Shock Finding Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

BoSSS.Solution.CompressibleFlowCommon.ShockFinding

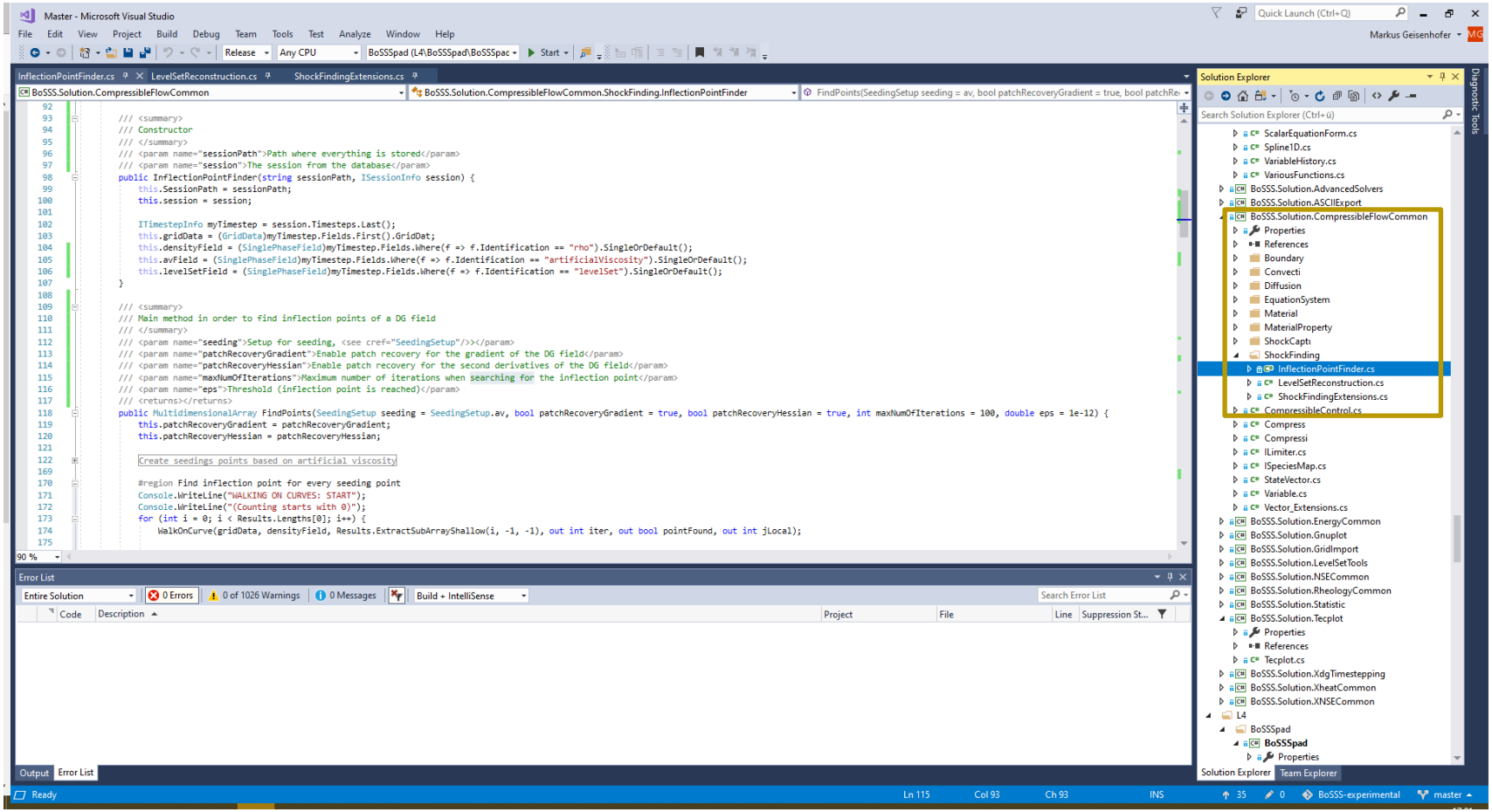
InflectionPointFinder.cs
LevelSetReconstruction.cs
ShockFindingExtensions.cs

