Modifying buoyantBoussinesqSimpleFoam solver to consider a crop transpiration model using fvOptions

Jonas Sohn
jsohn@btech.au.dk

MOTIVATION

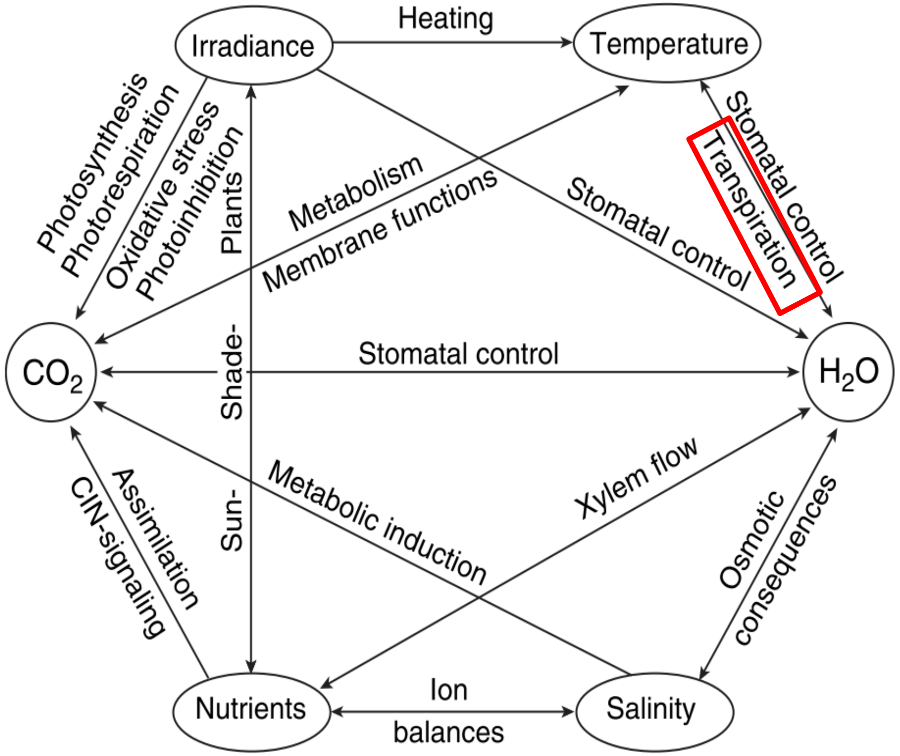


Figure 1: Plant Physiology adopted from Lüttge (2008).[1]

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code:T2EDK-VF2FARM).

AGENDA

1. Theoretical background
   - buoyantBoussinesqSimpleFoam
   - Porous media model
   - Crop transpiration model
2. Existing Implementations
3. Implementation of water vapor dynamics
   - buoyantBoussinesqSimpleFoam modification
   - Utilization of fvOptions
4. Hands-on tutorial
5. Conclusion

THEORETICAL BACKGROUND

1. Fundamentals of buoyantBoussinesqSimpleFoam solver

- Steady-state, incompressible buoyant flows using Boussinesq approximation

Boussinesq approximation:

- Assumes small density variations, only significant in the buoyancy term
- Governing equation: $\rho = \rho_0(1 - \beta\rho_0(T - T_{Ref}))$
- Implications: Efficient handling of buoyant flows for low Mach number scenarions

THEORETICAL BACKGROUND

1. Fundamentals of buoyantBoussinesqSimpleFoam solver

- **Continuity equation:**

$$\nabla \cdot \mathbf{U} = 0$$

- **Momentum equation:**

$$\rho \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla p + \nabla \cdot (\mu \nabla^2 \mathbf{U}) + \mathrm{g}\beta(T - T_{Ref})$$

- **Energy equation:** Tracks temperature variations.

$$\nabla \cdot (\rho \mathbf{U} T) - \nabla \cdot (\alpha \nabla T) = S_T$$

THEORETICAL BACKGROUND

2. Porous media model

- Models the interaction between fluid and porous structures, such as plant canopies
- **Darcy-Forchheimer Equation:**
  - K: Permeability
  - C_f: Inertial resistance coefficient

$$S = \underbrace{\frac{\mu}{K}\mathbf{u}}_{\text{Viscous Resistance}} + \underbrace{\frac{\rho C_{\mathrm{f}}}{\sqrt{K}}\|\mathbf{u}\|\mathbf{u}}_{\text{Inertial Resistance}}$$

