KEMS448 — Advanced lab-works in physical chemistry

Numerical derivatives and integration, 2 ECTS

Toni Kiljunen 2–2008

Abstract

Numerical differentiation by difference approximations, numerical integration by Newton-Cotes quadratures, and differential equation solving by Taylor and Runge-Kutta methods comprise the curriculum of this exercise. Numerical interpolation and solving systems of linear equations are additional topics covered by the given literature. Vibrational states and energies are solved for a diatomic I_2 molecule using Morse-type potential functions and compared to corresponding analytical values in the application part. Electronic excitation spectrum is calculated for the $B \leftarrow X$ transition. Intensities of fundamental vibrational transitions are compared to overtones. The numerical procedures are implemented using the MATLAB program throughout.

1 Introduction

In order to use numerical methods one needs to have an idea about their mathematical foundations, applicability, and accuracy. The computer program must be flawless and robust, i.e., capable of detecting singularities that cannot be solved. The result of a numerical evaluation is usually approximative and dependent on methodological, rounding, and accumulative errors.

Analytical differentiation and integration differ in that the latter is not a straightforward, mechanical procedure. Numerical integration, on the other hand, is a stable operation in contrast to numerical differentiation. The difference quotient used in the latter makes it troublesome because significant numbers/digits may become lost.

Ordinary differential equations (ODE) and differential equation systems are frequent models in studies of physical phenomena. Here we examine an initial value problem of a first order ODE system (such as that encountered in reaction kinetics or in a time-dependent Schrödinger equation). In the application part a time-independent Schrödinger equation, which is an eigenvalue problem, is solved by matrix diagonalization.

2 Numerical differentiation

Let the values of a function f be known at points $x-h$, x, and $x+h$. An estimation for $f'(x)$ is calculated by the difference quotient, without an analytical formula for the derivative.

Forward difference:

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} =: D_+(h)\,, \tag{1}$$

backward difference:

$$f'(x) \approx \frac{f(x)-f(x-h)}{h} =: D_-(h)\,, \tag{2}$$

central difference:

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} =: D_0(h)\,. \tag{3}$$

Higher derivatives can be approximated correspondingly:

$$f''(x) \approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2}\,. \tag{4}$$

Exercise: Using $D_0(h)$ derive a formula for $(\partial^2 f(x,y)/\partial x \partial y)$.

Error estimation for forward difference approximation can be obtained from a Taylor expansion:

$$e_h(x) \;=\; \frac{f(x+h)-f(x)}{h} - f'(x)$$

$$
\begin{aligned}
&= \frac{f(x) + f'(x)h + \frac{1}{2}f''(\xi)h^2 - f(x)}{h} - f'(x) \\
&= \frac{1}{2}f''(\xi)h\,, \quad \xi \in]x, x+h[\,,
\end{aligned}
$$

where f'' and ξ are unknown, but the error is seen to be of order $\mathcal{O}(h)$. For the central difference and second derivative we obtain $\mathcal{O}(h^2)$:

$$
\begin{aligned}
\frac{f(x+h) - f(x-h)}{2h} &= f'(x) + b_1 h^2 + b_2 h^4 + b_3 h^6 + \ldots \\
&= f'(x) + \mathcal{O}(h^2)\,.
\end{aligned}
$$

These mean that $e_h(x) \to 0$ as $h \to 0$. In practice one needs to consider round-off errors, so the best $h \neq 0$.

Richardson's extrapolation can be used for increasing the accuracy. Factor b_1 in the central difference

$$
D_0(h) = f'(x) + b_1 h^2 + \mathcal{O}(h^4) \tag{5}
$$

is usually unknown, but it is independent of h. By doubling the h we obtain

$$
D_0(2h) = f'(x) + b_1(2h)^2 + \mathcal{O}(h^4) = f'(x) + 4b_1 h^2 + \mathcal{O}(h^4)\,. \tag{6}
$$

Eliminating the constant b_1 from previous equations leads to

$$
D_R(h) := \frac{4}{3}D_0(h) - \frac{1}{3}D_0(2h) = f'(x) + \mathcal{O}(h^4)\,. \tag{7}
$$

Thus we obtained a $\mathcal{O}(h^4)$-approximation for f'. The process could be continued further on.

