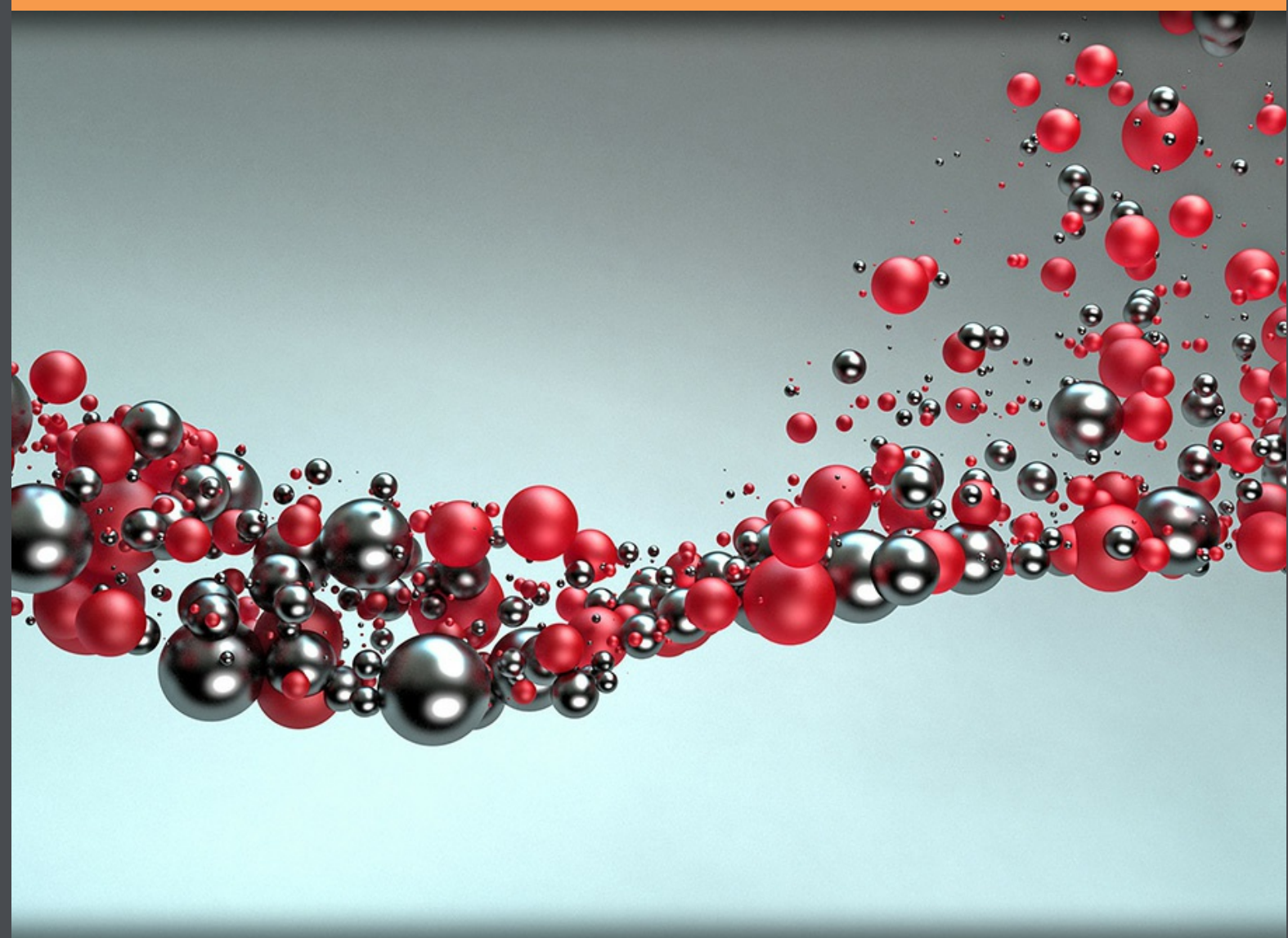


Chemical Reaction Engineering with IPython: Part I

Transport Processes and Reaction in Porous Pellets

Boris Golman



Download free books at

bookboon.com

BORIS GOLMAN

CHEMICAL REACTION ENGINEERING WITH IPYTHON PART I

TRANSPORT PROCESSES
AND REACTION IN
POROUS PELLETS

Chemical Reaction Engineering with IPython

Part I: Transport Processes and Reaction in Porous Pellets

1st edition

© 2016 Boris Golman & bookboon.com

ISBN 978-87-403-1316-1

Peer reviewed by Viatcheslav Kafarov, Dean of Engineering Faculty, Director of the Center for Sustainable Development in Energy and Industry, Professor of Chemical Engineering Department, Industrial University of Santander

CONTENTS

1	Introduction	6
1.1	General consideration on catalytic reaction in porous pellets	7
1.2	Mechanism of mass transfer in porous media	9
1.3	Mechanism of heat transfer in porous media	11
2	First-order Reaction in Isothermal Catalyst Pellet	12
2.1	Derivation of mass balance equation	13
2.2	Analytical solution of mass balance equation	17
2.3	Computer programs and simulation results	23
3	Second-order Reaction in Isothermal Catalyst Pellet	38
3.1	Mass balance equation	38
3.2	Numerical solution of model equation using orthogonal collocation method	39
3.3	Computer programs and numerical results	43



**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider

4	Chemical Reaction in Non-Isothermal Catalyst Pellet	57
4.1	Derivation of heat balance equation	57
4.2	Numerical solution of model equations using finite-difference method	64
4.3	Computer program description	72
4.4	Numerical results	83
5	Enzyme catalyzed reaction in isothermal pellet	87
5.1	Derivation of mass balance equation	88
5.2	Numerical implementation	91
5.3	Computer program description and numerical results	95
6	Non-catalytic Chemical Reaction in Agglomerate of Fine Particles	105
6.1	Derivation of mathematical model equations	106
6.2	Computational procedure using the method of lines	110
6.3	Program description	112
6.4	Numerical results	124
	Summary	131
	References	132
	Appendix A1. Installing IPython	134
	Appendix A2. Brief Overview of Python Language	138
	Appendix A3. Auxiliary Programs used in Orthogonal Collocation Method	142

1 INTRODUCTION

The focus of this textbook is to discuss both catalytic and non-catalytic chemical reactions that take place in a porous pellet. The target audience are advanced undergraduate or graduate students in the chemical engineering or in related areas. This textbook has been written to fulfill three major goals:

1. To introduce the mathematical models describing the chemical reactions accompanied by heat and mass transfer in the pellets.
2. To explain the numerical or analytical methods for solving the model equations.
3. To discuss the numerical results.

The features of this book can be summarized as follows: (a) model equations are fully derived, (b) all chapters and all figures are illustrated with computer programs and (c) programs are explained in the text. Computer programs are available to download on Bookboon's companion website.

The programs are written in Python and implemented as IPython notebooks. SciPy, NumPy and Matplotlib libraries are used to numerically solve the model equations and to visualize simulated results. All of these tools are easy to use, well supported by a large online community, and available for free. The installation of IPython system is explained in Appendix A1 and the brief overview of python computer language is given in Appendix A2. Using the developed tools, readers will be able to solve problems that appear in their study or research in the future.

We begin this book by reviewing the mechanism of mass and heat transfer in a porous media. Then we derive the mass balance equation and solve it analytically for the first-order reaction in isothermal spherical pellet. The following chapter describes the second-order reaction in isothermal pellet and an orthogonal collocation method is introduced as a numerical method for solving model equations. Then we discuss the chemical reaction in the non-isothermal pellet. We derive the heat balance equation and show how to solve numerically the system of mass and heat balance equations using a finite-difference method. Next we discuss the enzymatic reaction taking place in the pellet. We close the book with the chapter describing the non-catalytic reaction in an agglomerate of submicron particles. In this example we take into account the change in the agglomerate porous structure with reaction progress. We use a method of lines to solve the unsteady-state mass and heat balances.

Finally, the author wish to acknowledge and thank his wife, Nadezda, and his sons, Mikhail and Iakov, for their patient support and assistance during the preparation of this book.

1.1 GENERAL CONSIDERATION ON CATALYTIC REACTION IN POROUS PELLETS

Before we can derive the differential equation describing the chemical reaction, mass and heat transfer in a porous pellet, we need to consider the general steps through which the reaction proceeds and discuss the mechanisms of mass and heat transfer in porous media.

Here, we assume that the catalyst pellets are manufactured by agglomeration of primary fine particles. The catalytic material is dispersed in the micropores of primary particles. The void spaces among particles form macropores bounded by the outer particle surfaces, as shown in Fig. 1.1. The heterogeneously catalyzed reaction $A \rightarrow B$ takes place on active sites in the micropores of primary particles. The reaction proceeds through the following sequential steps:

- Diffusion of the gaseous reactant A from the bulk phase to the external pellet surface through a boundary layer located at the external surface of the pellet.
- Diffusion of the reactant A in the macropore spaces to the outer surface of primary particles. Then, the reactant A diffuses in the micropore from the pore mouth to the point where adsorption and reaction take place.
- Adsorption of the reactant A on the active catalytic site.
- Surface reaction of the adsorbed species A to produce the product B adsorbed on active site.
- Desorption of the product B .
- Diffusion of B through the micropore and macropore porous spaces to the external pellet surface.
- Diffusion of the product B from the external pellet surface into the bulk gas phase through the boundary layer.

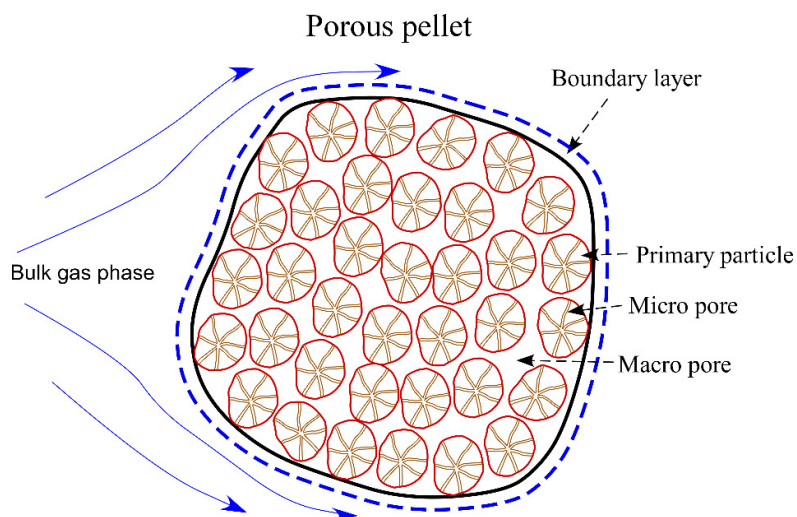


Figure 1.1: Illustration of sequential steps in reaction process in porous catalyst pellet.

An overall rate of reaction can be limited by the intrinsic rate of surface catalytic reaction, rate of mass transfer of reactant or product inside the catalyst pellet, rate of mass transfer through the boundary layer outside the pellet or by any combination of these processes. At the low temperature and for slow reactions, the intrinsic rate of surface reaction is slow, resulting in the absence of the concentration gradient inside and outside catalyst pellet. If the intrinsic rate of surface reaction has similar magnitude or faster rate than the mass transfer rates, the concentration gradient will developed in the pellet or in the boundary layer around catalyst pellet.

To characterize the ratio of intrinsic reaction rate to the rate of mass transfer, we introduce a catalytic effectiveness factor η , which is defined as the ratio of observed rate of reaction to the rate of reaction at the surface concentration, C_{A_0} . It accounts for the extent of reduction in the overall reaction rate due to the lower concentration of reactant inside the catalyst pellet as compared to the surface concentration. If the effectiveness factor is close to one, the all internal surface of catalyst pellet are utilized and the reaction rate at the pellet center is the same as the rate at the outer surface. In the case when effectiveness factor is approaching zero, only the outer surface of catalyst pellet is used, and the intrapellet diffusion will reduce the overall reaction rate. This usually occurs for active catalyst or when using the large pellet of low porosity.

1.2 MECHANISM OF MASS TRANSFER IN POROUS MEDIA

Depending on the pellet pore size, different mechanisms of mass transfer can be observed, such as ordinary bulk diffusion, Knudsen diffusion and surface diffusion (Froment et al. 2011, p. 172). For very large pores, the bulk flow should be taken into account. When the pore diameter is much larger than the mean free path of the diffusing molecule, the molecules are transported by ordinary bulk diffusion. The Knudsen diffusion is responsible for the mass transfer when the molecule mean free path is larger than the pore diameter. The surface diffusion is a dominant mechanism of mass transfer in the microporous pellet with pore diameter close to the size of diffusing molecule.

We can estimate the diffusion coefficient for a binary gas system at given temperature T using the Chapman-Enskog formula (Bird et al. 2002, p. 526):

$$D_{mi} = 1.8583 \times 10^{-7} \frac{T^{3/2} [(M_i + M_m) / M_i M_m]}{P \sigma_{mi}^2 \Omega_{mi}}, \quad (1.1)$$

where M_i and M_m are the molecular weights of i species and carrier gas m , respectively, P is the total pressure of gas mixture, σ_{mi} is the characteristic diameter of the binary mixture and Ω_{mi} is the dimensionless collision integral.

The following empirical approximation is used for estimation of Ω_{mi} :

$$\Omega_{mi} = \frac{A}{(T^*)^B} + \frac{C}{e^{D \cdot T^*}} + \frac{E}{e^{F \cdot T^*}} + \frac{G}{e^{H \cdot T^*}} \quad (1.2)$$

Values of constants A , B , C , D , E , F , G and H are given in Reid et al. (1987) as:

$A = 1.06036$, $B = 0.1561$, $C = 0.193$, $D = 0.47635$, $E = 1.053587$, $F = 1.52996$, $G = 1.76474$,
 $H = 3.89411$.

The dimensionless temperature is given by

$$T^* = k_B T / \varepsilon_{mi} \quad (1.3)$$

where k_B is the Boltzman's constant and ε_{mi} is the characteristic energy of the binary mixture. The following combining rules are used to determine ε_{mi} and σ_{mi} :

$$\varepsilon_{mi} = (\varepsilon_{ii} \varepsilon_{mm})^{1/2}, \sigma_{mi} = (\sigma_{ii} + \sigma_{mm}) / 2 \quad (1.4)$$

where ε_{ii} and σ_{mm} are the characteristic energy and the diameter for like pairs ii and mm , respectively.

We can calculate the Knudsen diffusivity using the correlation resulting from the kinetic theory of gases for a cylindrical capillary of a mean radius \bar{a} at normal pressure (Froment et al. 2011, p. 173):

$$D_{Ki} = 0.96987 \cdot \bar{a} \cdot \sqrt{\frac{T}{M_i}} \quad (1.5)$$

The mean radius of capillary is estimated as

$$\bar{a} = \frac{2\varepsilon}{S}, \quad (1.6)$$

where ε is the voidage and S is the specific surface area.

The combined diffusivity to describe the transition from ordinary molecular diffusion to Knudsen diffusion is given as

$$D_{ci} = \frac{1}{\frac{1}{D_{Ki}} + \frac{1 - \alpha \cdot y_i}{D_{mi}}}, \quad (1.7)$$

where y_i is the mole fraction of species i in the gas phase. Here, α is defined as

$$\alpha = 1 + \frac{N_m}{N_i},$$

where N_i and N_m are the molar fluxes of species i and m relative to the fixed coordinate system. In the case of equimolar counter-diffusion, $N_m = -N_i$ and Eq. (1.7) becomes

$$D_{ci} = \frac{1}{\frac{1}{D_{Ki}} + \frac{1}{D_{mi}}} \quad (1.8)$$

We describe the mass and heat transport with chemical reaction in a porous catalyst pellet using a concept of effective properties. The corresponding fluxes and reaction rates are averaged over a volume which is small relative to the pellet volume, but large enough with respect to primary particles and pore sizes.

The effective diffusivity of the i species, $D_{\text{eff},i}$, is frequently evaluated using the following correlation:

$$D_{\text{eff},i} = \frac{\varepsilon}{\zeta} \cdot D_{ci}, \quad (1.9)$$

where ζ is the tortuosity factor that accounts for increasing length of diffusional path and varying pore cross section (Butt 2000, p. 495). Using a random pore model, Wakao and Smith (1962) postulated that the tortuosity factor is in inverse proportion to the void fraction:

$$\zeta = \frac{1}{\varepsilon} \quad (1.10)$$

Thus, the effective diffusivity can be estimated as

$$D_{\text{eff},i} = \varepsilon^2 \cdot \frac{D_{\text{Ki}} \cdot D_{\text{mi}}}{D_{\text{Ki}} + D_{\text{mi}}}. \quad (1.11)$$

1.3 MECHANISM OF HEAT TRANSFER IN POROUS MEDIA

The effective thermal conductivity of a porous pellet depends in a complex manner on the geometry of porous space, and thermal conductivities of solid and fluid phases. The two limiting cases could be considered when the heat conduction in both phases occurs in parallel or in series. If the conduction in the solid and fluid phases takes place in parallel, the maximum value of effective conductivity could be achieved, because the effective conductivity is given as the weighted arithmetic mean of the phase conductivities:

$$k_{\text{eff}} = (1 - \varepsilon) \cdot k_s + \varepsilon \cdot k_f, \quad (1.12)$$

where k_s and k_f are the thermal conductivities of solid and fluid phases.

If the conduction proceeds in such a way that all heat passes through the solid phase and then through the fluid phase in series, the minimum value of effective conductivity is obtained. k_{eff} is given as the harmonic mean of k_s and k_f :

$$\frac{1}{k_{\text{eff}}} = \frac{1 - \varepsilon}{k_s} + \frac{\varepsilon}{k_f} \quad (1.13)$$

Assuming that the solid and fluid phases are distributed randomly, Woodside and Messmer (1961) derived the following expression:

$$k_{\text{eff}} = k_f \cdot \left(\frac{k_s}{k_f} \right)^{1-\varepsilon} \quad (1.14)$$

2 FIRST-ORDER REACTION IN ISOTHERMAL CATALYST PELLET

In this chapter, you will learn to:

1. Derive a mass balance equation for the reactant that accounts for the diffusion and first-order catalytic reaction in the isothermal spherical pellet.
2. Solve analytically the model equation.
3. Plot the reactant concentration profiles in the pellet and calculate the effectiveness factors for various values of process parameters using the elaborated IPython notebooks.



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



2.1 DERIVATION OF MASS BALANCE EQUATION

We first consider a first-order reaction $A \rightarrow B$ in an isothermal catalyst pellet of spherical shape. We use Fick's law to relate the diffusive flux of reactant A to the concentration gradient in the radial direction of the pellet under the assumption of dilute gas mixture:

$$N_{Ar} = -D_{\text{eff},A} \frac{dC_A}{dr}, \quad (2.1)$$

where N_{Ar} is the diffusive flux based on the total area of the spherical shell, πr^2 , including voids and solid, and C_A is the concentration of the gas species A within the pores.

We can perform a steady-state mass balance for species A over a spherical shell of thickness Δr located at radius r within a catalyst pellet as (Fogler 2008)

$$\left(\begin{array}{c} \text{Rate of input} \\ \text{of species } A \\ \text{by diffusion at } r \\ \text{(moles/time)} \end{array} \right) - \left(\begin{array}{c} \text{Rate of output} \\ \text{of species } A \\ \text{by diffusion at } r + \Delta r \\ \text{(moles/time)} \end{array} \right) + \left(\begin{array}{c} \text{Rate of generation} \\ \text{of species } A \\ \text{by reaction within } \Delta r \\ \text{(moles/time)} \end{array} \right) = 0$$

The molar rate of production of component A by the first-order reaction within the differential volume element, $4\pi r^2 \Delta r$, is

$$r_A \cdot 4\pi r^2 \Delta r = -kC_A \cdot 4\pi r^2 \Delta r \quad (2.2)$$

Thus, we can write the mass balance as

$$(N_{Ar} \times 4\pi r^2) \Big|_r - (N_{Ar} \times 4\pi r^2) \Big|_{r+\Delta r} + r_A \cdot 4\pi r^2 \Delta r = 0 \quad (2.3)$$

Dividing by $4\pi \Delta r$, we find:

$$-\left(\frac{(r^2 \cdot N_{Ar}) \Big|_{r+\Delta r} - (r^2 \cdot N_{Ar}) \Big|_r}{\Delta r} \right) + r^2 \cdot r_A = 0$$

Taking the limit as Δr goes to zero and using the definition of the first derivative gives

$$\frac{d(r^2 N_{Ar})}{dr} - r^2 \cdot r_A = 0 \quad (2.4)$$

Substituting the flux by Eq. (2.1) and the reaction rate by Eq. (2.2) into Eq. (2.4), we have:

$$\frac{d}{dr} \left(-r^2 D_{\text{eff},A} \cdot \frac{dC_A}{dr} \right) + r^2 \cdot kC_A = 0 \quad (2.5)$$

Changing the sign in Eq. (2.5) gives

$$\frac{d}{dr} \left(r^2 D_{\text{eff},A} \cdot \frac{dC_A}{dr} \right) - r^2 \cdot k C_A = 0$$

Assuming a constant effective diffusivity $D_{\text{eff},A}$, we rearrange the above equation as

$$\frac{D_{\text{eff},A}}{r^2} \frac{d}{dr} \left(r^2 \cdot \frac{dC_A}{dr} \right) - k C_A = 0 \quad (2.6)$$

Using the chain rule of differentiation, we write the term with the second derivative as

$$\frac{d}{dr} \left(r^2 \cdot \frac{dC_A}{dr} \right) = \frac{dr^2}{dr} \frac{dC_A}{dr} + r^2 \frac{d^2 C_A}{dr^2} = 2r \frac{dC_A}{dr} + r^2 \frac{d^2 C_A}{dr^2} \quad (2.7)$$

Introducing Eq. (2.6) into Eq. (2.7), we derive the mass balance equation for the first-order reaction in catalyst pellet as

$$\left(\frac{d^2 C_A}{dr^2} + \frac{2}{r} \frac{dC_A}{dr} \right) - \frac{k C_A}{D_{\text{eff},A}} = 0 \quad (2.8)$$

The boundary conditions are

- At the center of catalyst pellet:

There is no diffusive flux through the pellet center since this is a point of symmetry.

$$\frac{dC_A}{dr} = 0 \quad \text{at} \quad r = 0 \quad (2.9)$$

- At the external surface of catalyst pellet:

- o Fixed reactant concentration at the external surface.

We assume that the concentration of reactant species A at the external pellet surface, C_{As} , is equal to the bulk phase concentration, C_{Ab} .

$$C_A = C_{As} = C_{Ab} \quad \text{at} \quad r = R, \quad (2.10)$$

where R is the pellet radius.

- o Mass transfer across the boundary at the pellet external surface.

We derive the steady state mass balance at the pellet external surface as

$$(N_{Ar} \times 4\pi r^2) \Big|_{r=R} - 4\pi r^2 \cdot k_{gA} (C_A \Big|_{r=R} - C_{A,b}) = 0,$$

where k_{gA} is the mass transfer coefficient.

Simplifying the above equation and using the flux definition by Eq. (2.1), we find:

$$k_{gA}(C_{A,b} - C_A|_{r=R}) = D_{\text{eff},A} \cdot \frac{dC_A}{dr} \bigg|_{r=R} \quad (2.11)$$

Introducing dimensionless variables, $\rho = \frac{r}{R}$ and $c = \frac{C_A}{C_{Ab}}$, and using the chain rule of differentiation, we can write the first derivative of concentration with respect to radial position as

$$\frac{dC_A}{dr} = \frac{dC_A}{d\rho} \cdot \frac{d\rho}{dr} = \frac{dC_A}{dc} \cdot \frac{dc}{d\rho} \cdot \frac{d\rho}{dr}$$

Using definitions of dimensionless variables, the first derivatives $\frac{dC_A}{dc}$ and $\frac{d\rho}{dr}$ are expressed as

$$\frac{dC_A}{dc} = \frac{d(c \cdot C_{Ab})}{dc} = C_{Ab} \frac{dc}{dc} = C_{Ab}$$

$$\frac{d\rho}{dr} = \frac{d(r/R)}{dr} = \frac{1}{R} \frac{dr}{dr} = \frac{1}{R}$$

Therefore, we can rewrite $\frac{dC_A}{dr}$ as

$$\frac{dC_A}{dr} = C_{Ab} \cdot \frac{dc}{d\rho} \cdot \frac{1}{R} = \frac{C_{Ab}}{R} \cdot \frac{dc}{d\rho}$$

GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead

Similarly, we can express the second derivative $\frac{d^2 C_A}{dr^2}$ in dimensionless form as

$$\frac{d}{dr} \left(\frac{dC_A}{dr} \right) = \frac{d}{d\rho} \left(\frac{C_{Ab}}{R} \cdot \frac{dc}{d\rho} \right) \frac{d\rho}{dr} = \frac{d}{d\rho} \left(\frac{C_{Ab}}{R} \cdot \frac{dc}{d\rho} \right) \frac{1}{R} = \frac{C_{Ab}}{R^2} \frac{d^2 c}{d\rho^2}$$

Substituting the first and second derivatives into Eq. (2.8), we get:

$$\frac{C_{Ab}}{R^2} \frac{d^2 c}{d\rho^2} + \frac{2}{\rho \cdot R} \frac{C_{Ab}}{R} \frac{dc}{d\rho} - \frac{k}{D_{\text{eff},A}} c \cdot C_{Ab} = 0 \quad (2.12)$$

Dividing by $\frac{C_{Ab}}{R^2}$, we have:

$$\frac{d^2 c}{d\rho^2} + \frac{2}{\rho} \cdot \frac{dc}{d\rho} - \frac{k \cdot R^2}{D_{\text{eff},A}} \cdot c = 0 \quad (2.13)$$

Then, we will introduce the dimensionless Thiele modulus, ϕ , defined by

$$\phi^2 = \frac{k \cdot R^2}{D_{\text{eff},A}} \quad (2.14)$$

The Thiele modulus relates the reaction rate to the diffusion rate in the pellet.

Finally, we write the dimensionless mass balance equation describing the first-order reaction in the spherical pellet as

$$\frac{d^2 c}{d\rho^2} + \frac{2}{\rho} \cdot \frac{dc}{d\rho} - \phi^2 \cdot c = 0 \quad (2.15)$$

The dimensionless boundary conditions are

- At the center of catalyst particle:

$$\frac{dc}{d\rho} = 0 \quad \text{at} \quad \rho = 0 \quad (2.16)$$

- At the external surface of catalyst particle:
 - o Fixed reactant concentration at the external surface of the catalyst pellet.

$$c = 1 \quad (2.17)$$

- o Mass transfer across the boundary at the pellet external surface.

$$1 - c = \frac{1}{Bi_m} \cdot \frac{dc}{d\rho} \bigg|_{\rho=1}, \quad (2.18)$$

where $Bi_m = \frac{R \cdot k_{gA}}{D_{\text{eff},A}}$ is the Biot number for mass transfer. This dimensionless group compares the relative external and internal mass transport resistances.

2.2 ANALYTICAL SOLUTION OF MASS BALANCE EQUATION

Multiplying each term in Eq. (2.15) by ρ , we have:

$$\rho \frac{d^2 c}{d\rho^2} + 2 \frac{dc}{d\rho} - \phi^2 \cdot c \cdot \rho = 0 \quad (2.19)$$

We introduce a new variable $u = c \cdot \rho$. The first derivative of u with respect to ρ is $\frac{du}{d\rho} = \rho \cdot \frac{dc}{d\rho} + c$. The second derivative is

$$\frac{d^2 u}{d\rho^2} = \frac{d}{d\rho} \left(\rho \cdot \frac{dc}{d\rho} + c \right) = \frac{d}{d\rho} \left(\rho \cdot \frac{dc}{d\rho} \right) + \frac{dc}{d\rho} = \left\{ \rho \cdot \frac{d^2 c}{d\rho^2} + \frac{dc}{d\rho} \right\} + \frac{dc}{d\rho} = \rho \cdot \frac{d^2 c}{d\rho^2} + 2 \frac{dc}{d\rho}$$

Substituting the second derivative into Eq. (2.19) results in

$$\frac{d^2 u}{d\rho^2} - \phi^2 \cdot u = 0 \quad (2.20)$$

This is a linear second-order differential equation with constant coefficients.

We can write the boundary condition given by Eq. (2.16) in terms of variable u as

$$\frac{dc}{d\rho} = \frac{1}{\rho} \frac{du}{d\rho} - \frac{u}{\rho^2} = 0$$

Multiplying by ρ^2 results in

$$\rho \cdot \frac{du}{d\rho} - u = 0$$

Finely, we get:

$$u = 0 \quad \text{at} \quad \rho = 0 \quad (2.21)$$

Similarly, we rewrite the boundary condition by Eq. (2.17) as

$$u = 1 \quad \text{at} \quad \rho = 1 \quad (2.22)$$

The general solution of Eq. (2.20) is

$$u = C_1 \cdot e^{\phi \rho} + C_2 \cdot e^{-\phi \rho}, \quad (2.23)$$

where C_1 and C_2 are the integration constants. We will find these constants using the boundary conditions. From the boundary condition at $\rho = 0$ by Eq. (2.21) follows

$$C_1 + C_2 = 0$$

Thus,

$$C_2 = -C_1 \quad (2.24)$$

Using the definition of a hyperbolic sine function, $\sinh(x) = \frac{e^x - e^{-x}}{2}$, and substituting Eq. (2.24) into Eq. (2.23), we obtain:

$$u = C_1(e^{\phi \cdot \rho} - e^{-\phi \cdot \rho}) = C_1 \cdot 2\sinh(\phi \cdot \rho) \quad (2.25)$$

2.2.1 FIXED REACTANT CONCENTRATION AT PELLET EXTERNAL SURFACE

We will derive the constant C_1 from the boundary condition at $\rho=1$ by Eq. (2.22) using Eq. (2.25) as

$$1 = C_1 \cdot 2\sinh(\phi)$$

Thus, the constant C_1 is

$$C_1 = \frac{1}{2\sinh(\phi)} \quad (2.26)$$



Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)

Management

Time Management

Problem solving

Self-Confidence

Effectiveness

Project Management

Goal setting

Motivation

Coaching

Download free eBooks at bookboon.com

[Click on the ad to read more](#)

Substituting Eq. (2.26) into Eq. (2.25) gives

$$u = \frac{\sinh(\phi \cdot \rho)}{\sinh(\phi)} \quad (2.27)$$

Finally, back substituting $u = c \cdot \rho$ into Eq. (2.27), we can calculate the dimensionless concentration profile of reactant A in the spherical catalyst pellet as

$$c = \frac{1}{\rho} \cdot \frac{\sinh(\phi \cdot \rho)}{\sinh(\phi)} \quad (2.28)$$

The first derivative of C_A with respect to r is

$$\begin{aligned} \frac{dC_A}{dr} &= \frac{d}{dr}(c \cdot C_{Ab}) = \frac{d}{dr} \left[\frac{C_{Ab} \cdot R}{\sinh(\phi)} \cdot \frac{\sinh\left(\phi \cdot \frac{r}{R}\right)}{r} \right] \\ &= \frac{C_{Ab} \cdot R}{\sinh(\phi)} \cdot \frac{r \cdot \frac{d}{dr} \left\{ \sinh\left(\phi \cdot \frac{r}{R}\right) \right\} - \sinh\left(\phi \cdot \frac{r}{R}\right)}{r^2} \\ &= \frac{C_{Ab} R}{\sinh(\phi)} \cdot \frac{1}{r} \cdot \frac{\phi}{R} \cdot \cosh\left(\phi \cdot \frac{r}{R}\right) - \frac{C_{Ab} R}{r^2} \cdot \frac{\sinh\left(\phi \cdot \frac{r}{R}\right)}{\sinh(\phi)} \\ &= \frac{C_{Ab} \phi}{r} \cdot \frac{\cosh\left(\phi \cdot \frac{r}{R}\right)}{\sinh(\phi)} - \frac{C_{Ab} R}{r^2} \cdot \frac{\sinh\left(\phi \cdot \frac{r}{R}\right)}{\sinh(\phi)} \end{aligned} \quad (2.29)$$

Therefore, we can write the derivative of C_A with respect to r at the pellet surface $r = R$ as

$$\left. \frac{dC_A}{dr} \right|_{r=R} = \frac{C_{Ab} \phi}{R} \frac{1}{\tanh(\phi)} - \frac{C_{Ab}}{R} = \frac{C_{Ab} \phi}{R} \left(\frac{1}{\tanh(\phi)} - \frac{1}{\phi} \right) \quad (2.30)$$

At steady-state, the overall process rate should be equal to the rate of mass transfer into the pellet:

$$\text{Rate} = 4\pi R^2 D_{\text{eff},A} \left(\left. \frac{dC_A}{dr} \right|_{r=R} \right) \quad (2.31)$$

Combining Eqs. (2.30) and (2.31), we get:

$$\text{Rate} = 4\pi R^2 D_{\text{eff},A} \cdot \frac{C_{Ab} \phi}{R} \left(\frac{1}{\tanh(\phi)} - \frac{1}{\phi} \right) \quad (2.32)$$

We can calculate the intrinsic reaction rate in the absence of internal mass transfer limitation by assuming that $C_A = C_{Ab}$ throughout the pellet.

$$\text{Reaction Rate}|_{C_A=C_{Ab}} = \frac{4}{3}\pi R^3 k_1 C_{Ab} \quad (2.33)$$

We define the effectiveness factor η as the ratio of the consumption rate of reactant within the pellet to the intrinsic reaction rate (Kandiyoti 2009,p80). Thus, the effectiveness factor is given by the ratio of Eq. (2.32) to Eq. (2.33) as

$$\eta = \frac{3D_{\text{eff},A}\phi_s}{R^2 k_1} \left[\frac{1}{\tanh(\phi_s)} - \frac{1}{\phi} \right] \quad (2.34)$$

Using the definition of Thiele modulus by Eq. (2.14), we rewrite Eq. (2.34) as

$$\eta = \frac{3}{\phi} \left[\frac{1}{\tanh(\phi)} - \frac{1}{\phi} \right] = \frac{3}{\phi^2} [\phi \cdot \coth(\phi) - 1] \quad (2.35)$$

2.2.2 MASS TRANSFER AT PELLET EXTERNAL SURFACE

In the case of significant mass transfer limitations, we use the boundary conditions at the outer surface of the pellet by Eq. (2.18). We can express this boundary condition in terms of variable u as

$$1 - \frac{u}{\rho} = \frac{1}{Bi_m} \left[\frac{1}{\rho} \frac{du}{d\rho} - \frac{u}{\rho^2} \right] \quad (2.36)$$

Substituting $\rho=1$ in above equation, we get:

$$1 - u = \frac{1}{Bi_m} \left[\frac{du}{d\rho} - u \right] \quad (2.37)$$

We calculate the derivative of u with respect to ρ by differentiation of Eq. (2.25) as

$$\frac{du}{d\rho} = 2C_1 \cdot \phi \cdot \cosh(\phi \cdot \rho) \quad (2.38)$$

Substituting u by Eq. (2.25) and $\frac{du}{d\rho}$ by Eq. (2.38) at $\rho=1$ into Eq. (2.37), we get:

$$1 - 2C_1 \sinh(\phi) = \frac{2C_1}{Bi_m} \cdot [\phi \cosh(\phi) - \sinh(\phi)]$$

Thus, we obtain the constant C_1 as

$$C_1 = \frac{Bi_m}{2(\phi \cdot \cosh(\phi) + \sinh(\phi)(Bi_m - 1))} = \frac{Bi_m}{2(Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi))} \quad (2.39)$$

Substituting C_1 into Eq. (2.25) results in

$$u = \frac{Bi_m \cdot \sinh(\phi \cdot \rho)}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \quad (2.40)$$

Finally, we derive the dimensionless concentration profiles in the pellet as

$$\begin{aligned} c &= \frac{1}{\rho} \cdot \frac{Bi_m \cdot \sinh(\phi \cdot \rho)}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \\ &= \frac{1}{\rho} \cdot \frac{\sinh(\phi \cdot \rho)}{\sinh(\phi) + \frac{\phi \cdot \cosh(\phi) - \sinh(\phi)}{Bi_m}} \end{aligned} \quad (2.41)$$



Brain power

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations.

Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

SKF

The derivative of C_A with respect to r is

$$\begin{aligned}\frac{dC_A}{dr} &= \frac{d}{dr} \left(\frac{C_{Ab} \cdot R}{r} \cdot \frac{Bi_m \cdot \sinh\left(\phi \cdot \frac{r}{R}\right)}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \right) \\ &= \left(\frac{C_{Ab} \cdot R \cdot Bi_m}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \right) \frac{d}{dr} \left(\frac{\sinh\left(\phi \cdot \frac{r}{R}\right)}{r} \right) \\ &= \left(\frac{C_{Ab} \cdot R \cdot Bi_m}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \right) \left[\frac{r \frac{d}{dr} \left(\sinh\left(\phi \cdot \frac{r}{R}\right) \right) - \sinh\left(\phi \cdot \frac{r}{R}\right)}{r^2} \right] \\ &= \left(\frac{C_{Ab} \cdot R \cdot Bi_m}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \right) \left[\frac{\phi \cdot \cosh\left(\phi \cdot \frac{r}{R}\right)}{r \cdot R} - \frac{\sinh\left(\phi \cdot \frac{r}{R}\right)}{r^2} \right]\end{aligned}$$

The derivative of C_A with respect to r at the pellet surface $r = R$ is

$$\begin{aligned}\left. \frac{dC_A}{dr} \right|_{r=R} &= \left(\frac{C_{Ab} \cdot R \cdot Bi_m}{Bi_m \cdot \sinh(\phi) + \phi \cdot \cosh(\phi) - \sinh(\phi)} \right) \left[\frac{\phi \cdot \cosh(\phi)}{R^2} - \frac{\sinh(\phi)}{R^2} \right] \\ &= \left(\frac{C_{Ab} \cdot R \cdot Bi_m}{Bi_m + \frac{\phi}{\tanh(\phi)} - 1} \right) \left[\frac{\phi}{R^2} \cdot \frac{1}{\tanh(\phi)} - \frac{1}{R^2} \right]\end{aligned}$$

Finally, we can derive the effectiveness factor in the case of external mass transfer limitations as

$$\begin{aligned}\eta &= \frac{4\pi R^2 D_{\text{eff},A} \cdot \left. \frac{dC_A}{dr} \right|_{r=R}}{\frac{4}{3}\pi R^3 \cdot k_1 C_{Ab}} = \frac{3}{R^2} \cdot \frac{D_{\text{eff},A}}{k_1} \left(\frac{Bi_m}{Bi_m + \frac{\phi}{\tanh(\phi)} - 1} \right) \left(\frac{\phi}{\tanh(\phi)} - 1 \right) \quad (2.42) \\ &= \frac{3}{\phi} \cdot \left[\frac{\frac{1}{\tanh(\phi)} - \frac{1}{\phi}}{1 + \frac{\phi}{Bi_m} \cdot \left(\frac{1}{\tanh(\phi)} - \frac{1}{\phi} \right)} \right]\end{aligned}$$

2.3 COMPUTER PROGRAMS AND SIMULATION RESULTS

Notebook *FirstOrder_Isothermal_Concentration.ipynb*

The IPython notebook *FirstOrder_Isothermal_Concentration.ipynb* is a Python code to calculate the dimensionless reactant concentration profiles in the radial direction of isothermal spherical pellet for the first-order reaction. The reactant concentration is fixed at the external surface of the catalyst pellet.

◇ Import packages.

We will use the Python library *NumPy* for vector manipulations and calculation of *sinh* function. To plot the results of numerical simulations, we will use the *pyplot* module from *matplotlib* library. The *IPython.html* module is called to display widgets for interactive input of Thiele moduli.

```
%matplotlib inline
import numpy as np
from matplotlib.pyplot import *
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, LaTeX
```

◇ Define the main function *fmain*:

```
#
# Main function
#
def fmain ( $\phi_1$ ,  $\phi_2$ ,  $\phi_3$ , **kwargs):
#
#  $\phi_1$ ,  $\phi_2$ ,  $\phi_3$  are Thiele moduli
#
# calculate and plot dimensionless reactant concentration
# profiles in the radial direction of a spherical pellet
# for various Thiele moduli.
```

o Specify the array of radial points and allocate the array of reactant concentrations.

```
#
# specify the array of equally distributed radial positions
#  $\rho = \text{np.linspace}(1.e-5, 1, 501)$ 
# allocate array of concentrations
#  $c = \text{np.zeros}(\text{len}(\rho))$ 
```

- o Calculate and plot the reactant concentration profiles for various values of Thiele modulus.

```
# calculate and plot profile for  $\phi_1$ 
c = (1/ $\rho$ )*np.sinh( $\phi_1$ * $\rho$ )/np.sinh( $\phi_1$ )
plot( $\rho$ , c, '-', linewidth=1.5, color='b')
hold('on')
# calculate and plot profile for  $\phi_2$ 
c = (1/ $\rho$ )*np.sinh( $\phi_2$ * $\rho$ )/np.sinh( $\phi_2$ )
plot( $\rho$ , c, '-', linewidth=1.5, color='r')
# calculate and plot profile for  $\phi_3$ 
c = (1/ $\rho$ )*np.sinh( $\phi_3$ * $\rho$ )/np.sinh( $\phi_3$ )
plot( $\rho$ , c, '-', linewidth=1.5, color='k')
# specify plot axes, title and legend
matplotlib.rcParams.update({'font.size': 12})
xlabel(r'Dimensionless radius,  $\rho$  [-]')
ylabel('Dimensionless concentration,  $c$  [-]')
title1 = 'First-order reaction in isothermal spherical pellet \n'
title(title1)
legend([('$\phi_1 = '+str( $\phi_1$ )+'$'),('$\phi_2 = '+str( $\phi_2$ )+'$'),
        ('$ \phi_3 = '+str( $\phi_3$ )+'$')],loc='lower right')
axis([0.,1.,0.0,1.0])
show()
return
```

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on

◇ Define widgets.

```
#
# specify widgets
h1=HTML(value="<h4> <b> Diffusion and First-
Order Reaction in Isothermal Spherical Pellet</b>",color="brown")
h2=Latex(value="$$\mbox{Specify three values of Thiele modulus } \phi \mbox{ to compare
concentration profiles:}$$",)
h3=HTML(value = "<br> ")
display(h1)
display(h3)
display(h2)
# Use slider to input values of Thiele moduli
phi1_slider = FloatSlider(min=0.5, max=10, step=0.5, value=1)
phi1_slider.description = "$$\phi_1$$"
phi2_slider = FloatSlider(min=0.5, max=10, step=0.5, value=2)
phi2_slider.description = "$$\phi_2$$"
phi3_slider = FloatSlider(min=0.5, max=10, step=0.5, value=5)
phi3_slider.description = "$$\phi_3$$"
w = interact(fmain, phi1 = phi1_slider, phi2 = phi2_slider, phi3 = phi3_slider)
```

The screenshot in Fig. 2.1 shows the widgets to specify Thiele moduli and the plot obtained by simulating the concentration profiles in the pellet.

Diffusion and First-Order Reaction in Isothermal Spherical Pellet

Specify three values of Thiele modulus ϕ to compare concentration profiles:

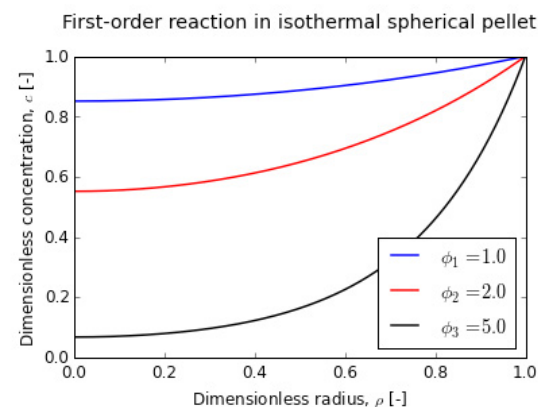


Figure 2.1: The screenshot of widgets to specify Thiele moduli for the first-order reaction in isothermal pellet.

Notebook *FirstOrder_Isothermal_Effectiveness.ipynb*

This notebook is used to calculate the effectiveness factor for the first-order reaction in the isothermal spherical pellet assuming a fixed reactant concentration at the external pellet surface.

◇ Import packages.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
```

◇ Define the main function *fmain*:

```
#
# Main program
#
def fmain(h, **kwargs):
#
# calculate the effectiveness factor for first-order reaction
# in isothermal spherical pellet
#
# parameter h is used to make a blank line in a widget
```

o Specify the array of Thiele moduli and allocate the array of effectiveness factors.

```
#
# specify the array of Thiele modulus
phi = np.linspace(0.01, 100., 1001)
# allocate array of effectiveness factor
eta = np.zeros(len(phi))
```

o Calculate and plot the effectiveness factor as a function of Thiele modulus.

```
#
# calculate and plot effectiveness factor
eta = (3./phi)*(1./np.tanh(phi) - 1./phi)
plt.rcParams.update({'font.size': 12})
plt.loglog(phi, eta, 'k-', basex=10, linewidth=1.5, color='b')
# specify plot axes, title and legend
plt.xlabel('Thiele modulus, $\phi$ [-]')
plt.ylabel('Effectiveness factor, $\eta$ [-]')
plt.axis([0.01, 100., 0.0, 1.2])
plt.title('First-order reaction in a spherical pellet \n')
plt.grid(True, which="both", ls="-")
plt.grid(True, 'major', linewidth=1.0)
plt.grid(True, 'minor', linewidth=0.5)
plt.show()
return
```

◇ Define a widget.

```
#  
# specify widgets  
h1=HTML(value="<h4> <b> Diffusion and First-  
Order Reaction in Isothermal Spherical Pellet</b>",color="brown")  
h2=HTML(value="<br>")  
display(h1)  
w = interact(fmain, h = h2)
```

The screenshot of the widget and resulting plot is shown in Fig. 2.2.

© 2013 Accenture. All rights reserved.

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be.

Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.

× **Diffusion and First-Order Reaction in Isothermal Spherical Pellet**

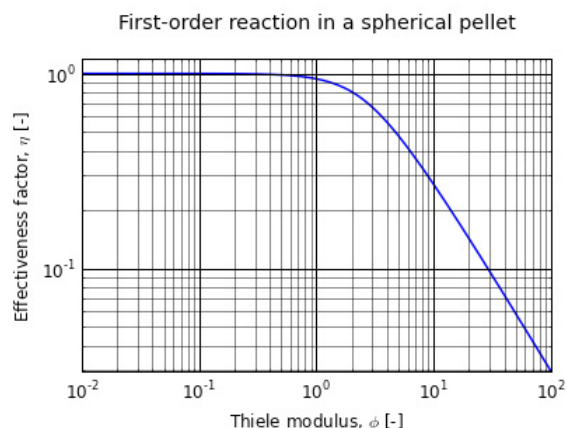


Figure 2.2: The widget to plot the effectiveness factor for the first-order reaction in isothermal pellet.

Notebook *FirstOrder_Isothermal_ExternalMassTransfer_Concentration.ipynb*

This notebook is utilized to calculate the reactant concentration profiles in the radial direction of isothermal spherical pellet for the first-order reaction taking into account the mass transfer resistance at the external pellet surface.

◇ Import packages.

```
%matplotlib inline
import numpy as np
from matplotlib.pyplot import *
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
```

◇ Define the main function *fmain*:

```
#
# Main function
#
def fmain (φ, h, Bi1, Bi2, Bi3, **kwargs):
#
# φ is the Thiele modulus, Bi1, Bi2 and Bi3 are Biot numbers
#
# calculate and plot dimensionless reactant concentration
# profiles in the radial direction of a spherical pellet
# for various Biot numbers
```

- o Specify the array of radial points and allocate the array of reactant concentrations.

```
#
# specify the array of equally distributed radial positions
ρ = np.linspace(1.e-5, 1, 501)
# allocate array of concentrations
c = np.zeros(len(ρ))
```

- o Calculate and plot the reactant concentration profiles for various values of Biot number.

```
# calculate and plot profile for Bi1
c = (1./ρ)*np.sinh(φ*ρ)/(np.sinh(φ) + (φ*np.cosh(φ)- np.sinh(φ))/Bi1)
matplotlib.rcParams.update({'font.size': 12})
plot(ρ, c, '-', linewidth=1.5, color='b')
hold('on')
# calculate and plot profile for Bi2
c = (1./ρ)*np.sinh(φ*ρ)/(np.sinh(φ) + (φ*np.cosh(φ)- np.sinh(φ))/Bi2)
plot(ρ, c, '-', linewidth=1.5, color='r')
# calculate and plot profile for Bi3
c = (1./ρ)*np.sinh(φ*ρ)/(np.sinh(φ) + (φ*np.cosh(φ)- np.sinh(φ))/Bi3)
plot(ρ, c, '-', linewidth=1.5, color='k')
# specify plot axes, title and legend
xlabel(r'Dimensionless radius, $ρ$ [-]')
ylabel('Dimensionless concentration, $c$ [-]')
title1 = 'First-
order reaction in isothermal spherical pellet\n with external mass transfer limitations, $
\phi$=%s'
title (title1 % (φ))
legend([(r'${Bi} = '+str(Bi1)+'$'),(r'${Bi} = '+str(Bi2)+'$'),
(r'${Bi} = '+str(Bi3)+'$')],loc='lower right')
axis([0.,1.,0.0,1.0])
show()
return
```

◇ Define widgets.

```
#
# specify widgets
h1=HTML(value="<h4><b> Diffusion and First-
Order Reaction in Isothermal Spherical Pellet <br> with External Mass Transfer Limitations
</b>",color="brown")
h2=HTML(value = "<br> ")
h3=Latex(value="$$\mbox{Specify Thiele modulus } \phi \mbox{ :}$$",)
h4=Latex(value="$$\mbox{Specify three values of Biot number } Bi \mbox{ to compare conc
entration profiles:}$$",)
display(h1)
display(h2)
display(h3)
# Use slider to input the value of Thiele modulus
phi_slider = FloatSlider(min=0.5, max=10, step=0.5, value=2)
phi_slider.description = "$$\phi$$"
# Use slider to input values of Biot number
Bi1_slider = FloatSlider(min=0.5, max=10, step=0.5, value=1)
Bi1_slider.description = "$$Bi_1$$"
Bi2_slider = FloatSlider(min=0.5, max=10, step=0.5, value=3)
Bi2_slider.description = "$$Bi_2$$"
Bi3_slider = FloatSlider(min=0.5, max=10, step=0.5, value=10)
Bi3_slider.description = "$$Bi_3$$"
w = interact(fmain, phi= phi_slider, h = h4, Bi1 = Bi1_slider, Bi2 = Bi2_slider, Bi3 = Bi3_s
lider)
```



"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download

The screenshot of widgets to specify the Thiele modulus and Biot numbers and the resulting plot is shown in Fig. 2.3.

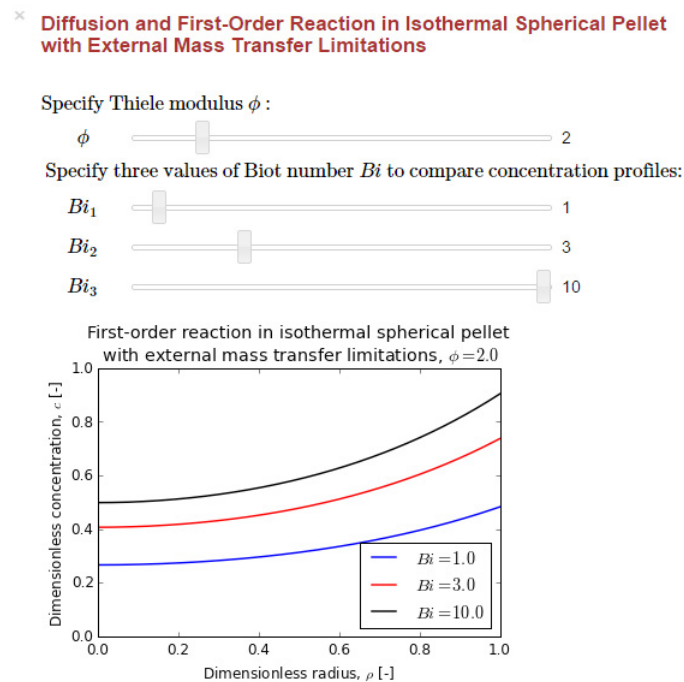


Figure 2.3: The widget to specify the values of Thiele modulus and Biot numbers for the first-order reaction with external mass transfer limitations in isothermal pellet.

Notebook *FirstOrder_Isothermal_ExternalMassTransfer_Effectiveness.ipynb*

This notebook is used to calculate the effectiveness factor for the first-order isothermal reaction with external mass transfer limitations in the spherical pellet.

◇ Import packages.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
```

◇ Define the main function *fmain*:

```
#
# Main program
#
def fmain(h, Bi1, Bi2, Bi3, **kwargs):
#
# calculate and plot relationships between the effectiveness factor
# and Thiele modulus for various values of Biot number
```

o Specify the array of Thiele moduli and allocate the array of effectiveness factors.

```
#
# specify the array of Thiele modulus
phi = np.linspace(0.01, 100., 1001)
# allocate array of effectiveness factor
eta = np.zeros(len(phi))
```

o Calculate and plot the effectiveness factor as a function of Thiele modulus.

```
#
# calculate and plot effectiveness factor for Bi1
a = 1./np.tanh(phi) - 1./phi
eta = (3./phi)*a/(1+(phi/Bi1)*a)
plt.rcParams.update({'font.size': 12})
plt.loglog(phi,eta,'-',basex=10,linewidth=1.5,color='b')
plt.hold('on')
# calculate and plot effectiveness factor for Bi2
a = 1./np.tanh(phi) - 1./phi
eta = (3./phi)*a/(1+(phi/Bi2)*a)
plt.loglog(phi,eta,'-',basex=10,linewidth=1.5,color='r')
# calculate and plot effectiveness factor for Bi3
a = 1./np.tanh(phi) - 1./phi
eta = (3./phi)*a/(1+(phi/Bi3)*a)
plt.loglog(phi,eta,'-',basex=10,linewidth=1.5,color='k')
# specify plot axes, title and legend
plt.xlabel('Thiele modulus, $\phi$ [-]')
plt.ylabel('Effectiveness factor, $\eta$ [-]')
plt.axis([0.01, 100., 0.0, 1.2])
plt.title('First-
order reaction in a spherical pellet with external mass transfer \n')
plt.grid(True,which="both",ls="-")
plt.grid(True,'major',linewidth=1.0)
plt.grid(True,'minor',linewidth=0.5)
legend = plt.legend([(r'${Bi}$ = '+str(Bi1)+'$'),(r'${Bi}$ = '+str(Bi2)+'$'),
(r'${Bi}$ = '+str(Bi3)+'$')],loc='lower left')
legend.get_frame().set_facecolor('white')
plt.show()
return
```


◇ Define a widget.

```
#
# specify widgets
h1=HTML(value="<h4><b> Diffusion and First-
Order Reaction in Isothermal Spherical Pellet <br> with External Mass Transfer Limitations
</b>",color="brown")
h2=HTML(value="<br>")
display(h1)
# Use slider to input values of Biot number
Bi1_slider = FloatSlider(min=0.5, max=10, step=0.5, value=1.)
Bi1_slider.description = "$Bi_1$:"
Bi2_slider = FloatSlider(min=0.5, max=10, step=0.5, value=3.)
Bi2_slider.description = "$Bi_2$:"
Bi3_slider = FloatSlider(min=0.5, max=10, step=0.5, value=10.)
Bi3_slider.description = "$Bi_3$:"
w = interact(fmain, h = h2, Bi1=Bi1_slider, Bi2=Bi2_slider, Bi3=Bi3_slider)
```

The screenshot of widgets to specify the Biot numbers is shown in Fig. 2.4.



DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

HERE.

www.fuqua.duke.edu/globalmba

Learn More >

× Diffusion and First-Order Reaction in Isothermal Spherical Pellet
with External Mass Transfer Limitations

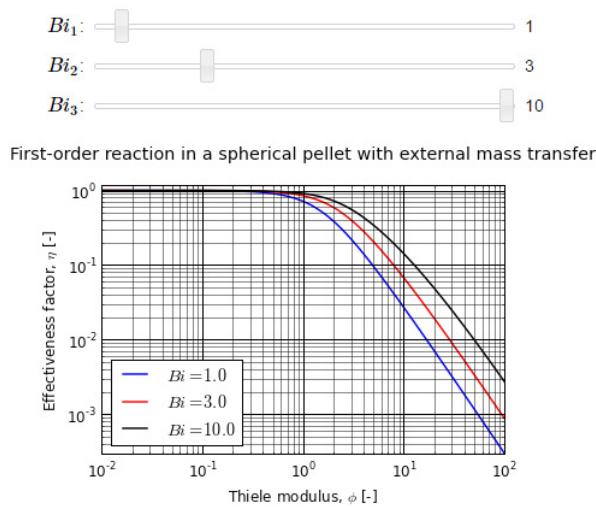


Figure 2.4: The widget to plot the effectiveness factor for the first-order reaction with external mass transfer limitations in isothermal pellet.

Figure 2.5 illustrates the effect of Thiele modulus on the reactant concentration distribution in the radial direction of the isothermal pellet for the first-order reaction. These distributions were calculated by using the *FirstOrder_Isothermal_Concentration_print.ipynb* notebook. At low values of Thiele modulus, the reactant is nearly uniformly distributed within the pellet since the reactant consumption rate by reaction is slower than the delivery rate by diffusion. At high values of Thiele modulus, the reactant is depleted close to the pellet outer surface due to the high reaction rate. As a result, the reactant concentration distribution within the pellet is highly non-uniform.

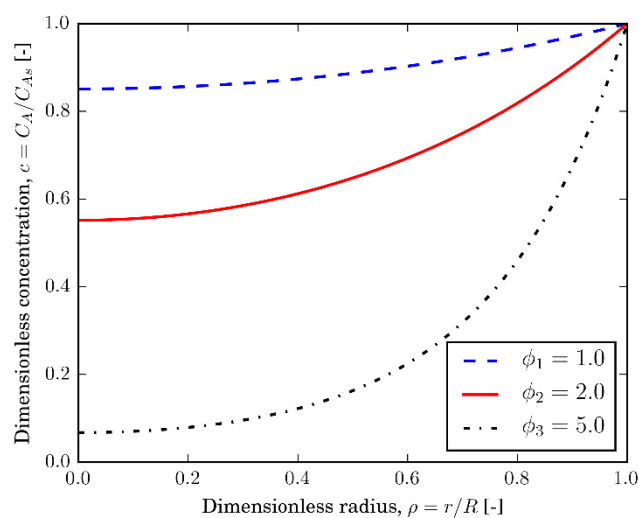


Figure 2.5: Concentration profiles for the first-order reaction in isothermal catalyst pellet.

The effect of Thiele modulus on effectiveness factor is illustrated in Fig. 2.6. This figure was calculated and plotted by using the *FirstOrder_Isothermal_Effectiveness_print.ipynb* notebook. The effectiveness factor approaches unity at low values of Thiele modulus ($\phi < 1$) indicating the complete utilization of internal catalyst volume. However, the effectiveness factor decreases rapidly at high values of Thiele modulus. This is an indication on non-effective utilization of catalyst volume as the reaction is taking place only near the outer pellet surface.

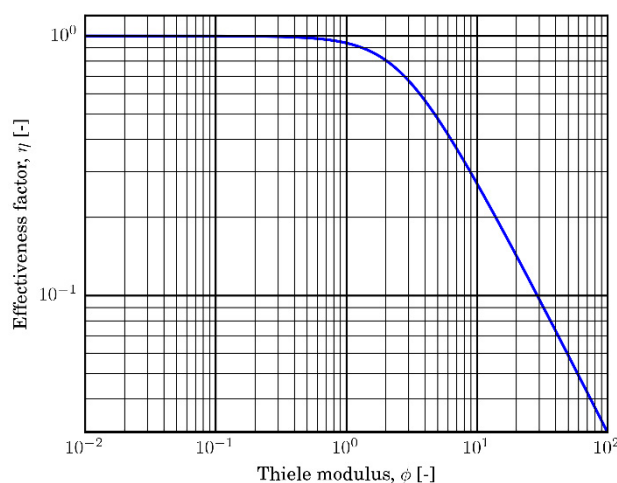


Figure 2.6: Effectiveness factor for the first-order reaction in isothermal catalyst pellet.

The effect of external mass transfer resistance on the reactant concentration distribution in the pellet is shown in Fig. 2.7. We used the *FirstOrder_Isothermal_ExternalMassTransfer_Concentration_print.ipynb* notebook to calculate and plot this figure. At high values of Biot number, the rate of mass transfer of reactant from the bulk phase to the external pellet surface is high, and the reactant concentration near the pellet surface is close to the bulk concentration. At low values of Biot number, the surface concentration is significantly lower than the bulk one. For example, the dimensionless reactant concentration is equal to 0.44 at $\rho = 1$ for $Bi_m = 1.0$ and $\phi = 2$. The reactant concentration profile is more uniform for low Biot number than that for high Bi_m due to the higher diffusion rate in the pellet.

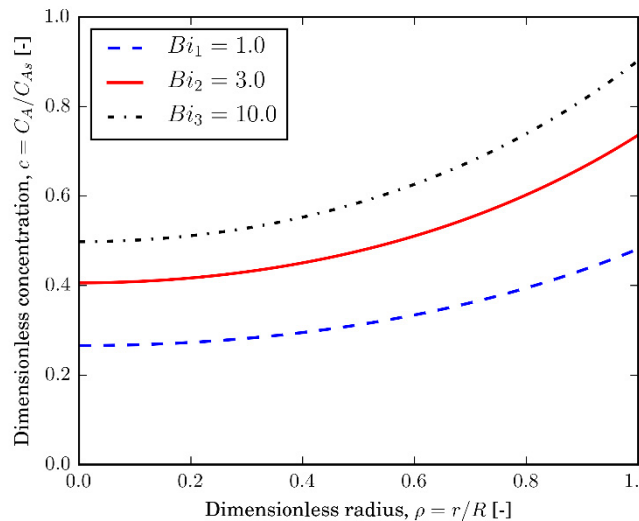


Figure 2.7: Concentration profiles for the first-order reaction in isothermal catalyst pellet with external mass-transfer limitations.

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](#).



The relationships between the effectiveness factor and Thiele modulus are shown in Fig. 2.8 for various values of Biot number. This figure was calculated and plotted utilizing the *FirstOrder_Isothermal_ExternalMassTransfer_Effectiveness_print.ipynb* notebook. The limiting value of Thiele modulus corresponding to $\eta \approx 1$ shifts to the low values of Thiele modulus with decreasing the Biot number to result in the narrow range of operation with full utilization of catalyst volume.

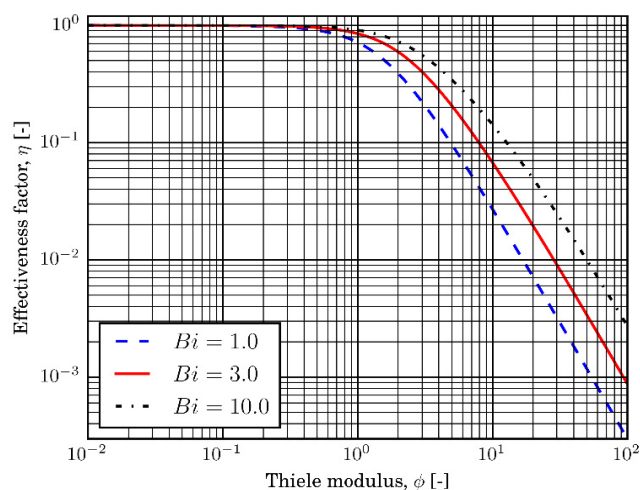


Figure 2.8: Effectiveness factor for the first-order reaction in isothermal catalyst pellet with external mass-transfer limitations.

3 SECOND-ORDER REACTION IN ISOTHERMAL CATALYST PELLET

In this chapter, you will study:

1. Setting up the mass balance equation that accounts for the diffusion and second-order catalytic reaction in the isothermal spherical pellet.
2. Solving numerically the model equation using the orthogonal collocation method.
3. Simulating the reactant concentration profiles in the pellet and the effectiveness factor for various values of Thiele modulus using the developed IPython notebooks.

3.1 MASS BALANCE EQUATION

We can write the dimensionless mass balance equation for the second-order chemical reaction in the isothermal spherical catalyst pellet similar to Eq. (2.15) as

$$\frac{d^2c}{d\rho^2} + \frac{2}{\rho} \cdot \frac{dc}{d\rho} - \phi^2 c^2 = 0 \quad (3.1)$$

Here, the Thiele modulus is defined as

$$\phi^2 = \frac{k \cdot R^2 \cdot C_{A,b}}{D_{\text{eff},A}}$$

The boundary conditions are

- At the pellet center $\rho=0$:

$$\frac{dc}{d\rho} = 0 \quad (3.2)$$

- At the outer pellet surface $\rho=1$:
 - o without external mass transfer limitations

$$c = 1 \quad (3.3)$$

- o with external mass transfer limitations

$$1 - c = \frac{1}{Bi_m} \cdot \frac{dc}{d\rho} \bigg|_{\rho=1} \quad (3.4)$$

3.2 NUMERICAL SOLUTION OF MODEL EQUATION USING ORTHOGONAL COLLOCATION METHOD

As the problem is symmetrical at $\rho = 0$, we introduce the following transformation

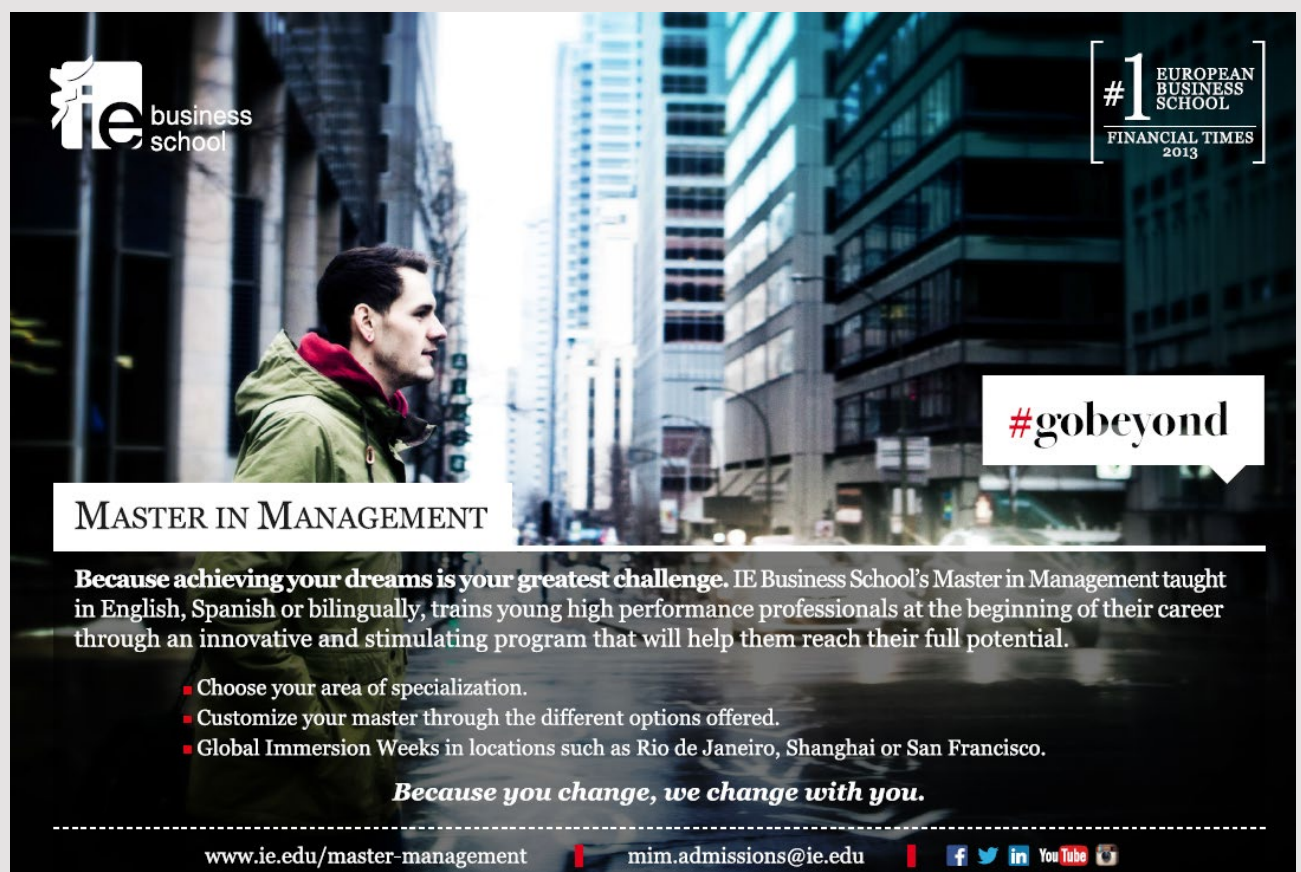
$$x = \rho^2 \quad (3.5)$$

Using this transformation, we can write the first derivative of concentration with respect to ρ in terms of the derivative with respect to x as

$$\frac{dc}{d\rho} = \frac{dc}{dx} \cdot \frac{dx}{d\rho} = \frac{dc}{dx} \cdot 2\rho = 2\sqrt{x} \frac{dc}{dx}$$

and the second derivative of concentration with respect to ρ as

$$\begin{aligned} \frac{d^2c}{d\rho^2} &= \frac{d}{d\rho} \left(2\sqrt{x} \frac{dc}{dx} \right) = \frac{d}{dx} \left(2\sqrt{x} \frac{dc}{dx} \right) \cdot \frac{dx}{d\rho} = \left[2 \cdot \frac{dc}{dx} \cdot \frac{1}{2} \cdot x^{-\frac{1}{2}} + 2\sqrt{x} \cdot \frac{d^2c}{dx^2} \right] \cdot \frac{dx}{d\rho} \\ &= \left[\frac{1}{\rho} \cdot \frac{dc}{dx} + 2\rho \cdot \frac{d^2c}{dx^2} \right] \cdot 2\rho = 4\rho^2 \frac{d^2c}{dx^2} + 2 \frac{dc}{dx} = 4x \frac{d^2c}{dx^2} + 2 \frac{dc}{dx} \end{aligned}$$



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu

f t in YouTube

Substituting the first and second derivatives into Eq. (3.1) results in

$$\left(4x \frac{d^2c}{dx^2} + 2 \frac{dc}{dx}\right) + \frac{2}{\sqrt{x}} \cdot \left(2\sqrt{x} \frac{dc}{dx}\right) - \phi^2 c^2 = 0$$

Simplifying the above equation, we have:

$$4x \frac{d^2c}{dx^2} + 6 \frac{dc}{dx} - \phi^2 c^2 = 0 \quad (3.6)$$

We use the orthogonal collocation method (Villadsen & Michelsen 1978) for the numerical solution of nonlinear ordinary differential equation, Eq. (3.6), with boundary conditions Eqs. (3.2)–(3.4). We select $N+1$ interpolation points in such a way that the first N points are the interior collocation points and the $N+1$ point is the boundary point at $x_{N+1}=1$. The interior points are chosen as roots of the Jacobi collocation polynomial $J_N(\alpha, \beta)$ with $\alpha=1$ and $\beta=1/2$. The unknown variable c is expanded in a set of Jacobi polynomials in terms of even powers of ρ as (Finlayson 1980):

$$c(\rho) = c(1) + (1-\rho^2) \sum_{j=1}^N a_j \cdot P_{j-1}(\rho^2) \quad (3.7)$$

where a_j are the unknown coefficients and $P_{j-1}(\rho^2)$ are the orthogonal polynomials. The function defined by Eq. (3.7) satisfy the boundary condition by Eq. (3.2). The spatial derivatives in Eq. (3.6) at the internal collocation points, ρ_j , can be expressed as functions of the concentration:

$$\left. \frac{\partial c}{\partial \rho} \right|_{\rho_j} = \sum_{k=1}^{N+1} A_{j,k} \cdot c_k = \mathbf{A} \cdot \mathbf{c}$$

$$\left. \frac{\partial^2 c}{\partial \rho^2} \right|_{\rho_j} = \sum_{k=1}^{N+1} B_{j,k} \cdot c_k = \mathbf{B} \cdot \mathbf{c} ,$$

where c_k is the approximate solution at the collocation point ρ_k . The matrices \mathbf{A} and \mathbf{B} are calculated using the method based on Lagrange's interpolation formula (Rice & Do 2012). The python program for calculation of matrices \mathbf{A} and \mathbf{B} is given in Appendix A3.

