Computational Materials Science (計算材料学特論)

http://d2mate.mdxes.iir.isct.ac.jp/D2MatE/D2MatE_programs.html?page=cms

Lecture materials for numerical analyses (by Kamiya)
数値解析に関する講義資料・pythonプログラム (神谷担当分)

Update News:

- **July 03, 2026, 05:56:** Lecture materials for July 3 has been updated: course_materials.zip
- **July 02, 2026, 11:43:** Lecture materials for July 3 has been uploaded
- **June 30, 2026, 12:13:** Final version: Lecture materials for June 30 has been updated: course_materials.zip
- **June 26, 2026, 11:31:** Final version: Lecture materials for June 26 has been update
- **June 23, 2026, 11:42:** Final version: Lecture materials for June 23 has been update
- **June 19, 2026, 10:49:** Final version: Lecture materials for June 19 has been update

FY2026

#06 June 30, 2026: Nonlinear optimization (非線形最適化), Fourier transform (フーリエ
(応用)

Course materials (Lecture slides and python programs):

- course_materials.zip

5-8min audio guide:

- 日本語: ▶ 0:00 / 4:47 (VOICEVOX 四国めたん&ずんだもん)
- English: ▶ 0:00 / 6:00

#05 June 26, 2026: Solution of equations (方程式の解法), Nonlinear optimization (非線形最適化)

Course materials (Lecture slides and python programs):

- course_materials.zip
- Slide files and Videos (monologue): tutorial web

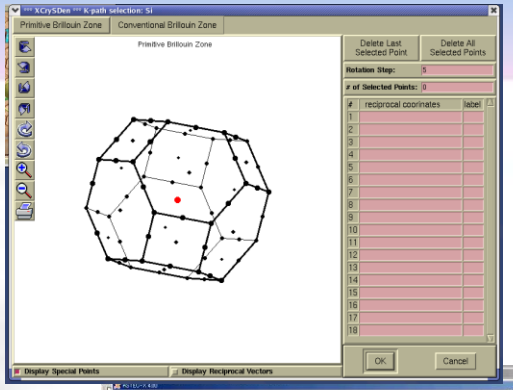
We would wait for five minuities (i.e., till 8:55).
In meantime

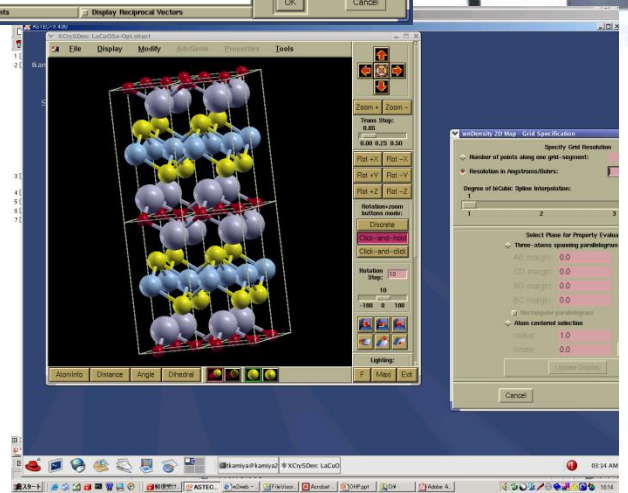
- **download the latest lecture materials (uploaded this morning)**
- **hear the short audio guide. English and Japanese versions available**

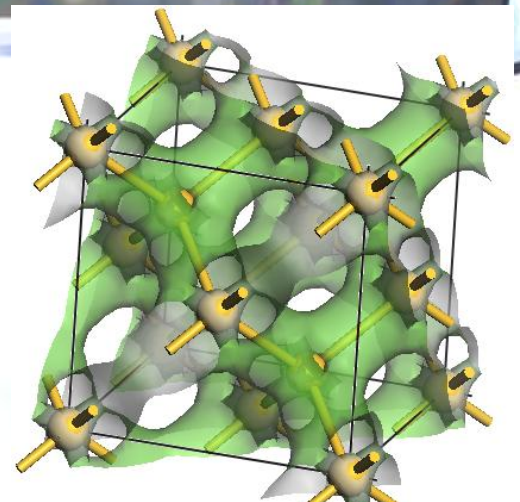
MAT.C505

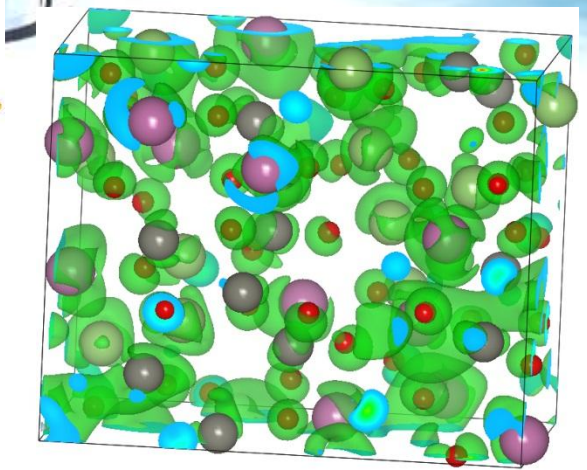
Computational Materials Science
計算材料学特論

Toshio Kamiya
神谷利夫









Class Schedule

Lecture materials (Kamiya's part): **http://d2mate.mdxes.iir.isct.ac.jp/D2MatE/?page=cms**