TECHNISCHE UNIVERSITÄT DARMSTADT

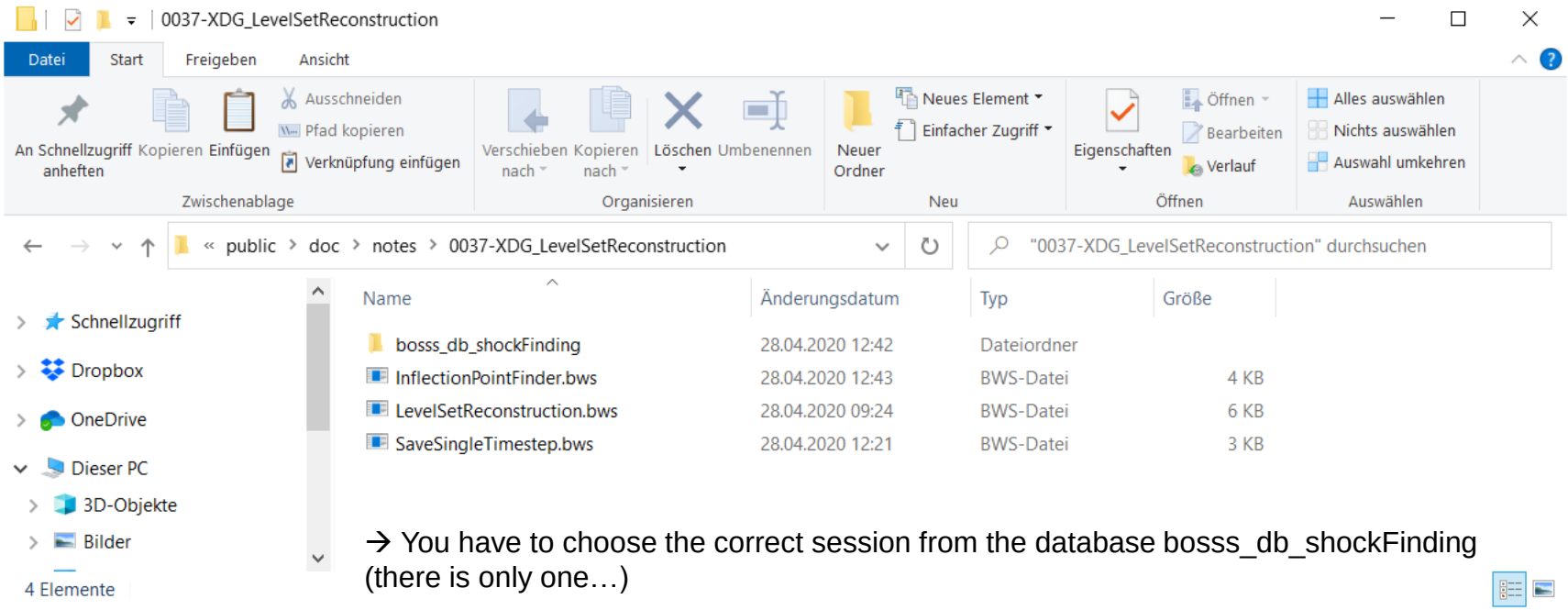
Where to find in BoSSS?

BoSSS.Solution.CompressibleFlowCommon.ShockFinding

TECHNISCHE UNIVERSITÄT DARMSTADT



Do you want to try?



→ You have to choose the correct session from the database bosss_db_shockFinding (there is only one…)

