Abstract

The iSALE hydrocode is a world-class tool to solve problems such as meteorite impacts. Due to the lack of regridding functions in iSALE3D, unnecessary time costs and computational costs can be produced. This project designed a regridding function for iSALE3D that can coarsen the resolution by a factor of two, and by comparing the time cost and accuracy of the simulations after regridding at different points in time, it is found that the best time for regridding was when the shock wave has just passed out of the impactor, thus realising accelerated simulations with an acceptable loss of accuracy of the results. The current regridding method has some limitations, for example, it can only be used when the number of cells in each dimension is a multiple of two. The principles, methodologies, and results of this project are presented in the paper. In addition, future work is also discussed for improvement.

Keywords: iSALE, numerical impact modelling, Fortran, MPI parallelisation, regridding, conservation laws

Acknowledgements

I would like to express my great thankfulness to my supervisors: Prof. Gareth S. Collins and Dr. Thomas M. Davision, who offered me a lot of support and advice. Without their patient help, I'm not able to complete the project successfully. Furthermore, I would like to thank all staff involved in delivering lectures and information to students. Finally, I would like to thank my partner Zifan Zhang for always encouraging me and helping me overcome difficulties in life.

1. Introduction

iSALE(impact Simplified Arbitrary Lagrangian-Eulerian) is a multi-material, multi-rheology shock physics hydrocode, which was based on a numerical fluid-dynamics computing technique SALE[1]. This program can be used for modelling impact cratering events over a range of scales, as well as researching the influences of different parameters on impact processes. At present, hypervelocity impact is regarded as one of the major factors that affect the formation of planets' surface and species evolution on Earth[2]. However, since large impact events generally involve huge bursts of energy and are expensive to simulate in the laboratory, numerical simulation on a computer is an effective and low-cost alternative. The process to quantify the effect of impacts is complicated, since either the properties of the impactor or the properties of the target control the formation of a crater[3]. In 1961, the first numerical impact simulation was developed[4], which laid the foundation for the development of hypervelocity impact modelling. At the early research stage, most of the impact modelings are carried out in 2D space, adopting an axis-symmetric method and assuming that the impact direction is perpendicular to the surface, to simplify the calculation during the simulation process. Nevertheless, in reality, the impact angle is random and most of the impact happens at 45°[5], while the vertical impact phenomenon is almost non-existent, which greatly hinders the realism of the impact simulation. In addition, the impact angle plays a significant role in geological problems, such as the giant impact theory of the origin of the moon[6]. Therefore, by applying 3D simulations to impact modelling, the impact angle can be taken into account, which ensures that the simulation could be closer to reality. In recent years, with the continuous exploration of developers, the functions of iSALE have been continuously improved in various aspects, including fragmentation models[1], a multi-material statistical model[7], a modified strength model[8], a porous compaction model[9][10] and a dilatancy model[11].

For large impact simulation, it can cause considerably expensive memory and computational costs when a high resolution is applied. Nevertheless, a high resolution is required at the early times of a cratering to capture the particles ejected with extremely high velocities[12]. A uniform high spatial resolution has the potential for waste of resources, while a uniform low resolution might increase the uncertainty of solutions. Therefore, regridding methods that allow a simulation domain to be coarsened can be useful. At present, regridding is frequently used in 2D simulations to speed up the calculation[12][13]. In this project, a regridding method is going to be designed in iSALE3D that works with the MPI parallelisation approach.

1.1 Fundamental theories

1.1.1 Governing equations

Most of the impact models suppose that there are continuous mediums in the interested region and the conservation equations of mass, momentum, and energy are applied[14](as shown below).

Conservation of Mass: — + — = − — (1)

Conservation of Momentum: — = + — (2)

Conservation of Energy: — = − — + ' (3)

This is the Eulerian form of the conservation equations. Here is the material density, is the velocity, is the external force to material per unit mass, is the stress tensor, is the specific internal energy, is the deviatoric stress tensor, and ' is the deviatoric strain rate.

In iSALE3D, there are eight vertices for each cell. Part of the variables in the conservation equations, such as the position(x, y, z) and velocity(u) are stored in vertices of the mesh, and the rest quantities, such as the pressure(p), density(ρ), viscosity(η), energy(E), volume(V) are stored in the center of a cell[13](as shown in Figure 1). Mass(M) is stored in both eight vertices and the cell. In this project, theoretically, the conservation laws should be observed throughout the impact simulation process, so compliance with conservation laws can be used as one of the criteria for evaluating whether a regridding method is reliable or not.

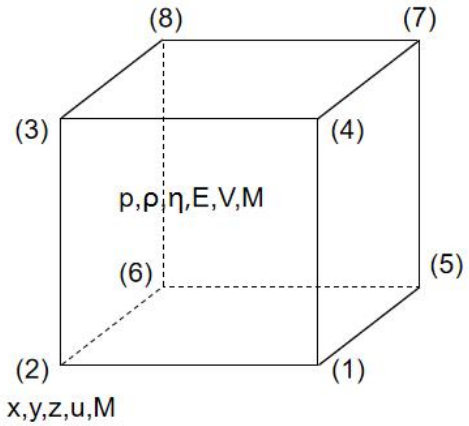


Figure 1: The assignment of variables and indexes for a cell in iSALE3D.

1.1.2 Discretization

Due to the limited memory allocation on computer hardware, the region of interest should be divided into a finite number of cells, and the calculation for the differential equations (1)-(3) only proceed on the points with spatial interval Δx and time interval Δt[6]. There are three versions of discrete difference equations for velocity u:

Forward difference: —(=) = $\frac{\Delta_{+1}-\Delta}{\Delta}$

Backward difference: —(=) = $\frac{\Delta-\Delta_{-1}}{\Delta}$ (4)

Central difference: $\frac{\partial}{\partial}(\quad = \quad) = \frac{\Delta_{\quad+1} - \Delta_{\quad-1}}{2\Delta}$

1.1.3 Frame of reference

There are two types of reference frames usually used in impact models: Lagrangian description and Eulerian description[14]. The Lagrangian description introduces a reference frame that moves with continuous mediums(as shown in Appendix A). The mass of each cell is constant, and as time progress, the shape and size of cells can vary due to the force on them. Therefore the cell's volume can be changed, which will lead to changes in density. On the contrary, the Eulerian description researches the material that passes through a fixed area, and in this case, all the points are fixed in the mesh. The cell's volume is invariant and the density varies with the change in the cells' mesh.