授業　6月10日(水)～7月28日(火), 7月30日(木)　月曜の授業　7月23日(木)　期末試験・補講　7月29日(水), 7月31日(金)～8月6日(木)

#01 June 12 (Fri)　Kamiya (Fundamentals of computer, Sources of error (コンピュータの基礎、誤差), Numerical differentiation (数値微分))

#02 June 16 (Tue)　Kamiya (Numerical differentiation (数値微分), Numerical integration (数値積分), Differential equation (微分方程式))

#03 June 19 (Fri)　Kamiya (Differential equation (微分方程式), Molecular dynamics (分子動力学法), Interpolation (補間), Smoothing (平滑化))

#04 June 23 (Tue)　Kamiya (Smoothing (平滑化), Linear least-squares method (線形最小二乗法), Numerical solutions of equations (方程式の数値解法))

#05 June 26 (Fri)　Kamiya (Numerical solutions of equations (方程式の数値解法), Nonlinear optimization (非線形最適化))

#06 June 30 (Tue)　Kamiya (Nonlinear optimization (非線形最適化), Fourier transform (フーリエ変換), Matrix (行列))

#07 July 3　(Fri)　Kamiya (Matrix (行列), Montecarlo methods, Baysian regressions, etc.)

#08 July 7　(Tue)　Sasagawa (Review of quantum theory 1: 量子論おさらい1)

#09 July 10　(Fri)　Sasagawa (Review of quantum theory 2: 量子論おさらい2)

#10 July 14 (Tue)　Sasagawa (First principles calculations: basics 1 第一原理計算：基礎1)

#11 July 17 (Fri)　Sasagawa (First principles calculations: basics 2 第一原理計算：基礎2)

#12 July 2 (Fri)　Sasagawa (First principles calc.: applications 1 第一原理計算：応用1)

#13 July 24 (Fri)　Sasagawa (First principles calc.: applications 2 第一原理計算：応用2)

#14 July 28 (Fri)　Sasagawa (Classical and Quantum Computers 古典および量子コンピュータ)

Explanation of the answers

課題解答の解説

PROBLEM, June 30

- **Answer in English or Japanese**
- **Submit electronic file(s) via LMS by midnight on July 1st**
  (If LMS doesn't work, send the files to kamiya.t.aa@m.titech.ac.jp.
  In this case, file name must include your STUDENT ID and FULL NAME)
- Common formats (.pdf, .txt., .docx, .xlsx, .pptx) are acceptable, but NO APPLE-ONLY files

PROBLEM: Answer can be in Japanese or English

(i) Find (x, y) to minimize $\exp(-x^2) * \sin(x+y)$ by your-chosen algorisms
hint: use SD method with fixed alpha
if you will approximate df/dx and df/dy, use central differences

Optional:

(i) Propose if you have any other numerical analysis you want to learn in Computational Materials Science

(ii) Propose if you have any python program (should be simple) you want to learn

PROBLEM, June 30

$f(x,y) = \exp(-x^2) * \sin(x+y)$

df/dx = exp(-x2) * (-2x * sin(x+y) + cos(x+y))

df/dy = exp(-x2) * cos(x+y)

