Implementation of a Sectional Population Balance Model (SPBM) in laminar combustion model

Sina Kazemi

PhD student

Mechanical and Aerospace Department,

Carleton University

Prepared for:

CFD with OpenSource Software course

At Chalmers University of Technology

What is soot

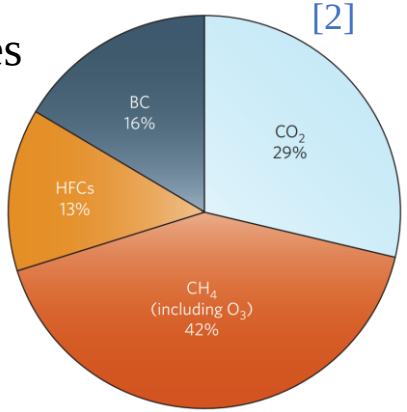
Soot as air pollution

[1]

- Released from industries and cars
- Around 10 mega tons per year
- One of the main sources

of air pollution

[2]



[3]

Carbon Black (CB)

- Exact structure as soot
- Industrially produced
- Various industrial applications:

  Lithium batteries, tire, paint, ink, etc.
- Most valuable flame-made nanoparticle ($ 17B / year)

[1] https://phys.org/news/2017-02-darkness-soot-air-pollution.html
[2] Soot and short-lived pollutants provide political opportunity

[3] https://bonlineen.cloneheads.net/category?name=carbon%20black%20uses

Soot formation

In the presence of oxidative agents, particles undergo oxidation leading to size reduction and potential fragmentation

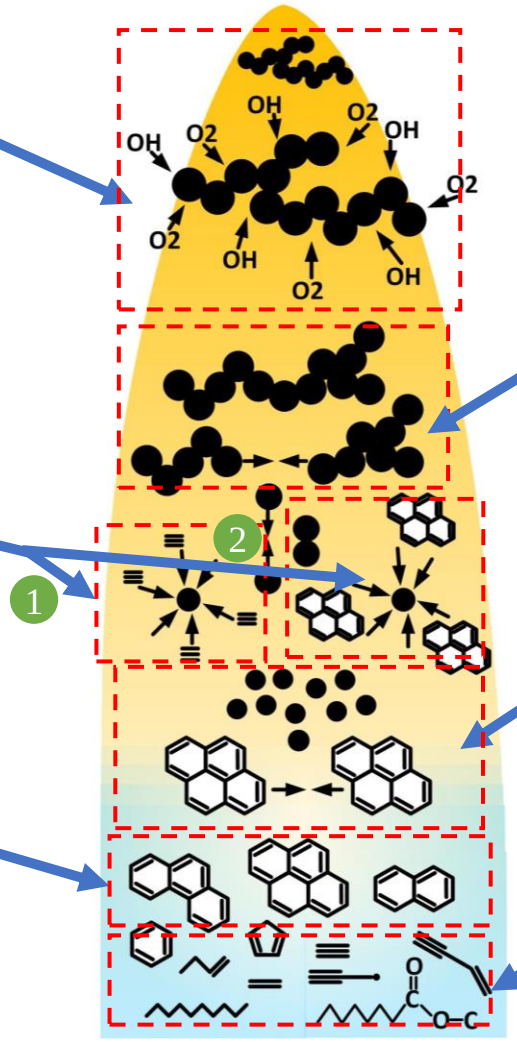
6

The size of particles increases by two surface growth mechanisms: 1) HACA, 2) PAH adsorption

4

Soot precursors collide and PAHs are formed. PAHs grow by HACA

2



[4]

Agglomeration happens and particles attached together. This makes morphology of particles complex

5

PAHs collide and premature soot particles (dimmers) are formed. The size range is 1-3 nm.

3

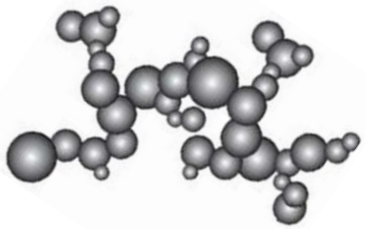
1 Formation of soot precursors (like Methane, Acetylene, etc.) in the gas phase via chemical reactions

[4] https://thomsonlab.mie.utoronto.ca/detailed-and-fundamental-modeling-of-soot-formation/

Soot Modelling

Modelling of soot

Morphology of particles

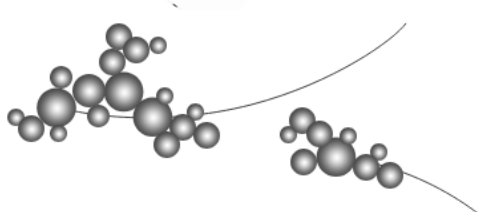


Carbon Black properties

Mitigating air pollution

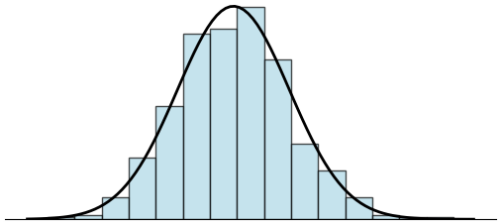
Computational cost / accuracy

3 Discrete Element Modeling (DEM)



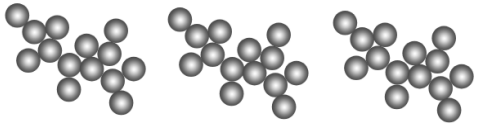
Tracking all individual particle make it impossible for practical scenarios

2 Sectional Population Balance Model (SPBM)



Considering PSD. Accurate and feasible

1 Monodisperse Population Balance Model (MPBM)



Assuming all particles to be identical, not very accurate

What is Population Balance Models

Eulerian description of particles:

1 Agglomerate number density

$$\frac{\partial}{\partial t}(\rho N_{agg}) + \nabla \cdot (\rho u N_{agg}) + \nabla^2 \cdot (\rho D N_{agg}) = \rho(S_{agg})$$

