Recent Progresses on a Meshless Euler Solver for Compressible Flows

Zhaowen Duan¹ and Z.J. Wang²
University of Kansas, Lawrence, KS, 66046

Bruce Vu³
NASA Kennedy Space Center, FL 32899

The conventional CFD solvers depend on a mesh to discretize the domain. Due to the flexible nature of meshless methods, which do not require a mesh, we re-examine several meshless solvers for possible applications to moving boundary problems. Like many mesh-dependent solvers, second order meshless solvers suffer from convergence problems for transonic and supersonic flows with shock waves. In this work we develop a convergent limiter following ideas for finite volume solvers. The meshless solver is tested with a supersonic flow in a channel and subsonic flow over an airfoil and rigid body, and machine zero convergence is achieved for both the testing cases. We plan to run more benchmark problems, and further extend the present meshless solver to handle moving boundary problems.

Nomenclature

\( c \) = shape parameter
\( C_p \) = pressure coefficient
\( \Delta t \) = time step
\( f, g \) = generic functions
\( F_1 \) = Flux vector in x-direction
\( F_2 \) = Flux vector in y-direction
\( G \) = Flux vector along a given direction
\( i \) = index of reference node
\( j \) = index of supporting node
\( k \) = index of neighboring nodes
\( N_1 \) = number of supporting nodes
\( n \) = time index during navigation
\( \Phi \) = Vector of conservative variables
\( r \) = distance between two nodes
\( T \) = temperature
\( x \) = position vector
\( \Phi \) = limiter
\( \phi \) = radial basis function
\( M_\infty \) = Mach number of the free stream

I. Introduction

Most numerical methods for computational fluid dynamics (CFD) simulations, including finite difference methods (FDM), finite volume methods (FVM), and finite element methods (FEM), rely on meshes with

---

¹ PhD student, Department of Aerospace Engineering, University of Kansas
² Spahr Professor and Chair, Department of Aerospace Engineering, 2120 Learned Hall, University of Kansas, Fellow of AIAA
³ Fluids Systems Lead, Design Analysis Branch, NASA Kennedy Space Center, FL 32899.
various connectivity. While those methods succeed in solving partial differential equations (PDEs), they have limits. The accuracy of numerical solutions of PDEs is sensitive to the mesh. One of the major challenges in CFD is the generation of a suitable mesh.\(^1\) Many mesh generation procedures often lack automation, which in turn require human intervention. For a complex configuration, the generation of a good quality mesh can be expensive in terms of human labor. It is sometimes expensive to apply the above approach in problems with complex configurations and with free or moving boundaries, for the mesh needs to be deformed or re-generated at each time step. On the other hand, meshless methods, which offer the potential of complete automation, often trigger the interest of researchers in CFD.\(^2\)–\(^9\) These methods completely discard the idea of a mesh for the spatial discretization of the governing PDEs. The only geometrical information for meshless methods is about nodes. No other information such as node-to-node or face-to-face connectivity is needed. Moreover, without the need of taking care of connectivity it is very convenient to automatically generate nodes in free or moving boundaries problems.\(^10\) Since meshless methods have a wide variety of applications, they also have other names, including meshfree, gridfree, gridless, and generalized finite difference.

Among various meshless methods, the radial basis functions (RBFs) based methods have become attractive for solving PDEs.\(^11\)–\(^13\) Although RBFs were originally developed for multivariate data and function interpolation, their truly meshfree nature has motivated researchers to employ them in solving PDEs. Since RBFs have shown the ability of solving PDEs on scattered point data with high accuracy and good efficiency, they have been applied in various engineering problems like structural dynamics\(^14\), fluid dynamics\(^15\)–\(^16\) and fluid structure interaction\(^17\).

