb(i,j), and the dntable field, dn(i,j), for the problem of detonation in a rate stick. To explore how the orientation of the rate stick relative to the mesh affects the results, the rate stick's axis of symmetry is tilted by 30 degrees off of the vertical direction. Only minor noise in the dn(i,j) field is observed near the side boundaries of the rate stick. Omega_s = 50-degrees, and omega_c = 55-degrees. The numerical resolution is nxpts = nypts = 704.

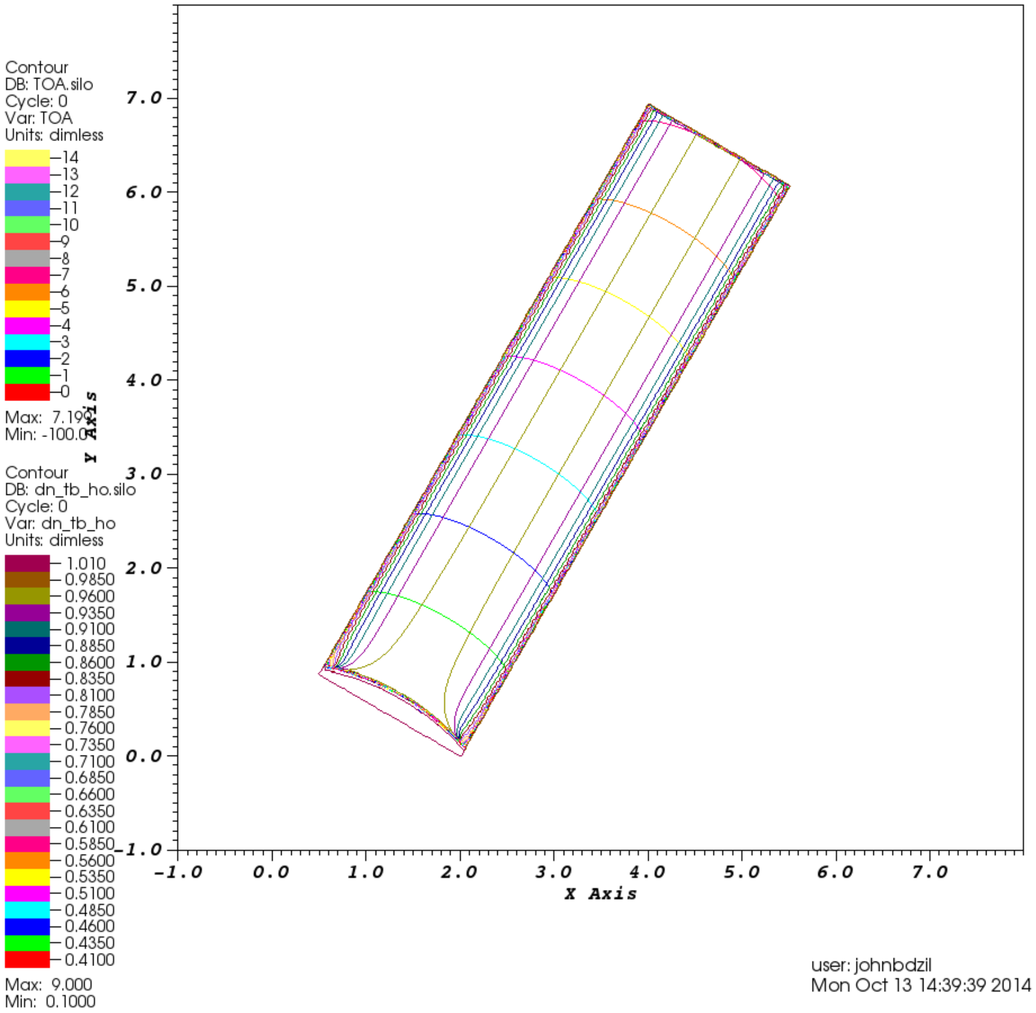

Figure-10d) Contours of the burn time field, tb(i,j), and the dn_tb_ho field, dn(i,j), for the problem of detonation in a rate stick. To explore how the orientation of the rate stick relative to the mesh affects the results, the rate stick's axis of symmetry is tilted by 30 degrees off of the vertical direction. Only minor noise in the dn(i,j) field is observed near the side boundaries of the rate stick. Omega_s = 50-degrees, and omega_c = 55-degrees. The numerical resolution is nxpts = nypts = 704.

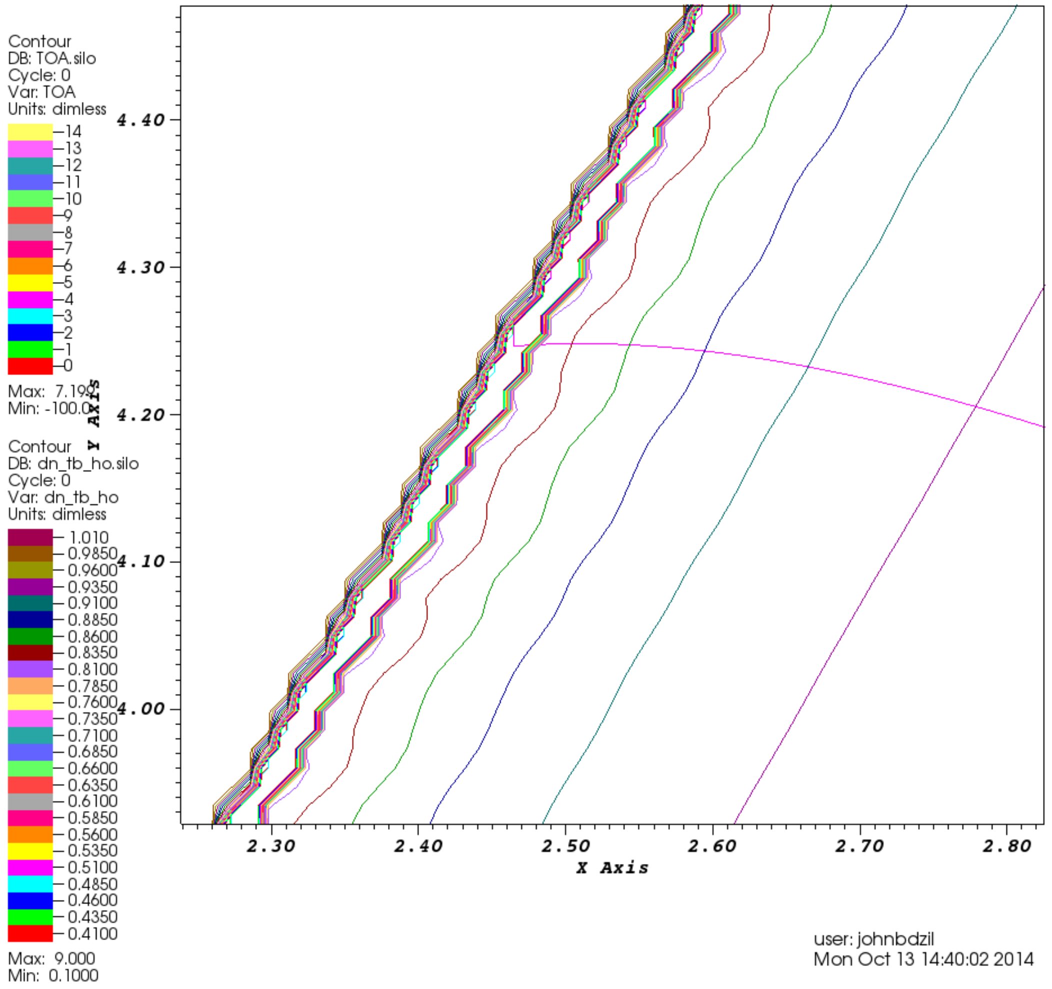

Figure-10e) Expanded view along the lateral boundary showing contours of the burn time field, tb(i,j), and the dn_tb_ho field, dn(i,j), for the problem of detonation in a rate stick. To explore how the orientation of the rate stick relative to the mesh affects the results, the rate stick's axis of symmetry is tilted by 30 degrees off of the vertical direction. Only minor noise in the dn(i,j) field is observed near the side boundaries of the rate stick. Omega_s = 50-degrees, and omega_c = 55-degrees. The numerical resolution is nxpts = nypts = 704.

