High Performance Computing Projects
Karthik Reddy Lyathakula

Contents

1. Introduction

In this project, three problems are solved using parallel programming features of MPI.

1. Two-Dimensional groundwater
2. Two-dimensional lamb wave propagation
3. Three-dimensional lab wave propagation

FORTRAN is used for all the problems.

2. Two-dimensional groundwater flow

In this problem, the two-dimensional groundwater flow equation as shown below (Darcy Law) is solved using the explicit finite-differencing method on a domain of 1000 m x 500 m.

$$S_s \frac{\partial h}{\partial t} = \nabla \cdot (kh\nabla h) + Q$$

where $h(x,y,t)$ is the hydraulic head(m), $k(x, y) = ax + by + cxy + d$ is the hydraulic conductivity, Ss is the storage or specific storage, t is the time in days, $Q(x,y)$ is the specific fluxes at the source and sinks(m/d). The values of a, b, c, d is calculated from the corner values $k(0,0)=2$, $k(1000,0)=1$, $k(1000,500)=2$, $k(0,500)=1$ and found out to be a=-1e-3, b=-0.2e-2, c=0.4e-5 and d=2.0. The contour plot of the hydraulic conductivity is shown in Fig.1

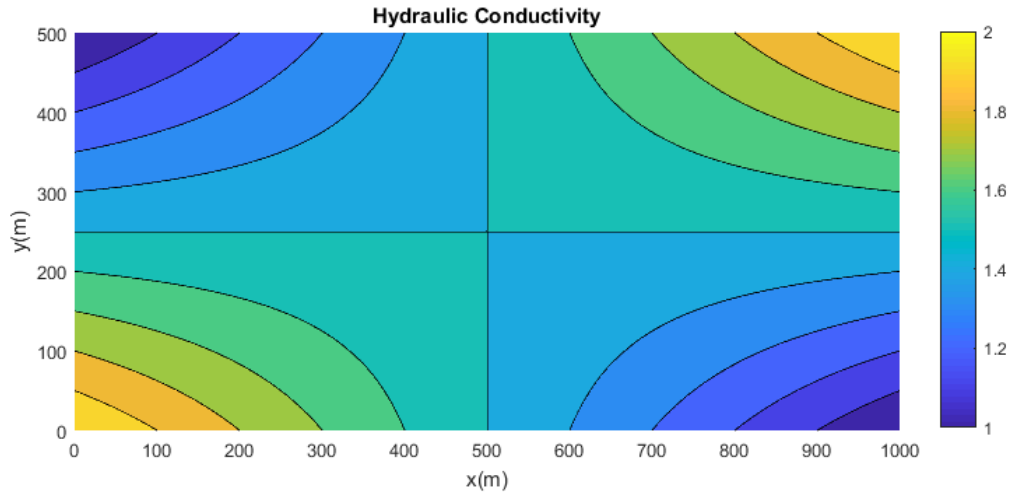


Fig.1 Hydraulic head contours in the computational domain

The boundary conditions are specified heads of 10 m and 5m at the left and right boundaries respectively and no flow at the top and bottom boundaries. The source is a uniform time-dependent recharge over the entire aquifer given by $Q = Q_0(1+\sin(\pi t/300))$ with $Q_0 = 0.001 m/d$.

2.1. Numerical Method

To solve the Darcy equation over the domain, the domain is divided into a rectangular grid of uniform spacing along x and y-direction and explicit Finite difference discretization is used to discretize the equations. The discretized equation is given by

$$S_s \frac{h_{ij}^{n+1} - h_{ij}^{n}}{\Delta t} = \frac{k_{i+1/2,j}}{2(\Delta x)^2}(h_{i+1,j}^2 - h_{ij}^2)^n - \frac{k_{i-1/2,j}}{2(\Delta x)^2}(h_{i,j}^2 - h_{i-1,j}^2)^n + \frac{k_{i,j+1/2}}{2(\Delta y)^2}(h_{i,j+1}^2 - h_{ij}^2)^n - \frac{k_{i,j-1/2}}{2(\Delta y)^2}(h_{i,j}^2 - h_{i,j-1}^2)^n + Q^n$$

where subscripts *i*, *j* are the indices of the grid points along x and y-directions respectively and superscript *n* indicates the time index.

2.2. Details of Parallel Code

To solve the problem in parallel over a different number of processors, the domain is decomposed into two dimensions and a single domain is assigned to each processor. Each processor solves the head values on the grid nodes of their respective domain and after each time step, the boundary values of the domains are exchanged with the boundary processors. For this purpose, ghost cells are created on all the boundaries of the domain. The code is written in such a way that the domain is decomposed automatically into two dimensions by factorizing the input number of processors. If the input number of processors is prime, the code will divide the domain only along the x-direction. The code for the processor domain decomposition is shown below

```
! 2D domain decomposition of processors

        nprocx=nprocs; nprocy=1;
        diff=abs(nprocx-nprocy)
        do iproc=1,nprocs/2
        if(mod(nprocs,iproc)==0) then
                if(abs(nprocs/iproc-iproc)<diff) then
                    nprocx=iproc; nprocy=nprocs/iproc
                    diff=abs(nprocs/iproc-iproc)
                endif
        endif
        enddo

        if(nprocx<nprocy) then
        diff=nprocx; nprocx=nprocy; nprocy=diff
        endif

        write(*,*) "Number of Processors in x-direction:",nprocx
        write(*,*) "Number of Processors in y-direction:",nprocy
```

A FORTRAN code is developed to solve the groundwater flow problem. The subroutine "*singleproc*" is used to solve the sequential part. Compiler optimization "-OFAST" is used to compile the code.

The grid spacing along x and y-directions (nx-1, ny-1), starting time(t0), time step(dt) and final time(tf) is hardcoded into the program as shown below.

```
a=-1e-3; b=-0.2e-2; c=0.4e-5; d=2.d0
Lx=1000; Ly=500; Ss=0.1; Q0=0.001
nx=1001; ny=501
t0=0; dt=1e-3; tf=1000
```

This program is tested on 1001x501, 501x251, 251x126, and 126x63 grid points with the same time step dt=1e-3 days and up to 1000 days on 1 to 32 processors. At the end of the simulation, the solution at the final time, of all the processors, is written to a file on the hard disk by only processor 0. These results are imported into MATLAB using the program: *validation.m* and plots are generated. Both FORTRAN and MATLAB programs are attached to this document.

2.3 Domain Decomposition

The domain decomposition part of the code is tested successfully on a sample geometry. Fig.2 shows the 2d decomposition of the domain and the corresponding processor of each domain is indicated by the label. It can be observed from Fig.2c that the domain is decomposed in one dimension when the input number of processors is prime.