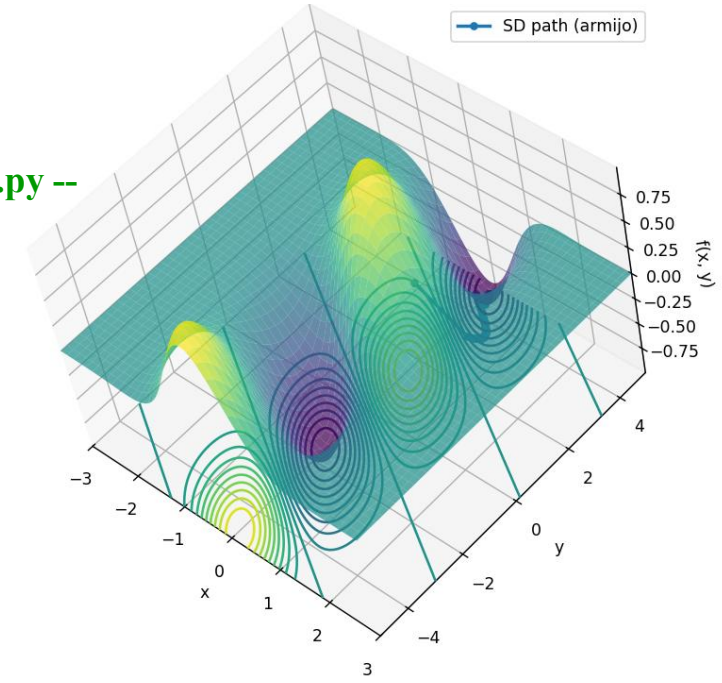
See minimize_answer.xlsx

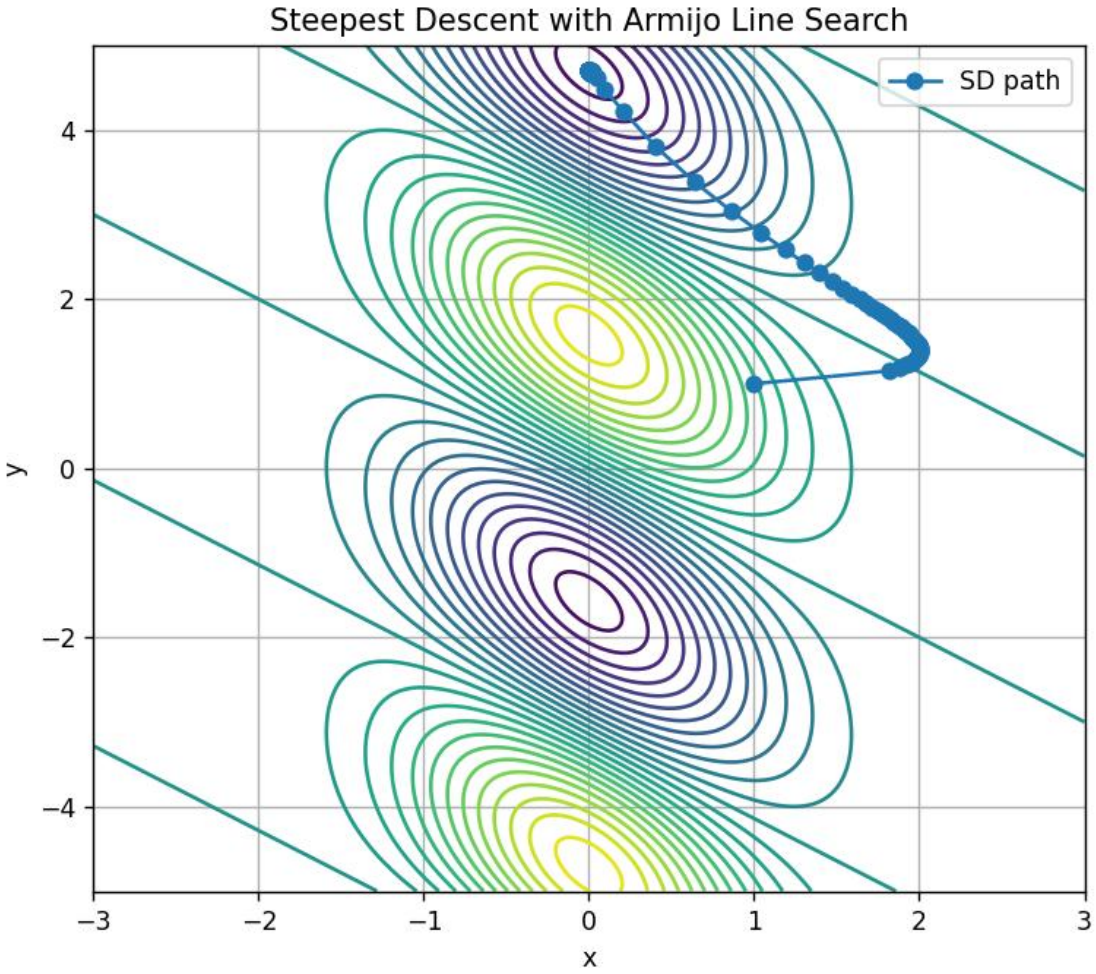
Program that combines SD method and Armijo condition

> python sd_armijo_assignment.py --mode armijo

```
def armijo_line_search(p, d, alpha0=1.0, rho=0.5, c=1.0e-4):
    alpha = alpha0
    g = grad_f(p)
    while f(p + alpha*d) > f(p) + c*alpha*np.dot(g, d):
        alpha *= rho
    return alpha
```

> python sd_armijo_assignment.py --mode plot3d





Notice

I used ChatGPT-5.5 to prepare several code examples
in response to your requests.

These programs are useful starting points for learning,
but they are not fully validated research codes.

Please do not trust the code blindly.
Run it, test it, understand it, and verify it by yourself.

Requests for additional methods

- For each type of problem, should we consider the methods introduced in this lecture to be sufficiently comprehensive for practical use, or should we regard them as only a small selection from the many methods available?
- Program that combines SD method and Armijo condition
- Short Python comparisons of bisection, secant, Newton-Raphson, and Brent methods applied to the same equation
- How to choose smoothing algorithms
- How do the numerical methods we study adapt to guarantee the conservation of physical invariants, like energy or angular momentum, over long timescales?
- Exact diagonalization / exact solution for Hubbard / Heisenberg model
- (Lagrange multipliers) Constrained optimization
  For constraints of symmetry, total sum
- Bayesian optimization
- Bayesian inversion, Markov Chain Monte Carlo (MCMC) methods, Metropolis-Hastings algorithm. Monte Carlo method
- Neural network
- How to read csv, xlsx, and other common files
- Characterize ICP-MS spectrum

Very important and useful question

Q: Do the methods taught in this lecture sufficiently cover practical problems?
Or, do we have many other better methods available?