2 Primary particle number density

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

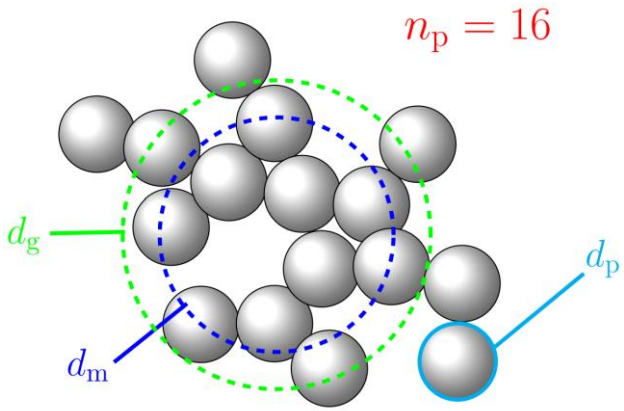
3 Total carbon content

$$\frac{\partial}{\partial t}(\rho C_{tot}) + \nabla \cdot (\rho u C_{tot}) + \nabla^2 \cdot (\rho D C_{tot}) = \rho(S_C)$$

Any properties of interest

Source terms

Inception | Surface growth | Coagulation | Oxidation



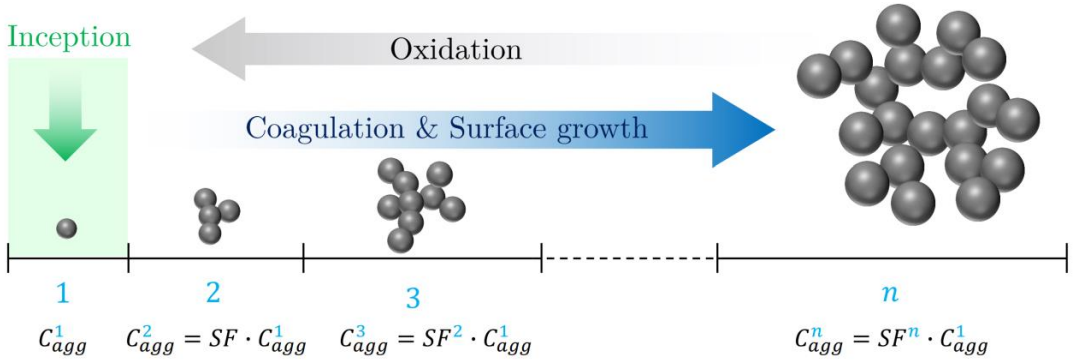
Particle Morphology

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

$$d_m = d_p n_p^{0.45} \qquad d_g = \frac{dm}{n_p^{-0.2} + 0.4}$$

They all depend on the particle morphology and the gas phase properties

Sectional Population Balance Model (SPBM)



A set of equations for each section should be solved

Inception

From chemistry

$$\left(S_{agg}\right)_{inc} = \left(S_{pri}\right)_{inc} = \frac{I_{inc}}{C^1_{agg}} \times \frac{1}{Av}$$

Surface growth

From chemistry

$$\left(S^i_{agg}\right)_{sg} = \frac{1}{Av} \times \begin{cases} -\dfrac{I_{sg,1}}{C^2_{agg} - C^1_{agg}}, & i = 1 \\ \dfrac{I_{sg,i-1}}{C^i_{agg} - C^{i-1}_{agg}} - \dfrac{I_{sg,i}}{C^{i+1}_{agg} - C^i_{agg}}, & i = 2, \dots, n-1 \\ \dfrac{I_{sg,MS-1}}{C^n_{agg} - C^{n-1}_{agg}}, & i = n \end{cases}$$

$$\left(S^i_{pri}\right)_{sg} = \frac{1}{Av} \times \begin{cases} -\dfrac{I_{sg,1}}{C^2_{agg} - C^1_{agg}}, & i = 1 \\ \dfrac{I_{sg,i-1}}{C^i_{agg} - C^{i-1}_{agg}} n_{p,i-1} - \dfrac{I_{sg,i}}{C^{i+1}_{agg} - C^i_{agg}} n_{p,i}, & i = 2, \dots, n-1 \\ \dfrac{I_{sg,MS-1}}{C^n_{agg} - C^{n-1}_{agg}} n_{p,i-1}, & i = n \end{cases}$$

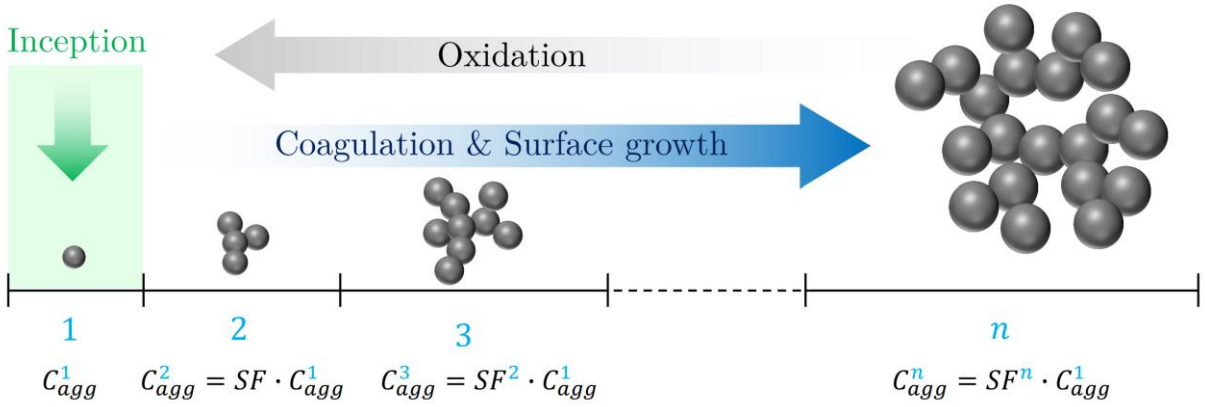
Oxidation

From chemistry

$$\left(S^i_{agg}\right)_{ox} = \frac{1}{Av} \times \begin{cases} \dfrac{I_{ox,2}}{C^2_{agg} - C^1_{agg}} - \dfrac{I_{ox,1}}{C^1_{agg}}, & i = 1 \\ \dfrac{I_{ox,i+1}}{C^{i+1}_{agg} - C^i_{agg}} - \dfrac{I_{ox,i}}{C^i_{agg} - C^{i-1}_{agg}}, & i = 2, \dots, n-1 \\ \dfrac{I_{ox,n}}{C^{n-1}_{agg} - C^n_{agg}}, & i = n \end{cases}$$