If the values of a function f were known only at predetermined points, the interpolating *spline* function could be differentiated [1, 2]. This is useful especially when there are lots of points for which the derivative is needed or when the points are not equally spaced.

3 Numerical integration

Newton–Cotes quadratures are based on interpolating polynomials. Let us consider calculating the integral

$$
\int_a^b f(x)dx \tag{8}
$$

by setting $a = x_0$ and $h = b - a$. Variable change $x = x_0 + sh$ leads to

$$
\begin{aligned}
\int_a^b f(x)dx &= \int_0^1 f(x_0 + sh)h\,ds \\
&= h\int_0^1 f(x_0)ds + h^2 \int_0^1 s f'(x_0 + \theta h)ds \\
&= hf_0 + h^2 \int_0^1 s f'(x_0 + \theta h)ds\,,
\end{aligned} \tag{9}
$$

where $f(x_0 + sh) = f_0 + shf'(x_0 + \theta h)$ and $\theta = \theta(s) \in]0, 1[$ are used. The integral can be processed further:

$$\begin{aligned} \int_0^1 sf'(x_0 + \theta h)ds &= f'(x_0 + \xi h) \int_0^1 sds \\ &= \frac{1}{2} f'(x_0 + \xi h)\,, \quad \xi \in]0, 1[\,. \end{aligned} \tag{10}$$

Thus the simplest way to to approximate an integral is the so called *rectangular rule*:

$$\begin{aligned} \int_a^b f(x)dx &= hf_0 + \frac{h^2}{2} f'(x_0 + \xi h) \\ &= hf_0 + e[f]\,. \end{aligned} \tag{11}$$

More complicated formulas can be derived similarly. The given points are $x_i = x_0 + ih$ $(i = 0\,, 1\,, \ldots, k)$, where $a = x_0$, $b = x_k$, and $h = (b - a)/k$. The integral is now approximated by at most k-degree (Newtonian) interpolating polynomial p_k[1]

$$\int_a^b p_k(x)dx\,, \quad p_k(x) = f(x_i) =: f_i\,, \quad i = 0, 1, \ldots, k\,. \tag{12}$$

By changing the variable $x = x_0 + sh$ and expanding p_k using forward differences[2] we obtain[3]

$$\begin{aligned} \int_a^b p_k(x)dx &= \int_0^k p_k(x_0 + sh)h\,ds \\ &= h \int_0^k \Big[f_0 + \Delta f_0 s + \frac{1}{2!}\Delta^2 f_0 s(s-1) + \ldots \\ &+ \frac{1}{k!}\Delta^k f_0 s(s-1)\cdots(s-k-1) \Big]\, ds\,. \end{aligned} \tag{13}$$

This integral is easy to calculate, and by changing the k different Newton–Cotes quadratures are obtained. The formulas are *closed*, because the interpolating polynomial interpolates the f at the end-points of $[a, b]$.

Choosing $k = 1$ leads to the *trapezoidal rule*

$$\begin{aligned} \int_a^b f(x)dx &\approx h \int_0^1 (f_0 + \Delta f_0 s)\, ds \\ &= hf_0 + h \int_0^1 (f_1 - f_0)s\, ds \\ &= \frac{h}{2}(f_0 + f_1)\,, \end{aligned} \tag{14}$$

which is accurate for $f'' = 0$, i.e., f is a line. Choosing $k = 2$ gives the *Simpson's rule*

$$\int_a^b f(x)dx \approx \frac{h}{3}(f_0 + 4f_1 + f_2) \tag{15}$$

---

[1] e.g. $f_0 = c_0 + c_1 x_0 + c_2 x_0^2 + \cdots + c_k x_0^k$, $f_1 = c_0 + c_1 x_1 + c_2 x_1^2 + \cdots + c_k x_1^k$, etc.

[2] Difference $\Delta^i f_j = f_j$, when $i = 0$ and $\Delta^i f_j = \Delta^{i-1} f_{j+1} - \Delta^{i-1} f_j$, when $i > 0$.

