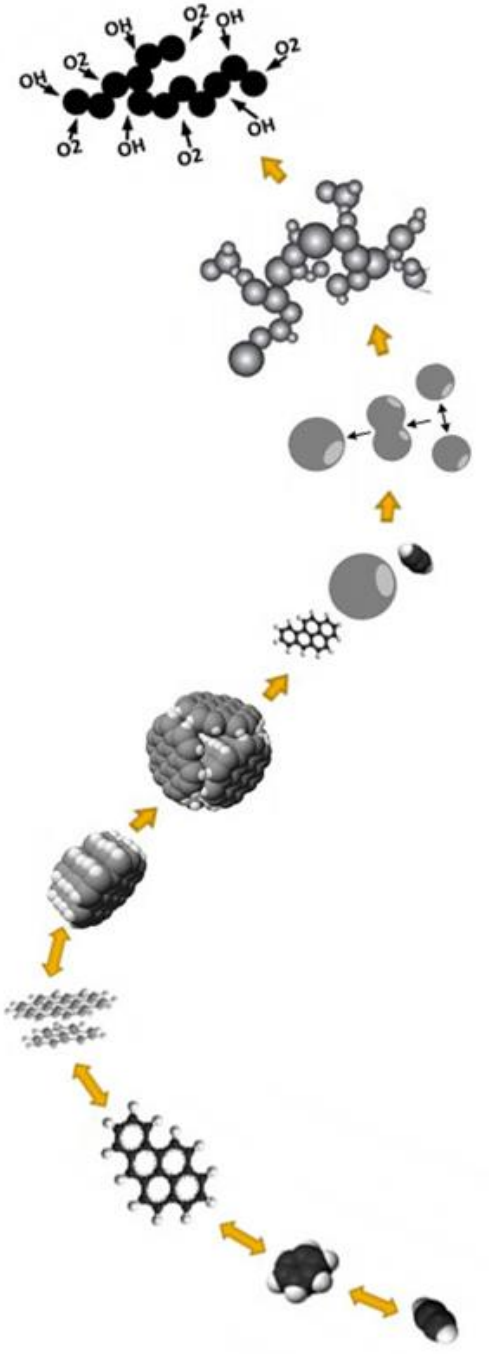



# Implementation of a Monodisperse Population Balance Model in Laminar Combustion Model

Mo Adib

PhD Student at Mechanical and Aerospace Department
Carleton University, Ottawa, Canada

Jan 17th, 2023

Kholghy, M.R., Veshkini, A. and Thomson, M.J., "The core–shell internal nanostructure of soot–A criterion to model soot maturity. Carbon" (2016)

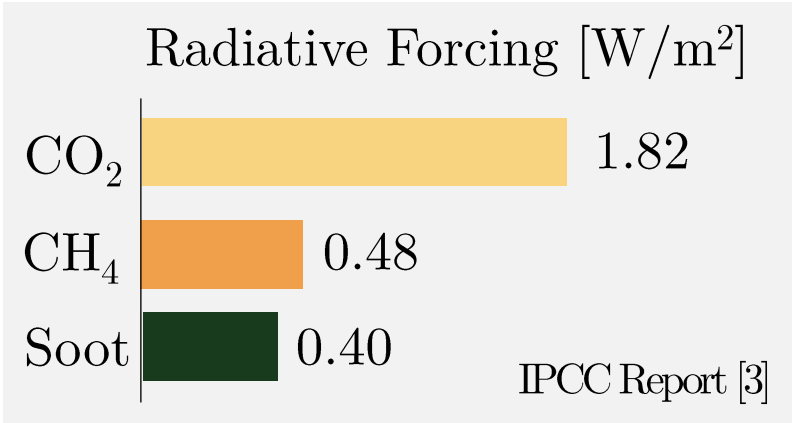
# Carbonaceous Nanoparticles

## Soot

[1]

9.5 megatons of soot per year!

Third strongest contributor to climate change →




Produced Industrially →

## Carbon Black

Largest flame-made nanomaterial by volume and value

↙ 15 megatons per year!

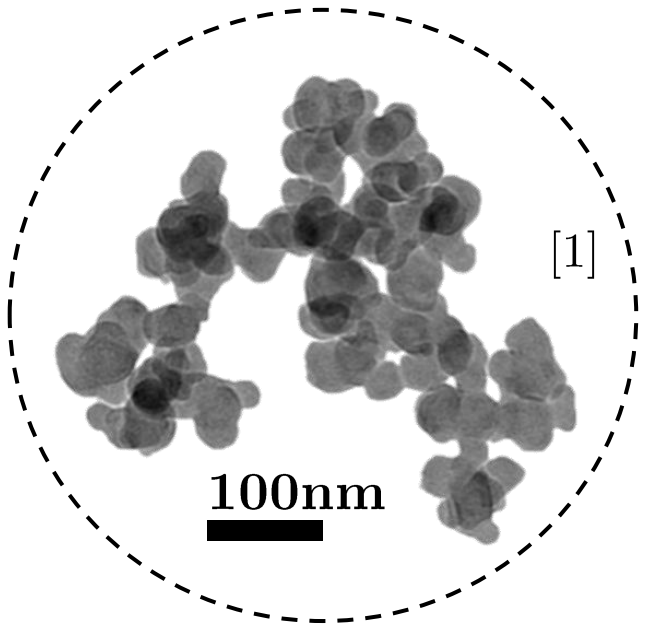
↘ $17B

[1] https://sootcolourant.wordpress.com/concept/

[2] Myhre G, Shindell D, Pongratz J. Anthropogenic and natural radiative forcing 2014

