POLITECNICO DI TORINO

Collegio di Ingegneria Chimica e dei Materiali

Corso di Laurea Magistrale
in Ingegneria Chimica e dei Processi Sostenibili

Tesi di Laurea Magistrale

Development of a compartment model to simulate the co-precipitation of Nickel-Manganese-Cobalt hydroxides in stirred tank reactors

Relatori

prof. Boccardo Gianluca
prof. Buffo Antonio
prof. Marchisio Daniele
dott. Shiea Mohsen
firma del relatore (dei relatori)

Candidato

Julián Andrés Piscitello
firma del candidato

Marzo 2021

Summary

0. Sommario

I. Introduzione

La Tesi presentata si propone di realizzare un modello a compartimenti per simulare il processo di co-precipitazione dell'idrossido di Nichel-Manganese-Cobalto (NMC) $Ni_xMn_yCo_z(OH)_2$. Questo composto è di interesse in quanto è il precursore del materiale costituente il catodo nelle batterie agli ioni di Litio ($LiNi_xMn_yCo_zO_2$), batterie il cui utilizzo è ormai molto diffuso in un gran numero di settori.

La sintesi di questo ossido è divisa in due step (Barai et al, 2019):

1) co-precipitazione: processo che produce il precursore di cui si è parlato sopra, a partire da soluzioni di solfati di Litio, Manganese e Cobalto in presenza di ammoniaca;

2) calcinazione: il precursore viene sottoposto a ossidazione e litiazione per ottenere $LiNi_xMn_yCo_zO_2$.

Il primo processo è quello trattato in questa Tesi. La reazione di co-precipitazione è un processo molto complesso in cui entrano in gioco diversi meccanismi. La base di lavoro di questa Tesi è l'utilizzo della geometria di un reattore fornita dall'azienda Umicore.

I principali fenomeni che vengono considerati per simulare la reazione di co-precipitazione in questo reattore sono: sistema pressione-velocità dell'acqua presente nel reattore (equazioni di Navier-Stokes), equazioni di trasporto delle specie coinvolte nella co-precipitazione, chimica del sistema ed equazioni di trasporto dei momenti (per le quali servono le cinetiche di precipitazione) (Gavi et al., 2007). La soluzione dei campi di pressione e velocità può essere risolta separatamente dagli altri meccanismi.

La soluzione dei campi di velocità e pressione viene calcolata tramite simulazioni fluidodinamiche utilizzando il software ANSYS Fluent.

Le equazioni di trasporto delle specie vengono definite per i cationi metallici, per l'ammoniaca e per le cariche inerti. Nelle equazioni dei metalli si trova un fattore sorgente che rappresenta la precipitazione di queste specie allo stato solido.

I meccanismi chimici presenti, ipotizzati da van Bommel e Dahn (2009), consistono in una serie di reazioni di equilibrio fra il metallo e l'ammoniaca che lo complessa, e in particolare la reazione grazie a cui avviene la crescita delle particelle è un equilibrio fra l'idrossido metallico e la soluzione di ammoniaca.

Le equazioni di trasporto dei momenti vengono derivate dal bilancio di popolazione (PBE) e risolte utilizzando il metodo di quadratura dei momenti (QMOM), del quale vengono calcolati pesi e nodi con appropriati algoritmi.

Risolvere tutte queste equazioni con ANSYS Fluent significherebbe fare tutti i calcoli per ogni cella del sistema e, dato il grande numero di celle presenti, questo significherebbe alti costi computazionali e tempi di simulazione. Lo scopo della Tesi è quindi quello di risolvere questo problema utilizzando un modello a compartimenti.

Questo è possibile grazie a uno strumento di ANSYS Fluent che permette di dividere il dominio (l'interno del reattore) in zone interconnesse. In questo modo è possibile considerare ognuno di questi compartimenti come un CSTR e risolvere le equazioni che descrivono il fenomeno non più per ogni cella, ma per un numero di gran lunga minore di compartimenti. I calcoli in questi compartimenti vengono svolti grazie all'utilizzo di uno script scritto in linguaggio Python.

La divisione in compartimenti deve essere fatta in modo ponderato, considerando che i parametri controllanti la co-precipitazione siano il più omogenei possibile in ogni compartimento. I parametri che si sono scelti di utilizzare per la suddivisione sono la supersaturazione e la velocità di dissipazione dell'energia cinetica turbolenta.

II. Fluidodinamica

Un fattore importante nel sistema considerato è la fluidodinamica all'interno del reattore. È quindi di fondamentale importanza calcolare il campo di moto nel sistema in esame per poterlo utilizzare successivamente accoppiandolo ai meccanismi di coprecipitazione. Per far ciò è necessario risolvere l'equazione di continuità e l'equazione di Navier-Stokes, tenendo però conto del fatto che all'interno del reattore miscelato le velocità sono tali da generare turbolenza, quindi sfruttando la decomposizione di Reynolds, per cui la velocità può essere vista come la somma di una velocità media e una fluttuante, è possibile ottenere le equazioni di continuità e Navier-Stokes in regime turbolento:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho \overline{u_i})}{\partial x_i} = 0 \tag{II.1}$$

$$\frac{\partial(\rho \overline{u_i})}{\partial t} + \frac{\partial\left(\rho \overline{u_i\, u_j}\right)}{\partial x_j} = -\frac{\partial(\overline{p})}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \overline{u_\iota}}{\partial x_j} + \frac{\partial \overline{u_J}}{\partial x_i}\right)\right) - \rho \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right) + \rho g_i \tag{II.2}$$

La quale non è però un'equazione risolvibile analiticamente, in quanto risente del problema della chiusura. Questo problema viene affrontando risolvendo le equazioni per energia cinetica turbolenta, definita come:

E per dissipazione turbolenta, definita come:

$$k = \frac{1}{2}\operatorname{tr}\left(\langle u_i' u_j' \rangle\right) \tag{II.3}$$

$$\varepsilon = 2\nu \overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} \tag{II.4}$$

Le cui equazioni di bilancio sono ottenibili a partire dall'equazione II.2 e considerando la teoria di Kolmogorov:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k \overline{u_i})}{\partial x_i} = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right) + \mu_t\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\frac{\partial \overline{u_i}}{\partial x_j} - \rho\varepsilon \tag{II.5}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \frac{\partial(\rho \varepsilon \overline{u_i})}{\partial x_i} = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right) + C_1 \mu_t \frac{\varepsilon}{k}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\frac{\partial \overline{u_i}}{\partial x_j} - \rho C_2 \frac{\varepsilon^2}{k} \tag{II.6}$$

In cui la viscosità turbolenta è definita come:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{II.7}$$

Le costanti e numeri di Prandtl turbolenti sono determinati empiricamente. Sono di uso comune i valori mostrati in tabella I.

Tabella I Valori delle costanti del modello k-eps utilizzate in Ansys Fluent

C_1	C_2	C_μ	σ_ε	σ_k
1.44	1.92	0.09	1.0	1.3

III. Aspetti chimici

La co-precipitazione viene condotta in presenza di ammoniaca in quanto la crescita degli idrossidi di NMC è dovuta all'equilibrio fra la soluzione di ammoniaca e le particelle di idrossido. L'ammoniaca aumenta la solubilità dell'idrossido:

$$M(OH)_2 + nNH_3 \leftrightarrow [M(NH_3)_n]^{2+} + 2OH^- \tag{III.1}$$

La crescita è quindi dovuta alla dissoluzione di particelle più piccole in favore della crescita di quelle più grandi:

$$M(OH)_2 \leftrightarrow M^{2+} + 2OH^- \tag{III.2}$$

Gli equilibri e le relative costanti che descrivono il fenomeno sono elencate in tabella 2.2.

Da questi equilibri si può definire un sistema di equazioni per il calcolo delle concentrazioni dei metalli nella soluzione e il successivo calcolo della supersaturazione:

$$C_{M_i^{2+}}^{total} - [M_i^{2+}] - \sum_{n_i=1}^{n_i^{(\max)}} k_n^{(Mi)}[M_i{}^{2+}][NH_3]^{n_i} = 0 \qquad with\ i = 1,2,3 \tag{III.3}$$

$$C_{NH_3}^{total} - [NH_3] - \frac{k_b[NH_3]}{[OH^-]} - \sum_{i=1}^{3} \sum_{n_i=1}^{n_i^{(max)}} n_i k_n^{(Mi)}[M_i{}^{2+}][NH_3]^{n_i} = 0 \tag{III.4}$$

$$2\sum_{i=1}^{3} C_{M_i^{2+}}^{total} + \frac{k_b[NH_3]}{[OH^-]} + \frac{k_w}{[OH^-]} + [Na^+] - 2[SO_4^{2-}] - [OH^-] = 0 \qquad \text{(III.5)}$$

In cui il termine $[Na^+] - 2[SO_4^{2-}]$ viene definito come concentrazione degli ioni inerti, che può essere considerata costante (non intervengono in reazioni) e sarà utile per la risoluzione del sistema.

La supersaturazione, forza spingente della precipitazione, è definita per il sistema come:

$$S = \left(\frac{\prod[M^{2+}]^{MConcRatio}\,[OH^-]^2}{K_{sp\_NMC}}\right)^{\frac{1}{3}} \qquad \text{(III.6)}$$

In cui *MConcRatio* è il rapporto fra la concentrazione del metallo *M* e quella di metalli totale, mentre K_{sp_NMC} è definita come:

$$K_{sp\_NMC} = (K_{sp\_Ni})^{NiConcRatio}(K_{sp\_Mn})^{MnConcRatio}(K_{sp\_Co})^{CoConcRatio} \qquad \text{(III.7)}$$

Con: $K_{sp\_Ni} = 10^{-15.22}$, $K_{sp\_Mn} = 10^{-12.70}$, $K_{sp\_Co} = 10^{-14.89}$.

IV. Bilancio di popolazione

La funzione densità di popolazione *n(L,**x**,t)* rappresenta la probabilità di avere un numero di particelle in un determinato intervallo di coordinata interna L (in questo caso lunghezza):

$$n(L,\boldsymbol{x},t)dL = numero\ di\ particelle\ con\ lunghezza\ fra\ L\ e\ L + dL$$

Il bilancio di popolazione ne descrive l'evoluzione in un sistema, in quanto è un bilancio del numero di particelle in un volume infinitesimo a un tempo t:

$$\frac{\partial n(L,\boldsymbol{x},t)}{\partial t} + \nabla\cdot\big(\boldsymbol{u}\cdot n(L,\boldsymbol{x},t)\big) = J(S) - \frac{\partial}{\partial L}\big(G(L)\cdot n(L,\boldsymbol{x},t)\big) + B_{agg}(L,\boldsymbol{x},t) - D_{agg}(L,\boldsymbol{x},t) \qquad \text{(IV.1)}$$

In cui *S* è la supersaturazione, $J(S)$ la velocità di nucleazione, $G(L)$ la velocità di crescita e $B_{agg}(L,\boldsymbol{x},t)$ e $D_{agg}(L,\boldsymbol{x},t)$ le velocità di nascita e morte dovute ad aggregazione.

Dalla definizione di momento di ordine q:

$$m_q(\boldsymbol{x},t) = \int_0^{+\infty} n(L,\boldsymbol{x},t) \cdot L^q \, dL \qquad \text{(IV.2)}$$

$$q = 0, 1, 2, 3$$

Si può scrivere il PBE in termini di momenti, ricavando un'equazione di continuità (momento di ordine zero ed equazioni di trasporto per le proprietà medie (Marchisio et al., 2003):

$$\frac{\partial m_q}{\partial t} + \nabla \cdot (\boldsymbol{u} \cdot m_q) = x_c^q \cdot J(S) + \int_0^{+\infty} q \cdot L^{q-1} \cdot G(L) \cdot n(L,\boldsymbol{x},t) \; dL + \int_0^{+\infty} (B_{agg}(L,\boldsymbol{x},t) - D_{agg}(L,\boldsymbol{x},t)) L^q dL \qquad \text{(IV.3)}$$

In cui x_c è la dimensione critica di nucleazione.

Questo tipo di formulazione presenta però un eccesso di incognite in confronto alle equazioni, ed è quindi affetto da un problema della chiusura (Dorao e Jakobsen, 2006). Questo problema si affronta con il metodo della quadratura dei momenti (QMOM):

$$m_q = \sum_{i=1}^{N} w_i \cdot L_i^q \qquad \text{(IV.4)}$$

E le equazioni dei momenti diventano quindi:

$$\frac{\partial m_q}{\partial t} + \nabla \cdot (\boldsymbol{u} \cdot m_q) = x_c^q \cdot J(S) + q \cdot \sum_{i=1}^{N} w_i \cdot L_i^{q-1} \cdot G(L_i) + \frac{1}{2} \sum_{i=1}^{N} w_i \sum_{j=1}^{N} w_j (L_i^3 + L_j^3)^{\frac{q}{3}} \beta_{agg} - \sum_{i=1}^{N} w_i L_i^q \sum_{j=1}^{N} w_j \, \beta_{agg} \qquad \text{(IV.5)}$$

A questo punto è necessario trovare i valori dei pesi e dei nodi del metodo di quadratura. Per farlo è possibile utilizzare algoritmi consolidati come quello product-difference (PD), Wheeler o Wheeler adattivo.

V. Cinetiche di precipitazione

I principali meccanismi coinvolti sono nucleazione, crescita e aggregazione.

La nucleazione è la formazione di nuove particelle di precipitato e si distingue in omogenea ed eterogenea, a seconda che si verifichi rispettivamente per la sola presenza della supersaturazione o per un effetto combinato di supersaturazione e interfacce solide.

La dipendenza della velocità di nucleazione dalla supersaturazione è molto simile nei due casi, differendo nel caso eterogeneo di un fattore *f*, compreso tra 0 e 1, dipendente dalle proprietà dell'interfaccia solida.

Le equazioni sono le seguenti:

$$J_{hom} = J_{hom}^{\infty} \exp\left(-\frac{k}{ln^2 S}\right) \tag{V.1}$$

$$J_{het} = J_{het}^{\infty} \exp\left(-\frac{f * k}{ln^2 S}\right) \tag{V.2}$$

In cui le costanti $J_{hom/het}^{\infty}$ e k dipendono dalle proprietà fisiche del sistema e delle sostanze.

La crescita è il meccanismo per cui le molecole passano dalla soluzione alla superficie delle particelle, causandone un aumento di dimensioni. La velocità di crescita può dipendere da due diversi fenomeni che avvengono in serie: il trasferimento di materia e l'integrazione superficiale. La velocità di crescita è strettamente legata al flusso molare per unità di superficie, che per i due fenomeni in serie può essere scritto come (Mersmann, 2001):

$$\dot{n} = k_{mt}\left(C_{bulk} - C_{surface}\right) = k_{si}\left(C_{surface} - C_{eq}\right)^{g} \tag{V.3}$$

I cui flussi devono essere appunto uguali perché avvengono in serie.

Infine l'aggregazione è il meccanismo nel quale particelle già esistenti si uniscono, creando particelle più grandi. L'aggregazione può essere dovuta a moto browniano e al regime turbolento, i cui kernel di aggregazione sono (Gavi et al, 2007):

$$k_{bm} = \frac{2k_B T}{3\mu}\frac{\left(L_i + L_j\right)^2}{L_i L_j} \tag{V.4}$$

$$k_{tf} = 1.29\sqrt{\frac{\varepsilon}{\nu}}\left(L_i + L_j\right)^3 \tag{V.5}$$

In cui k_B è la costante di Boltzmann, T la temperatura, μ la viscosità dinamica, L_i e L_j le dimensioni delle particelle coinvolte nell'aggregazione, ε la velocità di dissipazione turbolenta e ν la viscosità cinematica.

VI. Equazioni di trasporto di specie e momenti

Le equazioni da risolvere nel sistema in esame, oltre al campo di moto, riguardano il trasporto delle specie coinvolte nella precipitazione e il trasporto dei momenti della funzione di distribuzione della popolazione, rappresentanti proprietà fisiche del precipitato.

Le specie di cui si risolve l'equazione di traporto sono cinque: i tre cationi metallici $[M^{2+}]$ ($[Ni^{2+}]$, $[Mn^{2+}]$, $[Co^{2+}]$), l'ammoniaca e le cariche inerti (definite precedentemente come: $C_{cariche\ inerti} = [Na^{+}] - 2[SO_4^{2-}]$).

Per gli ioni metallici si ha:

$$\frac{\partial C_{M^{2+}}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{M^{2+}}) \ = S_{M^{2+}} \qquad \text{(VI.1)}$$

In cui il termine sorgente $S_{M^{2+}}$ rappresenta il meccanismo di precipitazione ed è espresso come:

$$S_{M^{2+}} = -precRate \times M^{2+}ConcRatio \qquad \text{(VI.2)}$$

In cui $M^{2+}ConcRatio$ è il rapporto molare fra il metallo M^{2+} e la somma di tutti i metalli, mentre $precRate$ dipende dalla quantità di precipitato formatasi, quindi dipende dal termine sorgente del terzo momento (che rappresenta infatti il volume di precipitato generato per unità di volume):

$$precRate = K_v \times (x_c^3 \cdot J(S) + \sum_{j=1}^{2} (3 \times w_j \times G(L_j) \times L_j^2)) \times \frac{\rho crystal}{aMassCrystal} \qquad \text{(VI.3)}$$

Per quanto riguarda ammoniaca e cariche inerti, non ci sono termini sorgente in quanto il primo è solo un intermedio della reazione, quindi non si consuma, mentre il secondo non interviene affatto nelle reazioni. Le loro equazioni di trasporto sono:

$$\frac{\partial C_{NH_3}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{NH_3}) = 0 \qquad \text{(VI.4)}$$

$$\frac{\partial C_{cariche\ inerti}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{cariche\ inerti}) = 0 \qquad \text{(VI.5)}$$

Le equazioni di trasporto dei momenti di ordine q sono invece nella forma:

$$\frac{\partial m_q}{\partial t} + \nabla \cdot (\boldsymbol{u} m_q) = S_{m_q} \qquad \text{(VI.6)}$$

In cui il termine sorgente è definito come:

$$S_{m_q} = x_c^q \cdot J(S) + q \cdot \sum_{i=1}^{N} w_i \cdot L_i^{q-1} \cdot G(L_i) + \frac{1}{2}\sum_{i=1}^{N} w_i \sum_{j=1}^{N} w_j \left(L_i^3 + L_j^3\right)^{\frac{q}{3}} \beta_{agg} - \sum_{i=1}^{N} w_i L_i^q \sum_{j=1}^{N} w_j \,\beta_{agg} \tag{VI.7}$$

Di cui è interessante notare che per q = 3 si ottiene il termine sorgente del momento di ordine tre, collegato come visto al termine sorgente dell'equazione di trasporto dei cationi metallici e non affetto dai termini di aggregazione, dal momento che questa non varia il volume totale del precipitato.

VII. Simulazione CFD

La geometria del reattore è data da Umicore. Si tratta di un reattore agitato operante in continuo, alimentato da ammoniaca, idrossido di sodio e solfati di Nichel, Manganese e Cobalto (questi metalli in proporzione 6-2-2, per ottenere un precipitato con la stessa proporzione in metalli).

Le equazioni che descrivono la fluidodinamica del sistema descritte nel capitolo II non sono risolvibili analiticamente e possono essere risolte tramite discretizzazione delle equazioni. Il metodo utilizzato da Fluent è quello dei volumi finiti: il dominio viene diviso in piccoli volumi (celle) in cui vengono integrate le equazioni fluidodinamiche e risolte grazie ad opportune approssimazioni, come per il termine convettivo (viene utilizzato il first order upwind e, per maggiore precisione, il second order upwind). L'accoppiamento pressione-velocitá viene effettutato con l'algoritmo SIMPLE.

Le celle intorno alle parti in rotazione sono raggruppate in zone, anche loro rotanti alla stessa velocità, secondo il modello 'multiple reference frame'.

VIII. Script Python

Lo script si propone di risolvere le equazioni che descrivono la co-precipitazione non per ogni cella, come avverrebbe effettuando una simulazione CFD-PBE, portando alla soluzione del sistema di equazioni per centinaia di migliaia di celle, ma per qualche decina di compartimenti, raggruppamenti di celle generati utilizzando lo strumento Reactor Network Model di ANSYS Fluent e le informazioni ad essi legati estratti tramite un'opportuna User Defined Function (UDF), riducendo considerevolmente tempi e costi computazionali.

Questi compartimenti hanno il comportamento di reattori CSTR e sono interconnessi: le equazioni risolte quindi sono quelle della precipitazione (Eq.s VI.1, VI.4, VI.5 E VI.6) in cui compaiono anche termini dovuti ai flussi fra i compartimenti:

$$\frac{dC_{M^{2+}}^j}{dt} = \sum_i \frac{Q^{i,j}}{V^j} C_{M^{2+}}^i - \sum_i \frac{Q^{j,i}}{V^j} C_{M^{2+}}^j + S_{M^{2+}}^j \tag{VIII.1}$$

$$\frac{dC_{NH_3}^j}{dt} = \sum_i \frac{Q^{i,j}}{V^j} C_{NH_3}^i - \sum_i \frac{Q^{j,i}}{V^j} C_{NH_3}^j \tag{VIII.2}$$

$$\frac{dC^{j}_{inert\,charges}}{dt} = \sum_{i} \frac{Q^{i,j}}{V^{j}} C^{i}_{inert\,charges} - \sum_{i} \frac{Q^{j,i}}{V^{j}} C^{j}_{inert\,charges} \tag{VIII.3}$$

$$\frac{dm^{j}_{k}}{dt} = \sum_{i} \frac{Q^{i,j}}{V^{j}} m^{i}_{k} - \sum_{i} \frac{Q^{j,i}}{V^{j}} m^{i}_{k} + S^{j}_{m_k} \tag{VIII.4}$$

I compartimenti vengono quindi creati utilizzando il tool sopra citato; a questo tool devono essere fornite due informazioni: il criterio utilizzato per dividere il reattore, ossia una Custom Field Function il cui campo nel dominio viene utilizzato per definire i compartimenti, e il numero di compartimenti desiderati.

Una volta creati, si utilizza una UDF per estrarre le informazioni sui compartimenti: i dati relativi ai volumi dei compartimenti vengono scritti nel file *react_zone_ave*, quelli relativi ai flussi fra i compartimenti nel file *react_zone_flux* e quelli relativi a flussi in ingresso e in uscita, con relativi compartimenti e concentrazioni di ingresso, vengono scritti nel file *react_zone_flux_toBoundary*. Da queste informazioni in input, lo script risolve le equazioni VIII.1, VIII.2, VIII.3 E VIII.4 per ogni compartimento, arrivando a trovare quindi le concentrazioni e i momenti allo stazionario di ogni compartimento. Queste equazioni come si può vedere sono scritte come dipendenti dal tempo, in quanto per risolverle viene utilizzata un risolutore di equazioni differenziali ordinarie presente nella libreria Scipy, che risolve un problema ai valori iniziali. Essendo la soluzione ricercata quella allo stazionario, è sufficiente calcolare la soluzione nel tempo finché non viene raggiunto uno stazionario.

Il file *init_run* legge tutti questi dati da questi file e li immagazzina in vettori, dopodiché dal file *caseSetup* legge le impostazioni del risolutore di equazioni differenziali, le impostazioni per l'algoritmo di Newton-Raphson, il numero di nodi utilizzato nel metodo di quadratura e altri dati utili come la densità dei cristalli.

Grazie a queste informazioni, nel file *NiMnCoHydroxidePrec* viene utilizzato il risolutore di sistemi di equazioni differenziali ordinarie *solve_ivp*, a cui bisogna fornire le equazioni, i cui termini cambiano nel tempo e devono essere aggiornati.

Le equazioni vengono aggiornate calcolando la supersaturazione tramite il file *ChemicalEquilibria.py,* a partire dalla soluzione delle concentrazioni del tempo precedente. Questo file si avvale dell'algoritmo di Newton-Raphson modificato per risolvere il sistema di equazioni che descrivono l'equilibrio delle specie (Eq.s III.3, III.4 E III.5) e poter così calcolare la supersaturazione (Eq. III.6).

Dopodiché vengono calcolati pesi e nodi del metodo di quadratura dei momenti con il file *MomentCalc.py*, che utilizza l'algoritmo Wheeler adattivo. A questo punto si possono utilizzare i modelli cinetici presenti nei relativi file (*growth.py, nucleateSize.py, nucleation.py, aggrEfficency.py* and *aggregation.py*) per calcolare le cinetiche di precipitazione.

Con ciò, è possibile definire per ogni compartimento le equazioni che ne descrivono l'evoluzione nel tempo, considerando anche i flussi fra i compartimenti che trasportano sia concentrazioni che momenti.

Infine il file *RunPrecSolver.py* è il quello che viene eseguito per avviare i calcoli. Raccoglie tutti i passaggi precedentemente descritti e li richiama nell'ordine giusto, dando alle funzioni gli input di cui hanno bisogno e arrivando così a calcolare la soluzione.

Altri file di supporto sono presenti, con funzioni secondarie di snellimento dello script, oppure con funzioni di post-processing, come creazioni di grafici e altro.

IX. Risultati CFD

Dalla simulazione CFD vengono calcolati i campi di moto e di turbolenza del reattore. Il campo di moto risulta simmetrico rispetto all'asse di rotazione. Il campo di moto garantisce la dispersione delle particelle precipitate e un buon grado di miscelazione dei reagenti in ingresso.

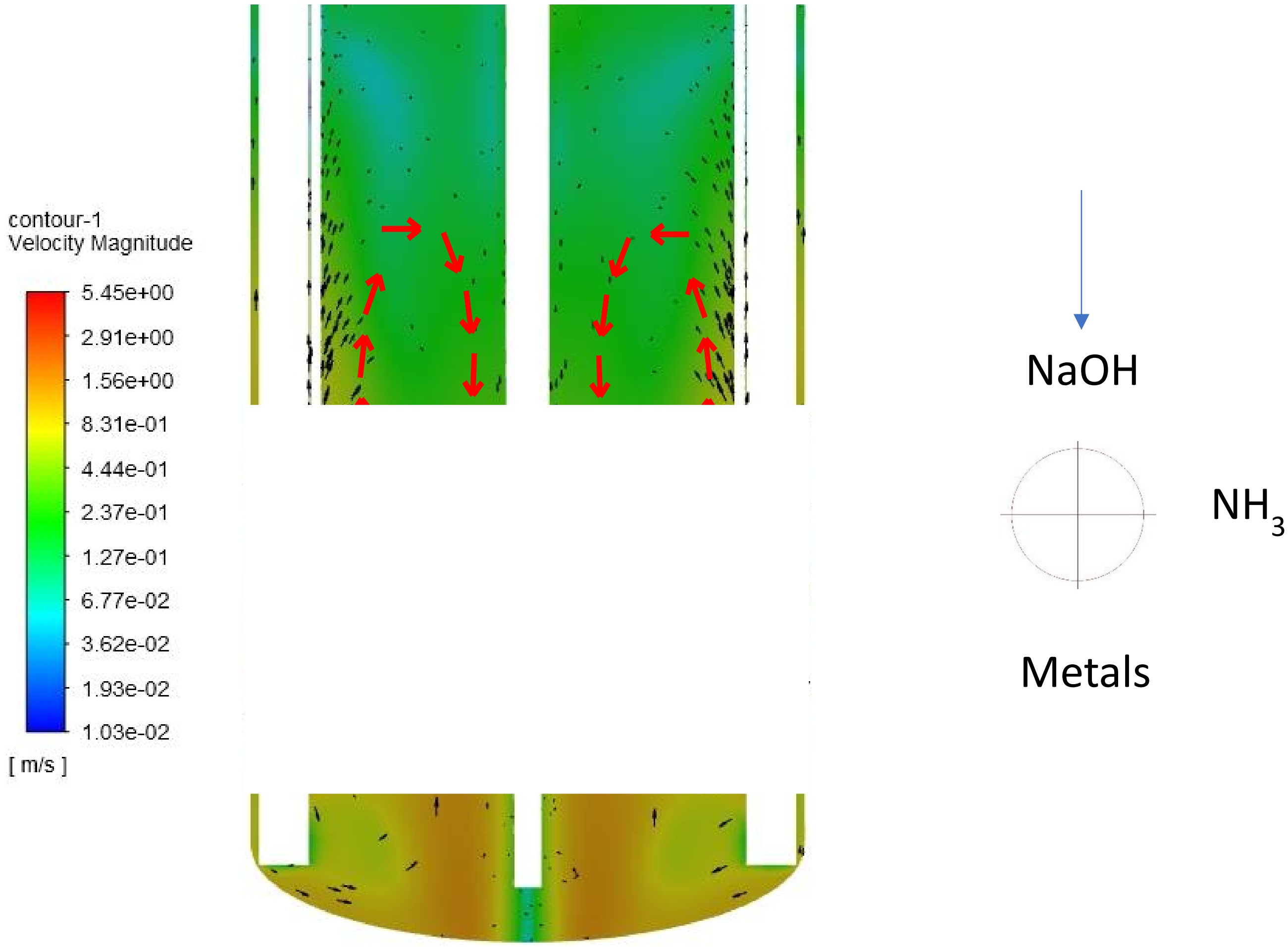


Figura I Valori di velocità (m/s) nel piano contenente l'ingresso di NH_3

Il valore di velocità di dissipazione dell'energia cinetica turbolenta media nel reattore è molto simile alla potenza utilizzata da Umicore per agitare il reattore.

X. Risultati dello script

Sono state esplorate tre strategie diverse per la divisione in compartimenti del reattore:

- Divisione utilizzando la velocità di dissipazione dell'energia cinetica turbolenta
- Divisione utilizzando la supersaturazione
- Divisione utilizzando entrambi i parametri precedenti

Utilizzando la velocità di dissipazione dell'energia cinetica turbolenta si nota che vengono formati compartimenti molto grandi agli ingressi: questo comporta una mancata descrizione degli alti gradienti di concentrazione che ci sono in prossimità degli ingressi. Questi gradienti vengono in questo modo ‘appiattiti’ in un volume (perfettamente miscelato) molto grande. In questo modo l’alto gradiente di supersaturazione che sarebbe presente non viene considerato, comportando una sottostima di nucleazione e crescita delle particelle.

Si è pensato quindi di utilizzare la supersaturazione. Il campo di questa viene ricavato da una simulazione in cui il PBM viene direttamente risolto da Ansys Fluent in ogni cella. Questa simulazione ancora non è giunta a convergenza, ma il campo di supersaturazione è consistente con quello previsto allo stazionario. Utilizzando la sola supersaturazione invece i compartimenti che vengono creati sono piccoli abbastanza da descrivere in modo appropriato il gradiente di concentrazione agli ingressi, ma non si creano compartimenti che suddividano le zone in base alla velocità di dissipazione dell’energia cinetica turbolenta, termine che compare nel calcolo della velocità di aggregazione, il che può portare a una cattiva stima dei termini di aggregazione.

Infine si è scritto un UDF per generare un campo che consideri i valori di supersaturazione vicino all’ingresso e i valori di dissipazione nel resto del reattore. In questo modo si son potuti creare sia i compartimenti vicino agli ingressi, sia quelli che descrivono la variazione della dissipazione (figura III).