Various meshless Euler solvers based on RBFs have been developed to solve inviscid compressible fluid flows.\(^18\),\(^19\) In the present study, a meshless Euler solver\(^18\) was combined with a limiter that is suitable for meshless schemes. The results show excellent convergence property for CFD problems which have steady state solutions. The algorithm consists two parts. The first part deals with the derivative approximation by using differential quadrature (DQ) method with RBFs as basis functions. The second part implements a Godunov’s type upwind scheme to evaluate the common fluxes. Euler equations are solved for compressible flow problems. To apply our methods on given geometrical bodies and visualize the solutions for moving boundary problems, we also develop a package with a graphical user interface (GUI). The GUI allows a user to input a geometry and other parameters for the simulation. Then a point set is generated around the input geometry and in the background. In the present paper, we focus on solving the steady problems with the automatically generated nodal set. Our ultimate purpose is to extend the meshless solver to handle moving boundary problems.

II. Numerical Method

1. Differential Quadrature Technique with Radial Basis Functions

Differential quadrature (DQ) is the approximation of derivatives by using weighted sums of function values. In this paper local RBF based differential quadrature (RBF-DQ) technique is used for calculating of derivatives. The partial derivatives at a reference point can be approximated by weighted linear sum of function values over a group of discrete neighboring nodes within its supporting domain. The weights could be determined by the interpolation of RBFs.

A. Basic Formulas of Differential Quadrature Technique

Consider a continuous function \(f : \Omega \rightarrow \mathbb{R}\). To estimate its \(m\)th order derivative at a reference node \(x_j\) a group of neighboring nodes \(x_k^j, j = 1, 2, \ldots, N_j\) are selected as supporting nodes. Given coordinates of the reference node in physical domain the KD-tree algorithm could be helpful to search \(N\) nearest neighbors. By local DQ technique the approximation of the \(m\)th order derivative in \(x\)-direction can be calculated as

\[
\frac{\partial^m f}{\partial x^m}(x_j) = \sum_{j=1}^{N_j} w_{j,m}^{(mx)} f(x_j)
\]  (1)

The weighting coefficients \(w_{j,m}^{(mx)}\) could be determined by solving the linear equations of the interpolation of RBFs

\[
\frac{\partial^m \varphi \|x_k^j - x_k^i\|}{\partial x_k^m} = \sum_{j=1}^{N_j} w_{k,j}^{(mx)} \varphi \|x_k^j - x_k^i\|, \quad k = 1, 2, 3, \ldots, N_j
\]  (2)

or in a matrix form

American Institute of Aeronautics and Astronautics
Supporting nodes are scattered around the reference node and its neighboring nodes.

In the DQ method, the weight 

\[
\begin{bmatrix}
\partial^m \varphi \left( \| x_i - x_j \| \right) \\
\partial x_i \\
\partial^m \varphi \left( \| x_i - x_j \| \right) \\
\partial x_i \end{bmatrix}
\end{bmatrix}
= \begin{bmatrix}
\varphi \left( \| x_i - x_j \| \right) \\
\varphi \left( \| x_i - x_j \| \right) \\
\cdots \\
\varphi \left( \| x_i - x_j \| \right)
\end{bmatrix}
\begin{bmatrix}
w_{i,1}^{(m)} \\
w_{i,2}^{(m)} \\
\cdots \\
w_{i,N}^{(m)}
\end{bmatrix}
\end{bmatrix}
\]

(3)

Note that matrix \([A]\) is symmetric real matrix it is also a positive definite matrix which is non singular. Thus the weights vector \([w]\) in above equation can be uniquely determined.

A series of RBFs have been proposed and the most commonly used RBFs are multiquadrics (MQs): 

\[
\varphi(r) = \sqrt{r^2 + c^2}, \ c > 0,
\]

Thin-Plate Splines (TPS): 

\[
\varphi(r) = r^2 \log(r),
\]

Gaussians: 

\[
\varphi(r) = e^{-\alpha r^2}, \ \alpha > 0,
\]

Inverse multiquadrics (IMQs): 

\[
\varphi(r) = 1/\sqrt{r^2 + c^2}, \ c > 0.
\]

Generally MQs performs better than other RBFs on the interpolation of 2D scattered data. The exponential convergence of MQs makes them superior to other RBFs. We will use the MQs as basis functions to calculate the weight vector \([w]\) in our present work.