A:

- This lecture covers classical but practical algorithms that you can use for your research
- There is no universally best method.
  The best method depends on the problem and you may find more advanced methods from recent literature, or from different fields like machine learning, AI, etc.

You can ask generative AI with care

(i) Ask: How can I solve problem *XXX* in Python?

(ii) Ask it to generate Python code and simple test cases.

(iii) Ask for textbooks, papers, and accessible references.

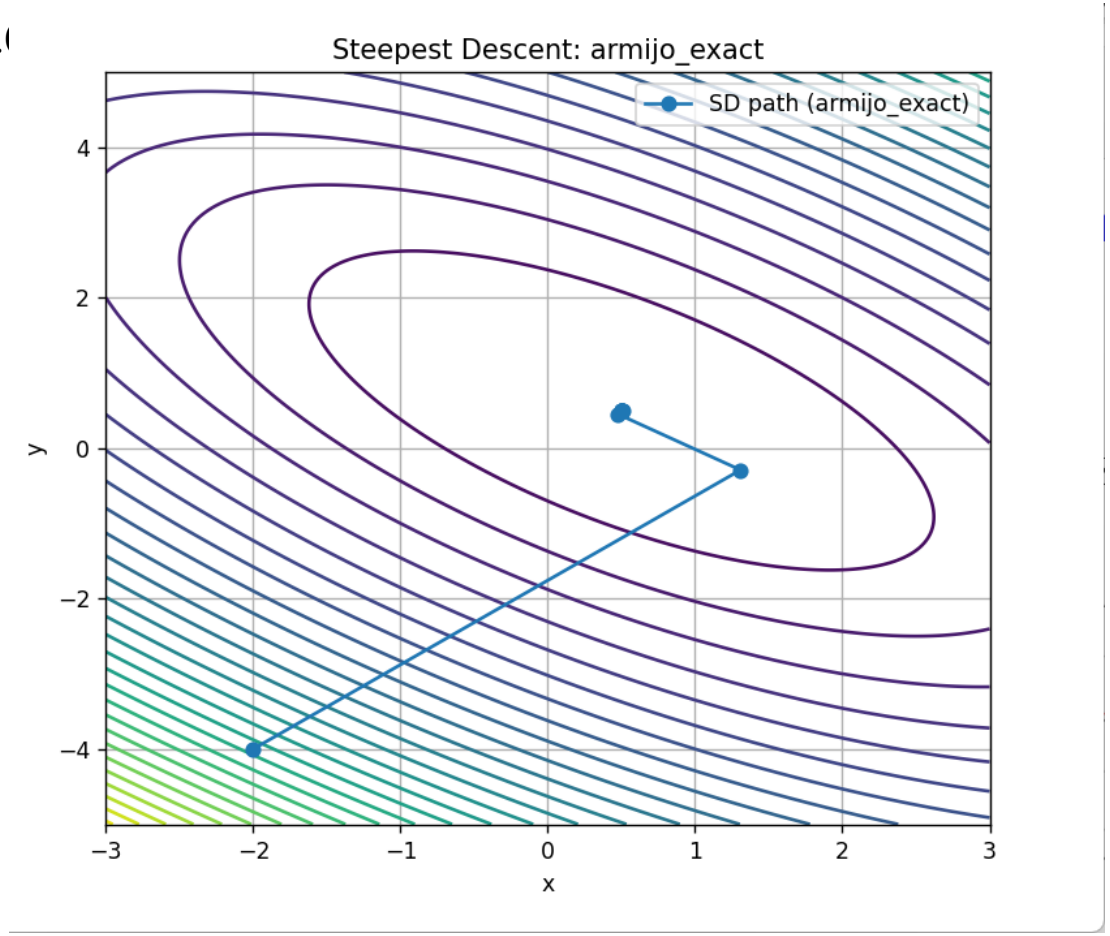
(iv) Verify the algorithm, implementation, and references yourself.

