**Physik auf dem Computer** **SS 2013**

# Worksheet 9: Integration

June 21, 2013

## General Remarks

- Deadline is **Wednesday, 26th June 2013, 9:00**.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Olaf (`olenz@icp.uni-stuttgart.de`; Wednesday, 14:00–15:30)
  - Elena (`minina@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Friday, 15:45–17:15)
- Attach all required files to the mailing. If asked to write a program, attach the *source code* of the program. If asked for a text, send it as PDF or in the text format. We will *not* accept MS Word files!
- The worksheets are to be solved in groups of two or three people.
- The tutorials take place in the CIP-Pool of the ICP in Allmandring 3.

## Task 9.1 (7 points): One-dimensional Integration

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.

- 9.1.1 (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.
- 9.1.2 (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 $N = 2^{\mathrm{kmax}}$ support points, so choose kmax accordingly.
- 9.1.3 (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.
- 9.1.4 (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.
- 9.1.5 (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.

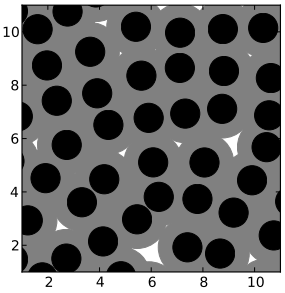
- 9.1.6 (1 point) Compute a reference value $e_{\text{ref}} = \text{erf}(1)$ using Romberg's method with $N = 512$. Create a loglog plot of the accuracy of all implemented methods when computing erf(1) for $N \in \{4, 8, 16, 32, 64, 128, 256, 512\}$.
- 9.1.7 (1 point) Use the `integrate1d_*` functions 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 9.2 (3 points): Two-dimensional integration

Consider a twodimensional square, filled by spheres of diameter 1, as illustrated in black on the right. If we want to add another sphere of the same diameter, so that it does not overlap with the existing spheres, most of the space (in gray) is blocked. The *accessible volume* denotes the remaining area, that is, the few remaining white spots in the figure. In this task, we want to compute this accessible volume, which can be characterized by its characteristic function




$$\chi(\vec{x}) = \begin{cases} 1 & \text{if } \min\limits_{\vec{p} \in P} ||\vec{x} - \vec{p}|| > 1 \\ 0 & \text{otherwise}\,, \end{cases} \tag{3}$$

where $P$ is the set of particles in the box. The accessible volume, *i.e.* the free area, is:

$$V_a = \int_{x_{\min}}^{x_{\max}} \int_{y_{\min}}^{y_{\max}} \chi(x, y)\, \mathrm{d}y\, \mathrm{d}x\,. \tag{4}$$

- 9.2.1 (1 point) Get the file `coords.xy`, which contains the coordinates of the spheres. You can load the positions into Python via:

```
P = genfromtxt('coords.xy')
```

  Write the characteristic function `accessible(x, y)`, which for a given point `x`, `y` returns $\chi(x, y)$.
- 9.2.2 (1 point) Integrate this function in order to calculate the free volume using the Monte Carlo method with $N \in \{5^2, 10^2, 15^2, 20^2 \ldots 50^2\}$ steps. The integration limits are $x_{\min}$, $y_{\min} = 1$ and $x_{\max}$, $y_{\max} = 11$.
- 9.2.3 (1 point) Plot the results for $V_a$ over the number of steps and make a two-dimensional plot of the accessible volume using `matplotlib.pyplot.imshow` Why did we choose Monte Carlo instead of e. .g. Romberg integration?