VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Guidance material for students

# Requirements and guidelines for written work

Vilnius
2023

# Contents

# 1 Illustrations, tables and pseudocode

## 1.1 Requirements for illustrations and tables

### 1.1.1 Requirements for description of illustrations

All illustrations used in the work must be numbered and must have a caption (short text at the bottom of the illustration), that mentions what the illustration (or each of its constituent parts, if the illustration consists, for example, of a number of different graphs) shows. If it makes sense, include additional information in the description of the illustration, such as:

- model name or number (if more than one model is analysed in the work);
- the name or number of the algorithm whose performance is reflected in the illustration (if you are analysing more than one algorithm for a particular model);
- the name or number of the data to be processed (if you are analysing a large number of different data);
- list the specific numerical values of all parameters (both model and algorithm) with which the result shown in the illustration was obtained. If a particular parameter is not dimensionless, do not forget to include its dimension (e. g. seconds, years, metres, centimetres, grams, gigaflops, etc., using commonly accepted abbreviations). If you are analysing only one set of parameters in your work, please refer to it in the main body of the paper rather than in the description of the illustration;
- other relevant information related to the illustration.

All illustrations must be cited in the work. In the body of the paper, give a brief interpretation of the result obtained in the illustration – what conclusions can be drawn from the illustration, perhaps comparing it with other illustrations, giving the numbers of the illustrations being compared.

### 1.1.2 Requirements for the design of illustrations

Please pay attention to the proper layout of the illustrations:

- the illustration must not be unduly reduced or enlarged. If possible, construct the illustration in two parts (left illustration and right illustration) using the full width of the page. The following examples demonstrate how large illustrations should be;
- the curves of the graphs should be sufficiently thick, and the illustration should not be overloaded with too many curves (if necessary, two or more illustrations can be provided instead of one);
- axis variables, gradations and other information in the illustration (text, arrows, etc.) must be clear and not small. Please note that the graphic file of an illustration is often significantly reduced during layout (when it is inserted into the text of the work). When the illustration is viewed in the body of the thesis, all textual information in the illustration should be displayed in a font size no smaller than that used in the body of the thesis, curves and other graphical information must be of optimal thickness, easily visible and distinguishable.
- the names of the axis variables and, if the variable is not dimensionless, the dimensions must be mentioned (e. g. seconds, years, etc., using commonly accepted abbreviations).

### 1.1.3 Other information on illustrations and tables

For the best possible graphic quality of illustrations, always create them only in *vektoriniame* format (not raster, so never use JPG format – it is prone to pixelation and other artifacts). A common question is which of the vector graphics formats to choose (for illustration graphics files): PostScript (PS), Encapsulated PostScript (EPS) or Portable Document Format (PDF)? We recommend PDF, the reasons for which are given, for example, here: http://tex.stackexchange.com/questions/2092/which-figure-type-to-use-pdf-or-eps.

If you plan to print the work (before binding) on a monochrome printer, do not use colour illustrations. After printing the job, check that all illustrations look as good on paper as they do on the monitor screen.

A common mistake is when showing the dependence of a curve on a variable (e.g. the dependence of exchange rates on time), not the values of the variable (e.g. time values: months of the current year) but the indices of an array of discrete values of the curve are placed on the abscissae axis of the illustration (usually not conveying any relevant information).

Note also that in modelling physical, chemical and similar phenomena, the dimensions must be consistent – the numerical values of all parameters and quantities of interest must be defined in a unified system during calculations (e. g. in the SI system – then 2 µm is entered as $2 \times 10^{-6}$ m in the calculations, since the SI system unit is the metre and not the micrometer).

If necessary, you can also present the results of the tests as tables of figures. Tables should also be numbered, have their own descriptions and be cited in the paper. According to the rules established in the scientific literature, the description of the table should be placed at the top of the table (as opposed to the description of the illustration, which should be placed below the illustration).

Examples of properly captioned illustrations and tables (and their description in the body of the paper) are given below. See 1.5 section for an example of pseudocode.

## 1.2 Examples of illustrations

### 1.2.1 First illustration example

We will model an electrochemical biosensor (see Fig. 1) consisting of an electrode, a membrane (with a thickness of $d_{m1} \geqslant 0$) surrounding the electrode, an enzyme layer (thickness $d_e > 0$) and an outer membrane (thickness $d_{m2} > 0$). In a particular case, the electrode envelope membrane (also sometimes referred to as the discriminating membrane) may not be present (if $d_{m1} = 0$). The biosensor is immersed in a solution in which a constant concentration of the substrate (a specific chemical, such as glucose, whose concentration is measured by the biosensor) is maintained. When the substrate (which is not electrochemically active) reacts with the enzyme (in the enzyme layer), the substrate is transformed into a product (another chemical such as hydrogen peroxide). The reaction can also be influenced by, involve or involve the formation of other chemical compounds. The product is an electrochemically active substance and therefore generates an electric current (on the electrode), which we measure to assess the concentration of the substrate in solution.

The thicknesses of the layers (membranes, enzyme layer) that make up the biosensor (and hence the dimensions of the biosensor itself) are measured in microns (synonymous with micrometers, notation µm). These thicknesses can range from a few to several microns. By comparison, objects such as the thickness of a sheet of paper or the diameter of a human hair can range from $50$ to $100$ microns. Thus, the biosensor is a truly ‘‘small’’ device.

The mathematical variable $x$ denotes the coordinate in the biosensor (distance to the electrode) and $t \geqslant 0$ is the time.