Thus, we can write the collocation equations at the interior points, x_i , $i=1, \dots, N$, as

$$4x_i \sum_{j=1}^{N+1} B_{i,j} c_j + 6 \sum_{j=1}^{N+1} A_{i,j} c_j - \phi^2 c_i^2 = 0 \quad (3.8)$$

Moving the $N+1$ term outside the summation sign, we rewrite Eq. (3.8) as

$$4x_i \left[\sum_{j=1}^N B_{i,j} c_j + B_{i,N+1} c_{N+1} \right] + 6 \left[\sum_{j=1}^N A_{i,j} c_j + A_{i,N+1} c_{N+1} \right] - \phi^2 c_i^2 = 0$$

Simplifying the above equation, we get:

$$\sum_{j=1}^N (4x_i B_{i,j} + 6A_{i,j}) c_j + (4x_i B_{i,N+1} + 6A_{i,N+1}) c_{N+1} - \phi^2 c_i^2 = 0 \quad (3.9)$$

The boundary condition at $\rho=0$ ($x=0$) is satisfied by using the transformation by Eq. (3.5). We then use the boundary condition at $\rho=1$ ($x=1$) to eliminate c_{N+1} from Eq. (3.9):

- o Using the boundary condition by Eq. (3.3), $c_{N+1}=1$ at $x_{N+1}=1$, we simplify Eq. (3.9) as

$$\sum_{j=1}^N (4x_i B_{i,j} + 6A_{i,j}) c_j + (4x_i B_{i,N+1} + 6A_{i,N+1}) - \phi^2 c_i^2 = 0 \quad (3.10)$$

- o In the case of external mass transfer limitations, the boundary condition by Eq. (3.4) at $x_{N+1}=1$ can be written in terms of the variable x as

$$1-c = \frac{1}{Bi_m} \cdot \frac{dc}{d\rho} \bigg|_{\rho=1} = \frac{1}{Bi_m} \cdot \left(2\sqrt{x} \frac{dc}{dx} \right) \bigg|_{x=1} = \frac{2}{Bi_m} \frac{dc}{dx} \bigg|_{x=1} \quad (3.11)$$

The first derivative at the point x_{N+1} is

$$\frac{dc}{dx} \bigg|_{x=x_{N+1}} = \sum_{j=1}^{N+1} A_{N+1,j} c_j$$

Substituting the derivative into Eq. (3.11), we get:

$$\sum_{j=1}^N A_{N+1,j} c_j + A_{N+1,N+1} c_{N+1} = \frac{Bi_m}{2} \cdot (1-c_{N+1}) \quad (3.12)$$

Solving the above equation for c_{N+1} results in

$$c_{N+1} = \frac{1 - \frac{2}{Bi_m} \sum_{j=1}^N A_{N+1,j} c_j}{1 + \frac{2A_{N+1,N+1}}{Bi_m}} = \frac{Bi_m - 2 \sum_{j=1}^N A_{N+1,j} c_j}{Bi_m + 2A_{N+1,N+1}}$$

Substituting c_{N+1} into Eq. (3.9) yields

$$\sum_{j=1}^N (4x_i B_{i,j} + 6A_{i,j}) c_j + (4x_i B_{i,N+1} + 6A_{i,N+1}) \cdot \frac{Bi_m - 2 \sum_{j=1}^N A_{N+1,j} c_j}{Bi_m + 2A_{N+1,N+1}} - \phi^2 c_i^2 = 0 \quad (3.13)$$

Rearranging the above equation, we get:

$$\sum_{j=1}^N \left(4x_i B_{i,j} + 6A_{i,j} - (4x_i B_{i,N+1} + 6A_{i,N+1}) \cdot \frac{2A_{N+1,j}}{B_{i_m} + 2A_{N+1,N+1}} \right) c_j + (4x_i B_{i,N+1} + 6A_{i,N+1}) \cdot \frac{B_{i_m}}{B_{i_m} + 2A_{N+1,N+1}} - \phi^2 c_i^2 = 0 \quad (3.14)$$

Finally, we obtain a system of nonlinear algebraic equations:

$$F_i = \sum_{j=1}^N C_{i,j} \cdot c_j + d_i + g_i = 0, \quad i=1,\dots,N \quad (3.15)$$

where elements of the matrix **C** and vector **d** depend on the boundary conditions at the pellet surface:

o without mass transfer limitations

$$\begin{aligned} C_{i,j} &= 4x_i B_{i,j} + 6A_{i,j} \\ d_i &= 4x_i B_{i,N+1} + 6A_{i,N+1} \\ g_i &= -\phi^2 c_i^2 \end{aligned} \quad (3.16)$$

SMS from your computer

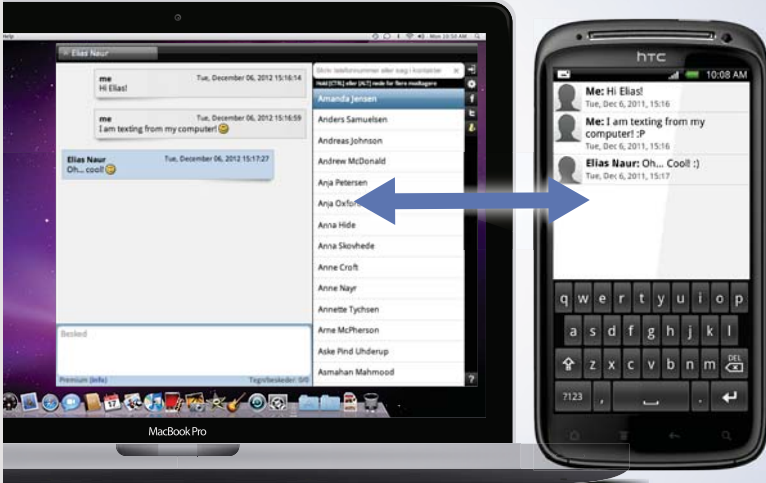
...Sync'd with your Android phone & number


FREE
30 days trial!

Go to

BrowserTexting.com

and start texting from
your computer!





BrowserTexting

o with mass transfer limitations

$$\begin{aligned}C_{i,j} &= (4x_i B_{i,j} + 6A_{i,j}) - (4x_i B_{i,N+1} + 6A_{i,N+1}) \cdot \frac{2A_{N+1,j}}{Bi_m + 2A_{N+1,N+1}} \\d_i &= \frac{(4x_i B_{i,N+1} + 6A_{i,N+1}) \cdot Bi_m}{Bi_m + 2A_{N+1,N+1}} \\g_i &= -\phi^2 c_i^2\end{aligned}\tag{3.17}$$

We use the Newton-Raphson method to solve numerically the system of nonlinear Eqs. (3.15). The vector of concentrations on the $k+1$ iteration, c^{k+1} , is calculated as

$$c^{k+1} = c^k - [J^k]^{-1} \cdot F^k,\tag{3.18}$$

where superscript k corresponds to the current k iteration and \mathbf{J} is the Jacobian matrix.

The elements of the Jacobian matrix are

$$J_{ij} = \frac{\partial F_i}{\partial c_j} = \begin{cases} C_{i,j} & \text{for } j \neq i \\ C_{i,j} - 2\phi_s^2 c_i & \text{for } j = i \end{cases}\tag{3.19}$$

Here, the non-diagonal elements of the Jacobian matrix are constant and only the diagonal entries depend on c_i , and thus change with iteration.

The effectiveness factor η is given by (Villadsen & Michelsen 1978, p214) as

$$\eta = \frac{3}{2} \int_0^1 c^2(x) \cdot \sqrt{x} dx\tag{3.20}$$

The effectiveness factor is calculated by the Radau quadrature.

3.3 COMPUTER PROGRAMS AND NUMERICAL RESULTS

Notebook *SecondOrder_Isothermal_Concentration.ipynb*

This notebook is used to calculate and plot the reactant concentration profile for the second-order reaction in the isothermal spherical pellet.

◇ Import packages.

We will use the Python library *NumPy* for vector manipulations and matrix formation. The *js_roots* module from the *scipy.special.orthogonal* library is used to calculate the roots of Jacobi polynomial. The *solve* and *norm* modules from the *SciPy* library are utilized to solve the dense system of linear equations and to find the vector norm. To plot the results of numerical simulations, we will use the *matplotlib* library. The *IPython.html* module is called to display widgets for interactive input of Thiele modulus. The *orthogonal_collocation* library is used to calculate the collocation matrices and interpolate the solution.

```
%matplotlib inline
import numpy as np
from scipy.special.orthogonal import js_roots
from scipy.linalg import solve
from scipy.linalg import norm
from matplotlib.pyplot import *
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
import orthogonal_collocation as oc
```

◇ Define the main function *fmain*:

- o Input parameters:
 - ϕ – Thiele modulus

```
#
# Main function
#
def fmain ( $\phi$ , h, **kwargs):
#
#  $\phi$  is a Thiele modulus
#
```

- o Specify the grid.

The number of internal collocation points is *ncolpt*. The total number of collocation points is equal to *ncolpt* + 1.

```
#-----
# Set the number of internal collocation points
#-----
ncolpt = 5
#
# Total number of points = No of internal collocation points +
#                          right boundary for x=1
ntpt = ncolpt + 1
```

- o Set the initial guess for solution of nonlinear equations.
We will use the uniform profile, i.e. $c = 1$, as the initial guess.

```
#
# Specify the initial guess for solution of nonlinear equations.
# We will use the uniform concentration profile,  $c = 1$ 
c0 = np.ones(ncolpt)
```

- o Calculate the position of internal collocation points as roots of Jacobi polynomial.

```
#
# Calculate internal collocation points (xj)
# The value of parameters alpha and beta of Jacobi polynomials:
# alpha = 1, beta = (s-1)/2
s = 2 # Spherical geometry
# Corresponding parameters p and q in the python function js_roots:
q = (s + 1.)/2.
p = 1. + q
# Assign internal collocation points, xj, to roots of Jacobi polynomial
xj = np.zeros(ncolpt)
wj = np.zeros(ncolpt)
[xj,wj] = js_roots(ncolpt,p,q)
# Prepare array of all collocation points = internal + right boundary
xr = np.zeros(ntpt)
xr[0:ncolpt] = xj[0:ncolpt]
xr[ntpt-1] = 1.0 # right boundary
```

The Wake

the only emission we want to leave behind

Low-speed Engines Medium-speed Engines Turbochargers Propellers Propulsion Packages PrimeServ

The design of eco-friendly marine power and propulsion solutions is crucial for MAN Diesel & Turbo. Power competencies are offered with the world's largest engine programme – having outputs spanning from 450 to 87,220 kW per engine. Get up front! Find out more at www.mandieselturbo.com

Engineering the Future – since 1758.

MAN Diesel & Turbo



- o Calculate vectors of the first, second and third derivatives of the polynomial at the collocation points.

```
#
# Calculate vectors of the first, second and third
# derivatives of the polynomial at the collocation points
[dif1,dif2,dif3] = oc.dif(ntpt, xr)
```

- o Calculate matrices of the first **Am** and second **Bm** derivative weights, matrix **Cm** and vector **d**.

```
#
# Allocate memory and calculate matrices of the first (Am) and second (Bm) derivative weights
Am = np.zeros((ntpt,ntpt))
Bm = np.zeros((ntpt,ntpt))
[Am,Bm] = oc.colmatrix (ntpt, xr, dif1, dif2, dif3)
#
# Allocate memory and calculate matrix Cm and vector d
Cm = np.zeros((ncolpt,ncolpt))
d = np.zeros(ncolpt)
d[0:ncolpt] = 4.0 * (xr[0:ncolpt] * Bm[0:ncolpt,ntpt-1] +
                    ((s+1.0)/2.) * Am[0:ncolpt,ntpt-1])
for i in range (ncolpt):
    Cm[i,0:ncolpt] = 4.0 * (xr[i] * Bm[i,0:ncolpt] +
                          ((s+1.0)/2.) * Am[i,0:ncolpt])
```

- o Solve the system of nonlinear equations using the Newton-Raphson method. Specify tolerances and the maximum number of iterations.

```
#
# Allocate memory
c = np.zeros(ncolpt)
#
# Solve the system of nonlinear equations using Newton-Raphson method
#
# Define tolerances and the maximum number of iterations
tolX = 1e-14
tolG = 1e-14
maxIter = 10
```

- o Allocate vectors and Jacobian matrix.

```
# Allocate vectors f and g
f = np.zeros(ncolpt)
g = np.zeros(ncolpt)
# Allocate the vector vect
vect = np.zeros(ncolpt)
# Allocate the Jacobian matrix
Jac = np.zeros((ncolpt,ncolpt))
# Make a copy of vector c
c = c0.copy()
```

- o Start Newton iterations.

```
#
# Start Newton iterations
    for i in range (maxIter):
#
# Set up a system of ncolpt nonlinear algebraic equations, f = 0
#-----
# reaction term
    g = - (phi**2)*c**2
# calculate f(i) at i-th internal collocation point
    for i in range (ncolpt):
        vect = Cm[i,:]*c[:]
        f[i] = vect.sum() + d[i] + g[i]
#
# Set up Jacobian
#
    Jac = Cm.copy()
# reaction term
    Jac[np.diag_indices_from(Jac)] -= 2.*(phi**2)*c
# Find new solution vector by solving the system of linear equations
    cNew = c - solve( Jac, f )
# Check algorithm convergence
    if (norm(cNew-c) < tolX):
        break
    if (norm(f) < tolG):
        break
    c = cNew.copy ()
#
# End Newton iterations
# Check if the number of iterations reaches maxIter
    if i >= maxIter-1:
        print ('Too many iterations: try to increase maxIter or decrease tolerances')
        sys.exit()
# else:
#
    print ('Successful solution: the number of iterations is %s%i )
```

- o Calculate the effectiveness factor.

```
#
# Calculate the effectiveness factor
    vector1 = np.zeros(ntpt)
    vector2 = np.zeros(ncolpt)
    [vector1] = oc.radau(ntpt, xr, dif1)
    vector2[0:ncolpt] = vector1[0:ncolpt]*c[0:ncolpt]**2
    eta = vector1[ntpt-1] + vector2.sum()
    print (" Effectiveness factor: eta = {0:<5.3f}".format(eta))
```

- o Find the solution vector by interpolation.

```
#  
# Find the solution at numdiv uniformly distributed interpolation points  
    numdiv = 51  
    x_int = np.linspace(0.0, 1.0, num=numdiv )  
    y_int = np.zeros(numdiv)  
#  
    y = np.ones(ntpt)  
    y[0:ncolpt] = c[0:ncolpt]  
#  
    xintp = np.zeros(ntpt)  
    for i in range(numdiv):  
        xxx = x_int[i]*x_int[i]  
        [xintp] = oc.intrp(ntpt, xxx, xr, dif1)  
        vector1 = xintp*y  
        y_int[i] = vector1.sum()
```

TURN TO THE EXPERTS FOR **SUBSCRIPTION** CONSULTANCY

Subscribe is one of the leading companies in Europe when it comes to innovation and business development within subscription businesses.

We innovate new subscription business models or improve existing ones. We do business reviews of existing subscription businesses and we develop acquisition and retention strategies.

Learn more at [linkedin.com/company/subscribe](https://www.linkedin.com/company/subscribe) or contact
Managing Director Morten Suhr Hansen at mha@subscribe.dk

SUBSCR✓**BE** - to the future

- o Finally, we plot the concentration profile.

```
#
# plot results
plot(x_int, y_int, '-', linewidth=1.5, color='b')
matplotlib.rcParams.update({'font.size': 12})
xlabel('Dimensionless radial distance,  $\rho$  [-]')
ylabel('Dimensionless concentration, c [-]')
title1 = 'Second-order reaction in isothermal spherical catalyst: \n'
title1 += ' Thiele modulus = %s '
title(title1 % ( $\phi$ ))
axis([0.,1.,0.0,1.0])
show()
return
```

- ◇ Define the widget.

```
#
# Specify widgets
h1=HTML(value="<h4><b> Diffusion and Second-  
Order Reaction in Isothermal Spherical Pellet</b>",color="brown")
h2=HTML(value = "<br> ")
h3=Latex(value="$$\mbox{Specify Thiele modulus } \phi \mbox{ :}$$",)
display(h1)
display(h2)
display(h3)
# Use slider to input value of Thiele modulus
phi_slider = FloatSlider(min=0.5, max=10, step=0.5, value=5)
phi_slider.description = "$$\phi$$"
w = interact(fmain,  $\phi$  = phi_slider, h=h2)
```

The screenshot illustrating the widget to specify the value of Thiele modulus and simulation results is shown in Fig. 3.1.

Diffusion and Second-Order Reaction in Isothermal Spherical Pellet

Specify Thiele modulus ϕ :



Effectiveness factor: $\eta = 0.397$

Second-order reaction in isothermal spherical catalyst:
Thiele modulus = 5.0

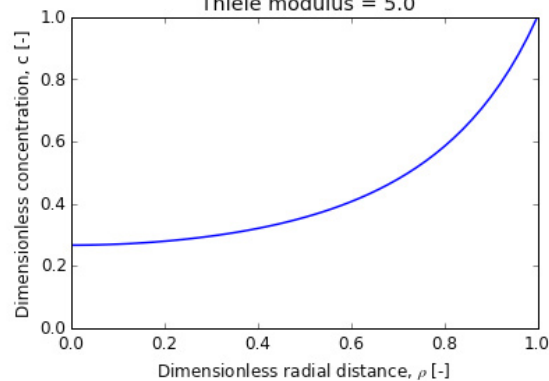


Figure 3.1: The widget to specify the value of Thiele modulus for the second-order reaction in isothermal pellet.

Notebook *SecondOrder_Isothermal_Effectiveness.ipynb*

This notebook is used to calculate and plot the effectiveness factor for the second-order reaction in the isothermal spherical pellet.

◇ Import packages.

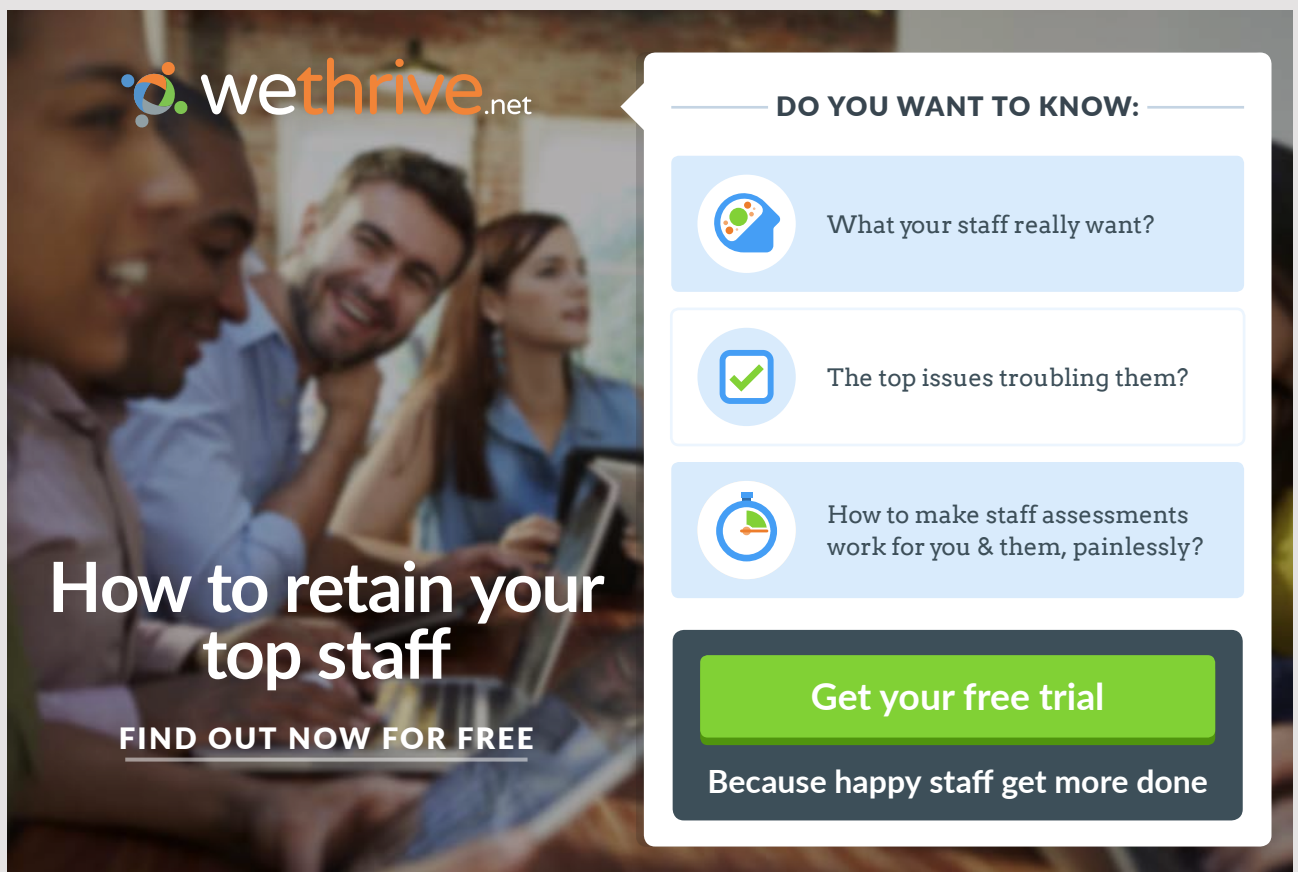
```
%matplotlib inline
import scipy as sp
import numpy as np
from scipy.special.orthogonal import js_roots
#
from scipy.linalg import solve
from scipy.linalg import norm
import matplotlib.pyplot as plt
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import HTML
#
import orthogonal_collocation as oc
import sys
```

◇ Define the main function *fmain*:

```
#  
# Main function  
#  
def fmain (φ):  
#  
# φ is a Thiele modulus
```

o Specify the grid.

```
#  
# -----  
# specify the number of internal collocation points  
# ncolpt = 7  
#  
s = 2          # Spherical geometry
```






wethrive.net

How to retain your top staff

FIND OUT NOW FOR FREE

DO YOU WANT TO KNOW:

-  What your staff really want?
-  The top issues troubling them?
-  How to make staff assessments work for you & them, painlessly?

Get your free trial

Because happy staff get more done

- o Set the initial guess for solution of nonlinear equations.

```
#
# set initial guess:
# Here we use a uniform concentration profile at inlet value
c0 = np.ones(ncolpt)
```

- o Calculate the position of internal collocation points as roots of Jacobi polynomial.

```
#-----
# total number of points = No of internal colloc. points +
#                               right boundary for x=1
ntpt = ncolpt + 1
#
# calculate internal collocation points (xj)
# the value of parameters alpha and beta in description of Jacobi polynomials:
#   alpha = 1, beta = (s-1)/2
# corresponding parameters p and q in the python function js_roots:
#   q = (s + 1.)/2.
#   p = 1. + q
# internal collocation points, xj, are the roots of Jacobi polynomial
xj = np.zeros(ncolpt)
wj = np.zeros(ncolpt)
[xj,wj] = js_roots(ncolpt,p,q)
# all collocation points = internal + right boundary
xr = np.zeros(ntpt)
xr[0:ncolpt] = xj[0:ncolpt]
xr[ntpt-1] = 1.0 # right boundary
```

- o Calculate vectors of the first, second and third derivatives of the polynomial at the collocation points. Calculate matrices of the first **Am** and second **Bm** derivative weights, matrix **Cm** and vector **d**.

```
#
# calculate vectors of the first (dif1), second (dif2) and third (dif3)
# derivatives of the node polynomial at the zeros
[dif1,dif2,dif3] = oc.dif(ntpt, xr)
#
# set up matrices of the first (Am) and second (Bm) derivative weights
Am = np.zeros((ntpt,ntpt))
Bm = np.zeros((ntpt,ntpt))
[Am,Bm] = oc.colmatrix (ntpt, xr, dif1, dif2, dif3)
#
# set up matrix Cm and vector d
Cm = np.zeros((ncolpt,ncolpt))
d = np.zeros(ncolpt)
d[0:ncolpt] = 4.0 * (xr[0:ncolpt] * Bm[0:ncolpt,ntpt-1] +
                    ((s+1.0)/2.) * Am[0:ncolpt,ntpt-1])
for i in range (ncolpt):
    Cm[i,0:ncolpt] = 4.0 * (xr[i] * Bm[i,0:ncolpt] +
                           ((s+1.0)/2.) * Am[i,0:ncolpt])
```

- o Solve the system of nonlinear equations using the Newton-Raphson method.


```
#
# allocate memory
c = np.zeros(ncolpt)
#
# solve the system of nonlinear equations using Newton-Raphson method
#
# define tolerances and the maximum number of iterations
tolX = 1e-14
tolG = 1e-14
maxIter = 10
#
f = np.zeros(ncolpt)
g = np.zeros(ncolpt)
vect = np.zeros(ncolpt)
Jac = np.zeros((ncolpt,ncolpt))
#
c = c0.copy()
#
# start Newton iterations
for i in range (maxIter):
#
# set up a system of ncolpt nonlinear algebraic equations, f = 0
#-----
# reaction term
#-----
g = - (phi**2)*c**2
#-----
# for each internal collocation point
for i in range (ncolpt):
    vect = Cm[i,:]*c[:]
    f[i] = vect.sum() + d[i] + g[i]
#
# set up Jacobian
#
Jac = Cm.copy()
#-----
# reaction term
#-----
Jac[np.diag_indices_from(Jac)] -= 2.*(phi**2)*c
#-----
# find the solution vector by solving system of linear equations
cNew = c - solve( Jac, f )
# check algorithm convergence
if (norm(cNew-c) < tolX):
    break
if (norm(f) < tolG):
    break
c = cNew.copy ()
#
# end Newton iterations
# check if the number of iterations reaches maxIter
if i >= maxIter-1:
    print ('Too many iterations: try to increase maxIter or decrease tolerances')
    sys.exit()
```

- o Calculate the effectiveness factor.

```
#  
# calculate the effectiveness factor  
vector1 = np.zeros(ntpt)  
vector2 = np.zeros(ncolpt)  
[vector1] = oc.radau(ntpt, xr, dif1)  
vector2[0:ncolpt] = vector1[0:ncolpt]*c[0:ncolpt]**2  
η = vector1[ntpt-1] + vector2.sum()  
#  
return [η, ncolpt]
```

- ◇ Define the widget.


```
#  
# display the title  
h1=HTML(value="<h4><b> Diffusion and Second-  
Order Reaction in Isothermal Spherical Pellet</b>",color="brown")  
h2=HTML(value="<br>")  
display(h1)  
display(h2)
```




Struggling to get interviews?

Professional CV consulting & writing assistance from leading job experts in the UK.

[Visit site](#)

 Take a short-cut to your next job!
Improve your interview success rate by 70%.

 **TheCVagency**
Visit thecvagency.co.uk for more info.

◇ Plot the effectiveness factor as a function of Thiele modulus.

```
#
# specify the array of Thiele moduli uniformly distributed in log space
# specify the number of divisions of this space
num_eta = 101
phi = np.logspace(-2, 2, num_eta, endpoint=True)
eta = np.zeros(num_eta)
# calculate the effectiveness factor
for i in range (num_eta):
    eta[i], ncolpt = fmain (phi[i])
#
# prepare log-log plot
plt.rcParams.update({'font.size': 12})
plt.loglog(phi, eta, '-', basex=10, linewidth=1.5, color='b')
# specify plot axes, title and legend
plt.title('Second-order reaction in a spherical pellet')
plt.xlabel('Thiele modulus,  $\phi$  [-]')
plt.ylabel('Effectiveness factor,  $\eta$  [-]')
plt.axis([0.01, 100., 0.0, 1.2])
plt.grid(True, which="both", ls="-")
plt.grid(True, 'major', linewidth=0.7)
plt.grid(True, 'minor', linewidth=0.3)
plt.show()
```

The screenshot displaying the widget and simulation results is shown in Fig. 3.2.

× **Diffusion and Second-Order Reaction in Isothermal Spherical Pellet**

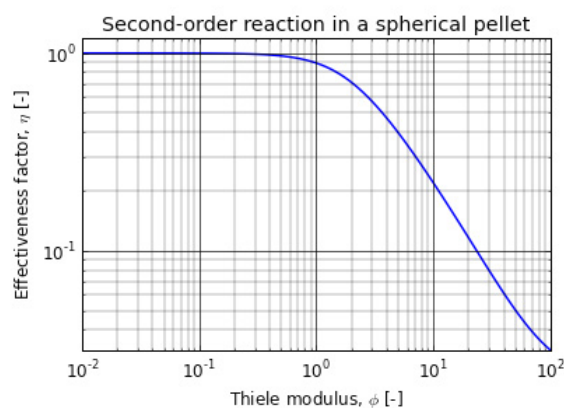


Figure 3.2: The widget to plot the effectiveness factor for the second-order reaction in isothermal pellet.

Figure 3.3 illustrates the reactant concentration profiles in the isothermal spherical pellet calculated for the second-order reaction using the *SecondOrder_Isothermal_Concentration_thiele.ipynb* notebook.

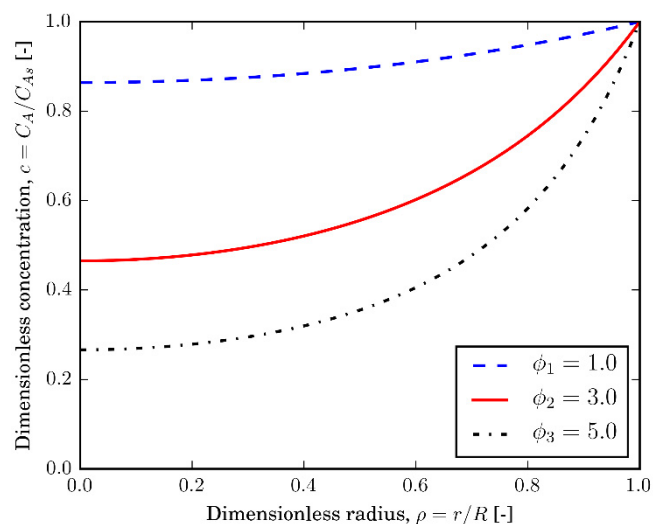


Figure 3.3: The reactant concentration distribution for the second-order reaction in isothermal pellet.

The reactant concentration is non-uniformly distributed in the radial direction of the pellet at large values of Thiele modulus due to increasing resistance to diffusion in the pellet. As a result, the effectiveness factor decreases significantly for $\phi > 1$, as shown in Fig. 3.4. This plot was prepared using the *SecondOrder_Isothermal_Effectiveness_print.ipynb* notebook.