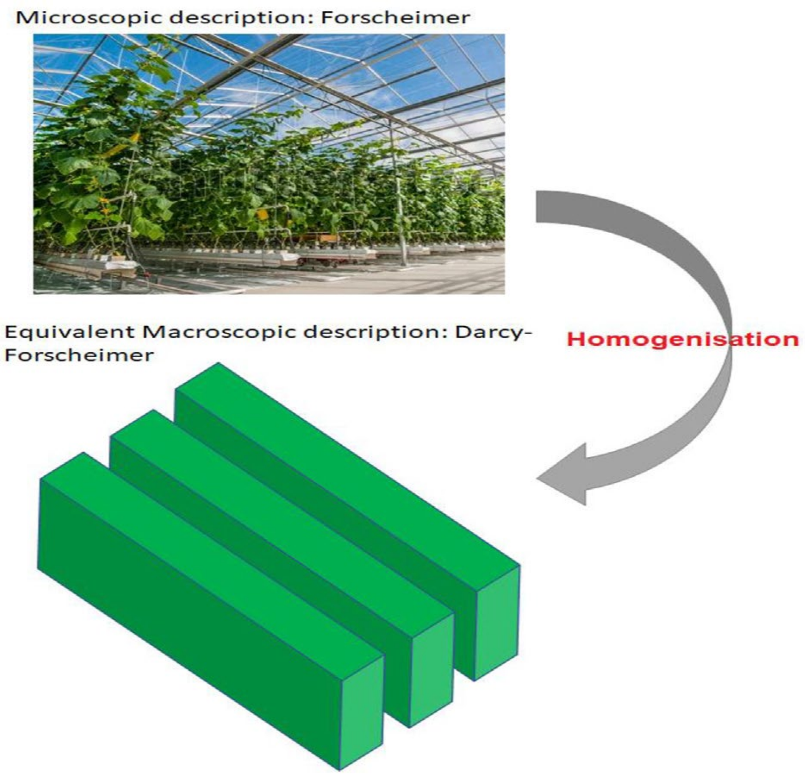


Figure 2: Description of the crop: homogenization method.[2]

THEORETICAL BACKGROUND

2. Porous media model

- Distinguishes between:
  - **Linear Resistance (Darcy term):** Dominant in laminar flow. $C_1 = \frac{1}{K}$
  - **Non-linear Resistance (Forchheimer term):** Relevant for turbulent or high-velocity flow $C_2 = \frac{C_f}{\sqrt{K}}$
- Handled using classes like $\text{explicityPorositySource}$ in OpenFOAM
- Helps to represent canopy drag and airflow resistance

THEORETICAL BACKGROUND

3. Crop transpiration model [3]

- To represent the exchange of water vapor from leaves into the atmosphere
- Energy balance of a crop: $R_{net} - Q_{sen} - Q_{lat} = 0$

$$R_{net} = 1(-\rho_r) \cdot I_{lighting} \cdot CAC \qquad Q_{sen} = LAI \cdot \rho_a \cdot c_p \cdot \frac{T_l - T_a}{r_a} \qquad Q_{lat} = \lambda_W \cdot ET$$

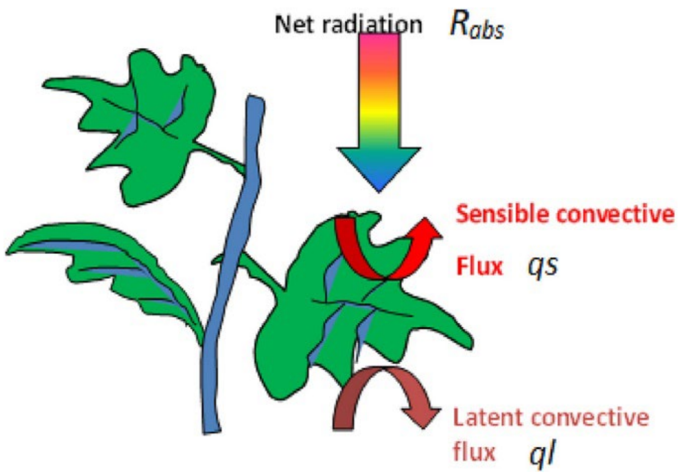


Figure 3: Net radiation, sensible and latent heat balances of leaves. [2]