1.1.4 Resolution and regridding

During the simulation process, the selection of resolution is an important part of the start stage, which determines the computational cost and memory cost. The memory cost is generally proportional to the total number of cells in the mesh to be stored. For instance, if N is the number of cells in each dimension and M is the number of dimensions, the required memory space and calculation time will be proportional to N^M and N^{M+1} respectively[6]. For 3D simulation, the memory allocation cost will increase by a factor of 8 and the calculation time cost will increase by a factor of 16 if the resolution is doubled.

In impact modelling, researchers usually use the number of computational cells per projectile radius(CPPR) to describe the resolution of a model[16]. As shown below, if the CPRR is 5 and the grid space is 3, the radius will be CPPR*GRIDSPC = 5*3 = 15. Studies showed that different impact models have different requirements of resolution[17][18], which means it is difficult to choose a suitable resolution for an unknown model at the beginning. Thus, regridding methods are necessary for a modelling program to modify the resolution during the process of simulation based on the requirement. For example, selecting a high resolution at the early stages and regrid the model with a coarsened mesh at some point of the simulation. In addition, regridding methods can be used to add or remove cells as materials or waves move through the mesh and approach the boundaries of the simulation domain. In the current iSALE hydrocode, regridding functions are only employed in iSALE2D[19], hence, restarting the modelling from the start point of simulation is required every time a different resolution is selected in iSALE3D, which can cause a large amount of time wasted.

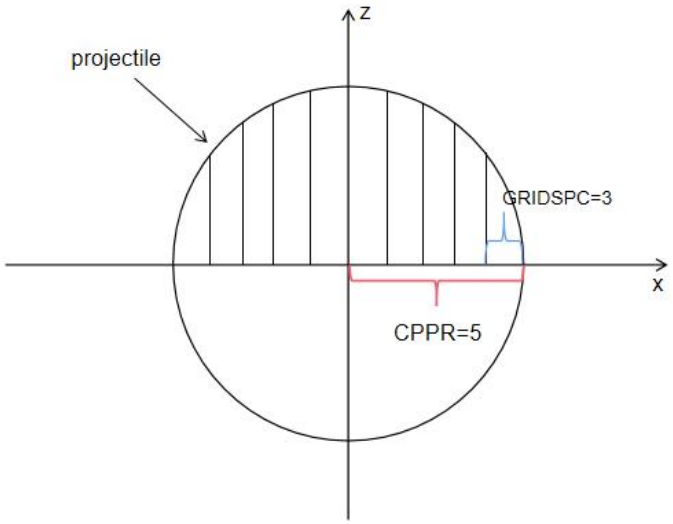


Figure 2: Cells per projectile radius(CPPR).

1.2 Objectives

This project aims to implement one coarsening regridding method in iSALE3D, meanwhile, keep the MPI approach applied to improve the efficiency of the regridding algorithm. The influences of regridding on the time cost and accuracy of simulation will be compared with the original code. By adding the regridding method into iSALE3D, users are able to inherit the previous model from the dump files and carry on the simulation with a lower resolution, which can mitigate the resource waste caused by unnecessary high resolution. The development of this project is based on the Fortran language.

This project is roughly divided into five stages: Firstly, since there is a lack of a suitable platform to test the function of a single subroutine for Fortran, the regridding method is implemented on the Python platform to check the feasibility of the algorithm. Secondly, the python code is transferred into Fortran, and performance is tested on a mini model. Thirdly, the MPI approach is involved and tested on the mini model. Fourthly, simulation with the regridding method is implemented on a larger model. Finally, the performance of added function for efficiency improvement is evaluated.

2. Software description/ methodology

In this section, the iSALE3D's structure and the development of the regridding method are described in detail. In addition, relevant methodologies applied in this project is elaborated. This project is developed based on the original iSALE3D code with Fortran in Linux. At the early stage, the regridding method is implemented in Jupyter Notebook with python. After that, the regridding method is added to the iSALD3D code and tested on two models.