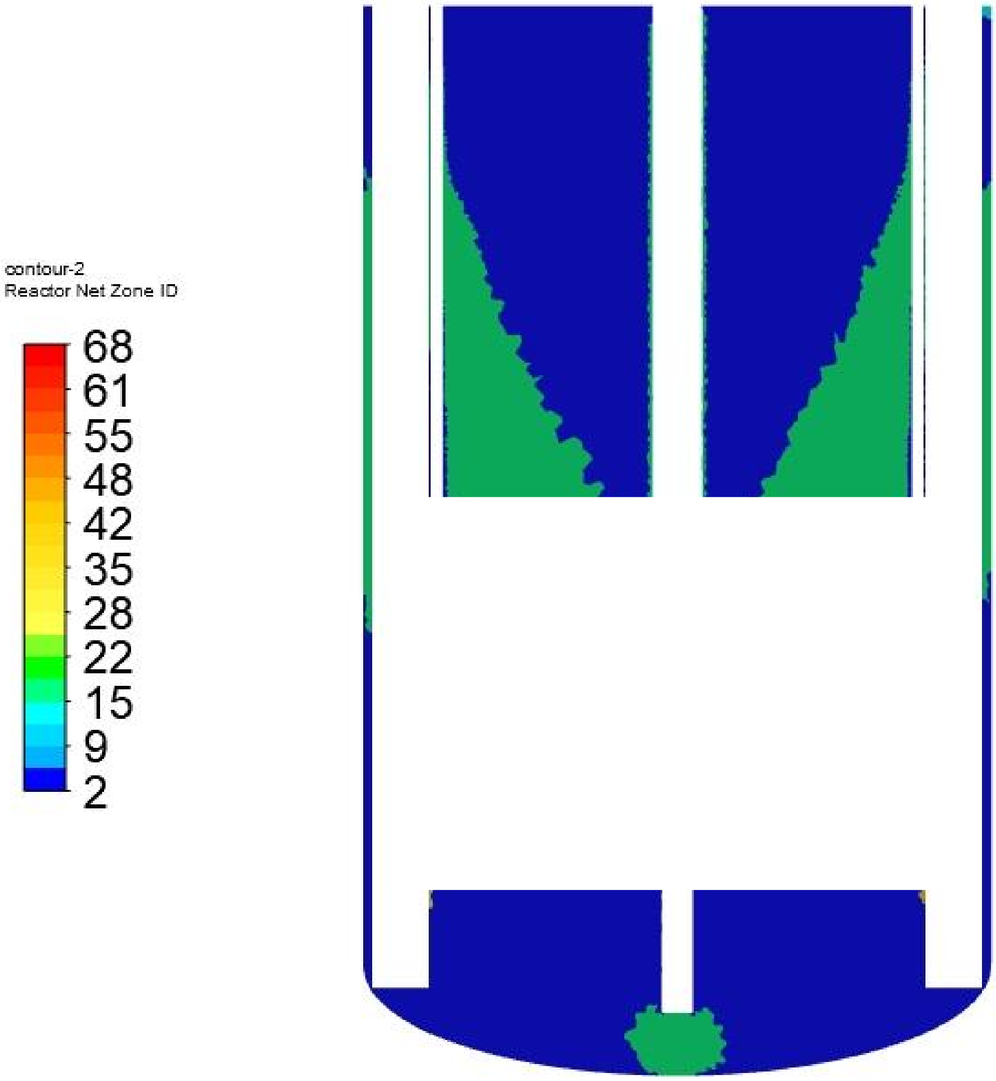


Figura II Compartimenti (70) creati con l’UDF in un piano che contiene l’ingresso di NaOH

Aumentando il numero di compartimenti si è visto che la presenza di piccoli compartimenti vicino agli ingressi permette una buona stima del gradiente della supersaturazione e quindi di nucleazione e crescita (figura IV), con risultati relativi ai momenti del PSD con ordini di grandezza ragionevoli (tabella III).

Outlet	**kmol/m^3**
Ni^{2+}	$4{,}94x10^{-7}$
Mn^{2+}	$1{,}65x10^{-7}$
Co^{2+}	$1{,}65x10^{-7}$
NH_3	$8{,}74x10^{-1}$
m_0	$6{,}7x10^{+12}$
m_1	$3{,}98x10^{+7}$
m_2	$1{,}17x10^{+3}$
m_3	$5{,}57x10^{-2}$
SMD	$4{,}8x10^{-5}$

Tabella II Concentrazioni, momenti e SMD (Sauter Mean Diameter) all'uscita del modello a 70 compartimenti creati con l'UDF

Al tempo stesso si è visto che l'aggregazione avviene solo nei compartimenti vicino all'ingresso dei metalli, compartimenti in cui la velocità di crescita è sufficientemente elevata da ottenere un valore di efficienza di aggregazione sensibilmente diverso da zero (figura V).

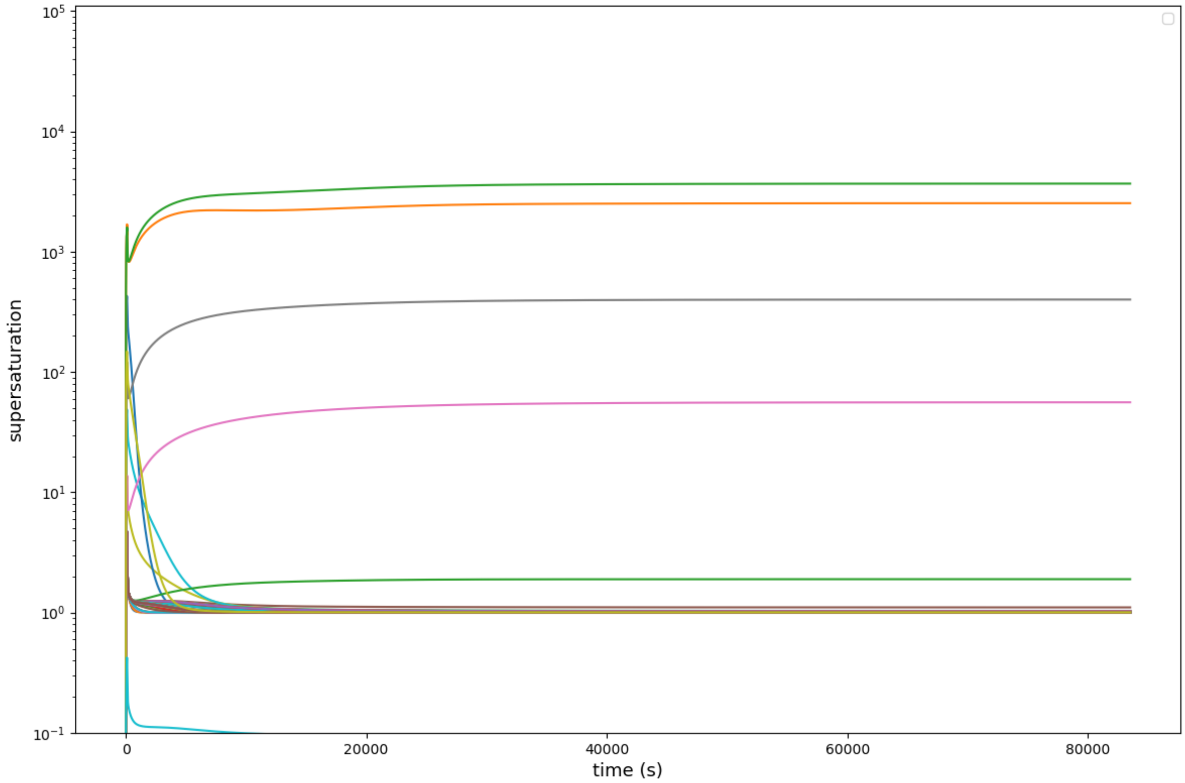


Figure III Supersaturazione dei 70 compartimenti creati dividendo il reattore con l'UDF

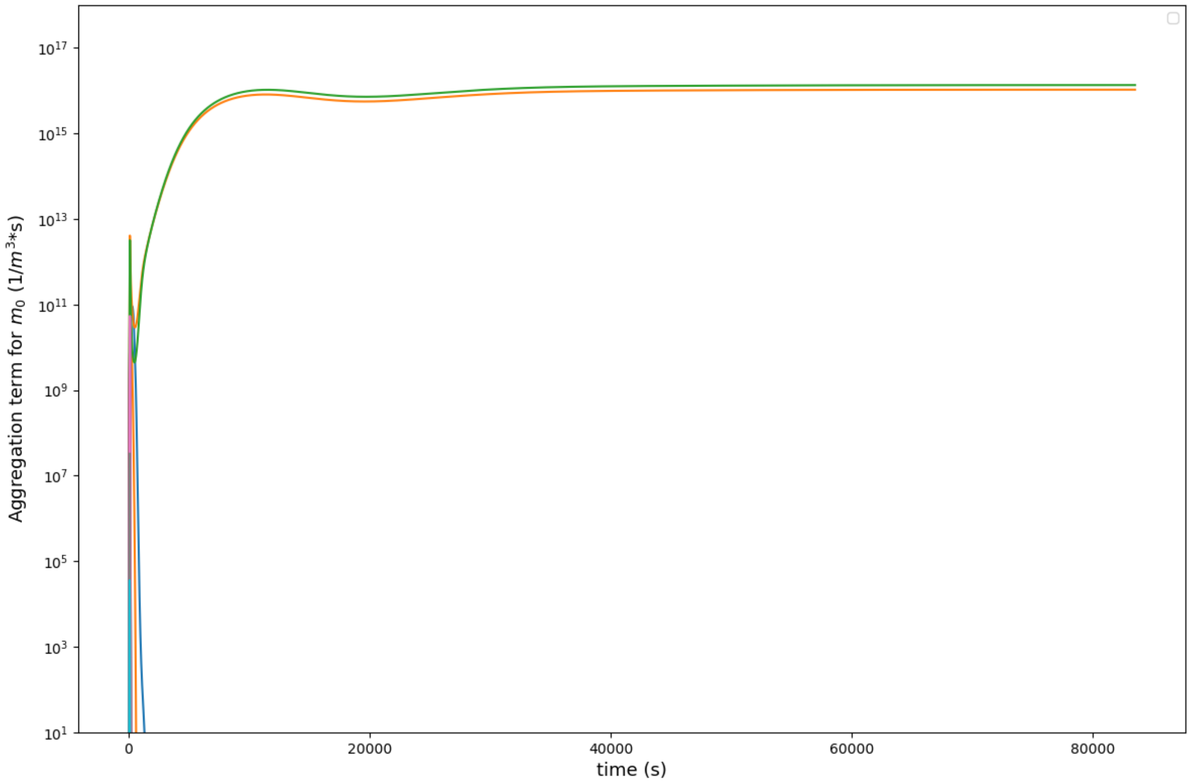


Figure IV Contributo all'aggregazione del momento di ordine zero nei 70 compartimenti creati dividend il reattore con l'UDF (comportamento simile per gli altri momenti)

XI. Conclusioni

Nella presente Tesi è stato studiato un modello a compartimenti per la risoluzione delle equazioni che descrivono la precipitazione di idrossidi di Nichel, Manganese e Cobalto in un reattore agitato in condizioni stazionarie.

Questo comporta una diminuzione dei costi computazionali e dei tempi derivanti dal risolvere utilizzando ANSYS Fluent questo sistema in ogni cella del dominio (simulazione CFD-PBM): infatti quest'ultima richiede oltre a un'ampia potenza di calcolo, dei tempi che possono essere nell'ordine di grandezza delle settimane, mentre utilizzando il modello a compartimenti, con un solo processore, i tempi si riducono a circa 12 ore con 20 compartimenti, fino a 48-72 ore con 70 compartimenti.

Le equazioni del modello a compartimenti vengono risolte utilizzando uno script scritto in linguaggio Python.

Questo script ha bisogno dei dati dei flussi, volumi e proprietà medie dei compartimenti. Questi compartimenti vengono creati utilizzando un tool di ANSYS Fluent, il Reactor Network Model, che divide il dominio in compartimenti utilizzando uno o più parametri. Una volta diviso il dominio, tramite l'utilizzo di un UDF si estraggono i dati necessari allo script.

Cruciale per l'attendibilità dei risultati è il criterio della divisione del reattore in compartimenti.

Utilizzando solamente la velocità di dissipazione dell'energia cinetica turbolenta, i compartimenti creati agli ingressi sono troppo grandi e descrivono male il gradiente di concentrazione e di supersaturazione, portando a una sottostima di velocità di nucleazione e di crescita.

Utilizzando la sola supersaturazione per dividere il reattore, il campo di velocità di dissipazione non viene considerato, fatto che può portare a una stima errata della velocità di aggregazione.

Utilizzando la supersaturazione agli ingressi e la velocità di dissipazione altrove (tramite un'opportuna UDF) vengono creati piccoli compartimenti agli ingressi e il resto del reattore viene diviso secondo la velocità di dissipazione.

Analizzando i termini dell'aggregazione, si può notare che la divisione tramite la velocità di dissipazione non è determinante dal momento che l'aggregazione non è trascurabile solo nei compartimenti più vicini all'ingresso dei metalli. Con i parametri cinetici attuali quindi la velocità di crescita ha un grande impatto sull'efficienza di aggregazione, facendo in modo che solo i compartimenti con alta velocità di crescita abbiano un'efficienza sensibilmente diversa da zero. Quindi la creazione di compartimenti di piccole dimensioni agli ingressi è il fattore più influenzante i risultati.

Un metodo per la creazione dei compartimenti e l'importanza dei compartimenti di ingresso sono stati definiti. Prospettive future del lavoro sono un confronto con i risultati di una simulazione CFD-PBM risolta in ogni cella e parallelizzazione dello script per maggiori prestazioni.

1. Introduction

Lithium-ion batteries are rechargeable energy storage devices whose use is widespread in many fields (for example in the electronics and automobile industry). The functioning of these batteries consists in lithium ions that during discharge move, through an electrolyte, from the negative electrode to the positive electrode.

These batteries show high energy and power density, they are not affected by memory effect, a longer cycle life duration (relatively to the others rechargeable batteries) and a low self-discharge. The technology development keeps requiring from these batteries an enhanced capacity and higher energy density, but at the same time a reduced weight and size. This is dependent on the cathode composition.

Usually the negative electrode is made by graphite and the positive electrode is made by a layered compound (rather than spinel or polyanion oxides, Manthiram, 2020). The type of battery often takes its name from the cathode material.

One of the more performing types of Lithium-ion battery is the NMC battery, that stands for Nickel-Manganese-Cobalt, since the cathode has the form of:

$LiNi_xMn_yCo_zO_2 \quad with\ x+y+z=1$

This type of cathode has lower costs, less toxicity and higher capacity in comparison to the single metal oxides ($LiCoO_2$, $LiMnO_2$ and $LiNiO_2$). The NMC material is subjected to irreversible capacity loss, accelerated by electrolyte decomposition at the electrode-electrolyte interface: the layer of decomposition product is called CEI (cathode-electrolyte interphase) at cathode and SEI (solid-electrolyte interphase) at anode (Fang et al., 2018). These layers hinder lithium transport at the interfaces.

The electrochemical characteristics of this cathode depend highly on how the synthesis is conducted. There are many synthesis techniques, but using a coprecipitation reaction is the most common since creates homogeneous cation distribution in their layered structures (van Bommel and Dahn, 2009).

The parameters that should be monitored, because of their involvement in the performance, are particle size, size distribution, internal porosity and cation mixing in structure (Barai et al, 2019; Li et al., 2011). As mentioned, these parameters depend on the synthesis performed.

The synthesis is divided in two steps (Barai et al, 2019):

1) co-precipitation: this step is the one covered in this Thesis, and consists in a co-precipitation process that produces the precursor $Ni_xMn_yCo_z(OH)_2$ (NMC hydroxide) from solutions of Lithium, Manganese and Cobalt sulfates under the presence of ammonia. The parameters mentioned above are determined during this step;

2) calcination: the precursor is subjected to oxidation and lithiation in order to obtain $LiNi_xMn_yCo_zO_2$.

The co-precipitation is a complex process that occurs at several scales, from atoms to chemical reactors. It is obviously of interest to predict the characteristic of the precursor particles after the process, in order to manipulate the operating conditions for the improvement of the performance of the battery. This can be easily done with a computational model based on Computational Fluid Dynamics (CFD). In this thesis the work focused on a specific reactor geometry (and the corresponding computational mesh) provided by the Umicore company partner of Politecnico di Torino of the European Project Simdome. For confidentiality reasons some details of the geometry will not be revealed and some figures will be censored.

There are four main phenomena that are considered to simulate the co-precipitation in the reactor: the fluid dynamic of the coprecipitation reactor, described by solving the Navier-Stokes equations, the mixing of the involved reactants, described by solving the transport equations of the species involved, the chemistry of the system, described by solving the corresponding chemical equilibria and the evolution of the formed particles of NMC hydroxides, described by solving the corresponding population balance equation, PBE (Gavi et al.,2007).

The fluid dynamic of the coprecipitation reactor is predicted with the commercial CFD code ANSYS Fluent. Since the flow is turbulent, a turbulence model, namely the standard k-ε model, was employed. Moreover as in the case considered the fluid is agitated by impellers, the multiple reference frame (MRF) was employed.

The transport equations of the involved species are written for the metal cations, for the ammonia and for the inert charges. These equations assume that these species move with water and in the case of metal ions have a source term that represents the consumption of reactants due to the precipitation mechanism.

The chemistry mechanisms involved in the phenomena have been speculated by van Bommel and Dahn (2009), who have hypothesized that growth of the particles depends on an instantaneous equilibrium between the metal hydroxide and the ammonia solution, and they wrote the equilibria equations that determine the concentrations of the species in the reactor.

The PBE was solved in terms of transport equations of the moments of the particle size distribution. The closure problem is overcome by using the quadrature method of moments (QMOM), in which weights and nodes are calculated using an appropriate algorithm. These equations need a known kinetic law, namely nucleation, growth and aggregation rates.

All these additional equations can be coupled with the equation for the flow field by using the user-defined functions of ANSYS Fluent and resulting in the so-called CFD-PBE model. However, repeating all these calculations in every cell of the computational domain leads to high computational costs. The purpose of this Thesis is therefore to study and develop a faster and cheaper compartment model, in order to carry out these simulations in reasonable times.

This is reached by using the Reactor Network tool of ANSYS Fluent, that divides the domain in interconnected volumes. In this way the additional equations of the co-precipitation process can be solved in every volume, considering each of them as CSTR reactors. The solution is calculated now no more in a huge number of cells, but in a very small number of compartments. The computational cost scales with the size of the system and this can be reduced from the initial number of cells (approximately 1 million) to the final number of compartments (order of magnitude of the tens). All the additional equations are solved via an efficient computer code written in Python.

The division into compartments can not be done in a random way, it must be done so that the relevant features in the precipitation are as homogeneous as possible (in each compartment).

In order to achieve that, both supersaturation and turbulent dissipation rate, which are the most important parameters in precipitation kinetics, are used to divide in compartments the reactor.

The thesis is structured as follows:

In chapter 1 a general introduction to the topic of the Thesis and its purpose is made.

In chapter 2 the theory of the four phenomena involved is described.

In chapter 3 the models and tools used in Fluent to reach the results, the setup of the simulations and the functioning of the written script are described.

In chapter 4 the results obtained by the CFD simulations and by the compartment model are described.

In chapter 5 the main conclusions and the future perspectives are described.

2. Theory and fundamentals

2.1 Fluid-dynamics

2.1.1 Governing equations

The fluid dynamics in the coprecipitation reactor is predicted by solving the continuity and Navier-Stokes equations for an incompressible fluid, namely water.

The continuity equation (conservation of mass) reads as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{2.1}$$

Whereas the conservation of momentum equation (or Navier-Stokes equation) as follows:

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial(p)}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i \tag{2.2}$$

In which the term:

$$\tau_{ij} = \left(\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \delta_{i,j} \frac{\partial u_i}{\partial x_i} \right) \tag{2.3}$$

is called deviatoric stress tensor and represents the momentum flux transport due to molecular phenomena. In this equation μ is the dynamic viscosity, a physical property of the specie that can be interpreted as the resistance placed by the fluid to deformation and $\delta_{i,j}$ is the Kronecker delta, a function equal to zero when $i \neq j$ and equal to one when $i = j$.

Equations (2.1) and (2.2) can be simplified considering the fluid incompressible and Newtonian and flowing under steady-state conditions. The time derivatives terms can be neglected and the density is considered constant on the domain, resulting in:

$$\frac{\partial(u_i)}{\partial x_i} = 0 \tag{2.4}$$

$$u_j \cdot \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial(p)}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\nu \frac{\partial u_i}{\partial x_j} \right) + g_i \tag{2.5}$$

We can see that the deviatoric stress tensor is reduced to $\nabla \cdot (\nu \, \nabla \mathbf{u})$, since is first applied the Newton's law and then this is simplified by the incompressible hypothesis. The term ν is called kinematic viscosity and is equal to the ratio between the dynamic viscosity and the density of the fluid.

2.1.2 Equations for turbulent flows

The equations described in the previous section are valid for flows under the laminar regime, i.e. for flows with low Reynolds number (Re).

If the Re is greater than a critical value (i.e. 2100 for flows in pipes) the condition of the flow cannot be considered laminar due to its chaotic behaviour and it is considered turbulent. The equations of the previous chapter can no more describe correctly this kind of regime because turbulent flows are characterized by the fluctuating motions of the fluid. These fluctuations can be modelled using the Reynolds decomposition:

$$u_i = \overline{u_i} + u_i' \tag{2.6}$$

In which $\boldsymbol{u_i}$ is decomposed in the sum of a mean value $\overline{\boldsymbol{u_i}}$ and a fluctuating value $\boldsymbol{u_i'}$. In this way it is possible to write the continuity and momentum balance equations (3.1) and (3.2) for the turbulent regime:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho \overline{u_i})}{\partial x_i} = 0 \tag{2.7}$$

$$\frac{\partial(\rho \overline{u_i})}{\partial t} + \frac{\partial\left(\rho \overline{u_i\, u_j}\right)}{\partial x_j} = -\frac{\partial(\overline{p})}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \overline{u_\imath}}{\partial x_j} + \frac{\partial \overline{u_\jmath}}{\partial x_i}\right)\right) - \rho \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right) + \rho g_i \tag{2.8}$$

As it is possible to see the continuity equation is identical to the equation under laminar conditions (equation 2.1), but instead of u_i the time averaged $\overline{\boldsymbol{u_i}}$ is used instead. The same can be stated for the Navier-Stokes equation with the difference that now the additional term: $\rho \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right)$ appears. This is known as Reynold stress tensor, it is constituted by the fluctuating velocities and represents the momentum transport caused by turbulent fluctuations (Bird et al, 2002). This term is the cause of the closure problem present in turbulent models: the term $\rho \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right)$ is a complicated function of unknown variable and any attempt to write it in terms of other variables will result in equations containing the third order term $\overline{u_i' u_j' u_k'}$. Therefore these equations can be solved with empirical models such as the one described in next section.

2.1.3 Standard k-ε model

The standard k-ε model is a second order model that has been proposed by Launder and Spalding (1973) and, as mentioned before, is used in computational fluid dynamics to deal with the closure problem of turbulence equations.

The model solves the transport equation for the turbulent kinetic energy, defined as:

$$k = \frac{1}{2}\operatorname{tr}\left(\langle u_i' u_j' \rangle\right) \tag{2.9}$$

For which it is possible to derive its balance equation, from Eq. (2.8):

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho k\overline{u}_{\boldsymbol{i}})}{\partial x_i}=\frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right)+\mu_t\left(\frac{\partial\overline{u}_{\boldsymbol{i}}}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\frac{\partial\overline{u}_i}{\partial x_j}-\rho\varepsilon \tag{2.10}$$

In which the generation of turbulent kinetic energy: $\mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\frac{\partial\overline{u}_i}{\partial x_j}$, occurring at the large scale is compensated by the turbulent dissipation rate, ε, occurring at the small scale an defined as follows:

$$\varepsilon = 2\nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} \tag{2.11}$$

Considering, from Kolmogorov theory, the relationship between k and ε in the energy containing range for eddies of typical dimension L:

$$\varepsilon = \frac{k^{\frac{3}{2}}}{L} \tag{2.12}$$

A turbulent dissipation rate balance equation can be written, using some physical consideration, in a form similar to that of Eq. (2.10):

$$\frac{\partial(\rho\varepsilon)}{\partial t}+\frac{\partial(\rho\varepsilon\overline{u}_i)}{\partial x_i}=\frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial\varepsilon}{\partial x_j}\right)+C_1\mu_t\frac{\varepsilon}{k}\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\frac{\partial\overline{u}_i}{\partial x_j}-\rho C_2\frac{\varepsilon^2}{k} \tag{2.13}$$

The term μ_t (turbulent or eddy viscosity) is calculated with the following equation:

$$\mu_t = \rho C_\mu\frac{k^2}{\varepsilon} \tag{2.14}$$

The constants seen in the equations above (C_1, C_2, C_μ and the turbulent Prandtl numbers σ_ε and σ_k) have been determined empirically and their values are reported in table 2.1 (ANSYS Fluent Theory Guide 15, 2013).

Table 2.1. Values of the constants appearing in the k-eps model as implemented in Ansys Fluent.

C_1	C_2	C_μ	σ_ε	σ_k
1.44	1.92	0.09	1.0	1.3

2.2 Coprecipitation and chemical equilibria

The synthesis of NMC hydroxides by a coprecipitation reaction is carried out in presence of ammonia. This is done because it has been showed that the growth of hydroxide particles is due to the equilibrium between metal hydroxide particles and aqueous ammonia solution (van Bommel and Dahn, 2009); with this mechanism, aqueous ammonia increases the solubility of the metal hydroxide, as shown in the following equilibrium:

$$M(OH)_2 + nNH_3 \leftrightarrow [M(NH_3)_n]^{2+} + 2OH^- \tag{2.15}$$

Therefore the reaction leading to the growth of the particles of interest is described by the dissolution of the metal hydroxide, which is improved by the presence of aqueous ammonia, and recrystallization mechanism. So smaller particles dissolve:

$$M(OH)_2 \leftrightarrow M^{2+} + 2OH^- \tag{2.16}$$

in order to make larger particles grow.

The description of the equilibria acting during the coprecipitation process, reported in table 2.2, coupled with the mass balance of the species (see 2.2.2), is necessary in order to calculate the equilibrium concentrations of the species in solution and the resulting supersaturation. The reactions considered are the coordination of ammonia to the metal ions, the base dissociation constant of aqueous ammonia, the solubility of the metal hydroxide and the water dissociation constant.

Table 2.2. Equilibria reactions and reaction constants in coprecipitation

Equilibrium reaction	Ni (log K)	Mn (log K)	Co (log K)
$M^{2+} + NH_3 \leftrightarrow [M(NH_3)]^{2+}$	2.81	1.00	2.10
$M^{2+} + 2NH_3 \leftrightarrow [M(NH_3)_2]^{2+}$	5.08	1.54	3.67
$M^{2+} + 3NH_3 \leftrightarrow [M(NH_3)_3]^{2+}$	6.85	1.7	4.78
$M^{2+} + 4NH_3 \leftrightarrow [M(NH_3)_4]^{2+}$	8.12	1.3	5.53
$M^{2+} + 5NH_3 \leftrightarrow [M(NH_3)_5]^{2+}$	8.93	-	5.75
$M^{2+} + 6NH_3 \leftrightarrow [M(NH_3)_6]^{2+}$	9.08	-	5.14
$NH_3 + H_2O \leftrightarrow NH_4^+ + OH^-$	-4.8	-4.8	-4.8
$M(OH)_2 \leftrightarrow M^{2+} + 2OH^-$	-15.22	-12.70	-14.89
$H_2O \leftrightarrow H^+ + OH^-$	-14	-14	-14

2.2.1 Supersaturation

Supersaturation is the driving force for coprecipitation and it is needed in order to obtain a solid phase from a saturated solution.

Supersaturation in this system is defined as follows:

$$S = \left(\frac{\prod[M^{2+}]^{MConcRatio}\,[OH^{-}]^{2}}{K_{sp\_NMC}}\right)^{\frac{1}{3}} \tag{2.17}$$

In which $[M^{2+}]$ is the concentration of the metal cation, *MConcRatio* is the fraction of the metal cation concentration on the overall metal cations concentration:

$$NiConcRatio = \frac{[Ni^{2+}]}{\sum[M^{2+}]} \tag{2.18}$$

$$MnConcRatio = \frac{[Mn^{2+}]}{\sum[M^{2+}]} \tag{2.19}$$

$$CoConcRatio = \frac{[Co^{2+}]}{\sum[M^{2+}]} \tag{2.20}$$

so the term $\prod[M^{2+}]^{MConcRatio}$ is the product of the three terms (one for every metal cation), $[OH^{-}]^{2}$ is the concentration of hydroxide anions and K_{sp_NMC} is defined as:

$$K_{sp\_NMC} = (K_{sp\_Ni})^{NiConcRatio}(K_{sp\_Mn})^{MnConcRatio}(K_{sp\_Co})^{CoConcRatio} \tag{2.21}$$

With: $K_{sp\_Ni} = 10^{-15.22}$, $K_{sp\_Mn} = 10^{-12.70}$, $K_{sp\_Co} = 10^{-14.89}$.

2.2.2 Equilibrium equations and cations balance

The supersaturation can be calculated only if the equilibrium equations are solved. The equations useful for this purpose are the equations from the equilibrium of the reactions seen in section 2.2:

Table 2.3 Equilibria equations in coprecipitation

Reaction	Equilibrium
$M_i^{2+} + nNH_3 \leftrightarrow [M_i(NH_3)_n]^{2+}$	$k_n^{(Mi)} = \frac{[M_i(NH_3)_n]^{2+}}{[M_i{}^{2+}][NH_3]^n}$

$NH_3 + H_2O \leftrightarrow NH_4^+ + OH^-$	$k_b = \dfrac{[NH_4^+][OH^-]}{[NH_3]}$
$H_2O \leftrightarrow H^+ + OH^-$	$k_w = [H^+][OH^-]$

In the first row $\sum_{i=1}^{3} n_i^{(max)}$ equations, i.e. the sum on every metal of the maximum number of complexes that can afford (six for Nickel and Cobalt, four for Manganese).

These equations are coupled with the mass balances involving total concentrations:

$$C_{M_i^{2+}}^{total} = [M_i^{2+}] + \sum_{n_i=1}^{n_i^{(\max)}} [M_i(NH_3)_{n_i}]^{2+} \qquad with\ i = 1,2,3 \tag{2.22}$$

These are three equations (one for every metal ion) that show that the total concentration of the metal ion is given by the sum of the concentration of the dissolved ion and the concentration of the ions complexed by ammonia.

$$C_{NH_3}^{total} = [NH_3] + [NH_4^+] + \sum_{i=1}^{3} \sum_{n_i=1}^{n_i^{(\max)}} n_i [M_i(NH_3)_{n_i}]^{2+} \tag{2.23}$$

This equation shows that the total ammonia is given by the sum of the concentrations of ammonia, ammonium ion and the ammonia complexing the metal ions.