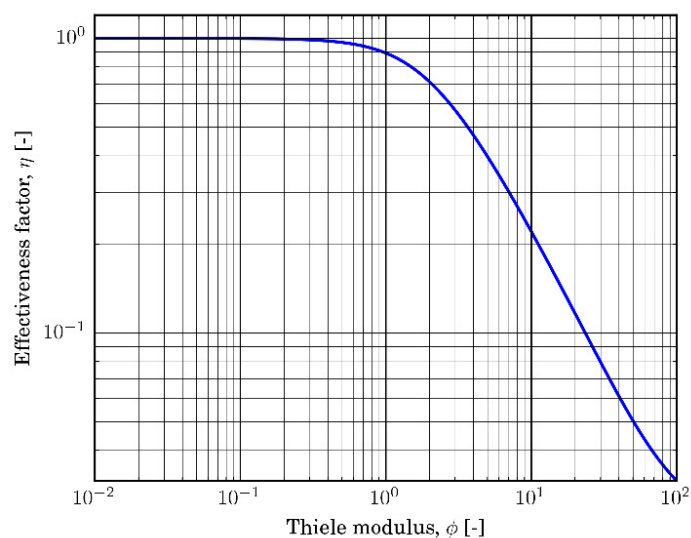


Figure 3.4: The effectiveness factor for the second-order reaction in isothermal pellet.

4 CHEMICAL REACTION IN NON-ISOTHERMAL CATALYST PELLET

In this chapter, you will learn to:

1. Derive the heat balance equation that accounts for the heat conduction and heat generation by chemical reaction in the non-isothermal spherical pellet.
2. Solve numerically the mass and heat balance equations using the finite-difference method.
3. Simulate and plot the reactant concentration and temperature profiles in the pellet for various values of process parameters using the elaborated IPython notebooks.

4.1 DERIVATION OF HEAT BALANCE EQUATION

Consider a first-order irreversible chemical reaction in non-isothermal spherical catalyst pellet.



gaiteye
Challenge the way we run

EXPERIENCE THE POWER OF
FULL ENGAGEMENT...

RUN FASTER.
RUN LONGER..
RUN EASIER...

READ MORE & PRE-ORDER TODAY
WWW.GAITEYE.COM

The advertisement features a background image of a person running on a path. Overlaid on the image are technical diagrams: a series of yellow dots forming a horizontal line, and a circular diagram with lines radiating from a point on a runner's foot, suggesting motion analysis or sensor placement.

We can derive a steady-state energy balance over a spherical shell of thickness Δr located at radius r within a spherical catalyst pellet as

$$\left(\begin{array}{c} \text{Rate of input} \\ \text{of energy} \\ \text{by conduction at } r \\ \text{(Joules/time)} \end{array} \right) - \left(\begin{array}{c} \text{Rate of output} \\ \text{of energy} \\ \text{by conduction at } r + \Delta r \\ \text{(Joules/time)} \end{array} \right) + \left(\begin{array}{c} \text{Rate of generation} \\ \text{of energy} \\ \text{by reaction within } \Delta r \\ \text{(Joules/time)} \end{array} \right) = 0$$

The heat flux by conduction in the catalyst pellet is defined as

$$q_r = -k_{\text{eff}} \frac{dT}{dr}, \quad (4.1)$$

where q_r is the heat flux and k_{eff} is the effective conductivity.

The energy released by the first-order reaction within the differential volume element, $4\pi r^2 \Delta r$, is

$$k(T) \cdot C_A \cdot 4\pi r^2 \Delta r \cdot (-\Delta H_{R_A}),$$

where $(-\Delta H_{R_A})$ is the heat of reaction per mole of A reacted.

Combining all terms, we formulate the energy balance as

$$(q_r \times 4\pi r^2) \Big|_r - (q_r \times 4\pi r^2) \Big|_{r+\Delta r} + k(T) \cdot C_A \cdot 4\pi r^2 \Delta r \cdot (-\Delta H_{R_A}) = 0 \quad (4.3)$$

Dividing by $4\pi \Delta r$, taking the limit as Δr goes to zero and using the definition of the first derivative gives

$$\frac{d(r^2 q_r)}{dr} - r^2 \cdot k(T) \cdot C_A \cdot (-\Delta H_{R_A}) = 0$$

Substituting the heat flux by Eq. (4.1), we get:

$$\frac{d}{dr} \left(r^2 \cdot k_{\text{eff}} \cdot \frac{dT}{dr} \right) + r^2 \cdot (-\Delta H_{R_A}) \cdot k(T) \cdot C_A = 0 \quad (4.4)$$

Assuming a constant effective conductivity, we rewrite Eq. (4.4) as

$$\left(\frac{d^2 T}{dr^2} + \frac{2}{r} \cdot \frac{dT}{dr} \right) + \frac{k(T) \cdot C_A \cdot (-\Delta H_{R_A})}{k_{\text{eff}}} = 0 \quad (4.5)$$

We use the Arrhenius equation to describe the temperature dependence of rate constant:

$$k(T) = k(T_b) \cdot \exp\left(-\frac{E}{R_g T_b} \left(\frac{T_b}{T} - 1\right)\right) = k(T_b) \cdot \exp\left(-\gamma \left(\frac{T_b}{T} - 1\right)\right), \quad (4.6)$$

where $\gamma = \frac{E}{R_g T_b}$ is the Arrhenius number, E is the intrinsic activation energy, R_g is the gas constant and T_b is the bulk temperature. The Arrhenius number reflects the sensitivity of the reaction rate to temperature changes.

Substituting Eq. (4.6) into Eq. (4.5), we could write the steady-state energy balance for the first-order reaction in the spherical catalyst pellet as

$$\left(\frac{d^2 T}{dr^2} + \frac{2}{r} \cdot \frac{dT}{dr}\right) + \frac{k(T_b) \cdot \exp\left(-\gamma \left(\frac{T_b}{T} - 1\right)\right) \cdot C_A \cdot (-\Delta H_{R_A})}{k_{\text{eff}}} = 0 \quad (4.7)$$

The boundary conditions are

- At the center of catalyst pellet:

There is no heat flux through the pellet center since this is a point of symmetry.

$$\frac{dT}{dr} = 0 \quad \text{at } r = 0 \quad (4.8)$$

- At the external surface of catalyst pellet:

- ◊ Fixed temperature at the external surface of the catalyst.

The temperature at the pellet external surface is equal to the bulk temperature due to the negligible resistance to external mass transfer.

$$T = T_b \quad (4.9)$$

- ◊ Heat transfer across the boundary at the pellet surface.

We derive the energy balance at the pellet surface as

$$(q_r \times 4\pi r^2) \Big|_{r=R} - 4\pi r^2 \cdot h \cdot (T \Big|_{r=R} - T_b) = 0,$$

where h is the heat transfer coefficient between the catalyst pellet and the bulk fluid. Simplifying the above equation and using the flux definition by Eq. (4.1) yields

$$h \cdot (T_b - T \Big|_{r=R}) = k_{\text{eff}} \cdot \frac{dT}{dr} \Big|_{r=R} \quad (4.10)$$

Introducing dimensionless variables $\theta = \frac{T}{T_b}$, $\rho = \frac{r}{R}$ and $c = \frac{C_A}{C_{Ab}}$, we rewrite Eq. (4.7) as

$$\frac{T_b}{R^2} \frac{d^2\theta}{d\rho^2} + \frac{2}{\rho} \cdot \frac{T_b}{R} \frac{d\theta}{d\rho} + \frac{k(T_b) \cdot \exp\left(-\gamma\left(\frac{T_b}{T} - 1\right)\right) \cdot (-\Delta H_{R_A})}{k_{\text{eff}}} (c \cdot C_{Ab}) = 0 \quad (4.11)$$

Dividing through by $\frac{T_b}{R^2}$ results in

$$\frac{d^2\theta}{d\rho^2} + \frac{2}{\rho} \cdot \frac{d\theta}{d\rho} + \frac{(-\Delta H_{R_A}) D_{\text{eff},A} C_{Ab}}{k_{\text{eff}} T_b} \cdot \phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta}\right)\right) \cdot c = 0 \quad (4.12)$$

where ϕ is the dimensionless Thiele modulus evaluated at T_b , $\phi^2(T_b) = \frac{k(T_b) \cdot R^2}{D_{\text{eff},A}}$.

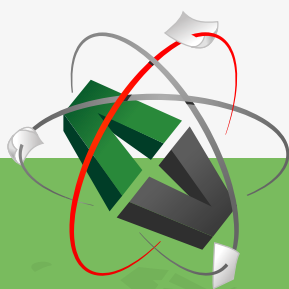
Introducing the energy generation function

$$\beta = \frac{(-\Delta H_{R_A}) D_{\text{eff},A} C_{Ab}}{k_{\text{eff}} T_b}, \quad (4.13)$$

we derive the dimensionless form of energy balance equation for the first-order reaction in non-isothermal spherical pellet as

$$\frac{d^2\theta}{d\rho^2} + \frac{2}{\rho} \cdot \frac{d\theta}{d\rho} + \beta \phi_s^2 \exp\left(\gamma\left(1 - \frac{1}{\theta}\right)\right) c = 0 \quad (4.14)$$

This e-book
is made with
SetaPDF



PDF components for PHP developers

www.setasign.com



The dimensionless boundary conditions are

◇ At the center of catalyst pellet:

$$\frac{d\theta}{d\rho} = 0 \quad \text{at} \quad \rho = 0 \quad (4.15)$$

◇ At the external surface of catalyst pellet:

o Fixed temperature at the external surface of the catalyst pellet.

$$\theta = 1 \quad (4.16)$$

o Heat transfer across the boundary at the pellet external surface.

$$1 - \theta = \frac{1}{Bi_h} \cdot \frac{d\theta}{d\rho} \bigg|_{\rho=1} \quad (4.17)$$

where $Bi_h = \frac{R \cdot h}{k_{\text{eff}}}$ is the Biot number for heat transfer. The Biot number is the ratio of resistance to heat conduction in the pellet to the external resistance to heat convection at the pellet surface. The temperature profile within the pellet is uniform at the low value of Biot number due to the small resistance to heat conduction.

Damkohler relation between concentration and temperature

Next we will derive the linear relation between the concentration and temperature. Dividing Eq. (4.4) by $(-\Delta H_{R_A})$ and adding to Eq. (2.5), we have:

$$\frac{d}{dr} \left(r^2 \cdot D_{\text{eff},A} \cdot \frac{dC_A}{dr} \right) + \frac{d}{dr} \left(r^2 \cdot \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot \frac{dT}{dr} \right) = 0 \quad (4.18)$$

Integrating Eq. (4.18) from a pellet center $r=0$ to a radius r yields

$$r^2 \left(D_{\text{eff},A} \cdot \frac{dC_A}{dr} + \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot \frac{dT}{dr} \right) = C_1, \quad (4.19)$$

where C_1 is the integration constant.

Dividing by r^2 and rearranging gives

$$\frac{d}{dr} \left(D_{\text{eff},A} \cdot C_A + \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot T \right) = \frac{C_1}{r^2} \quad (4.20)$$

From the boundary conditions by Eqs. (2.9) and (4.8), we get $C_1 = 0$.

Thus,

$$\frac{d}{dr} \left(D_{\text{eff},A} \cdot C_A + \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot T \right) = 0$$

A second integration from the radius r to the pellet surface R gives

$$\left\{ D_{\text{eff},A} \cdot C_{Ab} + \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot T_b \right\} = \left\{ D_{\text{eff},A} \cdot C_A(r) + \frac{k_{\text{eff}}}{(-\Delta H_{R_A})} \cdot T(r) \right\} \quad (4.21)$$

Rearranging Eq. (4.21), we obtain

$$T(r) - T_b = D_{\text{eff},A} \cdot \frac{(-\Delta H_{R_A})}{k_{\text{eff}}} \cdot (C_{Ab} - C_A(r)) \quad (4.22)$$

This linear relation between $C_A(r)$ and $T(r)$ is called a Damkohler relation. Thus, we can solve only one differential equation corresponding either mass or heat balance and find another variable from Eq. (4.22).

The Damkohler relation in dimensionless form is

$$\theta - 1 = D_{\text{eff},A} \cdot \frac{(-\Delta H_{R_A})}{k_{\text{eff}} \cdot T_b} \cdot C_{Ab} \cdot (1 - c) = \beta \cdot (1 - c) \quad (4.23)$$

Then, we can relate the dimensionless temperature to the dimensionless concentration by

$$\theta = 1 + \beta \cdot (1 - c) \quad (4.24)$$

Therefore, we can rewrite Eq. (4.6) as

$$k(T) = k(T_b) \cdot \exp \left(-\gamma \left(\frac{1}{1 + \beta \cdot (1 - c)} - 1 \right) \right) = k(T_b) \cdot \exp \left(\frac{\gamma \cdot \beta \cdot (1 - c)}{1 + \beta \cdot (1 - c)} \right) \quad (4.25)$$

Substituting Eq. (4.25) in Eq. (2.15), we rewrite the dimensionless mass balance as

$$\frac{d^2 c}{d\rho^2} + \frac{2}{\rho} \cdot \frac{dc}{d\rho} - (\phi(T_b))^2 \cdot \exp \left(\frac{\gamma \cdot \beta \cdot (1 - c)}{1 + \beta \cdot (1 - c)} \right) \cdot c = 0 \quad (4.26)$$

Thus, we can solve Eq. (4.26) with boundary conditions by Eqs. (2.16) and (2.18) to obtain the dimensionless concentration profile in the radial direction of the pellet. Then, we calculate the dimensionless temperature profile by Eq. (4.24).

The maximum temperature difference between the external surface and the center of catalyst pellet can be evaluated using the Damkohler relation by Eq. (4.22) and substituting $C_A = 0$ at $r = 0$.

$$(T|_{r=0} - T_b)_{\max} = \frac{D_{\text{eff},A}(-\Delta H_{R_A})C_{Ab}}{k_{\text{eff}}} \quad (4.27)$$

Using β by Eq. (4.13), we can rewrite Eq. (4.27) as

$$\beta = \frac{(T|_{r=0} - T_b)_{\max}}{T_b}$$

Therefore, the energy generation function β characterizes the ratio of the maximum temperature difference that can exist within the pellet to the bulk temperature.



**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider



4.2 NUMERICAL SOLUTION OF MODEL EQUATIONS USING FINITE-DIFFERENCE METHOD

The dimensionless mass and heat balance equations for the first-order non-isothermal chemical reaction in the spherical catalyst pellet are summarized as follows:

- o mass balance equation

$$\frac{d^2c}{d\rho^2} + \frac{2}{\rho} \cdot \frac{dc}{d\rho} - \phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta}\right)\right)c = 0 \quad (4.28)$$

- o heat balance equation

$$\frac{d^2\theta}{d\rho^2} + \frac{2}{\rho} \cdot \frac{d\theta}{d\rho} + \beta\phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta}\right)\right)c = 0 \quad (4.29)$$

- o boundary conditions at the pellet center are

$$\frac{dc}{d\rho} = 0, \quad \frac{d\theta}{d\rho} = 0 \quad \text{at} \quad \rho = 0 \quad (4.30)$$

- o boundary conditions at the outer surface of the pellet

$$\frac{dc}{d\rho} = Bi_m \cdot (1 - c), \quad \frac{d\theta}{d\rho} = Bi_h \cdot (1 - \theta) \quad \text{at} \quad \rho = 1 \quad (4.31)$$

We use a finite-difference method for discretization of the second-order differential equations and boundary conditions (Beers 2007, p270). We introduce a grid of uniformly-spaced N internal points, ρ_1, \dots, ρ_N , and two end points, ρ_0 and ρ_{N+1} , as

$$\rho_0 = 0 < \rho_1 < \dots < \rho_N < \rho_{N+1} = 1$$

Thus, the total number of grid points is $N+2$ and the number of grid intervals is $N+1$. The internal points are located at $\rho_k = k \cdot \Delta\rho$, $k=1, \dots, N$, where $\Delta\rho = \frac{1}{N+1}$.

We apply the central difference approximations to the first and second order derivatives at the grid point ρ_k as

$$\left. \frac{dc}{d\rho} \right|_{\rho_k} = \frac{c(\rho_{k+1}) - c(\rho_{k-1}))}{2\Delta\rho}, \quad \left. \frac{d\theta}{d\rho} \right|_{\rho_k} = \frac{\theta(\rho_{k+1}) - \theta(\rho_{k-1}))}{2\Delta\rho}$$

$$\left. \frac{d^2c}{d\rho^2} \right|_{\rho_k} = \frac{c(\rho_{k+1}) - 2c(\rho_k) + c(\rho_{k-1}))}{(\Delta\rho)^2}, \quad \left. \frac{d^2\theta}{d\rho^2} \right|_{\rho_k} = \frac{\theta(\rho_{k+1}) - 2\theta(\rho_k) + \theta(\rho_{k-1}))}{(\Delta\rho)^2}$$

Substitution of these approximations into the differential equation Eqs. (4.28) and (4.29) results in

$$\frac{c(\rho_{k+1}) - 2c(\rho_k) + c(\rho_{k-1}))}{(\Delta\rho)^2} + \frac{2}{\rho_k} \cdot \frac{c(\rho_{k+1}) - c(\rho_{k-1}))}{2\Delta\rho} - \phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \quad (4.32)$$

$$\frac{\theta(\rho_{k+1}) - 2\theta(\rho_k) + \theta(\rho_{k-1}))}{(\Delta\rho)^2} + \frac{2}{\rho_k} \cdot \frac{\theta(\rho_{k+1}) - \theta(\rho_{k-1}))}{2\Delta\rho} + \beta\phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \quad (4.33)$$

Rearrangement of Eqs. (4.32) and (4.33) yields

$$\begin{aligned} \left(\frac{1}{(\Delta\rho)^2} - \frac{1}{\rho_k \Delta\rho}\right) \cdot c(\rho_{k-1}) + \left(-\frac{2}{(\Delta\rho)^2}\right) \cdot c(\rho_k) + \left(\frac{1}{(\Delta\rho)^2} + \frac{1}{\rho_k \Delta\rho}\right) \cdot c(\rho_{k+1}) - \\ \phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \\ \left(\frac{1}{(\Delta\rho)^2} - \frac{1}{\rho_k \Delta\rho}\right) \cdot \theta(\rho_{k-1}) + \left(-\frac{2}{(\Delta\rho)^2}\right) \cdot \theta(\rho_k) + \left(\frac{1}{(\Delta\rho)^2} + \frac{1}{\rho_k \Delta\rho}\right) \cdot \theta(\rho_{k+1}) + \\ \beta\phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \end{aligned}$$

The above equations can be rearranged as

$$A_k \cdot c(\rho_{k-1}) + B \cdot c(\rho_k) + C_k \cdot c(\rho_{k+1}) - \phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \quad (4.34)$$

$$A_k \cdot \theta(\rho_{k-1}) + B \cdot \theta(\rho_k) + C_k \cdot \theta(\rho_{k+1}) + \beta\phi^2 \exp\left(\gamma\left(1 - \frac{1}{\theta(\rho_k)}\right)\right) c(\rho_k) = 0 \quad (4.35)$$

Here we define the coefficients A_k , B and C_k as

$$A_k = \frac{1}{(\Delta\rho)^2} - \frac{1}{\rho_k \Delta\rho}, \quad B = -\frac{2}{(\Delta\rho)^2}, \quad C_k = \frac{1}{(\Delta\rho)^2} + \frac{1}{\rho_k \Delta\rho}$$

Using a second-order forward approximation formula we formulate the discretized equations for the boundary conditions at $\rho_0 = 0$ by Eq. (4.30) as follows

$$\frac{-3c(\rho_0) + 4c(\rho_1) - c(\rho_2)}{2\Delta\rho} = 0, \quad \frac{-3\theta(\rho_0) + 4\theta(\rho_1) - \theta(\rho_2)}{2\Delta\rho} = 0$$

Thus, we can calculate the concentration and temperature at ρ_0 as

$$c(\rho_0) = -\frac{1}{3}c(\rho_2) + \frac{4}{3}c(\rho_1) \quad (4.36)$$

$$\theta(\rho_0) = -\frac{1}{3}\theta(\rho_2) + \frac{4}{3}\theta(\rho_1) \quad (4.37)$$

Introducing $c(\rho_0)$ by Eq. (4.36) and $\theta(\rho_0)$ by Eq. (4.37) into Eqs. (4.34) and (4.35) for $k=1$, we have:

$$A_1 \cdot \left[-\frac{1}{3}c(\rho_2) + \frac{4}{3}c(\rho_1) \right] + B \cdot c(\rho_1) + C_1 \cdot c(\rho_2) - \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_1)}\right)\right) c(\rho_1) = 0 \quad (4.38)$$

$$A_1 \cdot \left[-\frac{1}{3}\theta(\rho_2) + \frac{4}{3}\theta(\rho_1) \right] + B \cdot \theta(\rho_1) + C_1 \cdot \theta(\rho_2) + \beta \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_1)}\right)\right) c(\rho_1) = 0 \quad (4.39)$$

Rearranging Eqs. (4.38) and (4.39) results in

$$\left[B + \frac{4}{3}A_1 \right] \cdot c(\rho_1) + \left[C_1 - \frac{1}{3}A_1 \right] \cdot c(\rho_2) - \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_1)}\right)\right) c(\rho_1) = 0 \quad (4.40)$$

$$\left[B + \frac{4}{3}A_1 \right] \cdot \theta(\rho_1) + \left[C_1 - \frac{1}{3}A_1 \right] \cdot \theta(\rho_2) + \beta \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_1)}\right)\right) c(\rho_1) = 0 \quad (4.41)$$



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



We use a second-order backward approximation formula for discretizing the boundary condition at ρ_{N+1} by Eq. (4.31) as

$$\frac{3c(\rho_{N+1}) - 4c(\rho_N) + c(\rho_{N-1}))}{2\Delta\rho} = Bi_m \cdot (1 - c(\rho_{N+1}))$$

$$\frac{3\theta(\rho_{N+1}) - 4\theta(\rho_N) + \theta(\rho_{N-1}))}{2\Delta\rho} = Bi_h \cdot (1 - \theta(\rho_{N+1}))$$

The above equations can be rearranged as

$$\left(Bi_m + \frac{3}{2\Delta\rho}\right) \cdot c(\rho_{N+1}) - \left(\frac{2}{\Delta\rho}\right) \cdot c(\rho_N) + \left(\frac{1}{2\Delta\rho}\right) \cdot c(\rho_{N-1}) - Bi_m = 0$$

$$\left(Bi_h + \frac{3}{2\Delta\rho}\right) \cdot \theta(\rho_{N+1}) - \left(\frac{2}{\Delta\rho}\right) \cdot \theta(\rho_N) + \left(\frac{1}{2\Delta\rho}\right) \cdot \theta(\rho_{N-1}) - Bi_h = 0$$

We can calculate the concentration and temperature at ρ_{N+1} as

$$c(\rho_{N+1}) = \frac{2 \cdot Bi_m \cdot \Delta\rho + 4c(\rho_N) - c(\rho_{N-1}))}{3 + 2 \cdot Bi_m \cdot \Delta\rho} \quad (4.42)$$

$$\theta(\rho_{N+1}) = \frac{2 \cdot Bi_h \cdot \Delta\rho + 4\theta(\rho_N) - \theta(\rho_{N-1}))}{3 + 2 \cdot Bi_h \cdot \Delta\rho} \quad (4.43)$$

Introducing $c(\rho_{N+1})$ by Eq. (4.42) and $\theta(\rho_{N+1})$ by Eq. (4.43) into Eqs. (4.34) and (4.35) for $k = N$, we get:

$$A_N \cdot c(\rho_{N-1}) + B \cdot c(\rho_N) + C_N \cdot \left[\frac{2 \cdot Bi_m \cdot \Delta\rho + 4c(\rho_N) - c(\rho_{N-1}))}{3 + 2 \cdot Bi_m \cdot \Delta\rho} \right] - \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_N)}\right)\right) c(\rho_N) = 0 \quad (4.44)$$

$$A_N \cdot c(\rho_{N-1}) + B \cdot c(\rho_N) + C_N \cdot \left[\frac{2 \cdot Bi_m \cdot \Delta\rho + 4c(\rho_N) - c(\rho_{N-1}))}{3 + 2 \cdot Bi_m \cdot \Delta\rho} \right] - \phi^2 \exp\left(\gamma \left(1 - \frac{1}{\theta(\rho_N)}\right)\right) c(\rho_N) = 0 \quad (4.45)$$

Rearranging Eqs. (4.44) and (4.45) results in

$$\left[C_N \left(\frac{-1}{3+2 \cdot Bi_m \cdot \Delta \rho} \right) + A_N \right] \cdot c(\rho_{N-1}) + \left[C_N \left(\frac{4}{3+2 \cdot Bi_m \cdot \Delta \rho} \right) + B \right] \cdot c(\rho_N) + C_N \left[\frac{2 \cdot Bi_m \cdot \Delta \rho}{3+2 \cdot Bi_m \cdot \Delta \rho} \right] - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right) c(\rho_N) = 0 \quad (4.46)$$

$$\left[C_N \left(\frac{-1}{3+2 \cdot Bi_h \cdot \Delta \rho} \right) + A_N \right] \cdot \theta(\rho_{N-1}) + \left[C_N \left(\frac{4}{3+2 \cdot Bi_h \cdot \Delta \rho} \right) + B \right] \cdot \theta(\rho_N) + C_N \left[\frac{2 \cdot Bi_h \cdot \Delta \rho}{3+2 \cdot Bi_h \cdot \Delta \rho} \right] + \beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right) c(\rho_N) = 0 \quad (4.47)$$

Therefore, we obtain the following system of $2 \times N$ nonlinear algebraic equations:

$$f(1) = \left[B + \frac{4}{3} A_1 \right] \cdot c(\rho_1) + \left[C_1 - \frac{1}{3} A_1 \right] \cdot c(\rho_2) - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_1)} \right) \right) c(\rho_1) = 0$$

$$f(k) = A_k \cdot c(\rho_{k-1}) + B \cdot c(\rho_k) + C_k \cdot c(\rho_{k+1}) - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_k)} \right) \right) c(\rho_k) = 0, \\ k = 2, \dots, N-1$$

$$f(N) = \left[C_N \left(\frac{-1}{3+2 \cdot Bi_m \cdot \Delta \rho} \right) + A_N \right] \cdot c(\rho_{N-1}) + \left[C_N \left(\frac{4}{3+2 \cdot Bi_m \cdot \Delta \rho} \right) + B \right] \cdot c(\rho_N) + C_N \left[\frac{2 \cdot Bi_m \cdot \Delta \rho}{3+2 \cdot Bi_m \cdot \Delta \rho} \right] - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right) c(\rho_N) = 0 \quad (4.48)$$

$$f(N+1) = \left[B + \frac{4}{3} A_1 \right] \cdot \theta(\rho_1) + \left[C_1 - \frac{1}{3} A_1 \right] \cdot \theta(\rho_2) + \beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_1)} \right) \right) c(\rho_1) = 0$$

$$f(N+k) = A_k \cdot \theta(\rho_{k-1}) + B \cdot \theta(\rho_k) + C_k \cdot \theta(\rho_{k+1}) + \beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_k)} \right) \right) c(\rho_k) = 0, \\ k = 2, \dots, N-1$$

$$f(2N) = \left[C_N \left(\frac{-1}{3+2 \cdot Bi_h \cdot \Delta \rho} \right) + A_N \right] \cdot \theta(\rho_{N-1}) + \left[C_N \left(\frac{4}{3+2 \cdot Bi_h \cdot \Delta \rho} \right) + B \right] \cdot \theta(\rho_N) + C_N \left[\frac{2 \cdot Bi_h \cdot \Delta \rho}{3+2 \cdot Bi_h \cdot \Delta \rho} \right] + \beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right) c(\rho_N) = 0$$

Then, we specify the Jacobian matrix, which is the matrix of derivatives of i equation, $i=1,\dots,2N$, with respect to $c(\rho_k)$ and $\theta(\rho_k)$, $k=1,\dots,N$ as

$$\mathbf{J} = \begin{matrix} & \begin{matrix} \overbrace{j=1}^{k=1} & \dots & \overbrace{j=N}^{k=N} & \overbrace{j=N+1}^{k=1} & \dots & \overbrace{j=2N}^{k=N} \end{matrix} \\ \begin{matrix} i=1 \\ \vdots \\ i=N \\ i=N+1 \\ \vdots \\ i=2N \end{matrix} & \begin{bmatrix} \frac{\partial f_1}{\partial c(\rho_1)} & \dots & \frac{\partial f_1}{\partial c(\rho_N)} & \frac{\partial f_1}{\partial \theta(\rho_1)} & \dots & \frac{\partial f_1}{\partial \theta(\rho_N)} \\ \vdots & \dots & \vdots & \vdots & \dots & \vdots \\ \frac{\partial f_N}{\partial c(\rho_1)} & \dots & \frac{\partial f_N}{\partial c(\rho_N)} & \frac{\partial f_N}{\partial \theta(\rho_1)} & \dots & \frac{\partial f_N}{\partial \theta(\rho_N)} \\ \frac{\partial f_{N+1}}{\partial c(\rho_1)} & \dots & \frac{\partial f_{N+1}}{\partial c(\rho_N)} & \frac{\partial f_{N+1}}{\partial \theta(\rho_1)} & \dots & \frac{\partial f_{N+1}}{\partial \theta(\rho_N)} \\ \vdots & \dots & \vdots & \vdots & \dots & \vdots \\ \frac{\partial f_{2N}}{\partial c(\rho_1)} & \dots & \frac{\partial f_{2N}}{\partial c(\rho_N)} & \frac{\partial f_{2N}}{\partial \theta(\rho_1)} & \dots & \frac{\partial f_{2N}}{\partial \theta(\rho_N)} \end{bmatrix} \end{matrix}$$



GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead

The structure of the Jacobian matrix is shown in Fig. 4.1, where ❶, ❷, ❸, ❹ and ❺ are the matrix diagonals.

$$J =$$

	$j=1$				N		$N+1$		$2N$	
$i=1$	❶	❸	0	0	0	❺	0	0	0	0
	❷	❶	❸	0	0	0	❺	0	0	0
	0	❷	❶	❸	0	0	0	❺	0	0
	0	0	❷	❶	❸	0	0	0	❺	0
N	0	0	0	❷	❶	0	0	0	0	❺
$N+1$	❹	0	0	0	0	❶	❸	0	0	0
	0	❹	0	0	0	❷	❶	❸	0	0
	0	0	❹	0	0	0	❷	❶	❸	0
	0	0	0	❹	0	0	0	❷	❶	❸
$2N$	0	0	0	0	❹	0	0	0	❷	❶

Figure 4.1: The structure of Jacobian matrix.

The diagonal elements of the Jacobian matrix are summarized below:

◇ J_{d1}

$$J(1,1) = \left[B + \frac{4}{3} A_1 \right] - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_1)} \right) \right)$$

$$J(i,i) = B - \phi_s^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_i)} \right) \right), \quad i = 2, \dots, N-1$$

$$J(N,N) = C_N \left(\frac{4}{3 + 2 \cdot Bi_m \cdot \Delta \rho} \right) + B - \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right)$$

$$J(N+1,N+1) = \left[B + \frac{4}{3} A_1 \right] + \frac{\beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_1)} \right) \right) \cdot c(\rho_1)}{\theta^2(\rho_1)}$$

$$J(i,i) = B + \frac{\beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_{i-N})} \right) \right) c(\rho_{i-N})}{\theta^2(\rho_{i-N})}, \quad i = N+2, \dots, 2N-1$$

$$J(2N,2N) = C_N \left(\frac{4}{3 + 2 \cdot Bi_h \cdot \Delta \rho} \right) + B + \frac{\beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_N)} \right) \right) c(\rho_N)}{\theta^2(\rho_N)}$$

◇ J_{d2}

$$J(i, i-1) = A_i, \quad i = 2, \dots, N-1$$

$$J(N, N-1) = C_N \left(\frac{-1}{3 + 2 \cdot Bi_m \cdot \Delta \rho} \right) + A_N$$

$$J(N+1, N) = 0$$

$$J(i, i-1) = A_{i-N}, \quad i = N+2, \dots, 2N-1$$

$$J(2N, 2N-1) = C_N \left(\frac{-1}{3 + 2 \cdot Bi_h \cdot \Delta \rho} \right) + A_N$$

◇ J_{d3}

$$J(1, 2) = C_1 - \frac{1}{3} A_1$$

$$J(i, i+1) = C_i, \quad i = 2, \dots, N-1$$

$$J(N, N+1) = 0$$

$$J(N+1, N+2) = C_1 - \frac{1}{3} A_1$$

$$J(i, i+1) = C_{i-N}, \quad i = N+2, \dots, 2N-1$$

◇ J_{d4}

$$J(N+i, i) = \beta \phi^2 \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_i)} \right) \right), \quad i = 1, \dots, N$$

◇ J_{d5}

$$J(i, N+i) = - \frac{\phi^2 c(\rho_i) \exp \left(\gamma \left(1 - \frac{1}{\theta(\rho_i)} \right) \right)}{\theta^2(\rho_i)}, \quad i = 1, \dots, N$$

4.3 COMPUTER PROGRAM DESCRIPTION

Notebook *FirstOrder_Nonisothermal.ipynb*

◇ Import packages.

We will use the Python library *NumPy* for vector manipulations and formation of diagonal matrices. The *sparse* module from the library *SciPy* is called to generate a sparse matrix and convert it to various formats as well as to compute the norm of a vector. The *spsolve* module from the *SciPy* library is utilized to solve the sparse linear system. The exponential function is calculated using the *exp* function from the *math* module. To plot the results of numerical simulations we will use the *matplotlib* library. The *IPython.html* module is called to display widgets for interactive input of parameter values.

```
%matplotlib inline
import numpy as np
from scipy import sparse
from scipy.sparse.linalg import spsolve
from scipy.linalg import norm
from math import exp
from matplotlib.pyplot import *
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, LaTeX
```



Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)

Management

Time Management

Problem solving

Self-Confidence

Effectiveness

Project Management

Goal setting

Motivation

Coaching

Download free eBooks at bookboon.com

[Click on the ad to read more](#)

- ◇ Define the main function *fmain*:

Input parameters:

ϕ – Thiele modulus

β – energy generation function

γ – Arrhenius number

Bi_m – Biot number for mass transfer

Bi_h – Biot number for heat transfer

```
#
#-----
#
# Main function
#
def fmain ( $\phi$ ,  $\beta$ ,  $\gamma$ ,  $Bi_m$ ,  $Bi_h$ , **kwargs):
```

- o Specify the grid.