2.1 iSALE3D framework

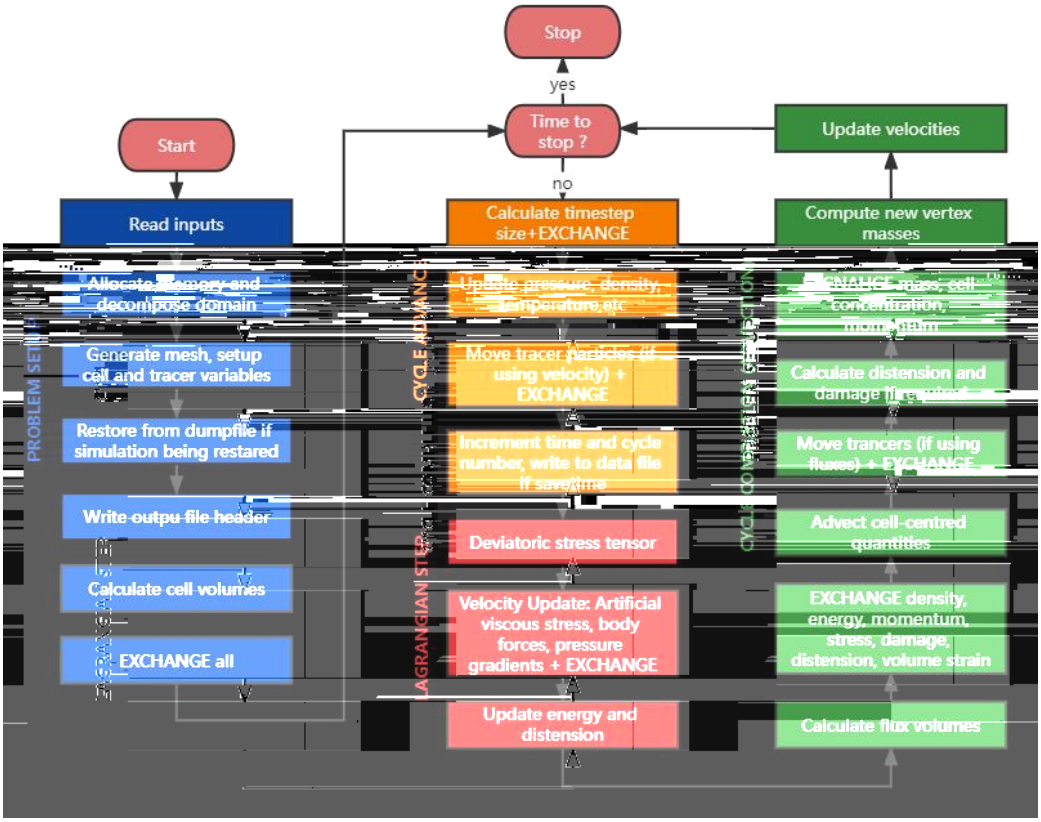


Figure 3: iSALE3D workflow.

As shown in Figure 3, the iSALE3D hydrocode includes four parts, which are problem setup, cycle advance, Lagrangian step, and cycle completion[19].

(a) At the stage of problem setup, the program processes the input file and reads the parameters to generate mesh, cells, and tracer variables with the required memory sizes allocated. When regridding, the previous model is restored from dump files. In iSALE3D, all field variables are three-dimensional scalar, vector, or tensor quantities, with three spatial indices i, j, and k. Once the model has been initialized, the program will start a time-increasing simulation loop.

(b) The program move to the next time step at the beginning of each cycle, then the optimal time step is recalculated and some of the cell-centered variables are updated. Meanwhile, the tracking particles are also moved during this phase. If the model needs to be stored, the relevant data, such as the number of materials, field variables, and the mesh, are written into dump files.

(c) At the stage of the Lagrangian step, the program computes the effects of all forces(artificial stabilizing forces, viscous, elastic stresses, etc.) to velocity field and internal energy field.

(d) At the final stage of a cycle, the calculation of volume fluxes across six faces of cells is processed. Then several cell-centered quantities are advected. Damage and distension are

calculated if required. Finally, vertex masses and velocity fields are updated. One cycle is completed now and the program goes into the next cycle.

Based on the above description, it can be seen that this project mainly focuses on the restoration process in the first stage.

2.2 MPI approach

Message-Passing Interface(MPI) has been the dominant parallel programming method since the 1990s[20]. By assigning tasks to different processes, the speed of the program will be effectively increased. Unlike the shared memory used by OpenMP, MPI applies a distributed memory system, which requires different processes to communicate with each other to exchange information. In MPI, there are two main types of communication, which are point-to-point communication and collective communication. For the former type, communication only occurs between a pair of processes, while for the latter type, communication is conducted between one process and every other process. In iSALE3D, since only cells and vertices on the boundaries of adjacent processes are involved during calculation, point-to-point communication is applied. At the stage of problem setup, domain decomposition is carried out along the x-direction at certain intervals. As detailed in Figure 4, ghost cells/nodes are generated at both sides of a subdomain to exchange and store data from adjacent processes. For cell-centered quantities, the range of fields is from is to ie, and the ghost cells are indexed with igs and ige respectively, where igs = is -1 and ige = ie + 1. For quantities on nodes, such as velocity fields, there is one more node than cells in each direction, therefore, the range is from is to iep, where iep = ige. The ghost nodes are indexed with igs and igep, where igep = iep + 1.

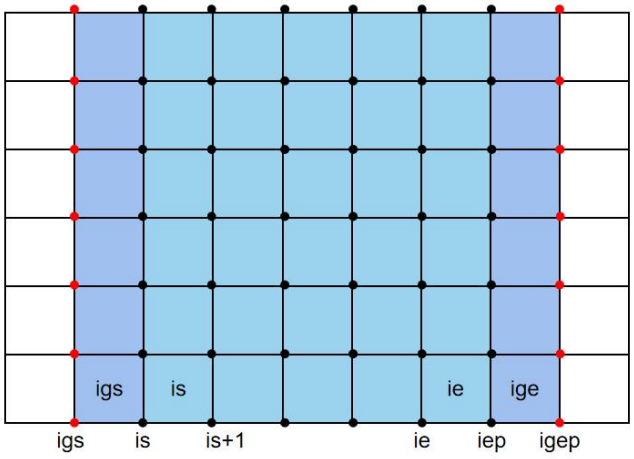


Figure 4: Ghost cells and nodes of a subdomain.

2.3 Regridding method

In this project, a regridding method that can coarsen the high-resolution zone by a factor of two or keep the original resolution is designed. As can be seen in Figure 5, when the regridding function is activated, the program will read the mesh size from dump files and determine whether coarsening is required. Then memory is allocated to read important fields

onto the temporary mesh. After that, the temporary mesh is coarsened(if necessary) and transferred onto the new mesh.