**B. The accuracy of MQ-DQ method**

The accuracy of MQ-DQ method for evaluating derivatives strongly depends on the shape parameter \(c\). The choice of shape parameter has been a topic with lots of discussions in researchers. To find the optimum shape parameter 

Franke suggested a formula as 

\[
c = 1.25D / \sqrt{N_j}
\]

where \(N_j\) is the number of supporting nodes and \(D\) is the radius of the smallest circle that contains supporting nodes. Hardy suggested another way to evaluate the shape parameter, 

\[
c = 0.815d,
\]

where \(d = \frac{1}{N_j} \sum_{i=1}^{N_j} d_i\) is the average distance between reference node and its neighboring nodes.

Kansa proposed a formula, 

\[
c_j^2 = c_N^2 \left( \frac{c_M^2}{c_N^2} \right)^{j-1}
\]

where \(c_M\) and \(c_N\) are input parameters. With carefully choosing of the input parameters the accuracy could increase up to five orders of magnitude for many monotonic functions.

Another factor that may influence the accuracy of MQ-DQ method is the number of supporting nodes, \(N_j\). In above works the number of neighbors for each node are assumed to be sufficient to find out the optimized \(c\) in \(c - N_j\) space. However performance associated with \(N_j\) should be considered in practical computing. Typically the accuracy of MQ-DQ method increases along with the increasing of number of supporting nodes, for more basis functions are introduced into interpolation. Using only a few supporting nodes will decrease the accuracy dramatically. On the other hand, using too many supporting points causes great amount of computation. An optimized \(N_j\) must be chose.

A general theoretical analysis of how to determine the optimum combination of \(c\) and \(N_j\) could be difficult. Therefore a numerical searching is performed on computing the first order derivatives of function 

\[
f(x, y) = x^2 + y^2.
\]

To omit the influence of the measurement of distance and make the problem dimensionless the reference node is placed at the center of a circle with a radius of 1 (Fig. 1 left, red dot). Supporting nodes are scattered around the reference node (Fig. 1 left, green). Using above RBF-DQ technique the max error of the first order derivatives, 

\[
\left\| \nabla f \right\|_{\text{num}} - \left\| \nabla f \right\|_{\text{ana}}
\]

was evaluated through the difference of numerical value and analytical value (Fig. 1 right).
Although the minimum error shows up at \( c=30 \) and \( N_I=18 \) it could be computational expense when \( \sim 20 \) neighbors are assigned to each reference point. Since the error maintains a magnitude of \( 10^{-5} \) in the region where \( c > 110 \) and \( N_I \geq 7 \) it is acceptable to compromise a little accuracy while save a great amount of computational costs. We will keep \( c=130 \) and \( N_I=8 \) in our calculations from now on. It is worth noting that our tests give a dimensionless value of \( c \). Measurement of the distance from reference node to its neighbors could be various in different meshes. And the absolute value of \( c \) should be depend on the measurement.

2. RBF-DQ Based Euler Solver

A. Euler Equations and Discretization Procedure

The governing equations of inviscid fluid dynamics are a set of hyperbolic nonlinear PDEs, which are know as Euler equations. In vector and conservation form in two-dimensional physical space, the Euler equations are written as

\[
\frac{\partial}{\partial t} Q + \frac{\partial}{\partial x} F_1(Q) + \frac{\partial}{\partial y} F_2(Q) = 0
\]

(4)

Where

\[
Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad F_1 = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{bmatrix}, \quad F_2 = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{bmatrix}
\]

where \( e = \rho (\varepsilon + (u^2 + v^2)/2) \) is the density of total energy and \( \varepsilon \) is the density of internal energy. There are five unknowns and four equations. Closing the system requires an additional equation. The most commonly used equation is the state equation of the ideal gas, \( p = (\gamma - 1)[e - \frac{1}{2} \rho (u^2 + v^2)] \).

