

# Calculation methods applied to chemistry

(Méthodes de calcul appliqué à la chimie)

- Learning outcomes teaching unit (UE):
  - Apply standard numerical methods or existing software to solve numerical problems related to scientific research activities
  - Be active in the search for existing numerical methods adapted to problems encountered by chemists
- Content of the UE:
  - widespread methods : linear systems, numerical integration, root findings
  - Ordinary differential equations (numerical solutions, kinetic applications,...)
  - Partial differential equations (finite differences, diffusion problems)
  - Nonlinear systems of equations (Newton-Raphson method)
  - Eigenvalues eigenvectors problems (applications to relaxation and population problems)
  - Approximation by linear and non-linear least squares methods (maximum likelihood, application to deconvolution)
  - Chebyshev approximation
  - Molecular modeling and visualization
  - Minimization and conformational problems
- Prerequisite skills
  - Basic knowledge of a programming language
  - Basics of Mathematics
- Exercises and applications: codes written in or mainly written in Python, with the general libraries matplotlib, numpy, scipy, pandas as well as other specialized libraries, especially in chemistry
- Types of evaluations: Oral examination based on an in-depth study on one of the chapters of the course or an additional theme
- Acquis d'apprentissage UE :
  - Appliquer des méthodes numériques standards ou des logiciels existant pour résoudre des problèmes fondamentaux ou annexes, liés à des activités de recherche scientifique
  - Être actif dans la recherche de méthodes de résolution numérique existantes et adaptées à des problèmes auxquels les chimistes sont confrontés
- Contenu de l'UE :
  - Équations différentielles ordinaires (résolutions numériques et applications cinétiques)
  - Équations aux dérivées partielles (différences finies, problèmes de diffusion)
  - Systèmes d'équations non linéaires (méthode de Newton-Raphson)
  - Problèmes aux valeurs propres (applications à des problèmes de relaxation et de population)
  - Approximation par moindres carrés linéaires et non-linéaires (application à la déconvolution)
  - Approximation de Tchébychev
  - Modélisation et visualisation de molécules
  - Minimisation et problèmes conformationnels
- Compétences préalables
  - Connaissance de base d'un langage de programmation
  - Bases des mathématiques
- Exercices et applications : codes écrits ou à écrire principalement en Python, avec les librairies générales matplotlib, numpy et scipy, ainsi que d'autres librairies spécialisées, notamment en

chimie


- Types d'évaluations : Examen oral sur base d'un travail approfondi sur un des chapitres du cours ou un thème additionnel

---

## Synopsis (english)

### Mathematical prerequisites

### Programming bases and tools

- Python programming language
-  [Anaconda Python distribution](https://www.anaconda.com/download/) (64 bits, with Python 3.x) :  
<https://www.anaconda.com/download/>
  - Includes these tools :
    - Jupyter notebook (interactive web-based environment)
    - qtconsole (high level Python console with graphics & colors)
    - spyder (powerful Python IDE)
  - includes lot of Python libraries : matplotlib, numpy, scipy, pandas,...
  - package management system (conda), virtual environments
  - Anaconda Navigator includes extensive documentation (on anaconda website and dedicated websites)
- GNU/Linux OS (preferred)
- Jupyter introductions, tutorials, ...
  - [Jupyter Notebook Tutorial](#), par Den Kasyanov (Medium)
  - [Jupyter official documentation](#)
  - [Jupyter DataCamp Cheat Sheet](#)
- Python scientific libraries (official websites including tutorials and documentation)
- cf. [this](#) (french)
- [Matplotlib](#) (scientific graphs)
- [NumPy](#) (array manipulations, linear algebra, Fourier transforms, random numbers,...)
- [SciPy](#) numerical methods (integrations, ODE, PDE,...)
- [SymPy](#) symbolis maths
- [Pandas](#), data analysis
- Pylab, combine Matplotlib, NumPy and SciPy
- [Scikit-learn](#), machine learning

Some MOOCs to help learning :

- [Introduction to Python: Absolute Beginner](#) (edX)
- [DataCamp MOOCs](#) on data science with Python