The last equation needed is the charge balance:

$$2\sum_{i=1}^{3} C_{M_i^{2+}}^{total} + [NH_4^+] + [H^+] + [Na^+] - 2[SO_4^{2-}] - [OH^-] = 0 \tag{2.24}$$

The term $[Na^+] - 2[SO_4^{2-}]$ is very useful, since represents the charge of inert ions, which is a constant in the system, defined only by the inlet conditions: Na^+ enter only with sodium hydroxide and SO_4^{2-} with the metals sulfates, and both ions do not intervene in any reaction.

All these equations can be rearranged in order to write only equations as a function of five unknowns: $[M_i^{2+}]$ ($[Ni^{2+}]$, $[Mn^{2+}]$, $[Co^{2+}]$), $[NH_3]$, $[OH^-]$.

We obtain five equations for five unknowns:

$$C_{M_i^{2+}}^{total} - [M_i^{2+}] - \sum_{n_i=1}^{n_i^{(\max)}} k_n^{(Mi)} [M_i{}^{2+}][NH_3]^{n_i} = 0 \qquad with\ i = 1,2,3 \tag{2.25}$$

$$C_{NH_3}^{total} - [NH_3] - \frac{k_b[NH_3]}{[OH^-]} - \sum_{i=1}^{3} \sum_{n_i=1}^{n_i^{(max)}} n_i k_n^{(Mi)} [M_i{}^{2+}][NH_3]^{n_i} = 0 \tag{2.26}$$

$$2\sum_{i=1}^{3} C_{M_i^{2+}}^{total} + \frac{k_b[NH_3]}{[OH^-]} + \frac{k_w}{[OH^-]} + [Na^+] - 2[SO_4^{2-}] - [OH^-] = 0 \tag{2.27}$$

This is a system of five non-linear equation which can be solved by using for example the Newton-Rapson method.

2.3 Population balance equation and closure problem

The number density function *n(L,**x**,t)* or particle size distribution (PSD) represents the probability of having a number of particles in a range of internal coordinate (the particle size L in this case):

$$n(L,\boldsymbol{x},t)dL = number\ of\ particles\ per\ unit\ volume\ with\ size\ between\ L\ and\ L + dL$$

The evolution of the PSD can be described by the population balance equation (PBE). The PBE is the balance of the number of particles in an infinitesimal volume, at a certain time, which are characterized by a specific particle size. The PBE reads as follows:

$$\frac{\partial n(L,\boldsymbol{x},t)}{\partial t} + \nabla \cdot \left(\boldsymbol{u} \cdot n(L,\boldsymbol{x},t)\right) = J(S) - \frac{\partial}{\partial L}\left(G(L) \cdot n(L,\boldsymbol{x},t)\right) + B_{agg}(L,\boldsymbol{x},t) - D_{agg}(L,\boldsymbol{x},t) \tag{2.28}$$

In which S is the supersaturation, $J(S)$ is the nucleation rate, $G(L)$ is the molecular growth rate and $B_{agg}(L,\boldsymbol{x},t)$ and $D_{agg}(L,\boldsymbol{x},t)$ are birth and death rates due to aggregation.

Nano-particles suspended in fluids moves with the fluids since their Stokes number values are very small (Gavi et al., 2007). Therefore the velocity appearing the PBE is the fluid velocity.

2.3.1 Method of moments

The PBE is in this work solved by using the method of moments. Using the definition of moment of order q:

$$m_q(\boldsymbol{x},t) = \int_0^{+\infty} n(L,\boldsymbol{x},t) \cdot L^q \, dL \tag{2.29}$$

$$with\, q = 0, 1, 2, 3$$

In order to be solved, Eq. (2.28) can be written in terms of the moments, deriving a continuity equation (zero order moment) and transport equations for mean properties (Marchisio and Fox, 2013):

$$\begin{aligned} \frac{\partial m_q}{\partial t} + \nabla \cdot (\boldsymbol{u} \cdot m_q) \\ = x_c^q \cdot J(S) + \int_0^{+\infty} q \cdot L^{q-1} \cdot G(L) \cdot n(L,\boldsymbol{x},t)\, dL \\ + \int_0^{+\infty} (B_{agg}(L,\boldsymbol{x},t) - D_{agg}(L,\boldsymbol{x},t)) L^q dL \end{aligned} \tag{2.30}$$

In which x_c is the critical size for nucleation.

The moment formulation presents an excess of unknowns with respect to the equations, so this formulation is affected by the closure problem (Dorao and Jakobsen, 2006). This problem is addressed using the quadrature method of moments (QMOM):

$$m_q = \sum_{i=1}^{N} w_i \cdot L_i^q \tag{2.31}$$

Using this method, the closure problem can be overcome and the equation becomes (Marchisio et al, 2003):

$$\begin{aligned} \frac{\partial m_q}{\partial t} + \nabla \cdot (\boldsymbol{u} \cdot m_q) \\ = x_c^q \cdot J(S) + q \cdot \sum_{i=1}^{N} w_i \cdot L_i^{q-1} \cdot G(L_i) \\ + \frac{1}{2} \sum_{i=1}^{N} w_i \sum_{j=1}^{N} w_j (L_i^3 + L_j^3)^{\frac{q}{3}} \beta_{agg} - \sum_{i=1}^{N} w_i L_i^q \sum_{j=1}^{N} w_j \, \beta_{agg} \end{aligned} \tag{2.32}$$

In which β_{agg} is the aggregation kernel, which indicates the frequency of collisions that cause aggregation between particles of size L_i and L_j.

The source terms are now defined with the quadrature method of moments, but now the issue is the calculation of the nodes L_i and the weights w_i.

In Marchisio and Fox (2003) is shown how to deal with this problem. Beginning with the definition of a set of orthogonal polynomials ($\{P_0(L), P_1(L), \dots P_\alpha(L), \dots\}$ with $P_\alpha(L) = k_{\alpha,0}L^\alpha + k_{\alpha,1}L^{\alpha-1} + \cdots + k_{\alpha,\alpha}$) in the integration interval Ω_L:

$$\int_{\Omega_L} n(L)P_\alpha(L)\, P_\beta(L)dL \begin{cases} = 0\, for\ \alpha \neq \beta \\ > 0\, for\ \alpha = \beta \end{cases} \tag{2.33}$$

With $n(L)$ weight function, which together with the integration interval, uniquely define the set of polynomials.

A theorem states that any set of orthogonal polynomials has a recurrence formula that relates three consecutive polynomials:

$$\begin{gathered} P_{\alpha+1}(L) = (L - a_\alpha)P_\alpha(L) - b_\alpha P_{\alpha-1}(L) \quad with\ \alpha = 0, 1, 2, \dots \\ P_{-1}(L) = 0 \quad and\ P_0(L) = 1 \end{gathered} \tag{2.34}$$

And the coefficients a_α and b_α are defined as:

$$a_\alpha = \frac{\int_{\Omega_L} n(L)LP_\alpha(L)\, P_\alpha(L)dL}{\int_{\Omega_L} n(L)P_\alpha(L)\, P_\alpha(L)dL} \quad with\ \alpha = 0, 1, 2, \dots \tag{2.35}$$

$$b_\alpha = \frac{\int_{\Omega_L} n(L)P_\alpha(L)\, P_\alpha(L)dL}{\int_{\Omega_L} n(L)P_{\alpha-1}(L)\, P_{\alpha-1}(L)dL} \quad with\ \alpha = 1, 2, \dots \tag{2.36}$$

This theorem, that shows the recursive relation existing between the polynomials, is the foundation of the use of orthogonal polynomials.

The coefficients a_α and b_α can be written in terms of moments and A quadrature method with N weights and N abscissas (or nodes) can be calculated from 2N moments (Marchisio and Fox, 2013), in this work will be used N = 2 nodes, so the moments will be m_0, m_1, m_2, m_3. From these moments can be calculated the coefficients:

$$a_0 = \frac{m_1}{m_0} \tag{2.37}$$

$$a_1 = \frac{m_3 m_0^2 + m_1^3 - 2m_2 m_1 m_0}{m_2 m_0 + m_1^2 - 2m_1^2 m_0} \tag{2.38}$$

$$b_1 = \frac{m_2 m_0 + m_1^2 - 2m_1^2 m_0}{m_0^2} 1 \tag{2.39}$$

That can be used to calculate the polynomial of order two $P_2(L)$ orthogonal to the weight function, which roots are the nodes of the Gaussian quadrature approximation.

Finding the roots of a polynomial is an ill-conditioned problem, but it can be turned into a well-conditioned problem of finding the eigenvalues and eigenvectors of a tri-diagonal symmetric matrix. This can be obtained from a diagonal similarity transformation that transforms the matrix form of the recursive system of polynomials:

$$L \begin{bmatrix} P_0(L) \\ P_1(L) \\ \vdots \\ P_{\alpha-1}(L) \end{bmatrix} = \begin{bmatrix} a_0 & 1 & & & \\ b_1 & a_1 & 1 & & \\ & b_2 & a_2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & b_{\alpha-1} & a_{\alpha-1} \end{bmatrix} \begin{bmatrix} P_0(L) \\ P_1(L) \\ \vdots \\ P_{\alpha-1}(L) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ P_\alpha(L) \end{bmatrix} \tag{2.40}$$

in a tri-diagonal symmetric Jacobi matrix:

$$\boldsymbol{J} = \begin{bmatrix} a_0 & \sqrt{b_1} & & & \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & & \\ & \sqrt{b_2} & a_2 & \sqrt{b_3} & \\ & & \ddots & \ddots & \ddots \\ & & & \sqrt{b_{\alpha-1}} & a_{\alpha-1} \end{bmatrix} \tag{2.41}$$

The nodes $L_{i,q}$ have been defined above as the roots of the polynomial $P_\alpha(L)$, with α = N (number of nodes). The N weights can be calculated as

$$w_\alpha = m_0 \varphi_{\alpha 1}^2 \tag{2.42}$$

with $\varphi_{\alpha 1}$ equal to the first component of αth eigenvector $\boldsymbol{\varphi_\alpha}$ of the Jacobi matrix.

The solution of this problem, considering the dependence of the coefficients appearing in the Jacobi matrix, in order to find weights $w_{i,q}$ and abscissas $L_{i,q}$, can be determined using specific algorithms, such as the product-difference (PD) algorithm or the Wheeler algorithm.

2.3.2 Product-difference algorithm (PD)

Based on the theory of continued fractions of Stieltjes, is often efficient and can be applied to this case since there are not the conditions that limit this algorithm, i.e. a high value of N or a distribution with zero mean ($m_1 = 0$) (Marchisio and Fox, 2013).

First of all a matrix **P** is defined which components are:

$$P_{\alpha,1} = \delta_{\alpha 1} \quad with\ \alpha = 1, 2, \dots, 2N + 1 \tag{2.43}$$

in which $\delta_{\alpha 1}$ is the Kronecker delta.

$$P_{\alpha,2} = (-1)^{\alpha-1} m_{\alpha-1} \quad with\ \alpha = 1, \dots, 2N \tag{2.44}$$

$$P_{\alpha,\beta} = P_{1,\beta-1} P_{\alpha+1,\beta-2} - P_{1,\beta-2} P_{\alpha+1,\beta-1} \tag{2.45}$$

$$with\ \beta = 3, \dots, 2N + 1 \quad and\ \alpha = 1, \dots, 2N + 2 - \beta$$

For example in our case, with N=2:

$$\boldsymbol{P} = \begin{bmatrix} 1 & m_0 & m_1 & m_0 m_2 - m_1^2 & m_0(m_3 m_1 - m_2^2) \\ 0 & -m_1 & -m_2 & -(m_0 m_3 - m_2 m_1) & \\ 0 & m_2 & m_3 & & \\ 0 & -m_3 & & & \\ 0 & & & & \end{bmatrix} \tag{2.46}$$

Then the coefficients of the continued fraction ζ_α are calculated using a recursive relationship:

$$\zeta_1 = 0 \tag{2.47}$$

$$\zeta_\alpha = \frac{P_{1,\alpha+1}}{P_{1,\alpha} P_{1,\alpha-1}} \quad with\ \alpha = 2, \dots, 2N \tag{2.48}$$

Finally, the coefficients of the symmetric tridiagonal Jacobi matrix are calculated with sum and products of ζ_α:

$$a_\alpha = \zeta_{2\alpha} + \zeta_{2\alpha-1} \quad with\ \alpha = 1, \dots, N \tag{2.49}$$

$$b_\alpha = -\sqrt{\zeta_{2\alpha+1}\, \zeta_{2\alpha}} \quad with\ \alpha = 1, \dots, N - 1 \tag{2.50}$$

2.3.3 Wheeler algorithm

The idea on which this algorithm is based is the use of a different set of basis functions $\pi_\alpha(L)$ to represent the orthogonal polynomials, which better sample the integration interval Ω_L. This has the consequence of increasing the stability of the algorithm (Marchisio and Fox, 2013). This algorithm does not suffer from the issues that affect the PD algorithm.

The modified moments ν_q are calculated analogously to the standard ones:

$$\nu_q = \int_{\Omega_L} \pi_q(L) n(L) dL \quad with\, q = 0, 1, \ldots, 2N-1 \tag{2.51}$$

and from these the coefficients are calculated. It is assumed that $\pi_q(L)$ follows the recursive relation:

$$\pi_{-1}(L) = 0 \tag{2.52}$$

$$\pi_0(L) = 1 \tag{2.53}$$

$$\pi_{\alpha+1}(L) = (L - a'_\alpha)\pi_\alpha(L) - b'_\alpha \pi_{\alpha-1}(L) \tag{2.54}$$

where the coefficients must be known. Using the intermediate quantities:

$$\sigma_{\alpha,\beta} = \int_{\Omega_L} n(L)\, \pi_\alpha(L) \pi_\beta(L) dL \quad with\, \alpha, \beta \geq -1 \tag{2.55}$$

initialized in the following way:

$$\sigma_{-1,\alpha} = 0 \quad with\, \alpha = 1, 2, \ldots, 2N-2 \tag{2.56}$$

$$\sigma_{0,\alpha} = \nu_\alpha \quad with\, \alpha = 0, 1, \ldots, 2N-1 \tag{2.57}$$

$$a_0 = a'_0 + \frac{\nu_1}{\nu_0} \tag{2.58}$$

$$b_0 = 0 \tag{2.59}$$

can be computed that:

$$\begin{gathered} \sigma_{\alpha,\beta} = \sigma_{\alpha-1,\beta+1} - \left(a_{\alpha-2} - a'_\beta\right)\sigma_{\alpha-1,\beta} - b_{\beta-1}\sigma_{\alpha-2,\beta} + b'_\beta \sigma_{\alpha-1,\beta-1} \\ with\, \alpha = 1, 2, \ldots, N-1 \quad and\, \beta = \alpha, \alpha+1, \ldots, 2N-\alpha-1 \end{gathered} \tag{2.60}$$

In the end, the coefficients of the Jacobi matrix are calculated as:

$$a_\alpha = a'_\alpha - \frac{\sigma_{\alpha-1,\alpha}}{\sigma_{\alpha-1,\alpha-1}} + \frac{\sigma_{\alpha,\alpha+1}}{\sigma_{\alpha,\alpha}} \tag{2.61}$$

$$b_\alpha = \frac{\sigma_{\alpha,\alpha}}{\sigma_{\alpha-1,\alpha-1}} \tag{2.62}$$

The algorithm presents higher stability than the PD algorithm even if the modified moments ν_q are chosen equal to the standard ones (so $a'_\alpha = b'_\alpha = 0$).

2.3.4 Adaptive wheeler algorithm

The Wheeler algorithm is able to calculate weights and nodes for a realizable moment set, but if they are unrealizable it will fail.

An adaptive version of the Wheeler algorithm is explained in Yuan and Fox (2011). This version uses the fact that in a realizable set of moments $\sigma_{\alpha,\alpha} > 0$, so the check for realizability is carried by the inequality:

$$\sigma_{\alpha,\alpha} \leq 0 \quad with\, 1 \leq \alpha \leq N \tag{2.63}$$

If at a certain $\alpha = m$ results that $\sigma_{m,m} \leq 0$, then there are only $m - 1$ weights and nodes that can be computed.

This adaptive version is useful in degenerate cases in which the exact number density function is composed of less than N weighted Dirac delta functions; in these cases, the algorithm gives the exact number density function (Marchisio and Fox, 2013).

2.3.5 Realizability of a moment set

If the nodes calculated with the algorithms described above are always within the support Ω_L, they represent realizable values of the internal coordinate (L). If the weights are always positive too, the quadrature used obtains accurate results.

These properties are respected only if the moment set is realizable, that means that exists a number density function which results in that specific set of moments (Marchisio and Fox, 2013). So if the algorithms receive in input an invalid moment set, unrealizable nodes will be calculated and this will be a danger for the stability of the simulation.

2.4 Precipitation kinetics

Precipitation is a complex phenomenon in which many mechanisms take part in the path that lead the solute to crystallize and that determines the particle size distribution of the crystals obtained.

There are three main mechanism considered in this work: nucleation, growth and aggregation, the theory of which will be shown in the next chapters.

2.4.1 Nucleation

Nucleation is the mechanism leading to the formation of new particles during precipitation. There are two main types of nucleation: homogeneous and heterogeneous.

Homogeneous nucleation is induced only by supersaturation and occurs in absence of foreign solid interfaces, such as impurities in the solution: clusters are formed and destructed by spontaneous fluctuations of density and composition. They can only increase or decrease in size by molecular addition and, considering these phenomena as reversible reactions, it can be shown that a critical cluster size exists, at which the clusters are thermodynamically stable and beyond which growth is favoured on decay. This occurs because there are two contributes on the reversible work necessary to form clusters: a volume contribute (to form the bulk of the crystal particle) and a surface contribute (to form the surface of the crystal particle) (Mersmann, 2001). These two terms depend on a third power of cluster dimensions and on a second power respectively, driving to the critical cluster size described before.

The rate at which these nuclei are formed is called nucleation rate and its dependence on supersaturation is shown in the following equation:

$$J_{hom} = J_{hom}^{\infty} \exp\left(-\frac{k}{ln^2 S}\right) \tag{2.64}$$

In which the constants J_{hom}^{∞} and k depend on physical properties of the system and of the substance.

Heterogeneous nucleation occurs due to the combined effect of supersaturation and the presence of solid interfaces (in real applications the solution can never be pure and impurities acting as solid interfaces will be always present). The solid interfaces provide sites where nucleation can easily occur due to reduced energy requirements (Randolph 1971). It can be shown that the dependence on supersaturation of heterogeneous nucleation rate is very similar to that homogenous, since the free energy needed to form the nuclei differs in the two cases of a constant f that can vary from 0 to 1:

$$J_{het} = J_{het}^{\infty} \exp\left(-\frac{f * k}{ln^2 S}\right) \tag{2.65}$$

In which the constants J_{het}^{∞} and f depend on physical properties of the system and of the substance and also on physical properties of the solid interface.

Analysing Eqs. (2.64) and (2.65), it is possible to observe that heterogeneous nucleation is more relevant than homogeneous at lower values of supersaturation. At higher supersaturation instead, homogeneous nucleation becomes the main phenomena leading to creation of nuclei, as shown in figure 2.1.

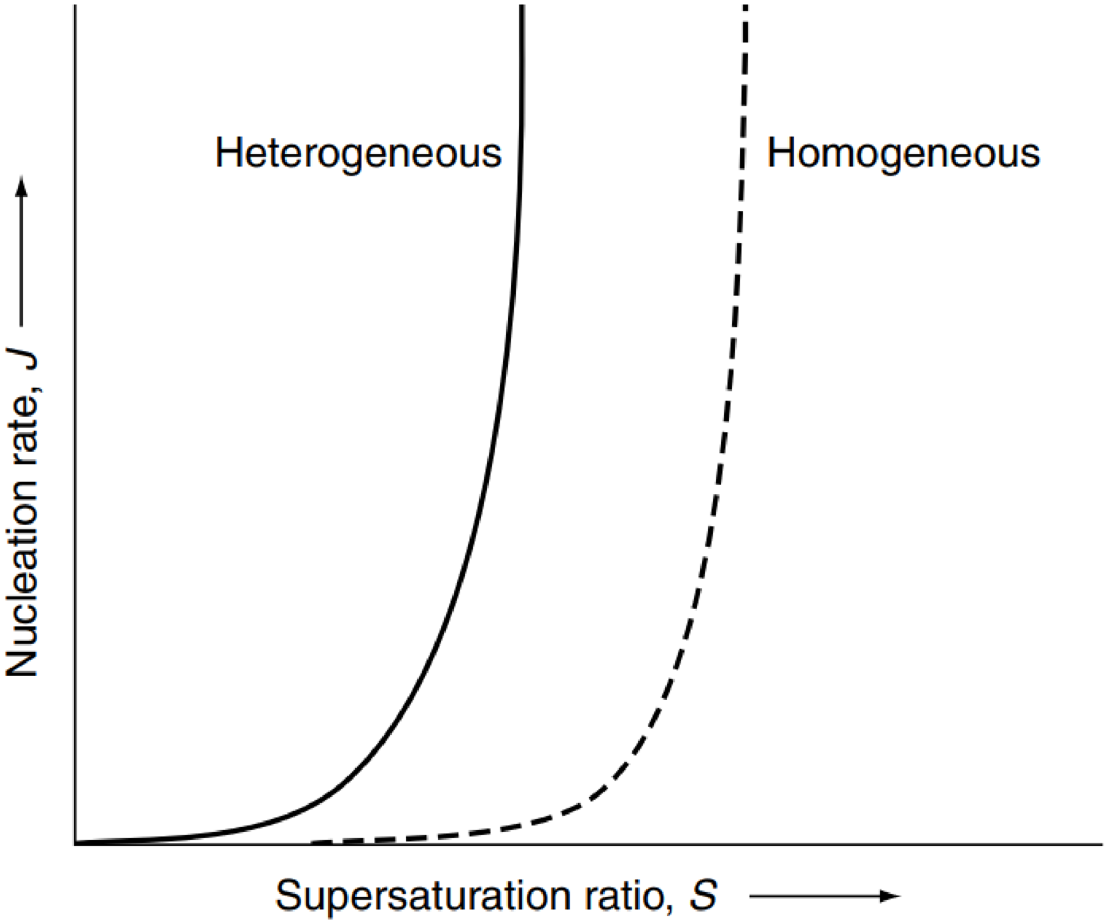


Figure 2.1 Homogeneous and heterogeneous nucleation dependence on supersaturation (Richardson et al, 2002)

2.4.2 Growth

Growth is the mechanism in which molecules transfer from the solution to the surface of the particles, causing size enlargement of the particles. There are two main steps in growth: a diffusional step in which particle moves from the bulk of the solution to the surface of the particle, and a surface integration step in which phase change occurs and the molecule is positioned in the lattice (Mersmann, 2001).

With this consideration, the molar flux per unit of surface (directly linked to the growth rate) can be described as:

$$\dot{n} = k_{mt}\left(C_{bulk} - C_{surface}\right) = k_{si}\left(C_{surface} - C_{eq}\right)^{g} \qquad (2.66)$$

In which k_{mt} is the mass transfer coefficient, k_{si} is the surface integration reaction rate constant, g is the order of the integration reaction, C_{bulk} and $C_{surface}$ are the concentration of solute in bulk and surface and C_{eq} is the equilibrium concentration (solubility). The two phenomena occur in series so their fluxes are put equivalents. If mass transfer is the limiting step, the growth rate depends on the first of the two terms, instead if mass transfer is not limiting, the surface integration rate determines the growth rate.

2.4.3 Aggregation

Aggregation is the mechanism leading to size enlargement of the particles due to the union of pre-existing particles. Contrarily to the two mechanism above described, this mechanism is discontinuous and do not involve the solute in the continuum phase, so it does not change the mass of the two phases.

Aggregation can be cause by Brownian motion (random collisions) and by turbulent flows (collisions caused by turbulent fluctuations). The aggregation kernel for Brownian motion is (Gavi et al, 2007):

$$k_{bm} = \frac{2k_B T}{3\mu} \frac{\left(L_i + L_j\right)^2}{L_i L_j} \tag{2.67}$$

In which k_B is the Boltzmann constant, T is the temperature, μ is the dynamic viscosity and L_i and L_j are the dimensions of the particles involved in aggregation.

The aggregation kernel for turbulent flows is:

$$k_{tf} = 1.29\sqrt{\frac{\varepsilon}{\nu}}\left(L_i + L_j\right)^3 \tag{2.68}$$

In which ε is the turbulent dissipation rate and ν is the kinematic viscosity.

The overall aggregation kernel is corrected using an aggregation efficiency coefficient that takes unsuccessful collision due to inter-particle forces into account (Gavi et al, 2007).

2.5 Species transport equations

Other equations that must be solved for the system are the equations that described the transport of the species. The equations considered are, for metal ions:

$$\frac{\partial C_{M^{2+}}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{M^{2+}}) \ = S_{M^{2+}} \tag{2.69}$$

in this equation we can see a source term $S_{M^{2+}}$, that represents the precipitation mechanism of the cations, turning to solid state and can be expressed as:

$$S_{M^{2+}} = -precRate \times M^{2+}ConcRatio \tag{2.70}$$

In which $M^{2+}ConcRatio$ is defined in Eqs. (2.18), (2.19) and (2.20), and the precipitation rate $precRate$ depends on the amount of crystal that is created: this value is represented by the source term of the third moment (volume of crystal generated per volume unit):

$$precRate = K_v \times (x_c^3 \cdot J(S) + \sum_{j=1}^{2} (3 \times w_j \times G(L_j) \times L_j^2)) \times \frac{\rho crystal}{aMassCrystal} \tag{2.71}$$

There are two more species for which is needed the transport equation: the first one is the ammonia:

$$\frac{\partial C_{NH_3}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{NH_3}) = 0 \tag{2.72}$$

There is no source term in this equation because ammonia is just an intermediate in the precipitation, forming complexes with the metal ions, but at the end it is not consumed but any reaction.

The last equation is the conservation of inert charges:

$$\frac{\partial C_{inert\ charges}}{\partial t} + \nabla \cdot (\boldsymbol{u} C_{inert\ charges}) = 0 \tag{2.73}$$

As seen in chapter 2.2.2 $C_{inert\ charges} = [Na^+] - 2[SO_4^{2-}]$.

2.6 Moment transport equations

The governing equations that describe the evolution of the moments of order q are:

$$\frac{\partial m_q}{\partial t} + \nabla \cdot \left(\boldsymbol{u} m_q\right) = S_{m_q} \tag{2.74}$$

In which the source term S_{m_q} can be written as:

$$\begin{aligned} S_{m_q} = x_c^q \cdot J(S) + q \cdot \sum_{i=1}^{N} w_i \cdot L_i^{q-1} \cdot G(L_i) + \frac{1}{2} \sum_{i=1}^{N} w_i \sum_{j=1}^{N} w_j \left(L_i^3 + L_j^3\right)^{\frac{q}{3}} \beta_{agg} \\ - \sum_{i=1}^{N} w_i L_i^q \sum_{j=1}^{N} w_j \, \beta_{agg} \end{aligned} \tag{2.75}$$

Leading to Eq. (2.32) obtained by the population balance equation. It is useful to see that for q = 3, we obtain the source term for the third moment, which represents the volume of crystal generated and is useful to express the source term for the metal ions (Eq. (2.71)) and is not affected by the aggregation term, since it do not change the total volume of precipitate.

3. Test cases and numerical details

The reactor geometry and mesh were prepared and provided by Umicore: the reactor is a continuous stirred reactor which geometry can not be shown for confidential reasons.

The reactor is continuously fed with ammonia, sodium hydroxide and Nickel, Manganese and Cobalt sulfates. These metals are fed in a 6-2-2 proportion, in order to obtain an NMC hydroxide precipitate with the same proportions of the metals.

In the following chapters the steps used to seek a solution will be discussed: a CFD simulation is needed to obtain the velocity and pressure fields in the reactor; with this information the reactor can be divided using turbulent dissipation rate in a compartment model that is studied to obtain this solution with lower computational costs.

3.1 CFD simulation

The governing equations for fluids seen in section 2.1 are partial differential equations that, excluded some notable case with particular geometries and flow conditions, are complicated functions of time and space and an analytical solution can not be computed.

The solution is sought using discretization methods: discretization consists in transform the partial differential equations, which are continuous functions, are approximated with analogous set of discrete equations that prescribes values in a finite number of points (or volumes) of the domain and that can be solved in an algebraic way (Anderson, 1995). These solutions are therefore numerical solutions in only discrete points of the domain.

The more common discretization techniques are:

- Finite difference method
- Finite element method
- Finite volume method

3.1.1 Finite volumes method

The finite volume method is the one used by Fluent due to its property of guaranteeing the conservation of quantities in each control volume, which is fundamental for most engineering cases.

The steps of this method are (Moukalled et al, 2016):

1. Divide the domain of interest is divided into non-overlapping discrete control volumes (or cells), in the center of which is located the variable
2. Integrate the differential form of the governing equations in each volume to construct the equations for the discrete variables
3. Linearize the discretized equations
4. Solve the linearized equations and find updated values for the variables

The governing equation for a generic scalar variable $\emptyset$ in differential form is:

$$\frac{\partial(\rho\emptyset)}{\partial t} + \frac{\partial(\rho\emptyset \boldsymbol{u}_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial\emptyset}{\partial x_j}\right) + S_\emptyset \tag{3.1}$$

In which the first one is the time dependent term, the second one the convective term, the third the diffusion term and the last the source term.

Considering a stationary case, we can write the equation in integral form for the generic volume V:

$$\int_V \frac{\partial(\rho\emptyset \boldsymbol{u}_j)}{\partial x_j} \mathrm{dV} = \int_V \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial\emptyset}{\partial x_j}\right) \mathrm{dV} + \int_V S_\emptyset \, \mathrm{dV} \tag{3.2}$$

And using the divergence theorem, the volume integrals of convection and diffusion can be replaced with surface integrals:

$$\oint_{\partial V} \rho \emptyset \boldsymbol{u_j} \, \mathrm{dS} = \oint_{\partial V} \Gamma \frac{\partial \emptyset}{\partial x_j} \mathrm{dS} + \int_{V} S_{\emptyset} \, \mathrm{dV} \tag{3.3}$$

In which the surface integral is over all the faces j of the volume considered:

$$\sum_{k\,faces} \left(\oint_{\partial V_k} \left(\rho \emptyset \boldsymbol{u_j} - \Gamma \frac{\partial \emptyset}{\partial x_j} \right) \mathrm{dS} \right) \tag{3.4}$$

As already said, the variable of interest is located at the center of the volume V. This means that for the volume integral, we can use the value in the center itself in a one point Gaussian quadrature integration with a second order accuracy (Moukalled et al, 2016), or more simply:

$$\int_{V} S_{\emptyset} \, \mathrm{dV} = S_{\emptyset} V \tag{3.5}$$

The surface integrals can instead be approximated using:

$$\oint_{\partial V_k} \left(\rho \emptyset \boldsymbol{u_j} - \Gamma \frac{\partial \emptyset}{\partial x_j} \right) \mathrm{dS} = \oint_{\partial V_k} f \, \mathrm{dS} = f_k S_k \tag{3.6}$$

So it is approximated multiplying the value of f in the center of the face (f_k) by the surface of the face itself (like for volume integrals, a one point Gaussian quadrature integration that gives a second order accurate approximation) (Moukalled et al, 2016). The value of the function f in the center of the face is not known, so a method to estimate it is needed.