[3] $p_k(x_0 + sh) = p_{k-1}(x_0 + sh) + \Delta^k f_0 \begin{pmatrix} s \\ k \end{pmatrix}$.

and choosing $k = 3$ gives the *Simpson's 3/8 rule*

$$\int_a^b f(x)dx \approx \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3)\,. \tag{16}$$

Both are accurate for at most third degree polynomials. The methodological error is of order $\mathcal{O}(h^5)$. Another possibility is to interpolate f only at the internal points of $[a, b]$: $x_i = x_0 + ih$ $(i = 0, \ldots, k)$, where $x_0 = a + h$, $x_k = b - h$, and $h = (b - a)/(k + 2)$. The end-points are $x_{-1} = a$ and $x_{k+1} = b$. The integral is now approximated by an *open* Newton–Cotes formula

$$\int_a^b p_k(x)dx = h\int_{-1}^{k+1}\left[f_0 + \Delta f_0 s + \ldots + \frac{1}{k!}\Delta^k f_0 s(s-1)\cdots(s-k-1)\right]ds\,. \tag{17}$$

Choosing $k = 0$ leads to the formula known as *central point rule*:

$$\begin{aligned}\int_a^b f(x)dx &= h\int_{-1}^{1} f_0 ds + \frac{h^3}{2!}f''(x_0+\xi h)\int_{-1}^{1} s^2\,ds \\ &= 2hf_0 + \frac{h^3}{3}f''(x_0+\xi h)\,, \quad \xi \in]-1, 1[\,.\end{aligned} \tag{18}$$

Low-order Newton–Cotes quadratures yield non-accurate results, if the integration range is too long. High-order polynomials on the other hand tend to oscillate. Therefore, the range of integrations is divided into **subintervals**, where the integration is carried out separately:

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1}\int_{x_i}^{x_i+1} f(x)dx\,, \tag{19}$$

where the range $[a, b]$ is divided to n subintervals $[x_i, x_{i+1}]$ of length h, where $x_i = x_0 + ih$ $(i = 0, 1, \ldots, n)$, $x_0 = a$, $x_n = b$, and $h = (b - a)/n$. Applying the trapezoidal rule for each of the subintervals, we obtain $(\xi_i \in]0, 1[)$

$$\begin{aligned}\int_a^b f(x)dx &= \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{n-1} + f_n) \\ &- \frac{h^3}{12}[f''(x_0+\xi_0 h) + f''(x_1+\xi_1 h) + \ldots + f''(x_{n-1}+\xi_{n-1}h)]\,.\end{aligned} \tag{20}$$

For the Simpson's rule the range $[a, b]$ is divided to $2n$ subintervals $[x_i, x_{i+1}]$ of length h, where $x_i = x_0 + ih$ $(i = 0, 1, \ldots, 2n)$, $x_0 = a$, $x_{2n} = b$, and $h = (b - a)/2n$, i.e.,

$$\int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots + \int_{x_{2n-2}}^{x_{2n}} f(x)dx \tag{21}$$

and we obtain

$$\int_a^b f(x)dx \approx \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 2f_{2n-2} + 4f_{2n-1} + f_{2n})\,. \tag{22}$$

Exercise: Derive the formula for 3/8 rule.

In contrast to the above Newton–Cotes quadratures with equally spaced, fixed points, the Gaussian quadratures [1, 2, 3] treat the integration points as free parameters (Legendre polynomial nodes) used for solving the weight factors. Adaptive integration algorithms use one or more elementary quadratures and determine the subinterval lengths

automatically so that a given accuracy requirement is fulfilled. Different step lengths are used in different regions: a long step for a flat part of the function and a shorter otherwise. Computational overhead raises rapidly in multidimensional integration due to multiplicative integrals. A completely different approach is the Monte Carlo method, which is based on a use of random number generators and applies well, when the dimensionality is large.