==== Fundamental numerical methods ====

- [Systems of linear equations](#)
  - Diagonalisation and triangularisation
  - LU decomposition : factorization in triangular matrices

- Numerical intégration (integrals)
  - Simpson method and gaussian quadratures
- Root findings : equations  $f(x) = 0$ 
  - Polynomial equations
  - Dichotomy
  - Secant method, Regula falsi
  - Newton-Raphson method

==== Classical numerical methods ==== === Ordinary\_differential\_equations (ODE) ===

### Numerical solutions of ODE

- principe de discrétisation, méthode d'Euler
- Améliorations et méthodes de Runge-Kutta
- Runge-Kutta d'ordre 4
- Contrôle du pas d'intégration
- Méthodes predictor-corrector
- Méthodes d'extrapolation (Richardson, Burlish-Stoer)
- applications :
  - équations de cinétique chimique
  - Équation logistique
    -  Modèle de Verhulst,  dynamique des populations et non-linéarité
    - Bifurcation, doublements de périodes et transition vers le chaos
  - Réactions chimiques oscillantes : Belousov-Zhabotinsky, Brusselator, Oregonator
  - Modèle proie-prédateur
  - Attracteur étrange, modèle atmosphérique de Lorenz

=== Partial\_differential\_equations (PDE) === Numerical solutions of PDE

- Domaine d'application des équations : équation de diffusion, équation d'ondes, équations de Navier-Stokes
- Types de traitements numériques
- Différences finies et problèmes de diffusion
- Schémas classiques de différences finies
  - Résolutions stationnaires
  - Résolutions dépendantes du temps
- Méthodes explicites, critère de (ou d'in)stabilités et méthodes implicites

=== Eigenvalues and eigenvectors === Eigenvalues and eigenvectors

applications à des problèmes de relaxation et de population, analyse de modes normaux de vibration, PCA (principal component analysis),...

=== Non-linear systems of equations ===

- Newton-Raphson method

=== Linear and non-linear least squares approximations === Application to deconvolution (Levenberg-Marquardt algorithm)

=== Chebyshev approximation ===

=== Molecules modelisation and visualization ===

=== Minimization === Conformational problems

---

===== Additional subjects =====

- Bioinformatics and related algorithm (biochemistry, mass spectrometry,...)
- Chemistry
  - quantum calculations, optimization, molecular mechanics
  - visualization, virtual reality
- Data science, statistics (Python modules : Scipy, Pandas,...)
  - Time series analysis
  - Machine learning (Scikit-learn,...)
- Data visualization
  - boxplot, 3D, animations, graphs,...
- Sensors and interfaces, Arduino, Raspberry Pi, IoT
- Simulations
  - Agent base modelling and complex systems
  - cellular automaton
  - Simpy,...
- Digital image processing, image recognition
  - particle tracking,...

===== Références =====

- Bioinformatics
  - [Biopython](#)
  - [Rosalind.info](#), plateforme d'apprentissage par la résolution de problèmes
- Machine Learning
  - Scikit-learn
  - [12 Algorithms Every Data Scientist Should Know](#)
- Arduino
  - [Fabrication of an Economical Arduino-Based Uniaxial Tensile Tester](#), Julien H. Arrizabalaga, Aaron D. Simmons, and Matthias U. Nollert, J. Chem. Educ., 2017, 94 (4), pp 530-533 DOI: 10.1021/acs.jchemed.6b00639
- PCA (principal component analysis)
  - [Learning Principal Component Analysis by Using Data from Air Quality Networks](#), Luis Vicente Pérez-Arribas, María Eugenia León-González, and Noelia Rosales-Conrado, J. Chem. Educ., 2017, 94 (4), pp 458-464 DOI: 10.1021/acs.jchemed.6b00550

===== Books =====

- [Solving Differential Equations in R](#), chez Springer, et en version électronique sur [SpringerLink](#)
  - ...
- 

