International Journal of Applied Science and Engineering

Open source extensions applied to meshing problems for KIVA 4

Carlos Alberto Barrera Soto, David Sebastian Perez Gordillo, Carlos Felipe Forigua Rodriguez, Juan Miguel Mantilla Gonzalez*

Universidad Nacional de Colombia, Mechanical and Mechatronics Engineering, Bogota Colombia

ABSTRACT

KIVA is a very successful software that was released around 1985. One of its advantages is the capacity to include computer routines made by the users. That is why it is mostly used for research and development and as a tool for simulation driven design and optimization of internal combustion engines. As with many CFD software, mesh generation with KIVA is time consuming when its own meshing program is used. This can be exacerbated in the case of complex geometries with a special mesh quality, or when unstructured meshes are needed. Commercially, there are several meshing tools that have features and are available for a price. But this study explores mesh generation for KIVA in four cases of difficult geometries using open source tools. This allows to control all the meshing steps and parameters like mesh refinement, anisotropy and directionality. The first case is a Venturi tube with diesel spray injected in the middle of the throat. The second one an unstructured mesh for an eccentric diesel piston bowl. The last two cases are hybrid meshing methods for an engine with two valves and a pre-chamber, and an engine with two valves and a non-axisymmetric bowl. All the cases were validated in KIVA. This study extends the applicability of KIVA in terms of possibilities and ease of mesh generation. Users can use their preferred CAD/mesh software and convert it to KIVA if the format is compatible with OpenFOAM. Accordingly, the presented methodology would reduce the time devoted to meshing.

Keywords: KIVA-4, Engine meshing, Complex geometry, Open source meshing tools, Open Foam, Grid generation, Engine design and optimization.

Received: June 5, 2020

Accepted: December 31, 2020

Corresponding Author:
Juan Miguel Mantilla Gonzalez
jmmantillag@unal.edu.co

Publisher:
Chaoyang University of Technology
ISSN: 1727-2394 (Print)
ISSN: 1727-7841 (Online)

1. INTRODUCTION

Currently, there is a great concern to reduce pollutant emissions and improve the performance of internal combustion engines (ICE) (Yu et al., 2001). The constant advance in computer technologies and the progress of the understanding of physical phenomena has led to the establishment of 3D simulation as a powerful tool in the study, design and development of internal combustion engines (Yi, 2008). For this reason, optimization and design processes had to modify the shape of these devices, which in some cases increase its geometric complexity. (Yu et al., 2001; Park, 2010; Park and Lee, 2010; Wickman et al., 2001). However, the more complex the geometry of the engine, the more difficult it is to generate the mesh of the computational domain. This promotes the need for improved meshing tools which allow 3D simulation of a more diverse set of engines (Yi, 2008). In computational studies, accuracy and user interaction time are considered important aspects of a simulation (Yi, 2008). The precision of the results is conditioned by the mesh quality, while the generation of the discretized domain consumes most of the user's time when compared to the simulation time (Yi, 2008).

KIVA is a Fortran-based software founded on the finite volume method that simulates the thermal, chemical and fluid processes that take place inside an internal combustion engine (Amsden, 1993; Torres, 2007; Torres, 2006). The software is open source after paying a fee, it has more than 30 years on the market and began with version 2. Right now, versions 3V and 4 are still maintained by Los Alamos National Laboratory. In 2011, a parallel version of KIVA 4 using message passing interface (mpi) was launched (Torres et al., 2009). The main drawback of the software is the programming methods. Therefore, since 2016 a new strategy has been deployed with the release of KIVA-hpFE, where the entire numerical method was updated to finite elements. In 2018, its name was changed to FEARCE (Fast, Easy, Accurate and Robust Continuum Engineering) (Carrington et al., 2018).

As it is common in CFD practice, KIVA has presented significant problems for obtaining meshes (Nishad, 2018; Sharma et al., 2010). First, the resulting low-quality meshes significantly degrade the accuracy of the CFD simulations and increase computation time (Shimada, 2006; Abani and Reitz, 2010). On the other hand, the creation of meshes for complex geometries has historically been a task with a high degree of difficulty using the existing mesh methods for this software (Xue, 2008). This has caused other programs such as Converge (Converge CFD Software, 2018), STAR-CD (Fischer and Schmidt, 2016) and AVL FIRE (AVL FIRE Team, 2018) to take advantage of this aspect with automatic meshing, since with them it is possible to mesh complex geometries in engines such as chambers, non-axisymmetric bowls and geometries different from an engine (Bestel et al, 2020; Sharma et al, 2010; Jafarmadar and Zehni, 2014). Nevertheless, one issue with automatic meshing is the control of mesh quality via anisotropy and directionality (Chawner and Taylor, 2019; Chawner et al., 2016).

The conventional KIVA-3V mesh generator, K3PREP, defines a variety of useful block shapes for patching, enabling the meshing of combustion chambers, conduits and valve ports, with moderately complex geometries (Amsden, 1993; Shi, 2016; Amsden, 1997). The results of previous studies show that the computational mesh generated by K3PREP accurately describes the complex geometries of real engines (Shi et al., 2016). However, the structured block method, used mainly for engines with valves, does not allow the creation of meshes with an uniform cell size (Amsden, 1997; Imamori et al., 2009). As a result, there has been an increase in computational costs caused by the high number of cells needed to ensure an acceptable accuracy and precision in the simulations to perform (Imamori et al., 2009). Another problem of K3PREP was that when simulating one section of the engine, only the use of polar meshes was possible (Amsden, 1993; Amsden, 1997; Imamori et al., 2009). The size of the cell in a polar mesh varies with the location in the radial direction, since it decreases as it is closer to the center of the engine. For this reason, these cells produce high computing times and can cause inaccuracies in the prediction (Torres, 2006; Imamori et al., 2009; Xue and Song, 2009). To avoid this inconvenience, some improvements were implemented in the code in KIVA-3V-release2, for enhancing the flexibility of the mesh, always allowing the use of Cartesian meshes and without increasing the complexity of use (Imamori et al., 2009; Amsden, 1999). However, there are still real applications that exceed the capabilities of the aforementioned meshing software.

KIVA-4, launched in 2006, has been generalized to support unstructured meshes. It encourages three types of elements: hexagonal, pyramidal and tetrahedral (Torres, 2007). This new ability to work with unstructured meshes facilitates the process of construction of the mesh in complex geometries and offers greater flexibility (Torres, 2007; Torres, 2006). A combination of meshes generated in different ways is used in order to take advantage of each of them and avoid the problems caused by them (Torres, 2007). For example, it is possible to use a structured mesh for the part of the squish that would allow a proper snapping, and an unstructured mesh to shape complex geometries, static and moving meshes. In any case, the possibilities are enhanced, but with K3PREP it is still not possible to generate unstructured meshes (Amsden, 1997; Amsden, 1999).

Due to the increase in the use of KIVA, some closed source code commercial mesh generators, capable of creating meshes for complex geometries, have been used more frequently for the meshing of engines in the KIVA-4 format (Torres, 2007). ICEM and TrueGrid are two of the software programs that currently allow obtaining meshes in this format (Torres, 2007; WERC, 2018). ICEM is a software developed by ANSYS, Inc. and TrueGrid by XYZ Scientific Applications, Inc. Among their possibilities, these programs have the generation and optimization of hexahedral, tetrahedral, structured and unstructured meshes (Yuan et al, 2020; Rainsberger, 2018). As a guide for these applications, Wisconsin Engine Research Consultants published a manual for the generation of meshes focused on simulations of internal combustion engines. The Manual is based on the use of ICEM and KIVA (WERC, 2018). In addition, KIVA-4 has an inhouse converter (Cubit2KIVA) for use with Cubit, a mesh generator developed by Sandia National Lab (Torres, 2007; Sandia National Lab, 2015). With these external meshing tools, mesh generation time is relatively low (WERC, 2018; Rainsberger, 2018), which makes them reliable alternatives, or in some cases complements, for K3PREP.

