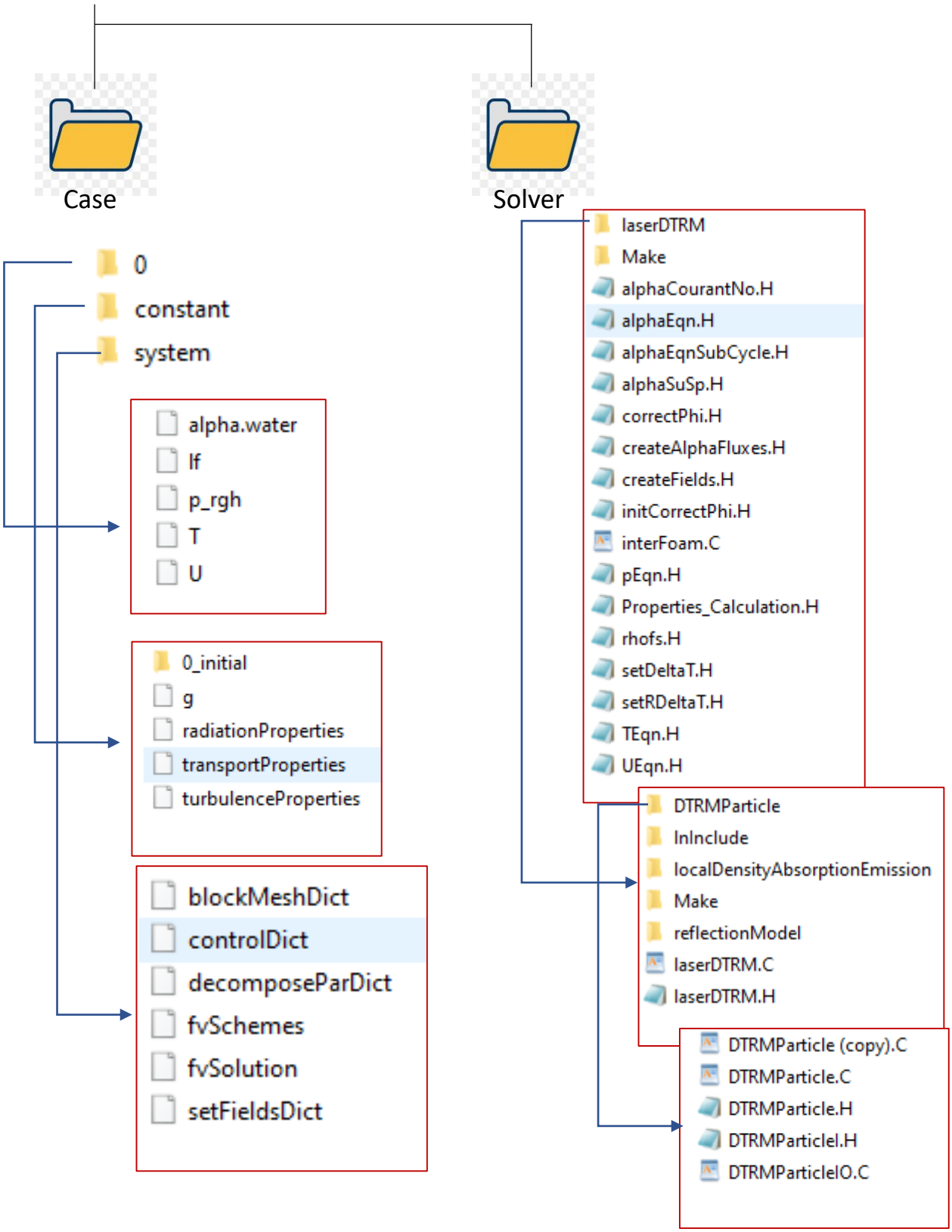
LPBF MODEL

Prerequisites

- OpenFoam V2006 installed with Swak4Foam Library
- Basic knowledge of OpenFoam
- Postprocessing knowledge with Paraview