THEORETICAL BACKGROUND

3. Crop transpiration model [3]

- Energy balance of a crop: $R_{net} - Q_{sen} - Q_{lat} = 0$

?

$$R_{net} = 1(-\rho_r) \cdot I_{lighting} \cdot CAC \qquad Q_{sen} = LAI \cdot \rho_a \cdot c_p \cdot \frac{T_l - T_a}{r_a} \qquad Q_{lat} = \lambda_W \cdot ET$$

- T_l is the only unknown variable but the equation can be solved iteratively using a root-finding algorithm (bi-section method)

$$ET = LAI \cdot \frac{\chi_l - \chi_a}{r_s + r_a}$$

$$r_s = 60\,\frac{1500 + PPFD}{200 + PPFD}$$

$$r_a = 350\left(\frac{l}{u}\right)^{0.5} LAI^{-1}$$

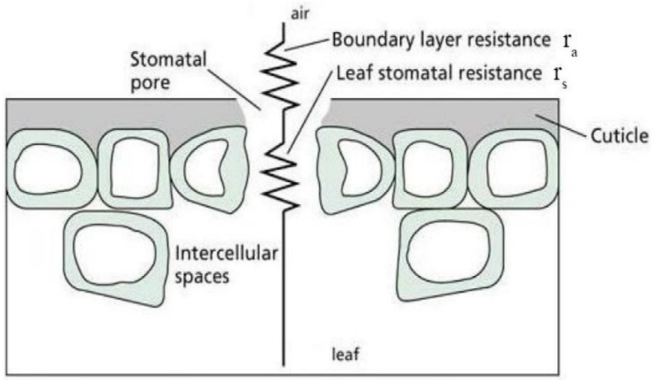


Figure 4: Resistances to water vapor transfer between leaf and air.[2]

EXISTING IMPLEMENTATIONS IN OPENFOAM

1. codedSource class

- Allows users to embed custom equations directly in the dictionary files and integrate “on-the-fly” source terms.
    - No source code modification or pre-compiling
    - Can define temperature, momentum, or transport source terms
- Inheritance from:
    - fv::cellSetOption – allows operation on specific cells in the mesh
    - codedBase – enables to execute custom code
- Constructor initialization of parent class and reads additional defined parameters from dictionary

Figure 5: Class declaration

```
template<class Type>
class CodedSource
:
    public fv::cellSetOption,
    protected codedBase
```

Figure 6: Constructor definition

```
template<class Type>
Foam::fv::CodedSource<Type>::CodedSource
(
    const word& name,
    const word& modelType,
    const dictionary& dict,
    const fvMesh& mesh
)
:
    fv::cellSetOption(name, modelType, dict, mesh)
{
    read(dict);
}
```

EXISTING IMPLEMENTATIONS IN OPENFOAM

2. explicitPorositySource class

- Also inherited from fv::cellSetOption
- A pointer is used to encapsulate different porosity models and calculate resistance contributions by pointing to the relevant model (e.g., Dracy-Forchheimer)
- Execute relevant apply() function inside correct() to update system equation UEqn
    - Templated apply() function computes the resistance terms

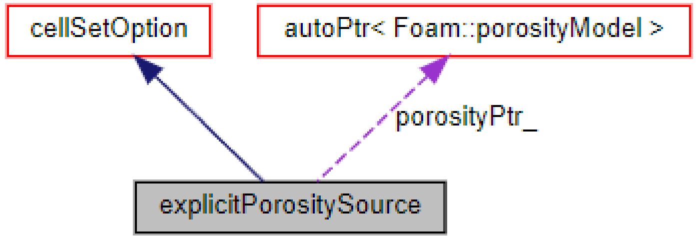


Figure 7: Source code in DarcyForchheimer.C file

```
void Foam::porosityModels::DarcyForchheimer::correct
(
    const fvVectorMatrix& UEqn,
    volTensorField& AU
) const
{
    const volVectorField& U = UEqn.psi();

    word rhoName(IOobject::groupName(rhoName_, U.group()));
    word muName(IOobject::groupName(muName_, U.group()));
    word nuName(IOobject::groupName(nuName_, U.group()));

    if (UEqn.dimensions() == dimForce)
    {
        const auto& rho = mesh_.lookupObject<volScalarField>(rhoName);
        const auto& mu = mesh_.lookupObject<volScalarField>(muName);

        apply(AU, rho, mu, U);
    }
    else
    {
        if (mesh_.foundObject<volScalarField>(nuName))
        {
            const auto& nu = mesh_.lookupObject<volScalarField>(nuName);

            apply(AU, geometricOneField(), nu, U);
        }
        else
        {
            const auto& rho = mesh_.lookupObject<volScalarField>(rhoName);
            const auto& mu = mesh_.lookupObject<volScalarField>(muName);

            apply(AU, geometricOneField(), mu/rho, U);
        }
    }
}
```