Despite their many advantages and capabilities, these proprietary applications have some setbacks. As the code is unknown, it lacks the necessary flexibility to be enhanced by the users according to their needs. Likewise, the user cannot incorporate the meshing algorithm (Chawner et al., 2016) into KIVA's source to enable features such as adaptive mesh refinement (Chawner et al., 2016), new mesh movement algorithms and others. These characteristics have been considered crucial in high performance computing environments that are required for complex geometries and Large Eddy Simulations (LES) (Chawner et al., 2016). This gives the user more freedom to generate structured and unstructured meshes, for simple or complex geometries to be simulated in KIVA-4.

Based on the above and considering the actual use of several versions of KIVA for research and development, this document presents new tools applied to specific cases for the construction of structured or unstructured meshes, applicable in KIVA-4. This aims to augment the utility of the meshing software, extending its applicability to cases in which the computational domain has a geometry with a high degree of complexity.

As the employed tools are free and open source, long-term robustness of the methodology is ensured and as the file formats are known in each step, any other tool can be used in substitution. This is a first step that enables the following possibilities for immediate future projects: benchmark with the same mesh against other CFD packages; automatic mesh generation (Chawner et al., 2016) and meshing with Computer Aided Design (CAD) kernels (Chawner et al., 2016); valve generation by mesh tools other than K3PREP; and benchmarking KIVA-4 against other software with the exact same mesh. As there is no open source meshing alternative for the KIVA software, this paper uses on an open source free software, OpenFOAM, seeking to take advantage of all the mesh generators compatible with it (OpenFOAM Foundation, 2017).

For this purpose, OpenFOAM mesh needs to be translated to the KIVA format. Therefore, a subroutine was developed in OpenFOAM to convert its mesh to the KIVA format. This software is called foamToKIVA4Mesh. The OpenFOAM mesh must be created with certain requirements to work with the converter. The edge faces and volume groups must be declared according to a convention to build the kiva4grid file (Torres, 2007). Likewise, it must be guaranteed that all the elements belong to the type of elements allowed by KIVA, in the same way as the boundary conditions.

Taking all of this into account, the layout of this paper is as follows. Initially, the OpenFOAM and KIVA-4 mesh formats are described, as well the developed conversion subroutine, foamToKIVA4Mesh. Then, different mesh cases are presented, based on their use of K3PREP, together with free mesh generators such as blockMesh (OpenFOAM Foundation, 2017) and Gmsh (Geuzaine and Remacle, 2017). Study case 4 shows the use of an in-house code for the generation of a mesh combined with K3PREP, specifically for the case of engines with valves and non-axisymmetric bowls. Finally, the conclusions obtained from the development of the work are outlined.

2. MESH DESCRIPTION AND CONVERSION

OpenFOAM is a widely used program, there are several mesh converters that allow to translate meshes from other formats to the OpenFOAM format. The converter developed in this research permit to convert an OpenFOAM mesh to the KIVA-4 format. This section briefly explains the mesh format used in KIVA-4 and the one used in OpenFOAM. The differences between them are presented. Finally, a brief explanation of the algorithm for the new converter is given.

2.1 KIVA-4 Unstructured Mesh Format

The input mesh is a file named "*kiva4grid*" that has the following structure:

- 1 line : It is reserved to the name of the problem.
- 1 line : Number of cells (ncells), Number of vertices (nverts).
- A block of nverts lines. With 3 columns of X, Y, Z coordinates for each vertex.
- A block of ncells lines: With 6 columns corresponding to the nodes of each cell. Its order is predetermined by a convention on a hexahedron (Torres, 2007).
- A block of ncells lines: This is a list of 7 ordered columns of integers, whose order corresponds to a code that describes the types of cells and the types of the faces by an ordered convention (left, front, bottom, right, derriere, top).
- 1 line: binary value that informs whether kiva4grid connectivity information is present (structured/unstructured).
- Other lines related to the structured mesh, periodic vertices (Amsden, 1993; Torres, 2007).

2.2 OpenFOAM Mesh Format: PolyMesh

OpenFOAM mesh format is described by 4 files that are dependent on each other and have the following contents:

- File of points: List of points of the mesh indexed by vertex numbers.
- File of faces: List of indexed faces defined by the vertex numbers in the previous file. Internal faces are printed first, as they connect 2 cells, and boundary faces are printed next, as they describe the boundary condition faces (patches).
- Files of neighbour and owner cells: These are two lists of indexed cells defined by the face numbers in the previous file. The faces are assigned to owner and neighbour cells. This is useful as the boundary condition faces do not have a neighbour, therefore their neighbour index is set to −1. This way, the connectivity of the cells is defined by the duo neighbour-owner of each face, thus the boundaries can be clearly distinguished and inconsistencies easily detected.
- File of patches (boundary conditions): List of faces that is called patches and described by the index of faces already defined. The patches group the faces according to boundary conditions. They are designated by a naming word and a couple of numbers that represent the start and end of a slice in the continuous index of faces.

In addition to these files, there is an optional file named cellZones, which is a list of indexed cells distinguished with a name. This is an option in OpenFOAM to group cells (Weller et al, 1998).

2.3 Mesh Differences

While OpenFOAM uses arbitrary cell shapes (Weller et al, 1998), KIVA is limited to a few cell shapes. Fortunately, OpenFOAM can represent its mesh in memory in a more conventional form via the cellShape class. This does not necessarily means a direct conversion, as the conventional vertex order and faces depiction of both hexahedral cell shapes are different for each program.

KIVA uses a numeric code for each of its faces and stores it in 6 ordered columns (top, bottom, left, right, front and derriere), and OpenFOAM uses the neighbor-owner representation. These KIVA-4 columns also store the boundary conditions of the faces and use a numeric code for each face. In contrast, OpenFOAM uses a word in a group of faces (patch).

A similar case occurs for the groups of cells. KIVA distinguishes every cell via a numeric code for volumetric contents such as valves, squish, among others. This is not a requirement for OpenFOAM. Therefore, the definition of such cell parameters is included in the OpenFOAM mesh via the optional cellZones file. This translation from a named list in OpenFOAM to a numeric array of ordered faces is another task of implementation.

All the elements and cells are described explicitly by kiva4mesh, in contrast to polyMesh, where less data is used for input to generate the mesh in memory. As this is needed, modifications to OpenFOAM were mandatory. The original library foam2STARCD was modified to access the binary representation of the OpenFOAM mesh and write a conforming file in kiva4grid format.

2.4 Mesh Conversion Algorithm: FoamToKIVA4Mesh

The mesh is read by OpenFOAM using a standard method. When stored to memory, it is saved into the more conventional representation of hexahedrons depicted in simulations. As OpenFOAM uses a generalized format, the mesh is asked to be presented in the more conventional form that KIVA also supports (Greenshields, 2018). This form it is then used to read the point and faces, so that can be ordered into Kiva format. OpenFOAM is then asked to list each discretized volume faces, position of face vectors and the opposite faces of each hexahedron. As the faces may present high distortion, it is of utmost importance to have a precise and consistent metric that tells which face is at each side, not only for each element, but for any other element in the mesh. This is critical, as KIVA-4 mesh uses this metric to define its mesh in terms of left-right, top-bottom and front-derriere (Torres, 2007).

The faces of each element are ranked in the 3 local axes, which are the furthest ones in the *X*, *Y* and *Z* axes. Some highly distorted faces can have the same rank in two or more axes (Fig. 1). In Fig. 1, the y_2 coordinate of face n_2 is higher than the y_1 location of face n_1, thus x_1 and x_2 coordinates are of equal value for both faces n_1 and n_2. When ranked in the *x* axis, n_1 and n_2 faces would be in a tie. Consequently, n_2 can be accidentally selected to be a top and right face at the same time. However, when ranked in the *y* axis, the y_2 is the highest rank for that axis and therefore face n_2 is banned to be in any other face. This prevents face n_2 to be at the same time a high rank face at the top and the right of the depicted control volume. In consequence, face n_1 can be selected as the right face as it is the highest rank in the *x* axis, now that n_2 was already selected as a top face because it is dominant in the *y* axis. Once this is completed, the 3 further faces of each axis are selected and, the other 3 faces of the hexahedron are their opposites. In this way, the faces of each control volume are related from OpenFOAM format to KIVA terminology of left-right, top-bottom and front-derriere.