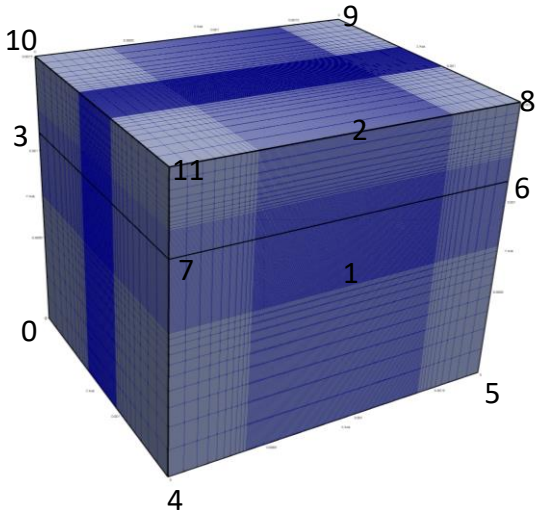
Model Structure



Case\systems\BlockMesh: To set up computational domain, cell arrangement and boundary types

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1e-6;

vertices
(
    (0 0 0)             // 0
    (2200 0 0)          // 1
    (2200 700 0)       // 2
    (0 700 0)          // 3
    (0 0 1350)          // 4
    (2200 0 1350)       // 5
    (2200 700 1350)    // 6
    (0 700 1350)       // 7
    (2200 1000 1350)    // 8
    (2200 1000 0)       // 9
    (0 1000 0)          //10
    (0 1000 1350)       //11
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (320 98 108)
    simpleGrading
    (
        (                                                                   // x-direction
            (500 10 0.2) // 0.45 mm y-dir, 10 cells, expansion = 0.2
            (1200 300 1) // 0.6  mm y-dir, 60 cells, expansion = 1
            (500 10 5) // 0.45 mm y-dir, 10 cells, expansion = 0.2
        )

        (                                                                   // y-direction
            (350 10 0.2) // 0.45 mm y-dir, 10 cells, expansion = 0.2
            (350 88 1) // 0.6  mm y-dir, 60 cells, expansion = 1
        )
        (                                                                   // z-direction
            (500 10 0.2) // 0.55 mm z-dir, 10 cells, expansion = 0.05
            (350 88 1) // 0.2  mm z-dir, 20 cells, expansion = 1
            (500 10 5) // 0.55 mm z-dir, 10 cells, expansion = 0.05
        )
    )
    hex (3 2 9 10 7 6 8 11) (320 35 108)
    simpleGrading
    (
        (                                                                   // x-direction
            (500 10 0.2) // 0.45 mm y-dir, 10 cells, expansion = 0.2
            (1200 300 1) // 0.6  mm y-dir, 60 cells, expansion = 1
            (500 10 5) // 0.45 mm y-dir, 10 cells, expansion = 0.2
        )

        (                                                                   // y-direction
            (100 25 0.2) // 0.45 mm y-dir, 10 cells, expansion = 0.2
            (200 10 5) // 0.6  mm y-dir, 60 cells, expansion = 1
        )
        (                                                                   // z-direction
            (500 10 0.2) // 0.55 mm z-dir, 10 cells, expansion = 0.05
            (350 88 1) // 0.2  mm z-dir, 20 cells, expansion = 1
            (500 10 5) // 0.55 mm z-dir, 10 cells, expansion = 0.05
        )
    )
);

edges
(
);

boundary
(
    top
    {
        type patch;
        faces
        (
            (8 9 10 11)
        );
    }
    leftair
    {
        type patch;
        faces
        (
            (10 3 7 11)

        );
    }
    leftwall
    {
        type wall;
        faces
        (

            (0 4 7 3)
        );
    }

    rightair
    {
        type patch;
        faces
```

See figure



Cells arrangement in Lower block

Cells arrangement in Upper block

Defining boundaries

Case\systems\BlockMesh: To set up computational domain, cell arrangement and boundary types