===== Synopsis (français) ===== Pré-requis mathématiques =====



==== Base de la programmation ====

==== Méthodes numériques de base ====

- [Systèmes d'équations linéaires](#)
  - Diagonalisation et triangularisation
  - Décomposition LU en matrices triangulaires
- [Intégration numérique](#)
  - Simpson et quadratures gaussiennes
- [Résolutions d'équations du type  \$f\(x\) = 0\$](#) 
  - Équations polynomiales
  - Recherche dichotomique
  - Méthode de la sécante
  - Méthode de Newton-raphson

==== Méthodes numériques usuelles ==== [Équations différentielles ordinaires](#)

### [Résolutions numériques des ODE](#)

- principe de discrétisation, méthode d'Euler
- Améliorations et méthodes de Runge-Kutta
- Runge-Kutta d'ordre 4
- Contrôle du pas d'intégration
- Méthodes predictor-corrector
- Méthodes d'extrapolation (Richardson, Burlish-Stoer)
- applications :
  - équations de cinétique chimique
  - Équation logistique
    -  [Modèle de Verhulst](#),  [dynamique des populations](#) et non-linéarité
    - Bifurcation, doublements de périodes et transition vers le chaos
  - Réactions chimiques oscillantes : Belousov-Zhabotinsky, Brusselator, Oregonator
  - Modèle proie-prédateur
  - Attracteur étrange, modèle atmosphérique de Lorenz

==== Équations aux dérivées partielles ==== [Résolutions numériques des équations aux dérivées partielles](#)

- Domaine d'application des équations : équation de diffusion, équation d'ondes, équations de Navier-Stokes
- Types de traitements numériques
- Différences finies et problèmes de diffusion
- Schémas classiques de différences finies
  - Résolutions stationnaires
  - Résolutions dépendantes du temps
- Méthodes explicites, critère de (ou d'in)stabilités et méthodes implicites

==== Problèmes aux valeurs propres ==== [Valeurs propres et vecteurs propres](#)

applications à des problèmes de relaxation et de population, analyse de modes normaux de vibration, PCA (principal component analysis),...

=== Systèmes d'équations non linéaires === Méthode de Newton-Raphson

=== Approximation par moindres carrés linéaires et non-linéaires === application à la déconvolution

=== Approximations de Tchébychev ===

=== Modélisation et visualisation de molécules ===

=== Minimisation === problèmes conformationnels

---

===== Thèmes additionnels =====

- Bioinformatique et algorithmes spécifiques
  - Chimie
    - calculs quantiques, de minimisation, de mécanique moléculaire
    - représentations
  - Data science, statistiques (bibliothèque Python Pandas,...)
    - Time series analysis
    - Machine learning (Scikit-learn,...)
  - Data visualization
    - boxplot, 3D, animations, graphes,...
  - Capteurs et interface, Arduino, Raspberry Pi, IoT
  - Simulations
    - Agent based modelling et systèmes complexes
    - Automates cellulaires
    - Simpy,...
  - Traitement d'image
    - particle tracking,...
- 

===== Références =====

- Bioinformatique
  - [Biopython](#)
  - [Rosalind.info](#), plateforme d'apprentissage par la résolution de problèmes
- Machine Learning
  - Scikit-learn
  - [12 Algorithms Every Data Scientist Should Know](#)
- Deep Learning
  - TensorFlow
  - [Keras Tutorial: Deep Learning in Python](#)
- Arduino
  - [Fabrication of an Economical Arduino-Based Uniaxial Tensile Tester](#), Julien H. Arrizabalaga, Aaron D. Simmons, and Matthias U. Nollert, J. Chem. Educ., 2017, 94 (4), pp 530-533 DOI: 10.1021/acs.jchemed.6b00639
- Principle Component Analysis :
  - [Principle Component Analysis in Python](#)
- K-Means

- [K-means Clustering in Python](#)

==== Livres ====

- [Solving Differential Equations in R](#), chez Springer, et en version électronique sur SpringerLink
- ...

From:

<https://dvillers.umons.ac.be/wiki/> - **Didier Villers, UMONS - wiki**

Permanent link:

<https://dvillers.umons.ac.be/wiki/teaching:methcalchim:start?rev=1507875932>

Last update: **2017/10/13 08:25**