As OpenFOAM uses words to define it's boundary conditions and the content of its cells as cellZones, these words are recovered to assign the numerical value to the file of the converted mesh. Each face is written with its boundary value and the volume type. For example, OpenFOAM assigns a value of -1 to fluid faces when a face is not included in a patch, while KIVA-4 assigns a value of

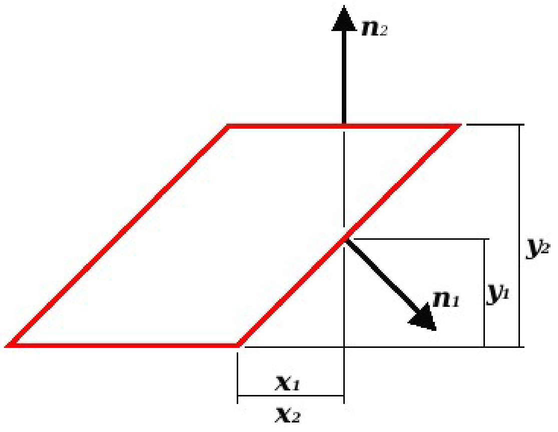


Fig. 1. Side view of a hypothetical distorted control volume shaped as an hexahedra

40 to those same faces. The cellshape of OpenFOAM is traced into the one of KIVA and then it is written into the translated mesh with its correspondent face value. As this is a non-structured mesh, there is no more data to write and the file is closed. As a procedure, foamToKIVA4Mesh can be summarized as follows:

1. Use standard methods in OpenFOAM to read the mesh.
2. Write the mesh file header.
3. Assign the cell shape hexahedron to the OpenFOAM mesh.
4. Translate the OpenFOAM hexahedron point order convention into KIVA hexahedron point order convention.
5. Write the translated point list for the hexahedron in the KIVA file.
6. List the normal vectors that correspond to each of the faces of the hexahedron.

7. Generate a list of the faces of each cell and their corresponding opposite face in OpenFOAM.
8. Rank all these normal vectors in descending order at each axis.
9. Assign non-repeating faces for each maximum face value for each of the 3 axes. i.e. The list of faces [1,2,3] for x, y and z axis is valid, but the list of faces [1,2,2] is not (See Fig. 1).
10. Take the most dominant face in each axis face and assign the corresponding KIVA convention, i.e. the highest X face [1] in OpenFOAM = Front; the highest Y face [2] in OpenFOAM = Right; and, highest Z face [3] in OpenFOAM = Top.
11. Assign the KIVA convention to the opposite of this faces, i.e. the opposite of the highest Z value face (Top) is now the bottom.
12. Write these faces in the order established in the KIVA format.
13. Read OpenFOAM listed names assigned to each patch and assign them the numeric value corresponding to KIVA. For example, a face corresponding to a patch named "moving" gets number 10 assigned. Faces without a boundary condition in OpenFOAM get the "40" number, which corresponds to a fluid face in KIVA.
14. Close the mesh file.

3. STUDY CASES OVERVIEW

The following sections show different cases that were used in the construction of meshes for KIVA-4. These generic cases for spatial discretization increase the versatility and applicability of KIVA-4. K3PREP is considered the conventional and default meshing tool of KIVA-4. However, the use of K3PREP remains indispensable in some applications, especially those involving the valve system and the intake and exhaust processes in the engine. This occurs mainly due to the fact that this software is characterized by a superior performance in the handling of mobile meshes in areas close to the valves, compared to the other meshing software used for KIVA-4 (Amsden, 1997; Amsden, 1999; Torres, 2007). This is a factor to consider, since the ability to generate moving meshes is a critical and essential component in ICE CFD simulations. For these reasons, the exposed cases can be divided into two groups: a standalone meshing program different from K3PREP and K3PREP used in combination with auxiliary software. In all of these cases, foamToKIVA4Mesh has been used to translate all the mesh or one of its components to the KIVA format.

The first two cases include those methodologies in which the complete mesh is created using only a software program different from K3PREP and then is converted by foamToKIVA4Mesh. In the first case, a stationary device is generated by using the meshing program blockMesh. In the second case, the mesh is an unstructured piston-bowl engine generated by Gmsh. These methodologies propose to create the entire mesh without using K3PREP, thus they must define the volume type of each element and the face type for each boundary surface.

In the last two cases, a software program is used to complement K3PREP and foamToKIVA4Mesh. These methodologies propose to create an initial mesh with K3PREP for the geometries that are possible with this program. Then, the mesh for the remaining geometries is created with another meshing software and then converted to the KIVA format using foamToKIVA4Mesh. It must be taken into account that the two meshes must share a surface to allow the patching of the meshes. This surface must have the same number of vertices and elements. Finally, the meshes must be patched with a third software to obtain a single mesh, which must be tested in KIVA-4 to verify its functionality. It should be noted that in this group of methodologies the use of K3PREP is done according to the indications stipulated in the corresponding software manuals (Amsden, 1997; Amsden, 1999).

4. STUDY CASE 1. STATIONARY ARBITRARY MESH

Originally, KIVA was developed to simulate engines, however, it can solve other phenomena such as reactive flow through pipes. A Venturi pipe was taken as an example. Air flows through the duct, and a diesel spray is placed in the middle of the throat. The purpose of this device is to evaporate diesel fuel.

The mesh plays an important role in the simulation of pipes. K3PREP allows to generate two types of mesh for this purpose: radial mesh and structured cartesian mesh. Each of these meshes has an undesirable feature in the use of KIVA for simulation. The radial mesh concentrates elements of small size and high aspect ratio in its center (Fig. 2(a)). For this reason, the simulation needs a small time step, and therefore the simulation duration is longer than the desired. The Cartesian mesh has elements with low orthogonal quality in the vicinity of four points of the cross-sectional area (Fig. 2(b)). These elements generate singularities during the simulation and add errors to results.

The concept of O-grid meshes takes the best of each of these types of mesh and allows obtaining better results in numerical simulations. The simplest O-grid is a grid structured by blocks composed of 5 blocks of cartesian mesh. This mesh does not have the mentioned problems of the other meshes. The elements have good size uniformity, small aspect ratio, and good orthogonal quality (Fig. 2(c)).

The program blockMesh is a basic utility of mesh generation in OpenFOAM. The final mesh was generated from blockMesh and converted into KIVA format by foamToKIVA4Mesh. K3PREP was not used at any time. The mesh is shown in Fig. 3.

5. STUDY CASE 2. ENGINE UNSTRUCTURED MESH

In its latest version, KIVA-4 is extended to handle unstructured meshes. However, K3PREP, the default mesh generator of KIVA, can only generate structured meshes. In the present section, an unstructured mesh was constructed for a diesel engine whose piston has a non-concentric bowl with the axis of the cylinder. The final mesh is shown in Fig. 4. The particularity of this mesh is that it is an unstructured mesh composed only of hexahedrons.