# Modelling Soot Characteristics

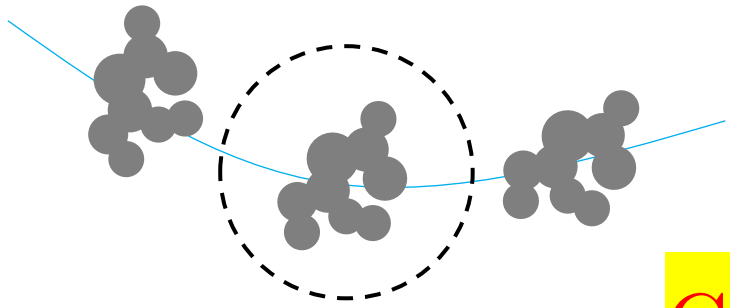
Agglomerate/Aggregate




Particle Morphology

- Climate impacts of soot
- Functional properties of Carbon Black

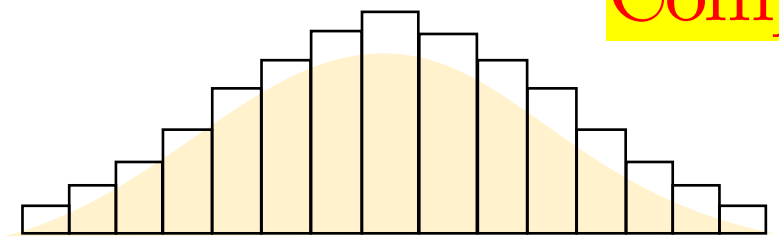
→ Discrete Element Modelling



Tracking individual particles

Computationally Expensive!

→ Sectional Population Balance Models



PSD

Tracking size sections

[1] Lapuerta, M., Barba, J., Sediako, A.D., Kholghy, M.R. and Thomson, M.J., J. Aerosol Sci. (2017).

# 4 Equation Monodisperse Population Balance Model [1,2]

$N_{agg}$

Agglomerate number density

$N_{pri}$

Primary particle number density

$C_{tot}$

Total carbon content

$H_{tot}$

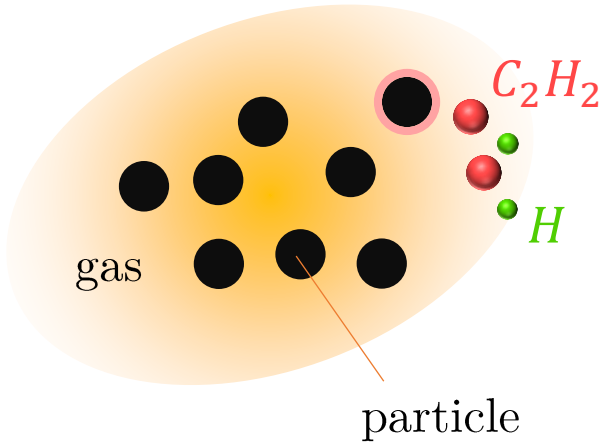
Total hydrogen content

$$d_p = \left(\frac{6}{\pi}\frac{C_{tot} W_{carbon}}{\rho_{soot} N_{pri} Av}\right)^{1/3}$$

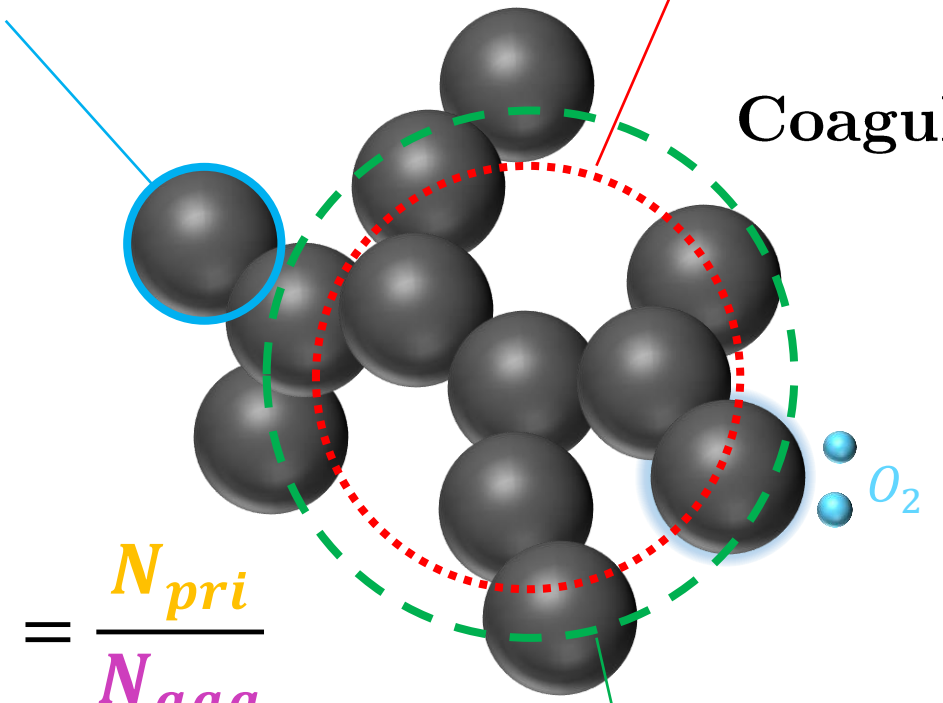
$$d_m = d_p \, n_p^{0.45} \quad [3]$$

## Inception

Dimerization of PAHs [5]