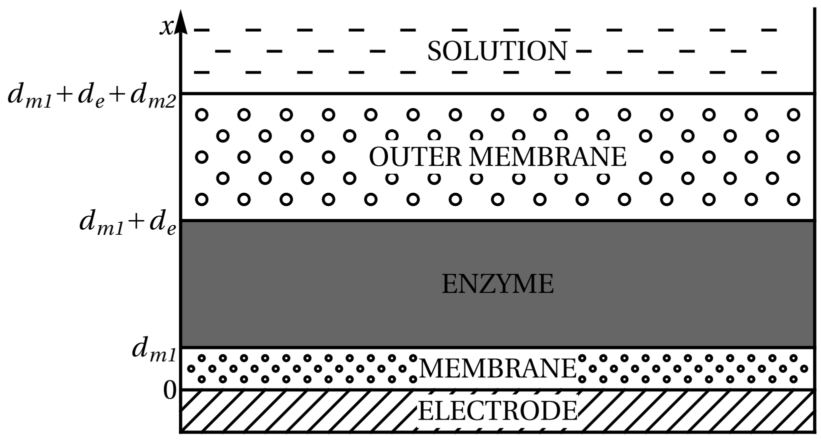

Figure 1. Schematic diagram of an electrochemical biosensor. The arrow marks the axis of the variable $x$.

The outer membrane is usually made of a polymer fibre. Very often it is cellophane films. Sometimes dense polymer films are used in which holes are artificially created. For example, Teflon films shot through with argon nuclei – under an electron microscope, such films look like a sheet of plywood shot with a hunting rifle.

### 1.2.2 Second illustration example

The boundary conditions for the substrate concentration $S$ at the edges of the simulation domain $0 \leqslant x \leqslant d_{m1} + d_e + d_{m2}$ are formulated considering that the substrate cannot penetrate the electrode and that the solution (in which the biosensor is immersed) is kept at a constant substrate concentration $S_0$:

$$\left.\frac{\partial S}{\partial x}\right|_{x=0} = 0, \qquad S(x = d_{m1} + d_e + d_{m2}, t) = S_0, \qquad t > 0. \tag{1.1}$$

In addition, we assume that the product reacts very quickly on the electrode. Therefore, the concentration of the product $P(x = 0, t)$ (on the edge with the electrode) will be zero at all times.

We will consider two different cases in the simulation.

**The case of a product diffusing into a solution.** Suppose that the wall of the outer membrane allows the product to diffuse into the solution. Then the concentration of the product $P$ at the edges of the interval $0 \leqslant x \leqslant d_{m1} + d_e + d_{m2}$ satisfies the conditions

$$P(x = 0, t) = 0, \qquad P(x = d_{m1} + d_e + d_{m2}, t) = 0, \qquad t > 0. \tag{1.2}$$

**The case where the product cannot enter the solution.** If the outer membrane wall is impermeable to the product, the second condition in the boundary conditions (1.2) must be replaced by the requirement, that the derivative of the function at the edge of the region must be zero (no product leakage). Thus, in this case, we formulate the following boundary conditions for the product concentration $P$:

$$P(x = 0, t) = 0, \qquad \left.\frac{\partial P}{\partial x}\right|_{x=d_{m1}+d_e+d_{m2}} = 0, \qquad t > 0. \tag{1.3}$$

If $C_1 > 0$ and / or $C_2 > 0$, then the response of the biosensor (the steady-state current density) $I$ decreases due to degradation (compared to the case without degradation $C_1 = 0\,\mathrm{s}^{-1}$, $C_2 = 0\,\mathrm{s}^{-1}$). We will investigate the dependence of this decrease on the parameters $C_1$ and $C_2$.

Illustration Fig. 2 graphically highlights the parts of the parameter space $(C_1, C_2)$ in which the biosensor response decreases from $0\%$ to $1\%$ (white shaded area), from $1\%$ to $2\%$ (white shaded area), from $2\%$ to $3\%$ (light grey area), from $3\%$ to $4\%$ (light grey hatched area), and so on, as the background of the area becomes darker.

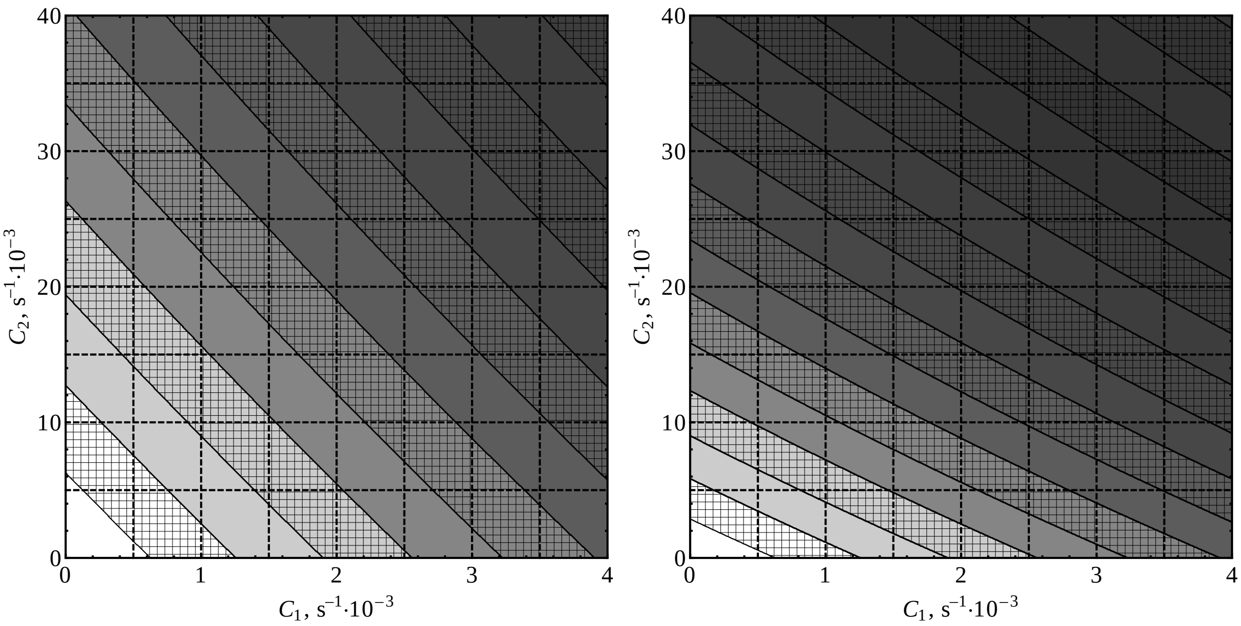

Figure 2. Degradation (decrease) of the biosensor response $I$ in the coordinate plane $(C_1, C_2)$. Illustration on the left: boundary conditions (1.2) (the case of a product diffusing into a solution). Illustration on the right: boundary conditions (1.3) (the case where the product cannot enter the solution). Model and algorithm parameters: $N = 100$, $\tau = 0.01\,\mathrm{s}$, $d_{m1} = 2\,\mu\mathrm{m}$, $d_e = 9\,\mu\mathrm{m}$, $d_{m2} = 10\,\mu\mathrm{m}$, $D_{S_{m1}} = 6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_{m1}} = 5\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{S_e} = 22\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_e} = 20\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{S_{m2}} = 7\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_{m2}} = 6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $V_{max} = 0.3\,\mathrm{mmol\,m}^{-3}\,\mathrm{s}^{-1}$, $K_M = 0.23\,\mathrm{mol\,m}^{-3}$, $S_0 = 0.07\,\mathrm{mol\,m}^{-3}$.