$$\left(S^i_{agg}\right)_{ox} = \frac{1}{Av} \times \begin{cases} \dfrac{I_{ox,2}}{C^2_{agg} - C^1_{agg}} n_{p,2} - \dfrac{I_{ox,1}}{C^1_{agg}}, & i = 1 \\ \dfrac{I_{ox,i+1}}{C^{i+1}_{agg} - C^i_{agg}} n_{p,i+1} - \dfrac{I_{ox,i}}{C^i_{agg} - C^{i-1}_{agg}} n_{p,1}, & i = 2, \dots, n-1 \\ \dfrac{I_{ox,n}}{C^{n-1}_{agg} - C^n_{agg}} n_{p,n}, & i = n \end{cases}$$

Sectional Population Balance Model (SPBM)



$$\eta_{ijk} = \begin{cases} \dfrac{C_{agg}^{i+1} - \left(C_{agg}^{j} + C_{agg}^{k}\right)}{C_{agg}^{i+1} - C_{agg}^{i}}, & if\, C_{agg}^{i} \le C_{agg}^{j} + C_{agg}^{k} < C_{agg}^{i+1} \\ \dfrac{C_{agg}^{i-1} - \left(C_{agg}^{j} + C_{agg}^{k}\right)}{C_{agg}^{i-1} - C_{agg}^{i}}, & if\, C_{agg}^{i-1} \le C_{agg}^{j} + C_{agg}^{k} < C_{agg}^{i} \\ 0, & else \end{cases}$$

Coagulation

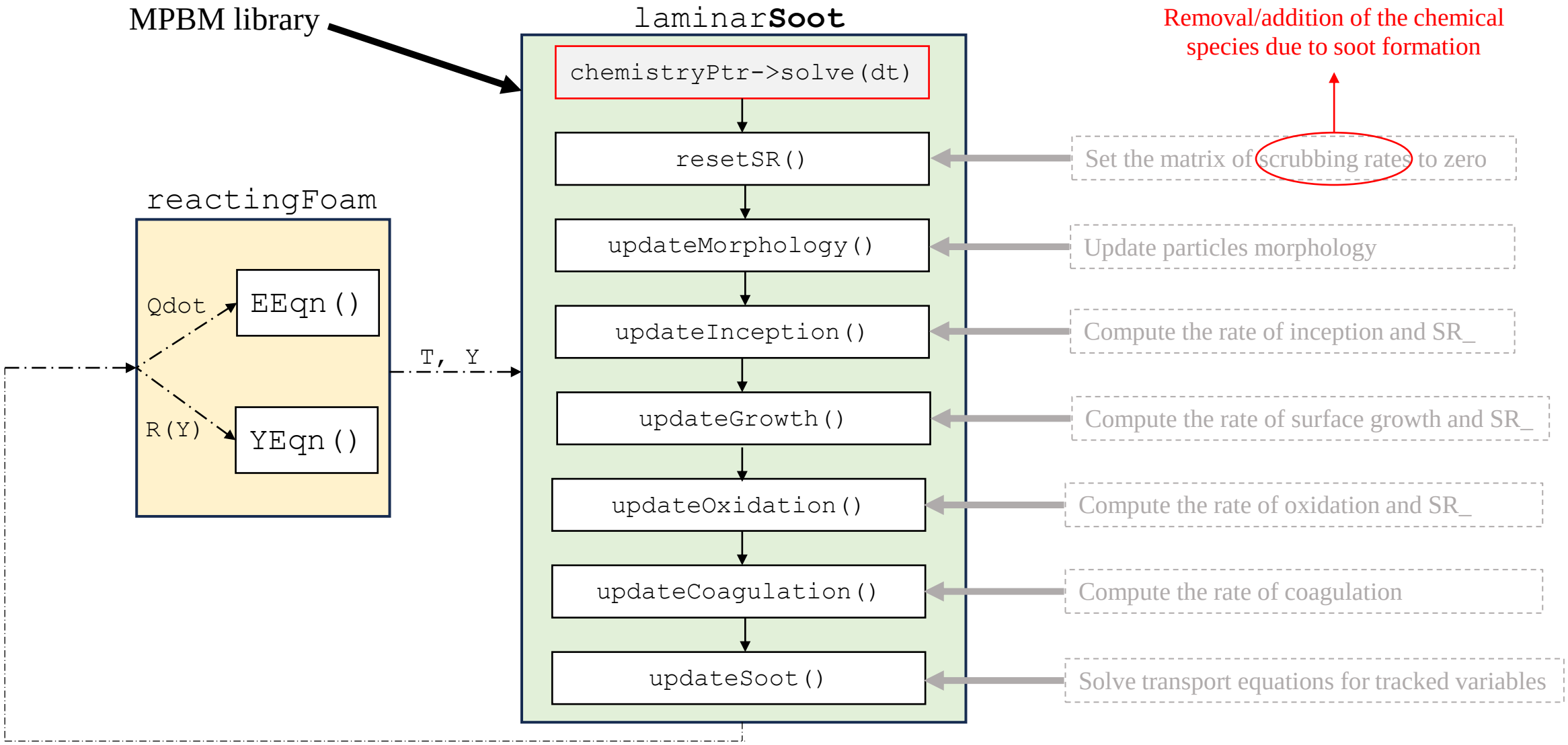
$$\left(S_{agg}^{i}\right)_{coag} = \frac{1}{Av} \times \sum_j \sum_k \left(1 - \frac{\delta_{jk}}{2}\right) \eta_{ijk} \beta_{jk} \xi_{jk} N_j^{agg} N_k^{agg} - \frac{1}{Av} \times N_i^{agg} \sum_{m=1}^{MS} \beta_{im} \xi_{im} N_m^{agg}$$

$$\left(S_{pri}^{i}\right)_{coag} = \frac{1}{Av} \times \sum_j \sum_k \left(1 - \frac{\delta_{jk}}{2}\right) \eta_{p,ijk} \eta_{ijk} \beta_{jk} \xi_{jk} N_j^{agg} N_k^{agg} - \frac{1}{Av} \times N_i^{pri} \sum_{m=1}^{MS} \beta_{im} \xi_{im} N_m^{agg}$$