While an exact analytical solution of those equations is often difficult to get, especially when the boundaries of the problem are irregular, numerical approximated solution will be the only choice. With the local RBF-DQ method we presented in the previous sections the divergence of flux in Equation (4) could be evaluated. To preserve upwind, similar with that in finite volume methods, the partial differentials of flux at reference point are evaluated through linear combination of flux at the mid-points between the reference node and its supporting nodes (Figure 2). In this way we are able to calculate the divergence of flux with upwind scheme (detailed in the next section).
After applying discretization based on RBF-DQ method the Euler equations become following form

$$\frac{\partial \mathbf{Q}}{\partial t} = - \sum_{k=0}^{N_i} \left( w_{i,k}^{(1x)} \mathbf{F}_1(\mathbf{Q}_{i,k}) + w_{i,k}^{(1y)} \mathbf{F}_2(\mathbf{Q}_{i,k}) \right)$$

(5)

Where $\mathbf{Q}_{i,k}$ are conservative variables at the mid points between reference node $i$ and its $k^{th}$ supporting node. Note that the right terms of equation (5) for each midpoint could be written as

$$w_{i,k}^{(1x)} \mathbf{F}_1(\mathbf{Q}_{i,k}) + w_{i,k}^{(1y)} \mathbf{F}_2(\mathbf{Q}_{i,k}) =$$

$$\sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2} \left( \frac{w_{i,k}^{(1x)}}{\sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2}} \mathbf{F}_1(\mathbf{Q}_{i,k}) + \frac{w_{i,k}^{(1y)}}{\sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2}} \mathbf{F}_2(\mathbf{Q}_{i,k}) \right)$$

(6)

If we define a unit vector $\mathbf{\bar{w}} = (\alpha_{i,k}, \beta_{i,k})^T$ where

$$\alpha_{i,k} = \frac{w_{i,k}^{(1x)}}{\sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2}} \text{ and } \beta_{i,k} = \frac{w_{i,k}^{(1y)}}{\sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2}}$$

(7)

then equation (6) actually calculates the projection of mid-point flux on the unit vector $\mathbf{\bar{w}}$, which is

$$\mathbf{G}_{i,k} = \alpha_{i,k} \mathbf{F}_1(\mathbf{Q}_{i,k}) + \beta_{i,k} \mathbf{F}_2(\mathbf{Q}_{i,k})$$

(8)

This provides a possibility to compute the projected flux through upwind scheme. As we define a new “weight” variable $W_{i,k} = \sqrt{(w_{i,k}^{(1x)})^2 + (w_{i,k}^{(1y)})^2}$, equation (5) becomes

$$\frac{\partial \mathbf{Q}}{\partial t} = - \sum_{k=0}^{N_i} W_{i,k} \mathbf{G}_{i,k}$$

(9)

Then the conservative flow variables could be updated by time integration methods such as forward Euler or Runge-Kutta methods.

**B. Upwind scheme for the Euler Solver with Limiters**

To keep our scheme upwind we will use the popular Godunov’s type scheme to obtain the projected flux at the mid points by solving local one-dimensional Riemman problem. However exactly solving Euler equations in Riemann problem needs iterations and is very time consuming. In the present work Rusanov solver is used for approximating the solution of Riemman problem. The Rusanov solver assumes that all the waves associated with the hyperbolic
equations travel with the maximum wave speed. Let \( \mathbf{Q}_L \) denote the conservative variables at the reference node and \( \mathbf{Q}_R \) denote the conservative variables at the supporting node. Similarly, let \( \mathbf{Q}'_L \) denote the interpolated conservative variables at the left side of mid point, and \( \mathbf{Q}'_R \) denote the interpolated conservative variables at the right side of mid point. Then the common flux at the mid point could be calculated by Rusanov’s scheme

\[
G(\mathbf{Q}_L, \mathbf{Q}_R) = \frac{1}{2}[G(\mathbf{Q}'_L) + G(\mathbf{Q}'_R)] - \frac{1}{2}|\hat{\mathbf{A}}|(\mathbf{Q}_R - \mathbf{Q}_L)
\]

where

\[
\hat{\mathbf{A}} = \begin{bmatrix}
|V_n| + c & 0 & 0 & 0 \\
0 & |V_n| + c & 0 & 0 \\
0 & 0 & |V_n| + c & 0 \\
0 & 0 & 0 & |V_n| + c
\end{bmatrix}
\]

\( |V_n| \) is absolute value of velocity along the direction \((\alpha_{i,k}, \beta_{i,k})^T\) at mid-point and \(c\) is the local speed of sound.