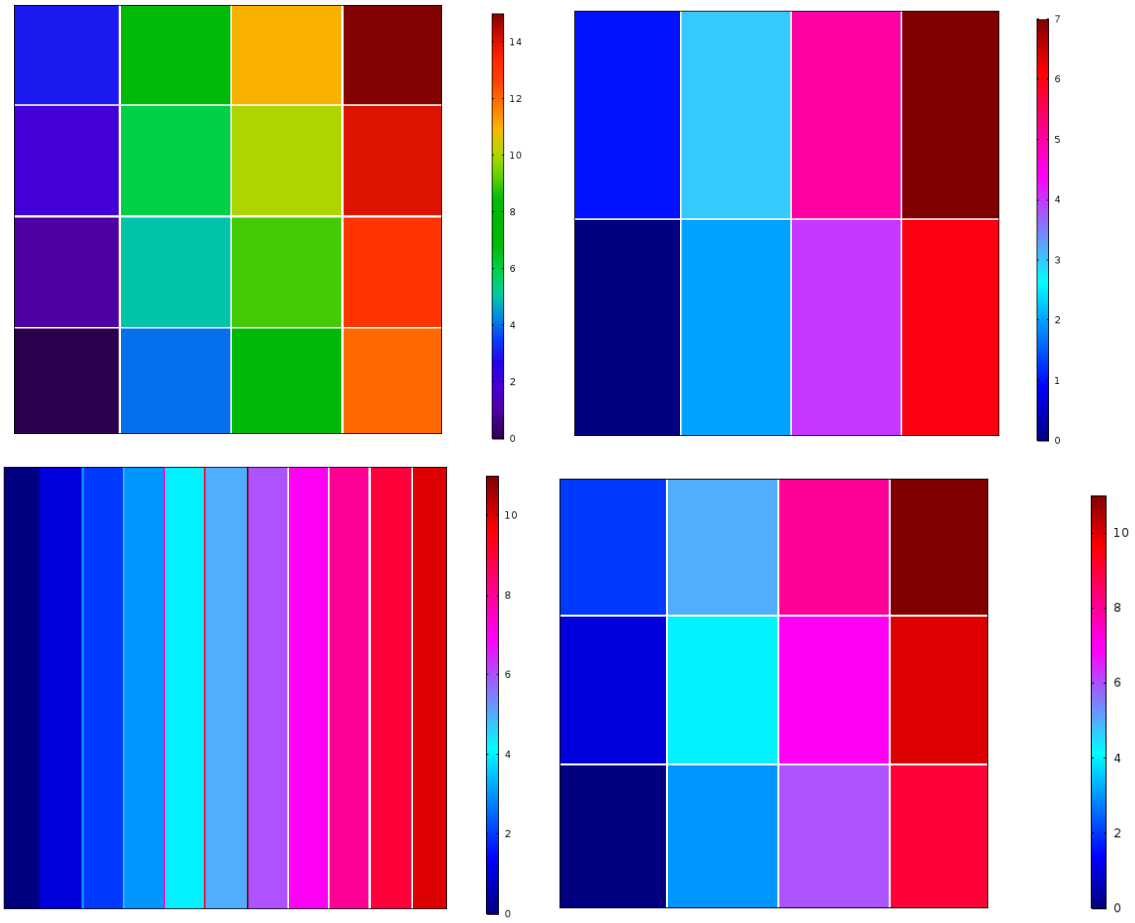


Fig 2. Label of domain decomposition for (a) 16 (b) 8 (c) 11 and (d) 12 number of processors

2.4. Groundwater flow solution

The solution returned by the FORTRAN code with 32 processors and on the grid of 1001x501 is plotted in Fig 3. Fig 3a shows the contours of the hydraulic head on the domain, Fig 3b shows the

solution of the hydraulic head along x-direction on plane y=250 m, and Fig. 3c shows the solution of the hydraulic head along y-direction on plane x=500 m. It can be observed from the solution that the hydraulic head at the center of the domain is higher compared to the values at the left and right boundaries. This is because the hydraulic conductivity values at the center are less compared to the values at the boundaries and the head developed due to the source, *Q* is accumulated at the center. Fig 3b supports this argument where the head values are high at the center compared to the boundary values. The variation of the hydraulic head along the y-direction is almost constant as no-flow boundary conditions are applied on top and bottom boundaries

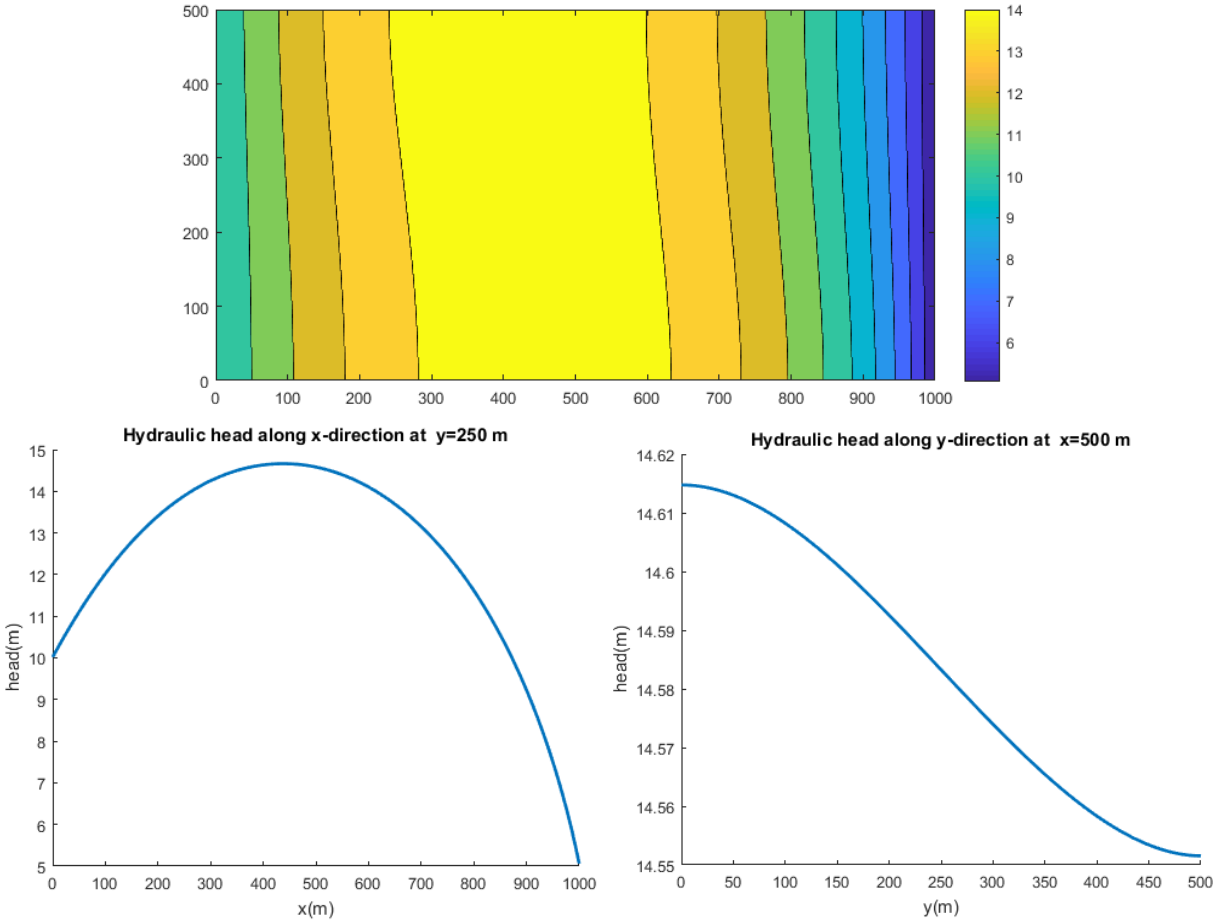


Fig 3. The solution of groundwater problem returned by 16 number of processors and with a grid size of 1001x501 plotted as (a) contours of a hydraulic head on the domain, (b) head values along the x-direction, and (c) y-direction on the corresponding mid-planes