APPENDIX-B: ASSORTED SIMULATIONS OF DETONATION IN AN ARC OF EXPLOSIVE

I begin by comparing the DSD2D-FLS solution of the 270-degree arc problem with the "exact" solution obtained from a Maple script. The solution developed with the Maple script solves the front propagation problem. That is, first the PDE for the detonation-front normal angle, phi(r,t), is solved as a function of the radial coordinate, r, and time, t, with a high-resolution, error controlled PDE solver available in Maple. Then in a second step, the motion of the front over a Cartesian grid is solved for with an ODE solver. Displayed in Figure-11) is a comparison of the DSD2D-FLS simulation for the case omega_s = 0.0, omega = 60-degrees and run at nxpts = nypts = 1408 points, with the "exact" solution of the problem.

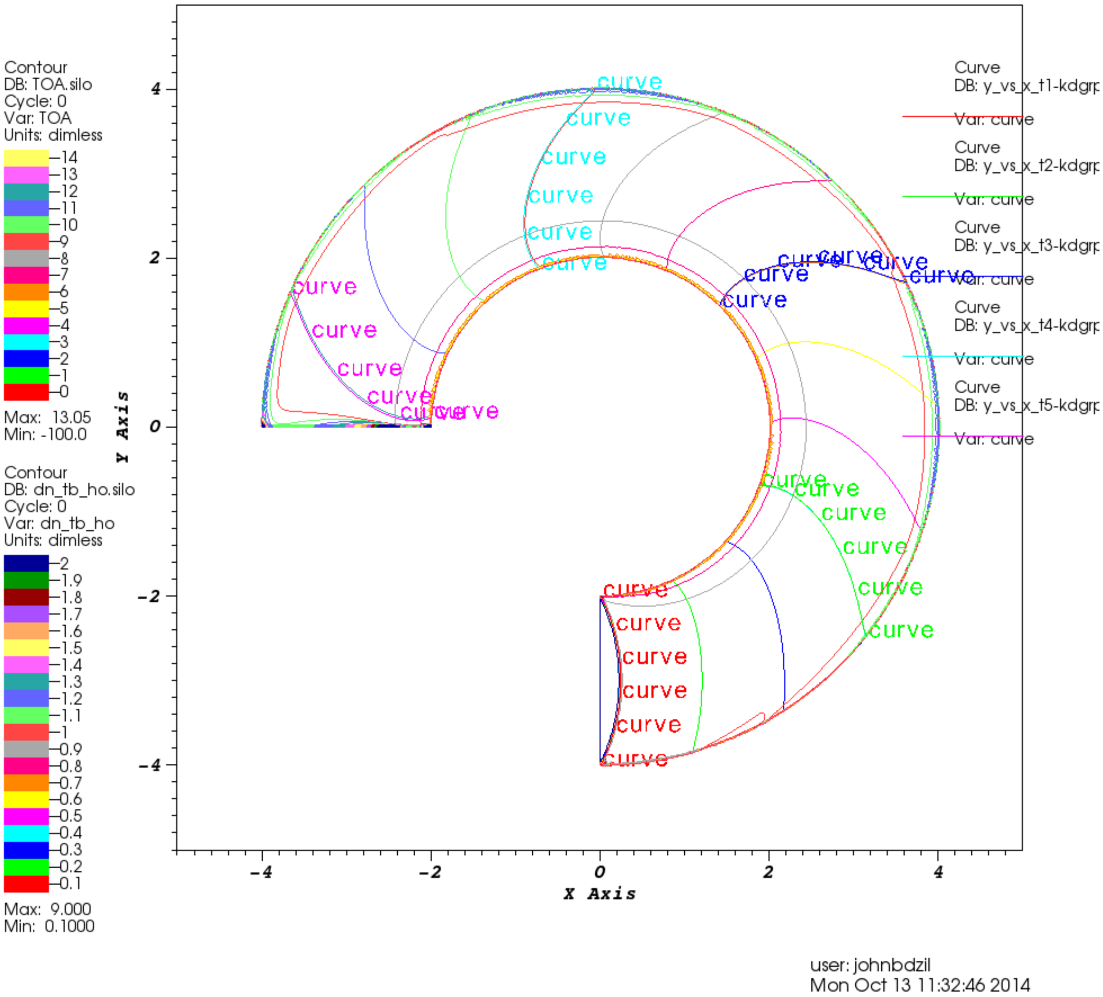

Figure-11) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc run at a resolution of nxpts = nypts = 1408 points. The DSD boundary angles are omega_s = 0 degrees, and omega_c = 60 degrees. Also displayed are the very-high resolution Maple script generated solutions of the front evolution equations, which can be considered to be the exact solution (labeled as "curve"). The agreement between these DSD2D-FLS simulation results and the exact solution for this initial value problem are good. The dn(i,j) field is computed with a smoothing stencil for the gradient applied to tb(i,j), designated as dn_tb_ho.