## Coagulation




$$n_p = \frac{N_{pri}}{N_{agg}}$$

$$d_g = \frac{d_m}{n_p^{-0.2} + 0.4} \quad [3]$$


[1] Kholghy and Kelesidis, Combust Flame (2021)
[2] Kholghy and Schumann, Energy Fuels (2021)
[3] G.A. Kelesidis, E. Goudeli, S.E. Pratsinis, Carbon (2017)
[4] G. Blanquart, P. Pepiot-Desjardins, H. Pitsch, Combust. Flame. (2009)
[5] Kholghy, M. R., Kelesidis, G. A., & Pratsinis, S. E., Phys. Chem. Chem. Phys. (2018)

# Transport Equations

$$\frac{\partial}{\partial t}(\rho N_{agg}) + \nabla \cdot (\rho u N_{agg}) + \nabla^2(\rho D N_{agg}) = \rho \left( S^N_{inc} + S^N_{coag} \right),$$

$$\frac{\partial}{\partial t}(\rho N_{pri}) + \nabla \cdot (\rho u N_{pri}) + \nabla^2(\rho D N_{pri}) = \rho S^N_{inc},$$

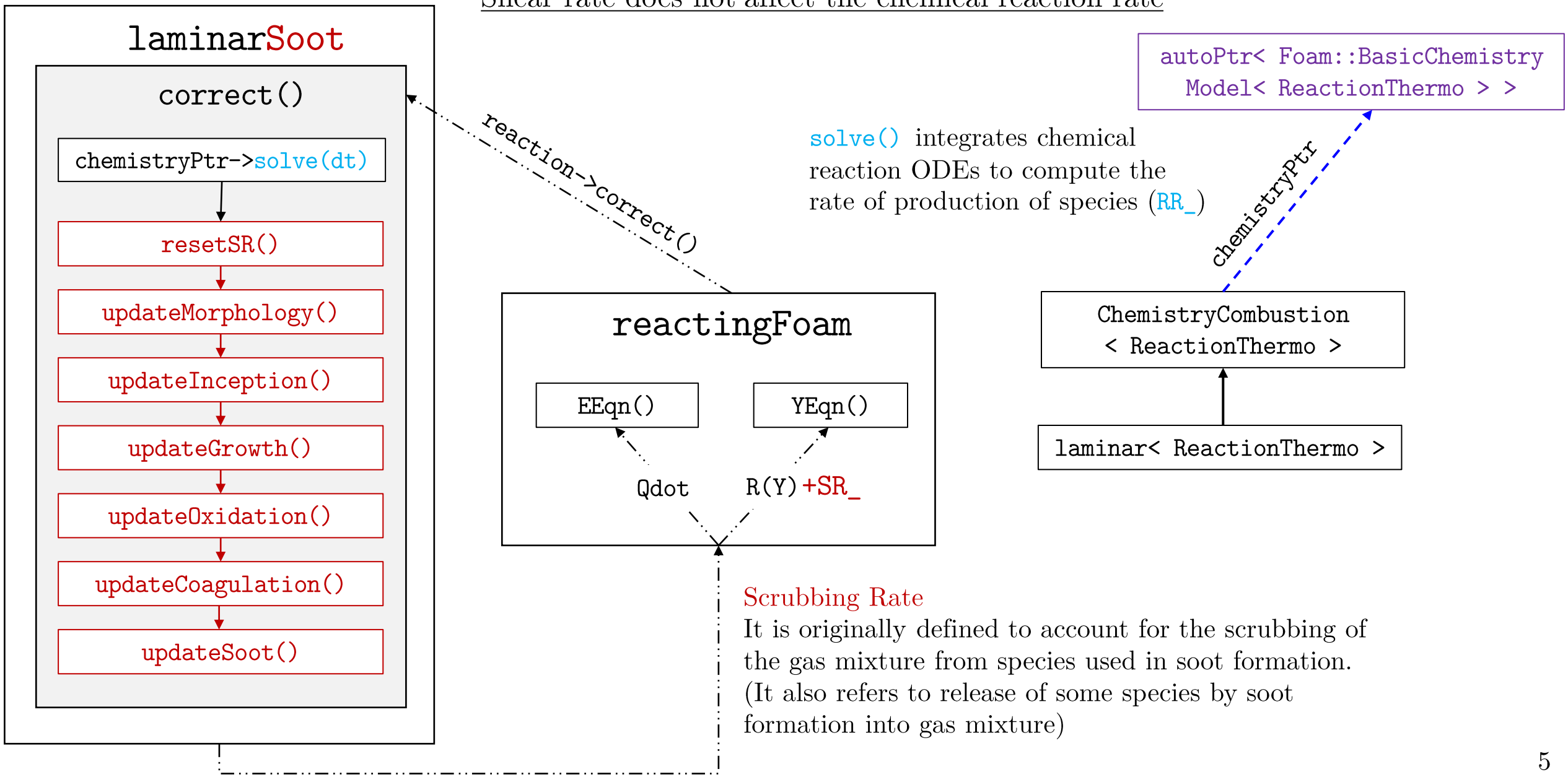
$$\frac{\partial}{\partial t}(\rho C_{tot}) + \nabla \cdot (\rho u C_{tot}) + \nabla^2(\rho D C_{tot}) = \rho \left( S^C_{inc} + S^C_{grow} + S^C_{ox} \right),$$

$$\frac{\partial}{\partial t}(\rho H_{tot}) + \nabla \cdot (\rho u H_{tot}) + \nabla^2(\rho D H_{tot}) = \rho \left( S^H_{inc} + S^H_{grow} + S^H_{ox} \right).$$