TECHNISCHE
UNIVERSITÄT
DARMSTADT

InflectionPointFinder.cs

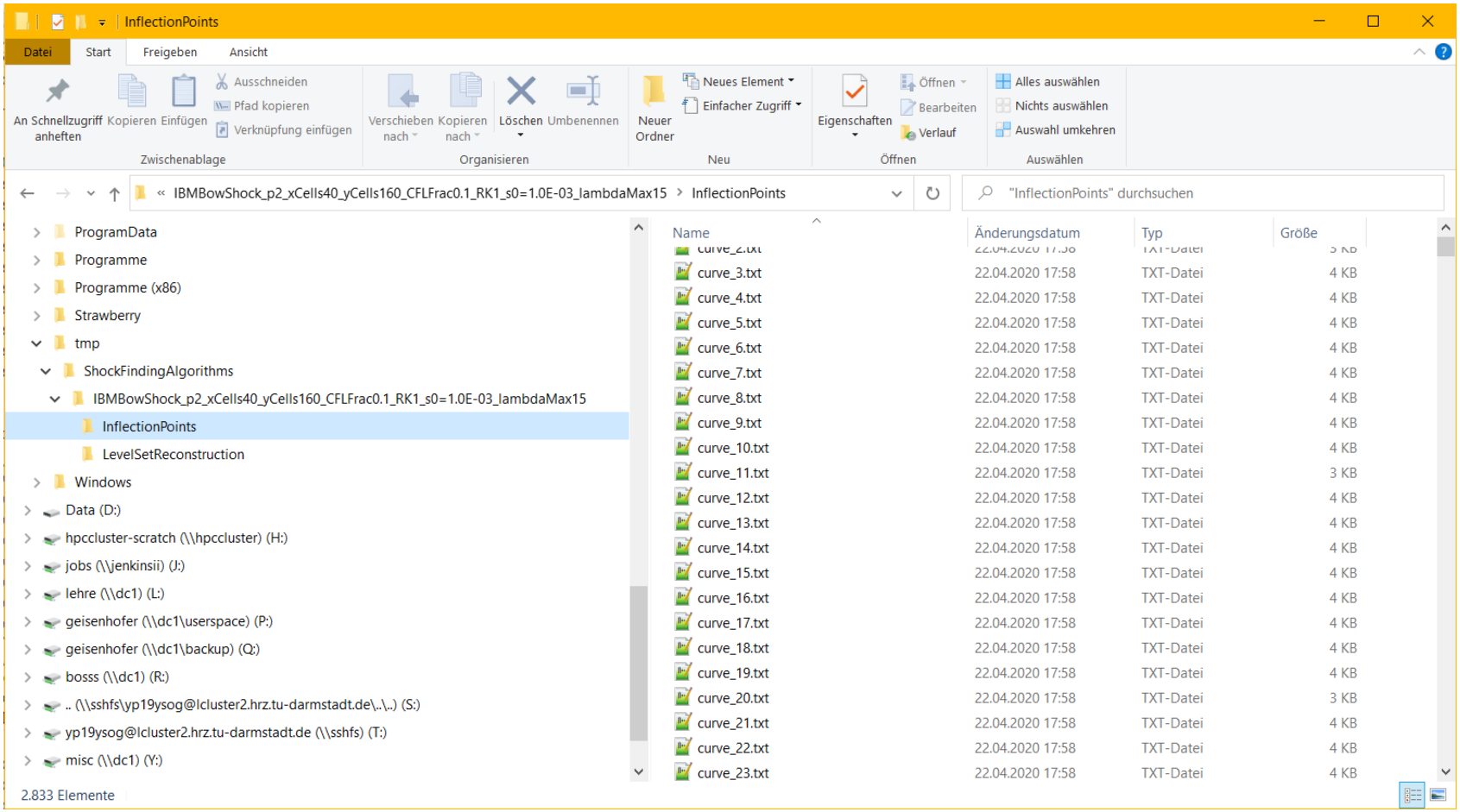
InflectionPointFinder.cs

BoSSSPad (InflectionPointFinder.bws*)

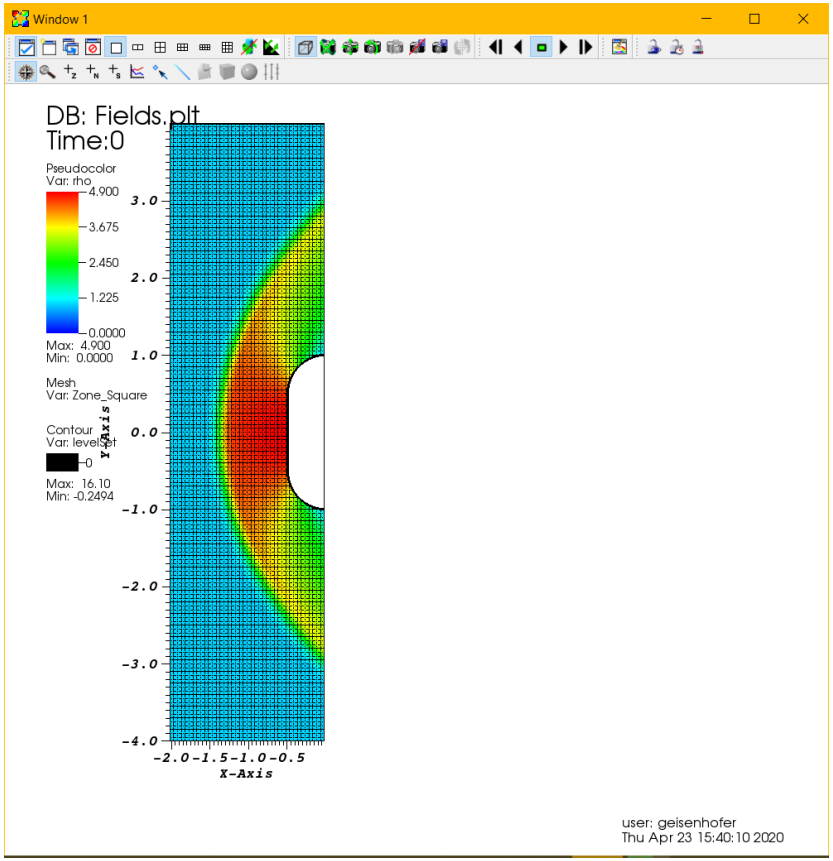
File Edit Commands

```
 6: IBMBowShock_P2 IBMBowShock_p2_xCells40_yCells160_CFLFrac0.1_RK1_s0=5.0E-04_lambdaMax15 01/14/2020 18:01:48 c8bec0df...
 7: IBMBowShock_P2 IBMBowShock_p2_xCells40_yCells160_CFLFrac0.1_RK1_s0=1.0E-03_lambdaMax15 01/14/2020 17:45:38 895907a0...
 8: IBMBowShock_P2 IBMBowShock_p2_xCells40_yCells160_CFLFrac0.1_RK1_s0=1.0E-04_lambdaMax15 01/15/2020 06:24:30 1004422c...
}
13
14 // Select sessions
15 List<ISessionInfo> sessions = new List<ISessionInfo>();
16 sessions.Add(database.Sessions.Pick(7));
17
18 // Create Direcotories
19 string mainPath      = @"C:\tmp\ShockFindingAlgorithms\";
20 string directoryName = @"InflectionPoints";
21 string[] sessionPathsInflec = ShockFindingExtensions.CreateDirectories(mainPath, directoryName, sessions);
22 //ShockFindingExtensions.EmptyDirectories(mainPath, direcotoryName, sessions);
23
24 // Select seeding setup
25 SeedingSetup seeding = SeedingSetup.av;
26
27 // Find the inflection points
28 InflectionPointFinder[] finders = new InflectionPointFinder[sessions.Count];
29
30 foreach(ISessionInfo session in sessions){
31     int count                    = 0;
32
33     InflectionPointFinder finder = new InflectionPointFinder(sessionPathsInflec[count], session);
34
35     finder.FindPoints(seeding: seeding, patchRecoveryGradient: true, patchRecoveryHessian: true);
36     finder.Plot(plotDGFields: true, plotSeedingsPoints: true, plotInflectionsPoints: true, plotCurves: true, plotStartEndPairs: true);
37     finder.Results.SaveResults(finder.SessionPath);
38     finder.ResultsExtended.SaveResults(finder.SessionPath);
39
40     finders[count] = finder;
41
42     count++;
43 }
Total number of seeding points: 1412
WALKING ON CURVES: START
(Counting starts with 0)
Patch recovery of field gradientX started...
finished
Patch recovery of field gradientY started...
```