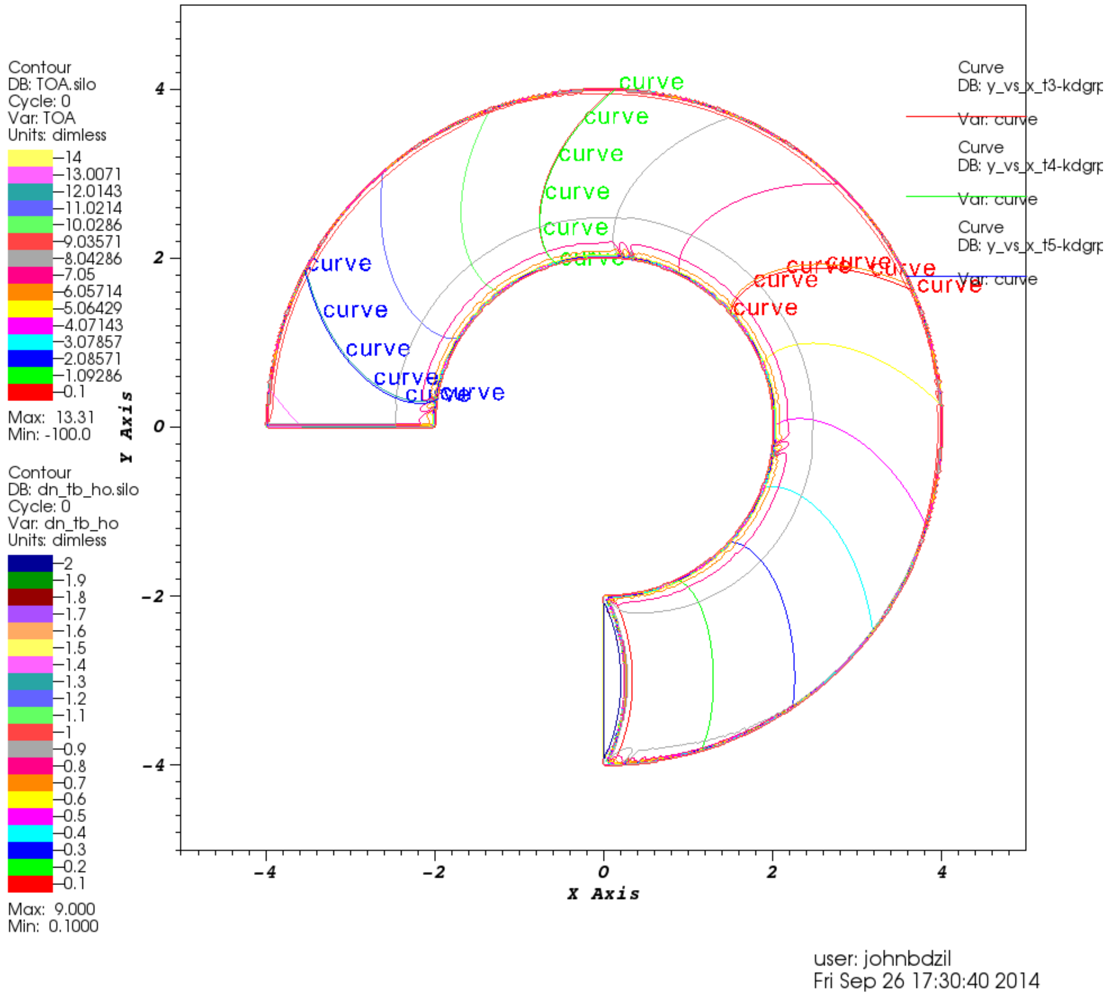

Figure-12a) Contours of the tb(i,j) field and dn(i,j) field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 44 degrees and omega_c = 45 degrees. Also displayed are the very-high resolution Maple script generated solutions of the front evolution equations, which can be considered to be the exact solution (labeled as "curve"). The agreement between these DSD2D-FLS simulation results and the exact solution for this initial value problem are good. There is a slight amount of noise visible in the dntable generated dn(i,j) contours near the 90 degree and 180 degree locations, and is likely related to the closeness of omega_s and omega_c. The dn(i,j) field is computed with a smoothing stencil for the gradient applied to tb(i,j), designated as dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

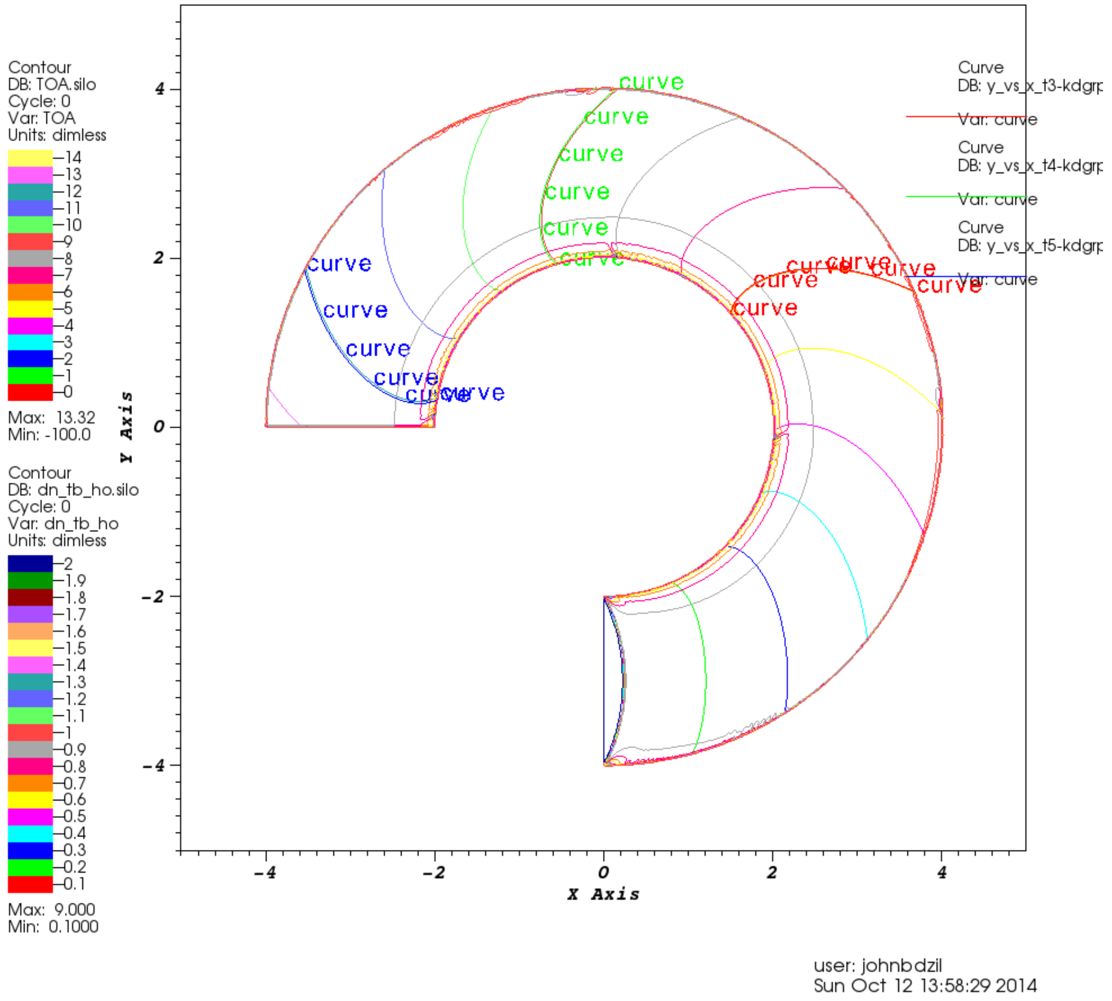

Figure-12b) Here we increase the resolution to nxpts=nypts=1408 points from the nxpts=nypts=704 points used to produce Figure-12a). The level of the noise at the 90 degree and 180 degree locations is reduced. Otherwise, things look much the same as they do in Figure-12a). The numerical resolution used is nxpts = nypts = 1408.