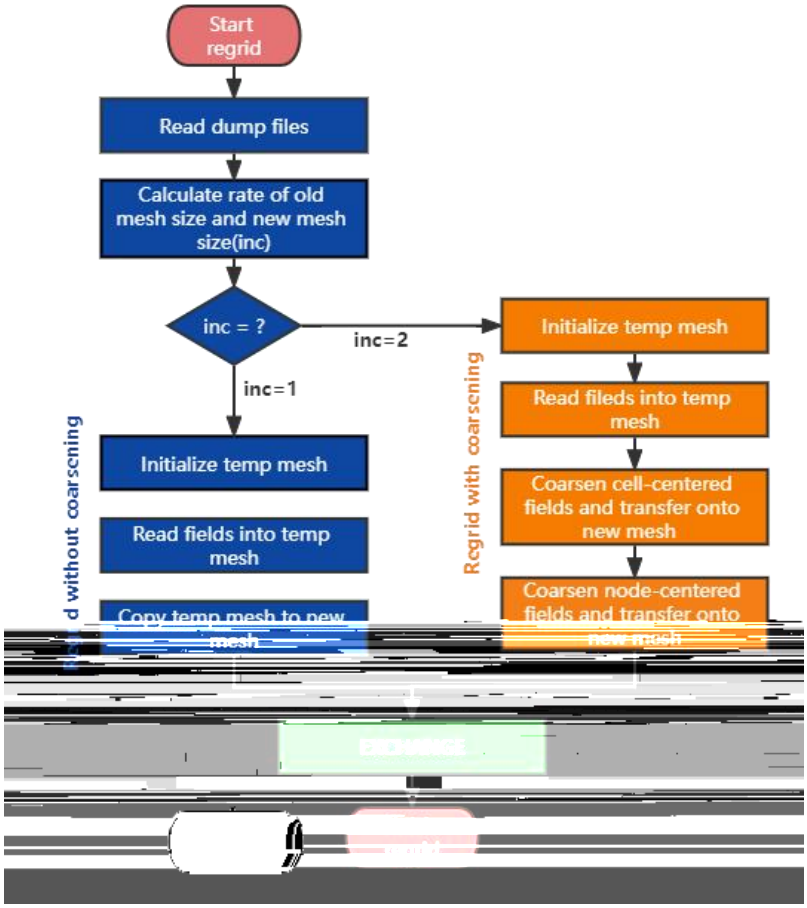


Figure 5: Regrid workflow diagram.

For non-coarsening regridding, only one copy process occurs. For coarsening regridding, depending on where the parameters are stored, this regridding process consists of two main parts: coarsening on vertices and coarsening on cells.

2.2.1 Coarsening of cells

The algorithm for coarsening a cell is shown below. Since most of the strategies for coarsening cell-centered fields are similar, here the vertex location position field(C), density field(ro), mass field(mc), material volume fraction(cmc), and specific internal energy field(sie) are involved as the example to implement the structure of this algorithm. Variables involved are listed as follows:

Inputs:

- iold, jold, kold: vertex location on temporary mesh
- inew, jnew, knew: vertex location on the new mesh
- nmat_in: number of materials in a cell

Outputs:

- coarsened cell-centered fields

Algorithm 1: Coarsen cell

Input: $iold, jold, kold, inew, jnew, knew, nmat\_in$

Output: ro, mc, cmc, sie

$totalmass[0: nmat\_in] \leftarrow 0;$ % total mass of one material of eight cells

$totalvol[0: nmat\_in] \leftarrow 0;$ % total volume of one material of eight cells

$volu[0: nmat\_in] \leftarrow 0;$ % volume of one material in one cell

$masse[0: nmat\_in] \leftarrow 0;$ % mass of one material in one cell

$mc_{inew,jnew,knew} \leftarrow 0;$

for $k \leftarrow kold$ **to** $kold + 1$ **do**

  for $j \leftarrow jold$ **to** $jold + 1$ **do**

    for $i \leftarrow iold$ **to** $iold + 1$ **do**

      $mc_{inew,jnew,knew} = mc_{inew,jnew,knew} + mc\_tmp_{i,j,k};$

      for $m \leftarrow 0$ **to** $nmat\_in$ **do**

        $volu[m] = (C\_tmp[x]_{i+1,j,k} - C\_tmp[x]_{i,j,k}) * (C\_tmp[y]_{i,j+1,k} - C\_tmp[y]_{i,j,k}) * (C\_tmp[z]_{i,j,k+1} - C\_tmp[z]_{i,j,k}) * cmc\_tmp[m]_{i,j,k};$

        $totalvol[m] = totalvol[m] + volu[m];$

        $masse[m] = ro\_tmp[m]_{i,j,k} * volu[m];$

        $sie[m]_{inew,jnew,knew} = sie[m]_{inew,jnew,knew} + masse[m] * sie\_tmp[m]_{i,j,k};$

      end for

    end for

  end for

end for

for $m \leftarrow 0$ **to** $nmat\_in$ **do**

  $cmc[m]_{inew,jnew,knew} = totalvol[m]/sum(totalvol);$

  if $totalvol[m] > 0$ **then**

    $ro[m]_{inew,jnew,knew} = totalmass[m]/totalvol[m];$

  else

    $ro[m]_{inew,jnew,knew} = 0;$

  end if

  if $totalmass[m] > 0$ **then**

    $sie[m]_{inew,jnew,knew} = sie[m]_{inew,jnew,knew}/totalmass[m];$

  else

    $sie[m]_{inew,jnew,knew} = 0;$

  end if

end for

In this algorithm, eight cells are merged into one cell. In addition, conservation of mass and conservation of energy are employed. Firstly, the total volume and mass of each material in eight cells are calculated. Then the total values of interested fields in eight cells are calculated. Finally, coarsened cell-centered fields are obtained by dividing the total value of fields by the total volume or total mass based on the characteristics of these fields.