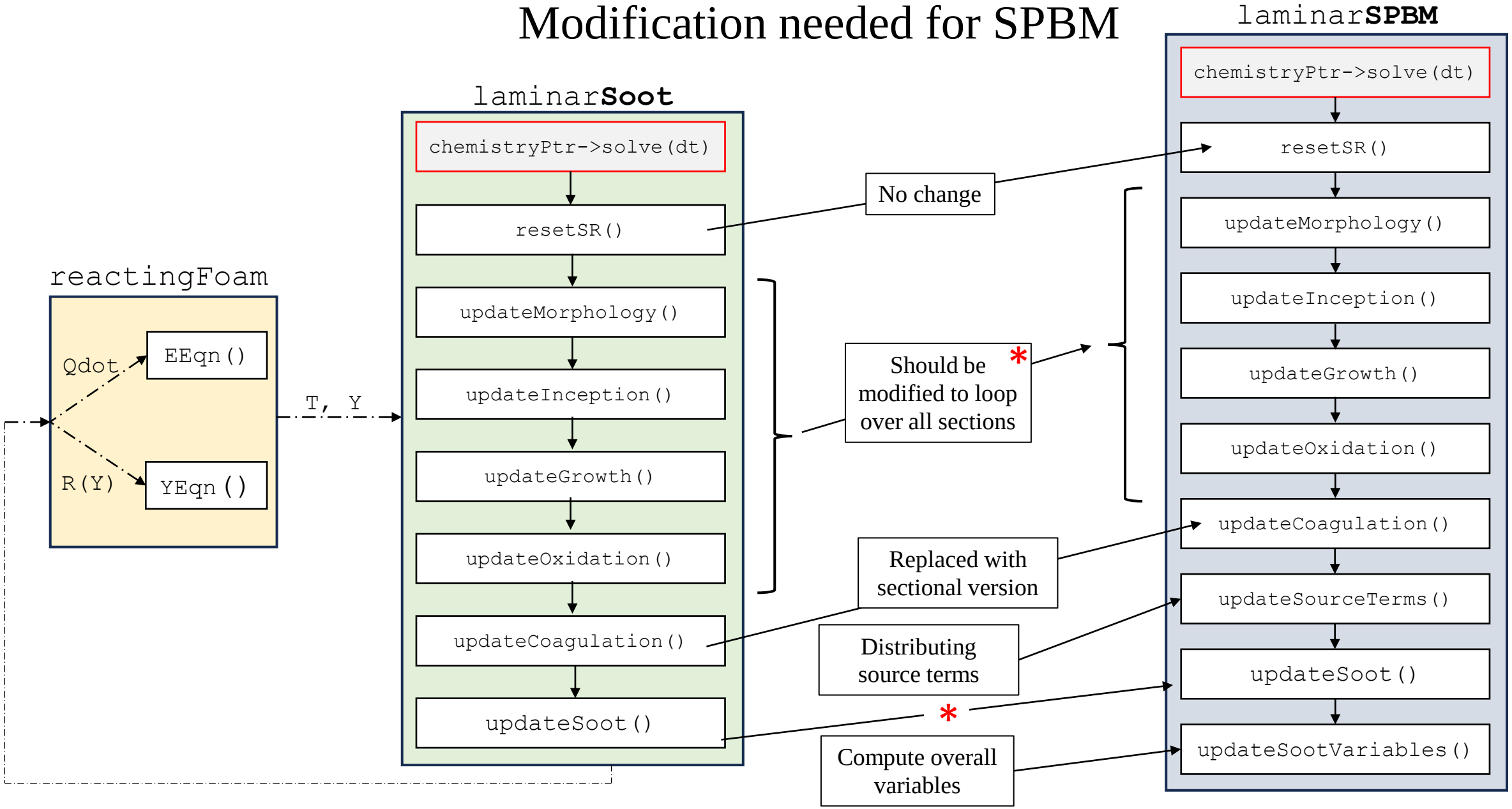
$$\eta_{p,ijk} = \frac{C_{agg}^{i}}{C_{agg}^{j} + C_{agg}^{k}} \left(n_{p,j} + n_{p,k}\right)$$

Collision frequency, here it is a fixed value!

What is available: `laminarSoot`



Modification needed for SPBM



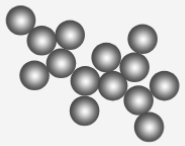
Implementation—Defining Fields

laminarSPBM.H

```
113  // Soot tracked fields - sectional
114  PtrList<volScalarField> N_agg_sec_;
115  PtrList<volScalarField> N_pri_sec_;
116  PtrList<volScalarField> C_tot_sec_;
117
118  // Morphology - sectional
119  PtrList<volScalarField> n_p_sec_;
120  PtrList<volScalarField> m_agg_sec_;
121  PtrList<volScalarField> d_p_sec_;
122  PtrList<volScalarField> d_m_sec_;
123  PtrList<volScalarField> d_g_sec_;
124  PtrList<volScalarField> A_tot_sec_;
125
126  // Source Terms - sectional
127  // Inception
128  volScalarField I_inc_C_tot_;
129  // Surface growth
130  PtrList<volScalarField> I_grow_C_tot_sec_;
131  // Oxidation
132  PtrList<volScalarField> I_ox_C_tot_sec_;
133  // Coagulation
134  PtrList<volScalarField> I_coag_N_agg_sec_;
135  PtrList<volScalarField> I_coag_N_pri_sec_;
136  volScalarField sum1_inside_; //for coagulation calculations
137  volScalarField sum2_all_; //for coagulation calculations
138
139  // Source terms
140  PtrList<volScalarField> S_N_agg_sec_;
141  PtrList<volScalarField> S_N_pri_sec_;
142
143  // Total variables
144  volScalarField N_agg_;
145  volScalarField N_pri_;
146  volScalarField C_tot_;
147  volScalarField d_m_;
```

