# Worksheet 10: Integration

June 6, 2014

## General Remarks

- The deadline for the worksheets is **Wednesday, 25 June 2014, 10:00** for the tutorial on Friday and **Friday, 27 June 2014, 10:00** for the tutorials on Tuesday and Wednesday.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Johannes (`zeman@icp.uni-stuttgart.de`; Tuesday, 9:45–11:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Shervin (`shervin@icp.uni-stuttgart.de`; Friday, 11:30–13:00)

## Task 10.1: One-dimensional Integration (7 points)

The error function is a sigmoidal function that is used in statistics and some other areas. It is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau^2} \,\mathrm{d}\tau \tag{1}$$

The job in this task is to implement the error function.

- **10.1.1** (1 point) Implement a Python function `integrate1d_midpoint(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x)\,\mathrm{d}x$ using the midpoint rule with `N` support points.
- **10.1.2** (1 point) Implement a Python function `integrate1d_romberg(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x)\,\mathrm{d}x$ using Romberg's method with `N` support points on the finest level, i. e. use $h = \frac{b-a}{N-1}$ as smallest step size.

  **Hint** Note that the implementation in the lecture script uses $N = 2^{kmax}$ support points, so choose $kmax$ accordingly.
- **10.1.3** (1 point) Implement a Python function `integrate1d_mc(f,a,b,N)` that approximates $\int_a^b \mathrm{f}(x)\,\mathrm{d}x$ using `N` steps of Monte-Carlo Integration.
- **10.1.4** (1 point) Implement the Python functions `erf_midpoint(x,N)`, `erf_romberg(x,N)` and `erf_mc(x,N)` that compute the error function for `N` points with the corresponding integration rules.
- **10.1.5** (1 point) Evaluate the error function $\operatorname{erf}(x)$ on the interval $[0, 2.0]$ at 100 equidistant points $x$ using `erf_romberg` with $N = 64$, and plot the resulting approximation of the error function.
- **10.1.6** (1 point) Compute a reference value $e_{\mathrm{ref}} = \operatorname{erf}(1)$ using Romberg's method with $N = 512$. Create a loglog plot of the accuracy of all implemented methods when computing $\operatorname{erf}(1)$ for $N \in \{4, 8, 16, 32, 64, 128, 256, 512\}$.

- **10.1.7** (1 point) Use the `integrate1d_*` functions to approximate $\pi$ via

$$\frac{\pi}{4} = \int_0^1 \sqrt{1-x^2}\,\mathrm{d}x \tag{2}$$

Create a loglog plot of the accuracy of the different integration schemes versus the number of points $N \in \{4, 8, 16, \ldots, 512\}$. Use `numpy.pi` as reference value. Why does the accuracy behave differently this time?

## Task 10.2: Multidimensional Monte-Carlo Integration (3 points)

In Gaussian units ($\epsilon = 1$), the interaction energy of two homogenously charged bodies $V_1$ and $V_2$ is given by

$$E = \int_{V_1}\int_{V_2} \frac{1}{|\vec{x}_1 - \vec{x}_2|} d\vec{x}_1 d\vec{x}_2$$

In this task, we consider the interaction of two homogenously charged cubes (in 3D; $Q = 1$) $V_1$ and $V_2$ with side length 1 at a distance $d$ (see the figure below).

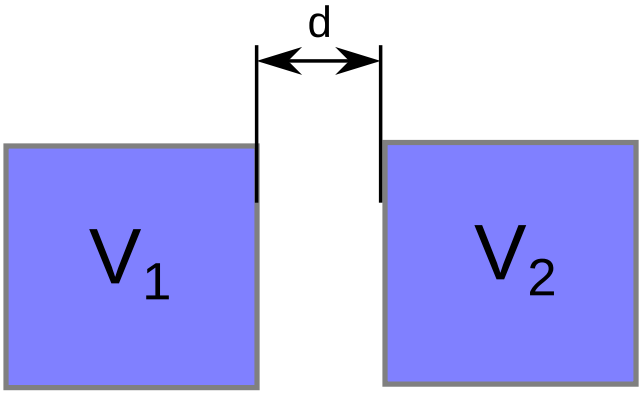


Figure 1: Two homogenously charged cubes $V_1$ and $V_2$ at distance $d$.

- **10.2.1** (2 points) Write a Python function that computes the interaction energy of the cubes at distance $d$ using Monte-Carlo integration with $N = 1000$ steps.
- **10.2.2** (1 point) Compute the interaction energy for 10 different values of $d$ between $10^{-2}$ and $10^2$. Plot the interaction energy $E$ versus the distance $d$ in log-log-scale. In the same figure, add a plot of the interaction energy of two unit point charges at distance $d$.