2.2.2 Coarsening of vertices

Algorithm 2 is implemented to coarsen fields on vertices, which are the velocity fields(V). The variables in this algorithm are described as follows:

Inputs:

- ilocs, iloce: start and end indices of temp mesh on x direction
- jlocs, jloce: start and end indices of temp mesh on y direction
- klocs, kloce: start and end indices of temp mesh on z direction
- inew, jnew, knew: positions to store data on the new mesh

Output:

- V: new coarsened velocity field

Algorithm 2: Coarsen velocity

Input: , , , , , , , ,
Output:
[:] ← 0;
← 0;
for ← **to** **do**
  for ← **to** **do**
    for ← **to** **do**
      = + _ , , ;
      [:] = [:] + 0.125 _ , , (_ [:] , , +
      _ [:] +1, , + _ [:] , +1, + _ [:] , , +1 + _ [:] +1, +1, +
      _ [:] +1, , +1 + _ [:] , +1, +1 + _ [:] +1, +1, +1);
    end for
  end for
end for
if > 0 **then**
  [:] , , = [:]/ ;
else
  , , ← 0;
end if

As shown in Algorithm 2, conservation of momentum is applied. The total momentum and mass of the eight adjacent cells are calculated based on equation (5). Then the new velocity is obtained by dividing the total momentum by the total mass.

$$M = mv \tag{5}$$

3. Code metadata

The early stage of development for the regridding method is implemented on the web-based interactive computing platform JupyterLab[21] with Python 3.8. For the development of iSALE3D code, it is carried out on Ububtu 20.04 using Fortran 90. The designed regridding method can be found in /iSALE3D/F90/write_dump.F90 from IRP repository. Relevant files that need to be edited in iSALE3D are also in the directory /iSALE3D/F90/. The preliminary test script and fields plotting scripts used for visualization can be found in the directory /iSALE3D/Python/. The input files used for simulations are in the directory /iSALE3D/inp/.

For the whole iSALE3D code, please visit the branch ***ms_regird_3d*** of Github repository ***isale-dev***. The README.md file provides the instructions for pre-requisites. After the configuration completed, the iSALE3D code can be run under the dictionary /iSALE/example/demo3D/ with the following command:

```
mpirun -n 4 ./iSALE3D -i asteroid.inp &
```

Then to regrid the model, the command should be run under the same dictionary:

```
mpirun -n 4 ./iSALE3D --ignore -i regrid.inp &
```

The visualisation scripts are written with python 3.8.

4. Results

This section talks about the performance of the regridding method and discusses the effects on time cost and accuracy when starting regridding at different time points during simulation. In this project, a python script *regridding-test.ipynb* is designed to conduct a preliminary test on the regridding method. Then two impact models are adopted for simulation. One model is a small model with a high-resolution impactor, short simulation period, and small mesh size, which is used to check the feasibility of the regridding method and realize the visualization of the regridding process. Another one is a large model with a low-resolution impactor, longer simulation period, and large mesh size, which is used for researching the effects of the regridding method and the best regridding time point in a simulation process.

4.1 Preliminary test on regridding strategy

The plots and calculated results are shown in Appendix B. In the test file, a density field is designed to test the feasibility of the regridding method. The plots of density fields before and after coarsening can approve the correctness of this method. Then a random velocity field and a random mass field are adopted to check whether conservation laws are followed. The total mass of the mass field is calculated to be identical before and after the coarsening, and the total momentum of the model in the x,y,z directions is essentially the same, which proves that the conservation law is successfully applied. For more details, please refer to the python script.

4.2 Visualization of regridding

Table 1: Model information.

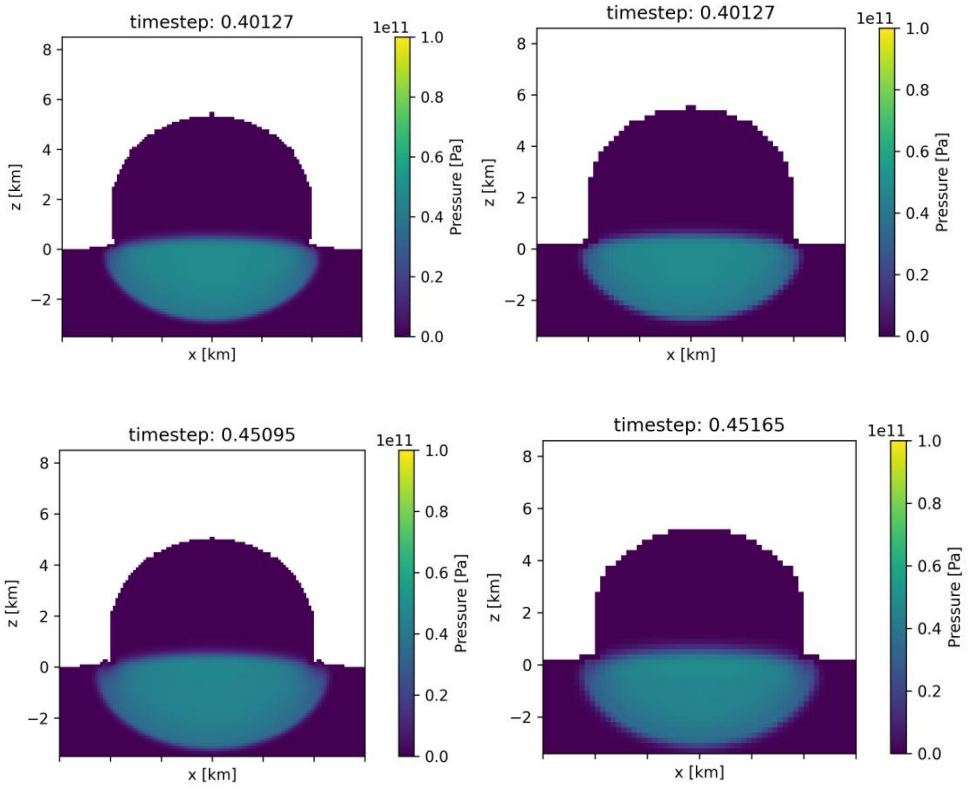
Model	Grid size	Grid space(m)	Object resolution
Before regridding	x: 121 y: 61 z: 121	dx: 100 dy: 100 dz: 100	x: 40 y: 40 z: 40
After regridding	x: 61 y: 31 z: 61	dx: 200 dy: 200 dz: 200	x: 20 y: 20 z: 20