EXISTING IMPLEMENTATIONS IN OPENFOAM

2. explicitPorositySource – Example

- Dictionary configures an fvOption named porosity1 which applies explicitPorositySource
- selectionMode specifies the mode of cell selection (e.g., all, cellZone, cellSet, or points)
- type specifies the selected porosity model
- d and f define the Darcy and Forchheimer coefficient
- coordinateSystem defines the local coordinate system of the porous zone

Figure 8: *fvOptions file under system directory*

```
porosity1
{
    // Type of the fvOption
    type            explicitPorositySource;
    active          true;

    explicitPorositySourceCoeffs
    {
        // Porosity model type
        type            DarcyForchheimer;
        // Region to apply - porosity zone
        selectionMode   cellZone;
        cellZone        porousZone;

        DarcyForchheimerCoeffs
        {
            // Darcy coefficients (viscous resistance)
            d   (50 50 50);

            // Forchheimer coefficients (inertial resitance)
            f   (2 2 2);

            // Coordinate system for anisotropic porosity
            coordinateSystem
            {
                origin  (0 0 0);
                e1      (1 0 0);
                e2      (0 1 0);
            }
        }
    }
}
```

IMPLEMENTATION OF WATER VAPOR DYNAMICS

1. Water vapor implementation

- Continuity and temperature equation remain unchanged
- Momentum equation is modified:

$$\rho \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla p + \nabla \cdot (\mu \nabla^2 \mathbf{U}) + \mathrm{g}\left[\beta_T \left(T - T_{Ref}\right) + \beta_{AH}(AH - AH_{Ref})\right]$$

- Transport equation added for water vapor:

$$\mathbf{U} \cdot \nabla AH - \nabla \cdot \left(D_{eff} \nabla AH\right) = S_{AH}$$

$$D_{eff} = \frac{\nu}{Sc} + \frac{\nu_t}{Sct}$$

IMPLEMENTATION OF WATER VAPOR

2. buoyantBoussinesqSimpleFoam modification

- Modified files: buoyantBoussinesqSimpleFoam.C, AHEqn.H, createFields.H and readTransportProperties.H
  - Make/files if solver name will be change (e.g., cropTranFoam)

Figure 9: modified buoyantBoussinesqSimpleFoam.C file

```
        // Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "TEqn.H"
            #include "AHEqn.H"
            #include "pEqn.H"
        }
```

Figure 10: new created AHEqn.H file

```
{
    volScalarField DYEff("DYEff", turbulence->nu()/Sc + turbulence->nut()/Sct);

    fvScalarMatrix AHEqn
    (
        fvm::div(phi, AH)
      - fvm::laplacian(DYEff, AH)
      ==
        fvOptions(AH)
    );

    AHEqn.relax();

    fvOptions.constrain(AHEqn);

    solve(AHEqn);

    fvOptions.correct(AH);
}
```

IMPLEMENTATION OF WATER VAPOR

2. buoyantBoussinesqSimpleFoam modification

Figure 11: modified readTransportProperties.H file

```
// Water expansion coefficient
dimensionedScalar betaAH
(
    "betaAH",
    dimless,
    laminarTransport
);

// Reference water vapor mass fraction
dimensionedScalar AHRef("AHRef", dimless, laminarTransport);

// Laminar and Turbulent Schmidt number for water vapor transport
dimensionedScalar Sc("Sc", dimless, laminarTransport);
dimensionedScalar Sct("Sct", dimless, laminarTransport);
```

Figure 12: modified createFields.H file

```
// Declare and initialize water vapor mass fraction
Info<< "Reading field AH (absolute humidity - dimensionless)\n" << endl;
volScalarField AH
(
    IOobject
    (
        "AH",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