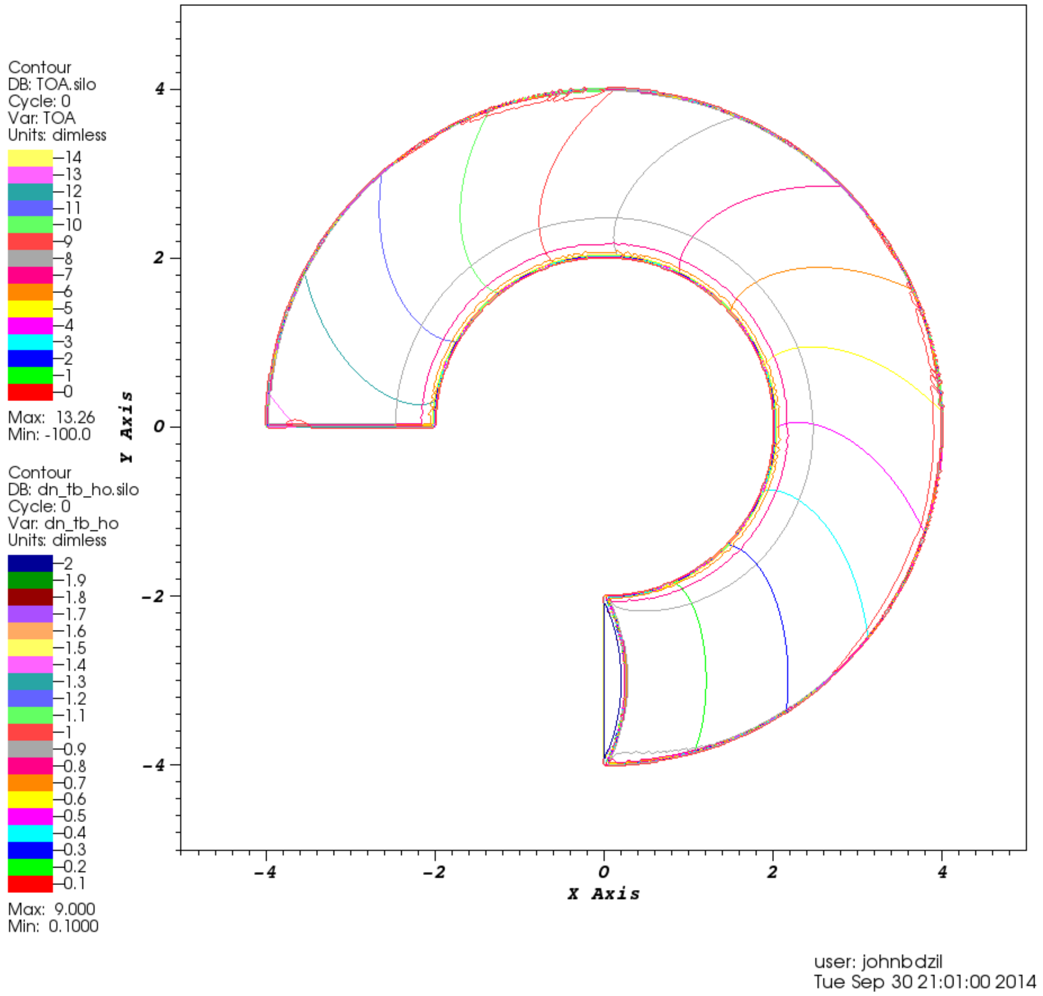

Figure-13) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 44 degrees and omega_c = 50 degrees. The dn_tb_ho contours are displayed for dn(i,j). The numerical resolution used is nxpts = nypts = 704.

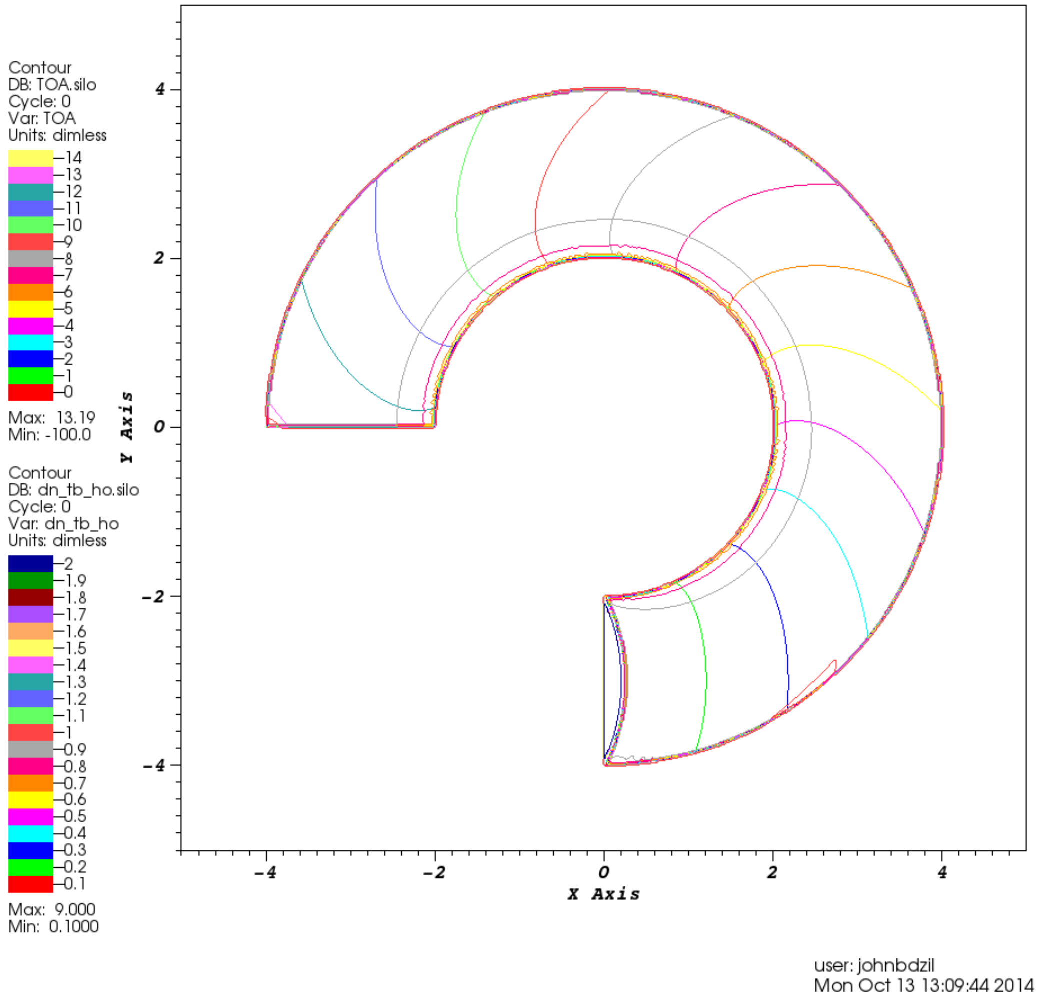

Figure-14a) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 50 degrees, and omega_c = 55 degrees. The contours of dn are dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