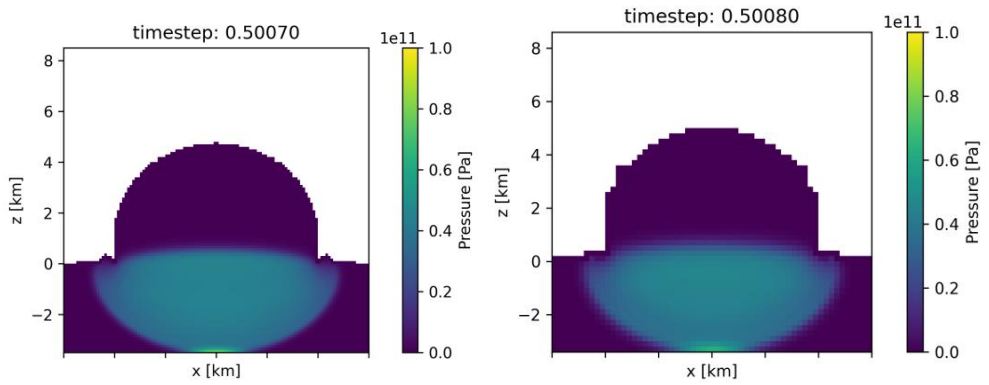
In the simulation of a small model, the total simulation time(virtual time) is set to 0.8s, the resolution of the projectile(CPPR) is set to 40, and regridding happens at T=0.4s. The model information before and after regridding which reads from python script is shown in table 1. The ratio of memory required for the original mesh and coarsened mesh can be calculated:

$$(121 \times 61 \times 121) / (61 \times 31 \times 61) \approx 8$$

Therefore, the memory cost can be reduced by a factor of eight after coarsening the mesh.

Figure 6 shows the plots of the non-regrided and regridded pressure fields at three timesteps after T=0.4. It can be seen from Figure 6 that the resolution is coarsened after regridding. The simulation domain is not changed, so the number of cells in each dimension is reduced by half and the size of each is doubled in each dimension.





(a) Before regridding (b) After regridding

Figure 6: No-coarsened and coarsened pressure fields after T=0.4s.

4.3 Regridding's effects on time cost and accuracy

In the simulation of a large model, the total simulation time(virtual time) is set to 3s, and the resolution of the projectile(CPPR) is set to 8. By investigating the images of the cratering, the shock wave has passed out of the impactor when T=0.2s, so we will explore the effects of regridding before and after this moment on time consumption and accuracy. In this section, five runs with different regridding points are conducted:

a. Keep high resolution throughout the simulation

b. Start from high resolution and coarsen the resolution by a factor of 2 at T=0.4s

c. Start from high resolution and coarsen the resolution by a factor of 2 at T=0.2s

d. Start from high resolution and coarsen the resolution by a factor of 2 at T=0.1s

e. Keep half resolution throughout the simulation

The results of time spent(real time) and the crater radius and volume at T=3s(end time) of the five simulations are shown in Table 2.

Table 2: Time costs for five simulations.

Simulation	Regridding point	Time cost(s)	Crater radius(m)	Crater volume(m^3)
a	None(T=3.0s)	$4.5234x10^3$	$2.5513x10^3$	$2.8931x10^{10}$
b	T=0.4s	$9.1055x10^2$	$2.5392x10^3$	$2.8643x10^{10}$
c	T=0.2s	$6.9276x10^2$	$2.5298x10^3$	$2.8212x10^{10}$
d	T=0.1s	$5.1798x10^2$	$2.4790x10^3$	$2.7401x10^{10}$
e	None(T=0s)	$3.8983x10^2$	$2.3398x10^3$	$2.3848x10^{10}$

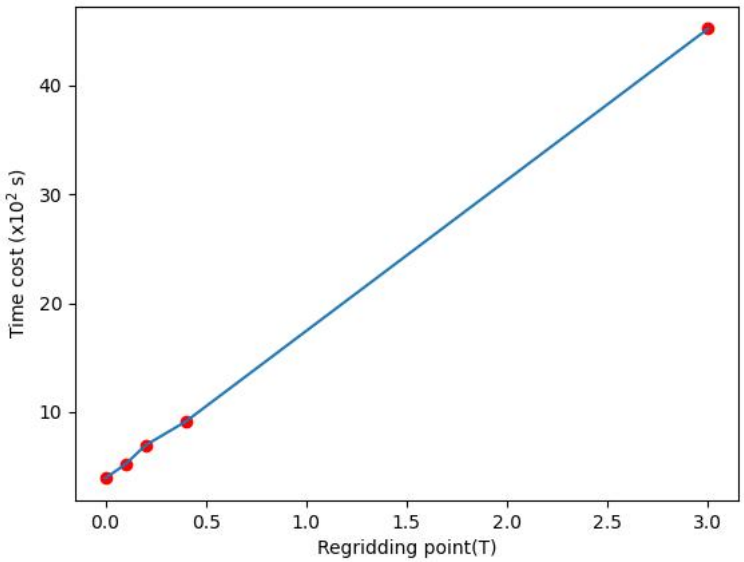


Figure 7: Time cost vs Regridding point.

As can be seen from Table 2, for time expense, the origin simulation time is roughly 11 times longer than the simulation with half resolution. In addition, the earlier the regridding is employed in the simulation, the shorter the simulation time will be required. As simulation **a** and **e** can be regarded as regridding at the end time(T=3.0s) and start time(T=0s) respectively, the relationship between the regridding point and time cost can be plotted as shown in Figure 7. It can be observed that the time cost and regridding point are linearly positively correlated.