4 Differential equations

Let us consider an initial value problem for a DE system:

$$\begin{cases} \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)) \\ \mathbf{y}(t_0) = \mathbf{y}^{(0)}\,, \end{cases} \tag{23}$$

where t is an independent variable and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$, i.e., component-wise:

$$\begin{cases} y_1'(t) = f_1(t, y_1, \ldots, y_m) \\ y_2'(t) = f_2(t, y_1, \ldots, y_m) \\ \vdots \\ y_m'(t) = f_m(t, y_1, \ldots, y_m)\,, \\ \\ y_1(t_0) = y_1^{(0)} \\ \vdots \\ y_m(t_0) = y_m^{(0)}\,. \end{cases} \tag{24}$$

Let us consider the $m = 1$ case and Taylor methods that generalize for equation systems. From the series expansion for $y(t)$ at $t + h$ we obtain an approximation for the formal solution $y(t') = y(0) + \int_0^{t'} f(t, y)dt$:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \mathcal{O}(h^2) \\ &= y(t) + hf(t, y) + \mathcal{O}(h^2)\,. \end{aligned} \tag{25}$$

By noting $t_n = t_0 + nh$, $y_n = y(t_n)$ we obtain the *Euler's method*

$$\begin{aligned} y_0 &= y^{(0)} \\ y_{n+1} &= y_n + hf(t_n, y_n)\,, \quad n = 0, 1, 2, \ldots \end{aligned} \tag{26}$$

The method accuracy ($\mathcal{O}(h^2)$ per step) can be enhanced by including additional terms from the Taylor series, e.g., the second-order Taylor method is

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + \mathcal{O}(h^3)\,, \\ y_{n+1} &= y_n + hf(t_n, y_n) + \frac{1}{2}h^2 g(t_n, y_n)\,, \end{aligned} \tag{27}$$

where

$$g(t,y) = \frac{\partial f(t,y)}{\partial t} + f(t,y)\frac{\partial f(t,y)}{\partial y}\,.$$

This method suffers a bit from the need of calculating the partial derivatives.

The Runge–Kutta method increases the accuracy by calculating the function f at several points. By forming Taylor expansions for $y(t)$ and $y(t+h)$ at $t+h/2$, we obtain a pair of equations

$$\begin{aligned} y(t) &= y(t+\frac{h}{2}) + y'(t+\frac{h}{2})\left(-\frac{h}{2}\right) + \frac{1}{2}y''(t+\frac{h}{2})\left[-\frac{h}{2}\right]^2 + \mathcal{O}[(h/2)^3]\,, \\ y(t+h) &= y(t+\frac{h}{2}) + y'(t+\frac{h}{2})\frac{h}{2} + \frac{1}{2}y''(t+\frac{h}{2})\left[\frac{h}{2}\right]^2 + \mathcal{O}[(h/2)^3]\,, \end{aligned} \tag{28}$$

which eliminates to

$$y(t+h) = y(t) + h\ \underbrace{y'(t+\frac{h}{2})}_{f(t+\frac{h}{2}\,,y(t+\frac{h}{2}))} + \mathcal{O}(h^3)\,. \tag{29}$$

By calculating $y(t+h/2)$ with Euler's method, i.e.,

$$y(t_n + \frac{h}{2}) = y_n + \frac{h}{2}f(t_n, y_n)\,, \tag{30}$$

we obtain the second-order Runge–Kutta method (a *central point method*):

$$\begin{cases} \hat{y} = hf(t_n, y_n) \\ y_{n+1} = y_n + hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}\hat{y})\,. \end{cases} \tag{31}$$

The error made during one step is here of order $\mathcal{O}(h^3)$. Expanding y' to a series at different points between $[y_n, y_{n+1}]$ and combining terms, the low-order errors terms eliminate. The best known result is the *fourth-order Runge–Kutta method*, where four calculations of f are needed for one full step:

$$k_1 = hf(t_n, y_n) \tag{32}$$

$$k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{33}$$

$$k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{34}$$

$$k_4 = hf(t_n + h, y_n + k_3) \tag{35}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5)\,. \tag{36}$$

For this to be as effective as the central point method, the step length must be twice as long while the error must remain at most the same. In case a high accuracy is not needed (thus h can be large) and f is nonregular, a lower-order method may be better.

These one-step solvers do not use any information about the previous steps. Multi-step methods utilize in addition to y_n also the y_{n-1}, y_{n-2}, ... values for determining the y_{n+1}. Quadrature results can be utilized here to obtain solutions in the form

$$y(t_{n+1}) = y(t_{n-k}) + \int_{t_{n-k}}^{t_{n+k}} f(t, y(t))dt$$

which lead to so called predictor-corrector solvers such as the Crank–Nicholson method.