Inception + Surface growth + Oxidation + Coagulation → Solve Transport Equations

# Laminar Combustion Model

Shear rate does not affect the chemical reaction rate




`solve()` integrates chemical reaction ODEs to compute the rate of production of species (`RR_`)

Scrubbing Rate

It is originally defined to account for the scrubbing of the gas mixture from species used in soot formation. (It also refers to release of some species by soot formation into gas mixture)

# Implementation – Model Constants

Physical constants used in the model are introduced as `static member data`

```
template<class ReactionThermo>
class laminarSoot
:
    public ChemistryCombustion<ReactionThermo>
{
    // Static data
    static const label pi_ = Foam::constant::mathematical::pi;
    static const Foam::dimensionedScalar Av_;
    static const Foam::dimensionedScalar kB_;
    static const Foam::dimensionedScalar Ru_;
    static const Foam::dimensionedScalar rho_soot_;
    static const Foam::dimensionedScalar W_carbon_;
    static const Foam::dimensionedScalar W_hydrogen_;
    static const Foam::dimensionedScalar C_min_;
    static const Foam::dimensionedScalar H_min_;
    static const Foam::dimensionedScalar PAH_rho_const_;
```

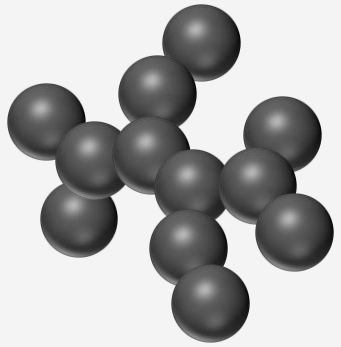
# Implementation – Tracked and Derived Fields

```
// Soot tracked fields
volScalarField N_agg_;
volScalarField N_pri_;
volScalarField C_tot_;
volScalarField H_tot_;
// Morphology
volScalarField n_p_;
volScalarField d_p_;
volScalarField d_m_;
volScalarField d_g_;
volScalarField A_tot_;
// Source Terms
// Inception
volScalarField S_inc_N_;
volScalarField S_inc_C_tot_;
volScalarField S_inc_H_tot_;
// Surface growth
volScalarField S_grow_C_tot_;
volScalarField S_grow_H_tot_;
// Oxidation
volScalarField S_ox_C_tot_;
// Coagulation
volScalarField S_coag_N_agg_;
```

MPBM

$N_{agg}$ $N_{pri}$ $C_{tot}$ $H_{tot}$

Morphology



Inception

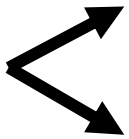
Surface growth

Oxidation

Coagulation

# Implementation – Precursor Properties

Generating properties of soot precursors (PAHs)



species that form dimers that leads to soot inception

species that are adsorbed on soot particles

```
PAHs (A2 A3 A4 A2R5);
```

```
template<class ReactionThermo>
bool Foam::combustionModels::laminarSoot<ReactionThermo>::createPAHProps()
{
    Info << "PAH names: " << PAH_names_ << endl;
    PAH_n_C_.resize(PAH_names_.size());
    PAH_n_H_.resize(PAH_names_.size());
    PAH_indicies_.resize(PAH_names_.size());

    const ReactionThermo& thermo = this->thermo();
    const dictionary thermoDict = IFstream(fileName(thermo.lookup("foamChemistryThermoFile")).expand()
     )();

    forAll(PAH_names_, i)
    {
        PAH_indicies_[i] = this->thermo().composition().species()[PAH_names_[i]];
        const dictionary* elemsDict = thermoDict.subDict(PAH_names_[i]).findDict("elements");
        wordList elemNames(elemsDict->toc());
    ...
```

# Implementation – Thermodynamic Properties

```
A2R5
{
    specie
    {
        molWeight       152.19756;
    }
    thermodynamics
    {
        Tlow            300;
        Thigh           3000;
        Tcommon         1000;
        highCpCoeffs    ( 45.883698 -0.027226903 4.1569336e-05 -1.8047093e-08 2.5351396e-12 13394.574 -223.04584 );
        lowCpCoeffs     ( -9.7011614 0.12019449 -9.8907694e-05 3.7240884e-08 -4.1124578e-12 29601.926 66.970596 );
    }
    transport
    {
        As              9.91969e-07;
        Ts              671.96171;
    }
    elements
    {
        C               12;
        H               8;
    }
}
```

# Implementation – Dimer Properties

Generating properties of dimers from PAH molecules → Dimers instantly merge into incipient soot particles accounting for the inception

```
template<class ReactionThermo>
bool Foam::combustionModels::laminarSoot<ReactionThermo>::createDimerProps()
{
    ...
    for (int i = 0; i < PAH_names_.size(); i++) {
        for (int j = i; j < PAH_names_.size(); j++) {
            dimer_id += 1;
            dimer_names_[dimer_id] = PAH_names_[i] + PAH_names_[j];
            dimer_n_C_[dimer_id] = PAH_n_C_[i] + PAH_n_C_[j];
            dimer_n_H_[dimer_id] = PAH_n_H_[i] + PAH_n_H_[j];
            dimer_PAH_1_index_[dimer_id] = composition.species()[PAH_names_[i]];
            dimer_PAH_2_index_[dimer_id] = composition.species()[PAH_names_[j]];
            dimer_PAH_1_id_[dimer_id] = i;
            dimer_PAH_2_id_[dimer_id] = j;
        }
    }
```