2.5. Performance of groundwater flow program

The ground flow problem is tested on Henry and Bezier machines to find the performance of the program. This program is computed using 1001x501, 501x251, 251x126, and 126x63 number of grid points with same time step dt=1e-3 days and up to 1000 days on 1 to 32 processors. Table 1, Fig.4-5 shows the results of the program on the henry machine and table 2, Fig.6-7 are the results on the Bezier machine. The performance of the program is analyzed using the number of floating-

point operations per second (MFLOPS), speedup, and efficiency plots. Speedup S_p and efficiency E is defined by the equation

$$S_p = \frac{T_s}{T_p}, \quad E = \frac{T_s}{n_p T_p}$$

where T_s is the sequential time, T_p is the parallel programming time, and n_p is the number of processors. Fig.4a shows that MFLOP performance increases for all the grid sizes, except for 126x63, up to 16 processors than decreases for 32 processors. This is also reflected in the speedup plot shown in Fig.5a. Fig.5b shows the variation of efficiency with the number of processors and it is observed that efficiency decreases with the number of processors and efficiency is best for 1001x501 grid elements. Comparing the performance of different grids, it is observed that MFLOP performance increases with an increase in grid elements for the same number of processors. The reason is that the computational time is more compared to the communication time as grid size increases and this serves the purpose of parallelizing the code. The reason for the decrease in performance at 32 processors is maybe because the communication time is very high due to the architecture of the Henry machine. However, the performance using the Bezier machine did not show this discrepancy.

Np	1001x501		751x376		501x251		256x126		126x63	
	MFLOPS	time	MFLOPS	time	MFLOPS	time	MFLOPS	time	MFLOPS	time
1	4677	2359	4790	1297	4951	570	4507	154	4600	38
2	8502	1297	8514	730	8337	332	7673	90	7616	25
3	12597	875	12173	510	12246	226	11223	62	10452	18
4	15655	704	15692	396	14495	191	13949	50	9920	19
8	30393	363	30344	204	27467	100	24997	28	14658	13
16	55406	199	43629	142	42834	65	28990	24	11708	16
32	47378	228	43353	144	34340	81	17726	40	6164	28

Table 1. Parallel performance results of ground water flow program on Henry

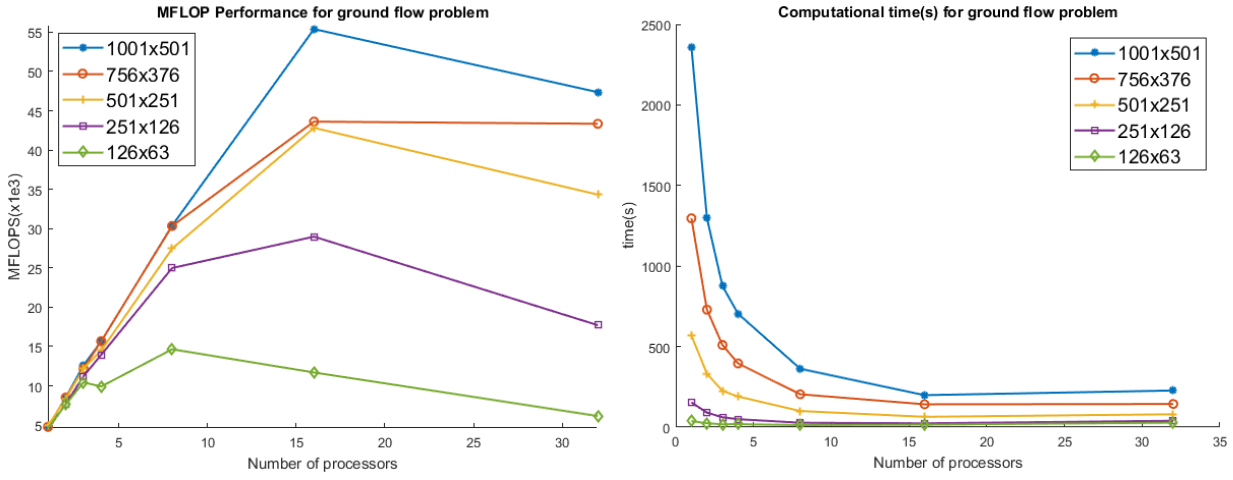


Fig 4. (a) MFLOP performance and (b) computational time vs number of processors on Henry machine for ground water flow problem

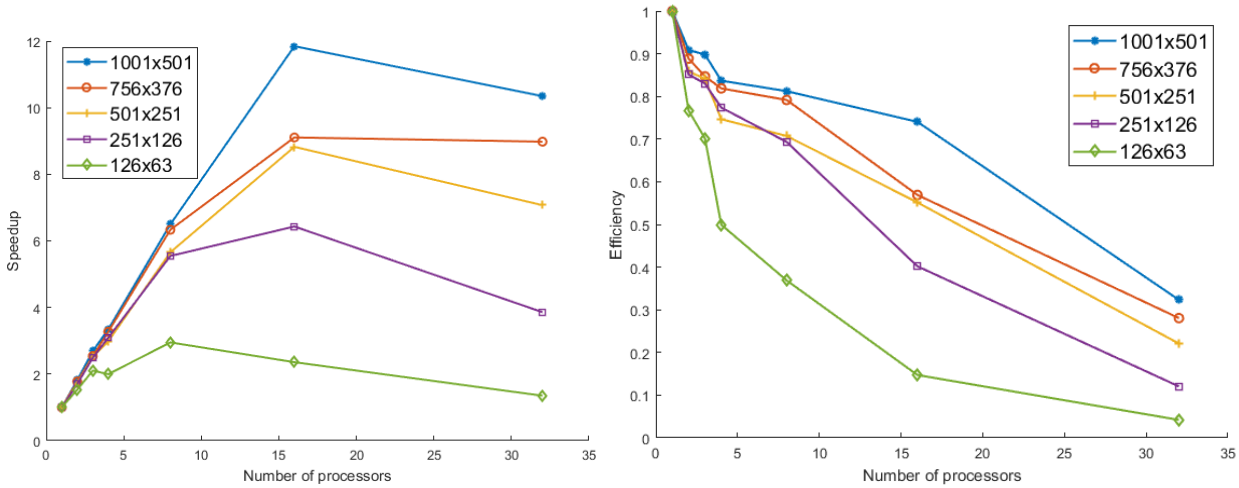


Fig 5. (a) Speedup and (b) Efficiency vs the number of processors on Henry machine for the groundwater flow problem

Table 2, Fig.6, and Fig.7 show the parallel programming performance results on the Bezier machine. It is observed from the results that MFLOP performance increases with the number of processors for larger grid size. For lower grid size, it increases linearly up to 8 number of processors and the MFLOP performance curve becomes almost flat for 251x126 grid but decreases for 126x63 grid

np	1001x501		751x376		501x251		256x126		126x63	
	MFLOPS	time	MFLOPS	time	MFLOPS	time	MFLOPS	time	MFLOPS	time
1	7251	1522	7821	795	8270	335	7959	87	7620	22.9
2	14245	774	14243	436	13329	207	12176	57	13203	14.8
4	24314	453	25329	245	23496	117	21094	33	19436	8.2
8	41790	264	44419	140	39048	70	40415	17	28612	7.8
16	86151	128	83789	74	73304	37	61544	11	31667	6.6
32	140754	78	140984	44	115355	23	65320	10	26529	9.6