3.1.2 Spatial discretization interpolation

In the previous chapter we have seen that the discrete values of the scalar $\emptyset$ are stored in the center of the cells, but for convection terms we need the value at the center of the faces and then a way to interpolate this value using the cell center values.

The schemes used to interpolate these values in this work are two upwind schemes: first order and second order upwind.

First of all is useful to define the mean of the word upwind: an upwind scheme has the characteristic of deriving the value of the scalar at the face using the value of the cell "upwind", that means the upstream cell relatively to the direction of the normal velocity $\boldsymbol{u_j}$.

In first order upwind, it is assumed that the face value $\emptyset_{\boldsymbol{f}}$ is equal to the upstream cell value $\emptyset_{\boldsymbol{c}}$. Considering a Taylor expansion centered in the upstream cell:

$$\rho \boldsymbol{u_j} \emptyset_{\boldsymbol{f}} = \rho \boldsymbol{u_j} \emptyset_{\boldsymbol{c}} + \rho \boldsymbol{u_j} (\boldsymbol{x_f} - \boldsymbol{x_c}) \left. \frac{\partial \emptyset}{\partial \boldsymbol{x}} \right|_{\boldsymbol{c}} \tag{3.7}$$

In which orders higher than one have been neglected. It is possible to see that using this interpolation scheme a numerical error is committed which is proportional to $\rho \boldsymbol{u_j} (\boldsymbol{x_f} - \boldsymbol{x_c})$, which can be written using cell dimension Δx as $\frac{\rho \boldsymbol{u_j} \Delta x}{\mathbf{2}}$ that is also called numerical diffusion, due the fact that it alters the value of the diffusion coefficient.

The second order upwind has second order accuracy and the face value is computed using the expression (ANSYS Fluent theory guide 15, 2013):

$$\emptyset_{\boldsymbol{f}} = \emptyset_{\boldsymbol{c}} + \nabla \emptyset \cdot \boldsymbol{r} \tag{3.8}$$

In which $\nabla \emptyset$ is the gradient of in the upstream cell and $\boldsymbol{r}$ is the displacement vector from the upstream cell center to the face, so the gradient in the cell is considered equal to the gradient in the upstream cell.

3.1.3 Pressure-velocity coupling

The solver used is pressure-based: this means that continuity is guaranteed by solving a pressure (or pressure correction) equation derived by continuity and momentum equations. The pressure-based solver used is the segregated one, that means that equations, which are non-linear and coupled, are solved one after another, in a "segregated" way (ANSYS Fluent theory guide 15, 2013) , as can be seen in figure 3.1.

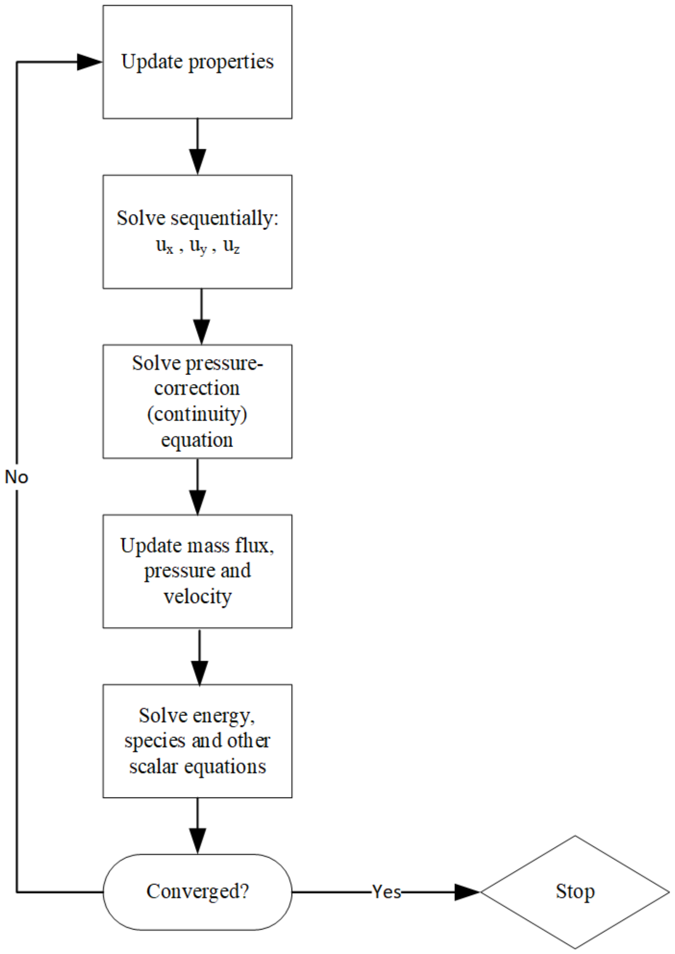

Update properties
Solve sequentially: u_x , u_y , u_z
Solve pressure-correction (continuity) equation
Update mass flux, pressure and velocity
Solve energy, species and other scalar equations
Converged?
Yes
Stop
No

Figure 3.1 Pressure-velocity coupling scheme from ANSYS Fluent theory guide 15

The algorithm used is SIMPLE (Semi-Implicit Method for Pressure Linked Equations) which solves the momentum equation with a guessed pressure field, compute the corresponding face flux, which does not satisfy the continuity equation and then is corrected and with this correction the pressure correction equation is solved and the corrected fields can be computed.

3.1.4 Moving reference frame

Moving reference frame theory is described in ANSYS Fluent theory guide 15: by default ANSYS Fluent solves the governing equations in a stationary reference frame. When moving parts are involved, it is more advantageous to solve these equations in a moving reference frame. That's because moving parts can make the problem unsteady if viewed from a stationary frame, while in a moving reference frame the flow near the moving parts can be modelled as a steady state problem (respect to the moving frame).

Being the reactor agitated, the fluid has rotational components that require this approach.

Using the moving reference frame, the equations of motion are transformed so that steady state solutions are achievable, including additional acceleration terms from the transformation from stationary to moving reference frame.

The equations of motion of the moving reference frame include additional terms of acceleration of the fluid. These equations can be expressed using either relative or absolute velocities as dependent variables.

For this job, absolute velocity formulation has been chosen because the choose between the two formulations is made based on most of the flow domain having the smallest velocities in that frame, in order to obtain a more accurate solution. So, such in our case, if most of the domain belongs to the subdomain that is stationary, the absolute velocity formulation is preferred.

Considering a reference system that rotates with angular velocity $\boldsymbol{\omega}$ and translates with velocity $\boldsymbol{v_t}$ with respect to the stationary reference frame, the following variables are defined:

$\boldsymbol{v_r} = \boldsymbol{v} - \boldsymbol{u_r}$ transformation of the velocities from the stationary frame to the moving frame, where $\boldsymbol{u_r} = \boldsymbol{v_t} + \boldsymbol{\omega} \times \boldsymbol{r}$.

$\boldsymbol{r}$ position vector from the origin of the moving system to the point considered

$\boldsymbol{v}$ absolute velocity, from the point of view of the stationary frame

$\boldsymbol{v_r}$ relative velocity, from the point of view of the moving frame

$\boldsymbol{u_r}$ velocity of the moving frame relative to the stationary one

Then the governing equations solved are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{v_r} = 0 \qquad \textit{conservation of mass} \tag{3.9}$$

$$\begin{aligned}\frac{\partial}{\partial t}\rho \boldsymbol{v} + \nabla \cdot (\rho \boldsymbol{v_r} \boldsymbol{v}) + \rho[\boldsymbol{\omega} \times (\boldsymbol{v} - \boldsymbol{v_t})] \\ = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F} \quad \textit{conservation of momentum}\end{aligned} \tag{3.10}$$

3.1.5 Multiple reference frame (MRF)

When zones rotating with different rotational speeds are present, the steady state approximation of the MRF model could be used. It consists in the division of the fluid zone in cell zones (subdomains), each rotating at his own rotational speeds. These zones are separated by interfaces boundaries and are solved with the moving reference frame equations.

If the velocity is formulated in absolute terms, at the interface between subdomains no transformations are needed, because the velocities of the governing equations in each subdomain are stored in the absolute frame.

In the MRF model the mesh remains fixed during the computation (ANSYS Fluent theory guide 15).

3.1.6 Simulation setup

The pressure and velocity fields are calculated using ANSYS Fluent. The solver is pressure-based and a steady solution is sought.

The turbulence model used is the standard k-ε and the profile near walls is calculated using the standard wall functions.

Around the rotating parts of the reactor, a rotating cell zone is defined with the same angular velocity.

The inlet fluxes are given by Umicore. The inlets are set as velocity inlets: the velocity is specified with a direction normal to the boundary and a magnitude calculated using the flux of the inlet itself and knowing the area of inlet. Turbulence conditions at inlets is specified with hydraulic diameter (known from geometry) and turbulence intensity ratio.

Outlet is set as a pressure outlet and is necessary to use, in order to minimize reverse flow issues, the option "target mass flow rate", set as the sum of the inlets flows (since stationary solution is sought).

In the following tables are resumed the settings and specified numerical values of what above said.

Solver	Pressure-based
Solution	Steady
Gravity	-9.81 $\frac{m}{s^2}$
Turbulence model	Standard k- ε
Near wall treatment	Standard wall functions
Inlets	Velocity-inlet
Outlet	Pressure-outlet

Table 3.1 CFD simulation set-up

The turbulent intensity at inlets is defined of 1% because the Reynold numbers at inlets are low:

$$Re_{Me} = \frac{v_{Me} * D}{\nu} \tag{3.11}$$

$$Re_{\mathrm{NH_3}} = \frac{v_{\mathrm{NH_3}} * D}{\nu} \tag{3.12}$$

$$Re_{\mathrm{NaOH}} = \frac{v_{\mathrm{NaOH}} * D}{\nu} \tag{3.13}$$

The first iterations are carried with the following settings:

Pressure velocity coupling	Scheme	SIMPLE
Spatial discretization	Gradient	Least square cell based
	Pressure	Standard
	Momentum	First order upwind
	Turbulent kinetic energy	First order upwind
	Turbulent dissipation rate	First order upwind

Table 3.2 CFD simulation first iterations schemes set-up

With warped-face gradient correction activated.

When residuals are no more lowering enough, the settings are changed to:

Pressure velocity coupling	Scheme	SIMPLE
Spatial discretization	Gradient	Least square cell based
	Pressure	Second order
	Momentum	Second order upwind
	Turbulent kinetic energy	Second order upwind
	Turbulent dissipation rate	Second order upwind

Table 3.3 CFD simulation last iterations schemes set-up

Since high oscillations have been observed in residuals of turbulent kinetic energy, in order to reduce this issue the initial iterations are carried with high order term relaxation, which are improved going on with the iterations and removed when the residuals are steady enough with high relaxation factors of this tool. When removed this option, the simulation is carried on until the residuals went under the desired threshold shown in table 3.4.

Variable	Absolute residual threshold
continuity	1e-5
momentum	1e-4
Turbulent kinetic energy	1e-4
Turbulent dissipation rate	1e-4

Table 3.4 Residuals of CFD simulation

3.3 Python script

3.3.1 Purpose of the script

Python is an interpreted programming language with high-level data structures. It is object-oriented programming.

A script is a computer program, written in an interpreted language, with low complexity and quite linear, without a graphic interface and that uses libraries to implement more sophisticated functions.

The use of Python for writing this script is due the fact that it is suitable for scripting development and it can have access to a collection of libraries which are really useful for the purpose of the work, for example two libraries are often used in the script: SciPy and NumPy. They contain useful scientific modules (for example the solver for ordinary differential equations).

The purpose of the work is to solve the equations governing the system (sections 2.5 and 2.6) in a simplified way, which will lead to less computational cost and time spent in simulation than a full coupled CFD-PBE simulation. With full coupled CFD-PBE simulation the simulation is carried cell by cell, but this means a long simulation with high computational costs, because there is a huge number of cells and in every single cell Fluent has to solve the equations for concentrations and moments for every iteration.

In the following chapters will be described the method used in this Thesis to seek for the precipitation solution by decreasing the computational costs of the operation.

3.3.2 Compartment model

The basic idea to decrease the computational costs of the simulation is to agglomerate the cells with similar characteristics (characteristics to be discussed in section 4), creating compartments. These compartments behave like CSTR reactors, so the values in each of them are homogeneous, and are connected between them, interchanging fluxes.

With such a division the precipitation equations can be solved in a small number of compartments, instead of solving them in a huge number of cells using ANSYS Fluent, describing the precipitation model and its equations in an UDF and carrying the calculations in every cell of the domain. This would considerably decrease the computational costs and time required for the simulation, considering that the number of cells is in the order of magnitude of millions, while the number of compartments is in the order of magnitude of tens (and the solution of a system of equations does not increase linearly with the number of interconnected elements, but much faster).

In order to carry the calculations to solve this system of equations, a Python script is written, that can be easily editable, improvable and manageable.

The script solves the equations described in sections 2.5 and 2.6 for every compartment. These equations should be modified to keep into account that the compartments are perfectly mixed and linked between them. Considering $Q^{i,j}$ as the volumetric flux from the compartment i to the compartment j with volume V^j, the precipitation equations can be written as follows:

$$\frac{dC^j_{M^{2+}}}{dt} = \sum_i \frac{Q^{i,j}}{V^j} C^i_{M^{2+}} - \sum_i \frac{Q^{j,i}}{V^j} C^j_{M^{2+}} + S^j_{M^{2+}} \tag{3.14}$$

$$\frac{dC^j_{NH_3}}{dt} = \sum_i \frac{Q^{i,j}}{V^j} C^i_{NH_3} - \sum_i \frac{Q^{j,i}}{V^j} C^j_{NH_3} \tag{3.15}$$

$$\frac{dC^j_{inert\ charges}}{dt} = \sum_i \frac{Q^{i,j}}{V^j} C^i_{inert\ charges} - \sum_i \frac{Q^{j,i}}{V^j} C^j_{inert\ charges} \tag{3.16}$$

$$\frac{dm^j_k}{dt} = \sum_i \frac{Q^{i,j}}{V^j} m^i_k - \sum_i \frac{Q^{j,i}}{V^j} m^i_k + S^j_{m_k} \tag{3.17}$$

In figure 3.2 can be seen a schematic example of a compartment model.

The crux of this method are the characteristics previously mentioned used to divide the reactor in compartments. The division itself of the reactor can be achieved using a tool given by ANSYS Fluent, as will be explained in next chapter.

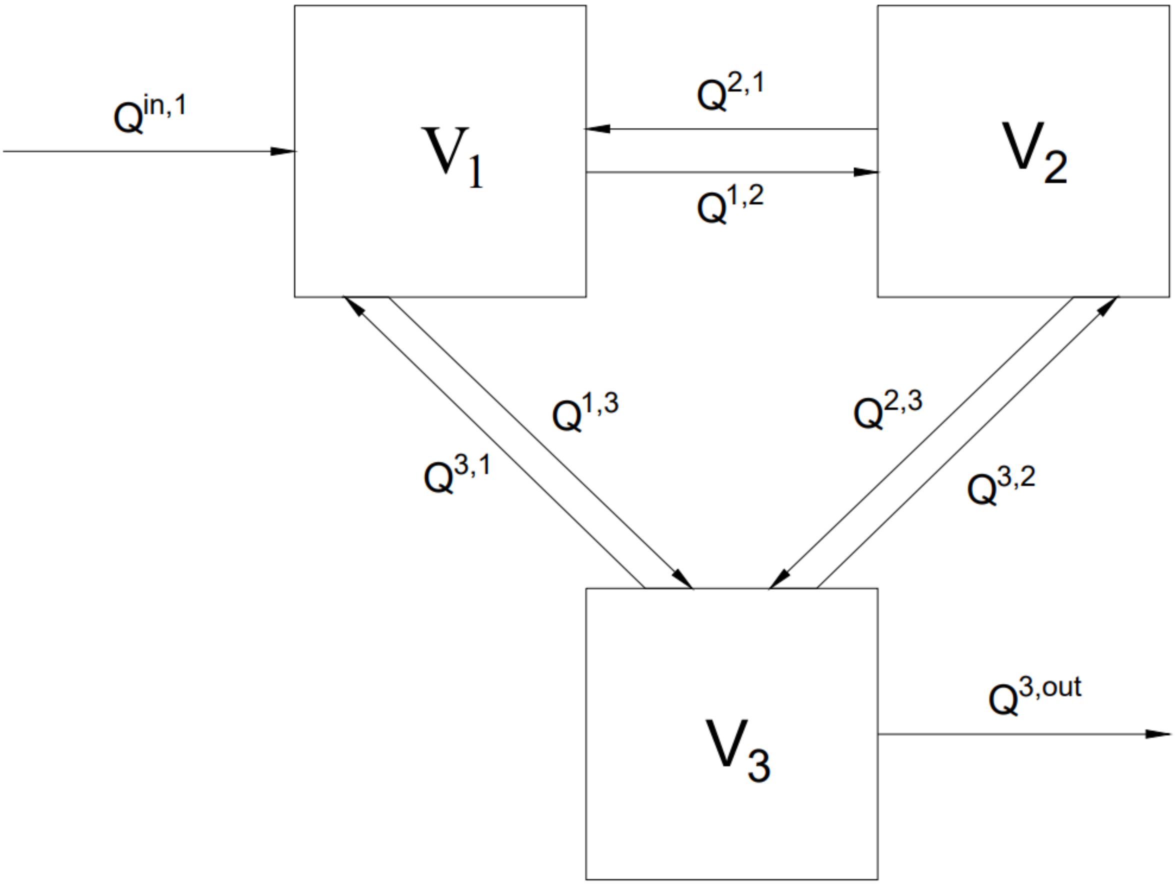


Figure 3.2 Schematic example of compartment model

3.3.3 Reactor network model

ANSYS Fluent provides a tool called Reactor Network Model that is used to divide the reactor in compartments, agglomerating cells of the domain with a certain criterion.

These compartments are linked between them and interchanging fluxes between neighbour reactors.

Looking at the kinetics (chapter 3.3.10), it can be seen that there are two parameters that vary in space that are involved in the calculation of the kinetics: supersaturation and turbulent dissipation rate. Then, in order to predict as accurately as possible the precipitation phenomena, it is important to create compartments agglomerating cells in which these parameters are as similar as possible, and divide the zones with high differences in this parameters.

Ansys Fluent allows to choose multiples Custom Field Functions in the Reactor Network Tool to divide the reactor. The other input given to the tool is the number of desired compartments. Further information about the division carried will be shown in the results section.

Once this step is completed, the following step is to obtain all the information of interest from these compartments. The information needed can be seen in equations (3.14), (3.15), (3.16) and (3.17): the volumes of the compartments and the fluxes between the compartments (including the inlet fluxes and the outlet ones) and the concentrations of the species at the inlets. In addition to these, it is necessary, in order to calculate the aggregation terms, to obtain the average of the turbulent dissipation rate in every compartment. All this information is not given by default by the Reactor Network Tool: an User-Defined Function is used to extract this information from Fluent once the compartments are created. The fluxes and the turbulent dissipation rate between the compartments are known since the division is carried after the CFD simulation. So the UDF extract all this information and write them in *.txt* files, which will be described in chapter 3.3.6.

3.3.4 User-defined function

A user-defined function (UDF) is an additional function present in Fluent and is directly programmed by the user. It can be loaded in the ANSYS Fluent solver in order to enhance and customize some of its features.

They must be written in C programming language (and need to be saved in .c extension). They are coded using additional macros and functions given by ANSYS Fluent (ANSYS Fluent UDF Manual 15, 2013) that access to the solver data to perform the required task.

The UDF files can be interpreted or compiled (ANSYS Fluent UDF Manual 15, 2013):

-if they are interpreted, they are interpreted and loaded directly at runtime, in a single step. The source code is compiled into an intermediate, architecture-independent machine code that executes on an interpreter when the UDF is invoked;

-if they are compiled, the steps are two: first a shared object library is built and then it is loaded to Fluent. They are built in the same way that Fluent executable is. The object code library contains the native machine language translation of the C source code. They are specific to the computer architecture being used.

3.3.5 Organization of the script

The goal of the script is to receive the information of the compartments described in the previous chapter, process them, solving mass balances and the precipitation equations for each compartment, in order to obtain the concentrations and the moments in every compartment at stationary, as described schematically in figure 3.3.

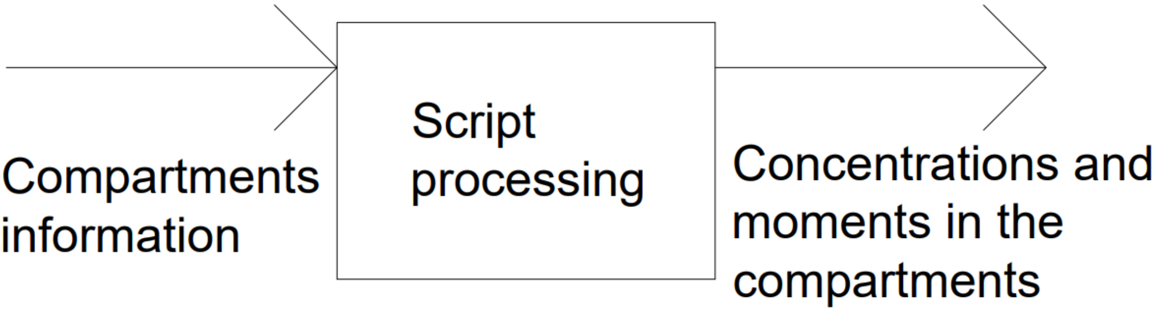


Figure 3.3 inputs and outputs of the script

The script is composed by many files, each with a specified function, to obtain a script that is more legible and easier to modify if needed.

The main files that take part to the calculations are:

- RunPrecSolver.py: the main file, which is run to start the calculations
- Init_run.py: initialize the settings of the system and read the data
- ChemicalEquilibria.py: contains the Newton-Raphson solver
- MomentCalc.py: contains algorithms to calculate weights and nodes of QMOM
- NiMnCoHydroxidePrec.py: contains the ODE solver

Besides these files, some other files are present, containing functions that calculate kinetic terms, post-processing files (to read the results, elaborate them, plot them, calculate overall mass error or compartments mass errors) or just containing support functions useful to streamline and make the script fluid.

3.3.6 File init_run

The function of this file is to set up all the information needed for the solver (such as physical parameters, settings and compartments information), reading them from the appropriate files and rearranging them as needed.

The files containing this information are five, in .txt format: *caseSetup*, *constantDict*, *react_zone_flux_toBoundary*, *react_zone_ave* and *react_zone_flux*.

In the *caseSetup* file are written all the settings for the set up of the script; most of the information contained here will be described below, since much of it is read and fed to the script through the *init_run* file.

From *caseSetup* are read the options needed for the differential equation solver, which will be passed to the solver function as parameters. Further details about these parameters can be found in chapter 3.3.7.

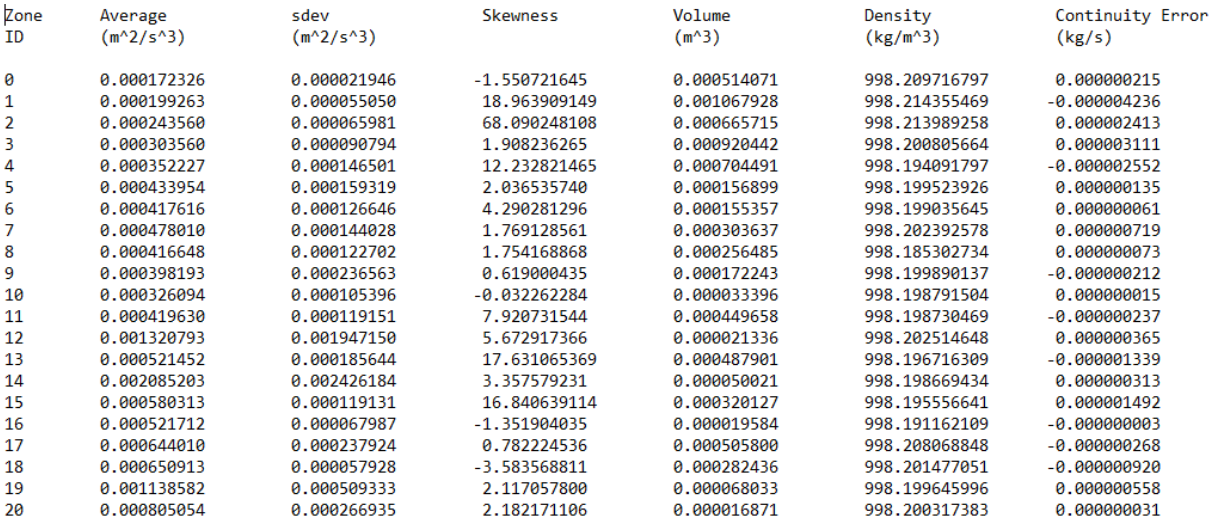
Still in this file is written information useful to the equations of moments, but the only one loaded in *init_run* is the number of nodes used for quadrature approximation. The other information (containing kinetics parameters) is acquired by another function (see chapter 3.3.7).

At least from *caseSetup* are taken the crystals density, the fluid density and a dictionary with the parameters for the Newton-Raphson solver (maximum number of iterations and tolerance, see chapter 3.3.8).

In the *constantDict* file is written a dictionary in which *init_run* finds the atomic masses of Nickel, Manganese and Cobalt.

At last must be read all the information about the compartments, inlets and outlets, and store that in functional arrays to use them in an efficient way. when needed adapt this in a form that will result optimal for the script. All these are in the three .txt files (*react_zone_flux_toBoundary*, *react_zone_ave* and *react_zone_flux*) created with the UDF described in chapter 3.3.3 from the compartment creation in ANSYS Fluent.

In *react_zone_ave* can be found the volumes (m^3) and the turbulent dissipation rate (m^2/s^3) of every compartment, which are stored in two arrays, with a correspondence between the compartment to which the stored value belongs and the position in the array.

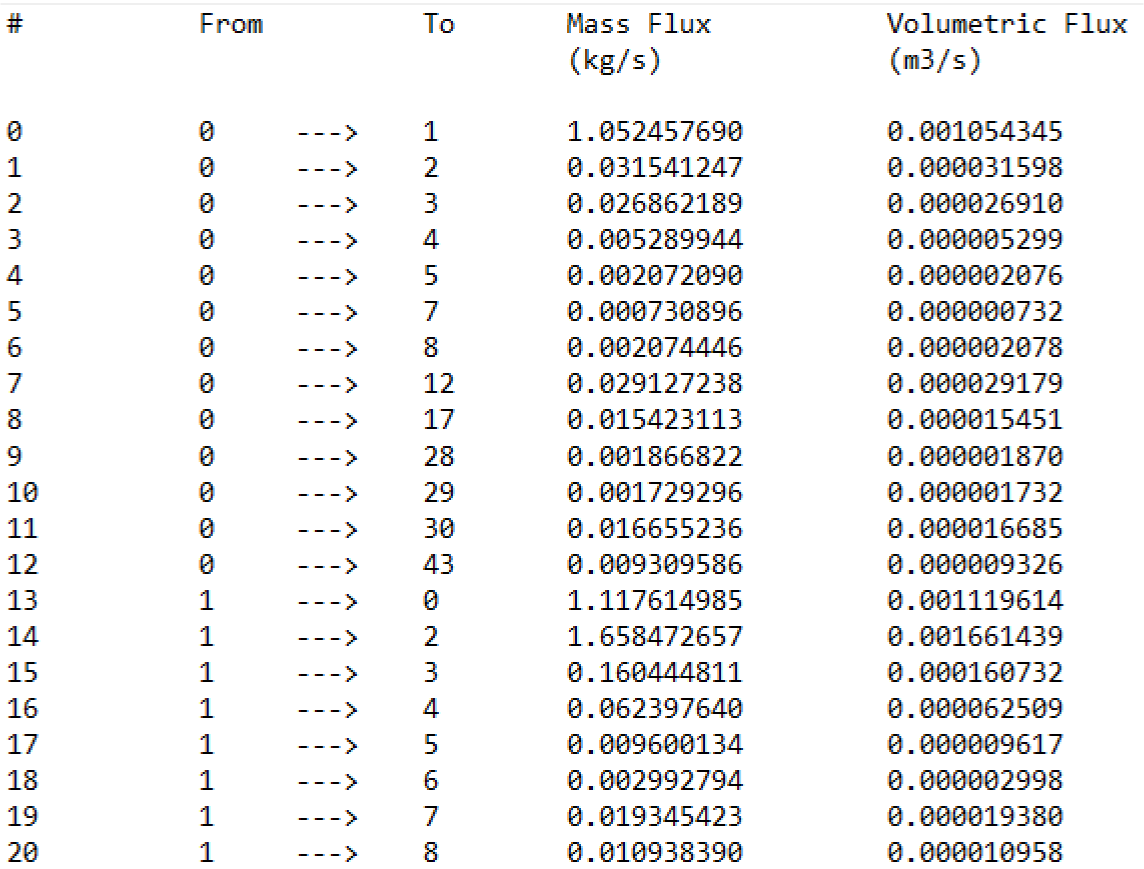


```
Zone      Average             sdev                Skewness            Volume              Density             Continuity Error
ID        (m^2/s^3)           (m^2/s^3)                               (m^3)               (kg/m^3)            (kg/s)

0         0.000172326         0.000021946        -1.550721645         0.000514071         998.209716797       0.000000215
1         0.000199263         0.000055050         18.963909149        0.001067928         998.214355469      -0.000004236
2         0.000243560         0.000065981         68.090248108        0.000665715         998.213989258       0.000002413
3         0.000303560         0.000090794         1.908236265         0.000920442         998.200805664       0.000003111
4         0.000352227         0.000146501         12.232821465        0.000704491         998.194091797      -0.000002552
5         0.000433954         0.000159319         2.036535740         0.000156899         998.199523926       0.000000135
6         0.000417616         0.000126646         4.290281296         0.000155357         998.199035645       0.000000061
7         0.000478010         0.000144028         1.769128561         0.000303637         998.202392578       0.000000719
8         0.000416648         0.000122702         1.754168868         0.000256485         998.185302734       0.000000073
9         0.000398193         0.000236563         0.619000435         0.000172243         998.199890137      -0.000000212
10        0.000326094         0.000105396        -0.032262284         0.000033396         998.198791504       0.000000015
11        0.000419630         0.000119151         7.920731544         0.000449658         998.198730469      -0.000000237
12        0.001320793         0.001947150         5.672917366         0.000021336         998.202514648       0.000000365
13        0.000521452         0.000185644         17.631065369        0.000487901         998.196716309      -0.000001339
14        0.002085203         0.002426184         3.357579231         0.000050021         998.198669434       0.000000313
15        0.000580313         0.000119131         16.840639114        0.000320127         998.195556641       0.000001492
16        0.000521712         0.000067987        -1.351904035         0.000019584         998.191162109      -0.000000003
17        0.000644010         0.000237924         0.782224536         0.000505800         998.208068848      -0.000000268
18        0.000650913         0.000057928        -3.583568811         0.000282436         998.201477051      -0.000000920
19        0.001138582         0.000509333         2.117057800         0.000068033         998.199645996       0.000000558
20        0.000805054         0.000266935         2.182171106         0.000016871         998.200317383       0.000000031
```

Figure 3.4 example of react_zone_ave file

With the information in *react_zone_flux*, two storages are created: one array containing the couples ('From' and 'To') of the IDs that are connected by fluxes, and an array containing the values of the fluxes itself (read as kg/s and converted to m^3/s using density), which position in the array correspond to the flux between the compartments in the respective position of the previous array.