The interpolated \( \mathbf{Q}'_L \) and \( \mathbf{Q}'_R \) depend on the order of interpolation. For the first order interpolation it simply let \( \mathbf{Q}'_L = \mathbf{Q}_L \) and \( \mathbf{Q}'_R = \mathbf{Q}_R \). The second order interpolated conservative variables could be calculated as

\[
\mathbf{Q}'_L = \mathbf{Q}_L + \nabla \mathbf{Q}_L \cdot (\mathbf{x}_{mid} - \mathbf{x}_L) \\
\mathbf{Q}'_R = \mathbf{Q}_R + \nabla \mathbf{Q}_R \cdot (\mathbf{x}_{mid} - \mathbf{x}_R)
\]

Higher order interpolations could also be approximated by Taylor series.

Although the accuracy of the solver increases by applying those high order interpolations the cost is convergent problems. Non-physical oscillations near the discontinuities and uniform solution region, which is general characteristic of high order scheme, will prevent us from getting a convergent solution. The oscillations are often caused by the appearance of new local extrema of conservative variables which are interpolated from the reference node and its supporting nodes. Another source of oscillations is the frequently changing of interpolation values. This often happens in uniform solution regions.

To suppress oscillations limiters are commonly used for the modification of interpolations. The second order of interpolation with limiter then becomes

\[
\mathbf{Q}_{ik} = \mathbf{Q}_i + \Phi_i \nabla \mathbf{Q}_i \cdot (\mathbf{x}_{mid} - \mathbf{x}_i)
\]

Where \( \Phi_i \) is the limiter at reference node \( i \). Barth and Jespersen\(^22\) proposed the first limiter for finite volume method for unstructured grids. The limiter could be calculated through the following steps.

1) Find the largest negative and positive difference between the solution in the neighbor nodes and reference node,

\[
\partial \mathbf{Q}_i^{\text{min}} = \min\{\mathbf{Q}_k - \mathbf{Q}_i\}, \quad \partial \mathbf{Q}_i^{\text{max}} = \max\{\mathbf{Q}_k - \mathbf{Q}_i\}.
\]

2) Compute a maximum allowable value \( \Phi_{ik} \) for each supporting node \( k \).

\[
y = \begin{cases}
\frac{\partial \mathbf{Q}_i^{\text{max}}}{\mathbf{Q}_{ik} - \mathbf{Q}_i}, & \text{if } \mathbf{Q}_{ik} - \mathbf{Q}_i > 0 \\
\frac{\partial \mathbf{Q}_i^{\text{min}}}{\mathbf{Q}_{ik} - \mathbf{Q}_i}, & \text{if } \mathbf{Q}_{ik} - \mathbf{Q}_i < 0
\end{cases}
\]

so that \( y \geq 0 \)

American Institute of Aeronautics and Astronautics
then \[ \Phi_{ik} = \begin{cases} \min(1, y), & \text{if } Q_{ik} - Q_i \neq 0 \\ 1 & \text{if } Q_{ik} - Q_i = 0 \end{cases} \] (13)

3) Get the limiter as the minimum \( \Phi_{ik} , \Phi_i = \min(\Phi_{ik}) \).