Table 2. Parallel performance results of groundwater flow program on Bezier

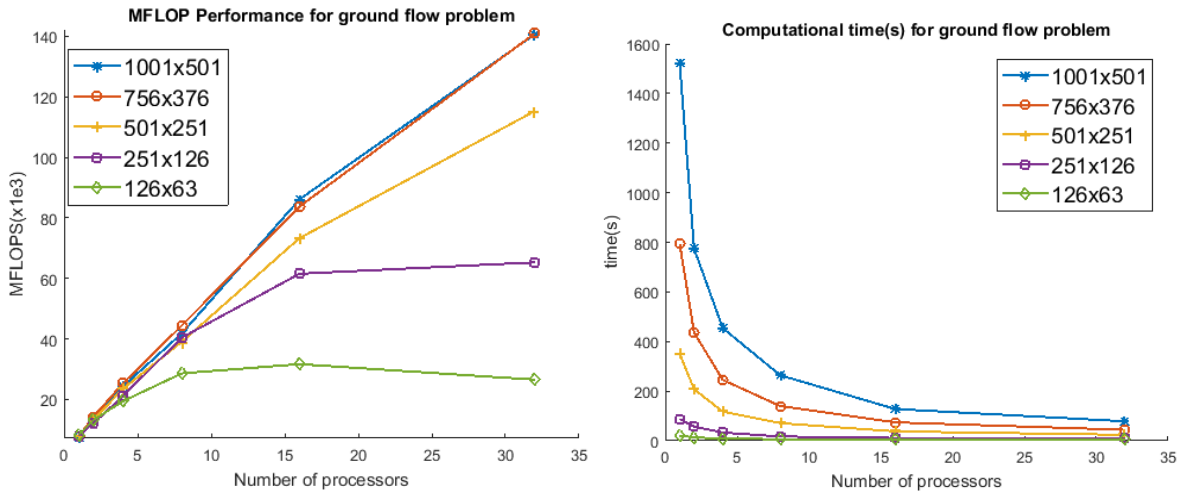


Fig 6. (a) MFLOP performance and (b) computational time vs number of processors on Bezier machine for ground water flow problem

elements at 32 number of processors. The reason for this is due to larger commination time than computational time for lower number of mesh elements. This is also evident from speedup plot shown in Fig. 7a. In summary, the performance of the program in Bezier machine is better than in henry and scalability is good in Bezier.

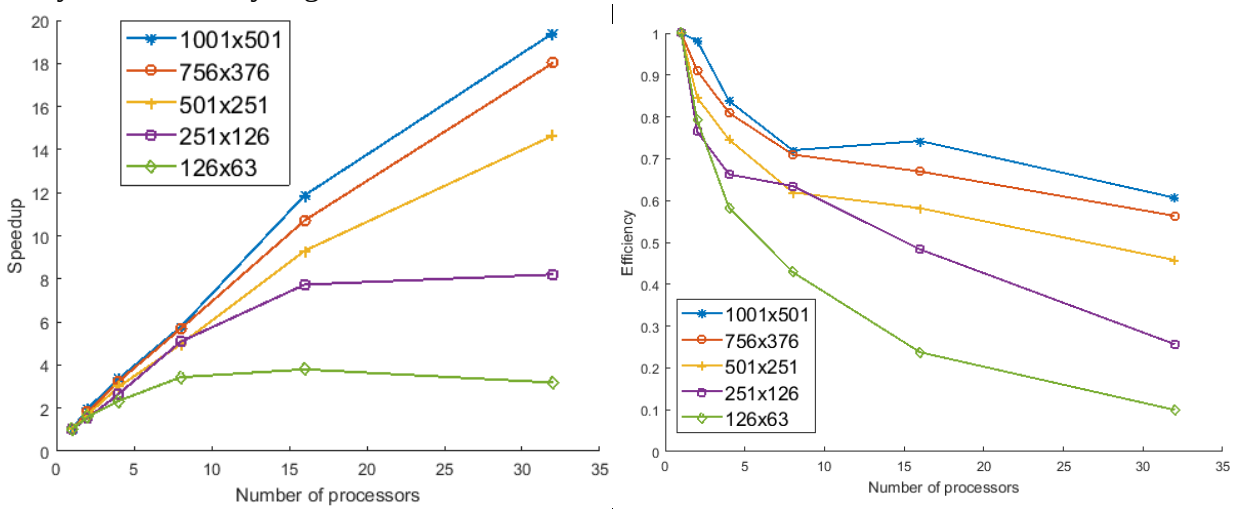


Fig 7. (a) Speedup and (b) Efficiency vs the number of processors on Bezier machine for ground water flow problem

3. Two-dimensional lamb wave propagation

Lamb waves are the subset of guided waves [1] that propagate in plates and shell structures. Lamb waves are multi-modal and are separated into symmetric and antisymmetric modes and the speed of these modes is calculated by Rayleigh-Lamb equations[1]. In this problem, the goal is to simulate the lamb wave propagation in 3d thin plates to evaluate different damage detection algorithms. In this project, only parallel programming performance is reported but not the damage detection techniques. However, the future goal is to use the 3d wave propagation parallel program to test different damage imaging techniques from the full wavefield measurement of thin plates with different types of grooves.

Lamb waves are generated in thin plates only when the excitation is at a very high frequency (kHz) and often they exist in pairs as symmetric and unsymmetrical wave modes. Simulating lamb waves is computationally intensive as small time steps and fine mesh resolution are required to solve the elastic-wave equation in large dimension plates. In this project, the code for simulating lamb wave propagation is entirely developed over this semester systematically starting from two dimensions to three dimensions.

Lamb wave propagation is simulated by solving the linear elastic equation of motion

$$\frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{f}$$

where $\boldsymbol{u}$ is the displacement, $\boldsymbol{\sigma}$ is the stress tensor, and $\boldsymbol{f}$ is the volumetric force. The stress tensor is related to the displacement by stress-strain relation and these relations for a two-dimensional problem depends on the length of the structure in the third dimension. Lamb waves are generated in thin plates where one of the dimensions is very small compared to the other dimensions. To simulate lamb waves in two dimensions, the third direction dimension (z-direction) must be one of the largest dimension and hence plane strain assumption need to be used. The stress-strain relation for plane strain condition is given by

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} = \sigma_{yx} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & 0.5-\nu \end{bmatrix} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix}$$

where

$$\varepsilon_x = \frac{\partial u}{\partial x},\ \varepsilon_y = \frac{\partial v}{\partial y},\ \gamma_{xy} = \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)$$

here $\boldsymbol{u} = [u, v]$ is the displacement field, E is the young's modulus and ν is the poisons ratio.

3.1. Numerical method for lamb wave propagation problem

To simulate the lamb wave propagation problem, the linear elastic equations are solved using finite elements method. The theory of FEM method is not explained in this report as it is well available in the literature [2] and the main goal of this project is to solve large lamb wave propagation problems by implementing the parallel programming techniques. Hence, only important details of FEM method used for this project is reported and the rest of the report is focused on performance of parallel programming techniques.

Linear quad elements are used to discretize the space and explicit time stepping method is used to discretize the time derivatives. Lumped mass method is used so that the mass matrix will be a diagonal matrix. This is achieved by evaluating the numerical integration at mesh node points for assembling the mass matrix and gauss quadrature points for assembling the stiffness matrix. BLAS package is used to calculate the matrix-matrix, matrix-vector vector multiplication while assembling the system matrices.