We can see that, with the parameters given in Fig. 2, the results quite *significantly depend on the boundary conditions with which the mathematical model is defined*: boundary conditions (1.2) or (1.3).

Note also (see Fig. 2) that when interpreting what (different) values of the parameters $C_1 > 0$ and $C_2 > 0$ can lead to some one and the same (fixed) decrease in $I$, the *almost linear* relationship between parameters $C_1$ and $C_2$ is obtained. However, the isolines for the (uniform) reduction $I$ seen in the illustration are not absolutely ideal lines.

It is also worth noting that the impact of $C_1$ on degradation is *several times higher* than that of $C_2$. Exactly how many times depends on the choice of boundary conditions. Calculating with the boundary conditions (1.2), it can be seen (see Fig. 2, illustration on the left) that the parameter choices $C_1 = 4\,\mathrm{m}(\mathrm{s}^{-1})$, $C_2 = 0\,\mathrm{m}(\mathrm{s}^{-1})$ and $C_1 = 0\,\mathrm{m}(\mathrm{s}^{-1})$, $C_2 = 40\,\mathrm{m}(\mathrm{s}^{-1})$ causes an approximately equal decrease in the steady-state current density $I$, so that the influence of the parameter $C_1$ is about $10$ times greater than that of the parameter $C_2$. However, the choice of the boundary conditions (1.3) shows (see Fig. 2, illustration on the right) that the influence of $C_1$ is only about $5$ times larger than that of $C_2$.

### 1.2.3 Third illustration example

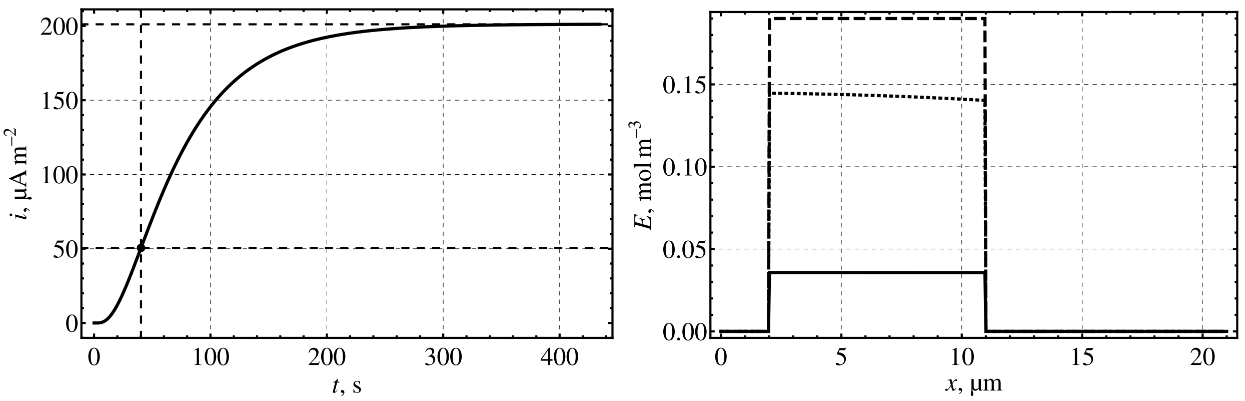

Figure 3. Illustration on the left: the dependence of the current density $i(t)$ on the time $t$ (dashed lines indicate the time instant $t = t_{slopeMAX}$ and the biosensor response $I$). Illustration on the right: the ependence of the enzyme concentration $E(x,t)$ on the biosensor coordinate $x$, at time $t = 1$ s (dashed curve), at time $t = t_{slopeMAX}$ (dotted curve), and at time $t = t_I$ (solid curve).

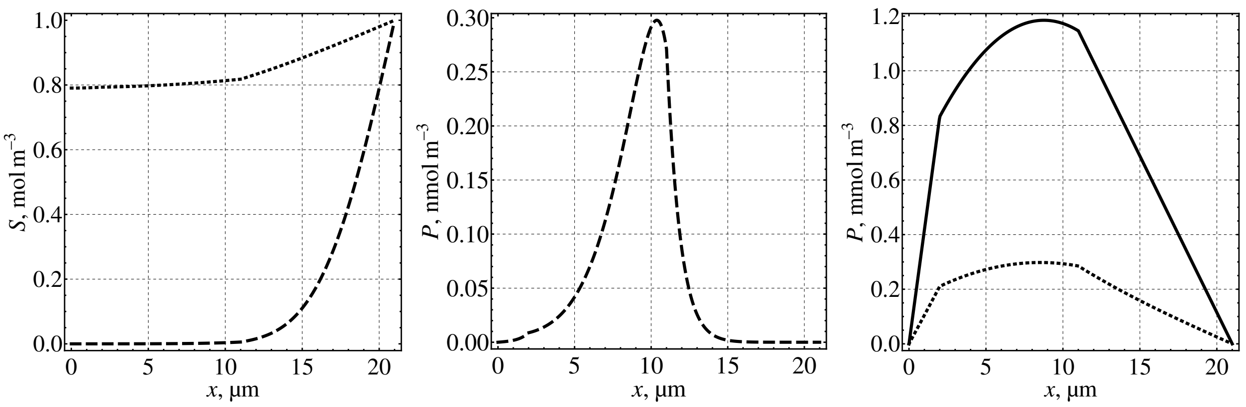

Figure 4. Illustration on the left: the dependence of the substrate concentration $S(x,t)$ on the biosensor coordinate $x$, at time $t = 1$ s (dashed curve) and at time $t = t_{slopeMAX}$ (dotted curve). Illustration in the centre: the dependence of the product concentration $P(x,t)$ on the biosensor coordinate $x$, at time $t = 1$ s (dashed curve). Illustration on the right: the dependence of the product concentration $P(x,t)$ on the biosensor coordinate $x$, at time $t = t_{slopeMAX}$ (dotted curve) and at time $t = t_I$ (solid curve).