```
// Kinematic density for buoyancy force
volScalarField rhok
(
    IOobject
    (
        "rhok",
        runTime.timeName(),
        mesh
    ),
    // Buoyancy contribution of AH
    1.0 - beta*(T - TRef) - betaAH*(AH - AHRef)
);
```

IMPLEMENTATION OF WATER VAPOR

3. Utilization of fvOptions

- type – indicating a user-defined source term for a scalar field
- active (optional) – to “turn” source term on/off during simulation
- name – defines the name of the source term
- selectionMode – as cellZone to apply the source term only in the selection zone (porousZone)
- fields – is chosen based on the affected source term (e.g., AH)

Figure 13: *fvOptions file under system directory*

```
AHSource
{
    type            scalarCodedSource;
    active          true;
    name            AHSource;

    scalarCodedSourceCoeffs
    {
        selectionMode  cellZone;
        cellZone       porousZone;
        fields         (AH);

```

IMPLEMENTATION OF WATER VAPOR

3. Utilization of fvOptions

- codeInclude section (optional) to define helper functions and constant
- Follows a python implementation[4] based on Graamans et al. (2018) [3] described theory

Figure 14: fvOptions file under system directory

```
fields          (AH);

codeInclude
#{
    // Constant values for case study
    const double lai = 3.0;
    const double l_leaf = 0.1;
    const double reflectionCoefficient = 0.05;
    const double ppfd = 200.0;
    const double cultivationAreaCoverage = 0.95;
```

```
//- Calculates the sensible heat exchange between the surface and air.
// @param tempAir Air temperature in degrees Celsius.
// @param tempSurface Surface temperature in degrees Celsius.
// @param lai Leaf area index (area of leaves per unit ground area).
// @param vapourResistance Vapor resistance in s/m.
// @return Sensible heat exchange in W/m².
double calcSensibleHeatExchange
(
    double tempAir,
    double tempSurface,
    double lai,
    double vapourResistance
```

```
double calcSensibleHeatExchange
(
    double tempAir,
    double tempSurface,
    double lai,
    double vapourResistance
)
{
    return lai * HEAT_CAPACITY_OF_AIR * DENSITY_OF_AIR * \
    (tempSurface - tempAir) / vapourResistance;
}
```

IMPLEMENTATION OF WATER VAPOR

3. Utilization of fvOptions

- Provide fields values (allow T to be modified)
- Access the source term in the governing equation (eqn)
- Access porous zone IDs
- Loop through all cells in the mesh and check if cell belongs to the porous zone

```
codeAddSup
#{
    // Provide field values
    const volScalarField& AH = mesh().lookupObject<volScalarField>("AH");
    const volVectorField& U = mesh().lookupObject<volVectorField>("U");
    // Allow modification of T
    volScalarField& T = const_cast<volScalarField&>(mesh().lookupObject<volScalarField>("T"));
    // Reference to the source term in the equation
    scalarField& AHSource = eqn.source();
    // Access the porous zone ID
    const label porousZoneID = mesh().cellZones().findZoneID("porousZone");
    // Loop through the cells in mesh
    forAll(AH, cellID)
    {
        // Execute code if cell is in porousZone
        if (mesh().cellZones().whichZone(cellID) == porousZoneID)
        {
```

Figure 15: fvOptions file under system directory

```
// Get field values for cellID
double tempInKelvin = T[cellID];
double absolute_humidity = AH[cellID];
vector Ucell = U[cellID];
scalar u = mag(Ucell);
// Convert K into °C
double tempAir = convertKelvinToCelsius(tempInKelvin);
// Convert AH into RH
double relativeHumidity = calcRelativeHumidity(tempAir, absolute_humidity);
// Calculate vapour/aerodynamic resistance
double vapourResistance = calcVapourResistance(l_leaf, u, lai);

// Calculate surface temperature - root-finding method
double tempSurface = calcTempSurface(tempAir, ppfd, relativeHumidity, lai,
                    vapourResistance, reflectionCoefficient, cultivationAreaCoverage);

// Calculate sensible heat exchange and latent heat flux
double sensibleHeatExchange = calcSensibleHeatExchange(tempAir, tempSurface, lai,
                    vapourResistance);
double latentHeatFlux = calcLatentHeatFlux(tempAir, tempSurface, relativeHumidity,
                    ppfd, lai, vapourResistance);

// Calculate the rate of mass flux per volume dervied from transpiration rate (ET)
scalar ET = latentHeatFlux / LATENT_HEAT_WATER / l_leaf;
// Normalized the mass flux to a flux per time-unit
ET = ET / DENSITY_OF_WATER;
// Add as a source term to the right-hand side of the transport equation
AHSource[cellID] -= ET;