```
    rightair
    {
        type patch;
        faces
        (
            (8 6 2 9)
        );
    }

    rightwall
    {
        type wall;
        faces
        (

            (1 2 6 5)
        );
    }

    bottomWall
    {
        type wall;
        faces
        (

            (4 0 1 5)
        );
    }

    backwall1
    {
        type wall;
        faces
        (
            (0 1 2 3)
        );
    }

    backwall2
    {
        type patch;
        faces
        (
            (2 3 10 9)
        );
    }


    frontwall1
    {
        type wall;
        faces
        (
            (7 4 5 6)
        );
    }

    frontwall2
    {
        type patch;
        faces
        (
            (11 7 6 8)
        );
    }

);

mergePatchPairs
(
);

// ************************************************************************* //
```

Defining boundaries

Leave emplty

Case\systems\ControlDict: To set up time controls

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

libs (
      "libOpenFOAM.so"
      "libsimpleSwakFunctionObjects.so"
      "libswakFunctionObjects.so"
      "libgroovyBC.so"
     );

application     SS;  // solver name

startFrom       latestTime;  // simulation starts from the latest available time

startTime       0;           // Start of simulation

stopAt          endTime;

endTime         1.66e-3;     // End of simulation

deltaT          1e-8;        // time steps (simulation starts with very small time steps)

writeControl    adjustableRunTime; // time steps change based on courant number

writeInterval   2e-5;              // time intervals to save data

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.6;
maxAlphaCo      0.6;

maxDeltaT       0.1;
// ************************************************************************* //
```

Libraries from swak4foam (must be included)

// No change

// Courant numbers should be between 0 and 1 for stable run (optimum value 0.5 or lesser)

// max time step (time step wont go above this, another way to stabilize your simulation, in case Courant numbers are shooting high)

Case\systems\decomposeParDict: To divide domain into subdomains for parallel processing

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 16;  // Dividing whole computational domain into 16 parts

method          scotch;  // Dividing whole computational domain using scotch method (this
                         method tries to keep equal number of cells in each subdomain)

simpleCoeffs
{
    n               (4 2 2);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (1 1 1);
    delta           0.001;
    order           xyz;
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots           ( );


// ************************************************************************* //
```

// Other methods, not using

Case\systems\FvSchemes: Equations discretization methods

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss upwind;
    div(phi,alpha)  Gauss upwind;
    div(phirb,alpha) Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
//    div(rhoCpphi,T)  Gauss limitedLinear 1;
    div((interpolate((((alpha.water*rho)*((cps*(1-lf))+(cpl*lf)))+((alpha.air*rho)*cp2)))*phi),T) Gauss upwind;
    //div(rhoCpphi,T)  Gauss vanLeer;
//    div(rhoAlpha1Phi,lf)  Gauss vanLeer;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}


// ************************************************************************* //
```

Case\systems\FvSolution: Equations solving methods

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  dev                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    5;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-7;           //1e-5
        relTol          0;
    }
    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-08;         //1e-7
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;           //1e-6
        relTol          0;
    }

     T
    {

        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-12;
        relTol          0.05;
    }

    TFinal
    {
        $T;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor   no;
    nOuterCorrectors    1;
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    equations
    {
        ".*" 1;
    }
}


// ************************************************************************* //

```

Case\systems\SetFieldsDict: To define substrate and particles position

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
   boxToCell
    {
        box (0 0 -0.1) (2200e-6 700e-6 1350e-6);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }

  sphereToCell
  {
    centre (0.00054396 0.000719999 0.00052 );
    radius 2e-05 ;
  fieldValues
  (
    volScalarFieldValue alpha.water 1
  );
  }

```

Set VOF variable to 0 (Argon gas) everywhere in domain.

substrate is defined as a box, and value of VOF variable is set as 1 (material). This replaces the Argon gas with material in this region.

Particles are defined as spheres, and value of VOF variable is set as 1 (material). This replaces the Argon gas with material in this region.

Case\Constant\0_initial: A backup folder where the initial values of variables is stored.