Calculations run with the parameters of the mathematical model and the algorithm: boundary conditions (1.2) (the case of a product diffusing into a solution), $N = 1000$, $h = 0.021\,\mu\mathrm{m}$, $\tau_{max} = 0.01$ s, $d_{m1} = 2\,\mu\mathrm{m}$, $d_e = 9\,\mu\mathrm{m}$, $d_{m2} = 10\,\mu\mathrm{m}$, $C_1 = 0\,\mathrm{s}^{-1}$, $C_2 = 0\,\mathrm{s}^{-1}$, $S_0 = 1\,\mathrm{mol\,m}^{-3}$, $E_0 = 0.19\,\mathrm{mol\,m}^{-3}$, $k_{+1} = 0.015\,\mathrm{m}^3\,\mathrm{mol}^{-1}\,\mathrm{s}^{-1}$, $k_{-1} = 0.0015\,\mathrm{s}^{-1}$, $k_{+3} = 0.002\,\mathrm{s}^{-1}$, $k_{-3} = 0.002\,\mathrm{m}^3\,\mathrm{mol}^{-1}\,\mathrm{s}^{-1}$, $n_e = 1$.

Figures 3 and 4 show the results when the diffusion coefficients are equal: $D_{S_{m1}} = 6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_{m1}} = 5\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{S_e} = 22\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_e} = 20\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{S_{m2}} = 7\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_{m2}} = 6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$.

The biosensor response value $I = 201.1\,\mu\mathrm{A\,m}^{-2}$ is evaluated after reaching the time point $t_I = 434.6$ s. The current density increased most intensively at time point $t_{slopeMAX} = 40.3$ s.

### 1.2.4 Fourth illustration example

Let's examine the *isoline* of the five per cent degradation of the biosensor's response $I$ (curve of equal relative decrease of $I$) in the coordinate plane $(C_1, C_2)$. The points on the isoline represent the values of the parameters $C_1$ and $C_2$ with which the steady-state current density $I$ decreases by exactly $5\%$ (compared to the value of $I$ obtained in the non-degraded case $C_1 = 0\,\mathrm{s}^{-1}$, $C_2 = 0\,\mathrm{s}^{-1}$).

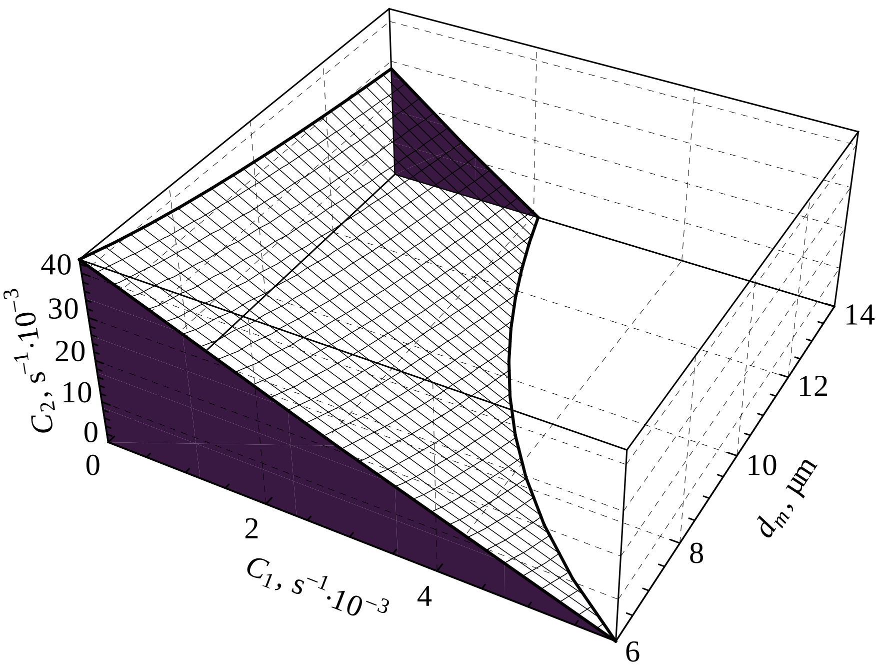

Figure 5. The dependence (surface) of the five per cent degradation isoline (defined in the coordinate plane $(C_1, C_2)$) on the response $I$ of the biosensor on the membrane thickness $d_m$. Model and algorithm parameters: $N = 100$, $\tau = 0.01\,\mathrm{s}$, $d_e = 9\,\mu\mathrm{m}$, $D_{S_e} = 22\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_e} = 20\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{S_m} = 7\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_{P_m} = 6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $V_{max} = 0.3\,\mathrm{mmol\,m^{-3}\,s^{-1}}$, $K_M = 0.23\,\mathrm{mol\,m^{-3}}$, $S_0 = 0.07\,\mathrm{mol\,m^{-3}}$.

This isoline (and others) can be seen in Fig. 2. In this section, we will additionally examine the above isolines (defined in the coordinate plane $(C_1, C_2)$) dependence on the membrane thickness $d_m$ in the interval $d_m \in [6, 14]\,\mu\mathrm{m}$.

The result of the calculations is shown in Fig. 5. It can be seen that the area under the isoline (highlighted with a grey background) decreases with increasing $d_m$. With parameters $C_1 \geqslant 0$, $C_2 \geqslant 0$, the degradation of this region (below the isoline) $I$ does not exceed $5\%$. Note that for all $d_m \in [6, 14]\,\mu\mathrm{m}$, the isoline roughly keeps a straight line shape, However, a parabolic curve should be used to approximate it more accurately.