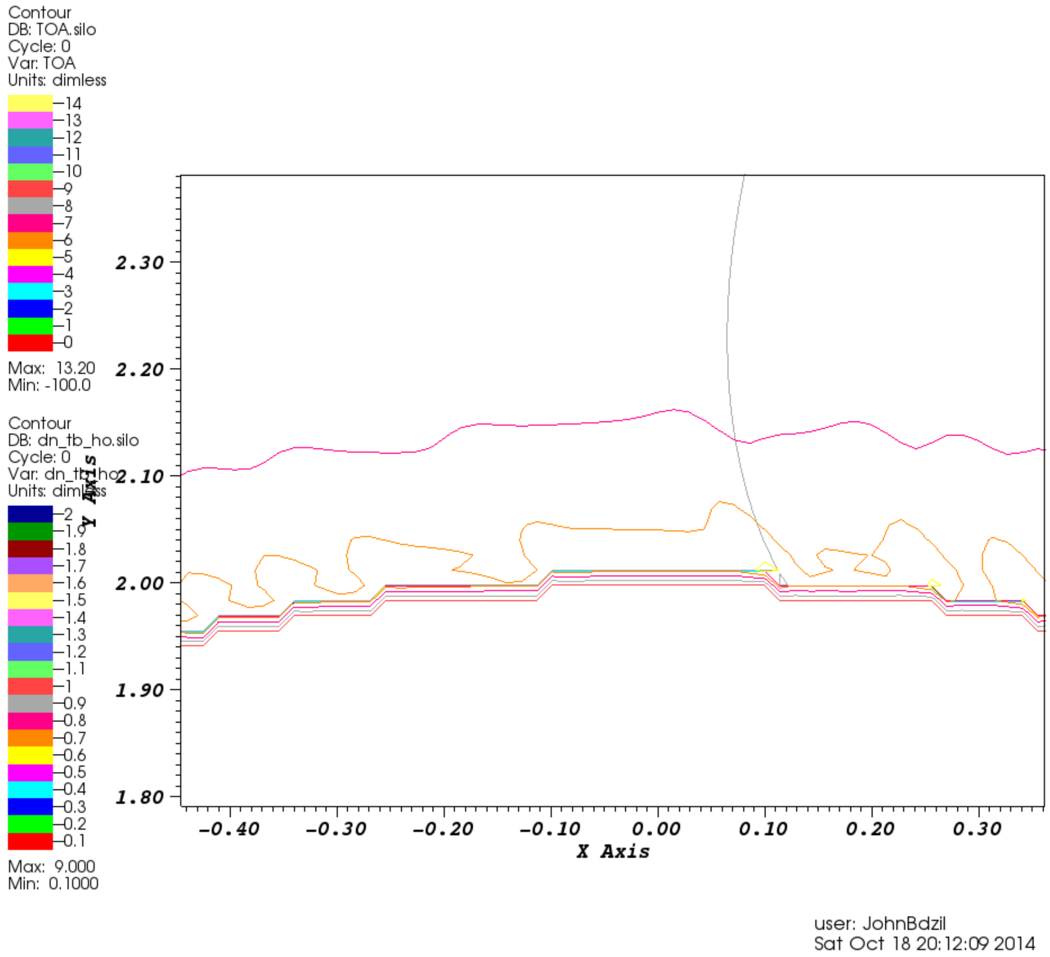

Figure-14b) Expanded view of contours (near the 90-degree point) of the tb(i,j) field and dn(i,j) field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 50 degrees and omega_c = 55 degrees. The contours of dn were generated from dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

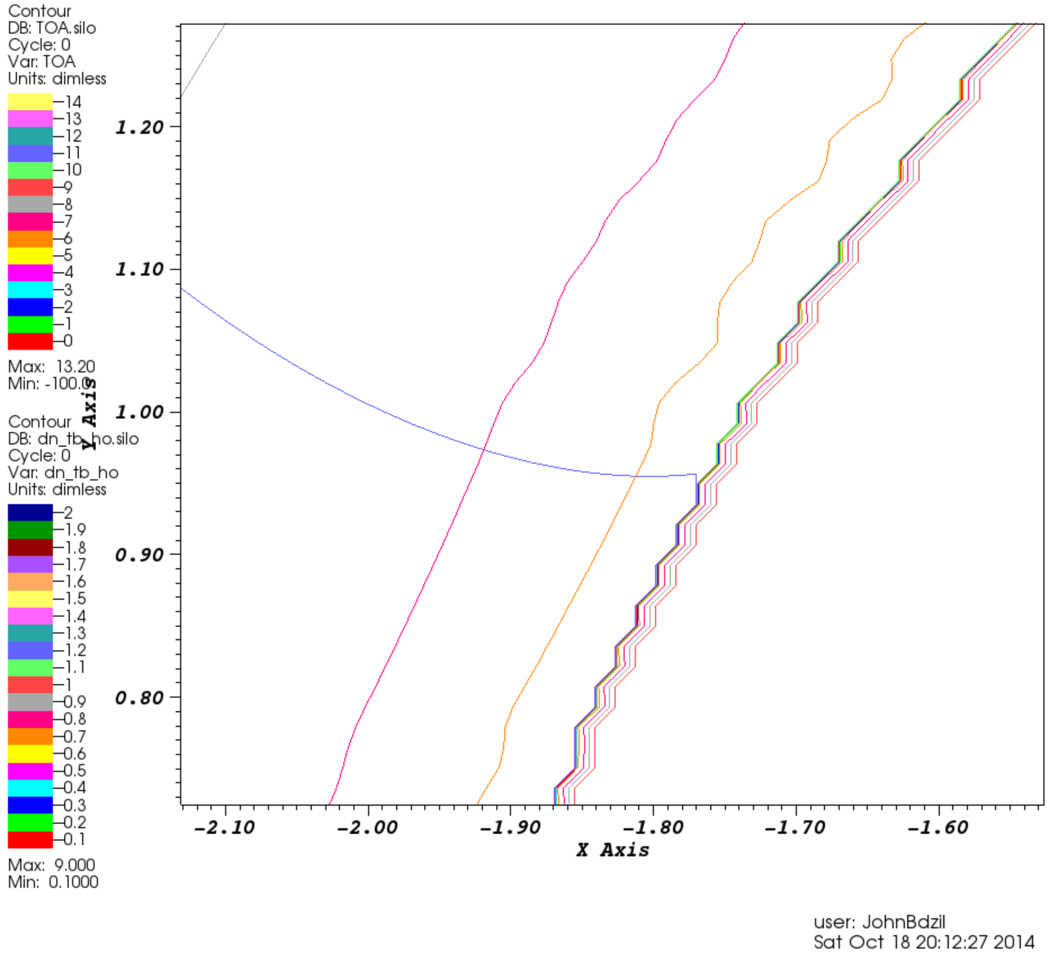

Figure-14c) Expanded view of contours (near the 150-degree point) of the tb(i,j) field and dn(i,j) field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 50 degrees and omega_c = 55 degrees. The contours of dn were generated from dn_tb_ho.