Gmsh was used to generate such hexahedral 3D base mesh. Gmsh is an open-source software. It has multiple functionalities that facilitate the mesh generation. The case in question (Fig. 4) was decomposed into two cylinders with

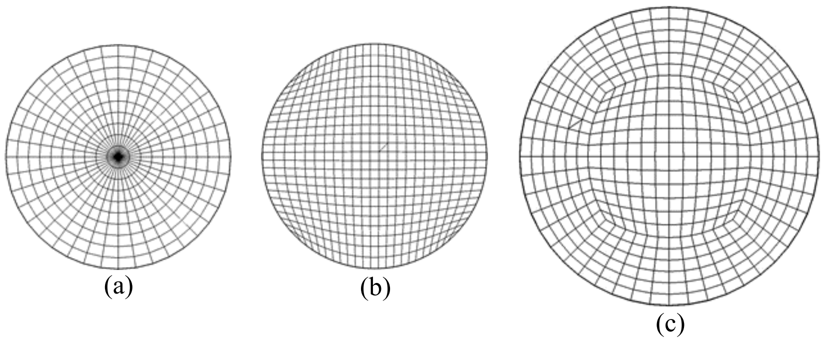


Fig. 2. Mesh types mentioned: (a) Radial (b) Cartesian (c) O-grid

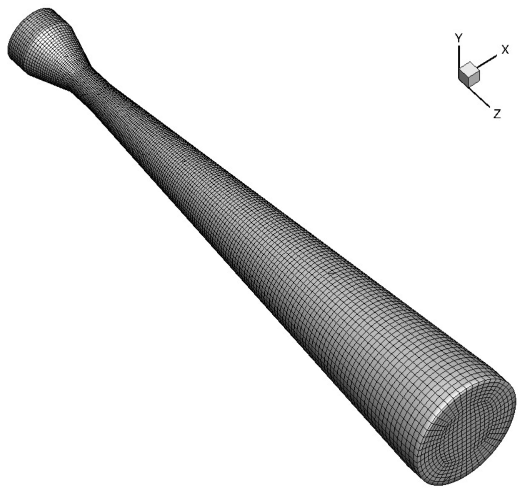


Fig. 3. Venturi mesh obtained with blockMesh and converted to KIVA format, 88992 cells and total volume of 1803(cm^3)

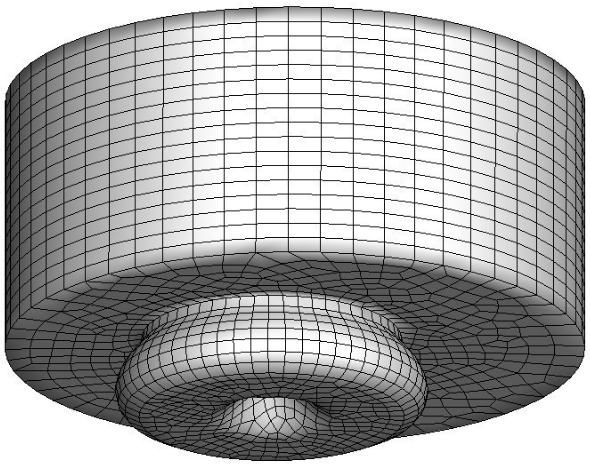


Fig. 4. Unstructured mesh obtained for an engine with eccentric bowl on piston. 23453 cells and displacement 406(cm^3)

different mesh sizes, which are joint in a face to get a base mesh. Fig. 5 and Fig. 6 show the top and lateral views of this mesh. Some important characteristics of this mesh are:

- There are two different zones. One for the bowl and one for the rest of the cylinder.
- The cell size is similar for all cells in each zone.
- It has smaller cell size for the bowl zone, which is the most important zone inside the engine.

After obtaining the base mesh, the vertices of the bowl must be rearranged to take the shape of the bowl (Fig. 7), which in this case is a solid of revolution around the z-axis. To change the coordinates of any vertex, the following procedure must be followed:

- Read the original coordinates of the vertex (x_a, y_a, z_a), and turn them into cylindrical coordinates (r_a, θ_b, z_a).
- Define the radial proportion of the vertex with the coordinate r_a.
- Define the axial proportion of the vertex with the coordinate z_a.
- Starting from segments 1-2 and 4-3 of the bowl geometry, the vertical curve (V) corresponding to the radial proportion is constructed.
- Starting from segments 1-4 and 2-3 of the bowl geometry, the horizontal curve (H) corresponding to the axial proportion is constructed.
- Calculate the coordinates (r_b,z_b) of the point where the H and V curves intersect.
- The θ_b coordinate remains the same
- Convert the found coordinates (r_b,θ_b,z_b) to Cartesian coordinates which are the transformed coordinates of the vertex (x_b,y_b,z_b).

After transforming the mesh, it is converted to the OpenFOAM format using gmshToFoam. Finally, it is converted to the KIVA format through the use of foamToKIVA4Mesh.

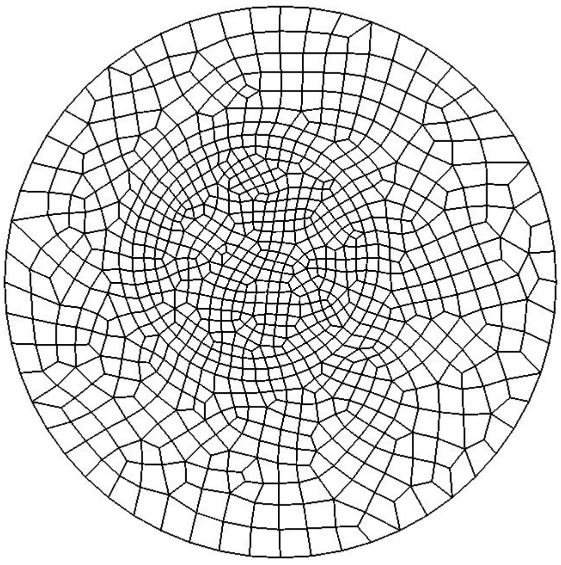


Fig. 5. Top view of the full unstructured mesh

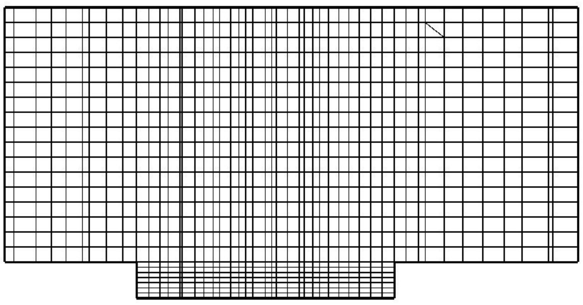


Fig. 6. Longitudinal split of the full unstructured mesh, before of conversion

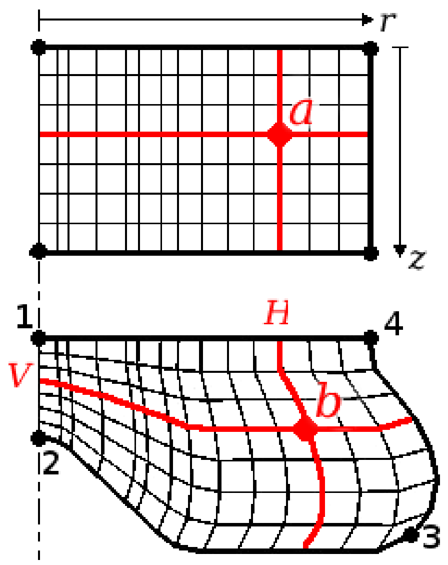
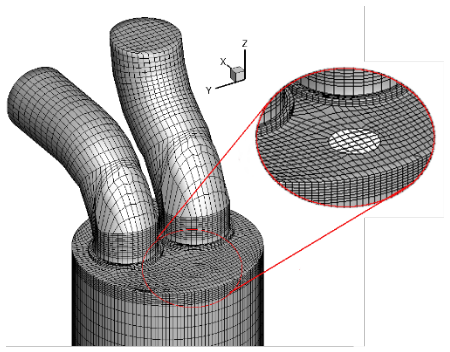


Fig. 7. Process of reshape of the bowl

6. STUDY CASE 3. ENGINE WITH VALVES AND A PRE-CHAMBER

In this section, a mesh was prepared for a gasoline engine with a pre-chamber. The mesh was built integrating two meshing programs: blockMesh and K3PREP. This is an example of a hybrid mesh, where a structured mesh is used for the cylinder, valves, and inlet and exhaust ducts, while a block structured mesh is employed for the pre-chamber. This allows a good approximation to the complex shape of the combustion chamber, preserving a structured mesh for the rest of the engine.