$N_{agg}, N_{pri}, C_{tot}$ For each section

Morphological variables
For each section



Inception

Surface growth and oxidation
for each section

Coagulation for each section

Source terms for each section

Overall variables

laminarSPBM.C

```
...
C_agg_sec(n_secs_.value()),
secNum(n_secs_.value()),
...
// Fields for tracked variables
N_agg_sec(n_secs_.value()),
N_pri_sec(n_secs_.value()),
C_tot_sec(n_secs_.value()),
...
// Fields for soot morphology
n_p_sec(n_secs_.value()),
m_agg_sec(n_secs_.value()),
d_p_sec(n_secs_.value()),
d_m_sec(n_secs_.value()),
d_g_sec(n_secs_.value()),
A_tot_sec(n_secs_.value()),
...
// Fields for source terms
I_grow_C_tot_sec(n_secs_.value()),
I_ox_C_tot_sec(n_secs_.value()),
I_coag_N_agg_sec(n_secs_.value()),
I_coag_N_pri_sec(n_secs_.value()),
...
// Fields for overall source terms
S_N_agg_sec(n_secs_.value()),
S_N_pri_sec(n_secs_.value()),
...
```

laminarSPBM.H

```
// Number of sections
Foam::dimensionedScalar n_secs_;
// Spacing factor of sections
Foam::dimensionedScalar spacing_;
// Carbon per agglomerate in mole/#
PtrList<volScalarField> C_agg_sec_;
```

By user

By user

laminarSPBM.C

```
529 // Building sections
530 template<class ReactionThermo>
531 void Foam::combustionModels::laminarSPBM<ReactionThermo>::build_C_agg_sec()
532 {
533     forAll(secNum_, sec)
534     {
535         C_agg_sec_[sec] = C_min_ / Av_ * pow(spacing_, sec);
536     }
537 }
```

Implementation—Defining Fields

createSectionalFields.H

```
39  tmp<volScalarField> tNPridefault;
40  tmp<volScalarField> tNAggdefault;
41
42  secNum_ = 0;
43
44  forAll(secNum_, sec)
45  {
46
47      C_agg_sec_.set
48          (
49              sec,
50              new volScalarField
51              (
52                  IOobject
53                  (
54                      "C_agg_sec" + std::to_string(sec),
55                      this->mesh().time().timeName(),
56                      this->mesh(),
57                      IOobject::NO_READ,
58                      IOobject::AUTO_WRITE
59                  ),
60                  this->mesh(),dimensionedScalar("C_agg_sec" + std::to_string(sec), dimensionSet
(0,0,0,0,1,0,0),0.0)
61              )
62      );
63
64      IOobject timeOAggIO
65      (
66              IOobject::groupName("N_agg", ""),
67              this->mesh().time().timeName(0),
68              this->mesh(),
69              IOobject::MUST_READ,
70              IOobject::NO_WRITE
71      );
72
73      tNAggdefault = new volScalarField(timeOAggIO, this->mesh());
74
75      N_agg_sec_.set
76      (
77          sec,
78          new volScalarField
79          (
80              IOobject
81              (
82                  IOobject::groupName( "N_agg_sec" + std::to_string(sec), ""),
83                  this->mesh().time().timeName(),
84                  this->mesh(),
85                  IOobject::NO_READ,
86                  IOobject::AUTO_WRITE
87              ),
88              tNAggdefault()
89          )
90      );
```

Looping over all sections

Creating a series of a fields that do not need B.C.
The name of the fields is `C_agg_sec_i`, where `i` is the section index

Creating a series of a fields that need B.C. (they are computed by solving the transport equations)
By defining a group name, there will be no need to specify B.C. for that field in each section.
It works in a same way as mass fraction fields:

src/thermophysicalModels/reactionThermo/mixtures/basicMultiComponentMixture.C

```
// Read Ydefault if not already read
if (!tYdefault.valid())
{
    word YdefaultName(IOobject::groupName("Ydefault", phaseName));
```

```
Y_.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            IOobject::groupName(species_[i], phaseName),
            mesh.time().timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        tYdefault()
    )
);
```

Implementation—`updateMorphology()` function

laminarSPBM.C

```
641 // Updating morphology - Sectional
642 template<class ReactionThermo>
643 void Foam::combustionModels::laminarSPBM<ReactionThermo>::updateMorphology()
644 {
645     forAll(secNum_, i)
646     {
647         //  Mobility diameter
648         d_m_sec_[i] = max(d_p_sec_[i], d_p_sec_[i] * pow(n_p_sec_[i], 0.45));
649
650         // Gyration Diameter
651         volScalarField n_p_lowerlimit (n_p_sec_[i]*0.0+1.5);
652         d_g_sec_[i] = (n_p_sec_[i] <= n_p_lowerlimit) * (d_m_sec_[i] / 1.29) + \
653         (n_p_sec_[i] > n_p_lowerlimit) * (d_m_sec_[i] / (pow(n_p_sec_[i], -0.2)+0.4));
654
655         // Surface area of each primary particle
656         A_tot_sec_[i] = N_pri_sec_[i] * Av_ * pi_ * d_p_sec_[i] * d_p_sec_[i];
657
658         // Primary particle diameter
659         d_p_sec_[i] = pow ((6.0 / pi_) * (C_agg_sec_[i] * W_carbon_) / (rho_soot_ * n_p_sec_[i]),
    1.0/3.0);
660
661         // number of primary particles
662         n_p_sec_[i] = min(max(N_pri_sec_[i] / N_agg_sec_[i], 1.0), C_agg_sec_[i] / C_agg_sec_[0]);
663
664         // mass of each agglomerate
665         m_agg_sec_[i] = rho_soot_ * n_p_sec_[i] * (pi_ / 6.0) * pow(d_p_sec_[i], 3.0);
666     }
667
668 }
```