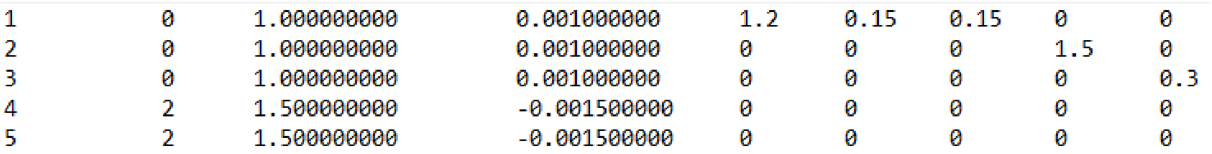


```
#           From           To       Mass Flux           Volumetric Flux
                                    (kg/s)              (m3/s)

0           0     --->     1        1.052457690         0.001054345
1           0     --->     2        0.031541247         0.000031598
2           0     --->     3        0.026862189         0.000026910
3           0     --->     4        0.005289944         0.000005299
4           0     --->     5        0.002072090         0.000002076
5           0     --->     7        0.000730896         0.000000732
6           0     --->     8        0.002074446         0.000002078
7           0     --->     12       0.029127238         0.000029179
8           0     --->     17       0.015423113         0.000015451
9           0     --->     28       0.001866822         0.000001870
10          0     --->     29       0.001729296         0.000001732
11          0     --->     30       0.016655236         0.000016685
12          0     --->     43       0.009309586         0.000009326
13          1     --->     0        1.117614985         0.001119614
14          1     --->     2        1.658472657         0.001661439
15          1     --->     3        0.160444811         0.000160732
16          1     --->     4        0.062397640         0.000062509
17          1     --->     5        0.009600134         0.000009617
18          1     --->     6        0.002992794         0.000002998
19          1     --->     7        0.019345423         0.000019380
20          1     --->     8        0.010938390         0.000010958
```

Figure 3.5 Example of react_zone_flux file

The information in *react_zone_flux_toBoundary* is about the real inlets and outlets of the reactor. This information is stored in an array containing the IDs relative to the compartment in which they are present and their fluxes and concentrations of metals and ammonia. This information is further processed: the objective is to create four vectors, containing respectively the compartments IDs of the inlets, those of the outlets, the fluxes and the concentrations of the inlets and the concentrations of the outlets. All this can be easily done by checking the sign of the flowrates and knowing that inlets and outlets are characterized by having as a convention the same 'From' and 'To' compartments ID in the file *react_zone_flux*, in order to distinguish between inlets and outlets. For the outlets, the flowrates are saved in their absolute values.



```
1         0      1.000000000         0.001000000     1.2     0.15     0.15     0       0
2         0      1.000000000         0.001000000     0       0        0        1.5     0
3         0      1.000000000         0.001000000     0       0        0        0       0.3
4         2      1.500000000         -0.001500000    0       0        0        0       0
5         2      1.500000000         -0.001500000    0       0        0        0       0
```

Figure 3.6 Example of react_zone_flux_toBoundary file

An on-off flag in *caseSetup* indicates if the pH is kept constant or not and this defines the function that will be used in the Newton-Raphson solver. If the constant pH is off, the column with the NaOH concentrations of the inlets is read in *react_zone_flux_toBoundary* and the flowrate weighted average concentration of NaOH is calculated and appended to those of the metals (the case in exam has not constant pH, so the NaOH is considered).

In the end a vector containing the cations concentrations ratio on the total of the inlets is defined, in which the cation concentration ratio is defined as in equations 2.4, 2.5 and 2.6.

From this the atomic mass a.m. of crystals is calculated as follows:

$$a.m. = \sum_{metal} (ConcentrationRatio_{metal} \cdot a.m._{metal}) + 2 \cdot (a.m._{oxygen} + a.m._{hydrogen}) \tag{3.18}$$

3.3.7 File NiMnCoHydroxidePrec

This script is the core of the solver. Here can be found the ordinary differential equation solver and the Newton-Raphson and moment calculator algorithms are imported to update the equations of the system every iteration.

An important aspect to underline is that the solution to be sought is stationary but to find it a solver of differential equations is used which solves an initial value problem (and therefore time dependent). This is done by using an ODE solver contained in the Scipy library and therefore already optimized: it is sufficient to wait for the solution to continue over time until it reaches a steady state.

The function ‘odeEqs’ defines the equations governing the phenomena in order to give them to the solver. The function receives in input the vector y, which is the previous solution of the solver (concentrations and moments) at the time t of iteration.

The values of y corresponding to the concentrations are used to calculate the supersaturation, calling the Newton-Raphson solver function contained in the file *ChemicalEquilibria.py* (see section 3.3.8).

Then the weights and the nodes of the quadrature method are calculated, calling the algorithm from the file *MomentCalc.py* (see chapter 3.3.9).

With this information, using the kinetics models contained in the files described in section 3.3.10 and read by an appropriate function, nucleation rate, growth rate and aggregation rate are computed.

At this point the script has all the necessary in order to define the equations that have to be solved (chapters 2.5 and 2.6). The aim is to define the right side of equation (3.19) ($f(t, y)$). The only terms missing at this point to complete the definition are the convection ones, that consist in fluxes between the compartments and real inlets and outlets.

Before adding the convection terms, all these calculations are carried looping over all the compartments (each with its own solution y).

Now the equations are stored in an array, which is long nine times the number of compartments (every compartment has the nine equations for the nine variables stored), so in the first nine positions are found the equations of the compartment with ID=0 and so on. These equations lack of the convection term that will be added at every compartment. Exploiting the arrays created in the file *init_run.py* containing all information about the volumes of the compartments and the fluxes present in every compartments, both the fluxes between compartments and the real inlets and outlets of the reactor, it is possible to add the missing convective terms.

It can be useful to remind that the compartments are considered as CSTR systems, so the inlet flowrates in a compartment are multiplied by the outlet concentrations/moments of the origin compartment (see Eqs. (3.14), (3.15), (3.16) and (3.17)).

The vector containing the equations for all the compartments is now defined and the solver has its inputs ready.

The solver used the solve_ivp function of the Scipy library, that solves an initial value problem (IVP) for systems of ordinary differential equations (ODE):

$$\frac{dy}{dt} = f(t, y) \tag{3.19}$$

$$y(t_0) = y_0 \tag{3.20}$$

In which $f(t, y)$ is the vector containing the equations for every compartment, created in the function mentioned above, y_0 is the initial condition (that for every compartment is the one calculated in the *init_run* script).

The solver aims to find the value of y(t) that satisfies the differential equations, integrating numerically the system.

Then the parameters given to the solver are: the array containing the equations and, from the information read in the file *init_run.py*, the initial and final time, the initial condition and the solver options. This options are: the integration method used by the solver (explicit Runge-Kutta RK45 method has been chosen), the relative and absolute tolerance (the absolute tolerance is a vector since it depends on the order of magnitude of the variable of interest), the initial step size (in time) and the maximum step size that the solver can use (since the solver adapts by itself the step size that it calculates as optimal). The maximum step size allowed changes in every interval of integration and assumes the values shown in table 3.5:

Final time	1e-13	1e-11	1e-9	1e-7	1e-5	1e-3	1e-1	1e+1	1.5e+5
Max step allowed	1e-15	1e-13	1e-11	1e-9	1e-7	1e-5	1e-3	1e-1	1e+1

Table 3.5 Final times and their time steps

At the beginning of the simulation must be used time steps small enough to do not have numerical errors of the solver, since initially the variables in the compartments change very fast. For a similar reason, the aggregation terms are calculated only after a certain time (about one thousandth of the final time) since with the initial moment distribution it can give some convergence error to the solver.

In the end one last parameter is given, called ‘events’, which is meant to stop the solver when we think that the solution has reached a stationary state. The solver reads the output of this function and if this output reaches zero, the solver stops its calculations. The output of this function is designed as:

$$output = \max\left(abs\left(\frac{f(t,y)}{y}\right)\right) - tolerance \tag{3.21}$$

So, when the maximum of the absolute values of the division between the time derivative of the differential equation (i.e. those calculated in the function ‘odeEqs’) and the values of the solution y at that iteration reaches the value of $tolerance$, the solver stops its iterations, because it means that the time derivative is enough low to be considered stationary. The division with y is needed because the solution values (moments or concentrations) have different order of magnitude.

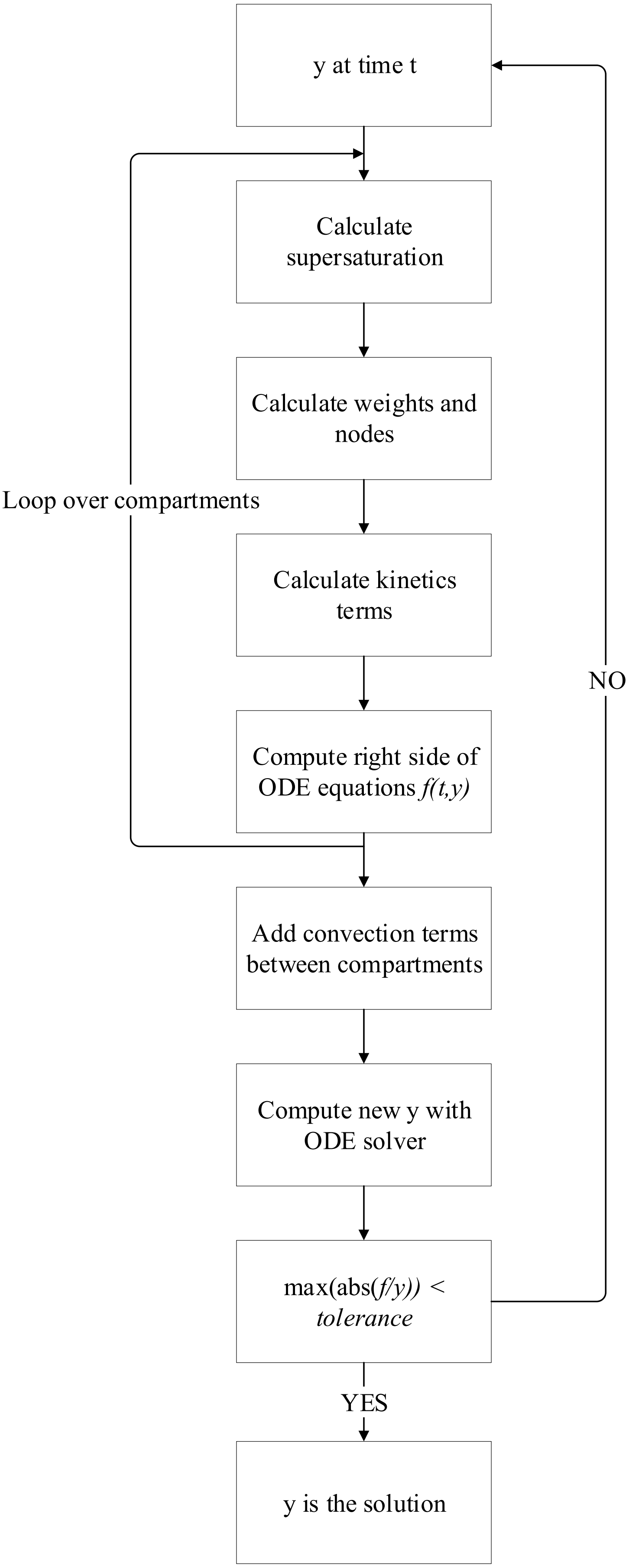

y at time t
Calculate supersaturation
Calculate weights and nodes
Loop over compartments
Calculate kinetics terms
NO
Compute right side of ODE equations *f(t,y)*
Add convection terms between compartments
Compute new y with ODE solver
max(abs(*f/y)) < tolerance*
YES
y is the solution

Figure 3.7 Schematic functioning of the calculation algorithm

3.3.8 File ChemicalEquilibria and NR-solver

The file *ChemicalEquilibria.py* contains the modified Newton-Raphson solver for the system of equilibrium equations (equations described in section 2.2.2). The inputs needed are the the cations concentrations ratio and the solver options read in the *init_run* script. Then when the function is called when supersaturation of a compartment must be calculated, the previous values of equilibrium concentrations obtained in the compartment are used as initial guess and the actual concentrations of the compartments (information contained in the solution of the IVP solver) are given as inputs.

The equilibria of the system can be described by Eqs. (2.25), (2.26), (2.27). This system of equations can be seen in the following form:

$$\boldsymbol{J}(\boldsymbol{x}^m) \cdot (\boldsymbol{x}^{m+1} - \boldsymbol{x}^m) = -\boldsymbol{f}(\boldsymbol{x}^m) \tag{3.22}$$

The variables of the system, $[{M_i}^{2+}]$, $[NH_3]$ and $[OH^-]$, concentrations in solution, are the unknown, since the known value of concentrations is the total concentration in the compartment. In the file *ChemicalEquilibria.py,* the vector $-\boldsymbol{f}(\boldsymbol{x}^m)$ and the matrix $\boldsymbol{J}(\boldsymbol{x}^m)$ of the current compartment are defined using the total values of concentrations of the compartment (solution of the IVP solver), and the initial guess of the solution of the system of equations. After that, these values are used in the function *linalg.solve* (from Numpy libraries), which solves the linear matrix equation (Eq. (3.22)), returning $(\boldsymbol{x}^{m+1} - \boldsymbol{x}^m)$ (the solver is called modified since works with this difference), that can be used to obtain the new values of the solution $\boldsymbol{x}^{m+1}$ that will be used in next iteration. If the l_1 norm of $(\boldsymbol{x}^{m+1} - \boldsymbol{x}^m)$ is less than the tolerance ϵ (defined in the solver options) or if the number of iterations overcome a limit (a limit read in the *caseSetup* file), the iterations are stopped:

$$\|\boldsymbol{x}^{m+1} - \boldsymbol{x}^m\|_1 < \epsilon \tag{3.23}$$

And the last value $\boldsymbol{x}^{m+1}$ is considered the solution and can be used to calculate supersaturation (see chapter 2.2.1). Then supersaturation is given as output of the function and will be used to define the equations of the compartment for the ODE solver. The last value of the solution $\boldsymbol{x}^{m+1}$ will be stored to use it as initial value for the compartment in the following time step.

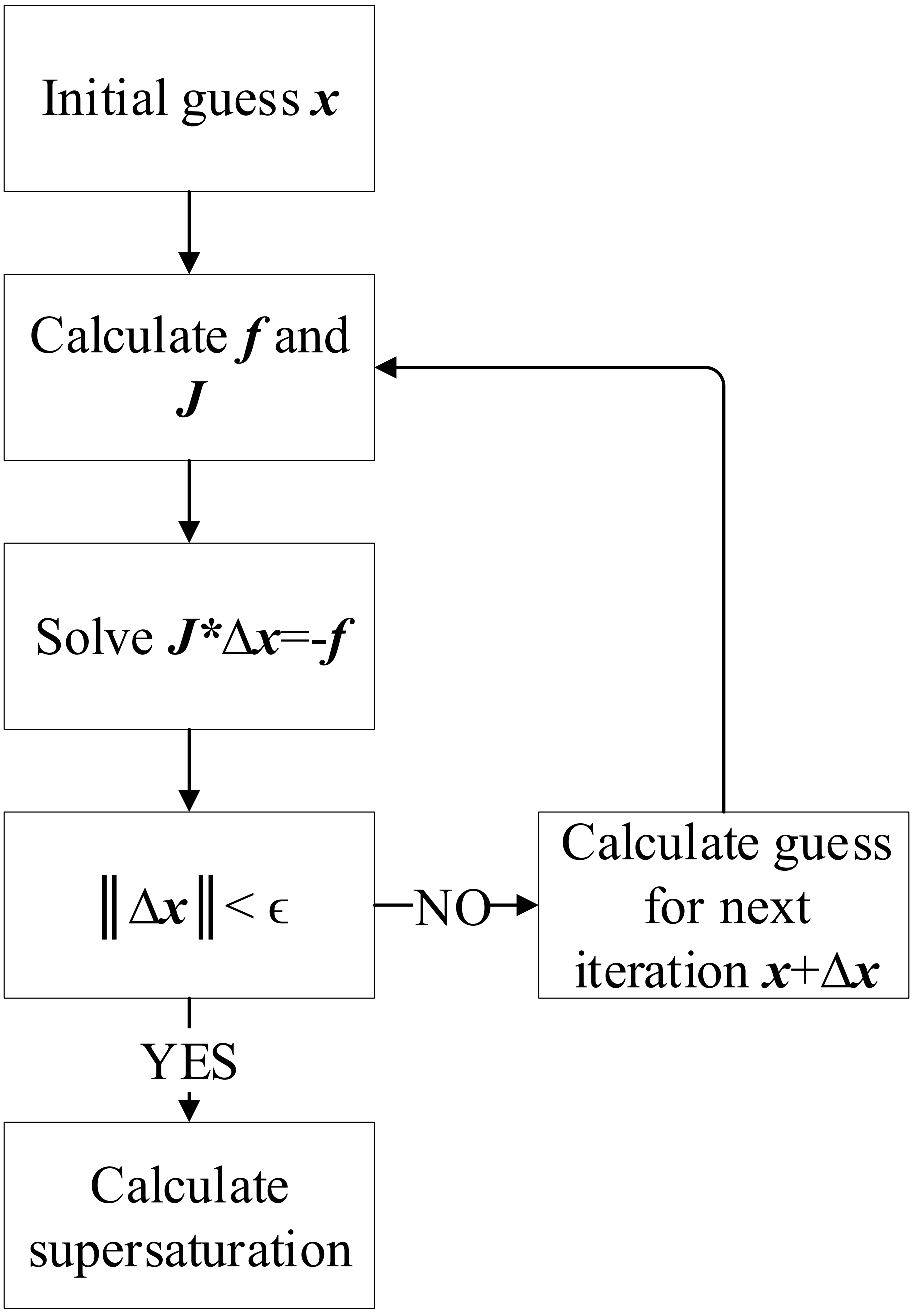

Initial guess $\boldsymbol{x}$
Calculate $\boldsymbol{f}$ and $\boldsymbol{J}$
Solve $\boldsymbol{J}^*\Delta\boldsymbol{x}=-\boldsymbol{f}$
$\|\Delta\boldsymbol{x}\| < \epsilon$
NO
Calculate guess for next iteration $\boldsymbol{x}+\Delta\boldsymbol{x}$
YES
Calculate supersaturation

Figure 3.8 Schematic functioning of the file ChemicalEquilibria

3.3.9 File Moment Calc

In this file are contained the algorithms which purpose is to compute the weights and the abscissas for the QMOM.

The algorithms used is the adaptive Wheeler (see chapter 2.3.3), at which, given as input the moments and the number of nodes, give as output the weights and nodes, or can raise an error in case of non realizability.

The use of this algorithm gives more stability to the script, especially in the initial time steps.

3.3.10 Kinetic files

Other noteworthy files are the files are the ones in which kinetics are defined: *growth.py, nucleateSize.py, nucleation.py, aggrEfficency.py* and *aggregation.py*.

These files use the information read with the function ‘createModels’ (see *NiMnCoHydroxidePrec*) to calculate the growth rate, the nucleate size, the nucleation rate, the aggregation efficiency and the aggregation term.

In these files the models used to describe the mechanisms that vary the population are defined. These models are given every iteration some information as input since they depend on different properties of the system that can vary from iteration to iteration. The models used and their parameters will be presented below.

The critical nucleate size is defined as constant:

$$x_c = constant \tag{3.24}$$

For nucleation, as seen in chapter 2.4.1, an exponential law dependent on $ln^2(S)$ can describe the phenomena, with two terms, one for homogeneous nucleation and one for heterogeneous nucleation:

$$J = K_{J1} \cdot \exp\left(\frac{-n_{j1}}{ln^2(S)}\right) + K_{J2} \cdot \exp\left(\frac{-n_{j2}}{ln^2(S)}\right) \tag{3.25}$$

In which S is the supersaturation, defined in section 2.2.1 and K_{J1}, K_{J2}, n_{j1} and n_{j2} are parameters of the model

Growth:

$$G = K_G \cdot (S-1)^{n_G} \tag{3.26}$$

In which K_G and n_G are parameters of the model.

Aggregation efficency:

$$A_{eff} = \exp\left(-\frac{\sqrt{\frac{\varepsilon}{\nu}} \cdot D_b}{G \cdot f}\right) \tag{3.27}$$

With:

$$D_b = \sqrt{\frac{\rho}{A_p}} \cdot (\varepsilon \cdot \nu)^{\frac{1}{4}} \cdot L_{eq} \tag{3.28}$$

$$L_{eq} = \frac{L_1 L_2}{\sqrt{(L_1 - L_2)^2 + L_1 L_2}} \tag{3.29}$$

$$f = \frac{4 \cdot (1 + r_L - \sqrt{{r_L}^2 - 1})}{\left(\frac{1}{3} + r_L - \sqrt{{r_L}^2 - 1}\right) - \left(r_L - \sqrt{{r_L}^2 - 1}\right)^2 \cdot \left(\frac{2}{3} r_L + \frac{\sqrt{{r_L}^2 - 1}}{3}\right)}$$

$$with \begin{cases} r_L = \frac{L_1}{L_2} \; if \; L_1 > L_2 \\ r_L = \frac{L_2}{L_1} \; if \; L_2 > L_1 \end{cases} \tag{3.30}$$

In which ε is the turbulent dissipation rate, ν is the kinmatic viscosity, G is the growth rate, ρ is water density, A_p is a parameter of the model and L_1 and L_2 are the abscissas of the QMOM.

Aggregation:

$$A = A_{eff} \cdot \left(\frac{2k_bT}{3\mu} \left(\frac{L_1}{L_2} + \frac{L_2}{L_1} + 2 \right) + 2{,}2943 \cdot c_{adj} \sqrt{\frac{\varepsilon}{\nu}} (L_1 + L_2)^3 \right) \tag{3.31}$$

In which k_b is the Boltzmann constant, T is the temperature and μ is the dynamic viscosity. The first term in the parenthesis is the aggregation contribute of Brownian motion and the second term is the contribute of turbulence, with c_{adj} as parameter of the model.

3.3.11 File runPrecSolver

The *RunPrecSolver.py* is the file that is launched in order to start the calculations.

This file contains the main function, from which are imported the functions from the other files, giving them the input variables they need to go on with the calculations.

So first it recalls the *init_run* main function and from that obtains all its outputs (parameters, settings and compartments information). These outputs are used as inputs to the ODE solver (in *NiMnCoHydroxidePrec*), which will compute the solution and give it as output.

It can be of interest to fill the reactor with some of the species (such as ammonia) before carry the precipitation calculations, to improve the solver performances and robustness: to do that is used the ODE solver with the source terms for the equations deactivated, in inlet just the species that is desired to fill the reactor and initial conditions equal to zero: this way a solution of just mixing of this species is calculated over all the compartments.

At this point the precipitation can be solved: the ODE solver is used, this time with the real inlets conditions (all species) as inlets and the mixing solution calculated in the previous step as initial condition. This first solution is sought until a first final time: the solution is sought dividing the total time of integration in many intervals, every interval has his own maximum time step allowed that will be used by the ODE solver to progress with the calculations and will be improved moving forward in time (and therefore approaching the stationary). This way the initial calculations, which are the most sensitive, are carried with little time steps and the robustness of the solver is improved. So the ODE solver is used many times, each with initial solution the solution from the previous interval time (see chapter 3.3.7). Intervals and time steps are shown in table 3.5.

```
settings, parameters, compartments info = initialize
function outputs
source terms = off
inlet = select species to mix
mixing solution = ODE function(settings, parameters,
compartments info, inlet)
source terms = on
inlet = real inlet
initial conditions = mixing solution
solution = ODE function(settings, parameters, compartments
info, inlet)
```

4. RESULTS

4.1 CFD results

In this section the results of the CFD simulations, carried with the setup described in section 3.1.6, will be described.

The contour plots of the velocity magnitude and the turbulent dissipation rate are presented in two different planes, perpendicular to each other and with *z* axis (parallel to the shaft of the stirrer and in the direction of the height of the reactor) in Fig. 4.1 and 4.2.

The contour plot for the velocity magnitude is shown with velocity vectors to highlight the direction of the flow. These plots show the effect of the stirrer on the velocity field and the symmetry with respect to the *z* axis.

The flow features guarantee a good degree of dispersion of the precipitated particles which are very heavy and a good degree of dispersion of the reactants. It is eventually worth noticing that the upper part of the reactor is instead characterised by a stagnant zone.

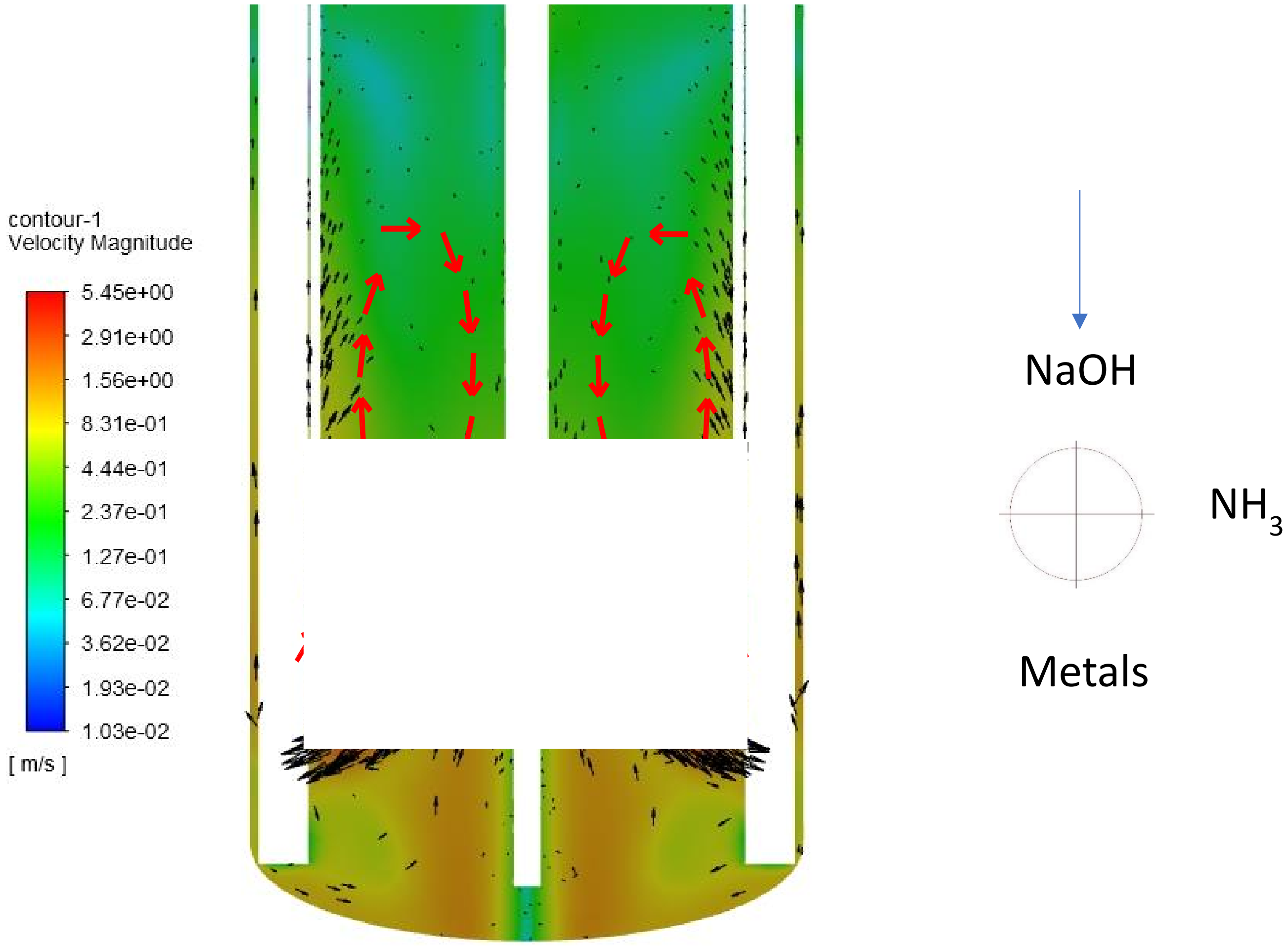


Figure 4.1 Velocity magnitude (m/s) in the plane containing NH_3 inlet

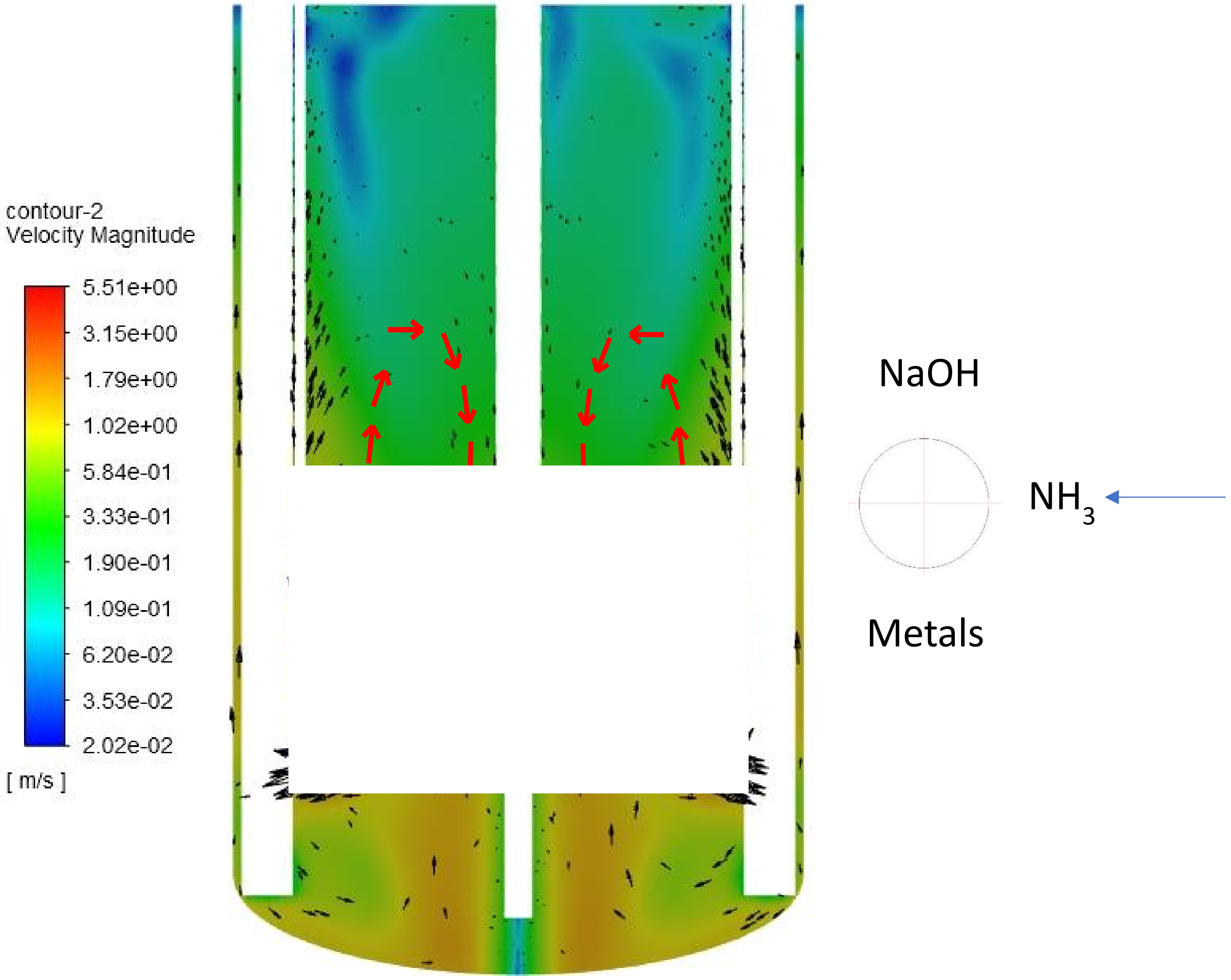


Figure 4.2 Velocity field in the plane containing metals and NaOH inlets

Another property of great interest to assess the mixing feature of the reactor is the turbulent dissipation rate. This property is important because it determines at the same time the mixing rate at different scales and the aggregation rate. It is typically used to divide to reactor into compartments.