# Implementation – Modifying `R(Y)` function

```
template<class ReactionThermo>
Foam::tmp<Foam::fvScalarMatrix>
Foam::combustionModels::laminarSoot<ReactionThermo>::R(volScalarField& Y) const
{
    tmp<fvScalarMatrix> tSu(new fvScalarMatrix(Y, dimMass/dimTime));

    fvScalarMatrix& Su = tSu.ref();

    if (this->active())
    {
        const label specieI =
            this->thermo().composition().species()[Y.member()];

        Su += this->chemistryPtr_->RR(specieI);
        // scrubbing rates are added
        if (scrubbing_enabled_){
            if (speciesList_.found(Y.member()))
            {
                label spid = speciesIds_[Y.member()];
                Su += SR_[spid];
            }
        }
    }

    return tSu;
}
```

Checks if the current species is among the species used by soot model

RR is adjusted for the scrubbing rate

# Implementation – `resetSR()` function

It loops over `fields` in `SR_` and zeros all the elements.

```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::resetSR()
{
    if (scrubbing_enabled_){
        forAll(SR_, fieldi){
            forAll(SR_[fieldi], celli){
                SR_[fieldi][celli] = 0.0;
            }
        }
    }
}
```

# Implementation – updateMorphology() function

```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::updateMorphology()
{
    // number of primary particles
    n_p_ = N_pri() / N_agg();
    n_p_.max(1.0);
    if (!coagulation_enabled_)
    {
        Info<< "enforcing n_p = 1\n" << endl;
        n_p_.min(1.00000001);
    }
    // Primary particle diameter
    d_p_ = pow (
        (6.0 / pi_) *
        (C_tot() * W_carbon_) / rho_soot_ *
        1.0 / (N_pri() * Av_)
        , 1.0/3.0
    );
    // Surface area of each primary particle
    A_tot_ = N_pri() * Av_ * pi_ * d_p_ * d_p_;
    //  Mobility diameter
    d_m_ = d_p_ * pow(n_p_, 0.45);
    // Gyration Diameter
    volScalarField n_p_lowerlimit (n_p_*0.0+1.5);
    d_g_ = (n_p_ <= n_p_lowerlimit) * (d_m_ / 1.29) + (n_p_ > n_p_lowerlimit) * (d_m_ / (pow(n_p_,
     -0.2)+0.4));

}
```

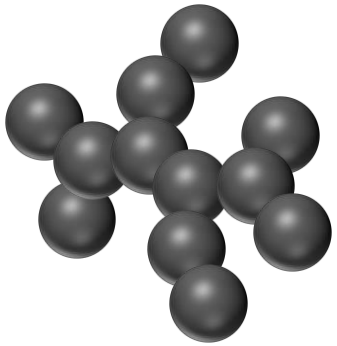
$$n_p = \frac{N_{pri}}{N_{agg}}$$

$$d_p = \left(\frac{6}{\pi}\frac{C_{tot} W_{carbon}}{\rho_{soot} N_{pri} Av}\right)^{1/3}$$

$$d_m = d_p \, n_p^{0.45}$$

$$d_g = \frac{d_m}{n_p^{-0.2} + 0.4}$$

Morphology



# Implementation – `updateInception()` function

```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::updateInception()
{
    if (inception_enabled_){
        volScalarField rho = this->thermo().rho();
        forAll(dimer_names_, i)
        {
            // PAH Index and Id
            label id1 = dimer_PAH_1_id_[i];
            label id2 = dimer_PAH_2_id_[i];
            volScalarField dimerROPField(dimerROP(id1, id2));
            // N_agg Source Term
            S_inc_N_ += (dimer_n_C_[i] / C_min_) * dimerROPField / rho;
            // C_tot Source Term
            S_inc_C_tot_ += dimer_n_C_[i] * dimerROPField / rho;
            // H_tot Source Term
            S_inc_H_tot_ += dimer_n_H_[i] * dimerROPField / rho;
            if (scrubbing_enabled_){
                // PAHs
                // species id
                label spid1 = speciesIds_[PAH_names_[id1]];
                label spid2 = speciesIds_[PAH_names_[id2]];
                // species index
                label spindex1 = speciesIndicies_[PAH_names_[id1]];
                label spindex2 = speciesIndicies_[PAH_names_[id2]];
                SR_[spid1] -= dimerROPField * W(spindex1);
                SR_[spid2] -= dimerROPField * W(spindex2);

                // H2
                label H2_id = speciesIds_["H2"];
                label H2_index = speciesIndicies_["H2"];
                SR_[H2_id] += dimerROPField * W(H2_index);
```

$$\mathrm{Dimer}^{*}_{ij} \xrightarrow{k_{REAC}} \mathrm{Dimer}_{ij}$$

Inception source term from the rate of production (`ROP`) of dimers

Removal of the PAHs involved in the inception

Release of $H_2$ during the inception

# Implementation – `updateGrowth()` function (1)

```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::updateGrowth()
{
    ...
    if (HACA_growth_enabled_)
    {
        volScalarField rho = this->thermo().rho();
        volScalarField HACAGrowthRateField(HACAGrowthRate());
        S_grow_C_tot_ += 2 * HACAGrowthRateField / rho;
        S_grow_H_tot_ += 2 * HACAGrowthRateField * (0.25 / 2.00) / rho;

        if (scrubbing_enabled_){
            // C2H2
            label C2H2_id = speciesIds_["C2H2"];
            label C2H2_index = speciesIndicies_["C2H2"];
            SR_[C2H2_id] -= HACAGrowthRateField * W(C2H2_index);

            // H
            label H_id = speciesIds_["H"];
            label H_index = speciesIndicies_["H"];
            SR_[H_id] += HACAGrowthRateField * W(H_index) * (1.75 / 2.00);
        }
    }
```