Although this limiter prevents introducing new extrema to the solution near the reference node it is hard to reduce the residue more than two or three orders of magnitude. For the limiter introduces non-differentiability in the computation of the reconstructed function. Venkatakrishnan\(^{23}\) suggested a differentiable limiter, where the function min\((1, y)\) in step 2 were replaced with a differentiable function \( \Phi(y) = -\frac{y^2 + 2y}{y^2 + y + 2} \). This limiter works well with uniform grids but loses accuracy for unstructured grids. In present work we use a new limiter that is suitable for meshless methods. It was originally developed by Gooch\(^{24}\) for finite volume methods. The \( \min(1, y) \) was replaced with
differentiable function

\[
\begin{cases}
-\frac{4}{27} y^3 + y & y < y_i \\
1 & y \geq y_i
\end{cases}
\]

where \( y_i = 1.5 \). Along with another factor the limiter we use is

\[
\tilde{\Phi}_i = \sigma_i + (1 - \sigma_i)\Phi_i , \text{ where } \sigma_i = \begin{cases} 1 & \delta Q^2 \leq (K\chi)^3 \\
\frac{s(\delta Q^2 - (K\chi)^3)}{(K\chi)^3} & (K\chi)^3 < \delta Q^2 < 2(K\chi)^3 \\
0 & \delta Q^2 \geq 2(K\chi)^3
\end{cases}
\]

Here \( s(y) = 2y^3 - 3y^2 + 1 \), \( \delta Q = Q_{\max} - Q_{\min} \), \( \chi \) is the minimum distance from reference node to its supporting nodes. \( K \) is a tunable parameter and we fix it to 1 in our calculations. The purpose of adding this factor \( \sigma_i \) is to suppress the rapidly changes of limiter value in uniform region and eliminate the effect of the limiter in region of uniform flow or near smooth extrema.

C. Boundary Conditions and Treatment

For fixed walls and symmetric boundaries the solutions of inviscid flow problems require the velocity of flows being parallel to the boundary. In other words the normal velocity \( u_n = 0 \). To implement this condition numerically ghost nodes are added for walls and symmetric boundaries. The imaged nodes have identical mass density and pressure with the original nodes, but their velocity is mirror reflected from that of original nodes with respect to the physical boundaries.

The treatment of inlet and outlet boundaries is based on the theory of characteristics. By solving a Riemman problem, the inlet and outlet boundary conditions could be setup. The Riemman invariants are

\[
w_1 = \frac{p}{\rho^\gamma} , \quad w_2 = u_t , \quad w_3 = u_n + \frac{2c}{\gamma - 1} , \quad w_4 = u_n - \frac{2c}{\gamma - 1}
\]

Where \( u_n \) and \( u_t \) are the normal and tangential velocities at the inlet or outlet boundary respectively. In case of supersonic flow the four variables are fixed to freestream conditions at the inlet and extrapolated from interior at the outlet. For subsonic flow \( w_1 , w_2 , w_3 \) have positive signs but \( w_4 \) has negative sign. They should be treated separately. \( w_1 , w_2 \) and \( w_3 \) are fixed with freestream value at inlet and extrapolated from interior at outlet while \( w_4 \) is extrapolated from interior at inlet and fixed with freestream value at outlet.
III, Point Set Generation

In the present study, we plan to develop a meshless solver with only an input geometry composed of one or multiple closed bodies. Nodes are then automatically generated using the following approach with the help of a GUI.

A. Body Point Set Generation

There are two steps in generating the point sets around the closed bodies, which are defined either analytically or discretely. If the body is defined by a function, the first step is to distribute mesh points along the body. The distance between neighboring points is controlled by a mesh size \( h \). The mesh size is in turn determined by both the curvature \( C \) at the mesh point and the range of mesh size \([h_{\text{min}}, h_{\text{max}}]\) given by the user in the input file.

\[
h = \begin{cases} 
1/5C, & \text{if } 1/5C \in [h_{\text{min}}, h_{\text{max}}] \\
h_{\text{min}}, & \text{if } 1/5C < h_{\text{min}} \\
h_{\text{max}}, & \text{if } 1/5C > h_{\text{max}} 
\end{cases}
\]

Whenever a mesh point on the wall boundary is generated, a unit normal vector is computed for that point (Figure 3). At the smooth part of the curve, the normal vector is assumed unique. At a sharp geometrical feature point, such as the trailing edge of an airfoil, multiple normal vectors are associated with the point. The angle between neighboring multiple vectors should be no larger than \( \theta_m \), where \( \theta_m \) is typically 30°.