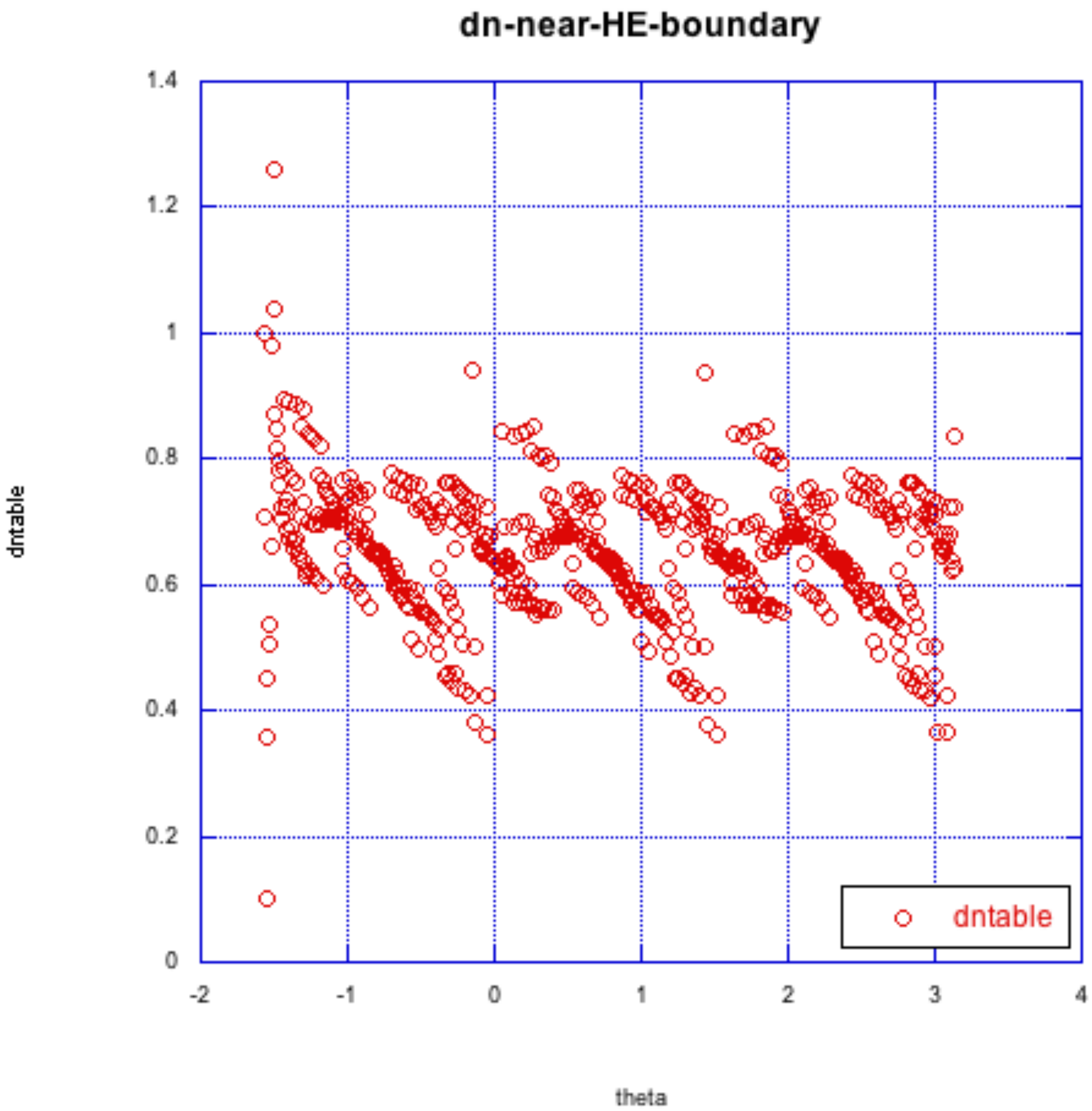

Figure-14d) Values of dn vs. the polar angle in radians, theta, are displayed. The computed values of dn at the mesh points within the narrow band of HE mesh points, 0.0 => psi(i,j) => -dx, near the inner boundary of the explosive are displayed for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 50 degrees and omega_c = 55 degrees. The values of dn were generated using the curvature of the level-set function, dn = 1 – 0.1*kappa, where kappa is the 2D slab geometry curvature. A small number of points above the 1.4 dn level are omitted from the plot.

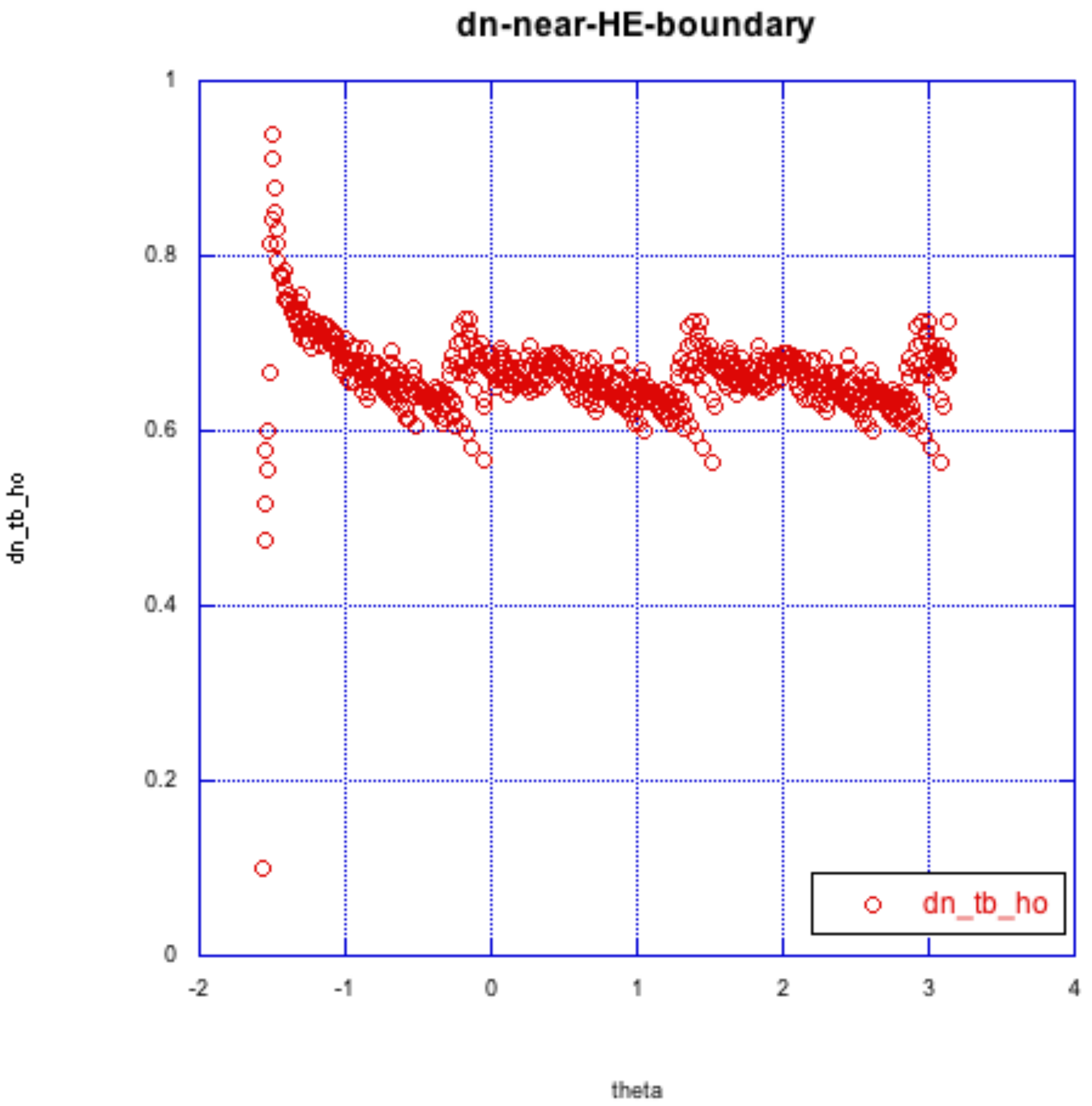

Figure-14e) Values of dn vs. the polar angle in radians, theta, are displayed. The computed values of dn at the mesh points within the narrow band of HE mesh points, 0.0 => psi(i,j) => -dx, near the inner boundary of the explosive are displayed for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 50 degrees and omega_c = 55 degrees. The contours of dn were generated from dn_tb_ho. All the data points are displayed.