The values of volume average turbulent dissipation rate and mixing power required to stir the reactor by Umicore are of the same order of magnitude, then the value obtained by the simulation can be considered reliable. The difference existing between the values can be a consequence of an insufficiently dense grid near the rotating parts, that results in a loss of precision when describing this property here, since as expected, the values of turbulent dissipation rate near the rotating zone are very high, changing of two or three orders of magnitude moving slightly away from them (Fig. 4.3 and 4.4).

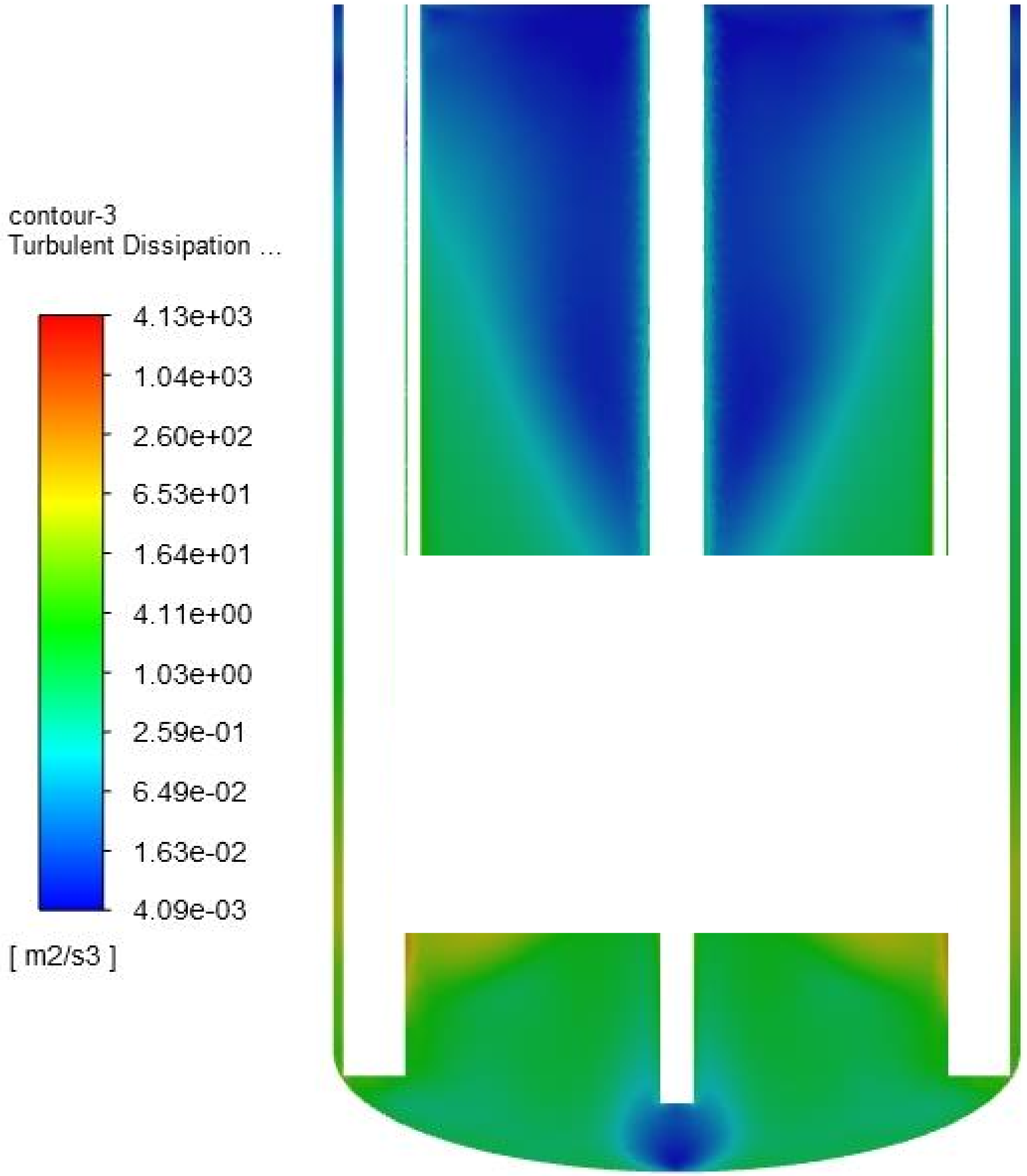
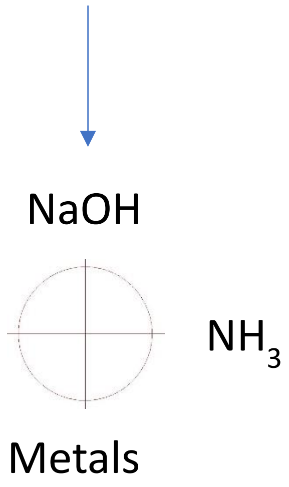


Figure 4.3 Turbulent dissipation rate field in the plane containing NH_3 inlet



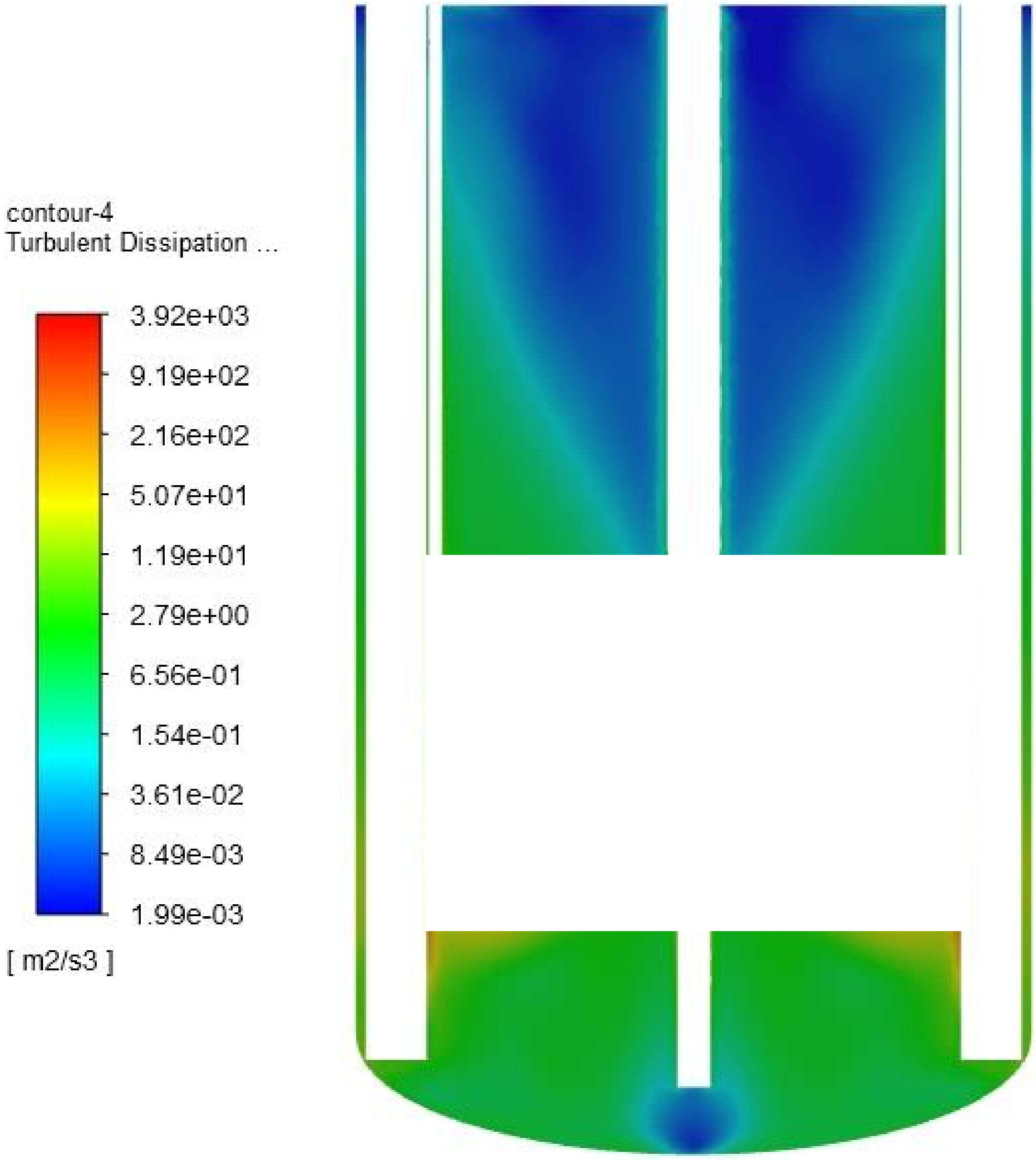
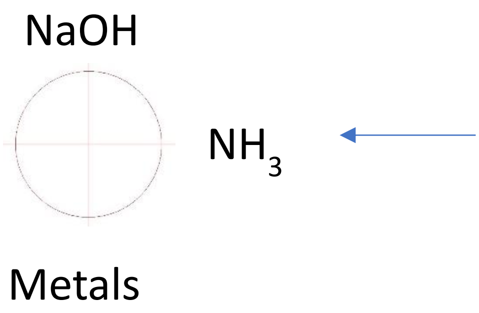


Figure 4.4 Turbulent dissipation rate field in the plane containing metals and NaOH inlets



4.2 Script results

The script is carried with some different settings and compartment divisions and some conclusion can be drawn concerning the criteria employed to divide the reactor in compartments.

Inlet concentrations of the species are given by Umicore. Moments at inlet are equal to zero, representing the fact that no particles are fed in the inlets.

Different strategies are explored and they are described below:

- Compartment division by using only turbulent dissipation rate
- Compartment division by using only supersaturation
- Compartment division by using turbulent dissipation rate and supersaturation

4.2.1 Compartment division using only turbulent dissipation rate

In this simulation, the reactor is divided by using the decimal logarithm of the turbulent dissipation rate. As explained before, the turbulent dissipation rate is important for the phenomena involved because it determines the overall mixing rate in the reactor and it determines both the collision rate and the aggregation efficiency.

Turbulent dissipation rate field is known from the CFD simulation of the reactor (see chapter 4.1).

The number of compartments chosen was 20 and in figure 4.5 there are shown in the plane containing the metal and NaOH inlets.

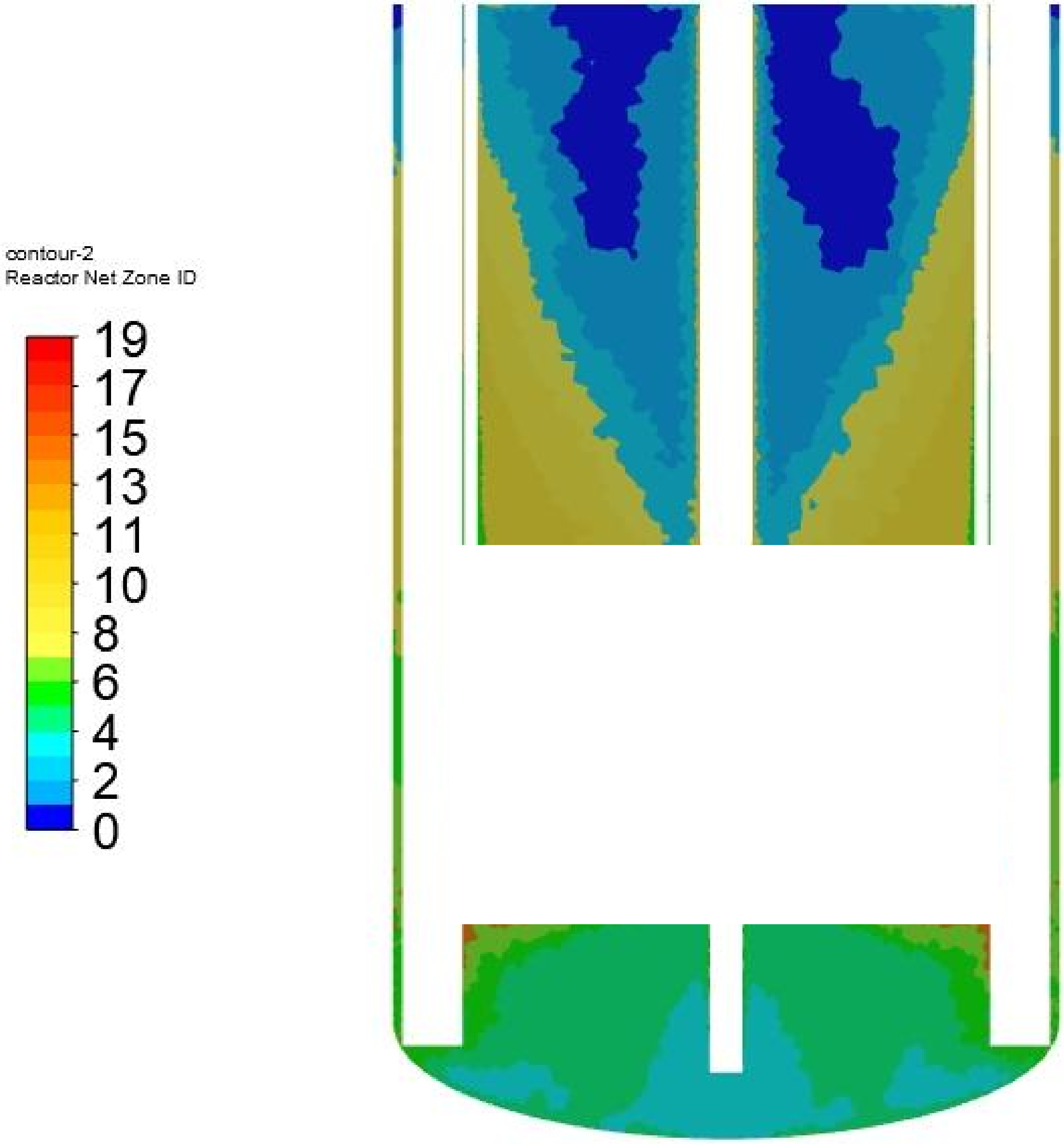


Figure 4.5 Compartments using turbulent dissipation rate in a plane containing metals and NaOH inlets

The concentrations of the chemical species involved and the moments of the PSD obtained at the outlet of the reactor are shown in table 4.1.

Outlet	
Ni^{2+} (kmol/m^3)	4,97x10^{-7}
Mn^{2+} (kmol/m^3)	1,66x10^{-7}

Co^{2+} ($kmol/m^3$)	$1{,}66x10^{-7}$
NH_3 ($kmol/m^3$)	$8{,}75x10^{-1}$
m_0 ($1/m^3$)	$2{,}98x10^{+9}$
m_1 (m/m^3)	$4{,}37x10^{+5}$
m_2 (m^2/m^3)	$1{,}27x10^{+2}$
m_3 (m^3/m^3)	$5{,}58x10^{-2}$
SMD	$4{,}37x10^{-4}$

Table 4.1 Concentrations, moments and SMD (Sauter Mean Diameter) at outlet of the compartment model created with turbulent dissipation rate

Results of table 4.1 can be explained observing this figure: nucleation and growth are heavily dependent on supersaturation (Eqs. 3.25 and 3.26); supersaturation, by definition (Eq. 2.17), depends on concentrations, in particular those of metals. It is therefore expected that supersaturation will be higher near the metal (and sodium hydroxide) inlets, quickly decreasing when moving away from them, due to mixing and the high precipitation rate which results in supersaturation depletion.

It is clear that a correct description of the process requires a large number of compartments in regions characterized by large concentration gradients.

The compartments at the inlets are relatively big: this means that the reactants enter into very large compartments, which are considered perfectly mixed resulting in concentration values that are smoothed out. This in turn results into an underestimation of the nucleation and growth rates, leading to low values of the predicted moments of the PSD at outlet (table 4.1). Figure 4.6 shows how supersaturation is low and similar in all the compartments.

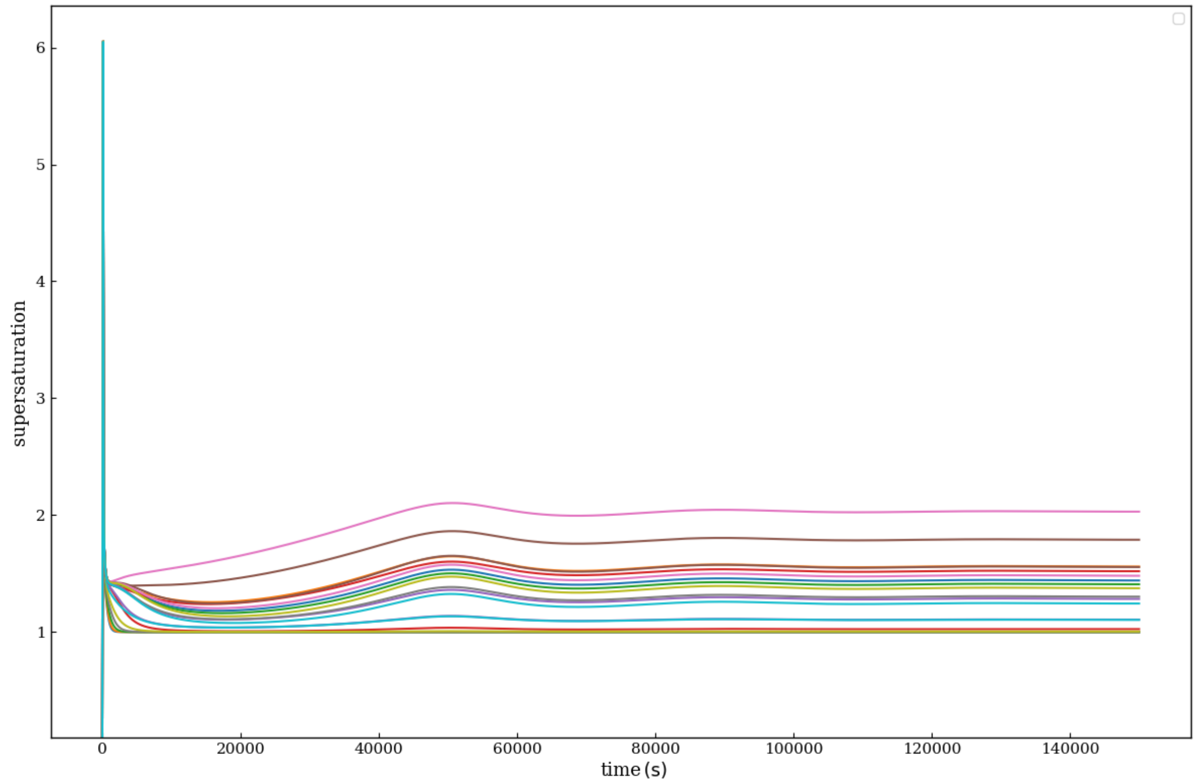


Figure 4.6 Supersaturation over compartments dividing the reactor with turbulent dissipation rate

4.2.2 Compartment division by using only supersaturation

In this simulation, the reactor is divided by using the decimal logarithm of the supersaturation. The supersaturation values are obtained by an on-going CFD simulation in with the PBM is directly solved within the code Ansys CFD Fluent (see chapter 3.3.2). The CFD simulation has not converged yet but results are consistent with what is expected.

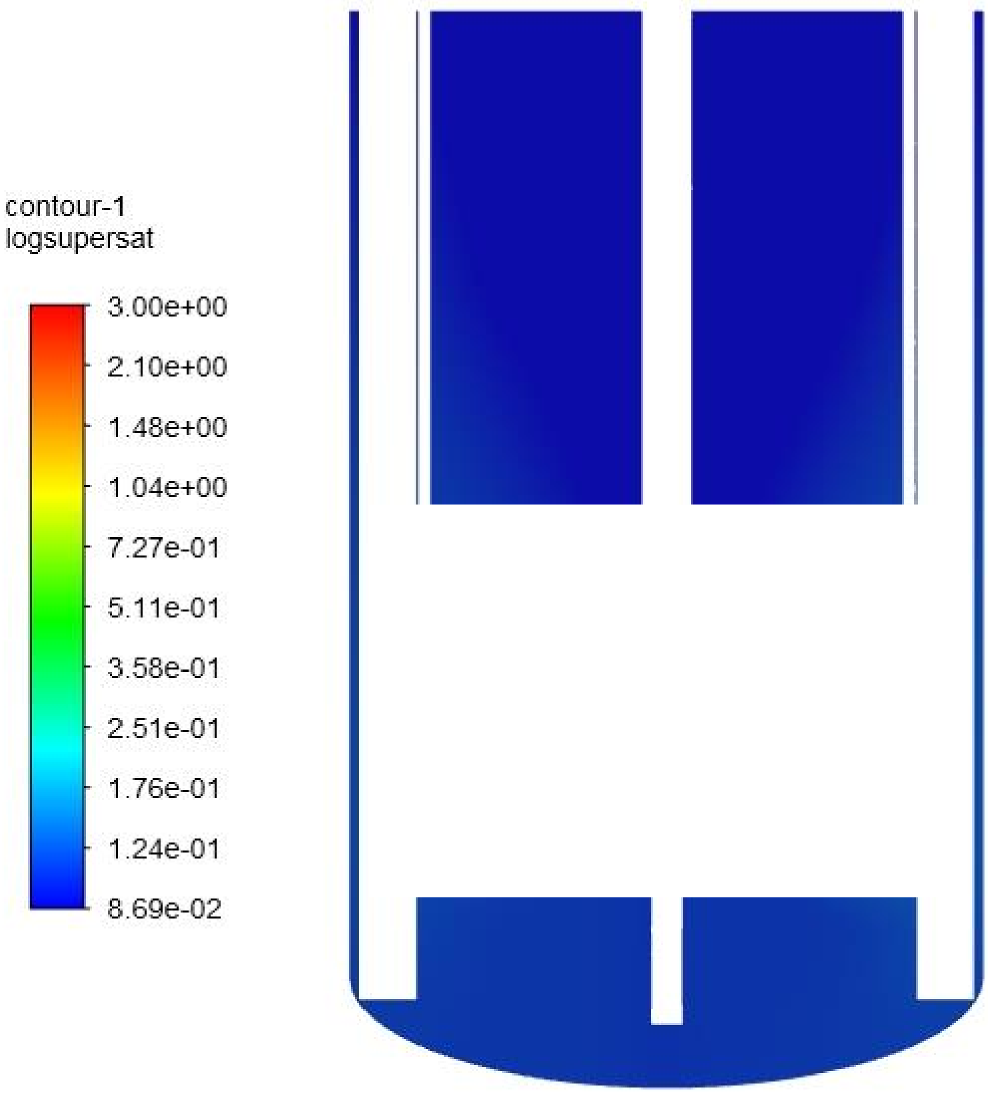


Figure 4.7 Supersaturation distribution in an on going full coupled CFD-PBE simulation

Supersaturation is a very important properties since it defines the rates of all the relevant phenomena, especially nucleation and molecular growth. In figure 4.7 the supersaturation spatial distribution is shown in one plane: as it can be seen there is a high gradient of supersaturation at the inlets, in particular in the inlets for the metals and NaOH. The compartment division must take into account these high values of supersaturation gradients near the inlets. The division into compartments must be carried out accordingly loosing as little information as possible and creating compartments small enough to properly represent these supersaturation gradients.

The number of compartments chosen to divide the reactor was 20. In figure 4.8 the compartments distribution in the plane containing the inlets for metals and NaOH is shown.

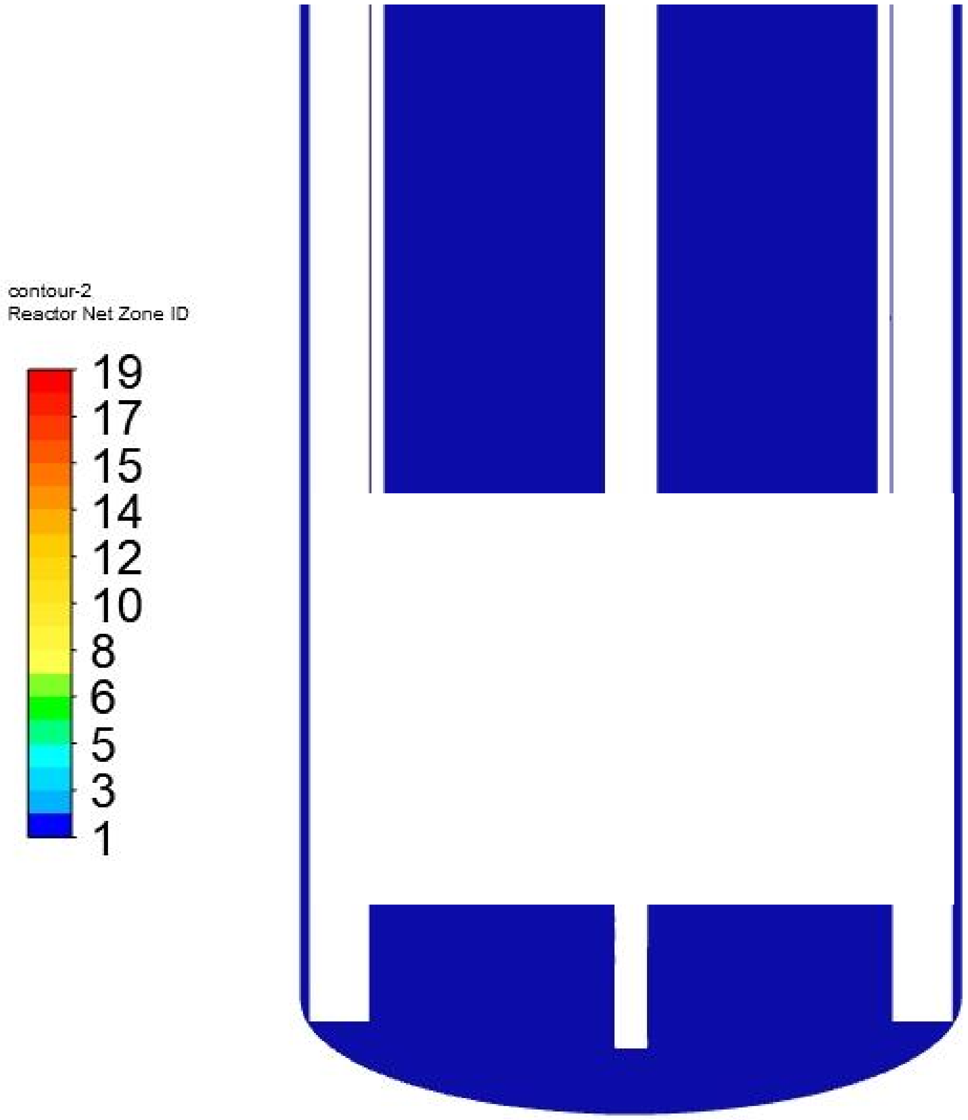


Figure 4.8 Compartments using supersaturation in a plane containing metals and NaOH inlets

Table 4.2 shows the metal concentrations, the concentration of all the other chemical species involved, as well as the values of the moments of the PSD and of the mean Sauter diameter of the particles at the reactor outlet.

Outlet	
Ni^{2+} (kmol/m^3)	4,97x10^{-7}
Mn^{2+} (kmol/m^3)	1,66x10^{-7}
Co^{2+} (kmol/m^3)	1,66x10^{-7}

NH_3 (kmol/m^3)	8,75x10^{-1}
m_0 (1/m^3)	2,2x10^{+12}
m_1 (m/m^3)	1,57x10^{+7}
m_2 (m^2/m^3)	7,43x10^{+2}
m_3 (m^3/m^3)	5,58x10^{-2}
SMD	7,5x10^{-5}

Table 4.2 Concentrations, moments and SMD (Sauter Mean Diameter) at outlet of the compartment model created with supersaturation

Comparing these results with the ones obtained by dividing the reactor using the turbulent dissipation rate, it can be seen that the values of the moment of the PSD are larger, proving that more particle nucleation and growth is predicted in this case.

The creation of smaller compartments at inlets allows a more accurate calculation of the nucleation and growth terms, which are responsible for the increasing of the moment of the PSD.

With that kind of division, it is possible to see a supersaturation gradient among the compartments (figure 4.9). This type of division, however, has a flaw: not taking into account the turbulent dissipation rate in the reactor can result in an incorrect estimate of the aggregation rate, which in this case shows to be not negligible just in the compartment of metals inlet.

The diagrammed value is the contribute to the moment of order 0 given by the aggregation (term $\frac{1}{2}\sum_{i=1}^{N} w_i \sum_{j=1}^{N} w_j \left(L_i^3 + L_j^3\right)^{\frac{q}{3}} \beta_{agg} - \sum_{i=1}^{N} w_i L_i^q \sum_{j=1}^{N} w_j \, \beta_{agg}$ in Eq. (2.75), with q = 0).