We specify the number of internal grid points N . Thus, the grid is divided on $N+1$ subintervals of the same length $\Delta\rho$. Then, we generate the grid of internal points and saved it in the array ρ .

```
#-----
# The number of internal grid points
# N = 299
#-----
# set grid spacing
#  $\Delta\rho = 1./(N + 1.)$ 
# specify uniform grid of internal points
#  $\rho = \text{np.linspace}(\Delta\rho, 1.-\Delta\rho, N)$ 
```

- o Specify common coefficients in the system of nonlinear equations.

```
# allocate arrays of common coefficients A and C
# A = np.zeros(N)
# C = np.zeros(N)
# define constant common coefficient B
# B = - 2./( $\Delta\rho^{**2}$ )
# define coefficients A and B at each mesh point
# A = 1./( $\Delta\rho^{**2}$ ) - 1./( $\Delta\rho * \rho$ )
# C = 1./( $\Delta\rho^{**2}$ ) + 1./( $\Delta\rho * \rho$ )
```

- o Set initial guesses for solution of nonlinear equations.

We will use uniform profiles, i.e. $c(\rho) = \theta(\rho) = 0.5$, as initial guesses. Here, the vector u_0 contains both the dimensionless concentration and temperature at the internal grid points.

```
#####
# specify initial guess for solution of nonlinear equations
# we will use uniform profiles: c=θ=0.5
u0 = 0.5*np.ones(2*N)
```

- o Allocate arrays of dimensionless concentration and temperature at the grid points.

```
#####
# specify initial guess for solution of nonlinear equations
# we will use uniform profiles: c=θ=0.5
u0 = 0.5*np.ones(2*N)
```

- o Use the Newton-Raphson method to solve the system of nonlinear equations.

```
#
# solve the system of nonlinear equations using the Newton-Raphson method
u,f_val,iter, exitflag = fsolve(Δρ, φ, β, γ, Bi_m, Bi_h, A, B, C, u0)
#
if not exitflag:
    print ('Too many iterations')
```

- o Save results.

The resulting profiles at internal points are saved in arrays *concentration* and *temperature*. The values of concentration and temperature are calculated at the pellet center and surface using the second-order forward and backward difference formulas to approximate the derivatives in the boundary conditions at $\rho = 0$ and $\rho = 1$, respectively.

```
#
# save resulting concentration and temperature profiles
# profiles at internal grid points
concentration[1:N+1] = u[0:N]
temperature[1:N+1] = u[N:2*N]
# calculate concentration and temperature at the pellet center, ρ = 0
concentration[0] = 4.*u[0]/3. - u[1]/3.
temperature[0] = 4.*u[N]/3. - u[N+1]/3.
# calculate concentration and temperature at the pellet surface, ρ = 1
concentration[N + 1] = (2.*Bi_m*Δρ + 4.*u[N-1] - u[N-2])/(3. + 2.*Bi_m*Δρ)
temperature[N + 1] = (2.*Bi_h*Δρ + 4.*u[2*N-1] - u[2*N-2])/(3. + 2.*Bi_h*Δρ)
```

- o Finally, we plot the concentrations and temperature profiles and output the values of conversion, selectivity and yield.

```
#
# plot graphs
rad = np.linspace(0.0, 1.0, N + 2)
matplotlib.rcParams.update({'font.size': 12})
title1 = 'First-order reaction in non-isothermal spherical catalyst: \n'
title1 += '$\phi$ = %s, $\beta$ = %s, $\gamma$ = %s, Bi_m = %s, Bi_h = %s \n'
fig = plt.figure()
axes = fig.add_subplot(111)
axes.plot(rad, concentration, color="green", linestyle="solid", linewidth=2)
axes.set_title(title1 % (\phi, \beta, \gamma, Bi_m, Bi_h))
axes.set_xlabel('Dimensionless radius, $\rho$ [-]')
axes.set_ylabel('Dimensionless concentration, c [-]')
axes.grid()
axes.set_xlim([0.,1.])
axes.set_ylim([0.,1.])
plt.show()
fig2 = plt.figure()
axes = fig2.add_subplot(111)
axes.plot(rad, temperature, color="red", linestyle="solid", linewidth=2)
axes.set_xlabel('Dimensionless radius, $\rho$ [-]')
axes.set_ylabel('Dimensionless temperature, $\theta$ [-]')
axes.grid()
axes.set_xlim([0.,1.])
axes.set_ylim([0.,3.])
plt.show()
return [rad, concentration, temperature]
```



Brain power

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations.

Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

SKF

- ◇ Solve the system of nonlinear equations by the Newton-Raphson method.

```
#
#-----
#
# Solve the system of nonlinear equations using Newton-Raphson
# method with diagonal Jacobian.
#
def fsolve( $\Delta\rho$ ,  $\phi$ ,  $\beta$ ,  $\gamma$ , Bi_m, Bi_h, A, B, C, x0):
#
    tolX = 1e-6
    tolG = 1e-6
    maxIter = 500
    x = x0
    for i in range (maxIter):
        gx = ff(x,  $\Delta\rho$ ,  $\phi$ ,  $\beta$ ,  $\gamma$ , Bi_m, Bi_h, A, B, C)
        JJ = jac(x,  $\Delta\rho$ ,  $\phi$ ,  $\beta$ ,  $\gamma$ , Bi_m, Bi_h, A, B, C)
        xNew = x - spsolve (JJ, gx )
        if (norm(xNew-x) < tolX):
            return [x,gx,i,'True']
        if (norm(gx) < tolG):
            return [xNew,gx,i,'True']
        x = xNew
    return [x, gx, i, 'False']
```

- ◇ Define the system of nonlinear equations.
 - o The function ff:

```
#
# Define the system of nonlinear equations
#
def ff (u,  $\Delta\rho$ ,  $\phi$ ,  $\beta$ ,  $\gamma$ , Bi_m, Bi_h, A, B, C):
```

- o At first, we allocate arrays and separate u vector into cc and θ vectors.

```
#
# allocate arrays
N = int(len(u)/2)
cc = np.zeros(N)
 $\theta$  = np.zeros(N)
f = np.zeros(2*N)
# convert u array to cc and  $\theta$  arrays
cc = u[0:N]
 $\theta$  = u[N:2*N]
```

- o Then, we specify the reaction terms in the mass and heat balance equations.

```
#
#*****
# reaction term in mass balance equation
f[0:N] -= (phi**2)*exp(gamma*(1.-1./theta[0:N]))*cc[0:N]
# reaction term in heat balance equation
f[N:2*N] += beta*(phi**2)*exp(gamma*(1.-1./theta[0:N]))*cc[0:N]
#*****
```

- o Next, we specify the discrete balance equations at the internal points.

```
#
f[0:N] += B * cc[0:N]
f[N:2*N] += B * theta[0:N]
# pellet internal points
for k in range(1,(N-1)):
    f[k] += A[k]*cc[k-1] + C[k]*cc[k+1]
    f[N+k] += A[k]*theta[k-1] + C[k]*theta[k+1]
```

- o Finally, we define the balance equations at the center and surface of the pellet.

```
# pellet center
f[0] += (4.*A[0]/3.)*cc[0] + (C[0]-A[0]/3.)*cc[1]
f[N] += (4.*A[0]/3.)*theta[0] + (C[0]-A[0]/3.)*theta[1]
# pellet surface
f[N-1] += (C[N-1]*(-1./(3.+2.*Bi_m*Delta_rho)) + A[N-1])*cc[N-2] + \
           (C[N-1]*(4./(3.+2.*Bi_m*Delta_rho)))*cc[N-1] + \
           C[N-1]*(2.*Delta_rho*Bi_m/(3.+2.*Bi_m*Delta_rho))
#
f[2*N-1] += (C[N-1]*(-1./(3.+2.*Bi_h*Delta_rho)) + A[N-1])*theta[N-2] + \
            (C[N-1]*(4./(3.+2.*Bi_h*Delta_rho)))*theta[N-1] + \
            C[N-1]*(2.*Delta_rho*Bi_h/(3.+2.*Bi_h*Delta_rho))
#
return f
```

- ◇ Define the Jacobian matrix.

- o The function *jac*:

```
#
#-----
# Define Jacobian matrix
#
def jac (u, Delta_rho, phi, beta, gamma, Bi_m, Bi_h, A, B, C):
```

- o We allocate arrays and separate u vector into cc and θ vectors.

```
#  
# allocate arrays  
N = int(len(u)/2)  
cc = np.zeros(N)  
theta = np.zeros(N)  
# convert u array to cc and theta arrays  
cc = u[0:N]  
theta = u[N:2*N]
```

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on

- o At first, we form the main diagonal d1.

```
#
# Main Diagonal d1
#
d1 = np.zeros(2*N)
#####
# derivative of reaction rate with respect to concentration
d1[0:N] += -(phi**2)*exp(gamma*(1.-1./theta[0:N]))
# derivative of reaction rate with respect to temperature
d1[N:2*N] += beta*(phi**2)*exp(gamma*(1.-1./theta[0:N]))*cc[0:N]/(theta[0:N]**2)
#####
#
d1 += B
# boundary condition at pellet center
d1[0] += 4.*A[0]/3.
d1[N] += 4.*A[0]/3.
# boundary condition at pellet surface
d1[N-1] += C[N-1]*(4./(3.+2.*Bi_m*Delta_rho))
d1[2*N-1] += C[N-1]*(4./(3.+2.*Bi_h*Delta_rho))
# position of non-zero elements
Jd1 = arange(0, 2*N)
Id1 = arange(0, 2*N)
```

- o Then, we specify other diagonals:
diagonal d2

```
#
# Diagonal d2
#
d2 = np.zeros (2*N-1)
d2[0:N-2] = A[1:N-1]
d2[N-2] = C[N-1]*(-1./(3.+2.*Bi_m*Delta_rho))+A[N-1]
d2[N-1] = 0.
d2[N:2*N-2] = A[1:N-1]
d2[2*N-2] = C[N-1]*(-1./(3.+2.*Bi_h*Delta_rho))+A[N-1]
Id2 = arange(1, 2*N)
Jd2 = arange(0, 2*N-1)
```

- o diagonal d3

```
#
# Diagonal d3
#
d3 = np.zeros (2*N-1)
d3[0] = C[0] - A[0]/3.
d3[1:N-1] = C[1:N-1]
d3[N-1] = 0.
d3[N] = C[0] - A[0]/3.
d3[N+1:2*N-1] = C[1:N-1]
Jd3 = arange(1, 2*N)
Id3 = arange(0, 2*N-1)
```

o diagonal d4

```
#
# Diagonal d4
#
d4 = np.zeros(N)
d4 +=  $\beta * (\phi^{**2}) * \exp(\gamma * (1. - 1./\theta))$ 
Id4 = arange (N, 2*N)
Jd4 = arange (0, N)
```

o diagonal d5

```
#
# Diagonal d5
#
d5 = np.zeros(N)
d5 +=  $-(\phi^{**2}) * \exp(\gamma * (1. - 1./\theta)) * c_c / (\theta^{**2})$ 
Jd5 = arange (N, 2*N)
Id5 = arange (0, N)
```

o Finally, we combine diagonals in one array and form the sparse Jacobian matrix.

```
#
# Combine diagonals in one array
#
d= concatenate((d1,d2,d3,d4,d5), axis = 0)
Ir = concatenate((Id1,Id2,Id3,Id4, Id5), axis = 0)
Jc = concatenate((Jd1,Jd2,Jd3,Jd4, Jd5), axis = 0)
#
# Form sparse Jacobian matrix
#
Jac = sparse.coo_matrix((d,(Ir,Jc)),shape=(2*N,2*N))
Jac = Jac.tocsr()
#
return Jac
```


◇ Define widgets.

```
# Specify widgets
h1=HTML(value="<h4><b> Mass Transfer and First-Order Reaction in Non-
Isothermal Spherical Pellet</b>",color="brown")
h2=HTML(value = "<br> ")
h3=Latex(value="$$\mbox{Specify parameters: } $$",)
display(h1)
display(h2)
display(h3)
phi_slider = FloatSlider(min=0.5, max=3.5, step=0.5, value=2, description="Thiele modulus,
$\phi$ .....")
beta_slider = FloatSlider(min=0.1, max=1., step=0.1, value=0.5, description=r"Energy genera
tion function, $\beta$ ....")
gamma_slider = FloatSlider(min=0.1, max=2., step=0.1, value=1., description="Arrhenius numb
er, $\gamma$ .....")
Bim_slider = FloatSlider(min=10, max=500, step=10, value=100, description="Biot mass number
, $Bi_m$ .....")
Bih_slider = FloatSlider(min=20, max=500, step=10, value=100,description="Biot heat number,
$Bi_h$ .....")
w = interact(fmain,  $\phi$  = phi_slider,  $\beta$  = beta_slider,
 $\gamma$  = gamma_slider, Bi_m = Bim_slider, Bi_h = Bih_slider)
```



be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be.

Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.

© 2013 Accenture. All rights reserved.

The screenshot of widgets to specify simulation parameters is shown in Fig. 4.2.

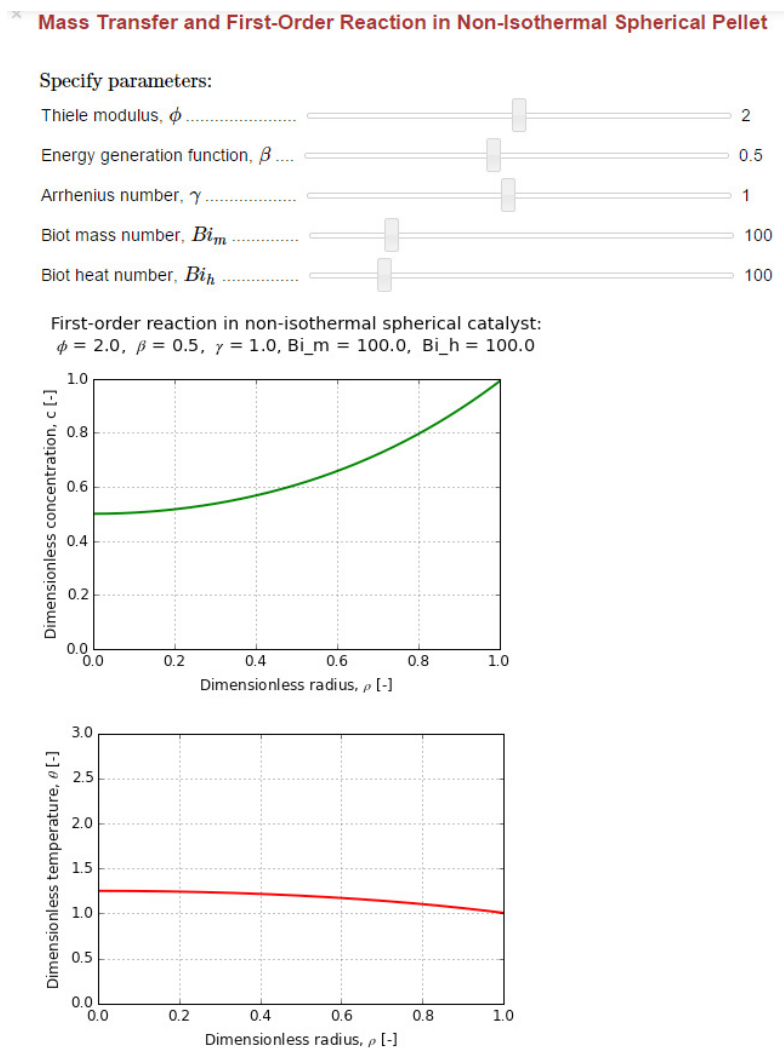


Figure 4.2: The widget to specify the values of parameters for the first-order reaction in non-isothermal pellet.

4.4 NUMERICAL RESULTS

The basic parameter values used for simulation are summarized in Table 4.1.

Parameter	Symbol	Value
Thiele modulus	ϕ	2.0
Energy generation function	β	0.5
Arrhenius number	γ	1.0
Biot number for mass transfer	Bi_m	100
Biot number for heat transfer	Bi_h	100

Table 1. Basic parameter values used for simulation

The reactant concentration and temperature profiles in the radial direction of agglomerate are shown in Fig. 4.3 for various values of Thiele modulus. These profiles were simulated using the *FirstOrder_Nonisothermal_thiele.ipynb* notebook. The reactant concentration and temperature are uniformly distributed in the pellet at low value of Thiele modulus due to the high diffusion rate of reactant as compared to the reaction rate. At high Thiele modulus, the reactant is consumed in a short distance from the outer pellet surface and its concentration drops to almost zero at the pellet surface. The pellet temperature increases with Thiele modulus and the temperature at the pellet center is 50% more than the bulk temperature at $\phi = 5.0$.

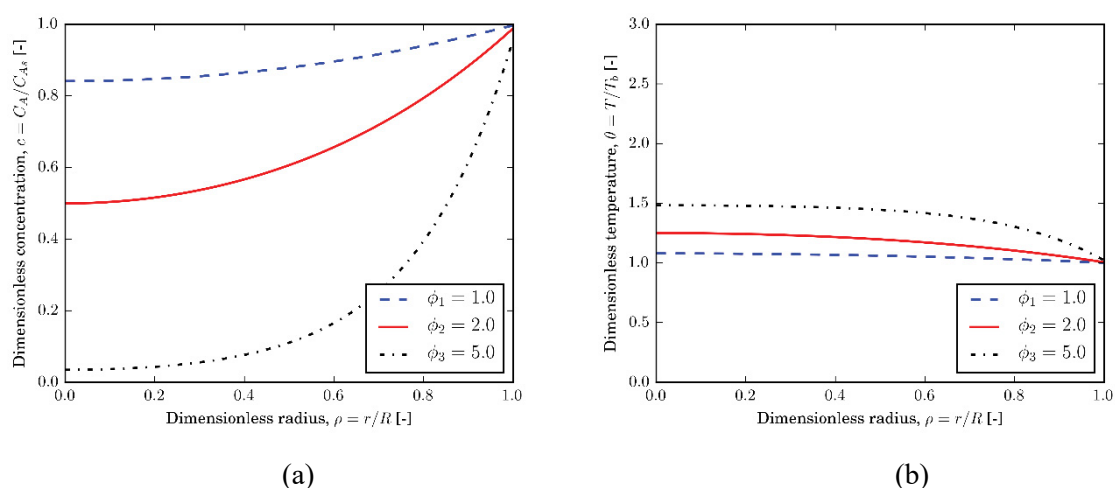


Figure 4.3: The effect of Thiele modulus on (a) concentration and (b) temperature profiles for the first-order reaction in non-isothermal pellet.

The effect of energy generation function on the temperature radial profile in the pellet is shown in Fig. 4.4. The *FirstOrder_Nonisothermal_beta.ipynb* notebook was used for simulation and plotting this figure. At high value of β , the temperature at the pellet center becomes significantly higher than that at the surface due to the low rate of heat transfer by conduction in the pellet.

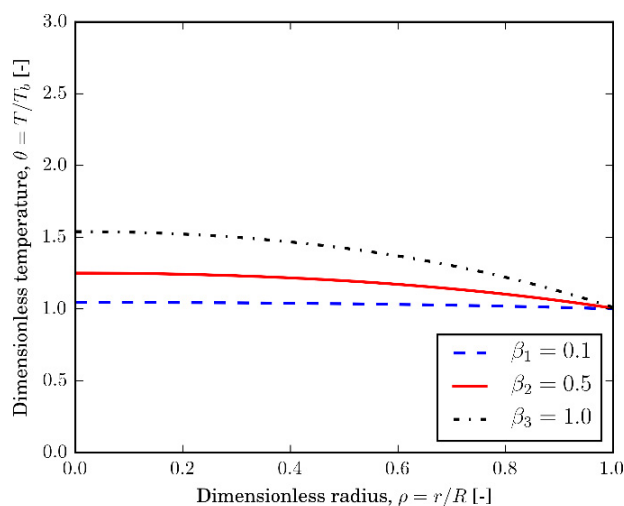


Figure 4.4: The effect of energy generation function on temperature profile for the first-order reaction in non-isothermal pellet.

"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download

Figure 4.5 shows the concentration profiles in the pellet calculated for various values of Arrhenius number using the *FirstOrder_Nonisotherma_gamma.ipynb* notebook. At high value of γ , even a small increase in temperature in the pellet as compared to the bulk temperature results in significant gain in the reaction rate because the rate constant increases exponentially with γ by Eq. (4.6). Thus, the concentration profile at $\gamma=10$ is considerably steeper than that at $\gamma=1$.

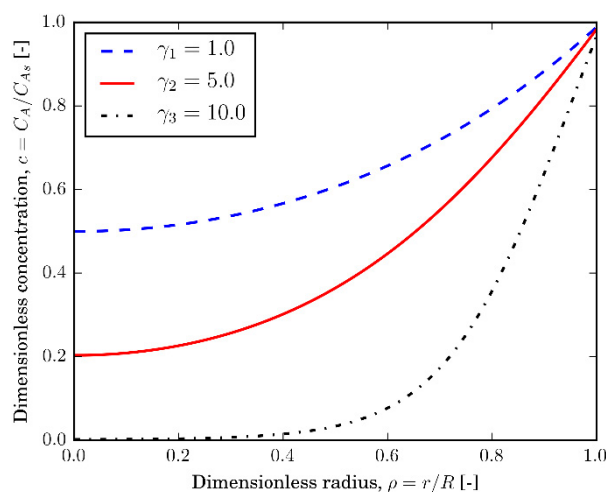


Figure 4.5: The effect of Arrhenius number on reactant concentration profile for the first-order reaction in non-isothermal pellet.

The effect of Biot number for mass transfer on reactant distribution in the pellet is illustrated in Fig. 4.6. These profiles were calculated using the *FirstOrder_Nonisothermal_Bim.ipynb* notebook. The reactant is uniformly distributed in the pellet at the small value of Bi_m because of the high diffusion rate in the pellet. However, the reactant concentration near the pellet surface is significantly lower than that in the bulk due to the low rate of external mass transfer from the bulk to the pellet surface.

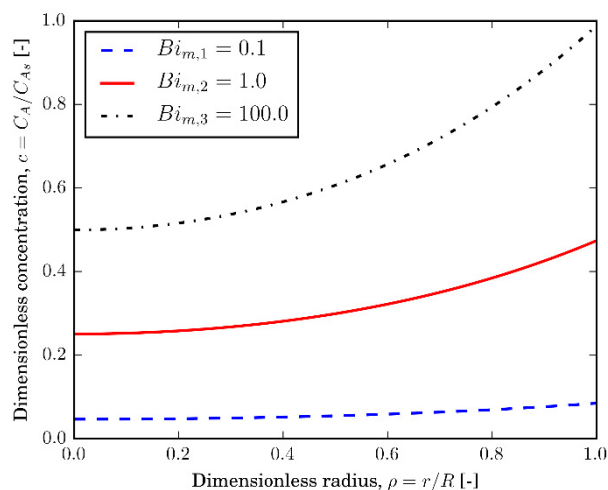


Figure 4.6: The effect of Biot number for mass transfer on concentration profile for the first-order reaction in non-isothermal pellet.

Figure 4.7 shows the temperature profiles in the pellet calculated for various values of Biot number for heat transfer using the *FirstOrder_Nonisothermal_Bih.ipynb* notebook. The pellet temperature at $Bi_h = 0.5$ is considerably higher than those at higher values of Bi_h because of the limiting heat transfer from the pellet to the bulk phase through the boundary layer around the pellet.

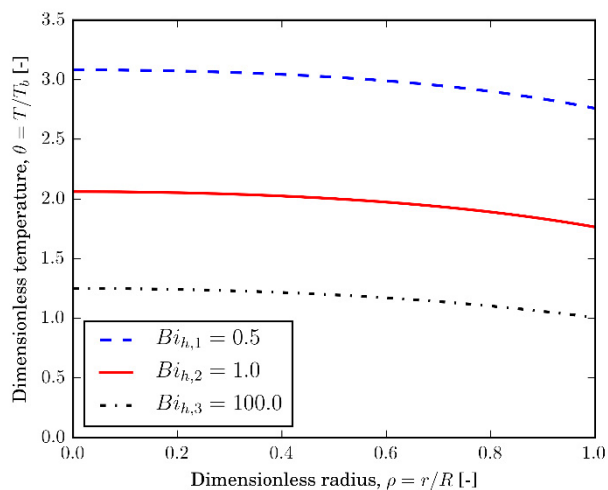


Figure 4.7: The effect of Biot number for heat transfer on temperature profile for the first-order reaction in non-isothermal pellet.

5 ENZYME CATALYZED REACTION IN ISOTHERMAL PELLET

In this chapter, you will learn to:

1. Derive the mass balance equation for enzyme catalyzed reaction in the isothermal spherical pellet.
2. Solve numerically the model equation using the finite difference method.
3. Simulate the substrate concentration profiles in the pellet for various values of process parameters using the developed IPython notebook.



The advertisement features a grey background with a faint world map. In the top left is the Duke University logo. The text 'BUSINESS HAPPENS' is prominently displayed in the center. Below it is the website 'www.fuqua.duke.edu/globalmba'. An orange button with the text 'Learn More >' is at the bottom. On the right side, there is a circular collage of six diverse business professionals' faces, with the word 'HERE.' in bold black letters in the center of the collage.

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

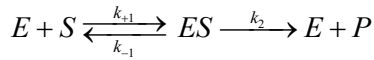
www.fuqua.duke.edu/globalmba

Learn More >

HERE.

5.1 DERIVATION OF MASS BALANCE EQUATION

We will consider an enzyme catalyzed reaction in a spherical pellet. The reaction proceeds through a fast reversible formation of an enzyme-substrate complex ES from an enzyme E and a substrate S , and a slow dissociation of the complex yielding a product P (Shuler & Kargi 2002):



The reaction rate is

$$r = \frac{dP}{dt} = k_2[ES] \quad (5.1)$$

The rate of variation of complex concentration is

$$\frac{d[ES]}{dt} = k_{+1}[E][S] - k_{-1}[ES] - k_2[ES] \quad (5.2)$$

Using a quasi-steady-state assumption, we can assume that $\frac{d[ES]}{dt} = 0$. Thus,

$$k_{+1}[E][S] = k_{-1}[ES] + k_2[ES] \quad (5.3)$$

We can formulate the conservation balance on the enzyme as

$$[E] = [E_0] - [ES], \quad (5.4)$$

where E_0 is the initial enzyme concentration.

Substituting Eq. (5.4) into Eq. (5.3), we get:

$$k_{+1}([E_0] - [ES])[S] = (k_{-1} + k_2)[ES] \quad \text{or} \quad (k_{-1} + k_2 + k_{+1}[S])[ES] = k_{+1}[E_0][S]$$

Therefore, we can express the concentration of enzyme-substrate complex as

$$[ES] = \frac{k_{+1}[E_0][S]}{k_{-1} + k_2 + k_{+1}[S]} = \frac{[E_0][S]}{\frac{k_{-1} + k_2}{k_{+1}} + [S]} \quad (5.5)$$

Substituting Eq. (5.5) into Eq. (5.1), we have:

$$r = \frac{dP}{dt} = \frac{k_2[E_0][S]}{\frac{k_{-1} + k_2}{k_{+1}} + [S]} = \frac{V_{max}[S]}{k_m + [S]}, \quad (5.6)$$

where V_{max} is the maximum rate, $V_{max} = k_2[E_0]$, and k_m is the Michaelis constant, $k_m = (k_{-1} + k_2) / k_{+1}$.

We can derive a steady-state mass balance for enzymes immobilized in a porous spherical pellet similarly to Eq. (2.5) as

$$\frac{d}{dr} \left(r^2 D_{\text{eff},S} \frac{dS}{dr} \right) - r^2 \cdot \frac{V_{\text{max}} S}{k_m + S} = 0 \quad (5.7)$$

Assuming a constant diffusivity $D_{\text{eff},S}$, Eq. (5.7) can be rewritten as

$$\frac{d^2 S}{dr^2} + \frac{2}{r} \cdot \frac{dS}{dr} - \frac{1}{D_{\text{eff},S}} \cdot \frac{V_{\text{max}} S}{k_m + S} = 0 \quad (5.8)$$

The boundary conditions are

- at the particle center, $r = 0$:

$$\frac{dS}{dr} = 0 \quad (5.9)$$

- at the outer particle surface, $r = R$:

$$S = S_0 \quad (5.10)$$

Multiplying Eq. (5.8) by R^2 / S_0 yields

$$\frac{R^2}{S_0} \cdot \frac{d^2 S}{dr^2} + \frac{2}{r} \cdot \frac{R^2}{S_0} \cdot \frac{dS}{dr} - \frac{R^2}{S_0 D_{\text{eff},S}} \cdot \frac{V_{\text{max}} S}{k_m + S} = 0 \quad (5.11)$$

The reaction term in Eq. (5.11) can be transformed as

$$\frac{R^2 / k_m}{S_0 D_{\text{eff},S} / k_m} \cdot \frac{V_{\text{max}} S}{k_m + S} = \frac{R^2 \frac{V_{\text{max}} / k_m}{D_{\text{eff},S}} \cdot \frac{S}{S_0}}{1 + \frac{S}{k_m} \cdot \frac{S_0}{S_0}}$$

Here, we introduce the dimensionless variables, concentration $s = \frac{S}{S_0}$ and radius $\rho = \frac{r}{R}$, and the dimensionless parameters, Thiele modulus $\phi = R \sqrt{\frac{V_{\text{max}} / k_m}{D_{\text{eff},S}}}$ and saturation parameter $\beta = \frac{S_0}{k_m}$. Then, we can write the reaction term as

$$\phi^2 \cdot \frac{s}{1 + \beta s}$$

Finally, we obtain the dimensionless mass balance for enzyme-catalyzed reaction in the spherical particle as

$$\frac{d^2 s}{d\rho^2} + \frac{2}{\rho} \cdot \frac{ds}{d\rho} - \phi^2 \cdot \frac{s}{1 + \beta s} = 0 \quad (5.12)$$

The boundary conditions are

- at the particle center, $\rho = 0$:

$$\frac{ds}{d\rho} = 0 \quad (5.13)$$

- at the outer particle surface, $\rho = 1$:

$$s = 1 \quad (5.14)$$

The observed rate at steady state is equal to the flux through the outer particle surface.

$$r_{\text{observed}} = \int_0^R \frac{V_{\max} S(r)}{k_m + S(r)} \cdot (4\pi r^2) dr = 4\pi R^2 D_{\text{eff},S} \left. \frac{dS}{dr} \right|_{r=R} \quad (5.15)$$

Using the dimensionless variables, we rewrite Eq. (5.15) as

$$\int_0^1 \frac{V_{\max} \cdot S_0 \cdot s(\rho)}{k_m + S_0 \cdot s(\rho)} \cdot R^3 \cdot \rho^2 d\rho = R \cdot D_{\text{eff},S} \cdot S_0 \cdot \left. \frac{ds}{d\rho} \right|_{\rho=1}$$

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](#).



Substituting the dimensionless parameters into above equation, we derive the derivative of s with respect to ρ as

$$\left. \frac{ds}{d\rho} \right|_{\rho=1} = \phi^2 \int_0^1 \frac{s}{1 + \beta \cdot s} \cdot \rho^2 d\rho \quad (5.16)$$

The intrinsic reaction rate is

$$r_{\text{intrinsic}} = \frac{4}{3} \pi R^3 \cdot \left[\frac{V_{\max} S_0}{k_m + S_0} \right] \quad (5.17)$$

The effectiveness factor is defined as the ratio of observed rate to the intrinsic reaction rate:

$$\eta = \frac{4\pi R^2 D_{\text{eff},S} \left. \frac{dS}{dr} \right|_{r=R}}{\frac{4}{3} \pi R^3 \frac{V_{\max} S_0}{k_m + S_0}}$$

Using the dimensionless variables and parameters, we can specify the effectiveness factor as

$$\eta = \frac{3(1 + \beta)}{\phi^2} \cdot \left. \frac{ds}{d\rho} \right|_{\rho=1}$$

Substituting the derivative by Eq. (5.16) into the above equation, we get:

$$\eta = 3(1 + \beta) \int_0^1 \frac{s}{1 + \beta s} \rho^2 d\rho \quad (5.18)$$

5.2 NUMERICAL IMPLEMENTATION

We use the finite-difference method to solve the second-order nonlinear differential equation. We introduce the uniformly-spaced grid $\rho_0 = 0 < \rho_1 < \dots < \rho_N < \rho_{N+1} = 1$ with internal points located at $\rho_k = k \cdot (\Delta\rho)$, $k = 1, \dots, N$, $\Delta\rho = 1/N + 1$.