$$C_{soot^\circ} + C_2H_2 \longrightarrow C_{soot-H}$$

Growth rate of soot particle via HACA

Removal of $C_2H_2$ from gas mixture by HACA

Release of H radical by HACA

# Implementation – `updateGrowth()` function (2)

It loops over PAHs and calculates the rate of adsorption of each PAH on soot particles

```
    if (PAH_growth_enabled_)
    {
        volScalarField rho = this->thermo().rho();
        forAll(PAH_names_, id)
        {
            volScalarField PAHAdsorptionRateField(PAHAdsorptionRate(id));
            S_grow_C_tot_ += PAH_n_C_[id] * PAHAdsorptionRateField / rho;
            S_grow_H_tot_ += (PAH_n_H_[id] - 2) * PAHAdsorptionRateField / rho;

            if (scrubbing_enabled_){
                // PAH
                // species id
                label spid = speciesIds_[PAH_names_[id]];
                // species index
                label spindex = speciesIndicies_[PAH_names_[id]];
                SR_[spid] -= PAHAdsorptionRateField * W(spindex);

                // H
                label H_id = speciesIds_["H"];
                label H_index = speciesIndicies_["H"];
                SR_[H_id] += PAHAdsorptionRateField * W(H_index) * 2;
            }
        }
    }

}
```

$$\text{Soot}-\text{PAH}^{*} \xrightarrow{k_{rc,ad}} \text{Soot}-\text{PAH}.$$

Growth rate of soot particle via PAH Adsorption

Removal of PAH from gas mixture by PAH adsorption

Release of H radical by PAH adsorption

# Implementation – `updateOxidation()` function

```
Foam::combustionModels::laminarSoot<ReactionThermo>::updateOxidation()
{
    S_ox_C_tot_ *= 0.0;
    if (HACA_oxidation_enabled_){
        volScalarField rho(this->thermo().rho());
        volScalarField HACAO2OxidationRateField(HACAO2OxidationRate());
        volScalarField HACAOHOxidationRateField(HACAOHOxidationRate());
        S_ox_C_tot_ += -1 * (HACAO2OxidationRateField + HACAOHOxidationRateField) / rho;

        if (scrubbing_enabled_){
            // O2
            label O2_id = speciesIds_["O2"];
            label O2_index = speciesIndicies_["O2"];
            SR_[O2_id] -= 0.5 * HACAO2OxidationRateField * W(O2_index);

            // CO2
            label CO_id = speciesIds_["CO"];
            label CO_index = speciesIndicies_["CO"];
            SR_[CO_id] += (HACAO2OxidationRateField + HACAOHOxidationRateField) * W(CO_index);

            // OH
            label OH_id = speciesIds_["OH"];
            label OH_index = speciesIndicies_["OH"];
            SR_[OH_id] -= HACAOHOxidationRateField * W(OH_index);
        }
    }
}
```

$$C_{soot^\circ} + O_2 \longrightarrow 2\,CO$$

$$C_{soot} - H + OH \longrightarrow CO + \frac{1}{2}\,H_2$$

Oxidation of soot particles by $O_2$ and OH

Removal of $O_2$ from gas mixture by oxidation

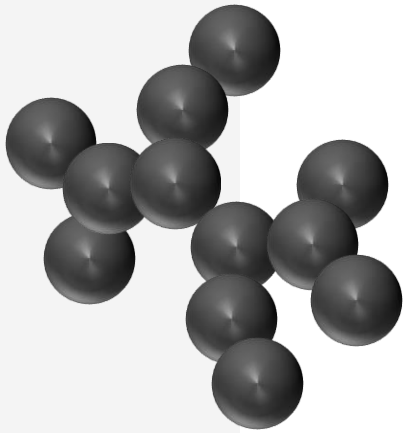
Release of $CO_2$ by oxidation

Removal of OH from gas mixture by oxidation

# Implementation – `updateCoagulation()` function

It calculates the coagulation rate which is the rate of decay of particles by binary collision.

```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::updateCoagulation()
{
    const volScalarField& T = this->thermo().T();
    volScalarField mu (this->thermo().mu());
    volScalarField rho (this->thermo().rho());
    // Free Molecule
    volScalarField beta_fm
    (
        4 * pow(pi_ * kB_ * T / m_agg(), 0.5) * d_g_ * d_g_
    );
    // Continuum
    volScalarField beta_cont(
        (8 * kB_ / (3 * mu)) * T * ( 1.0 + (2.0 * lambda_gas() / d_m_ )*(1.21 + 0.4*exp(-0.78*d_m_/
     lambda_gas())))
    );
    // Coagulation source term
    if (coagulation_enabled_){
        S_coag_N_agg_ = 0.5 * 1.82 * beta_fm * beta_cont / (beta_fm + beta_cont) * pow(N_agg(), 2.0) *
      Av_ * rho;
    }
}
```