The process followed to obtain the structured mesh (for the cylinder, valves, and ducts) is similar to the one shown in the KIVA manual, with the particularity that a new block is included through the cylinder blocks. This block gives the shape of the interface between the cylinder and the pre-chamber. Fig. 8 shows the result of the mesh generation with K3PREP, note the shape of the place where the pre-chamber will be connected.


(a) Detail of the interface between the cylinder and the pre-chamber

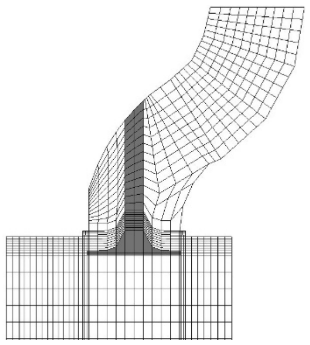

(b) Slice of the mesh at intake port with valve shadowed

Fig. 8. Structured mesh obtained for the cylinder, valves and ducts. 49812 cells

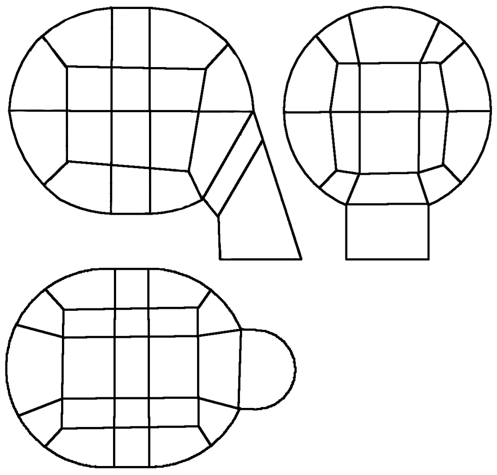


Fig. 9. Sketch of blocks used in mesh generation of the pre-chamber

The program blockMesh was used for the generation of the grid structured by blocks for the combustion chamber. The mesh was created from the concept of O-grid meshes. In the process, 62 blocks were used: 22 for each hemisphere, 16 for the section between the hemispheres and 2 for the connection duct to the rest of the engine. A sketch of the blocks used is shown in Fig. 9. Finally, the created mesh must be converted to the KIVA format using foamToKIVA4Mesh. Fig. 10 shows the result of this part of the mesh generation with blockMesh.

Then, a program was created by the research team to merge the meshes, based on the following rules:

- The coordinates of the center of the connection face of the pre-chamber are zero.
- The number of vertices and faces must be the same both in the connecting face of the pre-chamber and in the connection interface on the cylinder head.

Fig. 11 shows the complete mesh for an engine with a pre-chamber obtained at the end of the process. This mesh is fully functional in KIVA.

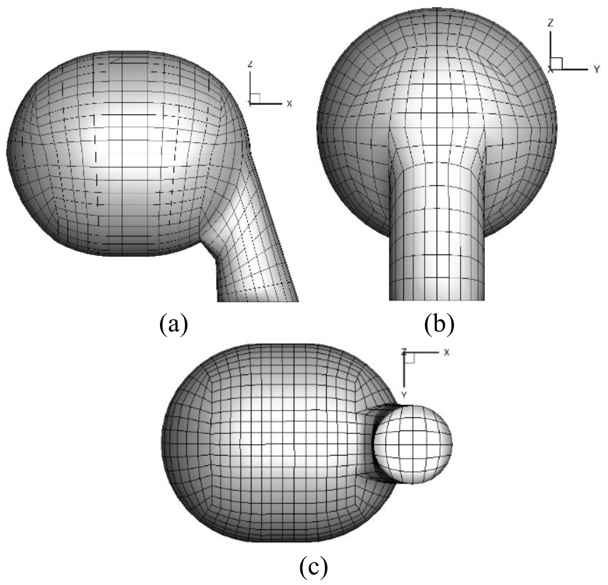


Fig. 10. Block structured mesh obtained for the pre-chamber (a) Lateral view (b) Derriere view (c) Bottom view. 18432 cells

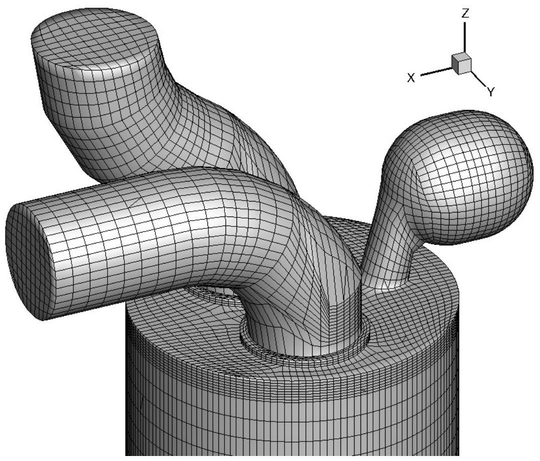


Fig. 11. Complete mesh obtained for a Lister engine with valves and pre-chamber. 68244 cells and displacement 804(cm^3)

7. STUDY CASE 4. ENGINE WITH VALVES AND NON-AXISYMMETRIC BOWL

Currently, the meshing programs available for KIVA-4 can be employed to develop meshes for engines with valves and pistons with a bowl, as long as it is obtained by revolution. This occurs because, as explained in the manuals of KIVA-3V (Amsden, 1997; Amsden, 1999), the geometry of the bowls is described as a function of the radius. For this reason, it was necessary to develop auxiliary meshing tools, applicable to engines with valves and non-axisymmetric bowls. To solve this problem, a new methodology is proposed to obtain meshes for arbitrary bowls, in a Cartesian or polar coordinates, as desired. This methodology is based on a code to obtain the mesh of the desired bowl in function of the mesh of the rest of the engine, obtained with K3PREP, and on the use of free software to patch these two meshes together.

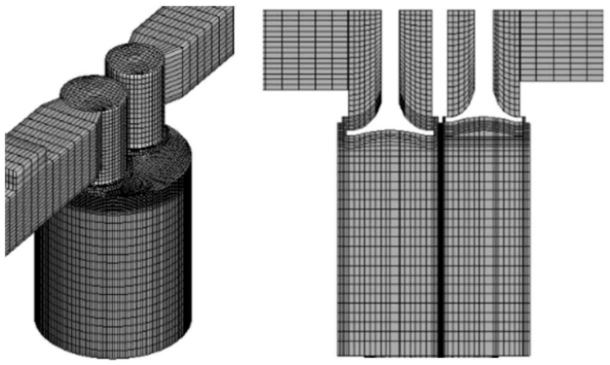


Fig. 12. Lombardini LGW-523 engine mesh, considering a flat piston. 100292 cells

The Lombardini LGW-523 engine is taken as a real example (Arroyo et al., 2014; Kosmadakis et al., 2015). This engine has two valves per cylinder and its piston has a non-axisymmetric bowl. First, the entire computational domain is meshed without considering the piston bowl, which is achieved by means of K3PREP. After this process, a large part of the engine mesh is obtained, and only the piston bowl mesh is missed, as shown in Fig. 12.

7.1 Piston Bowl Mesh

To discretize the piston bowl, a code was developed. It reads the kiva4grid file, obtained previously from K3PREP. Initially, it is necessary to define the piston bowl based on elementary geometric entities, in order to simplify the generation of its mesh. In this case, after the analysis of the image of the Lombardini LGW-523 engine, the computational domain of the piston bowl is modeled in a CAD software. It is concluded that the bowl of this engine is a conical hole in the piston with two truncations. Following these considerations, the code that allows to obtain the bowl mesh is developed. Fig. 13 shows the structure of this code.