Looping over all sections

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

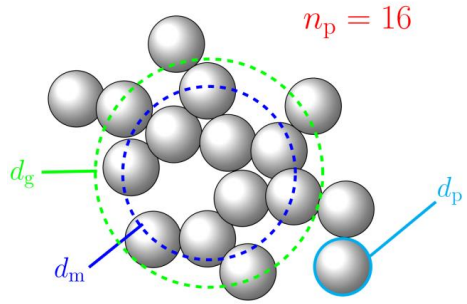
$$d_g = \frac{dm}{n_p^{-0.2} + 0.4}$$

$$A_{tot} = N_{pri} Av\pi\, d_p^2$$

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

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

$$m_{agg} = \frac{\rho_{soot} n_p \pi d_p^3}{6}$$



Implementation—Make functions size dependent

Class functions to be size dependent

laminarSPBM.C

```
761 template<class ReactionThermo>
762 void Foam::combustionModels::laminarSPBM<ReactionThermo>::updateOxidation()
763 {
764     forAll(secNum_, sec)
765     {
766         I_ox_C_tot_sec_[sec] *= 0.0;
767         if (HACA_oxidation_enabled_){
768         volScalarField rho(this->thermo().rho());
769         volScalarField HACAO2OxidationRateField(HACAO2OxidationRate(sec));
770         volScalarField HACAOHOxidationRateField(HACAOHOxidationRate(sec));
771         I_ox_C_tot_sec_[sec] += -1 * (HACAO2OxidationRateField + HACAOHOxidationRateField) * rho;
772
773         if (scrubbing_enabled_){
774             // O2
775             label O2_id = speciesIds_["O2"];
776             label O2_index = speciesIndicies_["O2"];
777             SR_[O2_id] -= 0.5 * HACAO2OxidationRateField * W(O2_index);
778
779             // CO2
780             label CO_id = speciesIds_["CO"];
781             label CO_index = speciesIndicies_["CO"];
782             SR_[CO_id] += (HACAO2OxidationRateField + HACAOHOxidationRateField) * W(CO_index);
783
784             // OH
785             label OH_id = speciesIds_["OH"];
786             label OH_index = speciesIndicies_["OH"];
787             SR_[OH_id] -= HACAOHOxidationRateField * W(OH_index);
788         }
789         }
790     }
791 }
```

Looping over all sections

```
virtual tmp<volScalarField> HACAO2OxidationRate(label sec)
{
    label O2_i = speciesIndicies_["O2"];
    volScalarField rho (this->thermo().rho());
    return tmp<volScalarField>
    (
        new volScalarField
        (
            max
            (
                2 * alpha(sec) * k_5_HACA() * C(O2_i) * C_soot_0(sec),
                dimensionedScalar(dimensionSet(0,-3,-1,0,1,0,0), scalar(0.0))
            )
        )
    );
}
```

```
virtual tmp<volScalarField> HACAOHOxidationRate(label sec)
{
    label OH_i = speciesIndicies_["OH"];
    volScalarField rho (this->thermo().rho());
    return tmp<volScalarField>
    (
        new volScalarField
        (
            max
            (
                k_6_HACA() * C(OH_i) * N_agg_sec_[sec] * rho,  // Mo
                dimensionedScalar(dimensionSet(0,-3,-1,0,1,0,0), scalar(0.0))
            )
        )
    );
}
```

No modifications for scrubbing part of the functions

Implementation—`updateCoagulation()` function

```
839             // Addition of particles to section
840             for (int k = 0; k <= sec; k++)
841             {
842                 for (int j = k; j <= sec; j++)
843                 {
844                     volScalarField C_agg_sec_jk (C_agg_sec_[j] + C_agg_sec_[k]);
845                     if (C_agg_sec_prev <= C_agg_sec_jk && C_agg_sec_jk <= C_agg_sec_next)
846                     {
847                         // Calculating eth_ijk
848                         if (C_agg_sec_curr < C_agg_sec_jk && C_agg_sec_jk < C_agg_sec_next)
849                         {
850                             eta_ijk = (C_agg_sec_next-C_agg_sec_jk)/(C_agg_sec_next-C_agg_sec_curr);
851                         }
852                         else if (C_agg_sec_prev < C_agg_sec_jk && C_agg_sec_jk < C_agg_sec_curr)
853                         {
854                             eta_ijk = (C_agg_sec_prev-C_agg_sec_jk)/(C_agg_sec_prev-C_agg_sec_curr);
855                         }
856
857                         // Calculation eta_p_ijk (NO MERGING)
858                         eta_p_ijk = C_agg_sec_curr / C_agg_sec_jk * (n_p_sec_[j] + n_p_sec_[k]);
859
860                         // Corresponds to 1-delta(j,k)/2
861                         if (j==k)
862                         {
863                             coag_prefactor = 0.5;
864                         }
865                         else
866                         {
867                             coag_prefactor = 1.0;
868                         }
869
870                         sum1_inside_ = coag_prefactor * eta_ijk * beta * N_agg_sec_[j] * N_agg_sec_[k
    ];
871                         sum1_N_agg += sum1_inside_;
872                         sum1_N_pri += sum1_inside_ * eta_p_ijk;
873                     }
874
875                 }
876             }
```

For all pairs of sections

MPBM

Coagulation doesn't impact N_{pri}!