3.2. Details of parallel code for lamb wave propagation

The code for solving this problem is developed systematically and arranged into folders: 1) main 2) twod and 3) threed. The subroutines are arranged appropriately as indicated by the folder name. A single makefile is developed to link all the subroutines in different folders and generate an executable file, *project* as shown in the below screenshot

```
FX :=mpif90
FXFLAGS := -Ofast

BLASLOC := -L/usr/local/intel/mkl/lib/intel64 -lmkl_core -lmkl_intel_lp64 -lmkl_sequential

FILESMAIN := variables.o main.o readinput.o checkinput.o phymanager.o intsort.o findpos.o intrmvdup.o dotproduct.o

ADD1D := ./../oned
FILES1D := $(ADD1D)/waveeqn1d_fd_expl.o

ADD2D := ./../twod
FILES2D := $(ADD2D)/readgrid2d.o $(ADD2D)/planestrain_fem_expl.o $(ADD2D)/qudasmbly.o $(ADD2D)/pspqudcnt.o $(ADD2D

ADD3D := ./../threed
FILES3D := $(ADD3D)/readgrid3d.o $(ADD3D)/triarea3d.o $(ADD3D)/psphexcnt.o $(ADD3D)/psphexll.o $(ADD3D)/invmat3.o

TOTADD := -I $(ADD1D) $(ADD2D) $(ADD3D)

MODS :=variables.mod

OBJECTS := $(FILESMAIN) $(FILES1D) $(FILES2D) $(FILES3D)

all: project
project: $(OBJECTS)
     $(FX) $(FXFLAGS) $(OBJECTS) -o project $(BLASLOC)
$(FILESMAIN): %.o: %.f
        $(FX) $(FXFLAGS) -c $< -o $@ $(TOTADD)
$(FILES1D): %.o: %.f
        $(FX) $(FXFLAGS) -c $< -o $@ -I.
$(FILES2D): %.o: %.f
        $(FX) $(FXFLAGS) -c $< -o $@ -I.
$(FILES3D): %.o: %.f
        $(FX) $(FXFLAGS) -c $< -o $@ -I.
clean:
    rm -rf $(OBJECTS) project *.mod
cleano:
    rm -rf $(OBJECTS)
```

The program starts with *main.f*, which reads the input from an input file and calls the subroutine *phymanager.f* , which calls appropriate subroutine based on the input provided in the input file. The input file is properly structured as shown in the below screenshot (A small section of input file) and the file need to be changed accordingly to run required simulation

```
#       Option Value= #2 - Mesh file name
#       Option Value= #3 - left coord for oned
#       Option Value= #4 - right coord for oned
#       Option Value= #5 - dx
#-------------------------------------------------------------------------------
start spacedim
1 3
2 test1dgrid.dat
3 0
4 1
5 0.005
#-------------------------------------------------------------------------------
#      Physics type and material properties
#       Physics type           # 1= Elastic Equation
#       Material Properties    # 2: Density
#                              # 3: Young's Modulus
#                              # 4: Poisson's ratio
#-------------------------------------------------------------------------------
start phy_type
1   1
2   2700
3   7e10
4   0.33
#-------------------------------------------------------------------------------
#       Discretization type and basis function
#       Discretization type  # 1= 1 Finite difference
#                                 2 Finite element
#       Discretization order # 2= 1 Linear
#                                 2 Quadratic
#       Basis function(FEM)  # 3= 1 Lagrangian
#                                 2 Taylor Basis(For DG Methods)
#                                 3 Legendre Basis(For DG Methods)
#-------------------------------------------------------------------------------
start disc
1   1
2   1
3   1
#-------------------------------------------------------------------------------
#       Boundary Conditions (Implemented only for 1D for now)
#       Left boundary       # 1= 1 Dirchilet
```

Mesh for this program is generated using an external program and the details of the mesh are stored in a file in unstructured format. The subroutines *readgrid2d.f* and *readgrid3d.f* reads the grid file

for two dimension and three dimensional problems respectively and the element, node and face details are stored in appropriate data structures. The mesh file also provides the node points where the solution need to be stored. The solution is stored only in these node points. This is done because the solution generated by the program for lamb wave propagation is very large and it is impractical to store the data at all the grid nodes throughout the time stepping process.

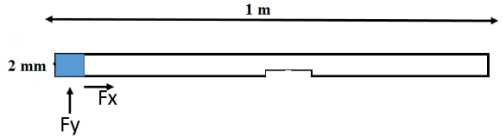
As this project uses FEM technique, the mass and stiffness matrix need to be assembled and it is assembled only before starting the time stepping and stored in compressed sparse row (CSR) storage form.

3.3. Domain Decomposition for the Parallel Program

To solve the 2d lamb waves, the domain is decomposed only in one dimension as the thickness of the plate is very small compared to the length. An algorithm is developed such that domain is automatically decomposed in one dimension based on the grid size. The ghost cells are generated only at the shared boundaries and they are not needed on the geometry boundary as the boundary conditions are added as a contribution or constraint in FEM technique.

3.4. Solution of Lamb Wave Propagation

The developed program for two dimensional lamb wave is tested on an aluminum plate of length 1 m and 2 mm thick with a slot of 1 cm long at the center of the plate as shown below. To show the details of the slot, the geometry is plotted with very low aspect ratio. The thin plate is meshed with a quad elements with 8 elements along the thickness and a maximum mesh size of 0.25 mm is used. This generated a mesh that makes the program to solve 72000 DOF.



An excitation of 5 cycle tone burst with a magnitude of 1e6 N/m^2 and central frequency of 400[kHz] is applied at the left end of the plate on 3 mm length. The plot of five cycle tone burst is shown in Fig. 7.