InflectionPointFinder.cs



InflectionPointFinder.cs

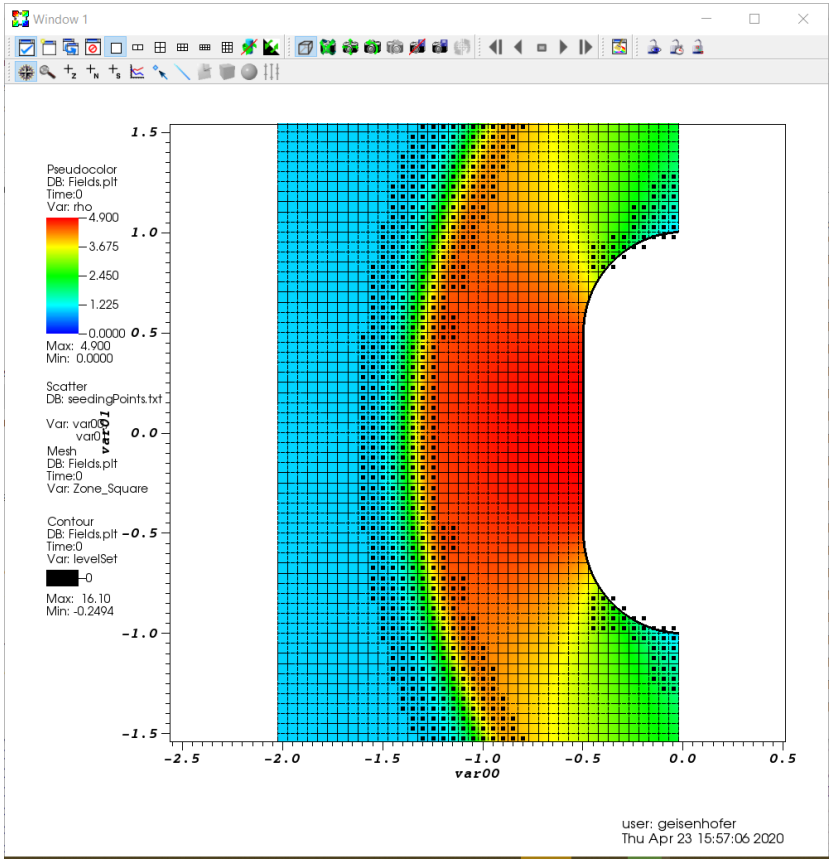


- CNS (IBM) simulation
  - 2nd order
  - With artificial viscosity

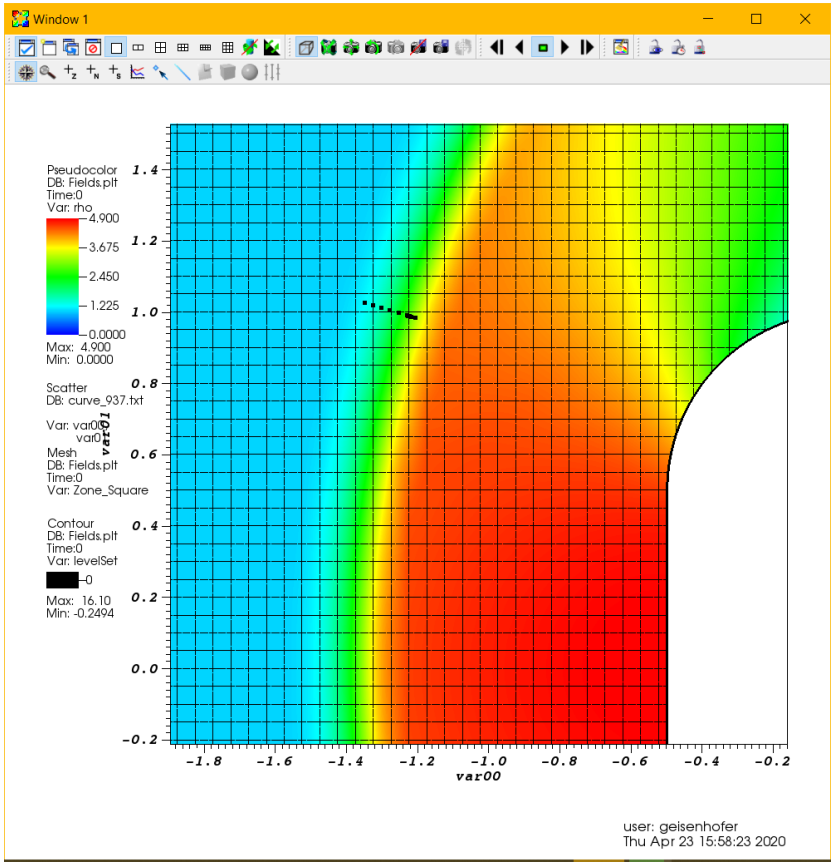
InflectionPointFinder.cs