Program that combines SD method and Armijo condition

f(x, y) = 5(x+y-1)^2 + (x-y)^2

> python sd_armijo.py --mode armijo --x0=-2 --y0=-4

```
def armijo_line_search(p, d, alpha0=1.0, rho=0.5, c=1.0e-4, alpha_min=1.(
    n_backtrack = 0
    while not armijo_ok(p, d, alpha, c=c):
        alpha *= rho
        n_backtrack += 1
        if alpha < alpha_min:
            break
    return alpha, {"backtracks": n_backtrack}
```



Comparisons of bisection, secant, Newton-Raphson, and Brent methods

> python equation_compare.py

Newton initial value = 0.0, Newton damping factor = 0.0, Initial interval = [0.0, 1.0]

Newton-Raphson method

Iter 7: x: 1.254347432222 => 1.254347432221, dx = -1.148e-12

Bisection method

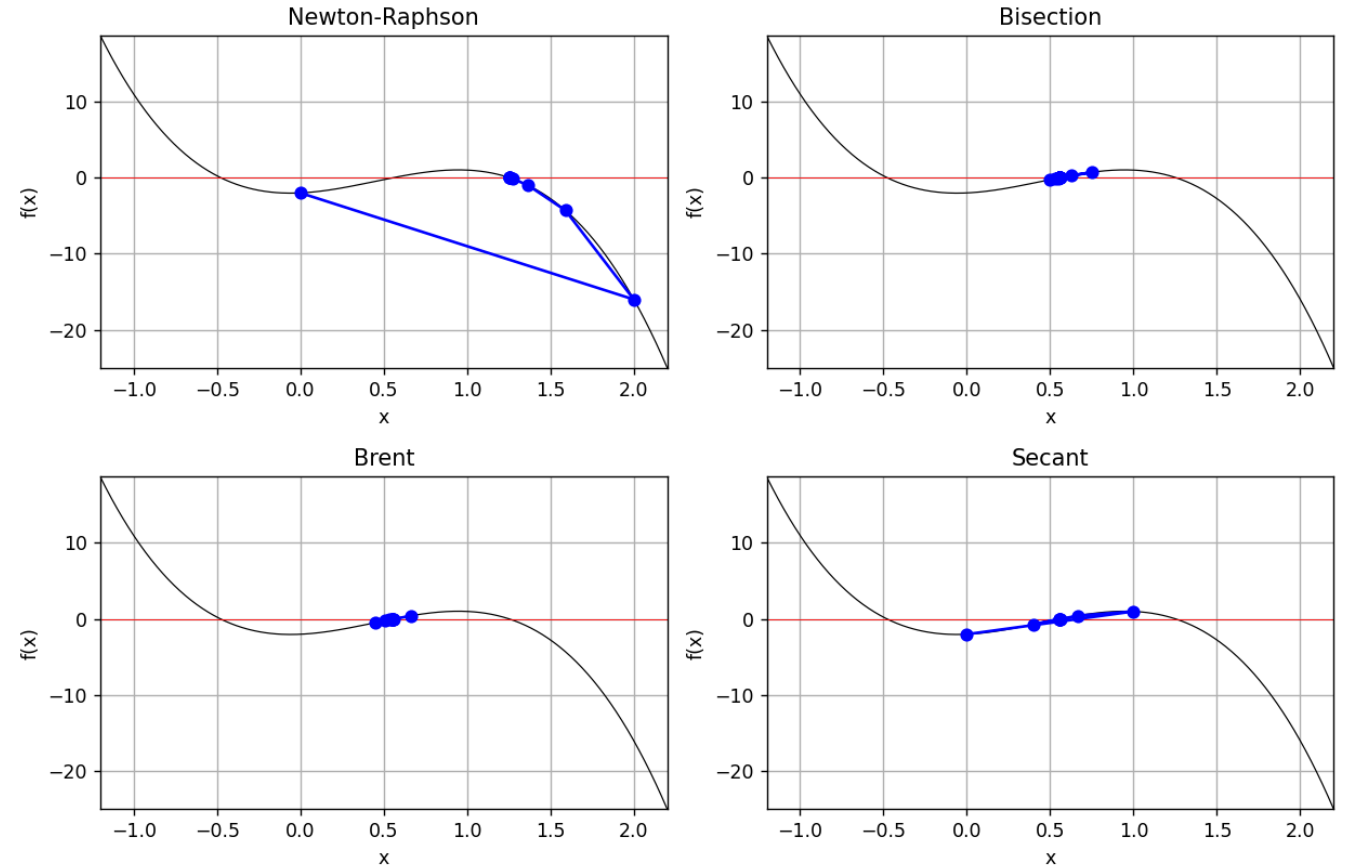
Iter 28: [0.556505639106, 0.556505642831],
mid = 0.556505640969, f(mid) = -1.746e-11

Brent method

Iter 9: x = 0.556505640973, f(x) = -2.22e-16

Secant method

Iter 6: x: 0.556505641087 => 0.556505640973,
dx = -1.145e-10, f(x) = -2.22e-16



Brent's method: improvement of bisection method

Brent's method automatically selects the most appropriate step from the data already obtained during the search:
- inverse quadratic interpolation
- linear interpolation (secant step)
- Bisection

Brent法は、既に探索したデータから、
・二次関数逆補間
・一次関数補間 (割線法)
・二分法
で最適なものを自動選択します

Choose smoothing by data type and by what should be preserved

Note that the filter convolution method is a practical implementation of weighted smoothing moving average and function convolution smoothing.