- alpha.water // VOF variable (0 for Argon and 1 for material)
- lf // Variable for liquid fraction
- p_rgh // Variable indicating Pressure
- T // Variable indicating Temperature
- U // Variable indicating Velocity

Case\Constant\0_initial folder\alpha.water:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];    // units of variable


internalField   uniform 0;          // Initial value of the variable in each cell

boundaryField
{
    top
    {
        type            zeroGradient;
    }
    leftair
    {
        type            zeroGradient;
    }
    leftwall
    {
        type            zeroGradient;
    }
    rightair
    {
        type            zeroGradient;
    }
    rightwall
    {
        type            zeroGradient;
    }
    bottomWall
    {
        type            zeroGradient;
    }
    backwall1
    {
        type            zeroGradient;
    }
    backwall2
    {
        type            zeroGradient;
    }
    frontwall1
    {
        type            zeroGradient;
    }
    frontwall2
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

// Variable boundary conditions

Note: The same structure is used for all variables

Case\Constant\g: Gravitational Acceleration

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0]; // units
value           (0 -9.81 0);      // Note: g is acting along y axis


// ************************************************************************* //
```

Case\Constant\radiationProperties: Laser properties are defined here

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1812                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      radiationProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

radiation       on;

radiationModel  laserDTRM;

// Number of flow iterations per radiation iteration
solverFreq 1;

absorptionEmissionModel localDensityAbsorptionEmission;

localDensityAbsorptionEmissionCoeffs
{
    alphaNames (alpha.gas alpha.metal);
    aCoeff (0 7.7e10);
    eCoeff (0 0);
    ECoeff (0 0);
}

mode    "Gaussian";
nTheta  100;          // Discretizing laser focal area in 100 lines and 10 rays on rach line
nr  10;

//focalLaserPosition constant   (250e-6 400e-6 250e-6);

focalLaserPosition table
(
    (0 (600e-6 790e-6 675e-6))          // Laser coordinates at different times (moving from 600um to
    (1.66e-3  (1600e-6 790e-6 675e-6))  1600 um in 1.66 msec.)
);


laserDirection constant     (0 -1 0);
laserPower          350;                // sigma is the effective radius beam
sigma               25.09e-6; //  Half of effective (waist) diameter 50.19 um at 0mm defocus
focalLaserRadius        35.13e-6;       // Nominal focal radius, 1.4 times the effective radius

scatterModel    none;

sootModel       none;

transmissivityModel none;
//reflectionModel FresnelLaser;

reflectionModel
(
    (gas and metal)
    {
        type            FresnelLaser;   // epsilon controls absorptivity. Epsilon value is defined in
        epsilon         0.35;           solver\laserDTRM\DTRMParticle\ DTRMParticle.C at line 158
    }
);
// ************************************************************************* //
```