TECHNISCHE UNIVERSITÄT DARMSTADT

seedingPoints.txt



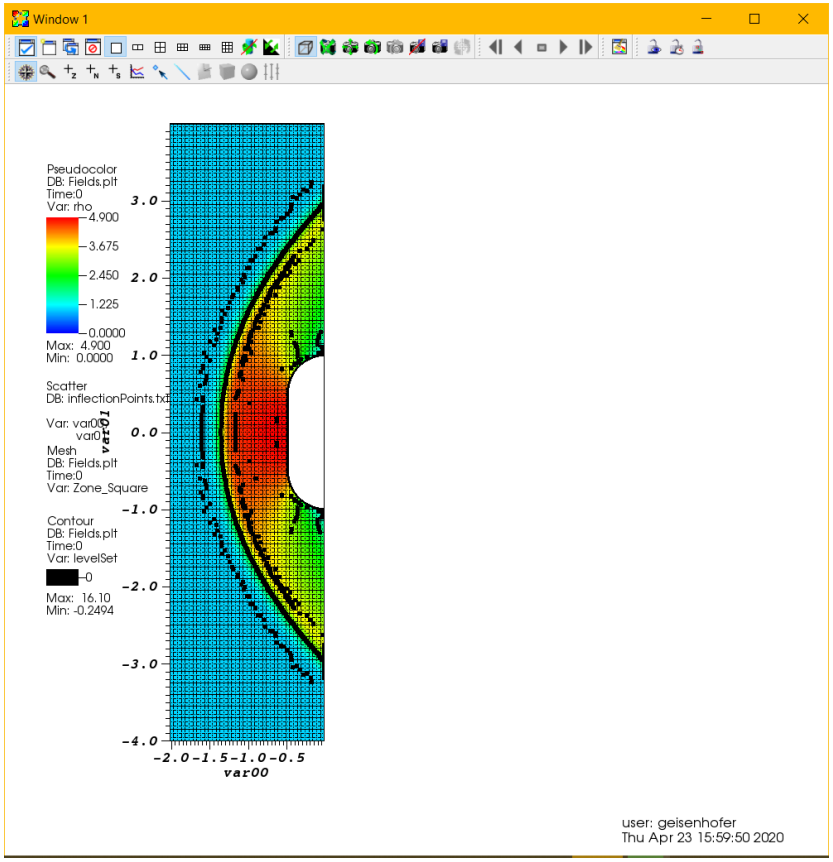
curves_*.txt



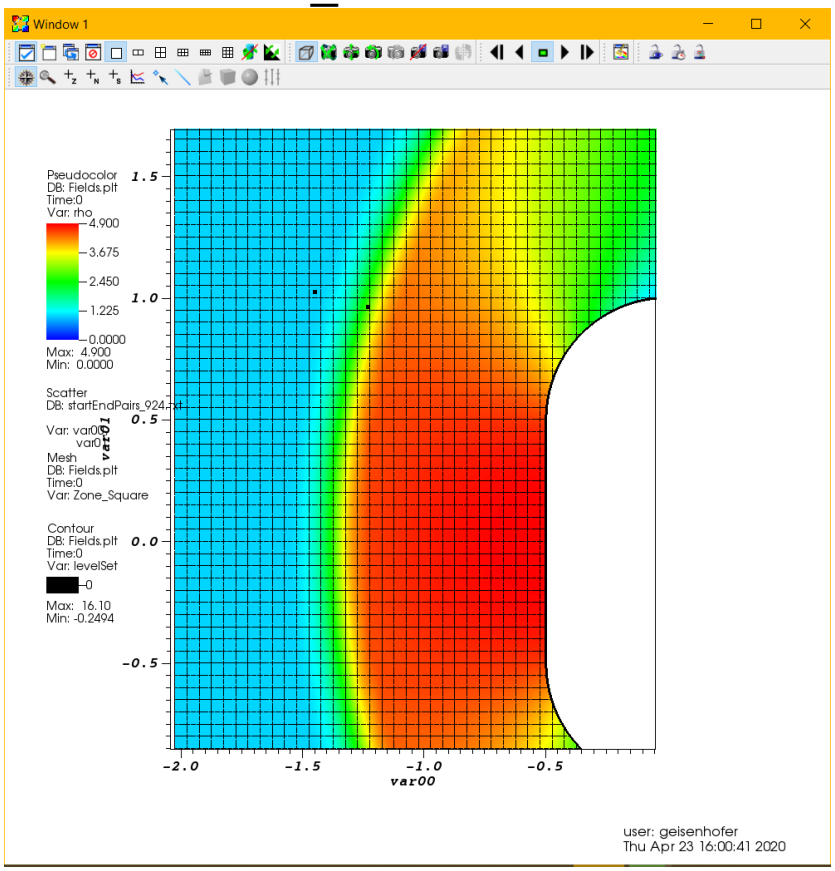
InflectionPointFinder.cs

TECHNISCHE UNIVERSITÄT DARMSTADT

inflectionPoints.txt