For accuracy of simulation, as can be seen from the radius and volume of the crater, the later the regridding happens, the closer the simulation results are to the high-resolution simulation results, i.e. the true results(assuming the original resolution is high enough to reach the real solution). However, the regridding time point is not linearly correlated with the variation in accuracy. The errors of crater radius and volume in half-resolution simulation are:

$$\text{Radius: } 2.5513 \times 10^3 - 2.3398 \times 10^3 = 0.2115 \times 10^3$$

$$\text{Volume: } 2.8931 \times 10^{10} - 2.3848 \times 10^{10} = 0.5083 \times 10^{10}$$

If regridding T=0.2s, the improvement in accuracy is:

$$\text{Radius: } \frac{(2.5298 \times 10^3 - 2.3398 \times 10^3)}{(0.2115 \times 10^3)} \times 100\% \approx 89.93\%$$

$$\text{Volume: } \frac{(2.8212 \times 10^{10} - 2.3848 \times 10^{10})}{(0.5083 \times 10^{10})} \times 100\% \approx 85.85\%$$

If regridding at T=0.4s, the improvement in accuracy is:

$$\text{Radius: } \frac{(2.5392 \times 10^3 - 2.3398 \times 10^3)}{(0.2115 \times 10^3)} \times 100\% \approx 94.28\%$$

$$\text{Volume: } \frac{(2.8643 \times 10^{10} - 2.3848 \times 10^{10})}{(0.5083 \times 10^{10})} \times 100\% \approx 94.33\%$$

Therefore, for crater radius, from fully half-resolution simulation to regridding at T=0.2s, the accuracy is increased by 89.83%, while from regridding at T=0.2 s to T=0.4s, the accuracy is improved by only $94.28\% - 89.83\% \approx 4.44\%$. For crater volume, from fully half-resolution simulation to regridding at T=0.2s, the accuracy is increased by 85.85%, while from regridding at T=0.2 s to T=0.4s, the accuracy is improved by only $94.33\% - 85.85\% \approx 8.48\%$. Hence, considering both the time expense and accuracy, the best time for regridding in the impact simulation process should be the point when the shock wave has passed out of the impactor, which can speed up the simulation without causing a significant loss of accuracy. When regridding at T=0.2s, the time cost reduced is:

$$\frac{4.5234 \times 10^3}{6.9276 \times 10^2} \approx 6.5$$

5. Discussion & Conclusions

Numerical impact modelling is a significant technique to help scientists study planetary origins and crater formation. iSALE hydrocode is designed to explore problems such as impact events at various scales. Large-scale impact simulations often require sufficiently high resolution in regions or periods of interest to ensure realistic simulations, which can result in long time cost and memory consumption in unique resolution simulations. To address this issue, regridding allows simulations to be coarsened in resolution at any point in time, thereby speeding up the simulation. Currently, regridding functions are only designed in iSALE2D, whereas 3D simulations are essential to improve the reliability of the simulation. Therefore, this project aims to realise a regridding function in iSALE3D.

In this project, a regridding method that can coarsen high-resolution zone by a factor of two is designed. In addition, comparing the results of regridding at various time points, it is found that the best regridding point is when the shock wave has just passed out of the impactor. There are several tough tasks during development. Firstly, conservation laws of mass, energy, and momentum should be met when coarsening the mesh. Secondly, since iSALE is a huge program that has been developed for over a decade, there will be a long and slow debugging progress when getting the regridding function to work in iSALE3D code. Thirdly, domain decomposition is required and the ghost layer should be considered when applying MPI parallelisation.

In conclusion, the objectives for the project are basically completed. The designed regridding method allows reducing the time cost by about 90% on the test model without causing considerable loss of accuracy. However, there are still several limitations in this regridding method. Firstly, the number of cells in each direction must be a multiple of two. Secondly, the regridded grid size, projectile position, and projectile resolution need to be changed manually in the input file. For further development, the robustness of the code needs to be improved to handle the case of odd cells' numbers. Then, it would be better if relevant changes in size and resolution within the mesh are calculated automatically. In addition, two more regridding methods that are able to change the domain size could be designed. One strategy is keeping cells' size constant, then adding or removing cells on the boundary. Another strategy is keeping the number of cells unchanged, then increasing or decreasing cells' size to change the

domain size.

Bibliography

[1] Amsden, A., Ruppel, H., and Hirt, C.(1980) SALE: A simplified ALE computer program for fluid flow at all speeds. *Los Alamos Scientific Lab.*, Los Alamos, New Mexico: LANL, Rep. LA-8095.

[2] Melosh, H. J., Ryan, E. V., and Asphaug, E. (1992). Dynamic Fragmentation in Impacts' Hydrocode Simulation of Laboratory Impact. *Journal of Geophysical Research*, vol. 97, pp. 14735-14759.

[3] Elbeshausen, D., Wünnemann, K., and Collins, G.(2009). Scaling of oblique impacts in frictional targets: Implications for crater size and formation mechanisms. *Icarus*, 204, 716-731.

[4] Bjork, R.J. (1961) Analysis of the formation of Meteor Crater, Arizona. A preliminary report. *Journal of Geophysical Research*, 66, 3379 – 3387.

[5] Pierazzo, E. and Melosh, H.J. (2000). Understanding oblique impacts from experiments, observations, and modeling. *Annual Reviews in Earth and Planetary Science*, 28, 141 – 167.

[6] Collins, G. S., Wuennemann, K., Artemieva, N., and Pierazzo, E. (2012). Numerical modelling of impact processes. *Impact Cratering: Processes and Products*, pp. 254 – 270. Wiley-Blackwell.

[7] Ivanov, B. Deniem, D., and Neukum, G.(1997). . Implementation of dynamic strength models into 2D hydrocodes: Applications for atmospheric breakup and impact cratering. *International Journal of Impact Engineering*, vol. 20, pp. 411-430.

[8] Collins, G.S., Melosh, H. J., and Ivanov, B. A. (2004). Modeling damage and deformation in impact simulations. *Meteoritics and Planetary Science*, vol. 39, pp.217 – 231.

[9] Wünnemann, K., Collins, G. S., and Melosh, H. (2006). A strain-based porosity model for use in hydrocode simulations of impacts and implications for transient crater growth in porous targets. *Icarus*, vol. 180, pp.514 – 527.

[10] Collins, G. S., Melosh, H. J., and Wünnemann, K. (2011). Improvements to the epsilon-alpha compaction model for simulating impacts into high-porosity solar system objects. *International Journal of Impact Engineering*, 38(6):434 – 439.

[11] Collins, G. S. (2014). Numerical simulations of impact crater formation with dilatancy. *Journal of Geophysical Research - Planets*, vol.119, pp. 2600 – 2619.