// Calculate the specific heat capacity of moist air
scalar specific_heat_moist_air = HEAT_CAPACITY_OF_AIR * \
                (1 - AH[cellID]) + (AH[cellID] * HEAT_CAPACITY_OF_WATER);
// Calculate the mass of moist air
scalar m_air = (ATMOSPHERIC_PRESSURE / (287 * tempInKelvin)) / (1 + 0.61 * AH[cellID]);
scalar delta_T = sensibleHeatExchange / (m_air * specific_heat_moist_air);
// Calculate temperature increase delta_T
T[cellID] += delta_T;
```

IMPLEMENTATION OF WATER VAPOR

3. Utilization of fvOptions

- Requires several input parameters
- Net radiation is calculated to return available energy for heat and mass exchange
- Use a lambda function for energy balance
  - Call helper functions to compute fluxes
- Solve energy balance using bisection method
  - Set initial temperature bounds
  - Iterative refinement of the bounds
- Return surface temperature

Figure 16: fvOptions file under system directory

```cpp
double calcTempSurface
(
    double tempAir,
    double ppfd,
    double relativeHumidity,
    double lai,
    double vapourResistance,
    double reflectionCoefficient,
    double cultivationAreaCoverage
)
{
    double netRadiation = calcNetRadiation
        (ppfd, reflectionCoefficient, cultivationAreaCoverage);
    auto calcEnergyBalance = [&](double tempSurface)
    {
        double sensibleHeat = calcSensibleHeatExchange
            (tempAir, tempSurface, lai, vapourResistance);
        double latentHeat = calcLatentHeatFlux
            (tempAir, tempSurface, relativeHumidity, ppfd, lai, vapourResistance);
        return netRadiation - sensibleHeat - latentHeat;
    };

    // Root finding using bisection method
    double limit = 10.0;
    double xa = tempAir - limit;
    double xb = tempAir + limit;

    double tol = 1e-6;
    while (std::fabs(xb - xa) > tol)
    {
        double xm = (xa + xb) / 2.0;
        if (calcEnergyBalance(xm) * calcEnergyBalance(xa) < 0) {
            xb = xm;
        } else {
            xa = xm;
        }
    }