$$S^N_{coag} = -\frac{1}{2}\beta N^2_{agg}$$

Coagulation source term

# Implementation – `updateSoot()` function

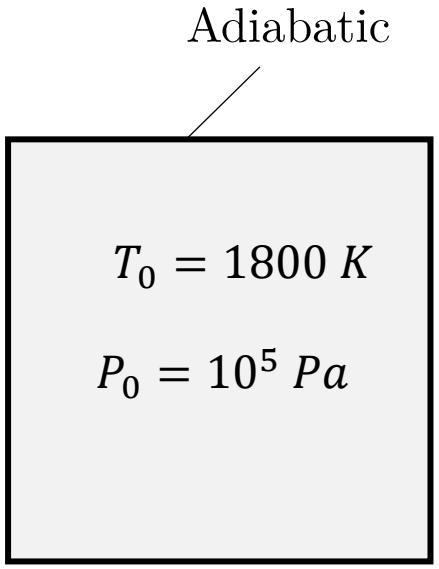
```
template<class ReactionThermo>
void Foam::combustionModels::laminarSoot<ReactionThermo>::updateSoot()
{
    const surfaceScalarField& phi = this->phi();
    const volScalarField D(diffusionCoeff());
    volScalarField rho (this->thermo().rho());
    fv::options& fvOptions(fv::options::New(this->mesh_));

    // N_agg Equation
    {
        Info<< "N_agg Equation \n" << endl;
        tmp<fvScalarMatrix> N_aggEqn
        (
            fvm::ddt(rho, N_agg_)
            + fvm::div(phi, N_agg_)
            - fvm::laplacian(D*rho, N_agg_)
        ==
            - fvm::Sp(rho * S_coag_N_agg_ /N_agg_, N_agg_)
            + rho * S_inc_N_
        );

        N_aggEqn.ref().relax();
        fvOptions.constrain(N_aggEqn.ref());
        solve(N_aggEqn);
        fvOptions.correct(N_agg_);
    }
...
```

$$\frac{\partial}{\partial t}(\rho N_{agg}) + \nabla \cdot (\rho u N_{agg}) + \nabla^2(\rho D N_{agg}) = \rho \left(S^N_{inc} + S^N_{coag}\right)$$

# Validation




$Y_{C_2H_4} = 0.2$

$Y_{N_2} = 0.8$

$N_{agg} = N_{pri} \approx 0$

$C_{tot} \approx 0$

$H_{tot} \approx 0$

**OpenFOAM**

reactingFoam + laminarSoot

**Cantera + Python**

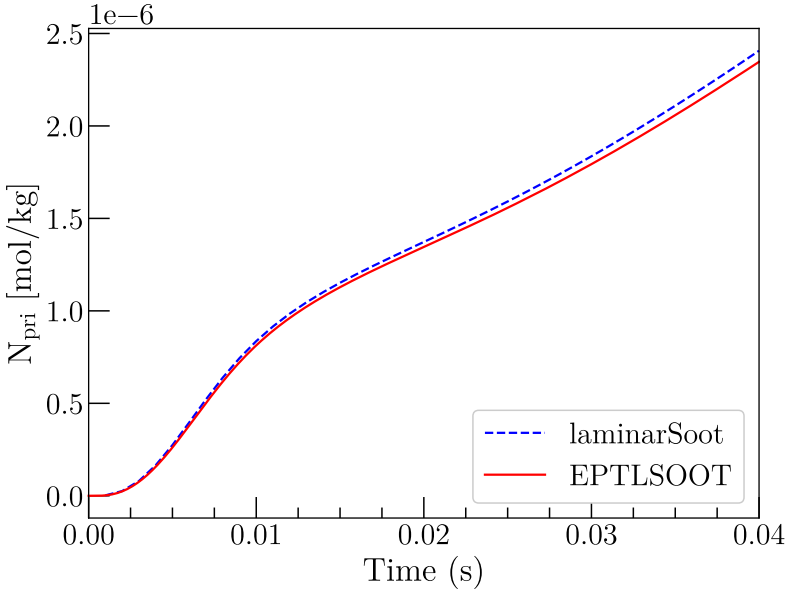
EPTLSOOT

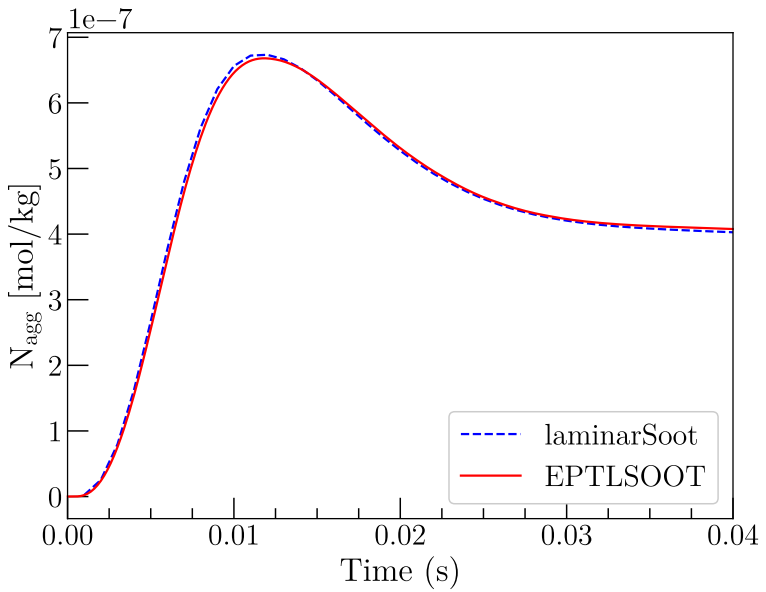
Kinetic solver and soot model for flames and low dimensional reactors