The central difference approximations are used to the first and second order derivatives at ρ_k as

$$\left. \frac{ds}{d\rho} \right|_{\rho_k} = \frac{s(\rho_{k+1}) - s(\rho_{k-1}))}{2\Delta\rho}, \quad \left. \frac{d^2s}{d\rho^2} \right|_{\rho_k} = \frac{s(\rho_{k+1}) - 2s(\rho_k) + s(\rho_{k-1}))}{(\Delta\rho)^2}$$

Substitution of these approximations into the differential equation Eq. (5.12) results in

$$\frac{s(\rho_{k+1}) - 2s(\rho_k) + s(\rho_{k-1}))}{(\Delta\rho)^2} + \frac{2}{\rho_k} \cdot \frac{s(\rho_{k+1}) - s(\rho_{k-1}))}{2\Delta\rho} - \phi^2 \cdot \frac{s(\rho_k)}{1 + \beta s(\rho_k)} = 0 \quad (5.19)$$

Rearrangement of Eq. (5.19) yields

$$\left(\frac{1}{(\Delta\rho)^2} - \frac{1}{\rho_k \Delta\rho}\right) \cdot s(\rho_{k-1}) + \left(-\frac{2}{(\Delta\rho)^2}\right) \cdot s(\rho_k) + \left(\frac{1}{(\Delta\rho)^2} + \frac{1}{\rho_k \Delta\rho}\right) \cdot s(\rho_{k+1}) - \phi^2 \cdot \frac{s(\rho_k)}{1 + \beta s(\rho_k)} = 0$$

Simplifying above equations, we get:

$$A_k \cdot s(\rho_{k-1}) + B \cdot s(\rho_k) + C_k \cdot s(\rho_{k+1}) - \phi^2 \cdot \frac{s(\rho_k)}{1 + \beta s(\rho_k)} = 0, \quad (5.20)$$

where

$$A_k = \frac{1}{(\Delta\rho)^2} - \frac{1}{\rho_k \Delta\rho}, \quad B = -\frac{2}{(\Delta\rho)^2}, \quad C_k = \frac{1}{(\Delta\rho)^2} + \frac{1}{\rho_k \Delta\rho}$$

The discretized equation for the boundary condition at $\rho_0 = 0$ by Eq. (5.13) is

$$\frac{-3s(\rho_0) + 4s(\rho_1) - s(\rho_2)}{2\Delta\rho} = 0 \quad (5.21)$$

Thus, the concentration at ρ_0 is given as

$$s(\rho_0) = -\frac{1}{3}s(\rho_2) + \frac{4}{3}s(\rho_1) \quad (5.22)$$

Introducing $s(\rho_0)$ by Eq. (5.22) into Eq. (5.20) for $k=1$, we obtain:

$$A_1 \cdot \left[-\frac{1}{3}s(\rho_2) + \frac{4}{3}s(\rho_1)\right] + B \cdot s(\rho_1) + C_1 \cdot s(\rho_2) - \phi^2 \cdot \frac{s(\rho_1)}{1 + \beta s(\rho_1)} = 0 \quad (5.23)$$

Rearrangement of Eq. (5.23) results in

$$\left[\frac{2}{3}B + A_1 - \frac{1}{3}C_1\right] \cdot s(\rho_1) + \left[C_1 - \frac{1}{3}A_1\right] \cdot s(\rho_2) - \phi^2 \cdot \frac{s(\rho_1)}{1 + \beta s(\rho_1)} = 0 \quad (5.24)$$

The boundary condition at ρ_{N+1} by Eq. (5.14) is

$$s(\rho_{N+1}) = 1 \quad (5.25)$$

Introducing $s(\rho_{N+1})$ into Eq. (5.20) for $k=N$ yields

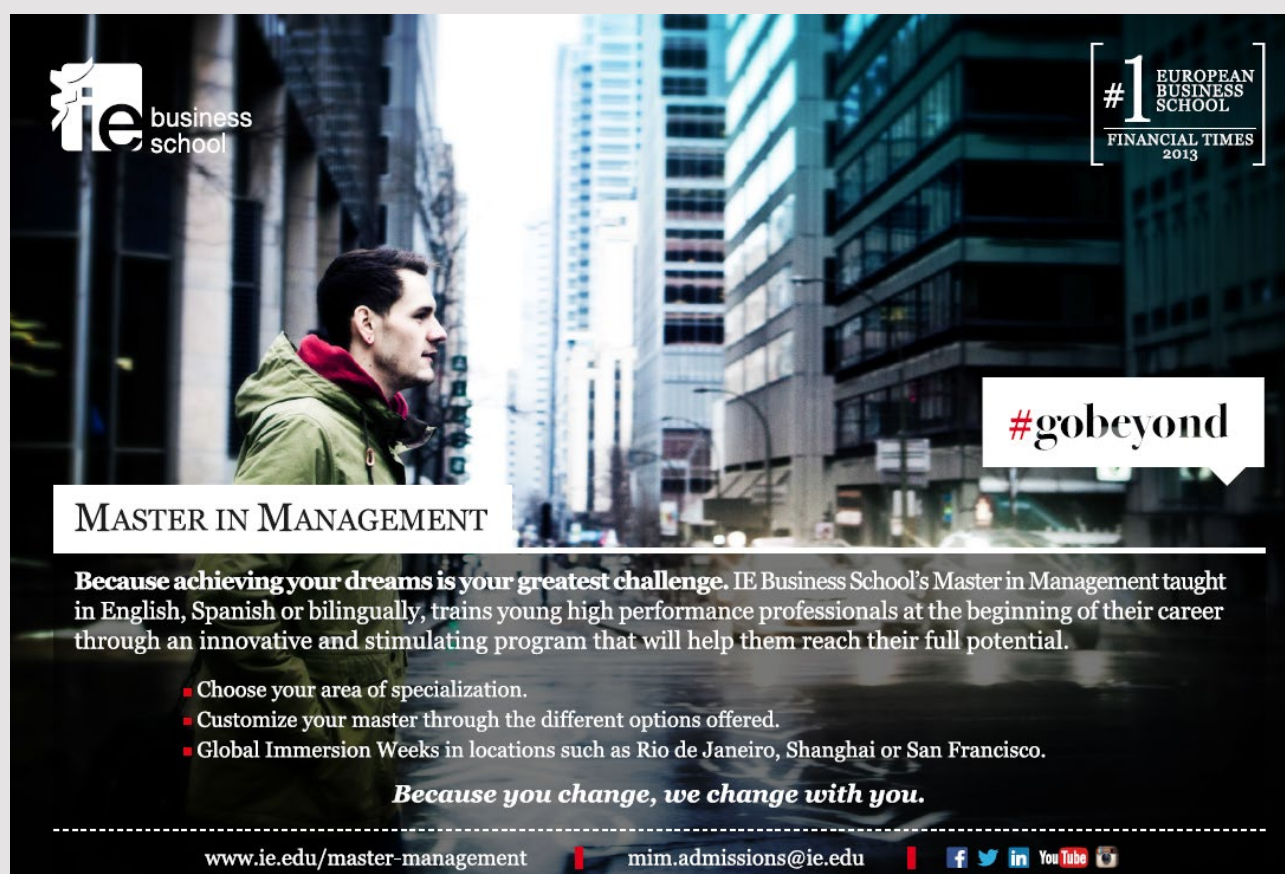
$$A_N \cdot s(\rho_{N-1}) + B \cdot s(\rho_N) + C_N - \phi^2 \cdot \frac{s(\rho_N)}{1 + \beta s(\rho_N)} = 0 \quad (5.26)$$

Rearrangement of Eq. (5.26) results in

$$s(\rho_{N-1}) \cdot A_N + s(\rho_N) \cdot B + C_N - \phi^2 \cdot \frac{s(\rho_N)}{1 + \beta s(\rho_N)} = 0, \quad (5.27)$$

Therefore, we obtain the following system of nonlinear algebraic equations:

$$\begin{cases} f(1) = \left[\frac{2}{3}B + A_1 - \frac{1}{3}C_1 \right] \cdot s(\rho_1) + \left[C_1 - \frac{1}{3}A_1 \right] \cdot s(\rho_2) - \phi^2 \cdot \frac{s(\rho_1)}{1 + \beta s(\rho_1)} = 0 \\ f(k) = A_k \cdot s(\rho_{k-1}) + B \cdot s(\rho_k) + C_k \cdot s(\rho_{k+1}) - \phi^2 \cdot \frac{s(\rho_k)}{1 + \beta s(\rho_k)} = 0, \quad k = 2, \dots, N-1 \\ f(N) = A_N \cdot s(\rho_{N-1}) + B \cdot s(\rho_N) + C_N - \phi^2 \cdot \frac{s(\rho_N)}{1 + \beta s(\rho_N)} = 0 \end{cases} \quad (5.28)$$



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu | f t in YouTube

The structure of the Jacobian matrix is shown in Fig. 5.1, where **1**, **2** and **3** are the matrix diagonals.

$J =$

	$j=1$									N
$i=1$	1	3	0	0	0	0	0	0	0	
	2	1	3	0	0	0	0	0	0	
	0	2	1	3	0	0	0	0	0	
	0	0	2	1	3	0	0	0	0	
	0	0	0	2	1	3	0	0	0	
	0	0	0	0	2	1	3	0	0	
	0	0	0	0	0	2	1	3	0	
	0	0	0	0	0	0	2	1	3	
	0	0	0	0	0	0	0	2	1	3
N	0	0	0	0		0	0	0	2	1

Figure 5.1: The structure of tridiagonal Jacobian matrix.

The diagonal elements of the Jacobian matrix are summarized below:

- main diagonal elements J_{d1}

$$J(1,1) = \left[\frac{2}{3}B + A_1 - \frac{1}{3}C_1 \right] - \phi^2 \cdot \frac{1}{(1 + \beta s(\rho_1))^2}$$

$$J(i,i) = B - \phi^2 \cdot \frac{1}{(1 + \beta s(\rho_i))^2}, \quad i = 2, \dots, N-1$$

$$J(N,N) = B - \phi^2 \cdot \frac{1}{(1 + \beta s(\rho_N))^2}$$

- lower diagonal elements J_{d2}

$$J(i,i-1) = A_i, \quad i = 2, \dots, N-1$$

$$J(N,N-1) = A_N$$

- upper diagonal elements J_{d3}

$$J(1,2) = C_1 - \frac{1}{3}A_1$$

$$J(i,i+1) = C_i, \quad i = 2, \dots, N-1$$

5.3 COMPUTER PROGRAM DESCRIPTION AND NUMERICAL RESULTS

Notebook *Enzyme_Concentration.ipynb*

◇ Import packages.

We will use the Python library *NumPy* for vector manipulations and formation of diagonal matrices. The sparse module from the library *SciPy* is called to generate a sparse matrix and convert it to various formats as well as to compute the vector norm. The *spsolve* module from the *SciPy* library is utilized to solve the sparse linear system and the *trapz* module to integrate function using the composite trapezoidal rule. To plot the results of numerical simulations, we will use the *matplotlib* library. The *IPython.html* module is called to display the widgets for interactive input of parameter values.

```
%matplotlib inline
import numpy as np
from scipy.sparse.linalg import spsolve
from scipy.linalg import norm
from scipy import sparse
from scipy.integrate import trapz
from matplotlib.pyplot import *
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
```

◇ Define the main function *fmain*:

o Input parameters:

ϕ – Thiele modulus

β – saturation parameter

```
# -----
#
# Main function
#
def fmain ( $\phi$ ,  $\beta$ , **kwargs):
```


o Specify the grid.

The number of internal grid points is N . Thus, the total number of equally spaced points including the boundary ones is $N+2$. The grid is divided on $N+1$ subintervals of the same length $\Delta\rho$. We generate the grid of internal points and saved it in the array ρ .

```
#-----
# The number of internal grid points
N = 299
#-----
# set grid spacing
Δρ = 1./(N + 1.)
# specify uniform grid of internal points
ρ = np.linspace(Δρ, 1.-Δρ, N)
```

SMS from your computer

...Sync'd with your Android phone & number

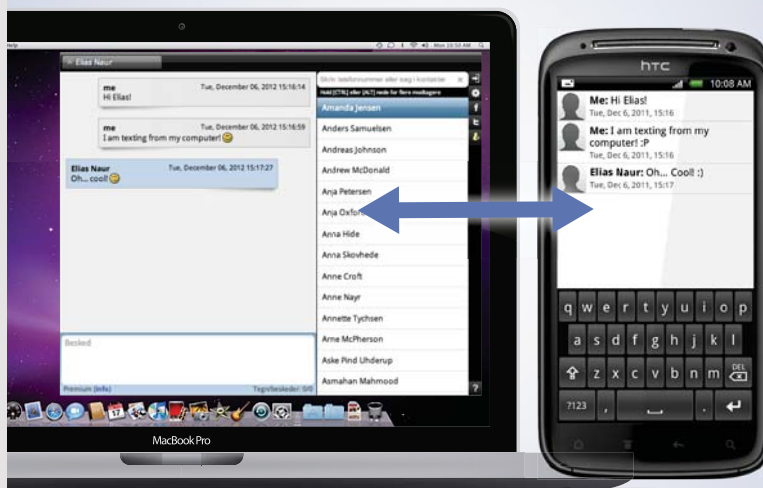
FREE
30 days trial!!

Go to

BrowserTexting.com

and start texting from
your computer!

 **BrowserTexting**



- o Specify common coefficients in the system of nonlinear equations.

```
# allocate arrays of common coefficients A and C
A = np.zeros(N)
C = np.zeros(N)
# define constant common coefficient B
B = - 2./(\Delta\rho**2)
# define coefficients A and B at each mesh point
A = 1./(\Delta\rho**2) - 1./(\Delta\rho*\rho)
C = 1./(\Delta\rho**2) + 1./(\Delta\rho*\rho)
```

- o Set initial guesses for solution of nonlinear equations.
We will use the uniform profile, $s_0=1$, as the initial guess.

```
#####
# specify initial guess for solution of nonlinear equations
# we will use uniform profiles: s0=1
s0 = np.ones(N)
```

- o Use the Newton-Raphson method to solve the system of nonlinear equations.

```
#
# solve the system of nonlinear equations using the Newton-Raphson method
#
u,f_val,iter, exitflag = fsolve(s0, \phi, \beta, A, B, C)
#
if not exitflag:
    print ('Too many iterations')
else:
    print ('Successful solution: the number of iterations is %s'%iter)
```

- o Save results.

Save the resulting profiles at internal points in arrays *concentration* and *temperature*. Calculate the concentration at the pellet center using the second-order forward difference formula to approximate the derivative in the boundary condition at $\rho=0$. The concentration at the outer boundary is constant and equal to 1.

```
#
# save resulting concentration profile
concentration = np.zeros(N+2)
concentration[1:N+1] = u[0:N]
# add concentration value at \rho = 0
concentration[0] = 4.*u[0]/3. - u[1]/3.
# add concentration value at \rho = 1
concentration[N + 1] = 1.
```

- o Then, we plot the concentration profile.

```
#
# plot graphs
rad = np.linspace(0.0, 1.0, N + 2)
matplotlib.rcParams.update({'font.size': 12})
fig = plt.figure()
axes = fig.add_subplot(111)
axes.plot(rad, concentration, color="green", linestyle="solid", linewidth=2)
title1 = 'Enzyme catalyzed reaction in spherical catalyst: \n'
title1 += '$\phi$ = %s, $\beta$ = %s '
axes.set_title(title1 % ( $\phi$ ,  $\beta$ ))
axes.set_xlabel(r'Dimensionless radius, $\rho$ [-]')
axes.set_ylabel('Dimensionless concentration, s [-]')
axes.grid()
axis([0., 1., 0.0, 1.0])
plt.show()
```

- o Calculate the effectiveness factor.

```
#
# calculate effectiveness factor
integrand = np.zeros(N+2)
integrand = concentration * (rad**2) / (1 +  $\beta$ *concentration)
 $\eta$  = 3*(1+ $\beta$ )*trapz(integrand, rad)
print (' Effectiveness factor:  $\eta$  = %.3f ' %  $\eta$ )
```

- ◇ Solve the system of nonlinear equations by the Newton-Raphson method.

```
#-----
#
# Solve the system of nonlinear equations using Newton-Raphson
# method with tridiagonal Jacobian.
#
def fsolve(x0,  $\phi$ ,  $\beta$ , A, B, C, tolX = 1e-8, tolG = 1e-8, maxIter = 100):
#
    x = x0
    for i in range (maxIter):
        gx = g(x,  $\phi$ ,  $\beta$ , A, B, C)
        xNew = x - spsolve (J(x,  $\phi$ ,  $\beta$ , A, B, C), gx )
        if (norm(xNew-x) < tolX):
            return [x,gx,i,'True']
        if (norm(gx) < tolG):
            return [xNew,gx,i,'True']
        x = xNew
    return [x, gx, i, 'False']
```

- ◇ Define the system of nonlinear equations.
 - The function g :

```
#
# Define the system of nonlinear equations
#
def g (s,  $\phi$ ,  $\beta$ , A, B, C):
```

- At first, we allocate the array f .

```
# allocate arrays
N = len(s)
f = np.zeros(N)
```

- Then, we specify the reaction term in the mass balance equations.

```
*****
# reaction term
f[0:N] -= ( $\phi$ **2) * s[0:N] / (1. +  $\beta$ *s[0:N])
*****
```

The Wake

the only emission we want to leave behind

Low-speed Engines Medium-speed Engines Turbochargers Propellers Propulsion Packages PrimeServ

The design of eco-friendly marine power and propulsion solutions is crucial for MAN Diesel & Turbo. Power competencies are offered with the world's largest engine programme – having outputs spanning from 450 to 87,220 kW per engine. Get up front! Find out more at www.mandieselturbo.com

Engineering the Future – since 1758.

MAN Diesel & Turbo



- o Next, we specify the discrete balance equations at the internal points.

```
# pellet internal points
for k in range (1,(N-1)):
    f[k] += A[k]*s[k-1] + B * s[k] + C[k]*s[k+1]
```

- o Finally, we define the balance equations at the center and surface of the pellet.

```
# pellet center
f[0] += (2.*B/3. + A[0] - C[0]/3.)*s[0] + (C[0]-A[0]/3.)*s[1]
# pellet surface
f[N-1] += A[N-1]*s[N-2] + B * s[N-1] + C[N-1]
#
return f
```

- ◇ Define the Jacobian matrix.

- o The function *jac*:

```
#
#-----
# Define Jacobian matrix
#
def J (s,  $\phi$ ,  $\beta$ , A, B, C):
```

- o At first, we form the main diagonal d1.

```
#
# Main Diagonal d1
#
N = len(s)
d1 = np.zeros(N)
#####
# derivative of reaction rate with respect to concentration
d1[0:N] += -( $\phi^2$ )/((1. +  $\beta$ *s[0:N])**2)
#####
# pellet internal points
for k in range (1,(N-1)):
    d1[k] += B
# pellet center
d1[0] += A[0] + 2.*B/3. - C[0]/3.
# pellet surface
d1[N-1] += B
# position of non-zero elements
Jd1 = arange(0, N)
Id1 = arange(0, N)
```

- o Then, we specify the lower diagonal d2.

```
#
# Lower Diagonal d2
#
    d2 = np.zeros (N-1)
# pellet internal points
    d2[0:N-2] = A[1:N-1]
# pellet surface
    d2[N-2] = A[N-1]
# position of non-zero elements
    Id2 = arange(1, N)
    Jd2 = arange(0, N-1)
```

- o Finally, we specify the upper diagonal d3.

```
# Upper Diagonal d3
#
    d3 = np.zeros (N-1)
# pellet center
    d3[0] = C[0] - A[0]/3.
# pellet internal points
    d3[1:N-1] = C[1:N-1]
# position of non-zero elements
    Jd3 = arange(1, N)
    Id3 = arange(0, N-1)
```

- o Then, we combine diagonals in one array and form the sparse Jacobian matrix.

```
#
# Combine diagonals in one array
#
    d= concatenate((d1,d2,d3), axis = 0)
    Ir = concatenate((Id1,Id2,Id3), axis = 0)
    Jc = concatenate((Jd1,Jd2,Jd3), axis = 0)
#
# Form sparse Jacobian matrix
#
    Jac = sparse.coo_matrix((d,(Ir,Jc)),shape=(N,N))
    Jac = Jac.tocsr()
#
    return Jac
```

◇ Define widgets.

```
#  
# Specify widgets  
h1=HTML(value="<h4><b> Mass Transfer and Enzyme Catalyzed Reaction in Isothermal Spherical  
Pellet</b>",color="brown")  
h2=HTML(value = "<br> ")  
h3=Latex(value="$$\mbox{Specify parameters: } $$",)  
display(h1)  
display(h2)  
display(h3)  
Phi_slider = FloatSlider(min=0, max=20, step=0.5, value=10)  
beta_slider = FloatSlider(min=0, max=1, step=0.05, value=0.5)  
w = interact(fmain,  $\phi$  = Phi_slider,  $\beta$  = beta_slider)
```

TURN TO THE EXPERTS FOR **SUBSCRIPTION** CONSULTANCY

Subscribe is one of the leading companies in Europe when it comes to innovation and business development within subscription businesses.

We innovate new subscription business models or improve existing ones. We do business reviews of existing subscription businesses and we develop acquisition and retention strategies.

Learn more at [linkedin.com/company/subscribe](https://www.linkedin.com/company/subscribe) or contact
Managing Director Morten Suhr Hansen at mha@subscribe.dk

SUBSCR✓**IBE** - to the future

The screenshot of widgets to specify parameters is shown in Fig. 5.2.

× **Mass Transfer and Enzyme Catalyzed Reaction in Isothermal Spherical Pellet**

Specify parameters:



Successful solution: the number of iterations is 5

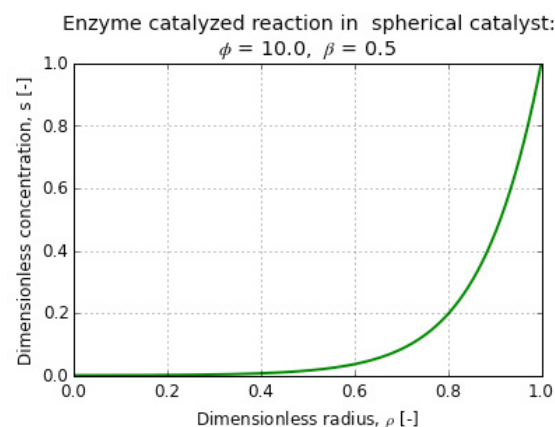


Figure 5.2: The screenshot of widget to specify parameters for the enzyme catalyzed reaction in isothermal pellet.

The dimensionless substrate profiles in the spherical immobilized particle are shown in Fig. 5.3 for various values of Thiele modulus. This plot was calculated using the *Enzyme_Concentration_thiele.ipynb* notebook. The effect of saturation parameter on the concentration profile is illustrated in Fig. 5.4. These profiles were calculated using the *Enzyme_Concentration_beta.ipynb* notebook. The uniform substrate profile is obtained at low Thiele modulus and high value of saturation parameter.

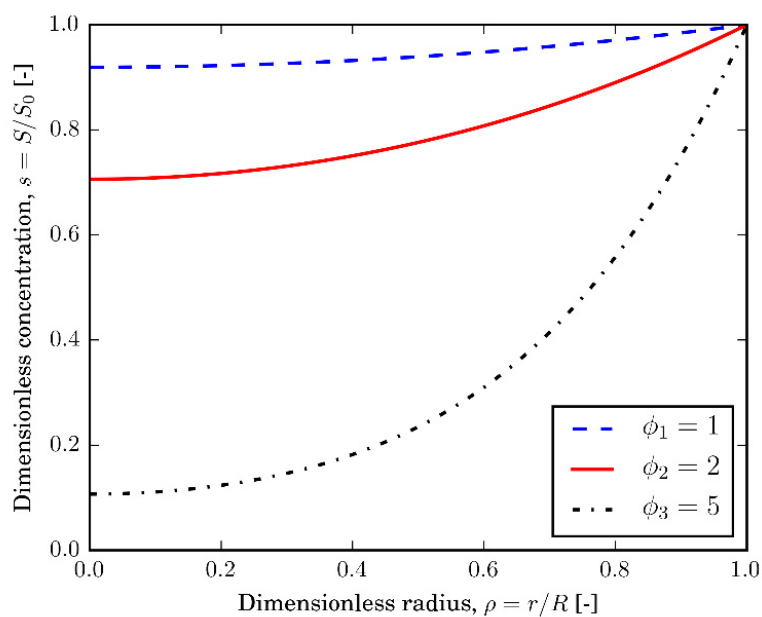


Figure 5.3: Substrate profiles in the spherical immobilized particle for various values of Thiele modulus, $\beta = 1$.

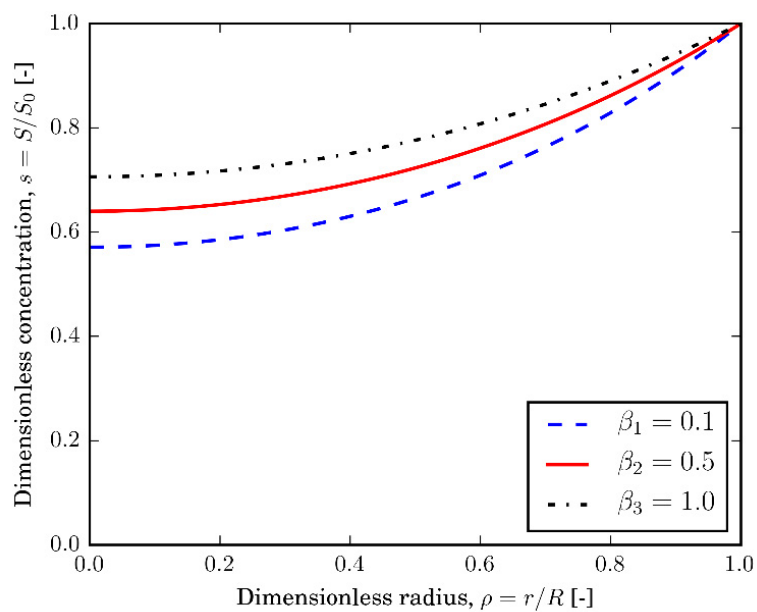
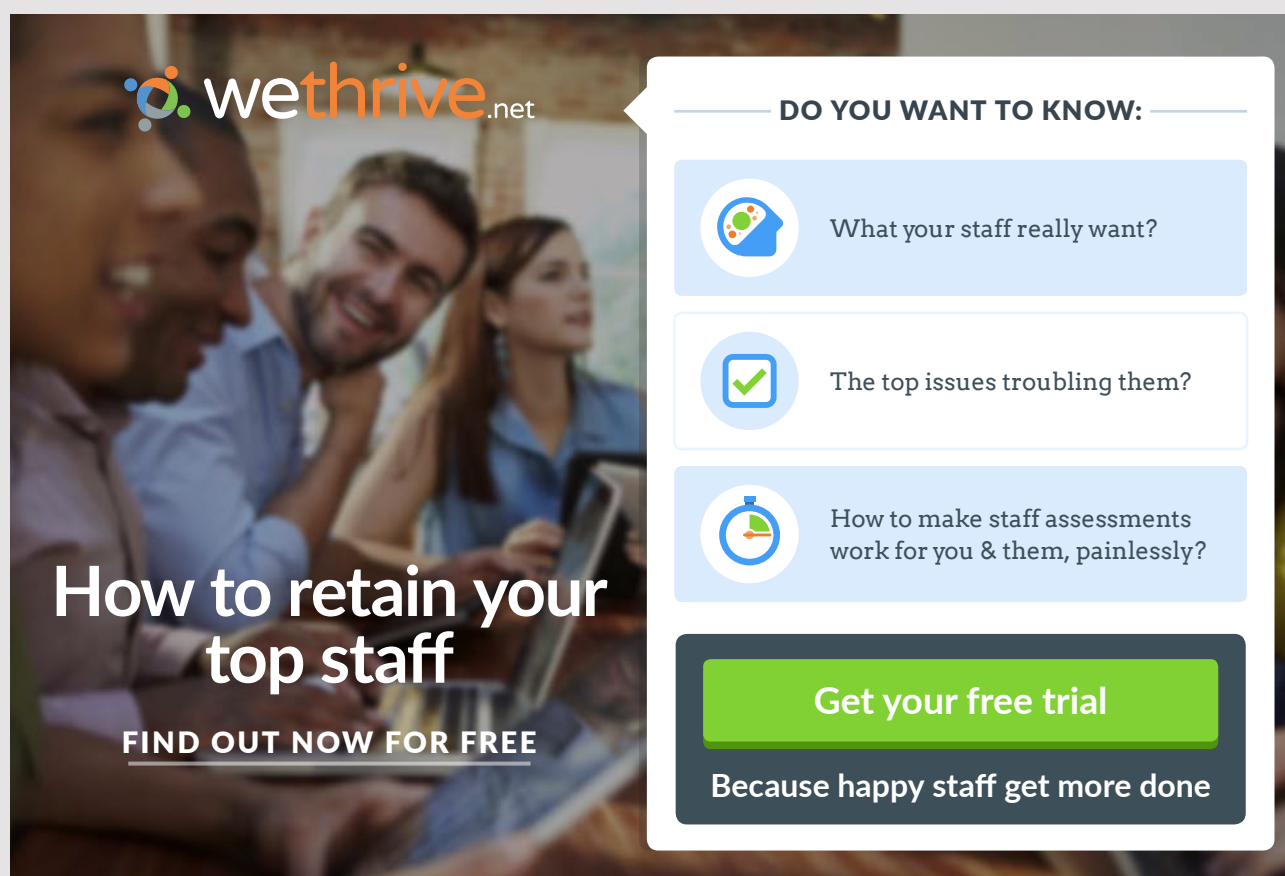


Figure 5.4: Substrate profiles in the spherical immobilized particle for various values of β parameter, $\phi = 2$.

6 NON-CATALYTIC CHEMICAL REACTION IN AGGLOMERATE OF FINE PARTICLES

In this chapter, you will learn to:

1. Derive the model equations for the non-catalytic chemical reaction in the agglomerate with porous structure that evolves during the reaction.
2. Solve the model equations numerically using the method of lines.
3. Simulate and plot the solid deposit and gaseous reactant concentration profiles for the chemical vapor deposition reaction in the agglomerate of fine particles using the elaborated IPython notebooks.
4. Study the effects of process parameters on the uniformity of deposit distribution in the agglomerate.



wethrive.net

How to retain your top staff

FIND OUT NOW FOR FREE

DO YOU WANT TO KNOW:

- What your staff really want?
- The top issues troubling them?
- How to make staff assessments work for you & them, painlessly?

Get your free trial

Because happy staff get more done

6.1 DERIVATION OF MATHEMATICAL MODEL EQUATIONS

We will consider the chemical reaction in a porous agglomerate made of primary fine particles. The gaseous reactant diffuses from the bulk phase to the outer agglomerate surface and then from the surface to the agglomerate center. The reaction takes place on the surface of primary particles. The chemical vapor deposition (Golman & Shinohara 2000) and chemical vapor infiltration (Xu & Yan 2010) are examples of such reactions. The mechanism of these processes is quite complex. It is assumed that the reactant species react in the gas phase generating the gaseous precursors. Then, these precursors nucleate and grow to form the deposit on the surface of primary particles. The amount of deposit increases with the reaction time, resulting in a decrease of pore space among primary particles. Another example of non-catalytic reaction is the sulfation of lime, which is used to control the sulfur emissions in the fluidized bed combustion of coal (Jeong et al. 2015). At high temperatures, the limestone calcines to form lime that reacts with sulfur dioxide released by the coal. The solid reaction product, calcium sulfate, deposits in the porous CaO particles. As the molar volume of calcium sulfate is larger than that of calcium oxide, the pore space in the CaO particles is gradually filled with calcium sulfate.

Therefore, we have to take into account both the temporal and spatial variations of effective transport properties in the radial direction of the agglomerate in deriving the mathematical model. These variations should be related to the changes in agglomerate structural properties, such as voidage and specific surface area, which in turn should be connected to the rate of the reaction front advancing on the surface of primary particles.

Here, we will derive the mathematical model for the chemical vapor deposition reaction in the agglomerate of fine particles. We assume that the gas-phase reaction is in steady-state at all times. We also neglect the temperature gradient in the agglomerate.

We first write the mass balance equation of i gaseous species in the spherical agglomerate as:

$$\frac{1}{R^2} \frac{\partial}{\partial R} \left(D_{\text{eff},i} R^2 \frac{\partial C_i}{\partial R} \right) - S \cdot R_{S_i} = 0 \quad (6.1)$$

where C_i and $D_{\text{eff},i}$ are the concentration and effective diffusivity of i species at the radial position R and time t in the porous structure of the agglomerate, respectively, S is the accessible internal surface area and R_{S_i} is the reaction rate per unit area.

Then, we introduce the conservation equation for the solid product in terms of the varying radius of the reaction surface of fine particles, r , as follows:

$$\frac{\partial r}{\partial t} = \frac{b}{a} \cdot \nu_s \cdot R_{S_i} \quad (6.2)$$

where $\nu_s = M_s / \rho_s$ is the solid molar volume, and M_s and ρ_s are the molecular weight and density of solid product, respectively. Coefficients a and b represent the stoichiometric coefficients for the reacting species and the solid product. Equations (6.1) and (6.2) are subject to the following initial and boundary conditions:

$$\frac{\partial C_i}{\partial R} = 0 \quad \text{at} \quad R = 0 \quad (6.3)$$

$$C_i = C_{i,b} \quad \text{at} \quad R = R_0 \quad (6.4)$$

$$r = r_0 \quad \text{at} \quad t = 0 \quad \text{and} \quad 0 < R < R_0 \quad (6.5)$$

where r_0 is the initial radius of fine particles, R_0 is the agglomerate radius and $C_{i,b}$ is the bulk concentration of growth species near the external surface of the agglomerate.

Here, we assume that the surface reaction kinetics follows a first-order rate law and the reaction is irreversible:

$$R_{S_i} = k \cdot c_i \quad (6.6)$$

The specific reaction rate constant k is expressed as a function of the reaction temperature, T , according to the Arrhenius equation:

$$k = k_0 \exp \left[\gamma \cdot \left(1 - \frac{T_0}{T} \right) \right] \quad (6.7)$$

where $\gamma = E / R_g T_0$ is the Arrhenius parameter, E is the activation energy, k_0 is the reaction rate constant at the reference temperature T_0 , and R_g is the ideal gas law constant.

We will use the modified random overlapping grain model, previously developed by the author (Golman & Shinohara 1998), to describe the evolution of porous structure of the agglomerate as the reaction progresses. The agglomerate prior to reaction is represented as a population of randomly distributed grains of initially uniform size. As the reaction proceeds, the solid product deposits on the surfaces of the grains and the grains begin to grow.

The variation of local void fraction in the agglomerate with reaction progress is given as:

$$\varepsilon = \varepsilon_0 \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} \left(\left(\frac{r}{r_0} \right)^3 - 1 \right) \right] \quad (6.8)$$

The relationship among the surface area, the void fraction and the grain size is as follows:

$$\frac{S}{S_0} = \frac{\varepsilon}{\varepsilon_0} \cdot \left(\frac{r}{r_0} \right)^2 \quad (6.9)$$

The effective diffusivity of the i species, $D_{\text{eff},i}$, can be estimated by Eq. (1.11). The mean radius of capillary used in the calculation of Knudsen diffusivity by Eq. (1.5) is evaluated on the basis of the random overlapping structural model by Eq. (6.9) as:

$$\bar{a} = \frac{2\varepsilon_0}{S_0} \cdot \left(\frac{r_0}{r} \right)^2 \quad (6.10)$$

Combining Eqs. (6.6)–(6.9), we rewrite Eqs. (6.1) and (6.2) in dimensionless form as:


$$\frac{1}{\rho^2} \frac{\partial}{\partial \rho} \left(D_{\text{eff},i}^* \rho^2 \frac{\partial c_i}{\partial \rho} \right) = \phi_0^2 c_i \xi^2 \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi^3 - 1) \right] \quad (6.11)$$


$$\frac{\partial \xi}{\partial \tau} = c_i \quad (6.12)$$


Struggling to get interviews?

Professional CV consulting & writing assistance from leading job experts in the UK.


Visit site







Take a short-cut to your next job!
Improve your interview success rate by 70%.



TheCVagency

Visit thecvagency.co.uk for more info.

where $\rho = \frac{R}{R_0}$, $c_i = \frac{C_i}{C_{i,b}}$, $\xi = \frac{r}{r_0}$, $\tau = \frac{tbv_s k C_{i,b}}{ar_0}$ are the dimensionless agglomerate radial coordinate, the growth species concentration, the reaction radius of the primary particles and the reaction time, respectively.