![Figure 3. Diagram of body point set generation.](image)

The second step in the body point set generation is to produce points in the vicinity of the body by marching in the normal direction. In this step surrounding points will be shoot out from each wall boundary point away from the body. The wall boundary points which have multiple normal vectors will also shoot out multiple points. The points are generated layer by layer around the body. The number of layers and thickness between the layers are controlled by the parameters in the input file. The generated point sets of an airfoil and a rigid circle are shown in Figure 4 as an example.
B. Background Point Set Generation
Since we deal with external flow problems only for the time being, we generate the background mesh by recursively refining a single Cartesian cell covering the entire computational domain. We use the outer layers of the body meshes to match the mesh resolution between the Cartesian and body meshes. When the size of the local Cartesian cell is comparable to the outer layer mesh size, the Cartesian cell refinement process stops. With a quad-tree data structure, each node of the tree represents a Cartesian cell. In order to ensure that the length scales do not change abruptly, the quadtree is balanced in that the size of neighboring elements does not differ by a factor over 2. Then the cell centers of the Cartesian cells form the background point sets. Any points of the background which are overlapped with the body point sets are set to inactive in the flow simulations.
IV Results and Discussion

A. Supersonic flow in a convergent nozzle with a ramp on the floor

The channel has 15° compression ramp followed by 15° expansion corner along both lower and upper walls (Fig. 3). The initial condition was uniformly identical to far field condition with a Mach number of 2.0. Since the nozzle is symmetric we only choose the lower half as the computational domain. Symmetry boundary conditions are applied on the symmetric line. The use of this symmetry nature brings down the computational costs.

To test the effectiveness of the new limiter and take mesh refinement into account we generated two nodal distributions with 97x33 and 193x65 nodes. The mach number contours are shown in Fig. 4. Both of them captured incident and reflected shocks and the shocks get sharper as the nodal density increases.

The history of residues are shown in Fig 5. The residues of both systems reach to machine zero after some time step (6000 in Fig 5 (a) and 13000 in Fig 5 (b) ). Although the number of nodes are four time more (number of nodes doubled on each dimension) mesh refinement does not impact the convergence of solution.
For comparison we also tested meshless Euler solver with Barth and Jespersen’s limiter, which we discussed previously (Eq. 13). The mach number distribution around the incident shock shows unsmooth (Fig 6(a)). The residues decreased only by less than a magnitude of 2 (Fig. 6(b)). The system oscillated after 3500 time steps so that a convergent solution was not obtained.

B. Subsonic Flow Over an Airfoil and a Circle

Based on above point set we ran a test case using our meshless solver to solve a subsonic compressible flow problem. The Mach number of the freestream is $M_\infty = 0.4$. The angle of attack is set to zero. The initial condition for all points is set as the free stream condition. The steady state solutions from the first order scheme are shown in Figure 8. The convergence history in terms of the L2 density residual is shown in Figure 9. All the solution variables show correct qualitative behavior. Note that machine zero convergence was obtained within 10000 explicit time steps.
Figure 8. Steady state solution for a testing case with Mach number 0.4 of the free stream and 0 angle of attack on the GUI.

Figure 9. The residual history of the testing case ran from the GUI.
V Conclusion

We have presented a RBF-DQ based meshless Euler solver with a new convergent limiter. The radial basis functions allow us to build solvers for compressible fluid flow equations without a mesh. Therefore the present method can work on randomly distributed nodes with no prerequisite connectivity. The solver was tested for a supersonic problem with shock waves. Machine zero convergence was achieved on both a coarse and fine meshes. According to the distribution of Mach number, expected characteristics of the solution, i.e. incident and reflected shocks are captured with high resolution. A GUI based mesh generation and flow visualization package is being developed, which allows a user to input only a geometry to obtain a CFD solution. A subsonic flow over an airfoil and circle was computed with this GUI, and qualitatively correct solution is obtained. Currently we are performing further verification studies, and extending the solver to handle moving boundary problems.

Acknowledgements

The authors gratefully acknowledge financial support from NASA under grant NNX12AK04A.

References