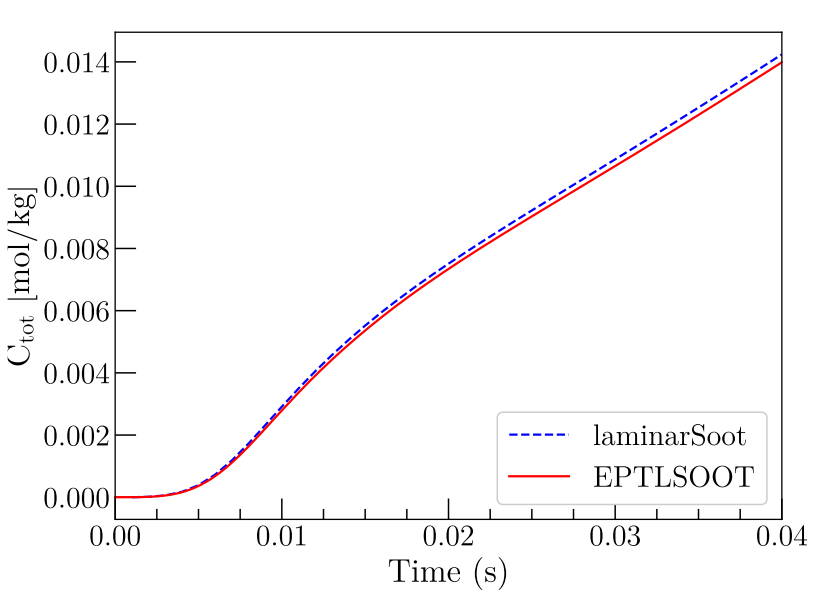
**ABF mechanism [1]**

101 species

544 reactions





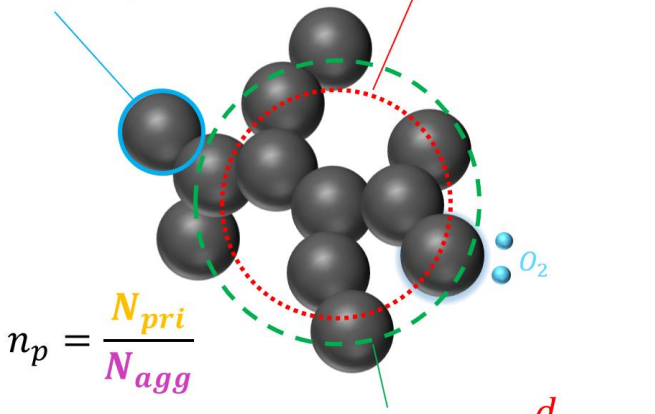

[1] Appel, J., Bockhorn, H. and Frenklach, M., 2000. Kinetic modeling of soot formation with detailed chemistry and physics: laminar premixed flames of C2 hydrocarbons. Combust. Flame

# Summary and Outlook

A monodisperse population balance model is implemented in `laminarSoot` model

$$d_p = \left(\frac{6}{\pi}\frac{C_{tot} W_{carbon}}{\rho_{soot} N_{pri} Av}\right)^{1/3}$$

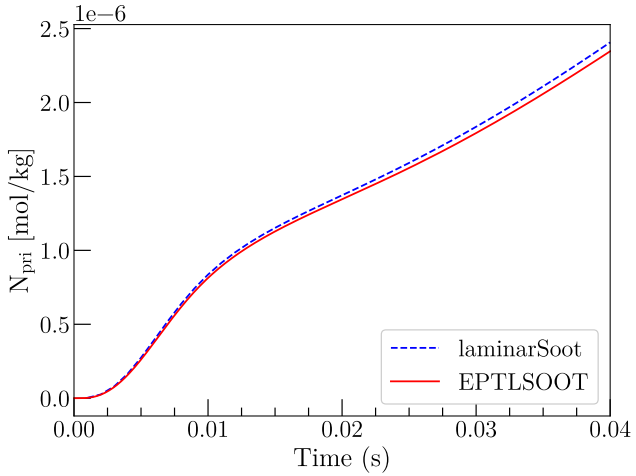
$$d_m = d_p\, n_p^{0.45}$$

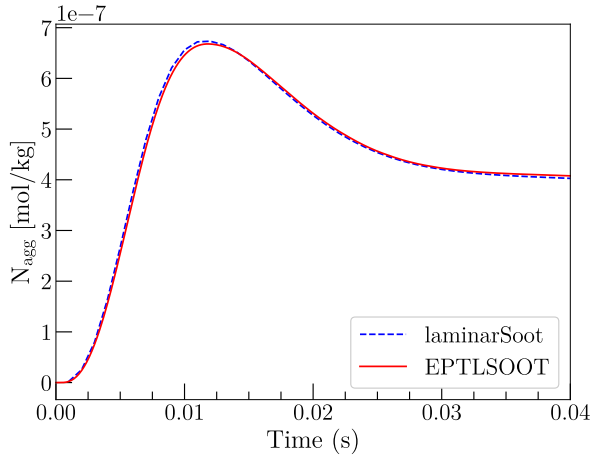



$$n_p = \frac{N_{pri}}{N_{agg}}$$

$$d_g = \frac{d_m}{n_p^{-0.2} + 0.4}$$

The model is validated against existing code for a zero dimensional reactor




More practical targets are required such burners stabilized premixed flames and counterflow diffusion flames