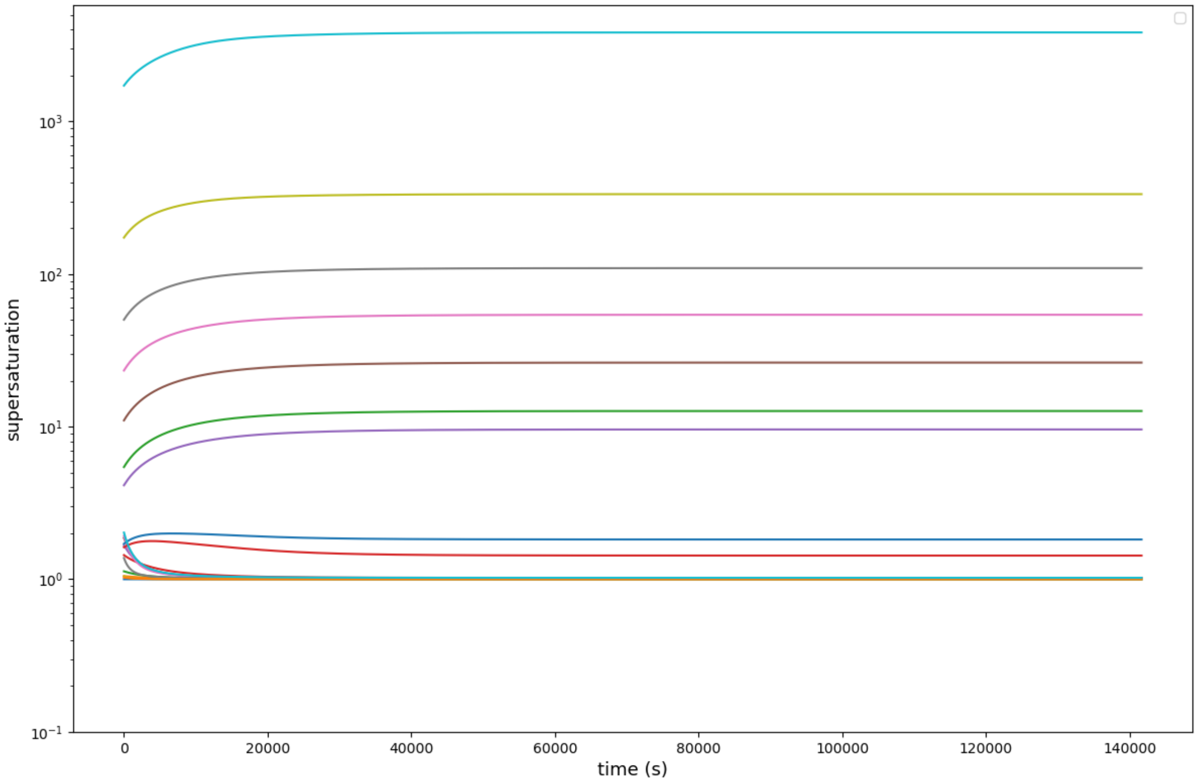


Figure 4.9 Supersaturation over compartments dividing the reactor with supersaturation

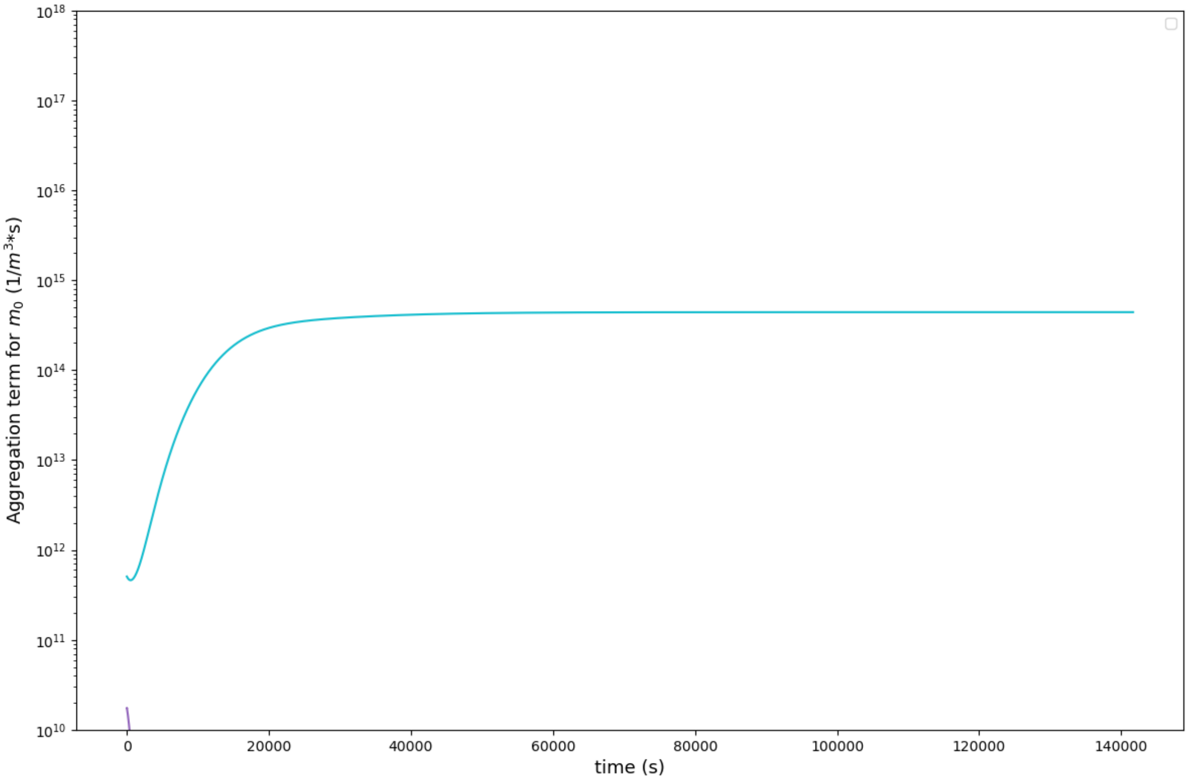


Figure 4.10 Aggregation contribute to the moment of order 0 over compartments dividing the reactor with supersaturation (similar behaviour for the other moments)

As it can be seen in figure 4.11 the nucleation rate is appreciable in many compartments, but most of the nucleation occurs near the metal inlets. In figure 4.12 it can be seen that the growth rate is more evenly distributed between compartments, since is less dependent from the supersaturation. The maximum growth rate is still in the compartment near the metal inlets.

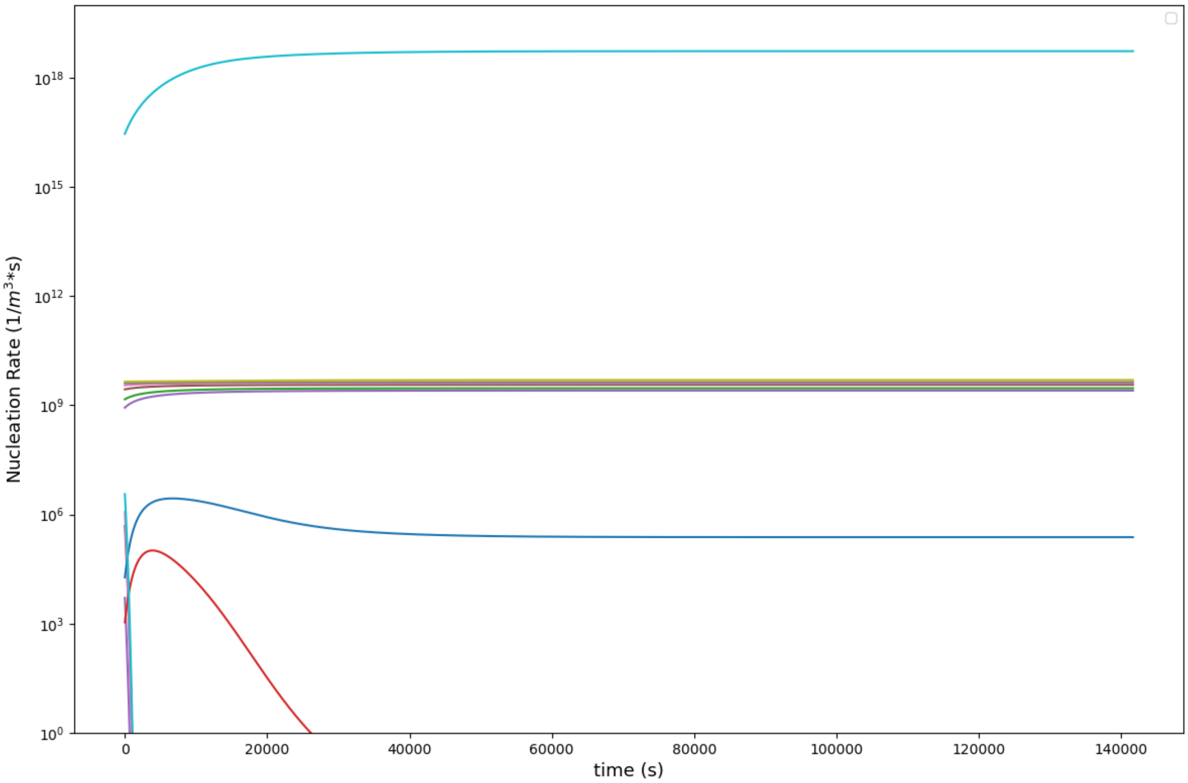


Figure 4.11 Nucleation rate over compartments dividing the reactor with supersaturation

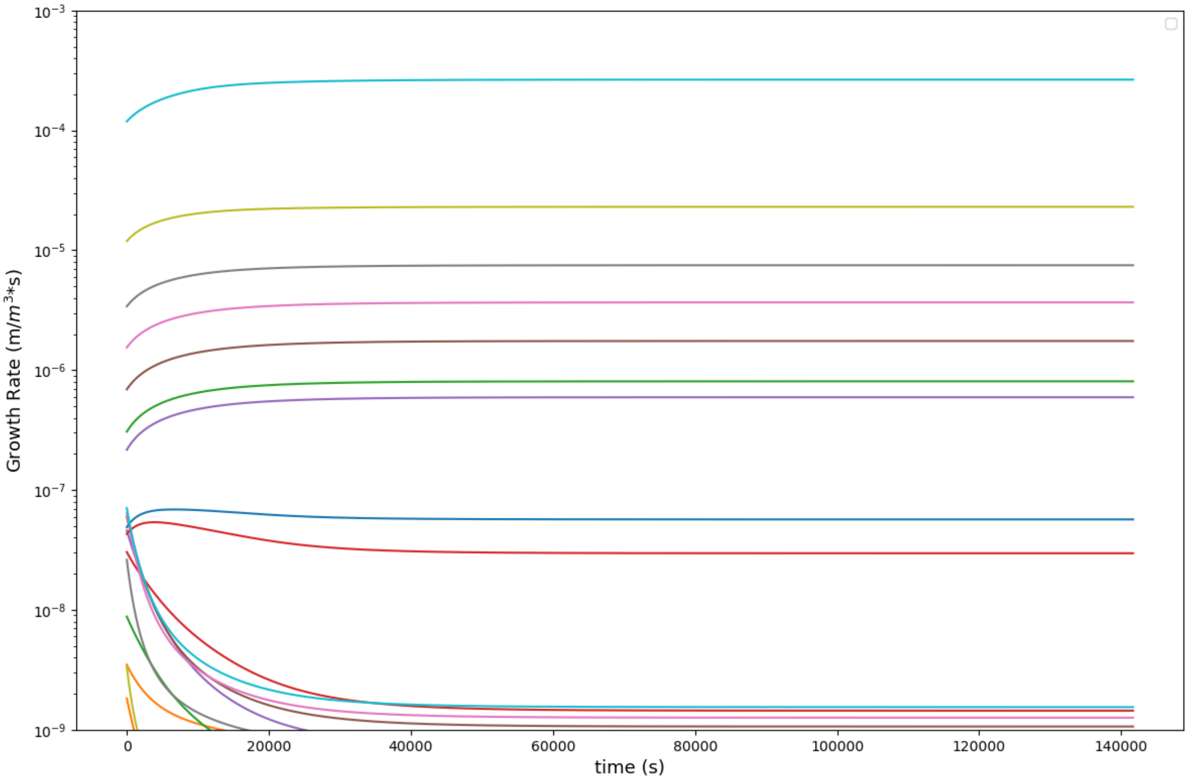


Figure 4.12 Growth rate over compartments dividing the reactor with supersaturation

4.2.3 Compartment division by using the turbulent dissipation rate and supersaturation

Examining figures 4.5 and 4.9, it has been described how the division considering turbulent dissipation rate lacks of small compartments at the inlet, while the division considering supersaturation lacks of compartments in the zones with greater difference in the turbulent dissipation rate.

Both supersaturation and the turbulent dissipation rate are important for the description of the phenomena involved and therefore it is a fundamental goal to develop a strategy that divides the reactor in compartments considering both properties, in order to minimize the loss of information.

The Reactor Network Tool provided by Fluent allows the user to select multiples Custom Field Functions in order to divide the reactor into compartments.

The first idea is naturally the selection of the two Custom Field Functions, the decimal logarithm of the supersaturation (or the supersaturation) and the decimal logarithm of the turbulent dissipation rate, and the creation of the compartments with this method. Carrying on with that idea, the results obtained were not optimal, since clearly the two different properties cannot be decoupled, interacting and making the created compartments not usable.

Then, in order to decouple the information an UDF has been written:

```
if (supersaturation > VALUE)
{
    UDMa = -100;
    UDMb = supersaturation;
}
else
{
    UDMa = log10(turbulent dissipation rate);
    UDMb = -100;
}
```

The aim of the UDF is to create two User Defined Memories (UDM): starting from the supersaturation distribution, an if condition discriminates the cells with supersaturation higher than VALUE, cells in which is of interest to use the supersaturation to create the compartments, from the other cells, in which is of interest to use the logarithm of the turbulent dissipation rate to create the compartments.

Therefore in the cells with supersaturation values larger than a threshold value (taken equal to 100 in this example) the supersaturation is stored in UDMa, while in the other cells the decimal logarithm of the turbulent dissipation rate is stored in UDMb. By giving a completely out of range value to UDMa (for example -100) in the cells in which we are interested in supersaturation, and vice versa with UDMb where we are interested in dissipation rate, we are able to completely uncouple the two parameters when compartments are created.

After compiling this UDF, in which the *VALUE* decided as threshold was 4 (which has been seen to describe an appropriate area around the inlets), the UDMs created are defined as Custom Field Functions and selected in the Reactor Network Tool to create the compartments.

The number of compartments chosen to divide the reactor was 50. In figure 4.15 the compartment distribution in the plane containing the inlets for the metals and NaOH is shown.

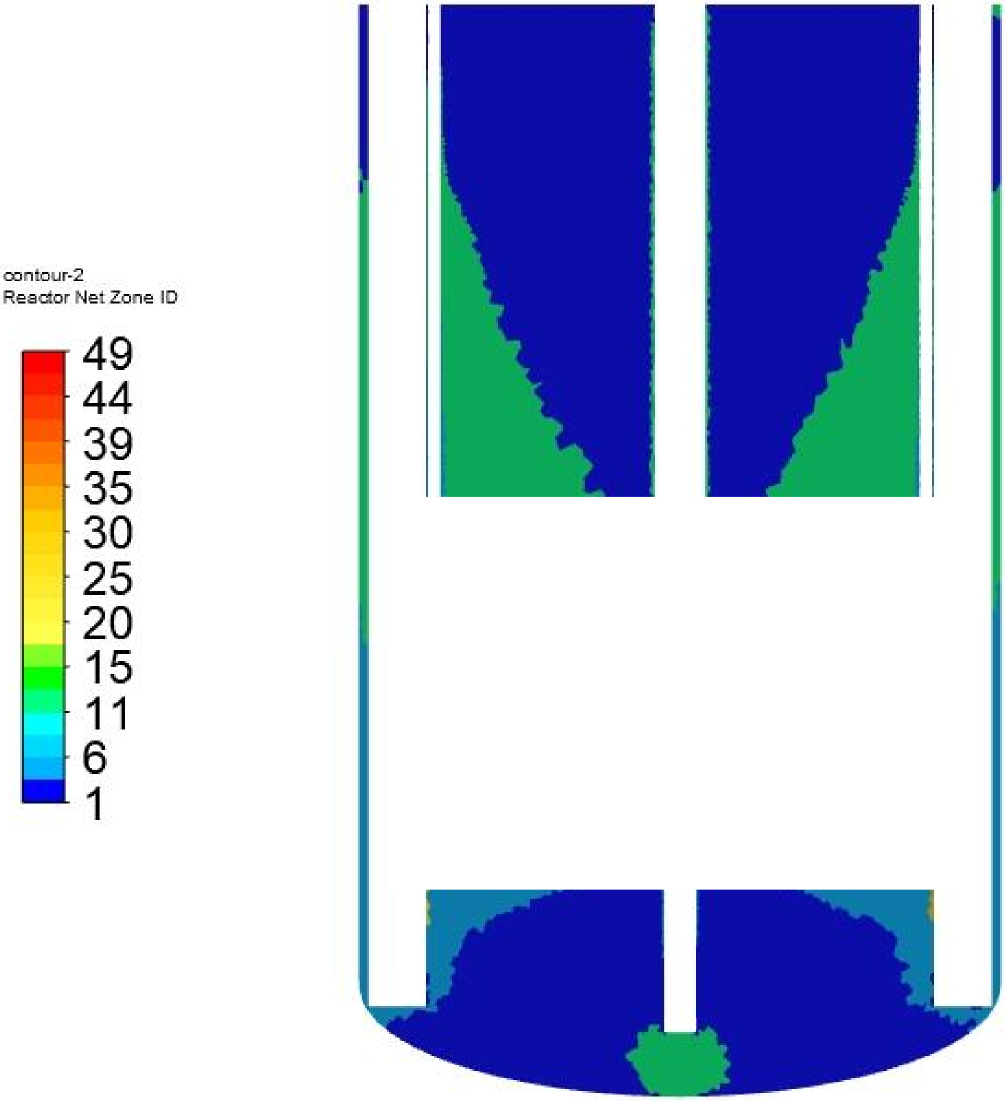


Figure 4.13 Compartments (50) using the UDF in a plane containing metals and NaOH inlets

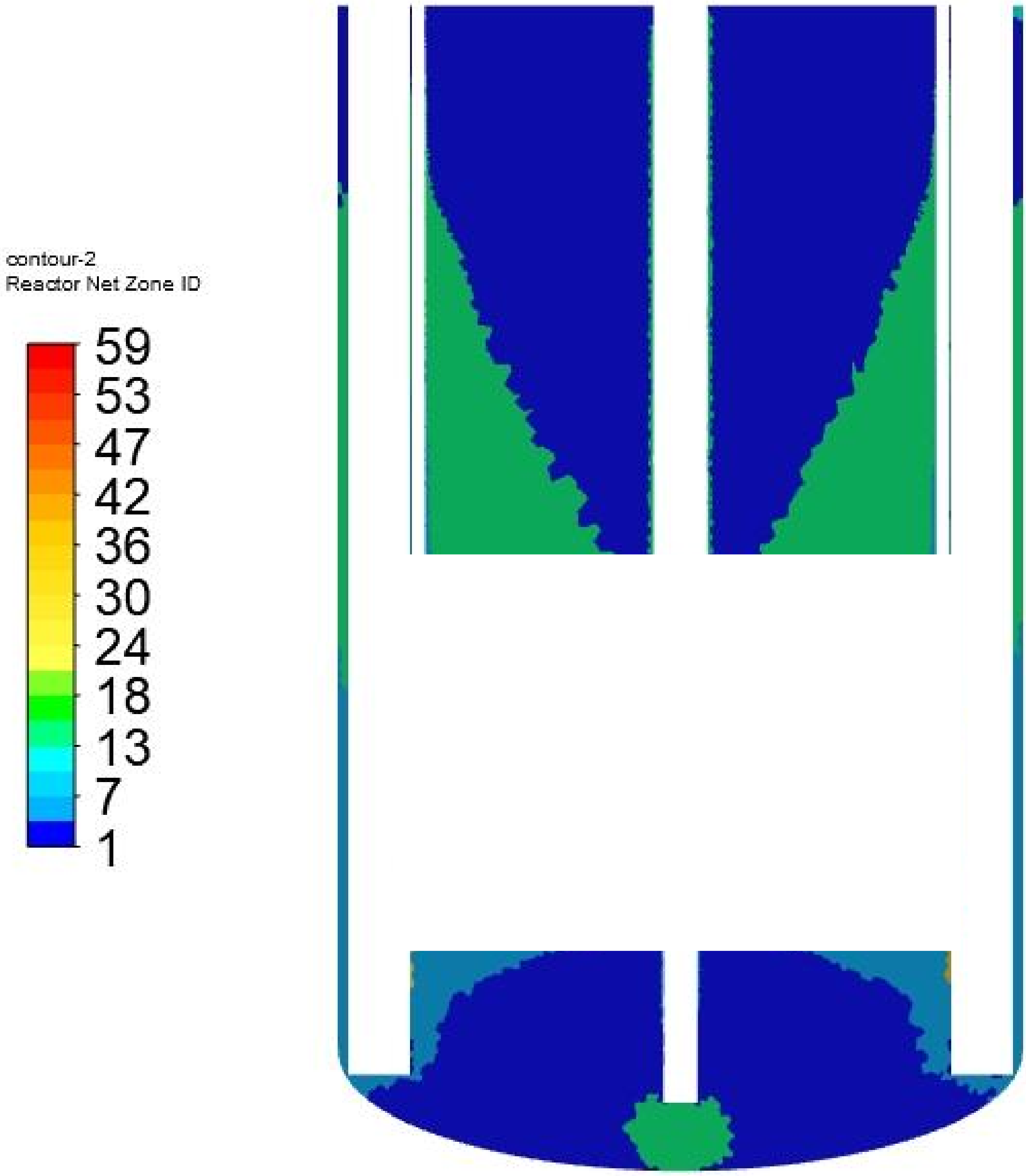
It can be seen in figure 4.15 that the compartments are created as desired: compartments near the inlets are created and in the rest of the reactor the compartments are similar to the ones created using only the turbulent dissipation rate (figure 4.5).

Providing the script with information about these compartments, the concentrations and the moments of the PSD obtained at the outlet of the reactor were of the order of magnitude of those obtained by dividing the reactor with the logarithm of turbulent dissipation rate (table 4.3). This is because, as can be seen in figure 4.16, the compartments created at the inlets were few and too big compared to those created using only the supersaturation.

Outlet	
Ni^{2+} (kmol/m^3)	4,97x10^{-7}
Mn^{2+} (kmol/m^3)	1,66x10^{-7}
Co^{2+} (kmol/m^3)	1,66x10^{-7}
NH_3 (kmol/m^3)	8,75x10^{-1}
m_0 (1/m^3)	2,51x10^{+9}
m_1 (m/m^3)	3,89x10^{+5}
m_2 (m^2/m^3)	1,20x10^{+2}
m_3 (m^3/m^3)	5,58x10^{-2}
SMD	4,7x10^{-4}

Table 4.3 Concentrations, moments and SMD (Sauter Mean Diameter) at outlet of the 50 compartments model created with UDF

The number of compartments has been increased to 60 in order to achieve a division at inlets more similar to the one obtained when using supersaturation only. Figure 4.17 shows the compartment distribution in the plane containing the inlets for the metals and NaOH.



4.14 Compartments (60) using the UDF in a plane containing metals and NaOH inlets

Providing the script with information about these compartments, the concentrations and moments obtained at the outlet of the reactor are shown in table 4.4.

Outlet	
Ni^{2+} (kmol/m^3)	4,97x10^{-7}

Mn^{2+} (kmol/m^3)	1,66x10^{-7}
Co^{2+} (kmol/m^3)	1,66x10^{-7}
NH_3 (kmol/m^3)	8,75x10^{-1}
m_0 (1/m^3)	3,8x10^{+12}
m_1 (m/m^3)	2,74x10^{+7}
m_2 (m^2/m^3)	9,80x10^{+2}
m_3 (m^3/m^3)	5,58x10^{-2}
SMD	5,7x10^{-5}

Table 4.4 Concentrations, moments and SMD (Sauter Mean Diameter) at outlet of the 60 compartments model created with the UDF

Results show that nucleation and growth are described with more details at the metal inlet, since there are more and smaller compartments there. This can be seen also in figure 4.19, in which supersaturation in every compartment is shown, underlining the presence of compartments with high supersaturation at inlets, as expected.

The turbulent dissipation rate is also well considered in the reactor. Observation of figure 4.17 and comparison with figure 4.5 (compartments created using only the logarithm of turbulent dissipation rate) reveals that in the reactor the compartments are divided correctly using the logarithm of the turbulent dissipation rate. Figure 4.20 shows the aggregation rate among the compartments. It can be seen that the aggregation rate is not negligible just in one compartment, that is the one directly near the metals inlet face.

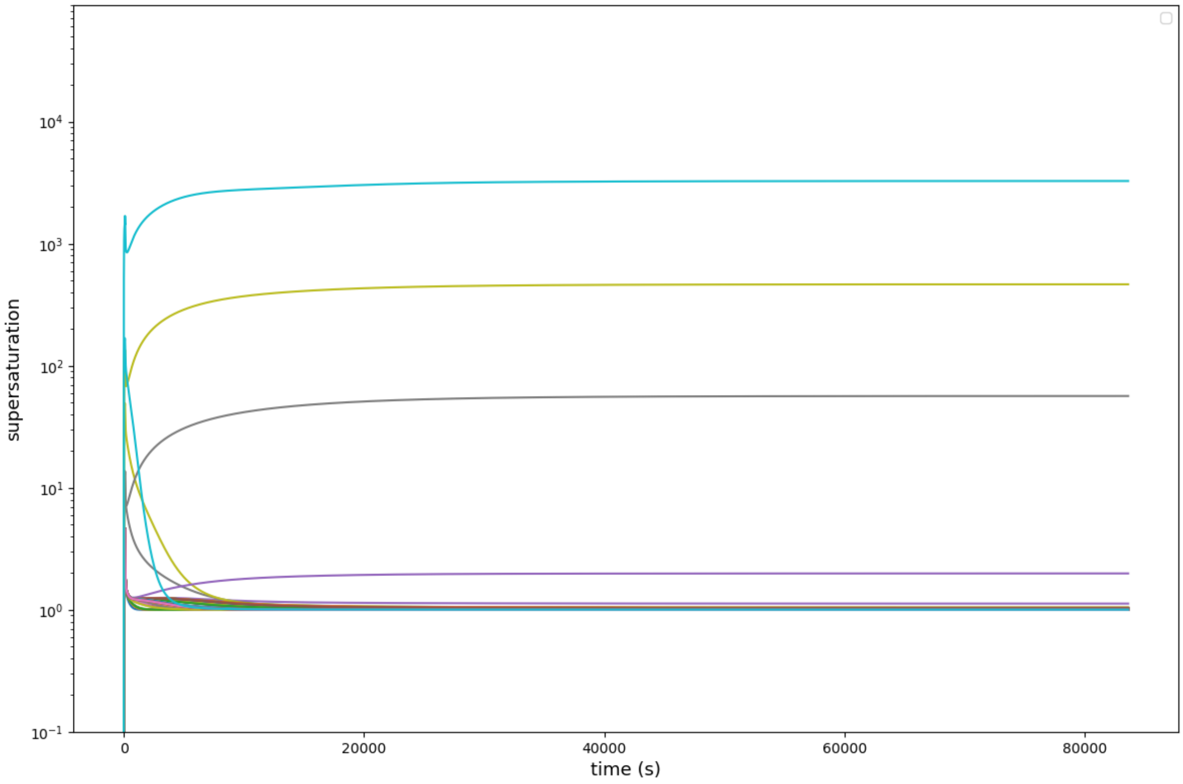


Figure 4.15 Supersaturation over 60 compartments dividing the reactor with the UDF

The behaviour of the aggregation rate can be explained as follows: the overall aggregation rate equals to zero because the aggregation efficiency is very low. In fact, in the compartments in which the growth rate, on which the aggregation efficiency depends (Eq. 3.27), is too small even under the presence of large collision rates no aggregation occurs. Only the compartments in which the growth rate is high enough (metals inlet) presents an aggregation efficiency significantly different from zero. Those are the compartments in which some aggregation occurs.