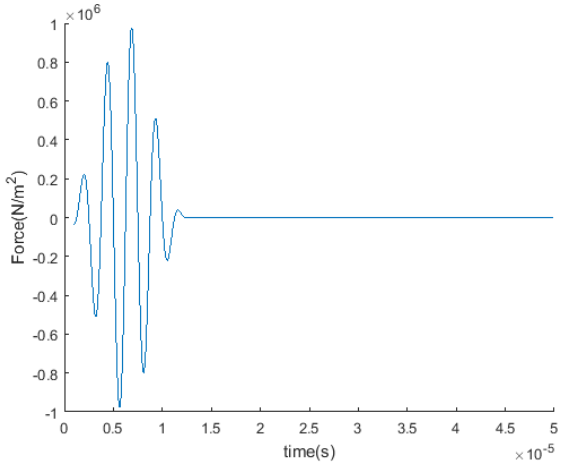


Fig 7. Five cycle tone burst

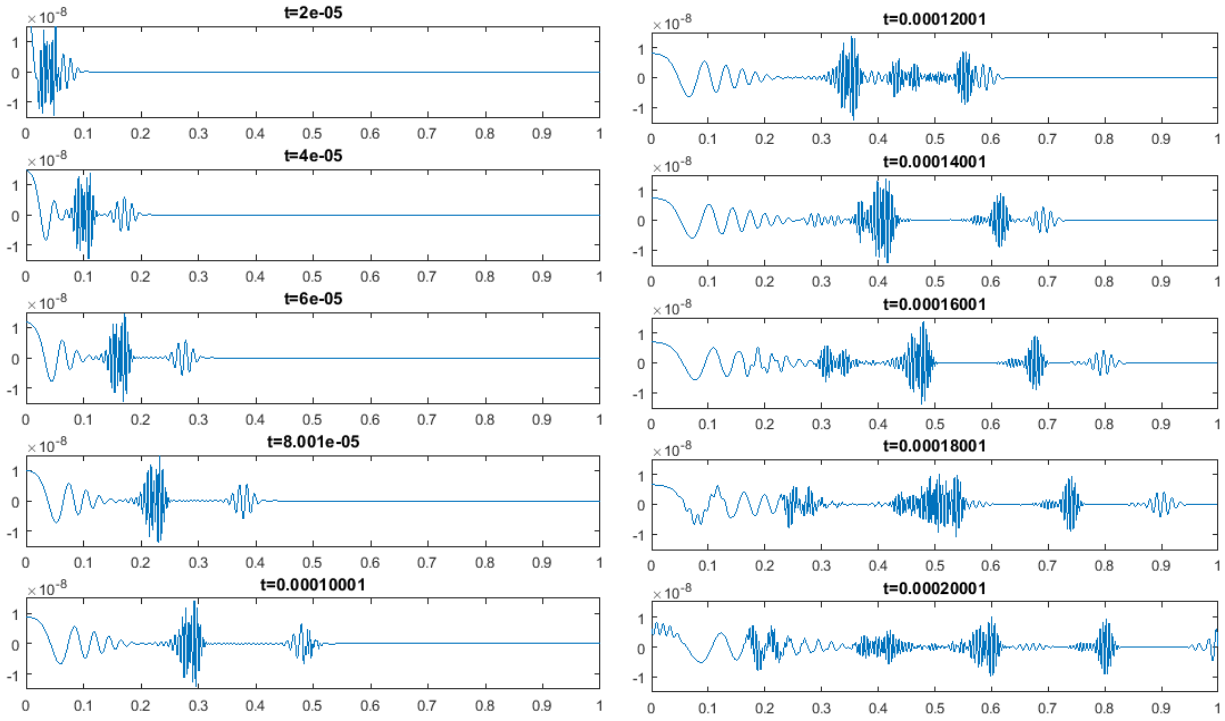


Fig. 8. Displacement along y-direction on the top surface of the plate shows (a) splitting of the tone burst into S0 and A0 mode and (b) the reflection of the wave from the slot at the midpoint.

The program is executed with time step of 1e-8 seconds up to 1e-3 seconds on Bezier machine from 1 to 32 processors. The 2d program is not tested for more than 32 processors but it will work if there are sufficient elements that can be divided between the processors. The displacement solution on the top surface of the plate along y-direction is saved after every 1e-6 seconds. Fig. 8 shows the response of the top surface due to the applied excitation with 32 processors. It can be observed that the wave splits into S0 and A0 modes, as expected, and S0 wave mode travels faster than A0 mode. The wave speeds of S0 and A0 mode matches with the literature. Further, some part of the wave reflects once it hits the damage and the reflected part further splits into S0 and A0 waves.

3.5. Parallel programming performance of 2d lamb wave propagation

Fig.9 shows the variation of MFLOP performance and computational time with number of processors on Bezier machine. It can be observed that the MFLOPS varies linearly and computational time decreases hyperbolically with number of processors. The speedup plot is similar to MFLOP performance. However, the efficiency decreases with number of processors and it is comparable with the efficiency of ground water flow problem on Bezier machine (Fig 7).

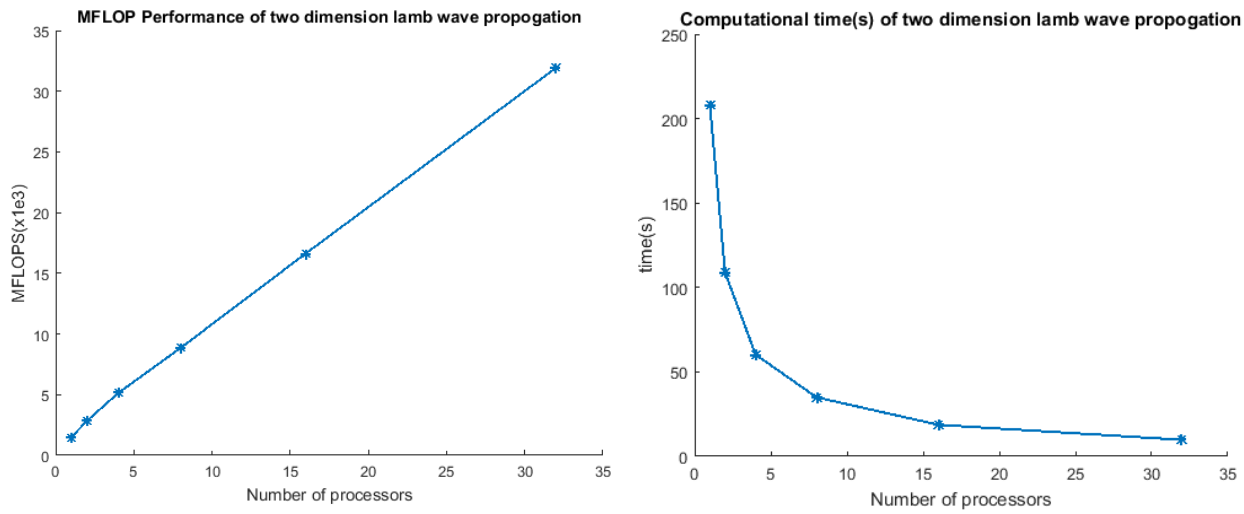


Fig. 9. Variation of (a) MFLOP performance and (b) computational time with number of processors of two dimensional lamb wave propagation on Bezier machine

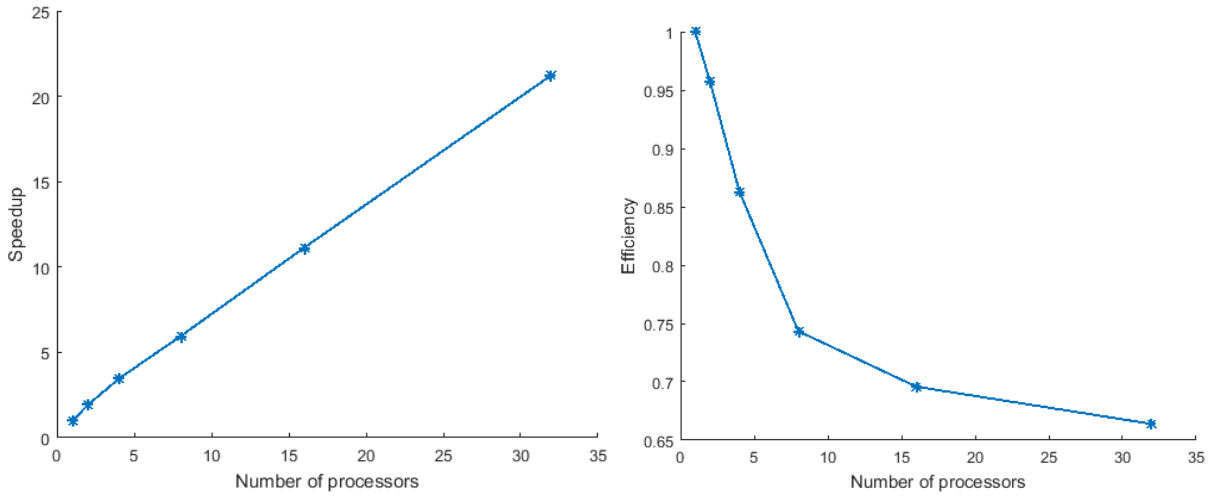


Fig. 10. Variation of (a) speedup and (b) efficiency with number of processors of two dimensional lamb wave propagation on Bezier machine.