- For many signal-processing data: **FFT smoothing** (low pass filter)
- Gaussian/Lorentzian **convolution**: For discrete point integration data (e.g., DOS), experimental spectrum etc
  often called 'smearing', 'broadening', etc.
- **Trailing weighted moving smoothing**: Used for stock market analysis with past data
  (10 days / 3 months / 200 days moving average
  10日移動平均, 75日移動平均, 200日移動平均)
- **Local polynomial fit smoothing**:
  Good for many experimental data processing
- **Global least-squares fitting**:
  For cases where number of data points is small
  but a reasonable parameterized function model is available

The best smoothing method depends on what should be preserved:
peak position, peak width, area, trend, phase, or derivatives.

How to guarantee the conservation of conserved quantities, like energy or angular momentum, over long timescales

1. Choose a **symplectic algorithm** such as Verlet method, which may result in small short-term oscillations but is good to preserve long-term energy conservation
2. Choose **fine MD conditions** such as satisfactory small MD time step, high precision of force evaluations

If these are not enough, you may apply artificial corrections like a scaling method

1. **Correct center-of-mass drift** to maintain momentum conservation
2. **Scaling** velocities so as to recover total energy, but you must remember this artificial correction **breaks other conservation (angular momentum conservation etc) and changes dynamics**
3. **Check continuity, smooth connection, etc for properties of your interest, in particular at the scaling steps**

Exact diagonalization

In large-size matrix diagonalization problems like quantum simulations:
Usually, only low-energy eigenstates are calculated to reduce computational cost.

Plain wave basis method for DFT

- # of PW basis is ~ 1,200 for Ecut = 400 eV and a = 0.4 nm
- Usually, only occupied states and a limited number of unoccupied states are calculated.
- Exact (Full) diagonalization: calculate all eigen states
  Important for HF, GW, optical calculations where many unoccupied states are required
- Hamiltonian matrix is not sparse
  Davidson method, RMM-DIIS, etc may be used

Hubbard / Heisenberg model

- # of configurations faces combinatorial explosion issue for large-size model
- **Sparse matrix**
- Exact diagonalization: Treat the full finite-size Hilbert space without mean-field approximation.
  Lanczos method is widely employed

Optimization under Constraints

Lagrange multiplier method (Lagrangeの未定乗数法):

A method to analytically optimize a function $f(x_i)$ under constraints such as $g(x_i) = 0$, $h(x_i) = 0$, *etc.*

Introduce Lagrange multipliers α and β:

$$L(x_i, \alpha, \beta) = f(x_i) - \alpha g(x_i) - \beta h(x_i)$$

Then, finding the stationary point of L with respect to x_i, α, β.

In **practical numerical calculations**, we either solve these stationary conditions directly,

or solve the problem as a **constrained optimization problem**, for example:

minimize $\boldsymbol{f(x_i)}$ subject to $\boldsymbol{g(x_i) = 0}$, $\boldsymbol{h(x_i) = 0}$

One way to impose constraints is the **penalty method**:

minimize $\boldsymbol{S(x_i) = f(x_i) + \rho_1 err_g(x_i)^2 + \rho_2 err_h(x_i)^2}$

Here, err_g and err_h are functions that return errors when the constraints are violated.

The penalty parameter ρ_1 and ρ_2 should be chosen appropriately by checking both convergence and whether the constraints are satisfied.

For inequality constraints, study the **KKT conditions**.

制約条件下での最適化

Lagrangeの未定乗数法:

関数 $f(x_i)$ を、$g(x_i) = 0, h(x_i) = 0, etc$ の制約条件で解析的に説く方法

未定乗数 α, β を導入し、

$$L(x_i, \alpha, \beta) = f(x_i) - \alpha g(x_i) - \beta h(x_i)$$

をx_i, α, βで偏微分して極値となる時のx_iを求めることと同値問題。

ただし、実用的な数値計算では、この条件式を直接解くか、制約付き最適化法として解く。

例えば、minimize $f(x_i),\ g(x_i) = 0, h(x_i) = 0$ を実行する。

制約条件を課す方法としてpenalty法がある。

$$\text{minimize } S(x_i) = f(x_i) + \rho_1 err_g(x_i)^2 + \rho_2 err_h(x_i)^2$$

ここで、err_g, err_hは、制約条件を外れた時に誤差を返す関数。

penalty parameter ρ_1, ρ_2 は収束と制約条件を満たしているかから適切に選択する

不等式制約条件の場合は、KKT条件を勉強してください

Python code: penalty_minimization.py

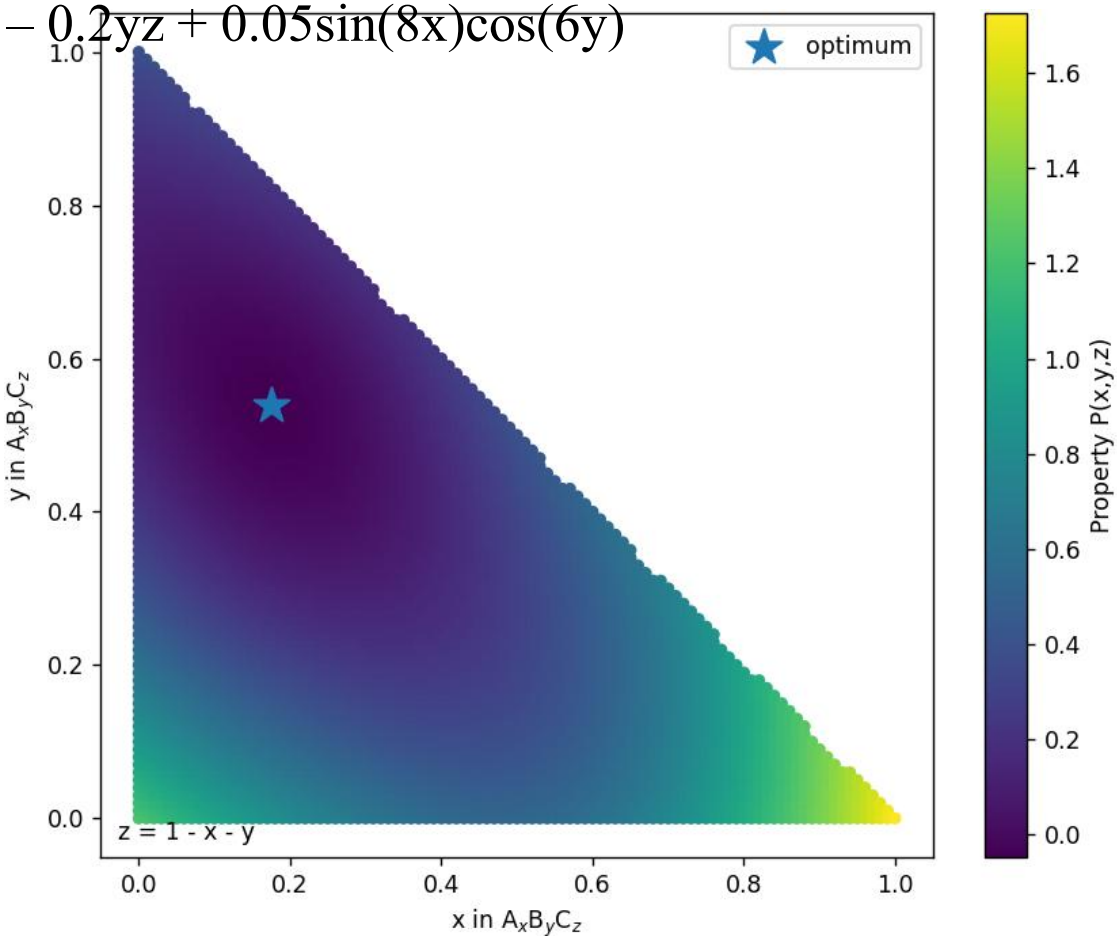
Model:

- Ternary compound $A_xB_yC_z$
- Constraint: x + y + z = 1
  x, y, z >= 0
- Objective property function to be minimized:
  $p(x,y,z) = 2(x - 0.2)^2 + (y - 0.55)^2 + (1.5(z - 0.25)^2 + 0.3xy - 0.2yz + 0.05\sin(8x)\cos(6y)$

Optimized composition:

x = 0.175761
y = 0.538572
z = 0.285668
x + y + z = 1.0000003474

Values:
Property P(x,y,z) = -0.0482830020
Penalty objective S = -0.0482830019
Constraint error = 3.474e-07



Neural network

Common libraries:

- Scikit-learn MLP
  Simple trial for small tabular data

For larger and more complex problems:

- currently recommended: PyTorch
  Widely used in research; flexible and easy to customize

- TensorFlow / Keras
  Standard deep-learning framework, useful for practical applications

How to read common files

Text (space / comma etc separated, simple format)

- numpy.loadtxt()

Excel (.xlsx), CSV

- pandas is a convenient package combined with data analysis

Excel (.xlsx)