startEndPairs_*.txt



TECHNISCHE
UNIVERSITÄT
DARMSTADT

LevelSetReconstruction.cs

LevelSetReconstruction.cs

TECHNISCHE UNIVERSITÄT DARMSTADT

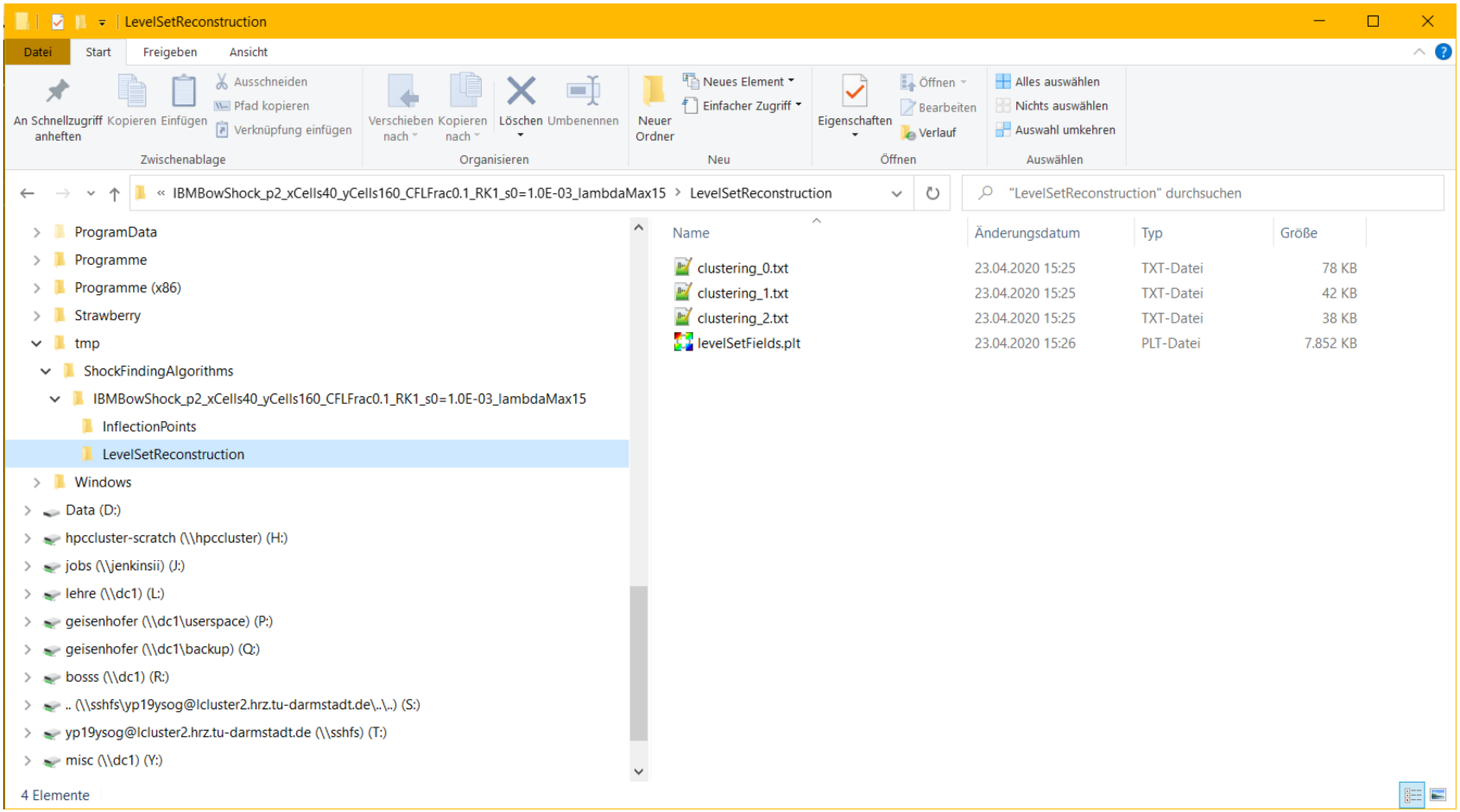
BoSSSPad (LevelSetReconstruction.bws*)