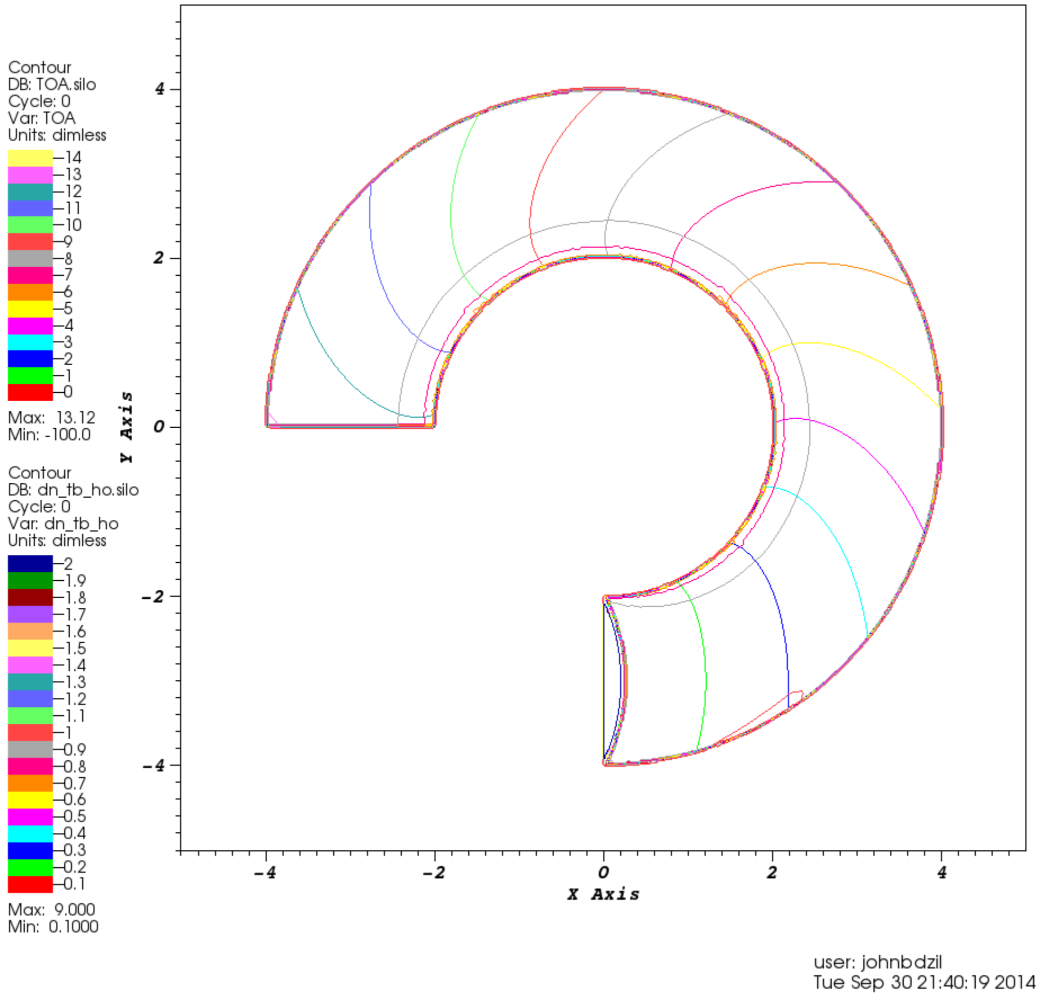

Figure-15) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 55 degrees and omega_c = 60 degrees. The contours of dn were generated from dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

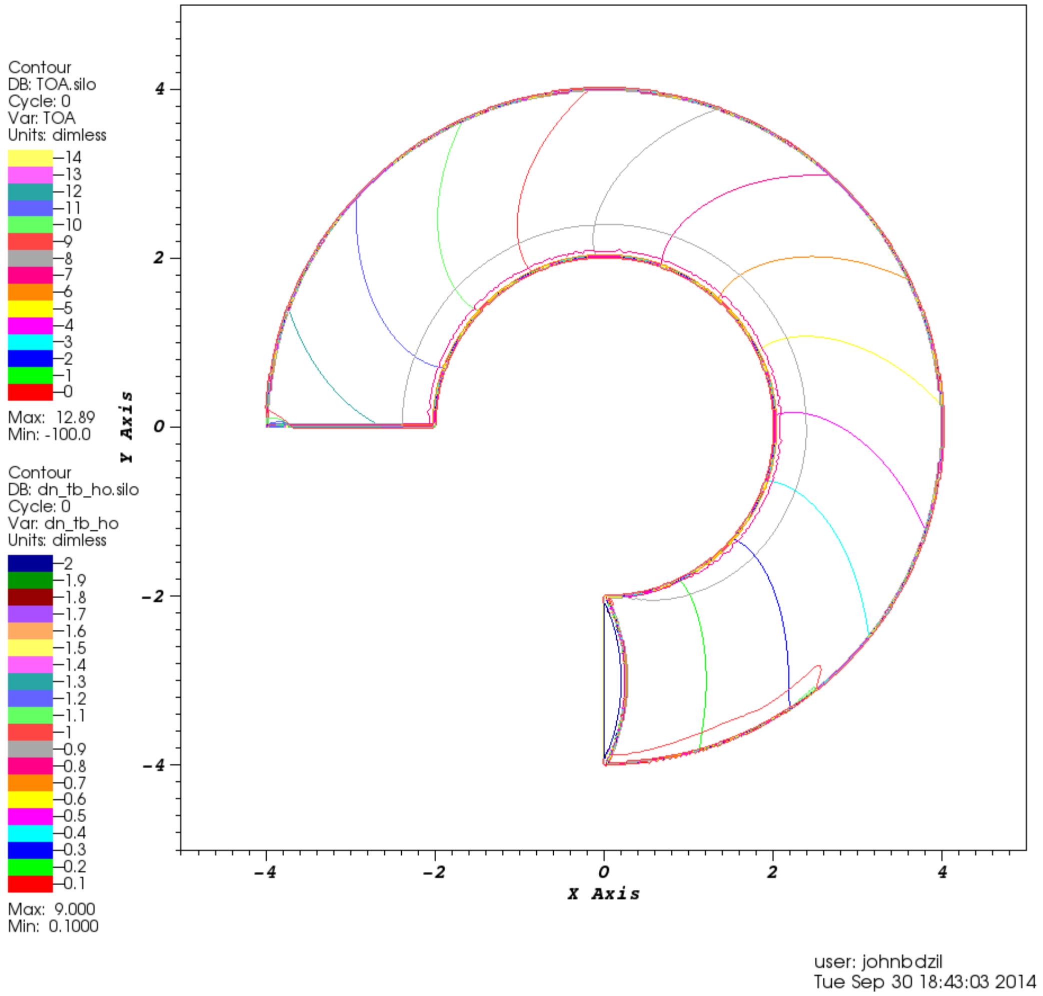

Figure-16) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 55 degrees and omega_c = 70 degrees. The contours of dn were generated from dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