$$(S_{pri})_{coag} = 0$$

The source term for N_{agg} due to coagulation is:

$$(S_{agg})_{coag} = -\frac{1}{2}\beta N_{agg}^2$$

SPBM

The source term for N_{pri} due to coagulation is:

$$(S_{pri}^i)_{coag} = \frac{1}{Av} \times \sum_j \sum_k \left(1 - \frac{\delta_{jk}}{2}\right) \eta_{p,ijk} \eta_{ijk} \beta_{jk} \xi_{jk} N_j^{agg} N_k^{agg} - \frac{1}{Av} \times N_i^{pri} \sum_{m=1}^{MS} \beta_{im} \xi_{im} N_m^{agg}$$

The source term for N_{agg} due to coagulation is:

$$(S_{agg}^i)_{coag} = \frac{1}{Av} \times \sum_j \sum_k \left(1 - \frac{\delta_{jk}}{2}\right) \eta_{ijk} \beta_{jk} \xi_{jk} N_j^{agg} N_k^{agg} - \frac{1}{Av} \times N_i^{agg} \sum_{m=1}^{MS} \beta_{im} \xi_{im} N_m^{agg}$$

Implementation—`updateSourceTerms()` function

```
911     forAll(secNum_, sec)
912     {
913         S_N_agg_sec_[sec] *= 0.0;
914         S_N_pri_sec_[sec] *= 0.0;
915
916         // First section
917         if(sec==0)
918         {
919             // Inception
920             S_N_agg_sec_[sec] += I_inc_C_tot_ / C_agg_sec_[0] / Av_;
921             S_N_pri_sec_[sec] += I_inc_C_tot_ / C_agg_sec_[0] / Av_;
922
923             // PAH adsorption & surface growth
924             S_N_agg_sec_[sec] += - I_grow_C_tot_sec_[0] / (C_agg_sec_[1] - C_agg_sec_[0]) / Av_;
925             S_N_pri_sec_[sec] += - I_grow_C_tot_sec_[0] / (C_agg_sec_[1] - C_agg_sec_[0]) / Av_;
926
927             // Oxidation
928             S_N_agg_sec_[sec] += (I_ox_C_tot_sec_[1] / (C_agg_sec_[1] - C_agg_sec_[0]) \
929             - I_ox_C_tot_sec_[0] / C_agg_sec_[0]) / Av_;
930             S_N_pri_sec_[sec] += (I_ox_C_tot_sec_[1] / (C_agg_sec_[1] - C_agg_sec_[0]) * n_p_sec_[1] \
931             - I_ox_C_tot_sec_[0] / C_agg_sec_[0]) / Av_;
932
933             // Coagulation
934             S_N_agg_sec_[sec] += I_coag_N_agg_sec_[0];
935             S_N_pri_sec_[sec] += I_coag_N_pri_sec_[0];
936         }
```

Looping over all sections

Reset both source terms at each time step

Only for the first section

It incudes both HACA and PAH

Almost similar piece of code for other sections:

```
// Middle sections
else if (sec > 0 && sec < (this->n_secs_.value() - 1))
{
```

```
// Last section
else if (sec == this->n_secs_.value() - 1)
{
```

Implementation—`updateSoot()` function

```
985     forAll(secNum_, sec)
986     {
987         const volScalarField D(diffusionCoeff(sec));
988         // N_agg_ Equation
989         {
990         Info<< "N_agg Equation for section " << sec << endl;
991         volScalarField& N_agg = N_agg_sec_[sec];
992         fvScalarMatrix N_aggEqn
993         (
994             fvm::ddt(rho, N_agg)
995             + fvm::div(phi, N_agg)
996             - fvm::laplacian(D*rho, N_agg)
997             ==
998             rho * S_N_agg_sec_[sec]
999         );
1000
1001        N_aggEqn.relax();
1002        fvOptions.constrain(N_aggEqn);
1003        N_aggEqn.solve(this->mesh().solver("N_agg"));
1004        fvOptions.correct(N_agg);
1005        }
1006
```

Looping over all sections

Pointer to a volScalarField

Similar equation for N_{pri}!

```
1026        // C_tot Equation
1027        {
1028            Info<< "C_tot Equation for section " << sec << endl;
1029            C_tot_sec_[sec] = N_agg_sec_[sec] * Av_ * C_agg_sec_[sec];
1030        }
```

Function `updateSootVariables()` determines overall values:

- Number-based arithmetic averaging for intensive properties
- Summation for extensive variables

laminarSPBM.C

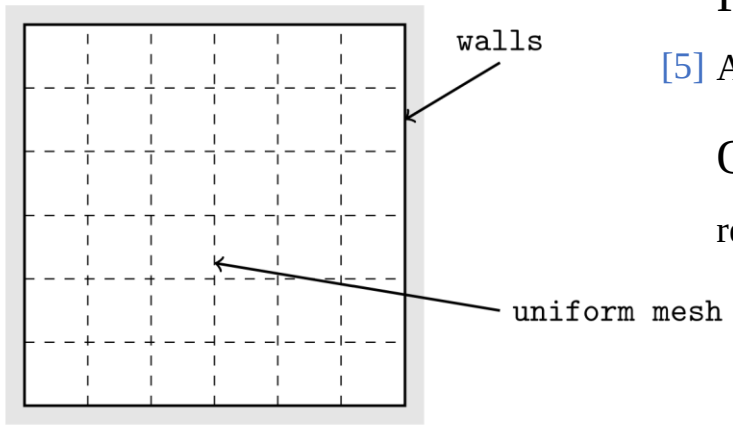
```
1036 template<class ReactionThermo>
1037 void Foam::combustionModels::laminarSPBM<ReactionThermo>::updateSootVariables()
1038 {
1039     N_agg_ *= 0.0;
1040     N_pri_ *= 0.0;
1041     C_tot_ *= 0.0;
1042     volScalarField d_m_sum (d_m_*N_agg_*0.0);
1043
1044     forAll(secNum_, sec)
1045     {
1046         N_agg_ += N_agg_sec_[sec];
1047         N_pri_ += N_pri_sec_[sec];
1048         C_tot_ += C_tot_sec_[sec];
1049         d_m_sum += N_agg_sec_[sec] * d_m_sec_[sec];
1050     }
1051
1052     d_m_ = d_m_sum / N_agg_;
1053 }
```