The definitions of characteristic dimensionless parameters are summarized as follows:

o normalized effective diffusivity, $D_{\text{eff},i}^*$, and specific surface area, s ,

$$D_{\text{eff},i}^* = \frac{D_{\text{eff},i}}{D_{\text{eff},i}^0}, s = \frac{S}{S_0} \quad (6.13)$$

o initial Thiele modulus, ϕ_0 , at $t=0$

$$\phi_0 = R_0 \sqrt{\frac{kS_0}{D_{\text{eff},i}^0}} \quad (6.14)$$

The voidage is calculated from Eq. (6.8) as

$$\varepsilon = \varepsilon_0 \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi^3 - 1) \right] \quad (6.15)$$

and the normalized effective diffusivity, $D_{\text{eff},i}^*$, is estimated from Eqs. (1.1) – (1.11) as

$$D_{\text{eff},i}^* = \left(\frac{\varepsilon}{\varepsilon_0} \right)^2 (\rho)^{-2} \left(\frac{T}{T_0} \right)^2 \left(\frac{\Omega_{mi}^0}{\Omega_{mi}} \right) \left(\frac{D_{Ki}^0 + D_{mi}^0}{D_{Ki} + D_{mi}} \right) \quad (6.16)$$

The corresponding initial and boundary conditions are

$$\xi = 1, c_i = 1 \quad \text{at} \quad \tau = 0 \quad \text{for all} \quad \rho \quad (6.17)$$

$$c_i = 1 \quad \text{at} \quad \rho = 1 \quad \text{for all} \quad \tau \geq 0 \quad (6.18)$$

$$\frac{dc_i}{d\rho} = 0 \quad \text{at} \quad \rho = 0 \quad \text{for all} \quad \tau \geq 0 \quad (6.19)$$

The local normalized mass concentration of deposit, w , is defined as

$$w = \frac{W_s}{W_{s,\text{max}}},$$

where W_s is the mass of solid product per unit agglomerate volume, $W_{s,\text{max}} = \rho_s \cdot \varepsilon$ is the maximum value of W_s corresponding to the complete filling of void spaces with solid product. Thus, using Eq. (6.15), we can evaluate w as

$$w = \frac{\varepsilon_0 - \varepsilon}{\varepsilon_0} = 1 - \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi^3 - 1) \right] \quad (6.20)$$

6.2 COMPUTATIONAL PROCEDURE USING THE METHOD OF LINES

The mass balance equations for the agglomerate consist of a system of the second-order ODEs of the boundary value type for the radial distribution of gas precursor by Eq. (6.11) and the first-order ODE for the rate of growth of reaction interface on the primary particles due to the solid deposition by Eq. (6.12).

We will use the orthogonal collocation method for the solution of second-order differential Eq. (6.11) for the agglomerate. Using the transformation $x = \rho^2$, we can rewrite Eq. (6.11) similar to Eq. (3.6) as

$$x \frac{d^2 c_i}{dx^2} + \frac{3}{2} \frac{dc_i}{dx} = \frac{\phi_0^2}{4D_{\text{eff},i}} \cdot c_i \xi^2 \cdot \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi^3 - 1) \right] \quad (6.21)$$

The boundary condition is

$$c_i = 1 \quad \text{at} \quad x = 1 \quad (6.22)$$

The application of the orthogonal collocation method to Eq. (6.21) results in the following algebraic equation at the j interior collocation point:

$$x_j \sum_{k=1}^{N+1} B_{jk} \cdot c_{i,k} + \frac{3}{2} \sum_{k=1}^{N+1} A_{jk} \cdot c_{i,k} = \frac{\phi_0^2}{4D_{\text{eff},i}^*(j)} \cdot c_{i,j} \cdot \xi_j^2 \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi_j^3 - 1) \right] \quad (6.23)$$

The boundary condition is

$$c_{i,N+1} = c_i|_{x=1} = 1 \quad (6.24)$$

Equation (6.23) can be rewritten using the above boundary condition as:

$$x_j \sum_{k=1}^N B_{jk} \cdot c_{i,k} + \frac{3}{2} \sum_{k=1}^N A_{jk} \cdot c_{i,k} - \frac{\phi_0^2}{4D_{\text{eff},i}^*(j)} \cdot c_{i,j} \cdot \xi_j^2 \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi_j^3 - 1) \right] = - \left(x_j B_{jN+1} + \frac{3}{2} A_{jN+1} \right) \quad (6.25)$$

Equation (6.25) can be represented in a matrix form as follows:

$$\mathbf{D} \cdot \mathbf{c} = \mathbf{F} \quad (6.26)$$

where

$$F_j = - \left(x_j B_{jN+1} + \frac{3}{2} A_{jN+1} \right)$$

and

$$D = \begin{bmatrix} D_{11} & x_1 B_{12} + \frac{3}{2} A_{12} & \cdots & x_1 B_{1N} + \frac{3}{2} A_{1N} \\ x_2 B_{2N} + \frac{3}{2} A_{2N} & D_{22} & \cdots & x_2 B_{2N} + \frac{3}{2} A_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ x_N B_{N1} + \frac{3}{2} A_{N1} & x_N B_{N2} + \frac{3}{2} A_{N2} & \cdots & D_{NN} \end{bmatrix}$$

$$D_{jj} = x_j B_{jj} + \frac{3}{2} A_{jj} - \frac{\phi_0^2}{4D_{\text{eff},i}^*(j)} \cdot \xi_j^2 \cdot \exp \left[-\frac{(1-\varepsilon_0)}{\varepsilon_0} (\xi_j^3 - 1) \right]$$

We solve the resulting system of N algebraic equations for the interior ordinates $c_i(1), \dots, c_i(N)$ and calculate the concentration profile by Lagrangian interpolation. Then, the conservation Eq. (6.12) for the solid deposit at each collocation point becomes as follows:

$$\frac{d\xi_j}{d\tau} = c_{i,j} \quad (6.27)$$

Thus, the mass balance equations for the agglomerate are reduced to a set of N simultaneous algebraic equations for the gas precursor concentration, Eq. (6.26), and N simultaneous first order ODEs for the solid deposit, Eq. (6.27).

gaiteye
Challenge the way we run

**EXPERIENCE THE POWER OF
FULL ENGAGEMENT...**

.....

**RUN FASTER.
RUN LONGER..
RUN EASIER...**

**READ MORE & PRE-ORDER TODAY
WWW.GAITEYE.COM**

6.3 PROGRAM DESCRIPTION

Notebook *AgglomerateCVD.ipynb*

◇ Import packages.

We will use the Python library *NumPy* for vector manipulations, calculation of exponential function, taking roots, etc. The *ode* module from the library *SciPy* is called to solve the system of ordinary differential equations and the *js_roots* module is used to find the roots of orthogonal polynomials. The *solve* module from the *SciPy* library is utilized to solve the dense linear system and the *InterpolatedUnivariateSpline* module is applied to interpolate the solid concentration profile in the radial direction of the agglomerate. To calculate the collocation matrices, we use the module *oc* from the *orthogonal_collocation* library. To plot the results of numerical simulations, we will use the *pyplot* module from the *matplotlib* library. The *IPython.html* module is called to display widgets for interactive input of parameter values.

```
%matplotlib inline
import numpy as np
from scipy.integrate import ode
from scipy.special.orthogonal import js_roots
from scipy.linalg import solve
from scipy.interpolate import InterpolatedUnivariateSpline
import orthogonal_collocation as oc
import matplotlib.pyplot as plt
from IPython.display import clear_output, display, HTML
from IPython.html.widgets import interact, FloatSlider, HTML, Latex
```

◇ Define the main function *fmain*:

o Input parameters:

T – temperature

Ragl – agglomerate radius

eag0 – initial agglomerate voidage

```
#+++++
#
# Main program
#
def fmain (T, Ragl, eag0, **kwargs):
```


- o Specify the global array Cgas.

```
#
global Cgas
```

- o Define the reaction and agglomerate parameters.

```
#
#+++++
# define reaction and agglomerate parameters
#-----
# temperature in [K]
T = T + 273.
# Arrhenius parameter
gamma = 18.3
# reference temperature, T0 [K]
T0 = 800.+273.15
# rate constant at reference temperature
k0 = 1.4e-4
# calculation of rate constant at temperature T
k = k0*np.exp(gamma*(1.0-(T0/T)))
# agglomerate size in [m]
Rag1 = Rag1*1.e-6
# initial size of primary particles, [m]
r0 = 0.75e-6/2.
# bulk gas concentration, [kmol/m3]
CAo = 1.e-2
```

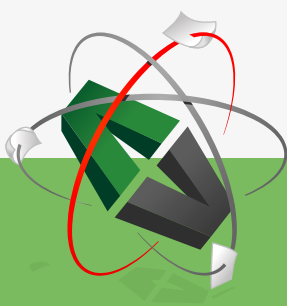
- o Calculate the initial diffusivity, surface area and Thiele modulus.

```
#
# calculate initial agglomerate and reaction parameters:
# initial effective diffusivity
rs = 1.0
De0 = Deff(εag0, r0, T, εag0, rs)
# initial internal area
S0 = 3.*(1.0 - εag0)/r0
# initial Thiele modulus
θ0 = Rag1*np.sqrt(k*(3.*(1 - εag0)/r0)/De0)
```

- o Specify the number of internal collocation points.

```
#  
#-----  
# specify number of internal collocation points  
    ncolpt = 7  
#  
# total number of points = No of internal colloc. points +  
#                           right boundary for x=1  
    ntpt = ncolpt + 1
```

This e-book
is made with
SetaPDF



PDF components for PHP developers

www.setasign.com



- o Set internal collocation points as roots of Jacobi polynomial.

```
#
# Calculate internal collocation points (xj)
# The value of parameters alpha and beta in Jacobi polynomials:
#   alpha = 1, beta = 1/2
# Corresponding parameters p and q in python function js_roots:
#   q = 3./2.
#   p = 1. + q
# internal collocation points, xj, are roots of Jacobi polynomial
xj = np.zeros(ncolpt)
wj = np.zeros(ncolpt)
[xj,wj] = js_roots(ncolpt,p,q)
# all collocation points = internal + right boundary
xr = np.zeros(ntpt)
xr[0:ncolpt] = xj[0:ncolpt]
xr[ntpt-1] = 1.0 # right boundary
```

- o Set matrices of collocation equation.

```
#
# calculate vectors of the first (dif1), second (dif2) and third (dif3)
# derivatives of the node polynomial at the roots
[dif1,dif2,dif3] = oc.dif(ntpt, xr)
#
# set up matrices of the first (Am) and second (Bm) derivative weights
Am = np.zeros((ntpt,ntpt))
Bm = np.zeros((ntpt,ntpt))
[Am,Bm] = oc.colmatrix (ntpt, xr, dif1, dif2, dif3)
#
# set up matrix Dm and vector Fm
Dm = np.zeros((ncolpt,ncolpt))
Fm = np.zeros(ncolpt)
Fm[0:ncolpt] = -
(xr[0:ncolpt]*Bm[0:ncolpt,ncolpt] + (3./2.)*Am[0:ncolpt,ncolpt])
for i in range (ncolpt):
    Dm[i,0:ncolpt] = (xr[i]*Bm[i,0:ncolpt] + (3./2.)*Am[i,0:ncolpt])
```

- o Collect parameters to pass to function f .

```
# combine parameters to pass to the function f
params = [εag0, r0, T, De0, θ0, Fm, Dm]
```

- o Allocate memory and set initial conditions for ODE.

```
# allocate memory and set initial conditions
Cgas = np.zeros(ncolpt)
r     = np.ones(ncolpt)
```

- o Set the time range for integration and calculate the number of time steps.

```
#
# set the time range for integration
    tau_start = 0.0
    tau_final = 0.2
# specify the time interval for data collection
    delta_tau = 0.05
# calculate the number of time steps required with an additional
# step for initial condition
    num_steps = np.floor((tau_final - tau_start)/delta_tau) + 1
```

- o Prepare plots.

```
#
# Prepare plots
    f, (ax1, ax2) = plt.subplots(2, 1, figsize=(6, 10))
```

- o Specify the parameters of ODE integrator.

```
#
# Specify the ODE integrator parameters:
# We use the vode integrator with backward differentiation formula to solve
# the system of stiff ODEs.
# The absolute tolerance is atol, the relative tolerance is rtol and
# the number of steps is nsteps
    rint = ode(ff).set_integrator('vode', method='bdf', nsteps=50000, atol = 1.0e-
05, rtol = 1.0e-05)
    rint.set_initial_value(r, tau_start).set_f_params(params)
```

- o Create vectors to store the axial distributions of r , c and w at specified time steps.

```
#
# Create vectors to store components axial distributions at specified time steps
    tau = np.zeros((num_steps, 1))
    r_at_tau = np.zeros((num_steps, ntpt))
    C_at_tau = np.zeros((num_steps, ntpt))
    w_at_tau = np.zeros((num_steps, ntpt))
```

- o Save initial values.

```
# Save initial values
    tau[0] = tau_start
    r_at_tau [0,0:ncolpt] = r[0:ncolpt]
```

- o Integrate the system of ODEs across each $\Delta\tau$ timestep.

```
#
# Integrate the system of ODEs across each delta_τ timestep
kk = 1
while rint.successful() and kk < num_steps:
    rint.integrate(rint.t + delta_τ)
```

- o Store the results of integration.

```
# store the results of integration:
# time
τ[kk] = rint.t
# normalized radius of primary particles
r_at_τ[kk,0:ncolpt] = rint.y[0:ncolpt]
# normalized gas concentration
C_at_τ[kk,0:ncolpt] = Cgas[0:ncolpt]
C_at_τ[kk,ncolpt] = 1.
# normalized mass concentration of deposit
for i in range(0,ncolpt):
    εag = np.exp(-(1.- εag0)*((r_at_τ[kk,i]**3) - 1.)/ εag0)
    w_at_τ[kk,i] = 1. - εag
```



**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider

o Interpolate gas and solid deposit radial profiles.

```
# interpolate radial gas concentration profile
numdiv = 51
x_int = np.linspace(0.0, 1.0, num=numdiv )
y_int = np.zeros(numdiv)

#
y = np.ones(ntpt)
y[0:ncolpt] = Cgas[0:ncolpt]

#
xintpr = np.zeros(ntpt)
for i in range(numdiv):
    xxx = x_int[i]*x_int[i]
    [xintpr] = oc.intrp(ntpt, xxx, xr, dif1)
    vector1 = xintpr*y
    y_int[i] = vector1.sum()

# interpolate solid concentration profiles using splines
f2 = InterpolatedUnivariateSpline(xr[0:ncolpt], w_at_tau[kk,0:ncolpt], k = 3)
```

o Plot graphs.

```
#
# plot graphs
plt.rcParams.update({'font.size': 12})
title1 = 'CVD reaction in agglomerate of fine particles: \n'
title1 += r'T [K] = %5.0f, Ragl [$\mu$m] = %5.0f, $\varepsilon_0$ [-] = %4.2f '
ax1.plot(x_int, y_int, linewidth=1.5, label=r"$\tau=%4.2f" % \tau[kk])
ax1.set_title(title1 % (T, Ragl/1.e-6, \varepsilon_0))
ax2.plot(x_int, f2(x_int), linewidth=1.5, label=r"$\tau=%4.2f" % \tau[kk])
ax1.set_ylabel('Gas concentration, $c$ [-]')
ax2.set_ylabel('Solid concentration, $w$ [-]')
ax1.set_xlim ([0.0, 1.])
ax1.set_ylim ([0.0, 1.])
ax2.set_xlim ([0.0, 1.])
ax2.set_ylim ([0.0, 1.])
ax1.set_xlabel('Radial position in agglomerate, $r$ [-]')
ax2.set_xlabel('Radial position in agglomerate, $r$ [-]')
ax1.legend(loc=4)
ax2.legend(loc=2)
```

o End of time integration loop.

```
#
kk += 1
```

o Save simulation results in csv files.

```
#
# save data as csv files
    with open("radial01.csv","w") as oname:
# save calculated profiles at collocation points
    for it in range (0, kk):
        oname.write("\n t =, %6.3f\n"% (τ[it]))
        for j in range (0, ncolpt):
            oname.write(" %6.3f ,%10.5e, %10.5e, %10.5e\n" % (xr[j], r_at_τ
[it,j],C_at_τ[it,j],w_at_τ[it,j]))
        with open("radial02.csv","w") as oname:
# interpolate and save interpolated profiles
        for it in range (0, kk):
# find the solution at numdiv interpolation points
            numdiv = 51
            x_int = np.linspace(0.0, 1.0, num=numdiv )
            y_int = np.zeros(numdiv)
            yr_int = np.zeros(numdiv)
#
            y = np.ones(ntpt)
            y[0:ncolpt] = C_at_τ[it, 0:ncolpt]
            yr = np.ones(ntpt)
            yr[0:ncolpt] = r_at_τ[it, 0:ncolpt]
#
            xintp = np.zeros(ntpt)
            for i in range(numdiv):
                xxx = x_int[i]*x_int[i]
                [xintp] = oc.intrp(ntpt, xxx, xr, dif1)
                vector1 = xintp*y
                vector2 = xintp*yr
                y_int[i] = vector1.sum()
                yr_int[i] = vector2.sum()
            oname.write("\n t =, %6.3f\n"% (τ[it]))
            for j in range (0, numdiv):
                oname.write(" %6.3f ,%10.5e, %10.5e\n" % (x_int[j], y_int[j], yr_int[j]))
```

◇ Specify the right-hand sides of ODEs.

```
#=====
#
def ff(τ, R, params):
#
# specify ODEs to solve
#
```

- o Set the global area, allocate arrays and unpack parameters.

```
#  
# global area  
#  
    global Cgas  
#  
# allocate arrays  
    ncolpt = len(R)  
    dRt = np.zeros(ncolpt)  
    Dm = np.zeros((ncolpt,ncolpt))  
#  
# unpack parameters  
    εag0, r0, T, Deff0, θ0, Fm, Dm1 = params
```



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.

- o Modify the main diagonal of matrix **Dm**.

```
# modify main diagonal of matrix Dm
Dm[:, :] = Dm1[:, :]
for i in range(ncolpt):
    rr = R[i]
# calculate:
# normalized voidage
    εag_norm = np.exp(-(1.- εag0)*((rr**3) - 1.)/ εag0)
# surface area
    S_norm = εag_norm*rr**2
# voidage
    εag = εag_norm * εag0
# normalized effective diffusivity
    Def = Deff(εag0, r0, T, εag, rr)/Deff0
# modify main diagonal
    Dm[i,i] = Dm[i,i] - (θ0**2)*S_norm/(4.*Def)
```

- o Solve the system of linear algebraic equations.

```
# solve the system of linear algebraic equations
Cgas[0:ncolpt] = 0.
Cgas = solve(Dm, Fm)
```

- o Calculate the vector of derivatives of ODEs.

```
#
# vector of derivatives of ODEs
dRt = Cgas
#
return dRt
```

- ◇ Calculate the effective diffusion coefficient.

- o The function *g*:

```
#
def Deff (ε0, r0, T, ε, rs):
#
# calculation of effective diffusion coefficient
#
# ε - input: local porosity
# rs - input: ratio r/r0
# De - output: effective diffusion coefficient
```

o Define parameters.

```
#
# Parameters:
# Pressure [atm]
P = 1.
# Molecular weight [kg/kmol]
# A - AlCl3; B - NH3; C - N2
M_A = 133.34; M_B = 17.03; M_C = 28.02
# Lennard-Jones parameters:
# characteristic energy of the A-B intermolecular potential
εA = 524.0; εB = 558.3; εC = 91.5
# molecular diameter
σA = 5.5; σB = 2.9; σC = 3.681
```

o Then, we calculate the molecular diffusivity.

```
#
# calculation of molecular diffusion coefficient using
# Chapman-Enskog theory [m2/s]
#
# characteristic energy (the well depth) of the
# A-B intermolecular potential
σAB = np.sqrt(σA * σB)
# the distance of the closest approach between two
# interacting molecular
σAB = (σA + σB)/2.0
TDs = T / εAB
# coef. in a collision integral approximation
A=1.06036; B=0.15610; C=0.19300; D=0.47635
E=1.03587; F=1.52996; G=1.76474; H=3.89411
# collision integral
Ω = A/(TDs**B) + C/np.exp(D*TDs) + E/np.exp(F*TDs) + G/np.exp(H*TDs)
# molecular diffusivity
Dm = 1.8583e-7*np.sqrt(T**3*((M_A+M_B)/(M_A*M_B)))/(P*(σAB**2)*Ω)
```

o Calculate the Knudsen diffusivity.

```
#
# calculation of Knudsen diffusion coefficient [m2/s]
#
# initial surface are
s0 = 3.*(1.-ε0)/r0
# initial pore size a0 [cm]
a0 = 2.*ε0*100./s0
# mean radius of capillary
ass = a0/(rs**2)
# Knudsen diffusivity
Dk = 0.96987*ass*np.sqrt(T/M_A)
```

- o Finally, we calculate the effective diffusivity.

```
#
# effective diffusion coefficient
#
De = ( $\epsilon^2$ )*(Dk*Dm)/(Dk+Dm)
#
return De
```

◇ Define widgets.

```
#
# Specify widgets
h1=HTML(value="<h4><b> Non-
Catalytic Reaction in an Agglomerate of Fine Particles</b>",color="brown")
h2=HTML(value = "<br> ")
h3=Latex(value="$$\mbox{Specify parameters: } $$",)
display(h1)
display(h2)
display(h3)
T_slider = FloatSlider(min=700, max=900, step=25, value=800, description="Temperature, $T$
[$^{\circ}C$] .....")
Ragl_slider = FloatSlider(min=50, max=200, step=25, value=100, description= "Agglomerate
radius, $R_{agl}$ [$\mu m$] ...")
 $\epsilon$ agl_slider = FloatSlider(min=0.3, max=0.6, step=0.05, value=0.4, description=r"Agglomerat
e voidage, $\varepsilon_{agl}$ [-] ..... ")
w = interact(fmain, T=T_slider, Ragl=Ragl_slider,  $\epsilon$ agl= $\epsilon$ agl_slider)
```



GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead

The screenshot of widgets to specify parameters and simulation results is shown in Fig. 6.1.

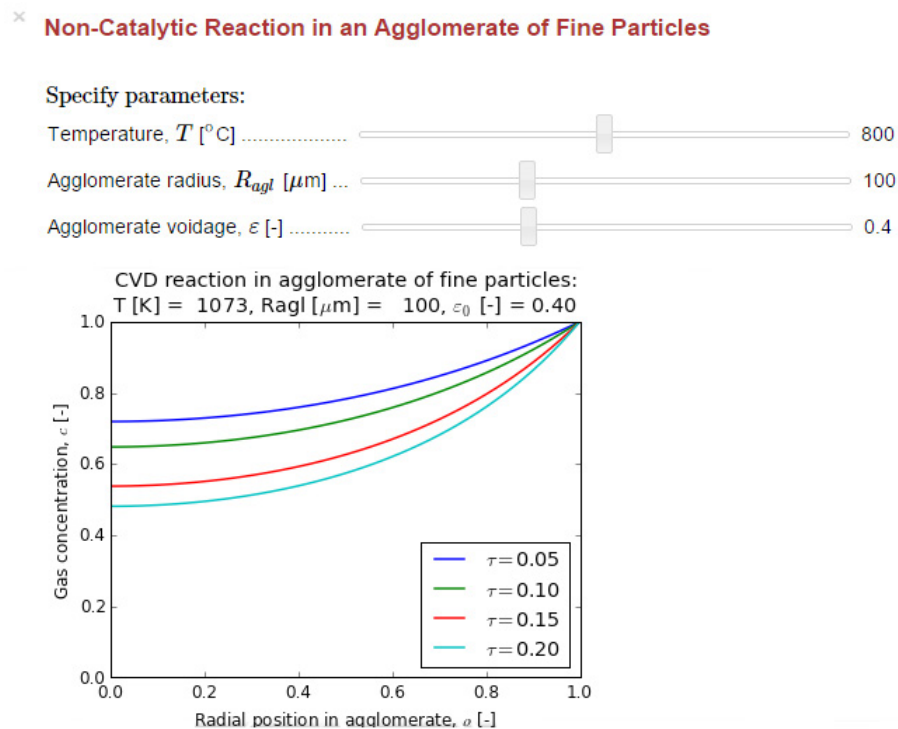


Figure 6.1: The widget to specify parameters for CVD reaction in agglomerate of fine particles.

6.4 NUMERICAL RESULTS

Here we consider the CVD reaction of aluminum trichloride with ammonia to deposit aluminum nitride on the surface of fine silicon nitride particles (Golman & Shinohara 1998).

The simulation parameters are summarized in Table 6.1.

Parameter	Symbol	Value
Temperature	T	800°C
Agglomerate size	R_0	100 mm
Initial radius of primary particles	r_0	0.375 mm
Initial voidage	ε_0	0.45
Initial Thiele modulus	ϕ_0	2.0

Table 6.1: Simulation parameters

Figure 6.2 illustrates the evolution of gas reactant and solid deposit concentration profiles in the agglomerate with reaction time. These profiles were calculated using the *AgglomerateCVD_time.ipynb* notebook. As the reaction proceeds, the gaseous reactant gets more non-uniformly distributed in the radial direction of the agglomerate (Fig. 6.2 (a)). The preferable growth of solid deposit (Fig. 6.2 (b)) results in a significant decrease in its voidage (Fig. 6.3 (a)) near the outer agglomerate surface at $\rho=1$. Thus, greater mass transfer limitations in the agglomerate prevent gaseous reactant to reach the area close to the center. The internal surface area available for reaction also decreases with reaction time, yielding a decline of reaction rate (Fig. 6.3 (b)). This should result in a more uniform distribution of solid product in the agglomerate. However, the local surface area at the radial position near the agglomerate center is significantly larger than that close to the agglomerate surface. The increasing diffusional resistance and non-uniformity of surface area distribution lead to the non-uniform solid distribution in the agglomerate with long reaction time in spite of decrease in reaction rate.

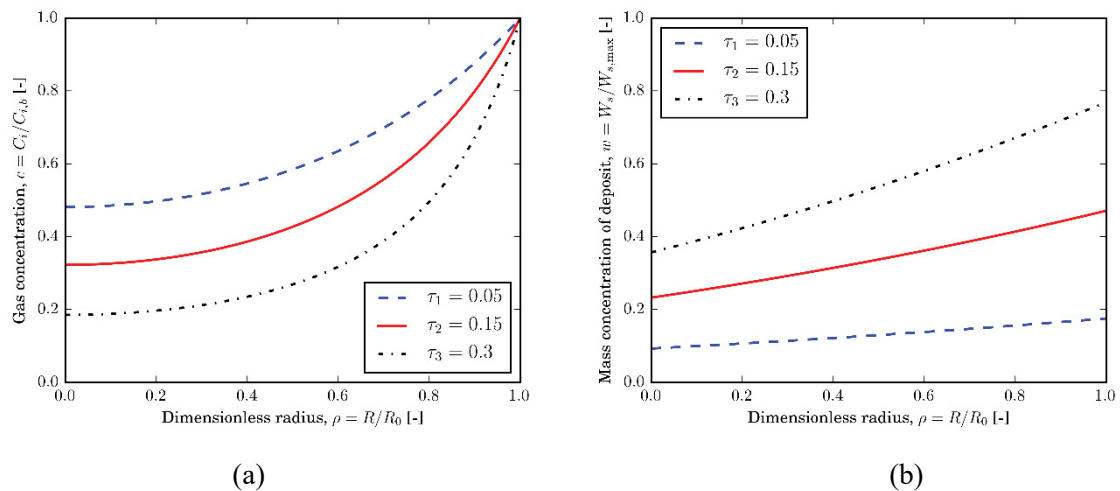


Figure 6.2: Evolution of (a) gas reactant and (b) solid deposit concentration profiles in the agglomerate with reaction time.

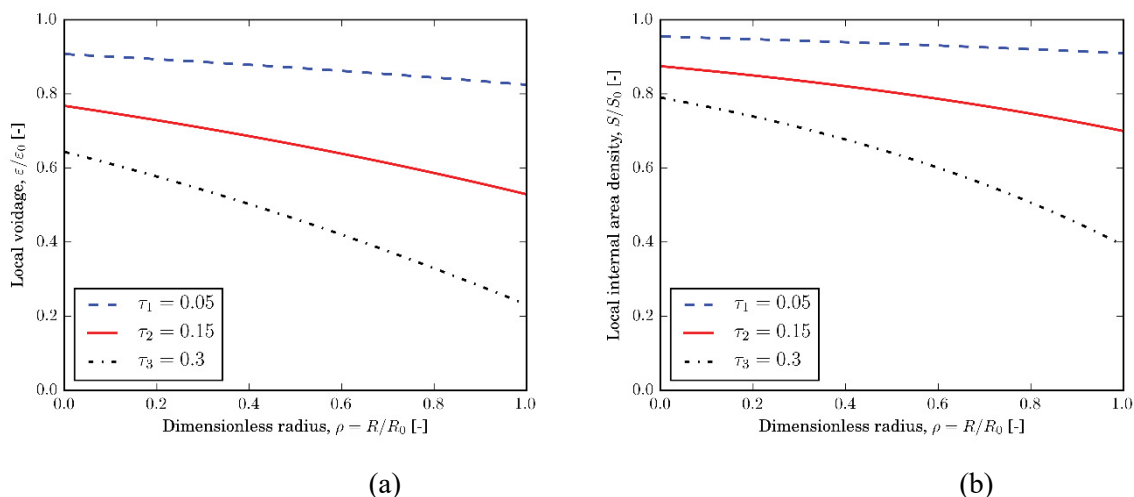


Figure 6.3: Radial profiles of (a) local normalized voidage and (b) local normalized surface area at various reaction times.

The effect of initial Thiele modulus on the gaseous reactant and solid product distributions is illustrated in Fig. 6.4. These profiles were calculated using the *AgglomerateCVD_thiele.ipynb* notebook. At high Thiele moduli the reactant and solid deposit are distributed non-uniformly along the agglomerate radius as a result of high reaction rate or high diffusion resistance in the agglomerate.

bookboon.com

Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)

Management

Time Management

Problem solving

Self-Confidence

Effectiveness

Project Management

Goal setting

Motivation

Coaching

Download free eBooks at bookboon.com

[Click on the ad to read more](#)

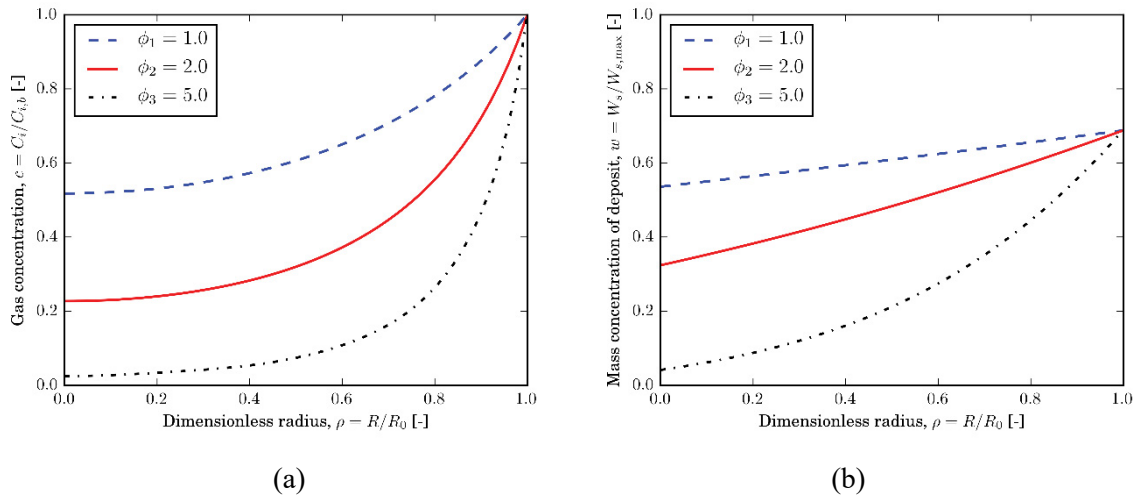


Figure 6.4: Effect of Thiele modulus on (a) gas reactant and (b) solid deposit concentration profiles in the agglomerate.

The influence of reaction temperature on the solid deposit profile shown in Fig. 6.5 was calculated using the *AgglomerateCVD_temperature.ipynb* notebook. Increasing the reaction temperature results in an enhancement of both reaction and diffusion rates. However, the reaction rate increases significantly with reaction temperature according to the Arrhenius dependence by Eq. (6.7), whereas the temperature dependence of diffusion rate is much weaker as the ordinary diffusion coefficient is proportional to $T^{3/2}$ (Eq. (1.1)) and the Knudsen diffusivity to $T^{1/2}$ (Eq. (1.5)). As a result, the solid product deposited at 900°C is non-uniformly distributed in the agglomerate, but the product deposited at 700°C is almost uniformly distributed.

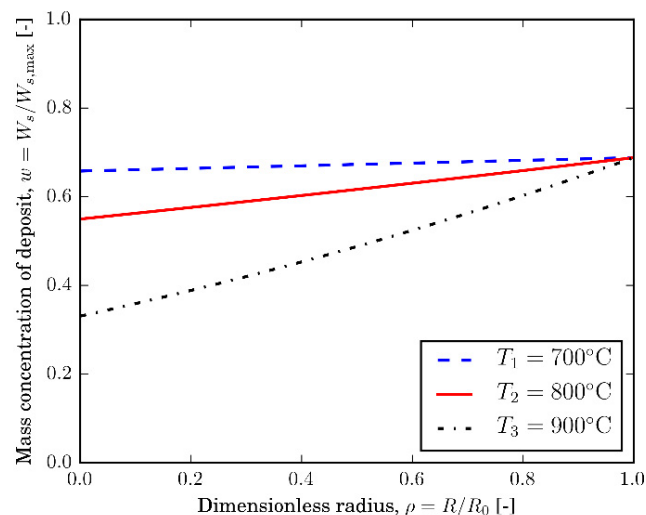


Figure 6.5: Effect of reaction temperature on the solid deposit concentration profile, $\tau = 0.25$.

The concentration profiles of solid product in the agglomerates of various initial voidages are shown in Fig. 6.6. These profiles were calculated using the *AgglomerateCVD_voidage.ipynb* notebook. The solid product becomes uniformly distributed in the agglomerate of large initial voidage due to the low resistance to the transport of gaseous product in the agglomerate. The diffusion coefficient is the strong function of dimensionless voidage, $D_{\text{eff},i}^* = (\varepsilon / \varepsilon_0)^2$, by Eq. (6.16). Moreover, the high initial porosity agglomerate has the low initial surface area, as $S_0 = 3 \cdot (1 - \varepsilon_0) / r_0$. Thus, both the low reaction rate and the high diffusion rate contribute to the formation of uniformly distributed solid product in the loose agglomerate.