Case\constants\transportProperties : To define thermophysical properties of material

```
/*--------------------------------- C++ ----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-----------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water air);

water      // Material
{
    transportModel  Newtonian;
    nu              [0 2 -1 0 0 0 0] 5.52e-7;          // Kinematic viscosity of material
    rho             [1 -3 0 0 0 0 0] 2643.34;          // Liquid density of material
}

air       // Argon
{
    transportModel  Newtonian;
    nu              [0 2 -1 0 0 0 0] 1.38e-5;         // Kinematic viscosity of Argon
    rho             [1 -3 0 0 0 0 0] 1.6337;          // Density of Argon
}

sigma                          [1 0 -2 0 0 0 0]    1.6;                 // This value is not used by the code, we are defining our own surface tension mode
a_sig                    a_sig [1 0 -2 0 0 0 0]    0.8878;              // surface tension equation coefficient             sig = 0.8878-1.56e-4(T-Tl)
b_sig                    b_sig [1 0 -2 -1 0 0 0]  -1.56e-4;             // surface tension equation coefficient
sig_limit            sig_limit [1 0 -2 0 0 0 0]    0.601;               // With temperature surface tension decreases, limiting surface tension to a value

cps                        cps [0 2 -2 -1 0 0 0]   1166.69;             // constant specific heat of solid material (can be taken as a function of T, with
cpl                        cpl [0 2 -2 -1 0 0 0]   1100.18;             // constant specific heat of liquid material(can be taken as a function of T, with
cp2                        cp2 [0 2 -2 -1 0 0 0]   520;                 // specific heat of Argon

ks                          ks [1 1 -3 -1 0 0 0]   194.46;              // constant thermal conductivity of Solid material (can be taken as a function of T, with modification
kl                          kl [1 1 -3 -1 0 0 0]   92.3;                // constant thermal conductivity of liquid material (can be taken as a function of T, with modification
k2                          k2 [1 1 -3 -1 0 0 0]   0.0177;              // thermal conductivity of Argon

beta1                    beta1 [0 0 0 -1 0 0 0]    0.85e-4;             // thermal expansion coefficient of material
beta2                    beta2 [0 0 0 -1 0 0 0]    3.4112e-3;           // thermal expansion coefficient of Argon

Tl                          Tl [0 0 0 1 0 0 0]     918.19;    // Liquidus temperature of material
Ts                          Ts [0 0 0 1 0 0 0]     843.95;    // Solidus temperature of material
L                            L [0 2 -2 0 0 0 0]    3.91e+5;   // Latent heat of fusion of material
C                            C [1 -3 -1 0 0 0 0]   1.5e+6;    // Constant used in Darcy drag equation (No need to change)
b                            b [0 0 0 0 0 0 0]     1e-10;     // Constant used in Darcy drag equation (No need to change)
dsigmadt               dsigmadt [1 0 -2 -1 0 0 0]  -1.56e-4;  // Dependency of surface tension coefficient with Temperature

E                            E [0 0 0 0 0 0 0]     0.09;      // Emissivity of material
sig                        sig [1 0 -3 -4 0 0 0]   5.67e-8;   // Stefan-Boltzman constant
Tref                      Tref [0 0 0 1 0 0 0]     300;       // Reference temperature (room temp)
h1                          h1 [1 0 -3 -1 0 0 0]   10;        // A reference value of heat transfer coefficient to calculate convective losses from the melt pool

L_v                        L_v [0 2 -2 0 0 0 0]    1.05e+7;   // Latent heat of vaporisation of material
R                            R [1 2 -2 -1 -1 0 0]  8.3143;    // Universal Gas constant
M                            M [1 0 0 0 -1 0 0]    0.026;     // Molar mass of material
P_0                        P_0 [1 -1 -2 0 0 0 0]   101000;    // Atmospheric pressure
T_v                        T_v [0 0 0 1 0 0 0]     2747.14;   // Evaporation temperature of material
```

Case\constants\turbulenceProperties : Laminar simulation

```
/*--------------------------------- C++ ----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-----------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


// ************************************************************************* //
```

Setting up the solver

1. Go to *solver\laserDTRM\DTRMParticle* directory and run "wclean" and "wmake" command using terminal. This will set up the DTRM class for ray tracing (before this, check the value of epsilon in solver\laserDTRM\DTRMParticle\ DTRMParticle.C at line 158).

```
        // Create a new reflected particle when the particles is not
        // transmissive and larger than an absolute I
        if (I_ > 0.01*myI0_ && ds > 0)  //I0_
        {
            vector pDir = dsv/ds;

            cellPointWeight cpw(mesh(), pos0, celll, face()); // position()
            vector nHat = td.nHatInterp().interpolate(cpw);

            nHat /= (mag(nHat) + ROOTSMALL);
            scalar cosTheta( pDir & nHat);
    scalar epsilon_ = 0.0625;            //0.0625
            // Only new incoming rays
            if (cosTheta > SMALL)
            {
                vector newDir = pDir + 2.0*(-pDir & nHat) * nHat;

                // reflectivity
                rho =
                    min
                    (
                        max
                        (
    0.5
```

1. Go to *solver* directory and run "wclean" and "wmake" command using terminal. This will set up the LPBF model.

Running the Case file

1. Go to *Case* directory and run "blockMesh" command command using terminal for mesh generation.
2. Run "setFields" command to set up the powder particles and substrate.
3. Run "decomposePar" command to divide the domain into subdomains for parallel processing.
4. Use "mpirun –np 16 solver_name -parallel" command to start computing.
5. After the end of simulation use "reconstructPar" command to integrate all time steps files.
6. Postprocess using paraview

ENJOY!!!!!!