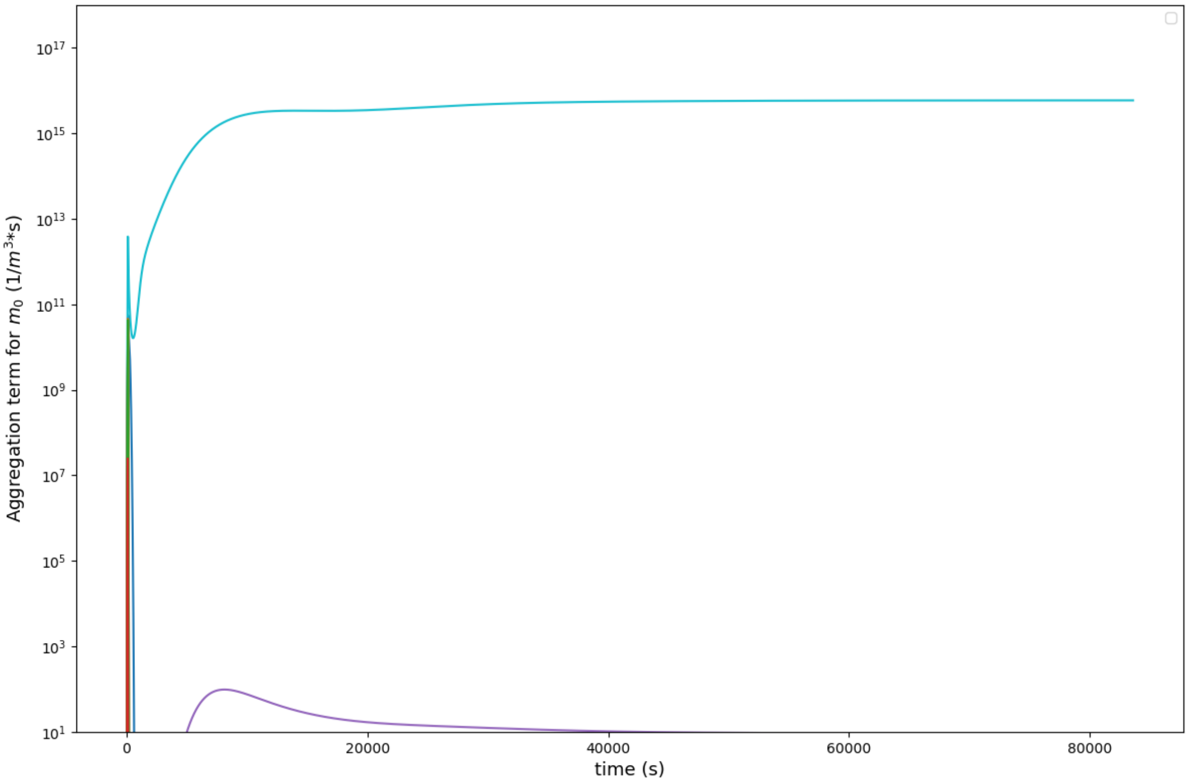


Figure 4.16 Aggregation contribute to the moment of order 0 over 60 compartments dividing the reactor with the UDF (similar behaviour for the other moments)

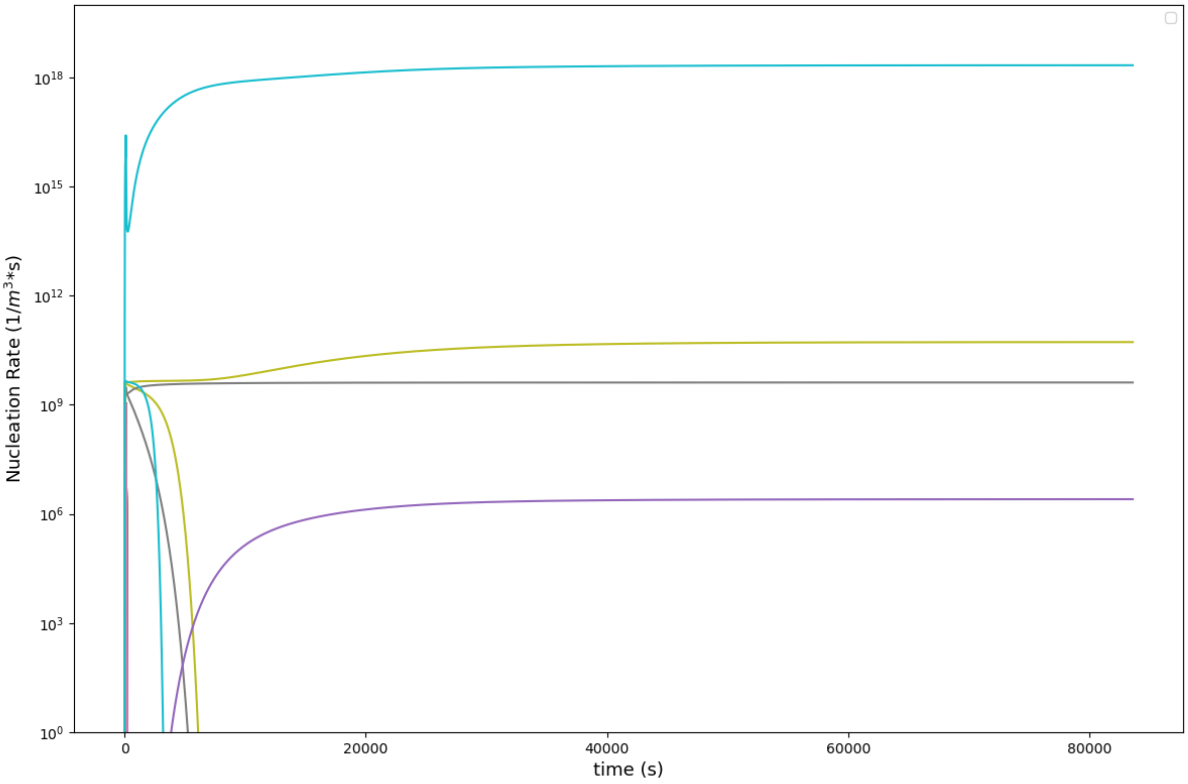


Figure 4.17 Nucleation rate over 60 compartments dividing the reactor with the UDF

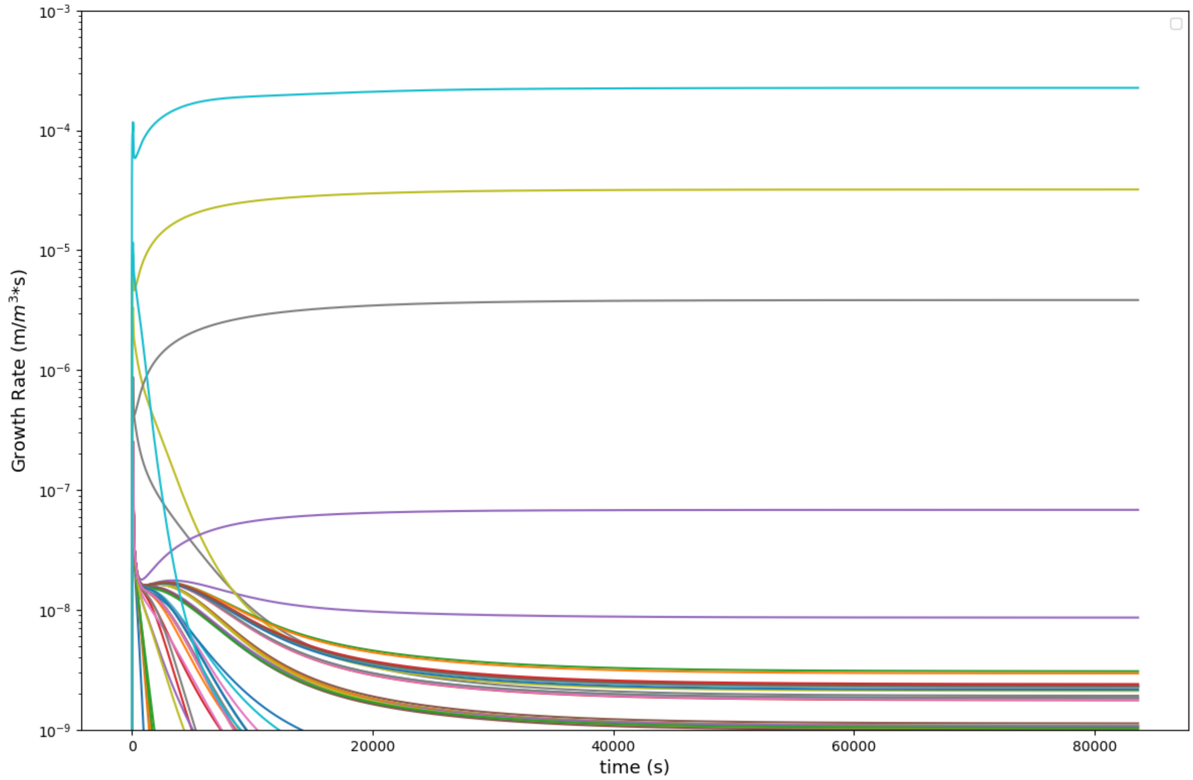


Figure 4.18 Growth rate over 60 compartments dividing the reactor with the UDF

In order to explore the role of the number of compartments on the final model accuracy, this variable was further increased up to 70. Figure 4.23 reports the compartments distribution in the plane containing the inlets of the metals and NaOH.

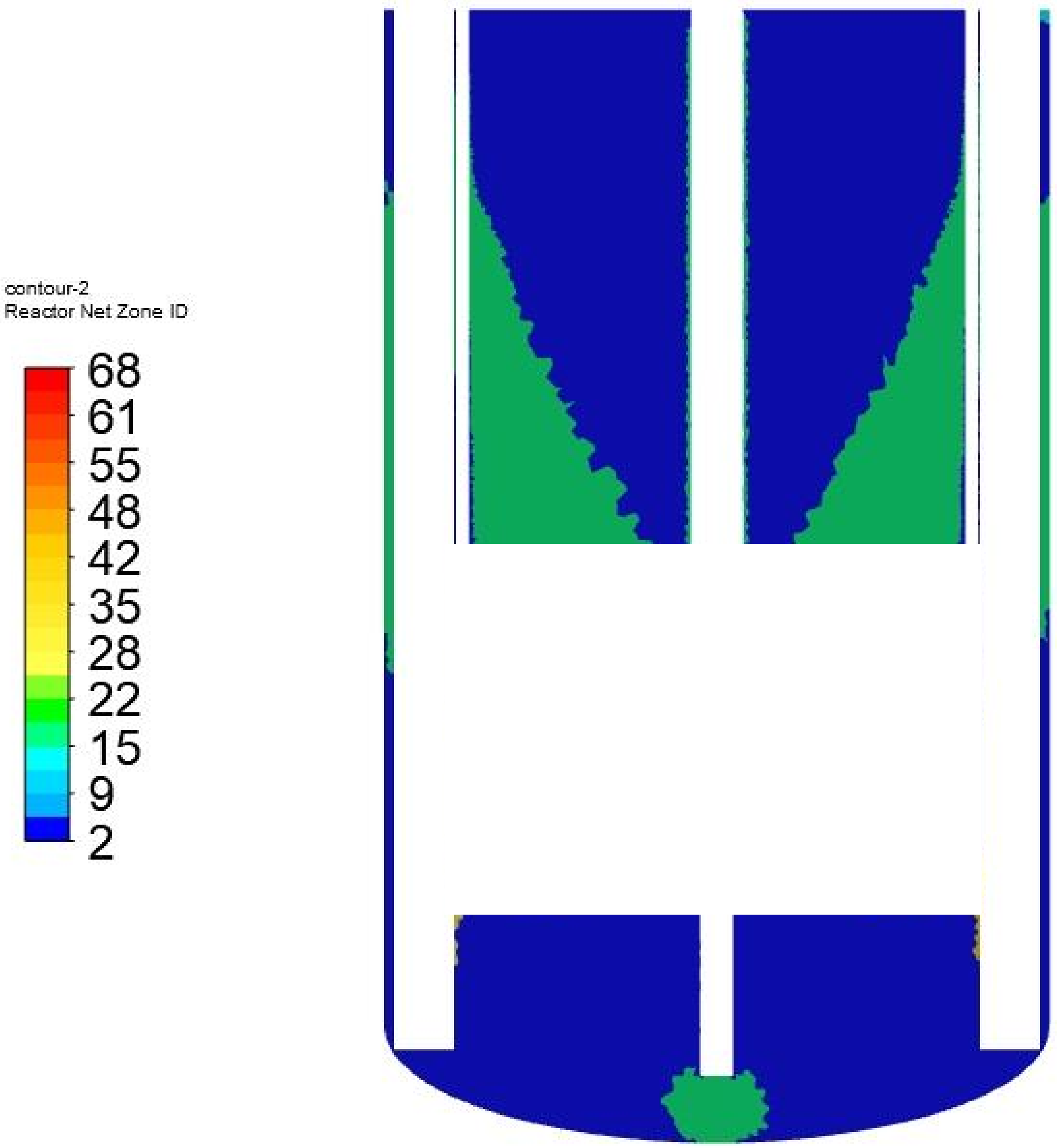


Figure 4.19 Compartments (70) using the UDF in a plane containing metals and NaOH inlets

Providing the script with information about these compartments, the concentrations and moments obtained at the outlet of the reactor are shown in table 4.5.

Outlet	
Ni^{2+} (kmol/m^3)	4,94x10^{-7}
Mn^{2+} (kmol/m^3)	1,65x10^{-7}

Co^{2+} (kmol/m^3)	1,65x10^{-7}
NH_3 (kmol/m^3)	8,74x10^{-1}
m_0 (1/m^3)	6,7x10^{+12}
m_1 (m/m^3)	3,98x10^{+7}
m_2 (m^2/m^3)	1,17x10^{+3}
m_3 (m^3/m^3)	5,57x10^{-2}
SMD	4,8x10^{-5}

Table 4.5 Concentrations, moments and SMD (Sauter Mean Diameter) at outlet of the 70 compartments model created with the UDF

A greater number of compartments at inlets is reached (figure 4.24). The compartment directly near the metal inlet face is halved in dimensions compared to that obtained with 60 compartments, one half still being the one in which metals enter, the other half directly connected with the first half. This leads mainly to a higher value of the total particle number density, represented by the moment of order zero of the PSD. This value is about 75% higher since nucleation is better estimated and then the Sauter mean diameter decreases by about 15%, since there are more and smaller particles.

The aggregation rate is again not negligible just in the two smaller compartments at metals inlet, in which growth rate is larger than orders of magnitude (figure 4.28).

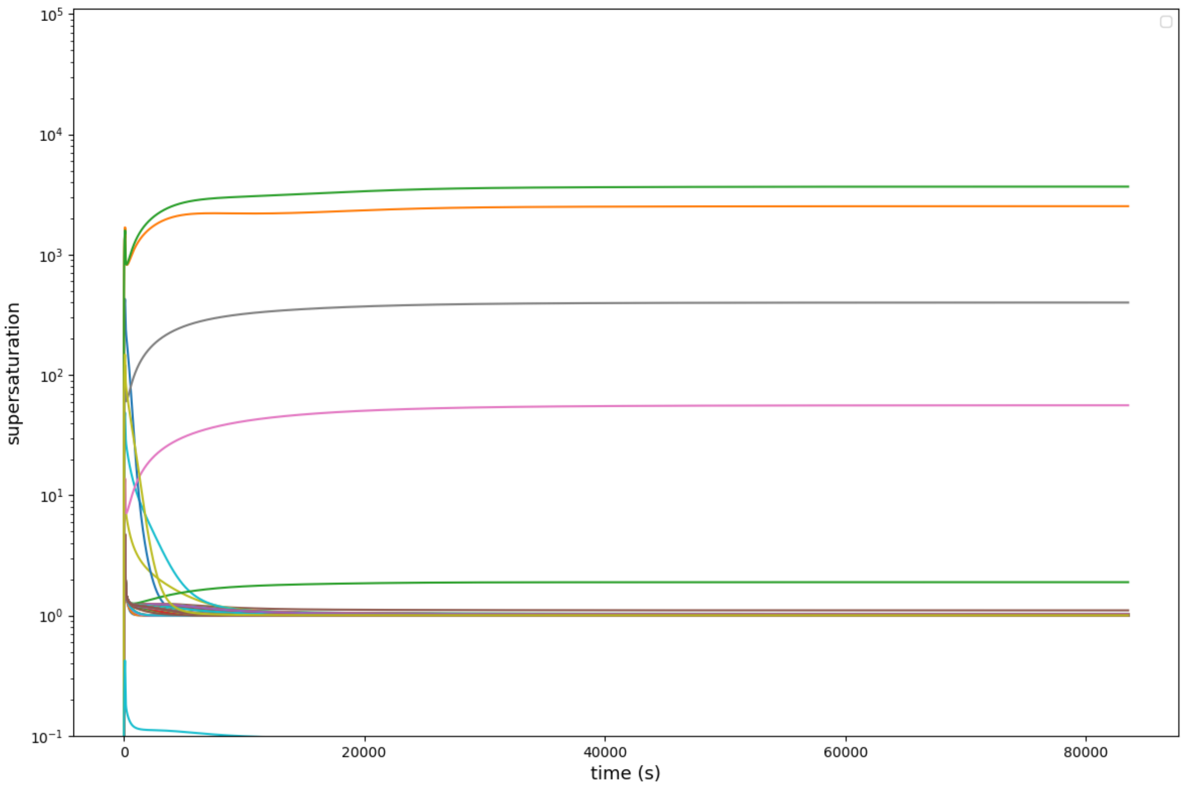


Figure 4.20 Supersaturation over 70 compartments dividing the reactor with the UDF

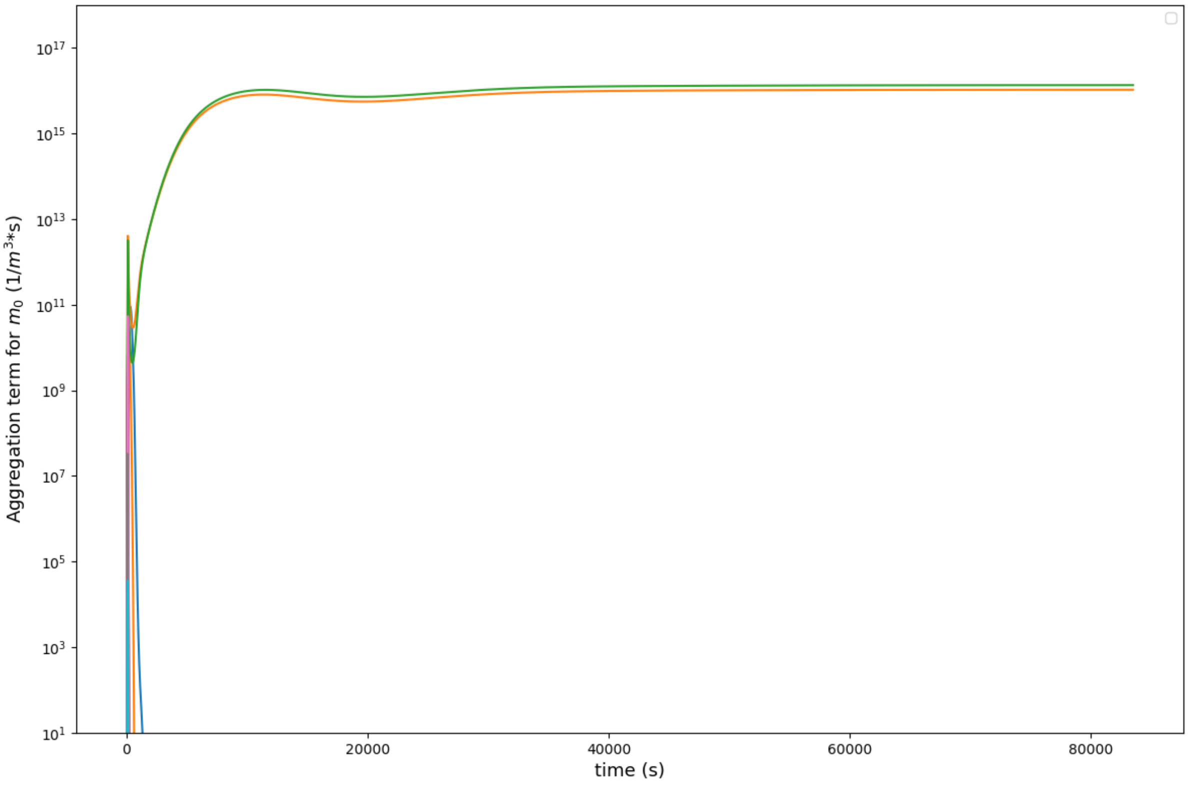


Figure 4.21 Aggregation contribute to the moment of order 0 over 70 compartments dividing the reactor with the UDF (similar behaviour for the other moments)

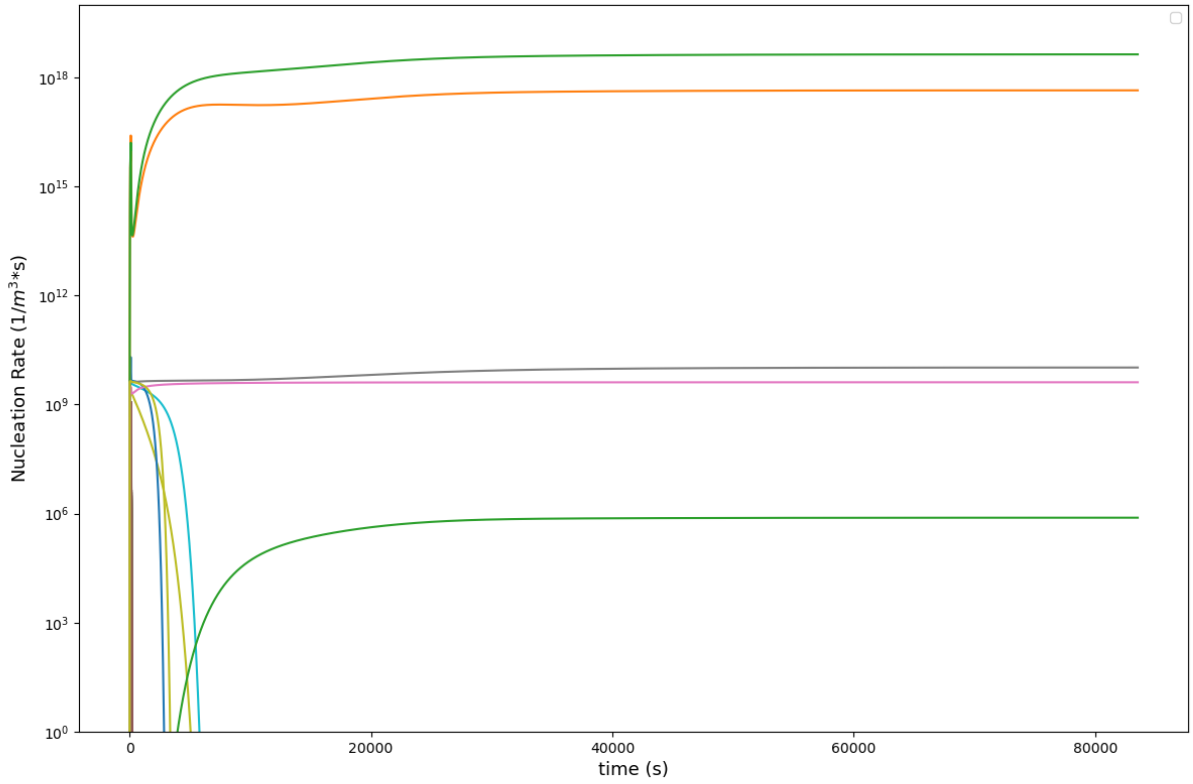


Figure 4.22 Nucleation rate over 70 compartments dividing the reactor with the UDF

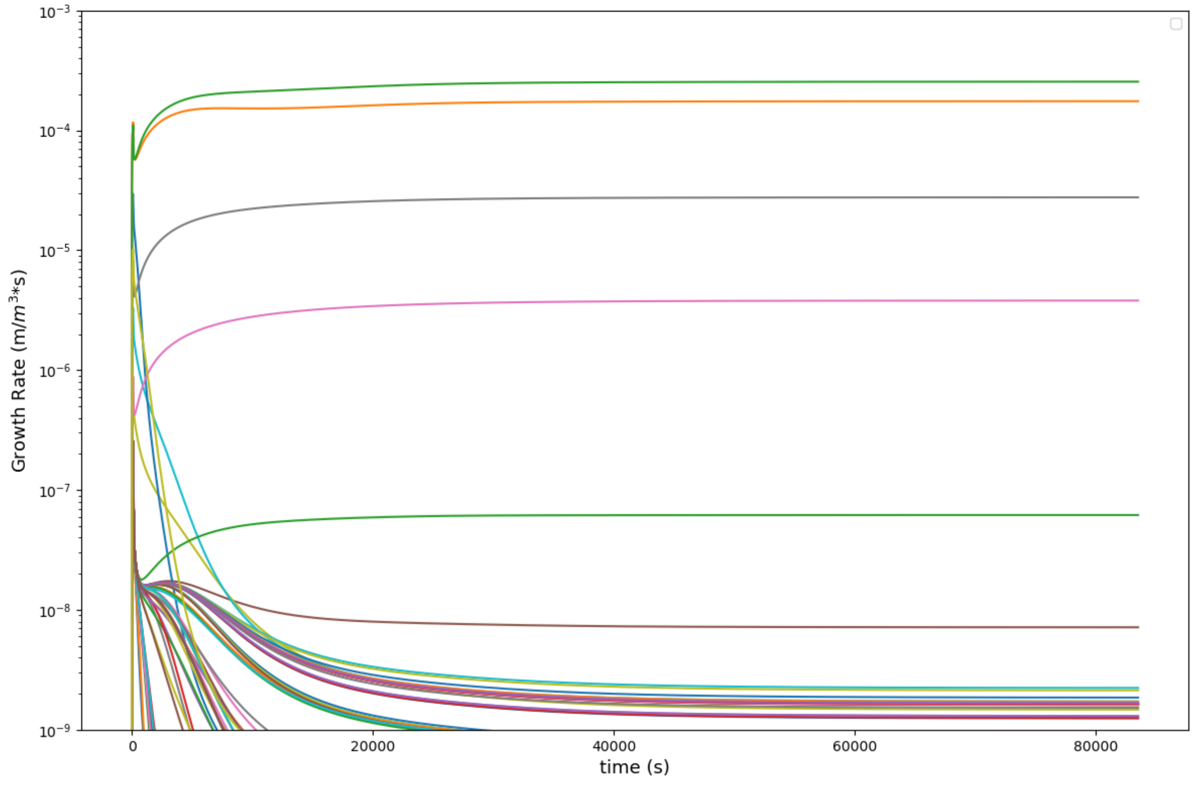


Figure 4.23 Growth rate over 70 compartments dividing the reactor with the UDF

5. CONCLUSIONS

In this work, a compartment model has been studied in order to solve the equations describing the precipitation of Nickel, Manganese and Cobalt hydroxides in a stirred tank reactor in steady state conditions without performing a fully coupled CFD-PBE simulation. The compartment model consists in the division of the reactor in compartments interconnected in which the governing equations for precipitation are solved by considering the compartments as perfectly mixed systems. The calculations are carried by a script written in Python language.

Performing a full coupled CFD-PBE simulation requires the solution of the precipitation governing equations in every cell of the domain, leading to high computational costs and long simulation times (in the order of magnitude of weeks). The use of a compartment model solved by the script is an approximation that significantly reduces the computational costs. For example with just one processor the typical simulation time for the compartment model, varies from 12 hours with 20 compartments to 48-72 hours with 70 compartments.

A crucial aspect of the compartment model is how the division of the reactor is carried out. Since the kinetics on variables that vary in the reactor, such as the supersaturation and the turbulent dissipation rate (aggregation), the compartments created must describe as best as possible their gradients in the reactor.

A CFD simulation of the reactor is carried in order to obtain the velocity and turbulence fields, using the Multiple Reference Frame model to describe the motion of the fluid around the moving parts and using as turbulence model the standard k- ε.

The supersaturation field is obtained from an on-going full coupled CFD-PBE simulation. The field is still not the steady one but the supersaturation distribution and its gradients are reasonably similar to those expected.

The division of the reactor is carried by ANSYS Fluent, which has implemented the Reactor Network Tool, useful to agglomerate the cells of the domain in compartments based on the selected Custom Field Functions.

By using just the logarithm of the turbulent dissipation rate (from the CFD simulation) to divide the reactor, the compartments at inlets were too big and the moments distribution was low, since nucleation and growth were underestimated.

By using just the logarithm of the supersaturation, turbulent dissipation rate gradient is not considered, leading to possible bad estimations of the aggregation rate.

By using two Custom Field Functions, one with the turbulent dissipation rate and one with the supersaturation, the reactor is divided with small compartments created by supersaturation at inlets and with compartments created by turbulent dissipation rate in the rest of the reactor.

Increasing the numbers of reactors has been seen to create smaller compartments at inlets, in particular at metal inlet, better estimating nucleation and growth and leading to particles with smaller SMD at outlet.

By analysing the aggregation rate over compartments, it was clear that it was not negligible only in the compartments directly near to the metals inlet face: this is due to the dependence of the aggregation efficiency on the growth rate in an exponential term, leading to aggregation efficiency of almost zero, no matter the turbulent dissipation rate, in the others compartments because of the difference in order of magnitudes of the growth rate term.

Therefore, with the kinetics used in this work turbulent dissipation rate division is not much incident; the main parameter influencing the result is the dimension of the compartments directly near the metals inlet face.

Then to create a compartment model, dissipation rate can surely be important, but the creation of small compartments positioned on the metals inlet face is fundamental.

A method to create compartments considering both parameters has been defined and the importance of the compartments near the inlets have been showed.

Future perspectives of the work are a comparison of the results with a full CFD-PBE coupled simulation and a breakage model, for which turbulent dissipation rate division could be more impacting, and a parallelization of the script, in order to solve the system much faster.

6. BIBLIOGRAPHY

Anderson J. D. Jr., 1995, Computational fluid dynamics: the basics with applications. McGraw-Hill, Inc., Department of Aerospace Engineering, University of Maryland, USA.

ANSYS Inc., 2013. *ANSYS Fluent theory guide*. 15th ed. s.l.: SAS IP, Inc.

ANSYS Inc., 2013. *ANSYS Fluent UDF manual.* 15th ed. s.l.: SAS IP, Inc.

ANSYS Inc., 2013. *ANSYS Fluent user's guide*. 15th ed. s.l.: SAS IP, Inc.

ANSYS Inc., 2017. *ANSYS Fluent tutorial guide.* 18th ed. s.l.: SAS IP, Inc.

Barai P., Feng Z., Kondo H., Srinivasan V., 2019, Multiscale computational model for particle size evolution during coprecipitation of li-ion battery cathode precursors. In *Journal of Physical Chemistry B,* **123**, pp. 3291-3303. [DOI: 10.1021/acs.jpcb.8b12004]

Bird B. R., Stewart W. E., Lightfoot E. N., Transport phenomena. John Wiley and Sons, Inc, Chemical Engineering Department, University of Wisconsin-Madison, USA.

Dirksen J. A., Ring T. A., 1991, Fundamentals of crystallization: kinetic effects on particle size distributions and morphology. In *Chemical Engineering Science,* **46**, pp. 2389-2427.

Dorao C. A., Jakobsen H. A, 2006, Numerical calculation of the moments of the population balance equation. In *Journal of Computational and Applied Mathematics*, **196**, pp. 619-633. [DOI: 10.1016/j.cam.2005.10.015]

Fang S., Jackson D., Dreibelbis M. L., Kuech T. F., Hamers R. J., 2018, Anode-originated SEI migration contributes to formation of cathodeelectrolyte interphase layer. In *Journal of Power Sources*, **373**, pp. 184-192. [DOI: 10.1016/j.jpowsour.2017.09.050]

Gavi E., Rivautella L., Marchisio D. L., Vanni M., Barresi A. A., Baldi G., 2007, CFD Modelling of nano-particle precipitation in confined impinging jet reactors. In *Institution of Chemical Engineers,* **85**, pp. 735-744. [DOI: 10.1205/cherd06176]

Launder B. E., Spalding D. B., 1974, The numerical computation of turbulent flows. In *Computer Methods in Applied Mechanics and Engineering*, **3**, pp. 269-289- [DOI: 10.1016/0045-7825(74)90029-2]

Li L., Li X., Wang Z., Guo H., Yue P., Chen W., Wu L., 2011, A simple and effective method to synthesize layered $LiNi_{0.8}Co_{0.1}Mn_{0.1}O_2$ cathode materials for lithium ion battery. In *Powder Technology,* **206**, pp. 3353-357. [DOI: 10.1016/j.powtec.2010.09.010]

Manthiram A., 2020, A reflection on lithium-ion battery cathode chemistry. In *Materials Science and Engineering Program.* University of Texas at Austin, Austin, USA.

Marchisio D. L., Fox R. O., 2013, Computational models for polydisperse particulate and multiphase systems. Cambridge University Press, New York, USA.

Marchisio D. L., Fox R. O., Solution of population balance questions using the direct quadrature method of moments. In *Journal of Aerosol Science,* **36**, pp. 43-73. [DOI: 10.1016/j.jaerosci.2004.07.009]

Marchisio D. L., Vigil D. R., Fox R.O., 2003, Quadrature method of moments for aggregation-breakage processes. In *Journal of Colloid and Interface Science*, **258**, pp. 322-334. [DOI: 10.1016/S0021-9797(02)00054-1]

Mersmann A., 2001, Crystallization techonology handbook. CRC Press, Technical University of Munich, Garching, Germany.

Moukalled F., Mangani L., Darwish M., 2016, The finite volume metheod in computational fluid dynamics. Springer International Publishing, Switzerland. [DOI: 10.1007/978-3-319-16874-6]

Omar H., Sohrab R., 2017, Crystal population balance formulation and solution methods: a review. In *Crystal Growth and Design*, **17**, pp. 4028-4041. [DOI: 10.1021/acs.cgd.7b00645]

Randolph A. D., Larson M. A., 1962, Transient and steady state size distributions in continuous mixed suspension crystallizers. In *American Institute of Chemical Engineers*, **8**, pp. 639-645.

Richardson J.F., Backhurst J.R., 2002, Chemical Engineering Series. Butterworth-Heinemann Books, pp. 827-900. University of Newcastle, UK.

Ullah A., Majid A., Rani N., 2017, A review on first principles based studies for improvement of cathode material of lithium ion batteries. In *Journal of Energy Chemistry*, **27**, pp. 219-237. [DOI: 0.1016/j.jechem.2017.09.007]

Van Bommel A., Dahn J. R., 2009, Analysis of the grow mechanism of coprecipitated spherical and dense nickel, manganese, and cobalt-containing hydroxides in the presence of aqueous ammonia. In *Chemistry of Materials*, **21 (8)**, pp. 1500-1503. [DOI: 10.1021/cm803144d]

Yuan C., Fox R. O., 2011, Conditional quadrature method of moments for kinetic equations. In *Journal of Computational Physics,* **230**, pp. 8216-8246. [DOI: 10.1016/j.jcp.2011.07.020]

Yang C., Mao Z., 2014, Numerical simulation of multiphase reactors with continuous liquid phase. Elsevier Ltd, Institute of Process Engineering, Chinese Academy of Sciences, pp. 263-294. [DOI: 10.1016/b978-0-08-099919-7.00006-8]

List of figures

List of tables