5 Execution of the project

Literature. The written report describes shortly what is meant by the Gauss elimination, LU-decomposition, pivoting, positive definite equation systems, Cholesky method, and iterative solvers for equation systems. An example is given for numerical interpolation.

Differentiation. For $f(x) = \ln(x)$, compute the estimation for $f'(2)$ using the single precision (32-bit floating point representation, values 10^{-38}–10^{38}, ca. 7 significant numbers) with different values of h: 0.2, 0.1, 0.05, 0.01, 0.001, and 0.0001. Tabulate the $D_+(h)$, $D_-(h)$, $D_0(h)$, and $D_R(h)$ as well as $f''(2)$. Compare to double-precision (64 bits) $D_+(h)$ and f'' results for h values of 0.2, 0.1, and 0.05.

Integration. Compute the integral $\int_0^\pi \sin x\, dx$ using the trapezoidal and Simpson's rules with values 1, 2, 4, 8, 16, 32, 64, and 128 for n. Tabulate the results as well as the absolute errors.

Differential equation. Solve the initial value problem

$$\begin{cases} y'(t) = -y(t)^2 \\ y(0) = 1 \end{cases}$$

numerically using Taylor (first and second order) and Runge–Kutta methods between [0,1] with $h = 1/10$ step length. Compare to the analytical result.

Electron-vibration spectrum of iodine. After the solutions of previous exercises are shown to a teaching assistant, he/she gives a Matlab-code that solves the Schrödinger equation for I_2 vibration

$$-\frac{\hbar^2}{2\mu}\frac{d^2\Psi(r)}{dr^2} + V(r)\Psi(r) = E\Psi(r)\,,$$

where $V(r) = D_e(1 - e^{-a(r-r_e)})^2$ and reduced mass $\mu = m_I/2$. In atomic units $\hbar = 1$, $u = 1822.8885$, $a_0 = 52.9177249$ pm, and $E_h = 27.2113957$ eV. For the ground electronic state $X\,{}^1\Sigma_g^+$ we use [4] $r_e = 2.666$ Å, $D_e = 12244$ cm^{-1}, and $a = 1.870$ Å^{-1} ($\tilde{\nu}_e = 213.36$ cm^{-1}, $\tilde{\nu}_e x_e = 0.14$ cm^{-1}). For the excited electronic state $B\,{}^3\Pi_{0,u}^+$ we use $r_e = 2.971$ Å, $D_e = 4112$ cm^{-1}, and $a = 1.998$ Å^{-1} ($\sigma_e = 15730$ cm^{-1}, $\tilde{\nu}_e = 132.11$ cm^{-1}, $\tilde{\nu}_e x_e = 1.051$ cm^{-1}). The program builds a Hamiltonian matrix using the difference approximation for the ∇^2 term. The internuclear separation is defined between $[r_{\min}, r_{\max}] = [4.3, 11]$ a_0 and the number of grid points N is varied between 2^6 and 2^{11}. Hamiltonian (tri-diagonal) matrix is diagonalized and the obtained eigenenergies are compared to the analytical formula. The eigenstates are used in computing the Franck–Condon factors for excitations ($v' \leftarrow v'' = 0$–2). A figure is formed, where the X and B state potentials and some wavefunctions (e.g. $\psi^{(X)}_{v''=0,1,2}$) are plotted.

For the report, check how the solution gets better as N is increased from 64 up to 2^{10} at least. The figures 1–4 (wavefunctions, eigenenergies, Birge–Sponer plot, electronic spectra) help in this task. For the best solution, tabulate the eigenvalues (9 lowest) of both states along with the values from the analytic expression. Finally, compare the intensity of the fundamental IR vibration $(1 \leftarrow 0)$ to overtones $(2 \leftarrow 0)$ and $(3 \leftarrow 0)$.

References

[1] J. Haataja et al., *Numeeriset menetelmät käytännössä*, CSC–Tieteellinen laskenta Oy (2002).

[2] R. A. E. Mäkinen, *Numeeriset menetelmät*, Jyväskylän yliopistopaino (1998).

[3] W. H. Press et al., *Numerical Recipes in C – The Art of Scientific Computing*, Cambridge University Press (1992).

[4] I. J. McNaught, *J. Chem. Educ.* **57**, 101 (1980).