We will write down the analytical dependencies of the isoline approximation on the parameter $d_m$, valid over the interval $d_m \in [6, 14]\,\mu\mathrm{m}$, with the parameters of the mathematical model given in Fig. 5. On the basis of the results of the calculations, presented in Fig. 5, we derive (by the *least squares* method) the following analytical expressions:

$$C_2 = K(d_m)\,C_1 + M(d_m)\,, \qquad C_1 \geqslant 0, \qquad C_2 \geqslant 0, \tag{1.4}$$

$$K(d_m) = -0.791771\,d_m/\mu\mathrm{m} - 2.48837, \tag{1.5}$$

$$M(d_m) = \left(-0.583423\,d_m/\mu\mathrm{m} + 28.3346 + \frac{110.039}{d_m/\mu\mathrm{m}}\right)\Big/\,1000. \tag{1.6}$$

For each $d_m$, the isoline in the coordinate plane $(C_1, C_2)$ passes through the points

$$(C_1 = 0, C_2 = M(d_m))\,\mathrm{s}^{-1}, \qquad \left(C_1 = -\frac{M(d_m)}{K(d_m)}, C_2 = 0\right)\mathrm{s}^{-1}.$$

## 1.3 Example of an illustration and a table

In order to further optimise the Cooley-Tukey Fast Fourier Transform algorithm, recursion can be dropped from the programming, and the data (signal values $f_j$) can be arranged immediately in the order in which they would be summed at the last (deepest) level of recursion.

It is also worthwhile to pre-calculate and store in arrays the complex constants $f_j$, which are independent of the signal values $W_m^k$, only for those integers $m$ and $k$ for which these constants will actually be needed in the calculation.

We will explain in what order the Cooley-Tukey algorithm processes the signal values $f_j$ and which coefficients $W_m^k$ are needed.

The working principle of the algorithm can be understood by considering the case of data with few values. Suppose $N = 8 = 2^3$. Then the data (the values of the digital signal) are numbers

$$f = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7)\,.$$

Our aim is to calculate the values of

$$C_k = f_0 + f_1\,W_8^k + f_2\,W_8^{2k} + f_3\,W_8^{3k} + f_4\,W_8^{4k} + f_5\,W_8^{5k} + f_6\,W_8^{6k} + f_7\,W_8^{7k},$$
$$k = 0, 1, 2, 3, 4, 5, 6, 7.$$

---

At the first level of recursion of the Cooley-Tukey algorithm, for each fixed $k = 0, 1, 2, 3$, we obtain two sums $A_k$ and $B_k$ of the four spots:

$$C_k = A_k + W_8^k B_k, \qquad C_{k+4} = A_k - W_8^k B_k, \qquad k = 0, 1, 2, 3,$$
$$A_k = \left[f_0 + f_2\,W_4^k + f_4\,W_4^{2k} + f_6\,W_4^{3k}\right], \qquad B_k = \left[f_1 + f_3\,W_4^k + f_5\,W_4^{2k} + f_7\,W_4^{3k}\right],$$

or

$$C_k = \left[f_0 + f_2\,W_4^k + f_4\,W_4^{2k} + f_6\,W_4^{3k}\right] + W_8^k\left[f_1 + f_3\,W_4^k + f_5\,W_4^{2k} + f_7\,W_4^{3k}\right].$$

---

At the second level of recursion, for each fixed $k = 0, 1$, we will have four sums $AA_k$, $AB_k$, $BA_k$, $BB_k$, each of the two components:

$$A_k = AA_k + W_4^k AB_k, \qquad A_{k+2} = AA_k - W_4^k AB_k,$$
$$B_k = BA_k + W_4^k BB_k, \qquad B_{k+2} = BA_k - W_4^k BB_k, \qquad k = 0, 1,$$
$$AA_k = \left(f_0 + f_4\,W_2^k\right), \qquad AB_k = \left(f_2 + f_6\,W_2^k\right),$$
$$BA_k = \left(f_1 + f_5\,W_2^k\right), \qquad BB_k = \left(f_3 + f_7\,W_2^k\right),$$

or

$$C_k = \left[\left(f_0 + f_4\,W_2^k\right) + W_4^k\left(f_2 + f_6\,W_2^k\right)\right] + W_8^k\left[\left(f_1 + f_5\,W_2^k\right) + W_4^k\left(f_3 + f_7\,W_2^k\right)\right].$$

The third level of recursion would formally define eight ''sums'', each consisting of a single component -- the signal value. The order in which these values are arranged, and the coefficients by which they are to be multiplied in the summation, is already apparent from the formulae of the second recursion level. We can therefore proceed immediately to the 1-stage of the algorithm, where the real calculations begin.