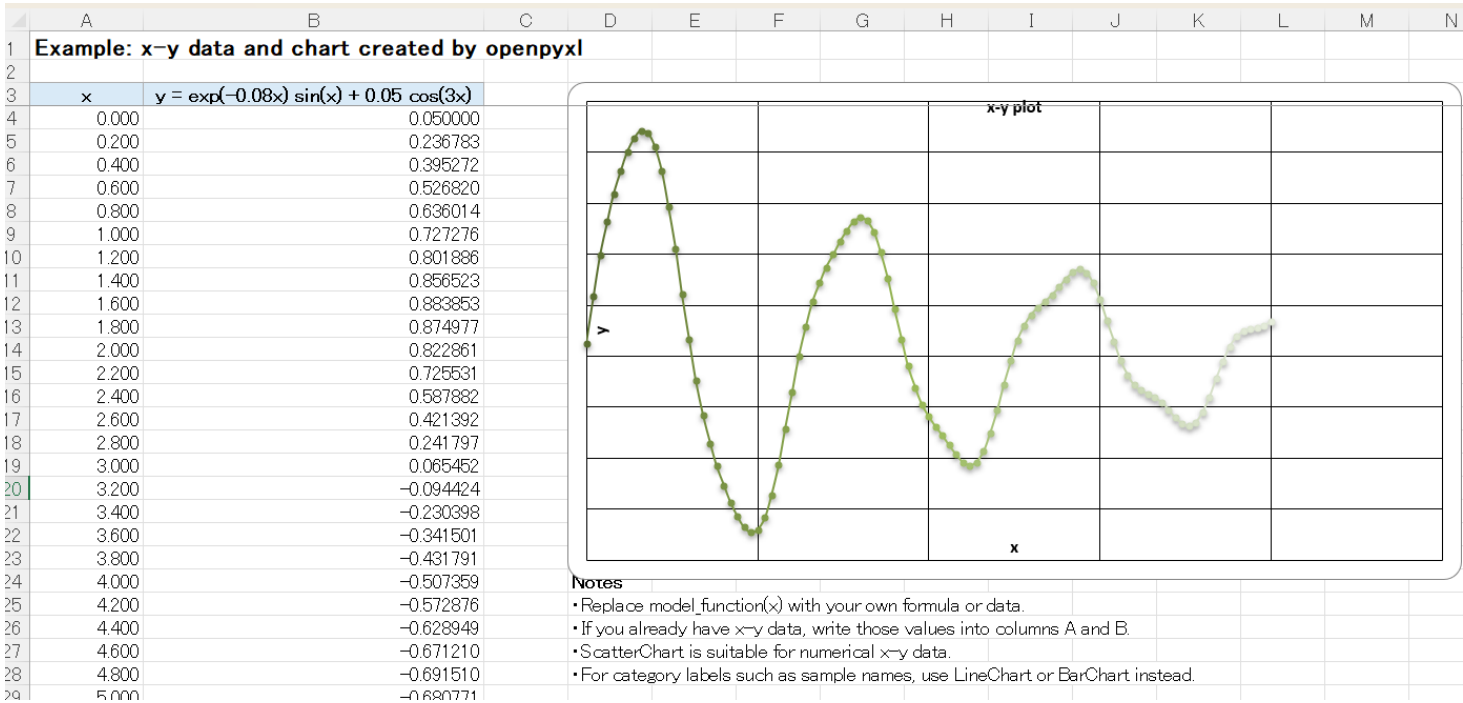
- openpyxl: read .xlsx by specifying cells,
  can directly edit Excel sheets, cell styles, formulas etc.
  A back-end of pandas

How to make Excel file with graph

Limited to specific applications;
not worth to learn detailed API's of openpyxl and Visual Basic Applications
Ask generative AI to make python programs

> make_graph_xlsx.py

output: xy_chart_example.xlsx



Example: x–y data and chart created by openpyxl

x	y = exp(−0.08x) sin(x) + 0.05 cos(3x)
0.000	0.050000
0.200	0.236783
0.400	0.395272
0.600	0.526820
0.800	0.636014
1.000	0.727276
1.200	0.801886
1.400	0.856523
1.600	0.883853
1.800	0.874977
2.000	0.822861
2.200	0.725531
2.400	0.587882
2.600	0.421392
2.800	0.241797
3.000	0.065452
3.200	−0.094424
3.400	−0.230398
3.600	−0.341501
3.800	−0.431791
4.000	−0.507359
4.200	−0.572876
4.400	−0.628949
4.600	−0.671210
4.800	−0.691510
5.000	−0.690771

Notes
- Replace model_function(x) with your own formula or data.
- If you already have x–y data, write those values into columns A and B.
- ScatterChart is suitable for numerical x–y data.
- For category labels such as sample names, use LineChart or BarChart instead.

Detect onset in ICP-MS time-series data

Problem:

- Determine when Pt dissolution starts in ICP-MS data
- The signal contains background noise and repeated peaks

Simple and practical method:

- Estimate background mean and standard deviation
- Define detection threshold: S_{th} = background mean + 3 × BG std
- Onset time:

  first time where the signal exceeds S_{th} **for several consecutive data points**

  Use smoothing to stabilize detection:

  Polynomial fit smoothing (Savitzky–Golay filter) is a good first choice

Important checks:

- Avoid false detection from a single noise spike
- If the background drifts, estimate a local background
- Compare the detected onset with the original raw data

Requests for additional methods

- Bayesian optimization
- Bayesian inversion, Markov Chain Monte Carlo (MCMC) methods, Metropolis-Hastings algorithm

Brief explanation will be given at the end of this lecture

Tutorials [in Japanese]: http://d2mate.mdxes.iir.isct.ac.jp/D2MatE/?page=tutorial&id=data

Bayesian statistics
Bayesian inversion
(Bayesian regression)

Gaussian process
Bayesian optimization