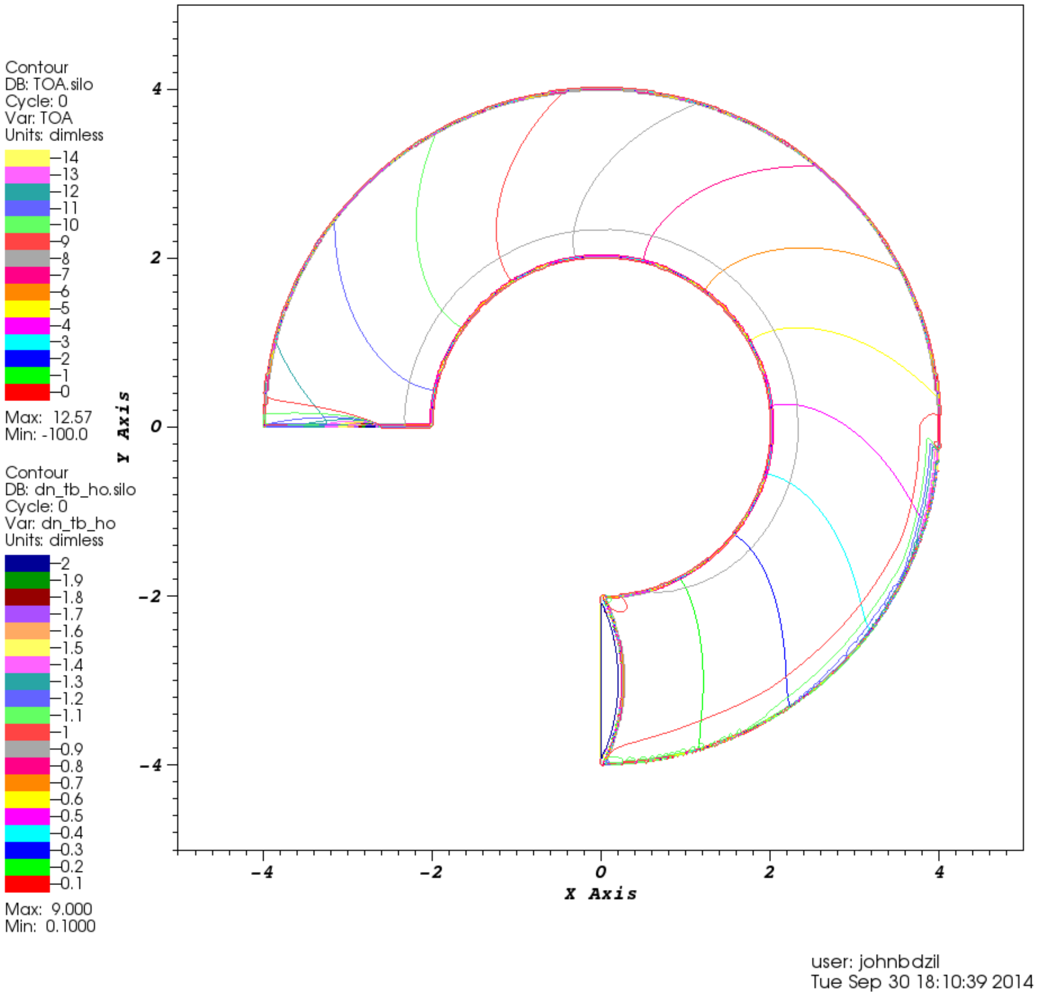

Figure-17) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 55 degrees and omega_c = 80 degrees. The contours of dn were generated from dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.

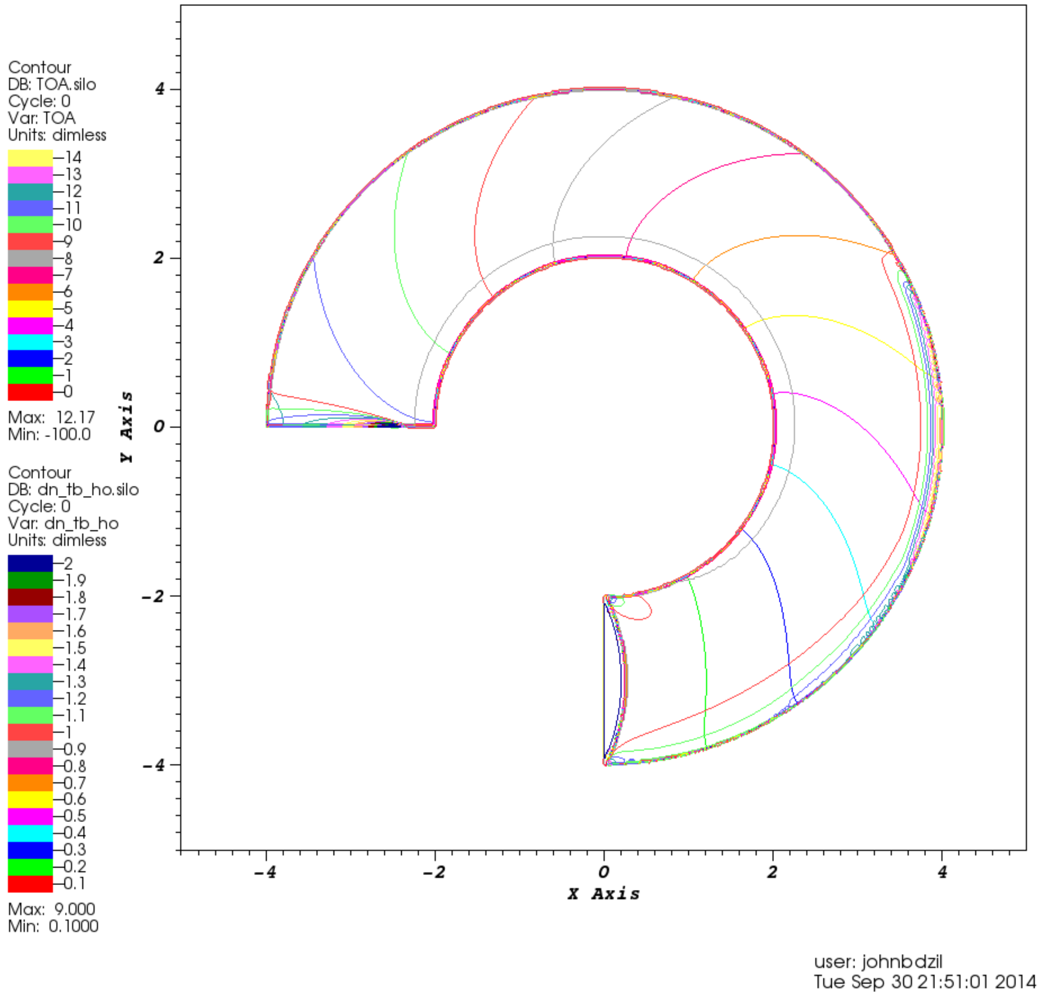

Figure-18) Contours of the tb(i,j)-field and dn(i,j)-field for the DSD2D-FLS simulation of detonation in a 270-degree, slab geometry explosive arc. The DSD boundary angles are omega_s = 55 degrees and omega_c = 90 degrees. The contours of dn were generated from dn_tb_ho. The numerical resolution used is nxpts = nypts = 704.