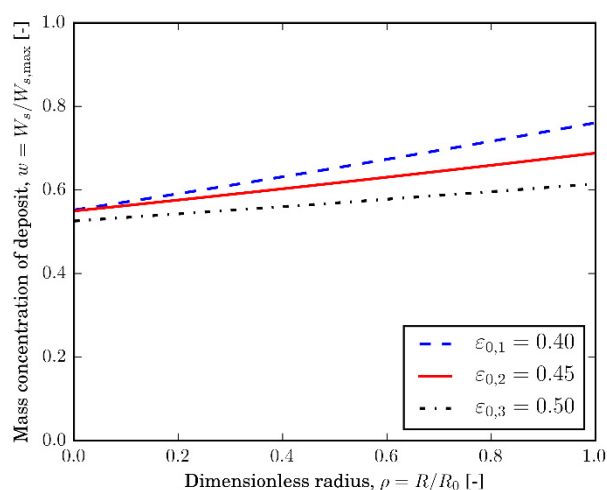


Figure 6.6: Influence of agglomerate initial voidage on solid deposit concentration profiles in the agglomerate, $\tau = 0.25$.

The effect of agglomerate size on the solid deposit profile is illustrated in Fig. 6.7. The *AgglomerateCVD_AgglSize.ipynb* notebook was used to calculate the deposit profiles. The radial profiles are steeper in the large size agglomerates due to the increasing diffusion resistance as the result of the longer diffusion path.

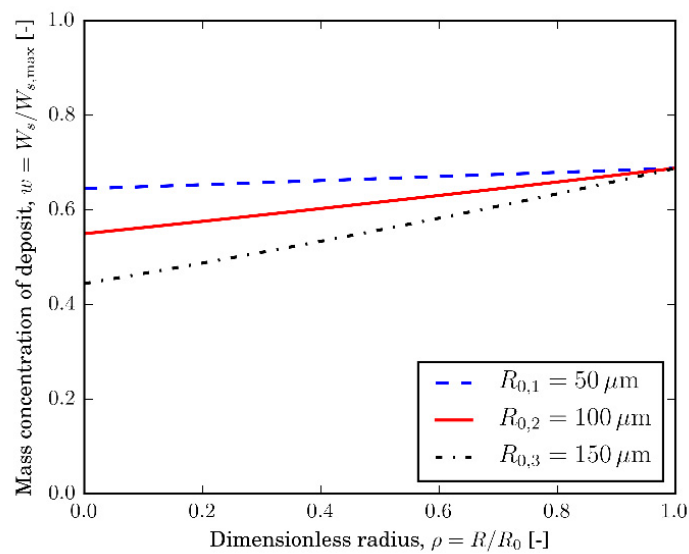


Figure 6.7: Effect of agglomerate size on solid deposit concentration profiles in the agglomerate, $\tau = 0.25$.

Brain power

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations. Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

The effects of initial size of primary particles on the solid deposit profiles is shown in Fig. 6.8. These profiles were calculated using the *AgglomerateCVD_ParticleSize.ipynb* notebook. The agglomerates made of large size primary particles have the small initial surface area and pores of large sizes by Eq. (1.6). These will lead to the lower reaction rate and higher Knudsen diffusivity and, as a result, to the more uniform distribution of solid deposit.

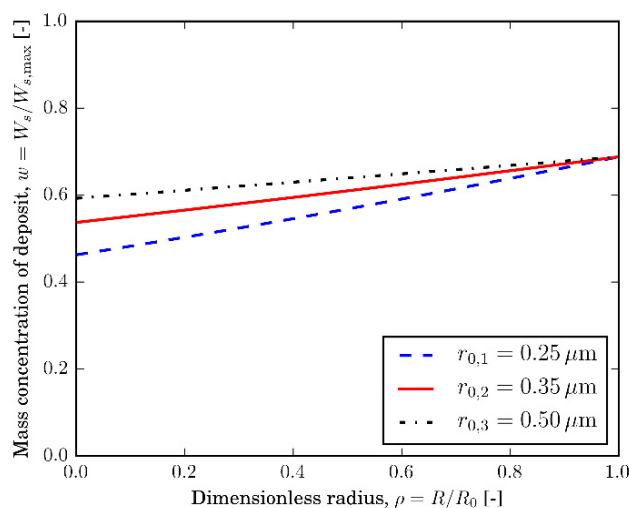


Figure 6.8: Influence of initial size of primary particles on solid deposit concentration profiles in the agglomerate, $\tau = 0.25$.

SUMMARY

In the previous chapters we mainly discussed chemical reactions and the heat and mass transport processes in the catalyst pellet. The basic mathematical models and numerical or analytical methods to solve these model equations have been also introduced together with the computer programs used for simulating and plotting the concentration and temperature profiles in the pellets. We also covered the first- and second-order catalytic, enzyme catalyzed and non-catalytic reactions in the isothermal and non-isothermal pellets in this textbook. Readers are advised to refer to the books by Aris (1975) and Kafarow (1993) for further exploration of complex phenomena involving simultaneous coupled heat and mass transfer with chemical reaction in the pellet. The pellet models comprise an essential part of the overall reactor models, which will be discussed in the Part II of this book series.

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on

REFERENCES

- Aris, R 1975, *The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts. Vol. 1: The Theory of the Steady State*, Clarendon Press, Oxford.
- Beers, KJ 2007, *Numerical Methods for Chemical Engineering*, Cambridge University Press, Cambridge, pp. 270–282.
- Bird, RB, Stewart, WE & Lightfoot, EN 2002, *Transport Phenomena*, 2nd edn. J. Wiley & Sons, New York.
- Butt, JB 2000, *Reaction Kinetics and Reactor Design*, 2nd edn. Marcel Dekker, New York.
- Finlayson, BA 1980, *Nonlinear Analysis in Chemical Engineering*, McGraw-Hill, New York, pp. 96–149.
- Fogler, HS 2008, *Elements of Chemical Reaction Engineering*, 4th edn., Pearson Prentice Hall, NJ, pp. 813–842.
- Froment, GF, Bischoff, KB & De Wilde J 2011, *Chemical Reactor Analysis and Design*, 3rd edn., J. Wiley Sons, New York, pp. 172–231.
- Golman, B & Shinohara, K 2000, ‘Fine particle coating by chemical vapor deposition for functional materials’, *Trends in Chemical Engineering*, vol. 6, pp. 1–16.
- Golman, B & Shinohara, K 1998, ‘Modeling of CVD coating inside agglomerate of fine particles’, *Journal of Chemical Engineering of Japan*, vol. 31, no. 1, pp. 103–110.
- Hunter, JD 2007, ‘Matplotlib: A 2D graphics environment’, *Computing in Science and Engineering*, vol. 9, pp. 90–95.
- Jeong, S, Lee, KS, Keel SIn, Yun JH, Kim YJ & Kim, SS 2015, ‘Mechanisms of direct and in-direct sulfation of limestone’, *Fuel*, vol. 161, pp. 1–11.
- Jones, E, Oliphant T, Peterson P, et al. 2001. SciPy: Open Source Scientific Tools for Python, <http://www.scipy.org/> [Online; accessed 2015-09-20].

Kafarow, WW 1993, *Methoden zur Entwicklung von industriellen katalytischen Prozessen*, VDI-Verlag, Düsseldorf.

Kandiyoti R 2009, *Fundamentals of Reaction Engineering*, Ventus Publishing ApS, 2009.

Langtangen, HP 2012, *A Primer on Scientific Programming with Python*, 3rd edn., Springer, Heifberg.

Perez, F & Granger, BE 2007, 'IPython: a system for interactive scientific computing', *Computing in Science and Engineering*, vol. 9, pp. 21–29.

Shuler, ML & Kargi F 2002, *Bioprocess Engineering. Basic Concepts*, Prentice Hall, NJ, pp. 60–90.

Reid, R, Prausnitz, J & Poling, B 1987, *The Properties of Gases and Liquids*, 4th edn., McGraw-Hill, New York.

Rice, RG & Do, DD 2012, *Applied Mathematics and Modeling for Chemical Engineering*, 2nd edn., John Wiley & Sons, New York, pp. 172–173.

Villadsen, J & Michelsen, ML 1978, *Solution of Differential Equation Models by Polynomial Approximation*, Prentice Hall, NJ.

Wakao, N & Smith, JM 1962, 'Diffusion in catalyst pellet', *Chemical Engineering Science*, vol. 17, pp. 825–834.

Woodside, W & Messmer, JH 1961, 'Thermal conductivity of porous media (parts I and II)', *Applied Physics*, Vol. 32, pp. 1688–1707.

Xu, Y & Yan, X-T 2010, 'Chemical Vapor Infiltration' in *Chemical Vapour Deposition: An Integrated Engineering Design for Advanced Materials*, Springer-Verlag, London, pp. 165–214.

APPENDIX A1. INSTALLING IPYTHON

There are many different ways of installing IPython (Pérez & Granger 2007). We recommend installing the *Anaconda* distribution by Continuum Analytics. This distribution contains a full Python set for scientific and engineering computation and data visualization including *NumPy*, *Scipy* (Jones et al. 2001) and *Matplotlib* (Hunter 2007). *Anaconda* is an open source platform and it can be installed using a free and easy to use installer. The installer is available at <https://www.continuum.io/downloads>. The notebooks described in this book were created using Python 3.4. You might also want to install a *pandoc* package available at <http://pandoc.org>. This is a universal document converter and it is used by *nbconvert* tool (<https://ipython.org/ipython-doc/3/notebook/nbconvert.html>) to convert IPython notebook to html, pdf, LATEX and other formats.

We can start *Anaconda* using a launcher for IPython Notebook (*IPython (Py 3.4) Notebook*) in the start menu on Windows computer. This launcher will open a local web server and we will interact with IPython using a web browser, as shown in Fig. A1.1.

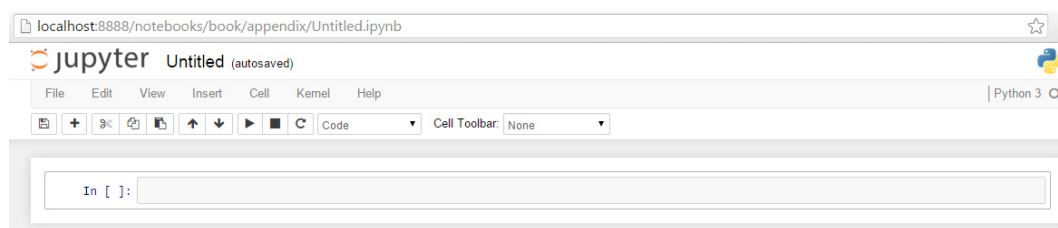


Figure A1.1: Illustration of IPython web based interface.

We can create a new file or open an existing one, as illustrated in Fig. A1.2. The notebook files are located in the directory in which *Anaconda* was installed, i.e. *c:/users/boris*. We have created a folder named '*book*' with subfolder '*appendix*' in this directory and saved there a newly created notebook '*Untitled*'.

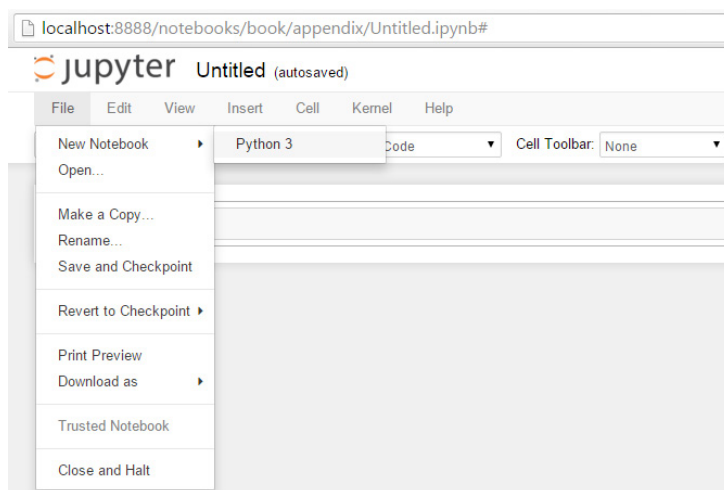


Figure A1.2: Creation of new IPython notebook.

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be.

Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.

© 2013 Accenture. All rights reserved.

The IPython (jupyter) notebook files contain the program code, text and reach media output. All of this information is stored in the cells. There are different types of cells, as illustrated in Fig. A1.3. The *Code* cell is used to type the Python code. The *Markdown* cell is utilized to show the text information such as model explanation using the markdown language and LATEX code.

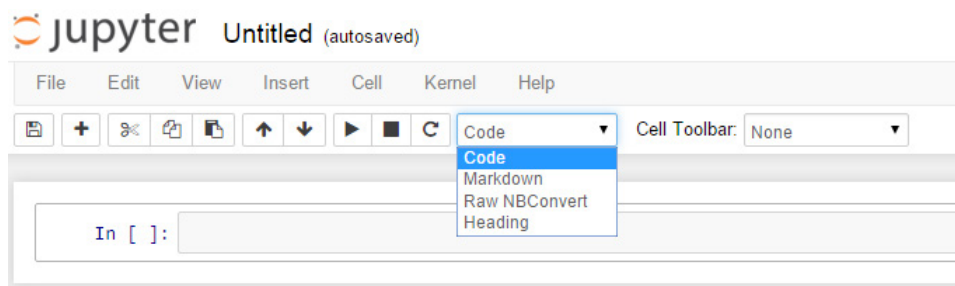
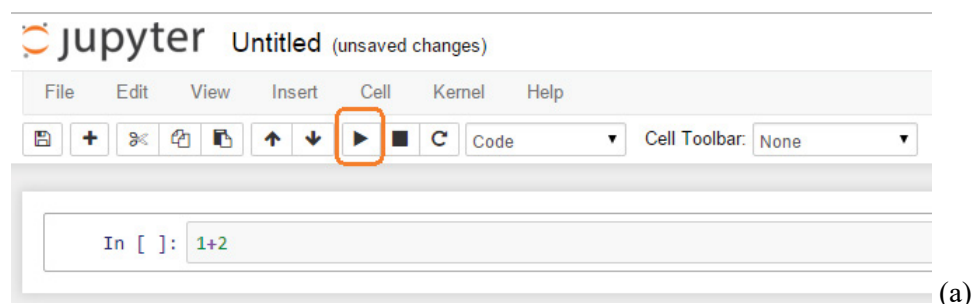
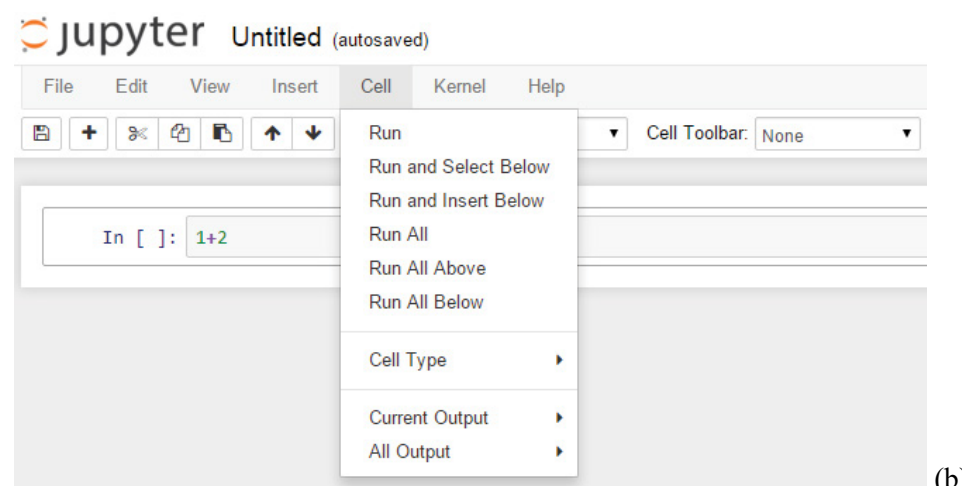


Figure A1.3: Illustration of different types of cells.

To run the code cell, we can click the run button, as shown in Fig. A1.4 (a), or choose the proper item in a dropdown menu, as illustrated in Fig. A1.4 (b).



(a)



(b)

Figure A1.4: Illustration of two ways to run the IPython cell: (a) using a button and (b) using a menu selection.

The results of simulation are shown in the output cell below the code cell, as displayed in Fig. A1.5.

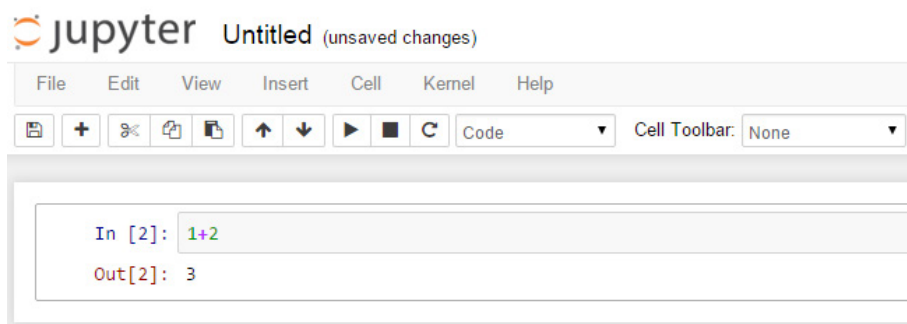


Figure A1.5: Screenshot of IPython input and output cells.

"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download

APPENDIX A2. BRIEF OVERVIEW OF PYTHON LANGUAGE

The Python syntax is briefly summarized below.

- Indentation is used to indicate the block of statements.
- The line break specifies the statement termination. Statements are usually one line long, but some statements could be located on the same line and separated by a semicolon.
- The comment starts from # symbol.
- The arithmetic operator ** indicates power.
- Decision making.

The syntax of condition statement is as follows:

```
if <condition>:  
    <then statements>  
else:  
    <else statements>  
next statement
```

The following condition logical operator are defined: < less than, > greater than, <= less than or equal, >= greater than or equal, == equal to, != not equal to.

```
In [5]: i=3  
        if i > 5:  
            j = 1  
        else:  
            j = 2  
        print('j = %s' % j)  
j = 2
```

- For loop.
For loop is used to repeat execution of a piece of code.
The syntax of loop statement is as follows:
for <variable> in range (start, stop, step):
 <body of for loop>
In the example below, the start value is equal to 1, the stop value is 6 and the step is 2.

```
In [1]: sum = 0
        for i in range (1, 6, 2):
            sum = sum + i
            print('i = %s' % i)
        print('sum = %s' % sum)

i = 1
i = 3
i = 5
sum = 9
```

If the start value is not defined, it is assumed to be zero. If the step value is not given it is presumed to be one. Therefore, the loop below will be executed 3 times starting from $i = 0$.

```
In [2]: sum = 0
        for i in range (3):
            sum = sum + i
            print('i = %s' % i)
        print('sum = %s' % sum)

i = 0
i = 1
i = 2
sum = 3
```

- The vector can be created using the *numpy* array functions.
For example, $a = np.zeros(n)$ is the statement to create a null vector **a** with n elements.
The following code creates the array **a** with 3 elements and prints it out.

```
In [7]: import numpy as np
        a=np.zeros(3)
        print('a[0] = %s' % a[0])
        print('a[1] = %s' % a[1])
        print('a[2] = %s' % a[2])

a[0] = 0.0
a[1] = 0.0
a[2] = 0.0
```

Below we specify the two-dimensional array **b** of size 2×3 and print out all array elements.

```
In [20]: import numpy as np
b = np.array([[1, 2, 3], [4, 5, 6]])
print('b[0,0] = %s' % b[0,0])
print('b[0,1] = %s' % b[0,1])
print('b[0,2] = %s' % b[0,2])
print('b[1,0] = %s' % b[1,0])
print('b[1,1] = %s' % b[1,1])
print('b[1,2] = %s' % b[1,2])

b[0,0] = 1
b[0,1] = 2
b[0,2] = 3
b[1,0] = 4
b[1,1] = 5
b[1,2] = 6
```

We can use slicing to generate the desired view of the array. The following statements slice the second column and the first row of two-dimensional array defined above.

```
In [23]: print ('The second column: %s' %b[:,1])
print ('The first row: %s' %b[0,:])

The second column: [2 5]
The first row: [1 2 3]
```

- Function

A function is a piece of code to perform a task. The function is defined with a keyword *def* followed by the function name and the list of arguments in parenthesis. The final statement is a *return* statement with the list of expressions in parenthesis. We can initialize the function calculation by calling it.

```
In [2]: def fun (a, b):  
        d = a + b  
        return [d]  
e = 1  
f = 2  
sum = fun(e,f)  
print ('sum: %s' % sum)  
  
sum: [3]
```

For further information on Python language, the reader is advised to consult numerous online recourses and books such as a comprehensive book by Langtangen (2012).



The advertisement features a grey background with a faint world map. In the top left is the Duke University logo. The text 'BUSINESS HAPPENS' is prominently displayed in the center. Below it is the website URL 'www.fuqua.duke.edu/globalmba'. On the right side, there is a circular collage of six diverse individuals' faces. An orange button with the text 'Learn More >' is located at the bottom center.

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

www.fuqua.duke.edu/globalmba

Learn More >

HERE.

APPENDIX A3. AUXILIARY PROGRAMS USED IN ORTHOGONAL COLLOCATION METHOD

The Jacobi polynomials of degree N can be written in the form of a power series as (Villadsen & Michelsen 1978; Rice & Do 2012)

$$J_N^{(\alpha, \beta)}(x) = \sum_{i=0}^N (-1)^{N-i} \cdot \gamma_{N,i} \cdot x^i, \quad (\text{A3.1})$$

where $\gamma_{N,i}$ are the coefficients, and α and β are the characteristic parameters.

The coefficients γ can be found using the orthogonality property as

$$\int_0^1 x^\beta (1-x)^\alpha J_j^{(\alpha, \beta)}(x) J_N^{(\alpha, \beta)}(x) dx = 0, j = 0, \dots, N-1 \quad (\text{A3.2})$$

To obtain an explicit expression for $J_N^{(\alpha, \beta)}(x)$, the recursive computation of $\gamma_{N,i}$ can be used as

$$\gamma_{N,i} = \frac{N-i+1}{i} \cdot \frac{N+i+\alpha+\beta}{i+\beta} \cdot \gamma_{N,i-1} \quad (\text{A3.3})$$

for $i=1, \dots, N$ starting from $\gamma_{N,0}=1$.

Equation (A3.3) can be rewritten as

$$p_N(x) = (x - g_N(N, \alpha, \beta)) p_{N-1} - h_N(N, \alpha, \beta) p_{N-2}, \quad (\text{A3.4})$$

where p_N is the rescaled polynomial defined as $p_N = \frac{J_N^{(\alpha, \beta)}(x)}{\gamma_{N,N}}$.

Here,

$$\begin{aligned}
 g_1 &= \frac{\beta+1}{\alpha+\beta+2} \\
 g_N &= \frac{1}{2} \left[1 - \frac{\alpha^2 - \beta^2}{(2N + \alpha + \beta - 1)^2 - 1} \right] \quad \text{for } N > 1 \\
 h_1 &= 0 \\
 h_2 &= \frac{(\alpha+1)(\beta+1)}{(\alpha+\beta+2)^2(\alpha+\beta+3)} \\
 h_N &= \frac{(N-1)(N+\alpha-1)(N+\beta-1)(N+\alpha+\beta-1)}{(2N+\alpha+\beta-1)(2N+\alpha+\beta-2)^2(2N+\alpha+\beta-3)} \quad \text{for } N > 2n
 \end{aligned}$$

An interpolation polynomial of N degree passing through $N+1$ points is defined using the Lagrange formula as

$$y_N(x) = \sum_{i=1}^{N+1} y_i l_i(x), \quad (\text{A3.5})$$

where y_i is the value of y at x_i , and $l_i(x)$ is the Lagrangian polynomial specified as

$$l_i(x_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (\text{A3.6})$$

The Lagrangian polynomial can be evaluated as

$$l_i(x) = \frac{p_{N+1}(x)}{(x-x_i) \frac{dp_{N+1}(x)}{dx}} \quad (\text{A3.7})$$

The first and second derivatives of the interpolation polynomial are given at the interpolation point x_i as

$$\frac{dy_N(x_i)}{dx} = \sum_{j=1}^{N+1} y_j \frac{dl_j(x_i)}{dx} \quad (\text{A3.8})$$

$$\frac{d^2 y_N(x_i)}{dx^2} = \sum_{j=1}^{N+1} y_j \frac{d^2 l_j(x_i)}{dx^2} \quad (\text{A3.9})$$

Using the matrix notation, Eqs. (A3.8) and (A3.9) can be written as

$$\mathbf{y}' = \mathbf{A} \cdot \mathbf{y} \quad (\text{A3.10})$$

$$\mathbf{y}'' = \mathbf{B} \cdot \mathbf{y} \quad (\text{A3.11})$$

where the elements of matrices **A** and **B** are given as

$$a_{i,j} = \begin{cases} \frac{dl_j(x_i)}{dx} = \frac{1}{2} \frac{p_{N+1}^{(2)}(x_i)}{p_{N+1}^{(1)}(x_i)} & \text{at } j=i \\ \frac{dl_j(x_i)}{dx} = \frac{1}{x_i - x_j} \frac{p_{N+1}^{(1)}(x_i)}{p_{N+1}^{(1)}(x_j)} & \text{at } j \neq i \end{cases} \quad (\text{A3.12})$$

$$b_{i,j} = \begin{cases} \frac{d^2l_j(x_i)}{dx^2} = \frac{1}{3} \frac{p_{N+1}^{(3)}(x_i)}{p_{N+1}^{(1)}(x_i)} & \text{at } j=i \\ \frac{d^2l_j(x_i)}{dx^2} = 2a_{i,j} \left(a_{i,i} - \frac{1}{x_i - x_j} \right) & \text{at } j \neq i \end{cases} \quad (\text{A3.13})$$

The first, second and third derivatives of the polynomial $p_{N+1}(x) = \prod_{j=1}^{N+1} (x - x_j)$ are evaluated using the following recurrence formulas

$$\begin{aligned} p_j^{(1)}(x_i) &= (x_i - x_j) p_{j-1}^{(1)}(x_i) \\ p_j^{(2)}(x_i) &= (x_i - x_j) p_{j-1}^{(2)}(x_i) + 2p_{j-1}^{(1)}(x_i) \\ p_j^{(3)}(x_i) &= (x_i - x_j) p_{j-1}^{(3)}(x_i) + 3p_{j-1}^{(2)}(x_i) \end{aligned} \quad (\text{A3.14})$$

with $j = 2, \dots, i-1, i+1, \dots, N+1$ and $p_1^{(1)}(x_i) = 1, p_1^{(2)}(x_i) = 0, p_1^{(3)}(x_i) = 0$.

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education.
More info [here](http://ligsuniversity.com).



The listing of auxiliary programs used in orthogonal collocation method is shown below. The programs were adopted from the FORTRAN programs given in Villadsen and Michelsen (1978).

- Function *dif* is used to calculate the first, second and third derivatives of Lagrangian polynomial at interpolation points.

```
import numpy as np
def dif(nt, root):
#
# Calculate first, second and third derivatives of Lagrangian polynomial
# at interpolation points
#
# Input:
#   nt - total number of interpolation points
#   root - zeros of node polynomial (nt)
# Output:
#   dif1 - first derivative at interpolation points (nt)
#   dif2 - second derivative at interpolation points (nt)
#   dif3 - third derivative at interpolation points (nt)
#
    dif1 = np.ones(nt)
    dif2 = np.zeros(nt)
    dif3 = np.zeros(nt)
    for i in range (nt):
        x = root[i]
        for j in range (nt):
            if j != i :
                y = x - root[j]
                dif3[i] = y*dif3[i] + 3.0*dif2[i]
                dif2[i] = y*dif2[i] + 2.0*dif1[i]
                dif1[i] = y*dif1[i]
    return [dif1, dif2, dif3]
```

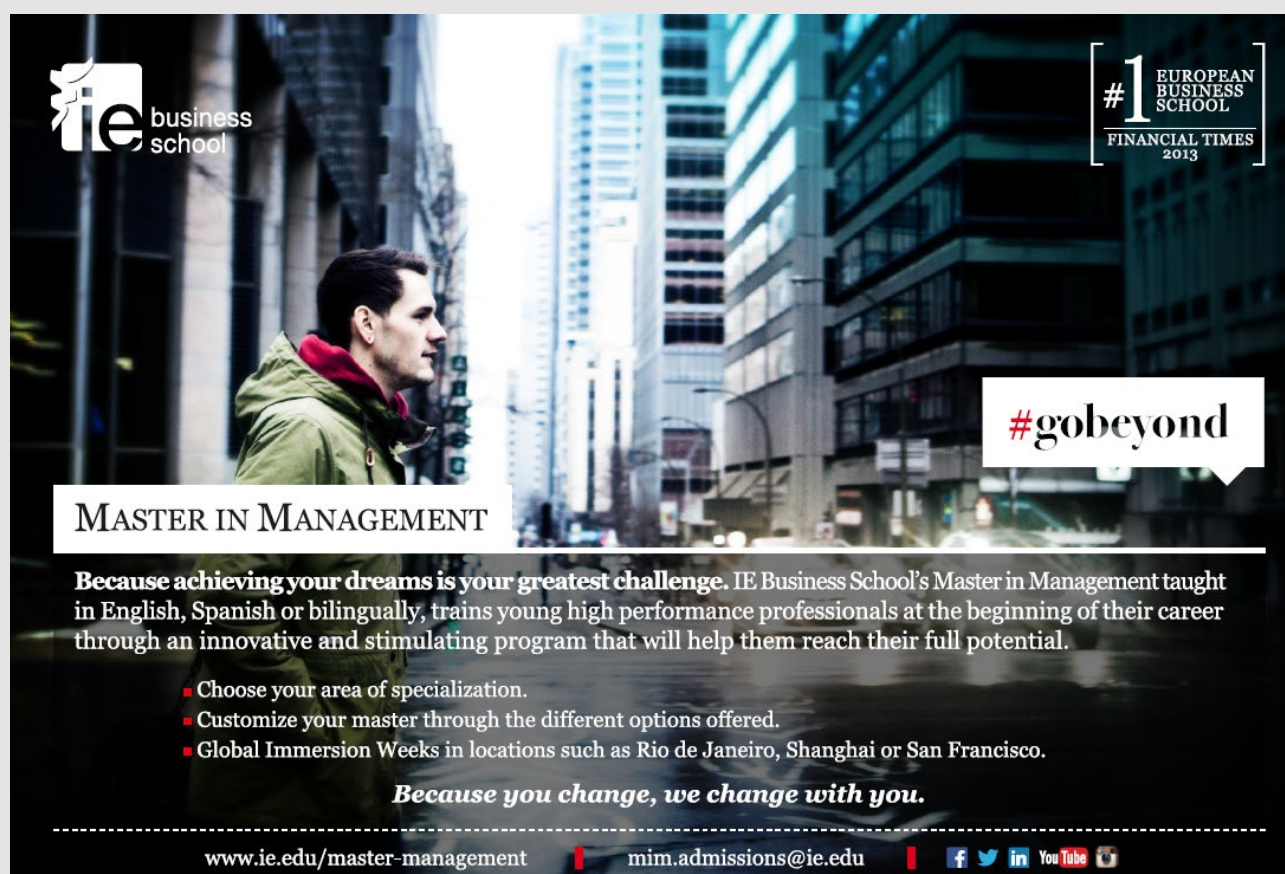
- Function *colmatrix* is utilized to calculate the matrices of first and second derivatives of interpolation polynomial.

```
def colmatrix (nt, root, dif1, dif2, dif3):
#
# Set up matrices of first, a, and second, b, derivatives of interpolation
# polynomial
#
# Input:
#   nt - total number of interpolation points
#   root - zeros of node polynomial (nt)
#   dif1 - first derivative at interpolation points (nt)
#   dif2 - second derivative at interpolation points (nt)
#   dif3 - third derivative at interpolation points (nt)
# Output:
#   a - matrix of first derivatives of interpolation polynomial (nt*nt)
#   b - matrix of second derivatives of interpolation polynomial (nt*nt)
#
a = np.zeros((nt,nt))
b = np.zeros((nt,nt))
vect1 = np.zeros(nt)
vect2 = np.zeros(nt)
for i in range (nt):
#   calculate first derivatives at point i
vect1[:i] = dif1[i]/dif1[:i]/(root[i] - root[:i])
vect1[i] = dif2[i]/dif1[i]/2.0
vect1[i+1:] = dif1[i]/dif1[i+1:]/(root[i] - root[i+1:])
#   calculate second derivatives at point i
vect2[:i] = ((dif1[i]/dif1[:i]/(root[i] - root[:i]))*
             (dif2[i]/dif1[i] - 2.0/(root[i] - root[:i])))
vect2[i] = dif3[i]/dif1[i]/3.0
vect2[i+1:] = ((dif1[i]/dif1[i+1:]/(root[i] - root[i+1:]))*
              (dif2[i]/dif1[i] - 2.0/(root[i] - root[i+1:])))

a[i,:] = vect1
b[i,:] = vect2
return [a, b]
```

- Function *intrp* is used to calculate the Lagrangian interpolation coefficients.

```
def intrp(nt, x, root, dif1):
#
# Calculate Lagrangian interpolation coefficients
#
# Input:
#   nt   - total number of interpolation points
#   x    - abscissa of point at which function value is desired
#   root - zeros of node polynomial (nt)
#   dif1 - first derivative at interpolation points (nt)
# Output:
#   xintp - lagrangian interpolation coefficients (nt)
#
    xintp = np.zeros(nt)
#
    pol = 1
    for i in range(0, nt):
        Y = x - root[i]
        if (Y == 0):
            xintp[i] = 1
        else:
            xintp[i] = 0
        pol = pol * Y
    if (pol == 0):
        return[xintp]
    for i in range(0, nt):
        xintp[i] = pol / dif1[i]/(x - root[i])
    return [xintp]
```



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu | f t in YouTube

- Function *radau* is utilized to calculate the weights of Radau quadrature.

```
def radau (nt, root, dif1):
#
# Calculate Radau quadrature weights
# (including x=1)
#
# Input:
# nt - total number of interpolation points
# root - zeros of node polynomial (nt)
# dif1 - first derivative at interpolation points (nt)
# Output:
# vect - Radau quadrature weights (nt)
#
    vect = np.zeros(nt)
    vect = (1.0/root)/(dif1**2)
    vect = vect/vect.sum()
    return [vect]
```