    return (xa + xb) / 2.0;
}
```

TUTORIAL CASE

- Modified version of hotRoom tutorial
    - Scale 0.1 & grid resolution increased to (40 20 40)
- Adjusted initial conditions in 0.orig directory according to new boundary conditions
- Updated transportProperties and controlDict file
- Created new topoSetDict file for porous zone

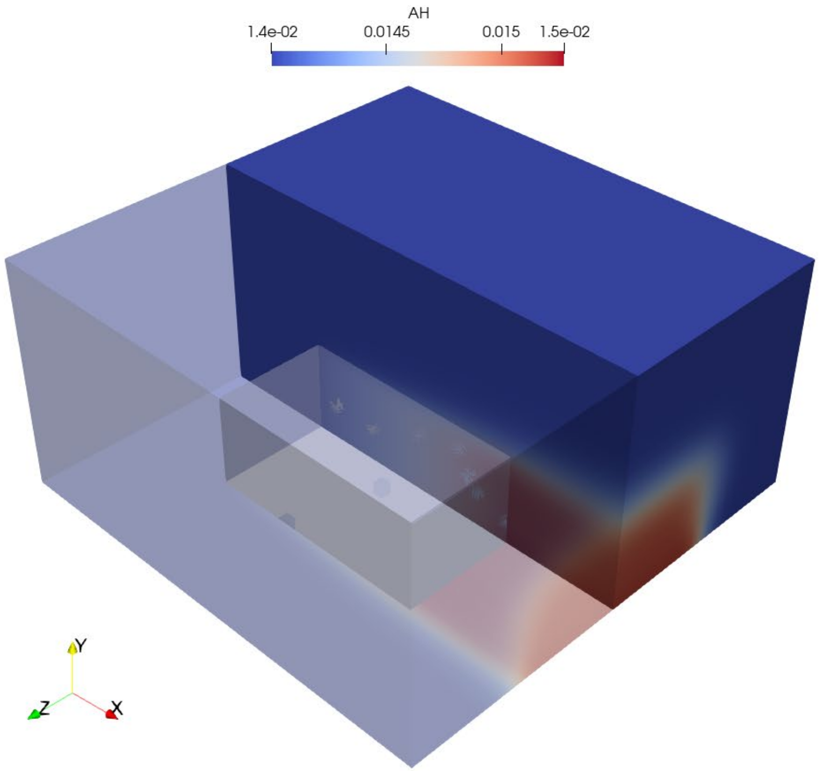
Figure 17: blockMeshDict in system-directory

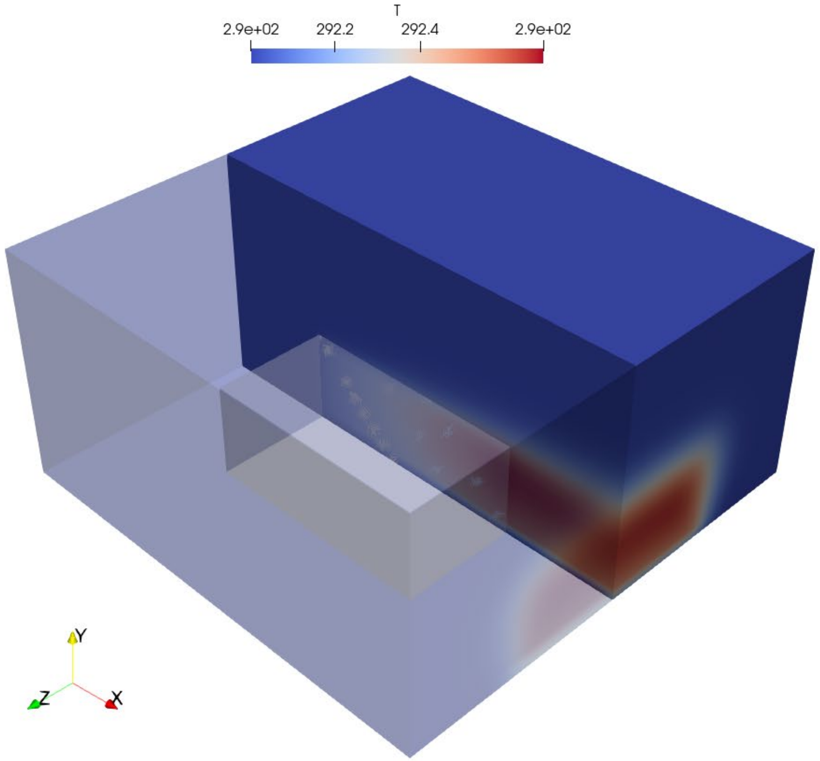
```
boundary
(
    floor
    {
        type wall;
        faces
        (
            (1 5 4 0)
        );
    }
    ceiling
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    leftWall
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    rightWall
    {
        type patch;
        faces
        (
            (1 5 6 2)
        );
    }
    frontWall
    {
        type wall;
        faces
        (
            (0 1 2 3)
        );
    }
    backWall
    {
        type wall;
        faces
        (
            (4 5 6 7)
        );
    }
);
```

Figure 18: U field in 0.orig directory

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    floor
    {
        type            noSlip;
    }

    ceiling
    {
        type            noSlip;
    }

    leftWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    rightWall
    {
        type zeroGradient;
    }

    "(backWall|frontWall)"
    {
        type            noSlip;
    }
}
```

TUTORIAL CASE





CONCLUSION

- Modified version of buoyantBoussinesqSimpleFoam solver
- Explained and used explicitPorositySource and codedSource under the fvOptions dictionary

Next step:

- Advance the fvOptions file to account for radiation models (fvDOM)

AARHUS
BSS

REFERENCES

[1] – Lüttge, U. (2008). Encyclopedia of Ecology. Encyclopedia of Ecology, 1999, 2837–2845. http://www.sciencedirect.com/science/article/pii/B9780080454054005358

[2} – H. Fatnassi, P. E. Bournet, T. Boulard, J. C. Roy, F. D. Molina-Aiz, and R. Zaaboul, "Use of computational fluid dynamic tools to model the coupling of plant canopy activity and climate in greenhouses and closed plant growth systems: A review," Biosystems Engineering, vol. 230, pp. 388–408, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1537511023000983

[3] - L. Graamans, A. Van Den Dobbelsteen, E. Meinen, and C. Stanghellini, "Plant factories; crop transpiration and energy balance," vol. 153, pp. 138–147. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0308521X16306515

[4] - https://github.com/linucks/fu_evapotranspiration