4. Three-dimensional lamb wave propagation

Once the 2d lamb wave propagation is successfully tested, the program is extended to three dimensional space. To simulate the lamb wave propagation in three dimensional plates, the complete linear elastic equation need to be solved. The stress-strain equation for linear elastic equation is given by

$$\boldsymbol{\sigma} = \boldsymbol{D\varepsilon}$$

where

$$\boldsymbol{D} = \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5-\nu \end{bmatrix}$$

is the elasticity matrix and

$$\boldsymbol{\varepsilon} = \left[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial z}, 0.5\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right), 0.5\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right), 0.5\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right]$$

is the strain matrix and $\boldsymbol{u} = [u, v, w]$ is the displacement field.

4.1 Details of parallel code for 3d lamb wave propagation

The previous 2d code is extended by adding subroutines for simulating three dimensional problem. To solve the three dimensional wave propagation, the domain is decomposed in two dimensions similar to the groundwater flow problem. However, this problem is complicated because unstructured grid format is used and moreover the geometry may not be symmetrical because of slots in the geometry. So, an algorithm is developed to decompose any arbitrary plate geometry geometry in two dimensions, which includes finding the elements of each processor and bookkeeping the boundary elements that need to be exchanged with the adjacent processors.

This task is very complicated for FEM methods as each elemental stiffness matrix is connected to 27 nodes and 81 DOFs. Thus this problem involves lot of book keeping and a subroutine *divisionofgrid3d_2d_parallel.f* is developed for this purpose. The book keeping involves finding the shared boundary points, corner points in case of 2d decomposition and storing in different arrays.

Initially, only one processor is used to read the grid file, decompose the domain and distribute the corresponding domain elements to each processor. This used to take around 2 hours to divide a grid of size 0.2 million number of elements. To avoid this, the division of grid is parallelized by broadcasting the mesh data, read by processor 0, to all processor and each processor finding its own mesh elements. This has reduced the time to 2d decomposition of the grid to 15 minutes

Similar to ground water flow problem, the program is self-sufficient to decompose the domain in one dimensional if the number of processors are prime otherwise decomposes in two dimensions. The same algorithm (screenshot in ground water flow problem section) as that of ground water flow problem is employed in this code as well.

4.2 Simulation of Lamb wave propagation on a 3d plate

To test this three dimensional program, lamb wave propagation is simulated on 0.5 m x 0.5 m x 2 mm plate as shown in Fig 11. The geometry has a slot at the center with a thickness of 1 mm as shown in the figure and it has two horizontal lines which are created only for generating mesh elements. The geometry is meshed using hexahedral elements with a spatial resolution of 2[mm]. The complete mesh elements of the domain is 490,000 and number of nodes are 557,405. To

simulate the wave propagation on this geometry, the developed code had to solve around 2 million DOFs and have to store more than 10 million DOFS: 6 million DOUBLE PRECISION memory sizes for storing the values of DOFs for three physical time values (Because of second derivative in time) and more than 4 million DOUBLE PRECISION memory sizes for storing stiffness and mass matrices.

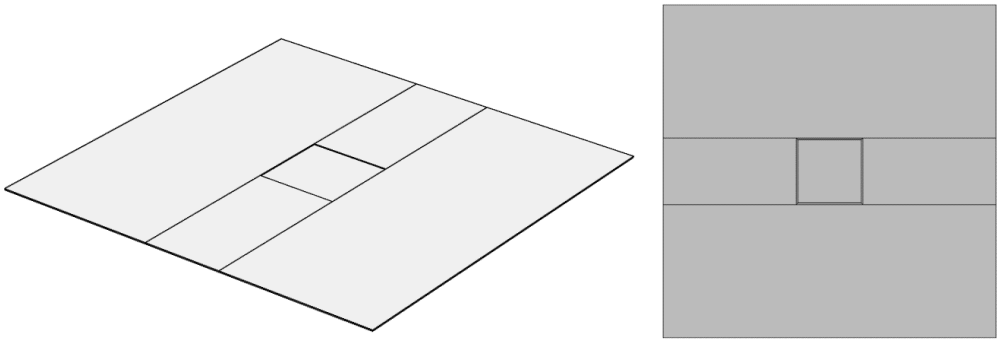
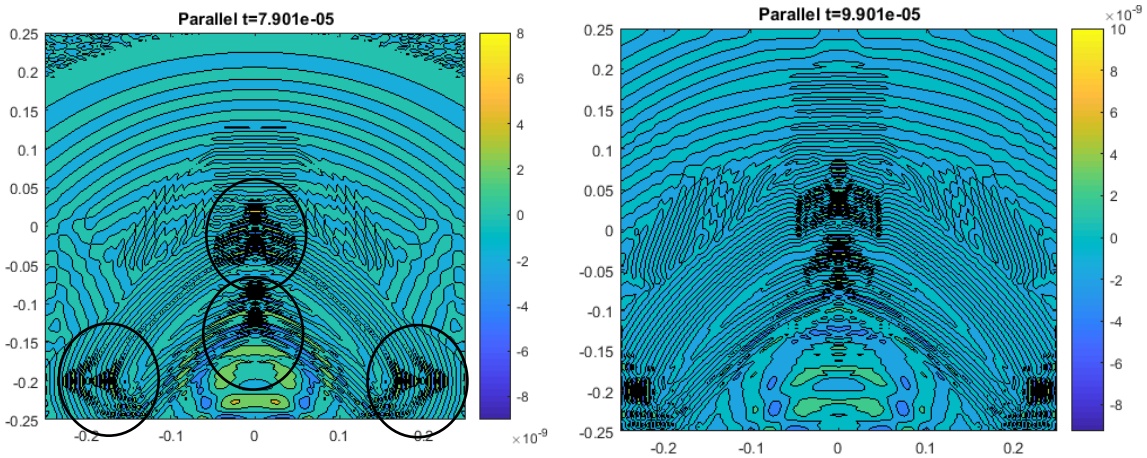


Fig 11. (a) Three dimensional geometry and (b) top view of the plate of dimensions 0.5 m x 0.5 m x 2 mm with a slot at the center.

The simulation is conducted using explicit time stepping with a step size of 1e-8 seconds up to 1e-3 seconds. The w-displacement solution on the opposite surface of the plane shown in Fig. 11b is stored in the output file after every 1e-6 seconds. The excitation to the plate is provided using a five cycle tone burst, similar to two dimensional, on a small surface of 1 cm^2 area at the bottom of the plate with a magnitude of 1e6 N/m^2 and frequency of 200[kHz].

Fig. 12 shows the w-displacement on the plane surface of the plate (bottom surface) plotted in MATLAB by reading the solution file. These plots are also often called full wave field solution. The solution file generated by the FORTRAN program is around 1 GB to store only w-displacement on 63001 mesh node points. It can be observed from the plots that the wave is propagating in x and y-direction and the waves in both direction have split into S0 and A0 as shown by two high intensity values in the contour plot at t=7.9e-5 seconds. From the remaining figures it can be observed that the wave further propagates and some waves are reflected from the location where there is change in thickness (stiffness) and some from the boundaries. The reflected wave as well spitted into A0 and S0 waves as shown in the results.