[12] Raducan, S. D., Davision, T. M., and Collins, G. S.(2020). The effects of asteroid layering on ejecta mass-velocity distribution and implications for impact momentum transfer. *Planet. Space Sci*., 180, 104756.

[13] Raducan, S. D., Davision, T. M., Luther, R., and Collins, G. S.(2019). The role of

asteroid strength, porosity and internal friction in impact momentum transfer. *Icarus*, 329, pp. 282-295.

[14] Malvern, L.E. (1969) An Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Englewood Cliffs, NJ.

[15] Collins, G. S., Elbeshausen, D., Wünnemann, K., Davison, T. M., Ivanov, B., and Melosh, H. J.(2022). iSALE: A multi-material, multi-rheology shock physics code for simulating impact phenomena in two and three dimensions. iSALE Developer version.

[16] Pierazzo, E., Artemieva, N. et al. (2008). Validation of numerical codes for impact and explosion cratering: Impacts on strengthless and metal targets. *Meteoritics & Planetary Science*, vol. 43, pp. 1917-1938.

[17] Pierazzo, E., Vickery, A.M. and Melosh, H.J. (1997) A reevaluation of impact melt production. *Icarus*, 127, pp.408 – 423.

[18] Wünnemann, K., Collins, G.S. and Osinski, G.R. (2008) Numerical modelling of impact melt production in porous rocks. *Earth and Planetary Science Letters*, 269 (3 – 4), pp.530 – 539.

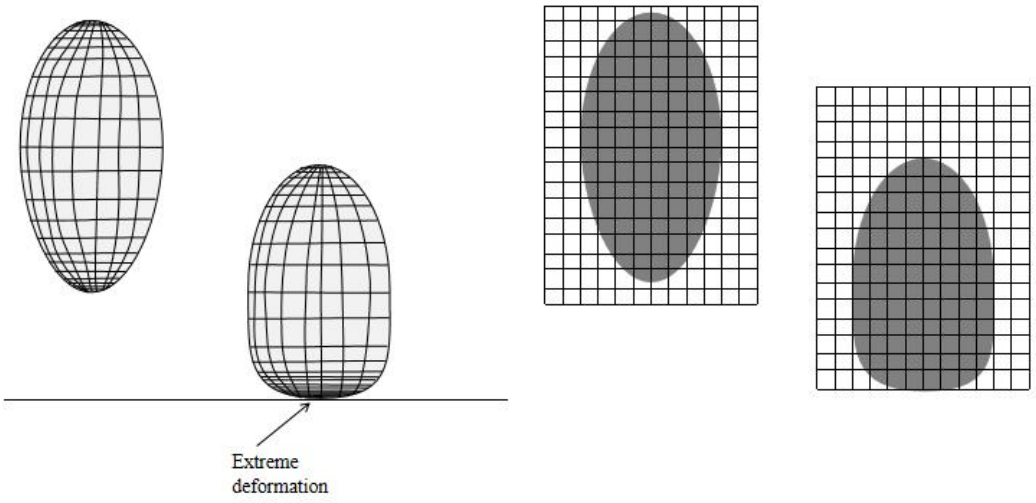
[19] Collins, G. S., Elbeshausen, D., Wünnemann, K., Davison, T. M., Ivanov, B., and Melosh, H. J.(2016). iSALE-Dellen manual: A multi-material, multi-rheology shock physics code for simulating impact phenomena in two and three dimensions. dx.doi.org/10.6084/m9.figshare.3473690.

[20] MPI Forum (2012) MPI: a message-passing interface standard. version 3.0.

[21] JupyerLab. https://jupyter.org/try-jupyter/lab/

Appendix

A. Lagrangian(left) and Eulerian(right) descriptions of reference frame.



B.

Plots and results from python script

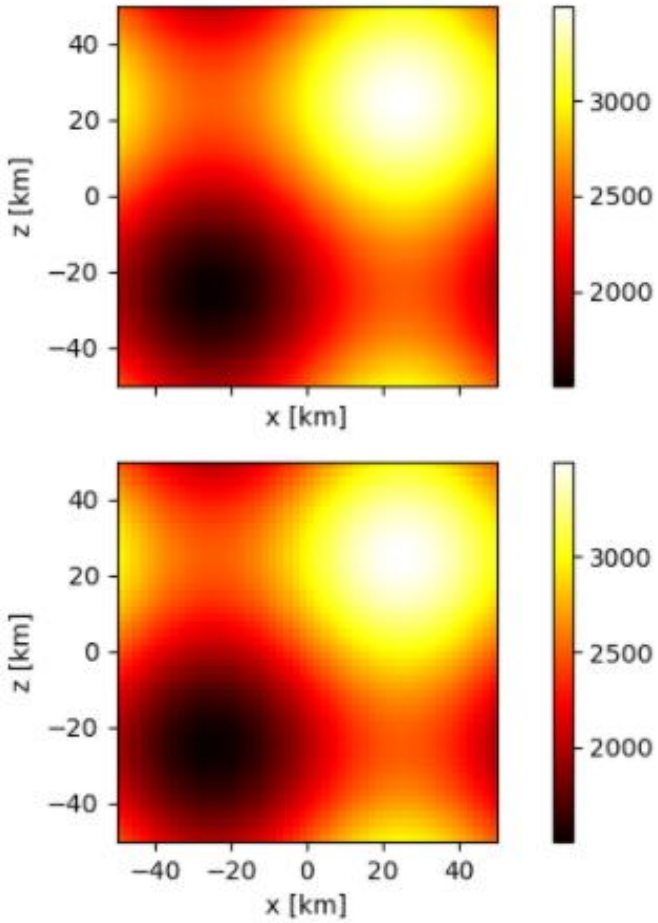


Figure: Density fields before and after regridding.

Table: Total mass and momentum.

	Total mass	Total momentum
Before coarsening	49539019.0	x: 2.45229617e+09 y: 2.45206813e+09 z: 2.45287527e+09
After coarsening	49539019.0	x: 2.45287527e+09 y: 2.45226984e+09 z: 2.45289912e+09