As Fig. 13 shows, to find the vertices of the upper surface of the bowl, a 2D mesh is necessary to define the first geometric restrictions, and then to take only the vertices that comply with them. In the kiva4grid file, the cells are defined based on the indexes of their vertices (Torres, 2007). Then, the cells that contain at least one of the vertices that comply with the restrictions are identified. This process is done to achieve the continuity of the 2D mesh obtained with the bowl-less engine mesh, so that the subsequent patching process is successful. In the previous process, although they are not within the constraints, the vertices that were not considered form cells with vertices that do comply with restrictions.

Therefore, it is necessary to save these new vertices in arrays, that is, the entire mesh of the bowl upper surface. Fig. 14 illustrates the vertices initially obtained that meet the restrictions, and the total vertices that comprise the 2D bowl upper surface mesh. In addition, Fig. 15 illustrates the final 2D mesh that forms the bowl upper surface.

In Fig. 13, extrusions are defined to obtain the new vertices, which in turn define the piston bowl bottom surface. It should be noted that these extrusions are normal to the 2D mesh (along the z coordinate). Vertices which do not comply with the restrictions to obtain the 2D mesh are extruded. The minimum extrusion length is defined in the input variables. The other vertices extrusions are defined according to the bowl geometry. Besides, to generate all vertices of the 3D bowl mesh each extrusion is divided into the number of divisions in z, which is defined in the input variables. Thus, new vertices inside the bowl domain are obtained.

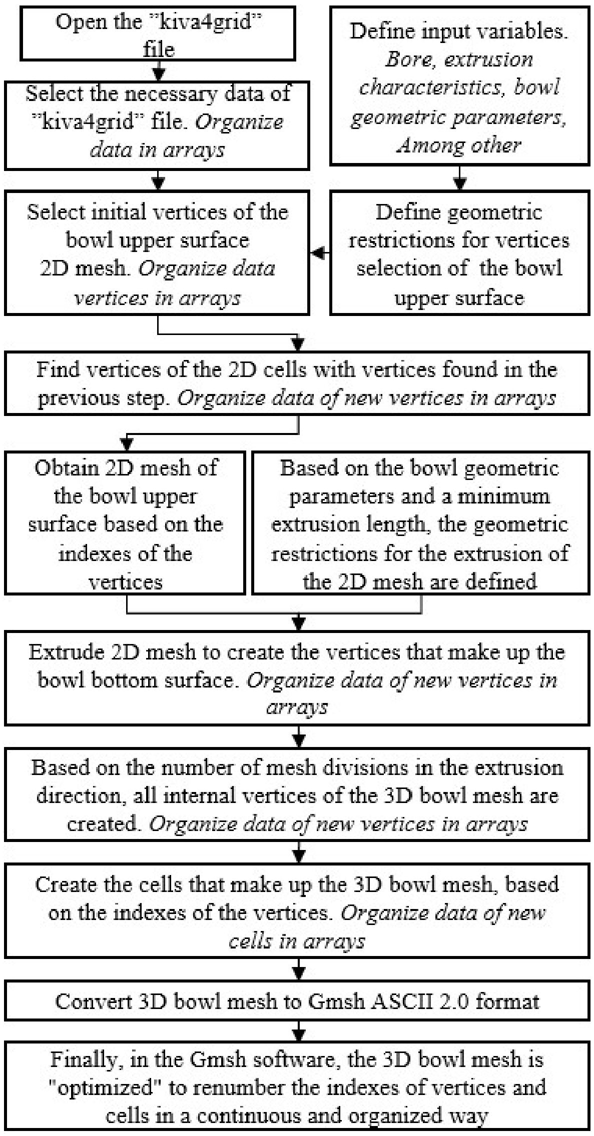

Open the "kiva4grid" file
Select the necessary data of "kiva4grid" file. *Organize data in arrays*
Define input variables. *Bore, extrusion characteristics, bowl geometric parameters, Among other*
Select initial vertices of the bowl upper surface 2D mesh. *Organize data vertices in arrays*
Define geometric restrictions for vertices selection of the bowl upper surface
Find vertices of the 2D cells with vertices found in the previous step. *Organize data of new vertices in arrays*
Obtain 2D mesh of the bowl upper surface based on the indexes of the vertices
Based on the bowl geometric parameters and a minimum extrusion length, the geometric restrictions for the extrusion of the 2D mesh are defined
Extrude 2D mesh to create the vertices that make up the bowl bottom surface. *Organize data of new vertices in arrays*
Based on the number of mesh divisions in the extrusion direction, all internal vertices of the 3D bowl mesh are created. *Organize data of new vertices in arrays*
Create the cells that make up the 3D bowl mesh, based on the indexes of the vertices. *Organize data of new cells in arrays*
Convert 3D bowl mesh to Gmsh ASCII 2.0 format
Finally, in the Gmsh software, the 3D bowl mesh is "optimized" to renumber the indexes of vertices and cells in a continuous and organized way

Fig. 13. Flowchart of code developed to obtain the Lombardini LGW-523 engine bowl mesh

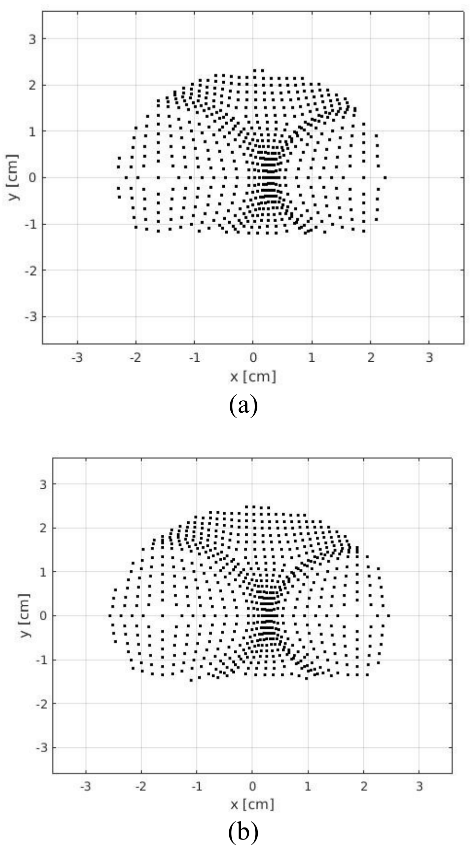


Fig. 14. Initial vertices (a) and definitive vertices (b) of the 2D mesh for the Lombardini LGW-523 bowl

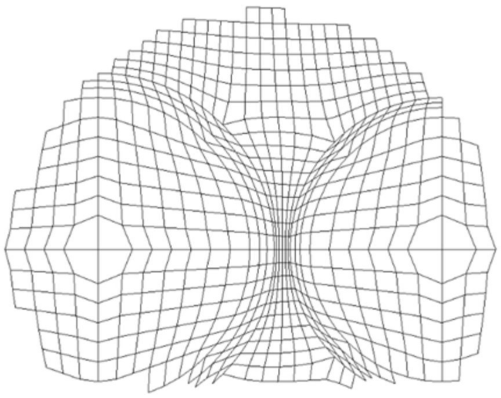


Fig. 15. 2D mesh that forms the upper surface of the Lombardini LGW-523 engine bowl

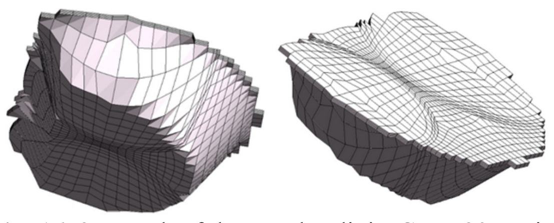


Fig. 16. 3D mesh of the Lombardini LGW-523 engine bowl. 3840 cells

Then, the cells of the 3D bowl mesh are defined, based on the vertices and the conventions of the kiva4grid file for the formulation of hexahedral cells (Torres, 2007). These cells will be formed according to the 2D mesh, which is repeated for each division in z, as observed when comparing Fig. 15 and Fig. 16. It should be noted that the similarity of the 3D mesh obtained to the real geometry depends directly on the refinement of the 2D mesh collected from the kiva4grid file generated with K3PREP. Finally, the conversion to the Gmsh ASCII 2.0 format is performed in accordance with section 9.1 (MSH ASCII file format) of the Gmsh 3.0 manual (Geuzaine and Remacle, 2018). The optimization of the mesh is done using the Gmsh software (Geuzaine and Remacle, 2017).