File Edit Commands

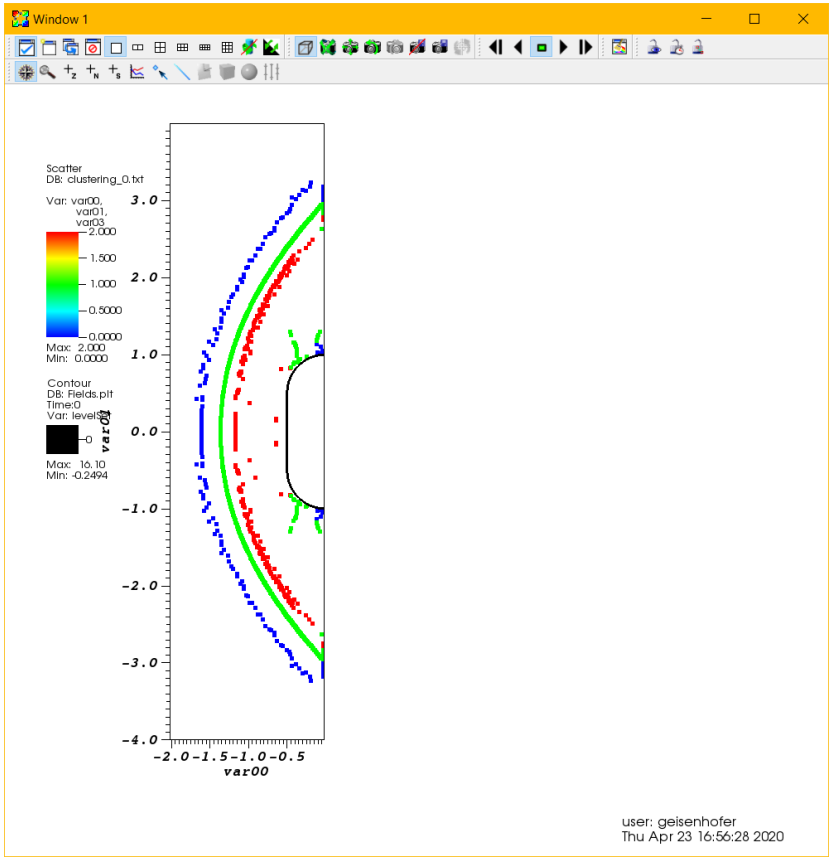
```
// Select sessions
List<ISessionInfo> sessions = new List<ISessionInfo>();
sessions.Add(database.Sessions.Pick(7));

// Create Direcotories
string mainPath = @"C:\tmp\ShockFindingAlgorithms\";
string[] sessionPathsRecon = ShockFindingExtensions.CreateDirectories(mainPath, @"LevelSetReconstruction", sessions);
ShockFindingExtensions.EmptyDirectories(mainPath, @"LevelSetReconstruction", sessions);

string[] sessionPathsInflec = ShockFindingExtensions.CreateDirectories(mainPath, @"InflectionPoints", sessions);
LevelSetReconstruction[] recons = new LevelSetReconstruction[sessions.Count];
InflectionPointFinder[] finders = null;

foreach(ISessionInfo session in sessions){
    int count                   = 0;

    MultidimensionalArray results;
    MultidimensionalArray resultsExtended;
    if (finders == null){
        results         = ShockFindingExtensions.LoadResults(sessionPathsInflec[count]);
        resultsExtended = ShockFindingExtensions.LoadResultsExtended(sessionPathsInflec[count]);
    } else {
        results         = finders[count].Results;
        resultsExtended = finders[count].ResultsExtended;
    }

    LevelSetReconstruction lsr = new LevelSetReconstruction(sessionPathsRecon[count], session, results, resultsExtended);

    // Clustering zero (density)
    MultidimensionalArray clusteringZero = lsr.CreateClustering_Density(3, new double[] {1.0, 2.7, 5.0});
    lsr.SaveClusteringToTextFile(clusteringZero);
    MultidimensionalArray clusterZero = lsr.SelectCluster(clusteringZero, clusterToSelect: 1);

    // Clustering two (artificial viscosity)
    MultidimensionalArray clusteringOne = lsr.CreateClustering_AV(clusterZero, 2, new double[] {0.0, 0.0});
    lsr.SaveClusteringToTextFile(clusteringOne);
    MultidimensionalArray clusterOne = lsr.SelectCluster(clusteringOne, clusterToSelect: 0);

    // Clustering three (eliminate boundary cells)
    MultidimensionalArray clusteringTwo = lsr.CreateClustering_Boundary(clusterOne);
    lsr.SaveClusteringToTextFile(clusteringTwo);

    // Reconstruct level set field
    lsr.ReconstructLevelSet(patchRecovery: true, continuous: true);

    // Plot level set fields
    lsr.PlotFields();

    recons[count] = lsr;

    count++;
}
```

```
CreateClustering_Density: START
CreateClustering_Density: END
```