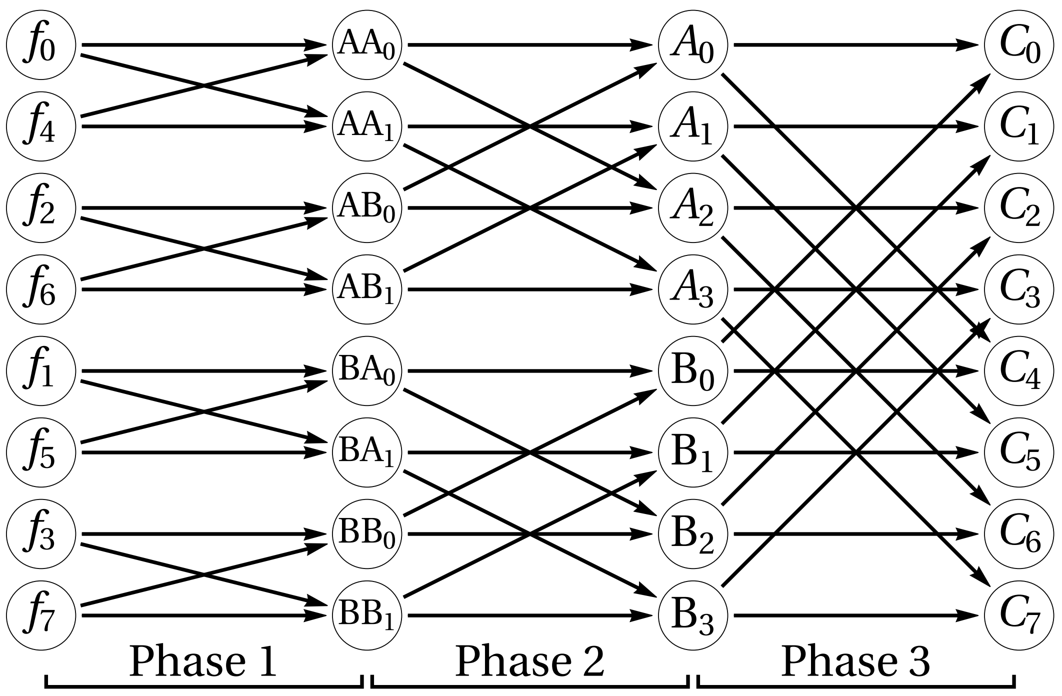

Figure 6. Schematic of the arithmetic operations in the Cooley-Tukey Fast Fourier Transform algorithm for 8 points ($N = 8$). Arrows indicate which values from the previous phase of the algorithm are involved in the calculation of the next phase.

*Phase* 1. Using the signal values $f_j$ and the coefficients $W_2^k$, $k = 0, 1$, we calculate the intermediate values $AA_k$, $AB_k$, $BA_k$, $BB_k$ and $k = 0, 1$. The calculation scheme is given in Fig. 6. In this scheme, the arrows indicate which quantities are involved in the arithmetic operations. For example,

$$AA_0 = f_0 + f_4\, W_2^0,$$

so there is an arrow from $f_0$ and an arrow from $f_4$ to $AA_0$. The arrow also represents a single arithmetic operation: $f_4$ is multiplied by the constant $W_2^0$; the result is added with $f_0$.

Incidentally, only one constant $W_2^0 = 1$ is needed in step 1 (since $W_2^1 = -W_2^0 = -1$). Since all multipliers are equal to one, the calculation of products can be avoided at this stage.

Also, we can see in which order the data should be arranged for fastest access:

$$f_0, f_4, f_2, f_6, f_1, f_5, f_3, f_7.$$

Looking at this order of arrangement, it is difficult to see how it could be generalised when $N = 16$, $N = 32$ and so on. Everything becomes clear when the data indices are expressed in binary rather than decimal not decimal notation, see Table 1. The binary bits of the indices of the values $f_j$ need only be written in reverse (mirror) order.

*Phase* 2. Now we use the quantities $AA_k$, $AB_k$, $BA_k$, $BB_k$, $k = 0, 1$ and $k = 0, 1$ (calculated in the previous Phase 1) to calculate the intermediate values $A_k$, $B_k$, $k = 0, 1, 2, 3$. The calculation scheme can be seen in Fig. 6.

Table 1. Optimal ordering of processed data in the Cooley-Tukey Fast Fourier Transform algorithm for 8 points ($N = 8$).

| Indices $j$ in traditional order | Binary bits of the index $j$ | Bits of the index $j$ in reverse | Indices $j$ in bit-reversed order |
|---|---|---|---|
| 0 | 000 | 000 | 0 |
| 1 | 001 | 100 | 4 |
| 2 | 010 | 010 | 2 |
| 3 | 011 | 110 | 6 |
| 4 | 100 | 001 | 1 |
| 5 | 101 | 101 | 5 |
| 6 | 110 | 011 | 3 |
| 7 | 111 | 111 | 7 |

Phase 2 requires two constants $W_4^k$, $k = 0, 1$.

*Phase* 3. Given the quantities $A_k$, $B_k$, $k = 0, 1, 2, 3$ calculated in the previous Phase 2, find the coefficients $C_k$, $k = 0, 1, 2, 3, 4, 5, 6, 7$. The calculation scheme is presented in Fig. 6.

In Phase 3 we use the constants $W_8^k$, $k = 0, 1, 2, 3$.

Phase 3 is the last phase, as we had to process a data sequence of length $N = 2^3$.

In all products in the algorithm, one of the multipliers is the complex constant $W_m^k$. Only constants (for these indices $m$ and $k$) are employed: Naudojamos tik konstantos (šiems indeksams $m$ ir $k$):

$$W_m^k = e^{-i\frac{2\pi}{m}k} = \cos\frac{2\pi k}{m} - i\,\sin\frac{2\pi k}{m}, \quad m = 2, 4, 8, 16, \ldots, N, \quad k = 0, 1, \ldots, \frac{m}{2} - 1.$$

By the way, if $k = 0$, then $W_m^0 = 1$.

*Complexity of the Cooley-Tukey Fast Fourier Transform algorithm*

Let's estimate how many arithmetic operations are carried out in the Cooley-Tukey Fast Fourier Transform algorithm. We will consider subtraction and addition as equivalent operations in terms of CPU time, and count multiplications separately.

As we can see from the example above, $n = \log_2 N$ phases (steps) are required. In each phase, $N$ additions and $N/2$ multiplications are calculated. There are twice as many multiplications as additions because, in each phase, the second half of the quantities is calculated in the same way as the first half, but with addition replaced by subtraction. Pavyzdžiui, mūsų nagrinėtame pavyzdyje su $N = 8$, 2-ajame etape skaičiavome dydžius $A_k$ pagal formules For example, in the case of $N = 8$, we calculated the quantities $A_k$ (in the Phase 2) using the formulae

$$A_k = AA_k + W_4^k AB_k, \qquad A_{k+2} = AA_k - W_4^k AB_k, \qquad k = 0, 1,$$

so there is no need to calculate the product $W_4^k AB_k$ when calculating $A_{k+2}$ (it has already been calculated when finding $A_k$).

This means that there are $(N/2)\log_2 N$ complex multiplications and $N \log_2 N$ complex additions in the algorithm. The number of multiplications can be further reduced slightly by taking advantage of the fact that $W_m^0 = 1$ and other special cases of the constants $W_m^k$. For example, as we have seen, it is possible to avoid calculation of products in the phase 1 of the algorithm.