7.2 Boundary Conditions and Volume Type of Bowl

With the optimized bowl mesh in the Gmsh ASCII 2.0 format, its boundary conditions and the type of volume are defined as follows.

1. As expected, all the cells in the bowl mesh have the same volume type, because they belong to the same engine section. Then, using the software Gmsh (Geuzaine and Remacle, 2017) and the geometry section, a new group of physical identities (physical groups) is created and it is called "bowl". This new identity is applied to all the cells selected as members of the bowl mesh. Therefore, the mesh of the bowl is obtained with the volume type defined for each cell in the Gmsh ASCII 2.0 format.
2. Then, it is necessary to transform the mesh to OpenFOAM format (OpenFOAM Foundation, 2017), because this software is used to assign the boundary conditions with the gmshToFoam utility.
3. The boundary conditions of the bowl are:
   - Fluid boundary for the upper surface (face to stick with the rest of the mesh).
   - Boundary in movement with the piston for the other surfaces.

7.3 Mesh Patching

In this section, the bowl mesh is patched with the mesh considering the flat piston, thus obtaining the final engine mesh. After the bowl mesh is in OpenFOAM format, the conversion to the kiva4grid format is done. To achieve this, the subroutine foamToKIVA4Mesh is used. Thus, a new kiva4grid file is obtained, containings the mesh of the bowl. Finally, using the code developed to patch meshes, both meshes are joined.

This code takes as input the files "kiva4grid.motor" and "kiva4grid.bowl". The output of the patching code is a *kiva4grid* file that contains the engine complete mesh. Fig. 17 illustrates the final mesh of the Lombardini LGW-523 engine, obtained with the illustrated methodology.

8. MESH VERIFICATION

All meshes constructed in this article were tested to validate their proper operation. Nohydro simulations were used to validate the correct motion. Motored simulations were used to check the functioning of flow subroutines. Finally, fired simulations were used to validate the functioning of the chemistry subroutines of KIVA-4.

Fig. 18 shows the results of cylinder pressure in KIVA-4 simulations using the constructed meshes. Engine 1 refers to the Lombardini engine of study case 4, it uses syngas as fuel. Engine 2 refers to the engine with the pre-chamber of study case 3, it uses gasoline as fuel. Engine 3 refers to the engine with a bowl on piston of study case 2, it uses diesel as fuel. Table 1 shows the general characteristics of each engine.

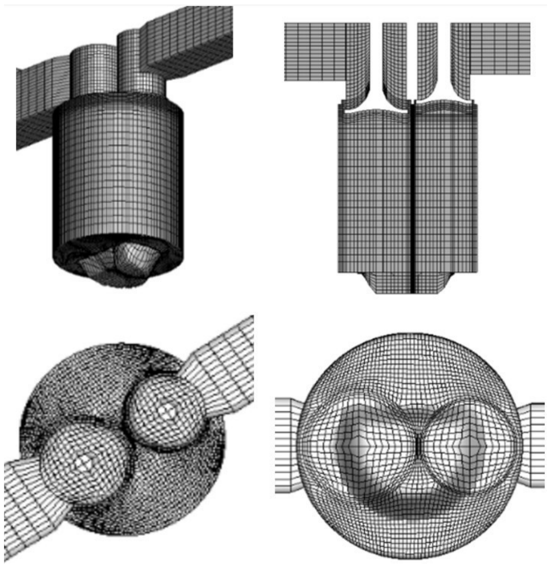


Fig. 17. Lombardini LGW-523 engine complete mesh. 104132 cells and displacement 252.5 (cm^3)

Table 1. General characteristics of the mentioned engines.

Engine	Bore (cm)	Stroke (cm)	Compression ratio	Fuel
1	7.2	6.2	10.7	Syngas
2	11.0	15.4	9.0	Gasoline
3	8.5	7.0	22.9	Diesel

For the venturi mesh, the airflow through the duct was simulated. Into the middle of the throat, diesel fuel was injected. Fig. 19 shows the air velocity and the percentage of fuel vapor along the duct.

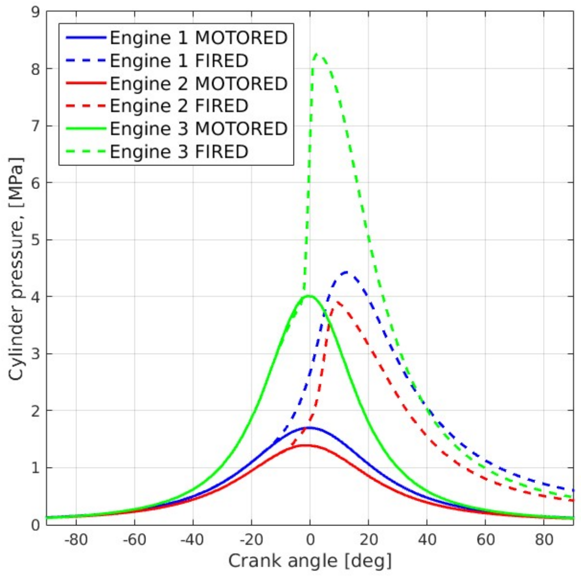


Fig. 18. Cylinder pressure obtained with meshes constructed

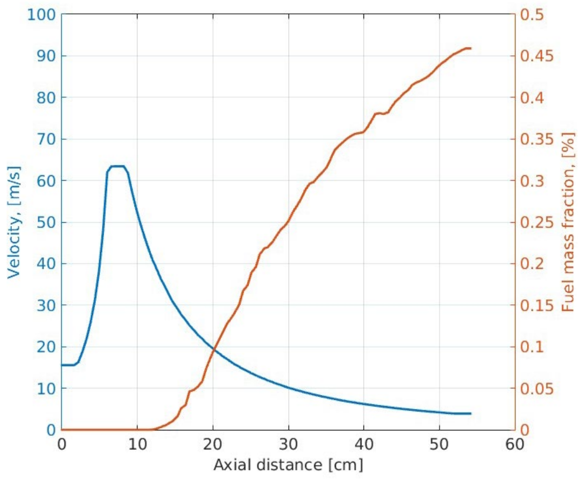


Fig. 19. Data obtained with Venturi mesh constructed

9. CONCLUSIONS

A methodology to extend unstructured mesh generation for KIVA-4 was presented. Validation via no-hydro, motored and fired simulations was carried out to corroborate the correct operation of the constructed meshes for real engines with complex geometries that feature a pre-chamber, a bowl and a mesh of a venturi tube. It is shown that the generation of meshes is simplified, and that the original mesh tool, K3PREP, can be used as a strong base for more complex geometries. A decreased learning curve is experienced by the user, as it is now possible to use their known CAD/mesh program if it supports any format compatible with OpenFOAM. As an OpenFOAM array of converters can be employed, many mesh formats can be imported from existing models.

It should be noted that in study case 4 it is not possible to accommodate the vertices in the mesh, initially generated with K3PREP, to match the contour of the bowl and have a smooth transition between both domains. This occurs because, unlike the pre-chamber and cylinder head in study case 3, the interface between the piston bowl and the flat piston in study case 4 is large enough to involve the mesh blocks belonging to the system of valves.