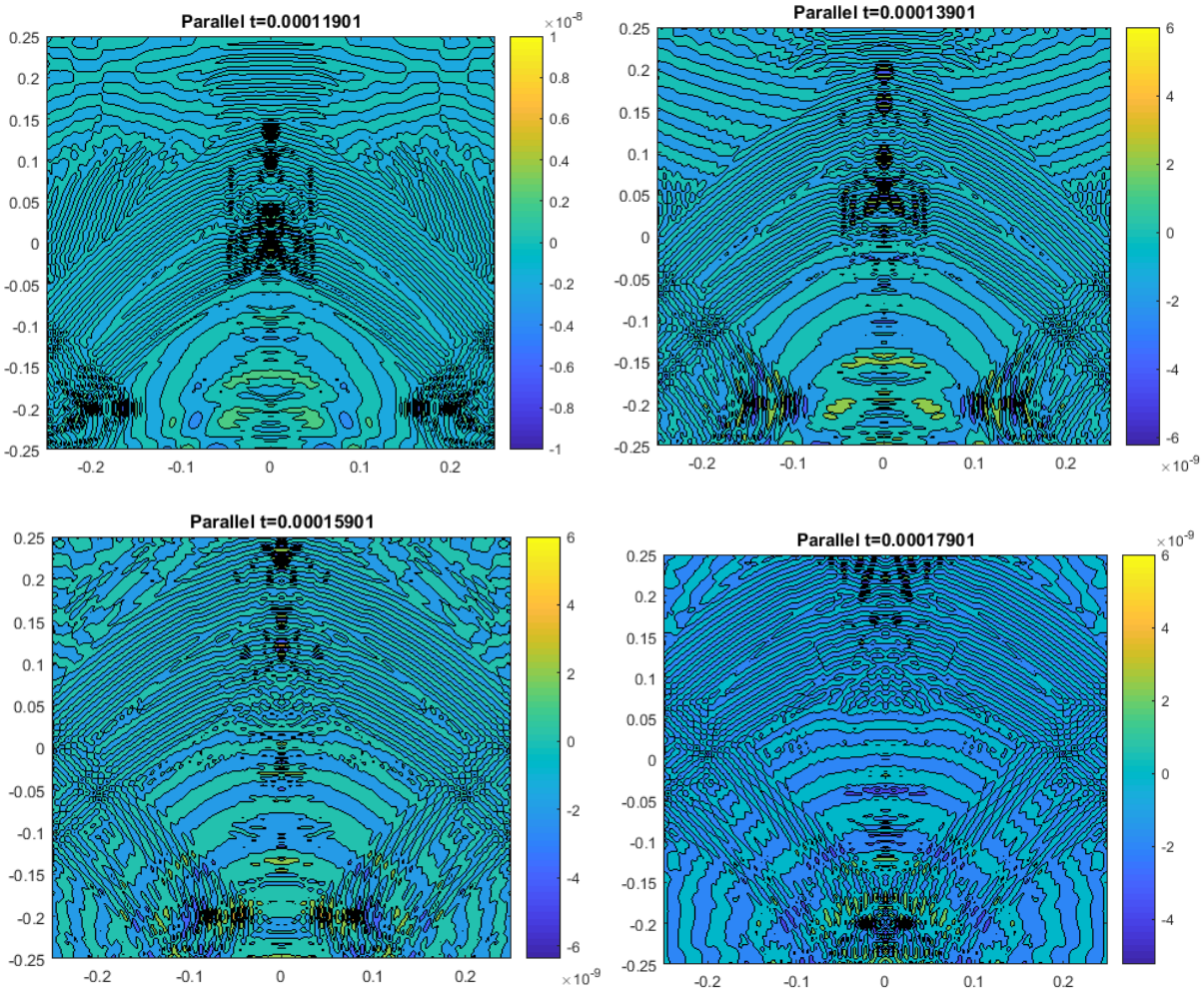


Fig. 12. Full wave field solution of lamb wave propogation on the surface of 3d plate at 200 kHz.

4.3 Parallel programming performance of 3d lamb wave propagation

To test parallel programming performance of the code, the 3d wave propagation problem is solved using processors 16, 32, 48, 64 and 128 on Bezier machine. The problem is not solved below 16 processors as the code is returning an error message. The reason for this behavior is due to the number of DOFs need to be solved by each processor increases with decreases in number of processors and each individual processor may not have sufficient memory to handle such high DOFs.

Fig 9 shows the variation to MFLOPS and computational time with number of processors. It is observed that MFLOP performance increases almost linearly and computational time decreases hyperbolically with number of processors. The plots are similar to the performance plot of ground water flow problem and 2d lamb wav propagation.

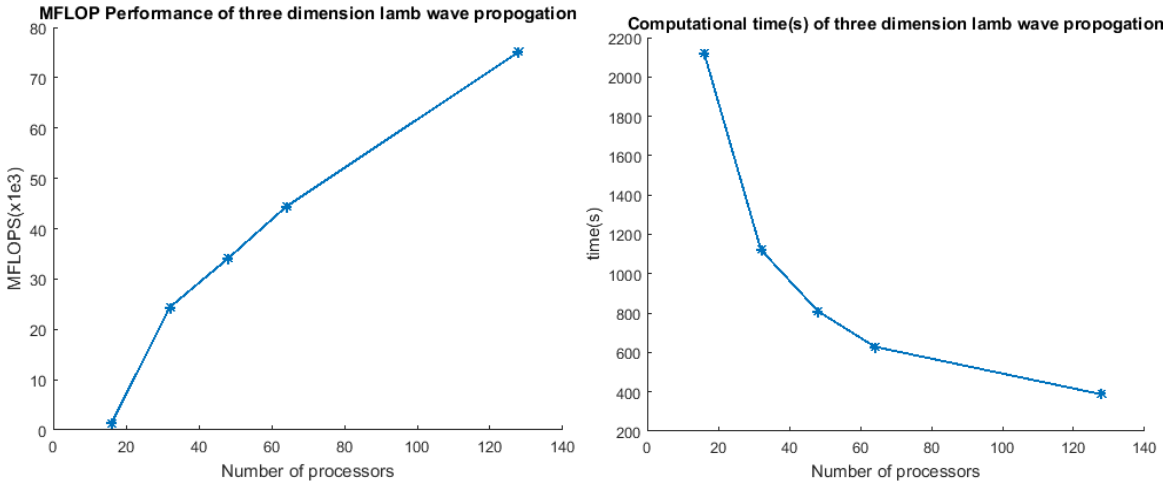


Fig. 13. Variation of (a) MFLOP performance and (b) computational time vs number of processors of three dimensional lamb wave propagation on Bezier machine

4.4 Simulation of Lamb wave propagation on refined 3d plate

For implementing the damage detection technique, it requires full wave field solution of lamb wave propagation with a spatial resolution of 1[mm]. So, simulation is conducted by refining the mesh and with exactly same conditions as above case. The refined mesh generated 1.9 million elements, 2.22 million nodes, which makes the problem to solve around 6.66 million DOFS. As the DOFs are very high, a single simulation is performed on 256 processors. This has taken 930 seconds and the MFLOP values for this simulation is 120,667. The full wave field solution of this simulation is compared with simulation on courser grid and the results have improved.

5. Future Work

The future work is to simulate different kind of geometries and test the damage imaging techniques.

6. References

1. https://en.wikipedia.org/wiki/Lamb_waves
2. https://en.wikipedia.org/wiki/Finite_element_method
3. Flynn, E.B., Chong, S.Y., Jarmer, G.J. and Lee, J.R., 2013. Structural imaging through local wavenumber estimation of guided waves. *Ndt & E International*, *59*, pp.1-10.