Thus, the complexity of the Cooley-Tukey Fast Fourier Transform algorithm is $O(N \log_2 N)$.

## 1.4 Example of a table

Table 2 examines the relative accuracy of the defect density $N_{def}$ estimate.

Table 2. Relative errors of defect density $N_{def}$ estimates.

| Defect radius $r_0$, nm | Frecuency $f_{min}$, Hz | Exact $N_{def}$, µm$^{-2}$ | Evaluated $N_{def}$, µm$^{-2}$ | Relative error |
|---|---|---|---|---|
| 1 | 516 | $2.89 \cdot 10^1$ | $3.02 \cdot 10^1$ | 4.5% |
| 10 | 1090 | $2.89 \cdot 10^1$ | $3.81 \cdot 10^1$ | 32.1% |
| 1 | 3.3 | $2.89 \cdot 10^{-1}$ | $2.72 \cdot 10^{-1}$ | 5.7% |
| 10 | 5.02 | $2.89 \cdot 10^{-1}$ | $2.57 \cdot 10^{-1}$ | 11.0% |
| 100 | 9.78 | $2.89 \cdot 10^{-1}$ | $3.01 \cdot 10^{-1}$ | 4.3% |
| 1 | 0.0242 | $2.89 \cdot 10^{-3}$ | $2.83 \cdot 10^{-3}$ | 2.0% |
| 10 | 0.0328 | $2.89 \cdot 10^{-3}$ | $2.39 \cdot 10^{-3}$ | 17.2% |
| 100 | 0.0502 | $2.89 \cdot 10^{-3}$ | $2.26 \cdot 10^{-3}$ | 21.7% |
| 1000 | 0.0976 | $2.89 \cdot 10^{-3}$ | $2.62 \cdot 10^{-3}$ | 9.3% |

## 1.5 Example of a pseudocode

Sometimes there is a need to generate a signal with certain statistical properties by computer. Such signals are called *synthetic* signals or *simulated* signals, since they are not experimental data of any real-life process, but only a mathematical simulation of it.

Signals of synthetic origin can be easily and quickly generated, whereas the collection of real experimental data is incomparably more time-consuming and requires other resources. Such synthetic signals can be and are used for academic purposes, to compare mathematical models with experimental results, for testing implementations of computer algorithms. For example, it is convenient to model various noises with the required properties by computer simulation.

Synthetic signals are obtained from a mathematical formula or by solving a given mathematical model (e. g. recurrent relations, differential equation, etc.). The mathematical formula or model may be deterministic (without random variables). In this case, we simply program the formula or algorithm for a given model.

However, it is often necessary to program signal simulations that contain one or more random variables. For example, when simulating noise, the movement of a randomly wandering particle, the arrival times of urban transport, etc.

We assume that we know how to generate a pseudo-random number with uniform probability at any point on the interval $[0, 1)$, called a random variable uniformly distributed over the interval $[0, 1)$. We denote such numbers by $u_1$, $u_2$, and so on. All programming languages and environments have tools for obtaining such numbers.

It should be said that most programming languages and environments generate these pseudo-random numbers in a rather primitive way, so their randomness is superficial. Serious applications would require the use of special libraries of programs using specialised methods (e. g. the Mersenne Twister algorithm) for obtaining high-quality pseudorandom numbers $u_1$, $u_2$, etc.

A Poisson random variable $X \sim P(\lambda)$ or a random variable from a Poisson distribution with parameter $\lambda > 0$ is used in Queueing Theory models, for example, to randomly generate the time a customer will have to wait at a hairdresser's, a clinic, a bus stop, etc., when the statistically average waiting time is equal to $\lambda > 0$. Thus, the parameter $\lambda > 0$ describes the mean as well as the variance of the Poisson random variable (in this distribution, the mean and the variance coincide). It should be noted that the Poisson random variable takes only non-negative integer values.

*A method for generating Poisson random variable.*

Suppose that $u_1, u_2, \ldots \in [0, 1)$ are mutually independent, pseudo-random numbers, uniformly distributed over the interval $[0, 1)$. We calculate the products $u_1$, $u_1u_2$, $u_1u_2u_3$, and so on, until the product is less than $e^{-\lambda}$. Then subtracting one from the number of factors $n$ in the last product (where $e^{-\lambda}$ is still exceeded: $u_1u_2 \cdots u_n \geqslant e^{-\lambda}$) gives $n - 1$, which is the pseudo-random number from the Poisson distribution with parameter $\lambda > 0$. The pseudocode for this algorithm:

```
$a = e^{-\lambda}$;
$r = 1$;
$n = -1$;
**while** $r > a$ **do**
    generate a pseudorandom $u \in [0, 1)$;
    $r = r * u$;
    $n = n + 1$;
**end while**
**return** $n$;
```

Computer generation of random variables from various distributions is also possible using dedicated computer tools.

# 2 Template

The use of the LaTeX template, provided by the Department, is recommended for thesis writing.

## 2.1 Types of papers

Students are required to write different types of written work during their studies. Type of written work is indicated on the cover page of the essay report. Types of work with translations into English are given in Table 3.

Table 3. Types of written work

| Type in Lithuanian | English translation | Level |
|---|---|---|
| Kursinis darbas | Semester Project | BSc |
| Problemų sprendimu grįstas projektas | Problem-Based Project | BSc |
| Problemų sprendimu grįstas projektas II d. | Problem-Based Project. Part II | BSc |
| Bakalauro baigiamasis darbas | Final Bachelor Thesis | BSc |
| Mokslo tiriamasis darbas I/II | Scientific Research I/II | MSc |
| Mokslo tiriamojo darbo projektas | Scientific Research Project | MSc |
| Magistro baigiamasis darbas | Final Master Thesis | MSc |
| *Dalyko ataskaita* | *Subject Related Report* | BSc |
| Objektinis programavimas. Mini-projektas | Object-Oriented Programming. Mini-Project | |
| Objektinis-programavimas. Recenzija | Object-Oriented Programming. Peer-Review | |

Different subjects have different types of essays, for example, mini-project, review, project documentation, statement of requirements etc. Thus, the types of written work in the subjects may be different.