For this reason, in study case 4, modifying the position of the vertices is an unfeasible practice, since altering the axial direction of the cells with respect to the engine cylinder causes problems in the movement algorithm of the valves and the piston (Amsden, 1997; Torres, 2007). This leads to engine CFD simulations fail due to the presence of nonconvex and/or inverted cells (Amsden, 1997; Torres, 2007). As the vertices in the study case 4 cannot be accommodated, the results accuracy is critically dependent on the study of the mesh independence. This is mainly due to the fact that the precision with which the mesh describes the bowl geometry is directly proportional to the cell size of the base mesh developed in K3PREP, in the plane of interface with the bowl. However, as observed in the validation of the meshes section (more specifically in Fig. 18) and based on an adequate convergence study, it is possible to obtain consistent and real results from the methodology developed in study case 4, for both the motored operation and the fired operation of the engine (Pérez Gordillo, 2019).

As an extension of this tool it is recommended to include other cell shapes supported by KIVA-4 (Pyramid and tetrahedron). Furthermore, structured meshes could be supported even at low-level integration with KIVA-4, in order to change the mesh during execution, for adaptive mesh refinement, testing of different snapping algorithms, among others.

REFERENCES

Abani, N., Reitz, R. 2010. Diesel engine emissions and combustion predictions using advanced mixing models applicable to fuel sprays. Combustion Theory and Modelling, 14, 715–746.

Amsden, A. 1993. KIVA-3: A KIVA program with blockstructured mesh for complex geometries. Report, Los Alamos National Laboratory, USA.

Amsden, A. 1997. KIVA-3V: A block-structured kiva program for engines with vertical or canted valves. Report, Los Alamos National Laboratory, USA.

Amsden, A. 1999. KIVA-3V, release 2, improvements to KIVA-3V. Report, Los Alamos National Laboratory, USA.

Arroyo, J., Moreno, F., Muñoz, M., Monné, C., Bernal, N. 2014. Combustion behavior of a spark ignition engine fueled with synthetic gases derived from biogas. Fuel; 117, Part A, 50–58.

Bestel, D, Bayliff, S, Marchese, A, Olsen, D, Windom, B, Xu, H. 2020. Multi-dimensional modeling of the CFR engine for the investigation of SI natural gas combustion and controlled End-gas autoignition. Proceedings of the ASME 2020 Internal Combustion Engine Division Fall Technical Conference. ASME 2020 Internal Combustion Engine Division Fall Technical Conference. Virtual, Online. V001T06A012. ASME. https://doi.org/10.1115/ICEF2020-2992

Carrington, D., Waters, J., Weismiller, M. 2018. FEARCE development: A robust and accurate engine modeling software. FY2018 Annual progress report. Los Alamos National Laboratory. Available at shorturl.at/glswY Accessed: February 20 2020

Chawner, J., Taylor, N. 2019. Progress in geometry modeling and mesh generation toward the CFD vision 2030. AIAA aviation forum. 1–13. https://doi.org/10.2514/6.2019-2945

Chawner, J.R., Dannenhoffer, J., Taylor, N. 2016. Geometry, mesh generation, and the CFD 2030 vision. In 46th AIAA Fluid Dynamics Conference. 1–16. https://doi.org/10.2514/6.2016-3485

Geuzaine, C., Remacle, J. 2009. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79, 1309–1331.

Imamori, Y., Hiraoka, K., Murakami, S. et al. 2009. Effect of mesh structure in the KIVA-4 code with a less mesh dependent spray model for DI diesel engine simulations. SAE Int. J. Engines, 2, 1764–1776. https://doi.org/10.4271/2009-01-1937.

Jafarmadar, S., Zehni, A. 2014. Numerical investigation of the effects of dwell time duration in a two-stage injection scheme on exergy terms in an IDI diesel engine by three-dimensional modeling. Energy Science & Engineering, 2, 1–13.

Kim, H., Lee, S., Kim, H.J., Chun, J. 2020. Numerical study on the effects of tumble and swirl on combustion and emission characteristics of an LPG direct injection engine. International Journal of Automotive Technology. 21, 623–632.

Kosmadakis, G.M., Rakopoulos, D.C., Rakopoulos, C.D. 2015. Investigation of nitric oxide emission mechanisms in a SI engine fueled with methane/hydrogen blends using a research CFD code. Int J of hydrogen energy; 40, 15088–15104.

Lee, S., Park, S. 2010. Optimization of the piston bowl geometry and the operating conditions of a gasoline-diesel dual-fuel engine based on a compression ignition engine. Energy 121, 423–448.

Nishad, K. 2018. Analysis of spray dynamics of urea-water-solution jets in a SCR-DeNOxsystem: An LES based study. International Journal of Heat and Fluid Flow 70, 247–258.

Park, S. 2010. Optimization of combustion chamber geometry for stoichiometric diesel combustion using a micro genetic algorithm. Fuel Processing, 91, 17421752.

Park, S., Shin, H. 2012. Efficient generation of adaptive Cartesian mesh for computational fluid dynamics using GPU. International journal for numerical methods in fluids. 70, 1393–404.

Pérez Gordillo D. 2019. Estudio computacional de la combustión premezclada de un gas producto de la gasificación de biomasa en un motor de combustión interna (MCI). Tesis de Maestría en Ingenieria Mecánica, Universidad Nacional de Colombia, sede Bogotá.

Rainsberger, R., Fong, J., Marcal, P. 2018. Application of an a priori Jacobian-Based Error Estimation Metric to the Accuracy Assessment of 3D Finite Element Simulations. Proceedings of the ASME 2018 Pressure Vessels and Piping Conference. Volume 6B: Materials and Fabrication. Prague, Czech Republic. V06BT06A076. ASME. https://doi.org/10.1115/PVP2018-84784.

Sharma, C., Anand, T., Ravikrishna, R. 2010. A methodology for analysis of diesel engine in-cylinder flow and combustion. Progress in Computational Fluid Dynamics, 10, 157–167.

Shi, Y., Liu, Y.F., Qiu, Z.H. 2016. Mesh generation of engine combustion chamber based on KIVA-3V. Mechanics and Mechanical Engineering, 446–450.

Shimada, K. 2006. Current trends and issues in automatic mesh generation. Computer-Aided Design and Applications, 3, 741–750.

Torres, D., Li, Y., Kong, S. 2009. Partitioning strategies for parallel KIVA-4 engine simulations. Computers and Fluids, 39, 301–309.

Torres, D.J. 2007. KIVA-4 manual. Report, Los Alamos National Laboratory, USA.

Torres, D.J., Trujillo, M.F. 2006. An unstructured ALE code for compressible gas flow with sprays. J of Computational Physics 219, 943–975.

Weller, H., Tabor, G., Jasak, H., Fureby, C. 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in physics, 12, nov/dec.

Wickman, D., Senecal, K., Reitz, R. 2001. Diesel engine combustion chamber geometry optimization using genetic algorithms, multi-dimensional spray, and combustion modeling. SAE paper 2001–01–0547.

Wisconsin Engine Research Consultants (WERC). 2018. Mesh generation manual. Manual, Wisconsin Engine Research Consultants, USA.

Xue, Q., Kong, S., Torres, D., Xu, Z. Yi, J. 2008. DISI Spray Modeling Using Local Mesh Refinement. SAE Technical Paper 2008-01-0967, https://doi.org/10.4271/2008-01-0967.

Xue, Q., Song, C. 2009. Development of adaptive mesh refinement scheme for engine spray simulations. Computers & Fluids; 38, 939–949.

XYZ Scientific Applications, Inc. 2002. TrueGrid. Software.

Yi, J. 2008. Rapid mesh generation and dynamic mesh management for KIVA-3V. Report, Ford Research Laboratories, USA.

Yu, S., Hai-Wen, G., Reitz, R. 2011. Computational optimization of internal combustion engines. 1st ed. Springer-Verlag London.

Yuan, H., Yildiz, M., Merzari, E., Yu, Y., Obabko, A., Botha, G., Busco, G., Hassan, Y., Nguyen, D. 2020. Spectral element applications in complex nuclear reactor geometries: Tet-to-hex meshing. Nuclear Engineering and design, 357, 1–14.