Despite MPBM, in SPBM C_{tot} is computed using an algebraic equation:

$$C_{tot} = N_{agg} Av C_{agg}$$

Tutorial case

Constant volume reactor



Reaction mechanism:

[5] ABF: 101 species, 544 reactions

OpenFOAM solver:

reactingFOAM

Combustion model:

laminarSPBM

Size sections:

Number of sections: 40
Progression factor, SF: 1.5

```
    class       dictionary;
    location    "constant";
    object      sootProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * *

PAHs (A2 A3 A4);

numberOfSections    40;
spacingFactor       1.5;

scrubbing_enabled   true;
PAH_growth_enabled  true;
HACA_growth_enabled true;
inception_enabled   true;
HACA_oxidation_enabled true;
coagulation_enabled true;
```

Initial/boundary conditions

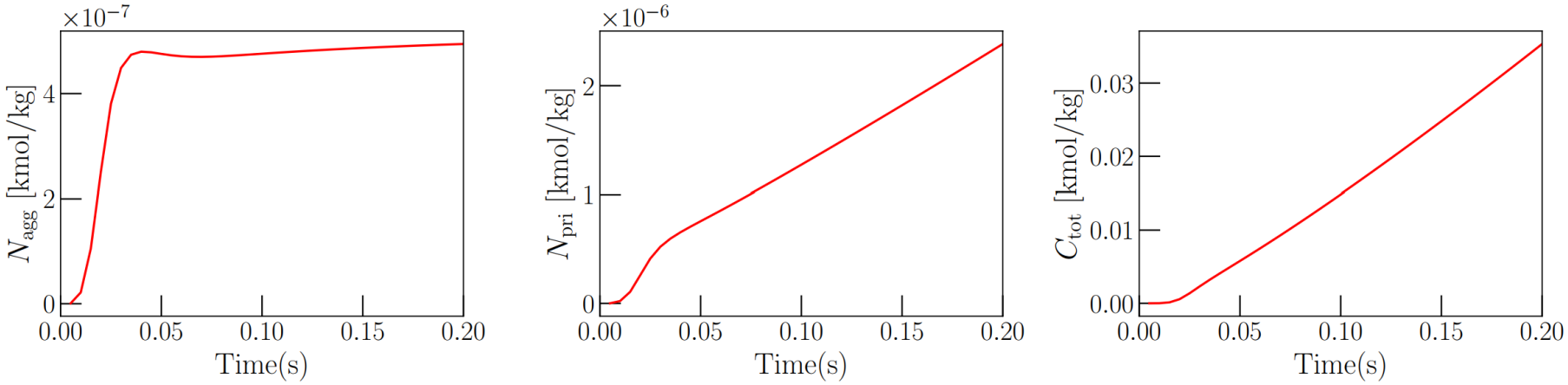
Field	Initial value	Boundary condition
U	(0 0 0)	zeroGradient
P	1.0e5	zeroGradient
T	1800	zeroGradient
N2	0.7	zeroGradient
CH4	0.3	zeroGradient
Ydefault	0	zeroGradient

Preprocessing Python script for setting up the probe function object

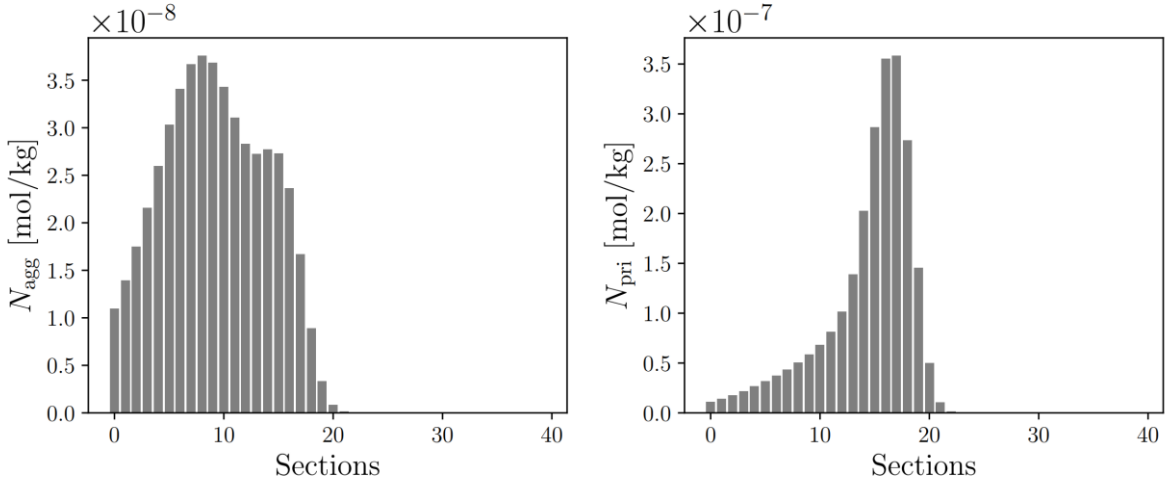
```
probes
{
    type            probes;
    libs            (sampling);
    name            probes;
    writeControl    outputTime;
    writeInterval   1;
    interpolationScheme cellPoint;
    sampleOnExecute yes;

    fields          (N_agg N_pri C_tot N_agg_sec0 N_agg_sec1 N_agg_sec2 N_agg_sec3 N_agg_sec4
     N_pri_sec0 N_pri_sec1 N_pri_sec2 N_pri_sec3 N_pri_sec4);

    probeLocations
    (
        (0.05 0.05 0.005)
    );
```

[5] 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

Tutorial case—results



Evolution of various soot variables over time.



Soot particle size distribution at t=0.2 (s)

Future projects

- Implementing commonly used sub-models to provide a more flexibility for the user
- Validating the solver in 2D cases like flames and sprays
- Implementing various collision kernel formulations
- Implementing moving sectional model