LevelSetReconstruction.cs



LevelSetReconstruction.cs

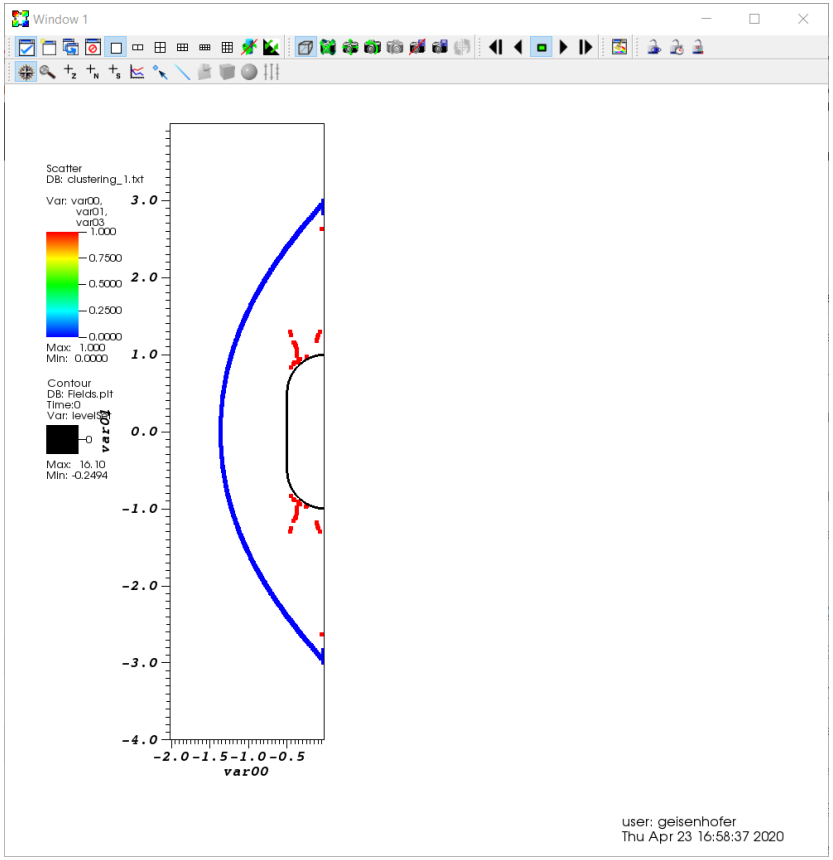


Clustering 0:

- By density

→ Select green cluster

LevelSetReconstruction.cs

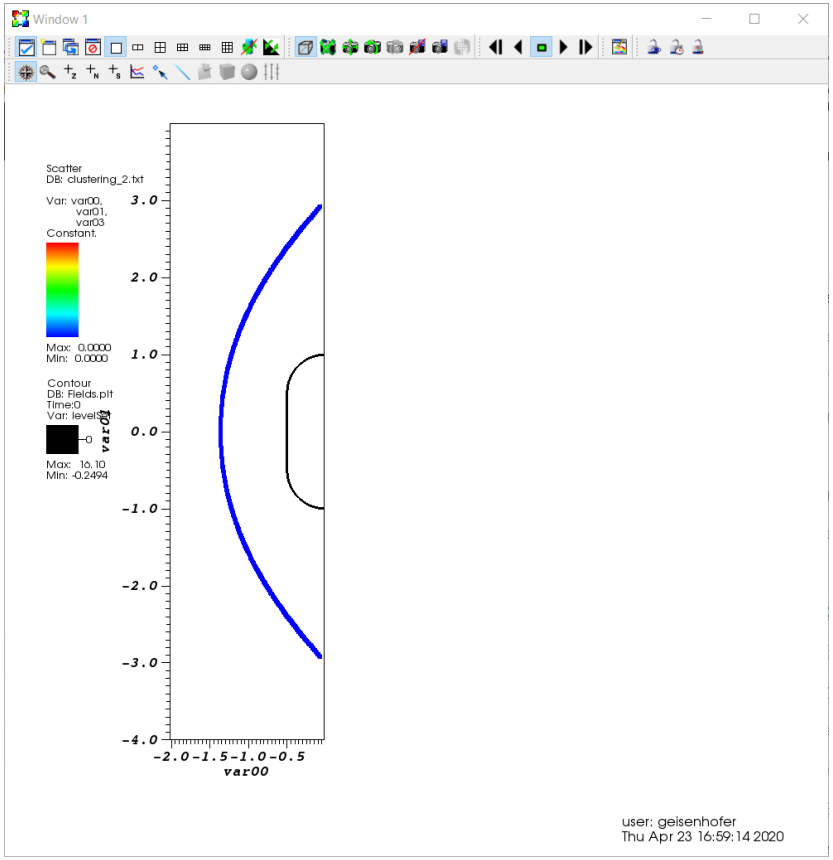


Clustering 1:

- By artificial viscosity

→ Select blue cluster

LevelSetReconstruction.cs



Clustering 2:

- Eliminate boundary cells