The papers can be written in English or in English. The template includes a special flag to indicate the language. The template assumes that one paper can be completed by up to five students.

## 2.2 Document structure

Summary, Introduction, Conclusions, References, Glossary of Contractual Terms are unnumbered but still must be included in the body of the paper. The template provides a command *sectionWithoutNumber* to create unnumbered chapters in combination with a standard tag such as *sec:intro*.

In some works, such as project reports, a preface is necessary explaining the semester in which the work was completed, thanking for the data or assistance (time) provided. Acknowledgements to external people from other departments of Vilnius University or companies. The prologue may also explain which parts of the work are from a previous stage of the work, if the work is continuous. When the work is carried out by a group of students, the students' signatures can be in the preface page. A special tag is set in the template *signaturesOnTitlePage*, which generates the signature locations either on the cover page or on the preface page.

The abstract is usually 100–200 words. To count words, the Unix command *wc -w fileName* [6] can be used. Template summary, English summary, preface, introduction, conclusions are in separate LaTeX files.

## 2.3 References

All items in the list of references must be cited in the text. Using LaTeX, bibliographic sources can be managed by the tool BibTeX. Sources come in different types. For example, [1] and [7] sources are books, [8] is a journal article, and [9] is a conference paper. By the way, scientific articles, books etc. BibTeX source preformats in *bib* format can be found in the Computer Science Bibliography *dblp* [3].

Only reliable sources of information should be used and referenced. References may also include the source of the data. For example, statistical data may be taken from the European Commission database [4]. Translations of Computer Science terms into Lithuanian can be consulted in the Computing Terms Dictionary [2].

## 2.4 Pseudo-code and parts of software code

It is recommended to represent algorithms in pseudo-code. In the template under the packages *algorithm* and *algorithmic*, it is possible in the pseudo-code to have tags for lines of code. For example, in Algorithm 1, the line 1 marks the pseudo-code condition. By default input and output parameters are denoted by the Lithuanian terms Įvestis and Išvestis, but these can be changed. Algorithms can also be described using another package *algorithm2e*.

**Algorithm 1.** Example of an algorithm

```
Require: $a, b, c \in \mathbb{N}$
Ensure: $x : \{\texttt{true}, \texttt{false}\} \times \mathbb{N}$
 1: if $a < b$ and $b < c$ then
 2:     $a \leftarrow c$
 3:     return $(\texttt{false}, a)$
 4: end if
 5: return $(\texttt{true}, a)$
```

If it is necessary to specify parts of the source code of programs, one may use the *listings* package, an example of which is given in Listing 1. In this document, the Java programming language is enabled as a parameter, although the package supports a number of different paradigms languages.

Listing 1. Example of Java method representation

```java
boolean method(String a, String b){
  if (a.contains(b)) return true;
  return false;
}
```

## 2.5 Preparing images

Prepare your data properly as images (graphs). This can be done with the cross-platform tool Gnuplot [5]. For example, the block of statistical data shown in Table 4 is displayed in Fig. 7 and the image was generated by the Gnuplot tool, employing the code from Listing 2.

Example 1 shows a block of data and its graphical representation generated by the Gnuplot tool.

Listing 2. Gnuplot code example

```
# store as e.g. script.gpi
# execute gnuplot script.gpi

set terminal pdf monochrome
set output 'EUdataLT.pdf'
set style data histogram
set samples 11
set boxwidth 0.5
set style fill pattern border
set style histogram cluster gap 1
set yrange [0:4000]

plot 'dataLT.dat' using 2:xtic(1) title columnheader(2),
    for [i=3:4] '' using i title columnheader(i)
```

**Example 1.** The example starts here. The end of the example is indicated by a square. Although the figures and table belong to the example, they may ‘‘float’’ in the text.

Table 4. Agile students by origin [4]

| | 2010 | 2011 | 2012 |
|---|---|---|---|
| Lithuania | 3103 | 3129 | 3379 |
| Latvia | 1760 | 1979 | 2716 |
| Estonia | 2556 | 2686 | 2819 |

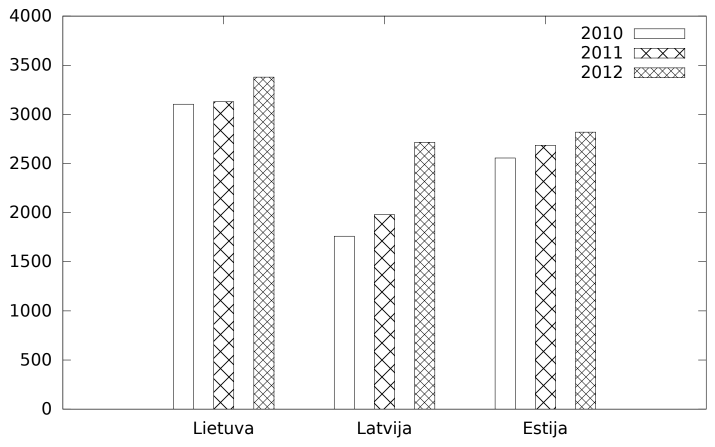

Figure 7. Agile student numbers. The $y$-axis of the graph shows the number of students.

□

A definition can also be prepared in the template. For example, Definition 1 defines a path.

**Definition 1.** The path $k$ is the sequence of points $< t_1, ..., t_n >$ where $t_i \in \mathbb{R} \times \mathbb{R}$.

## 2.6 More complex tables

Additional packages may be needed to describe more complex tables: for long tables *longtable*, for merging columns/rows *multirow*, for text rotation *rotating*. An example of a more complex table is given in Table 5.

Table 5. Example of a more complex table

| Module | ECTS | Student workload | Contact Hours | Private Work Hours | Key Programme Competences | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Generic Competences | | | | | Specific Competences | | | | |
| | | | | | GC1 | GC2 | | GC3 | | SC1 | | SC2 | | SC3 |
| | | | | | Learning Outcomes | | | | | | | | | |
| | | | | | GC1-1 | GC2-1 | GC2-2 | GC3-1 | GC3-2 | SC1-1 | SC1-2 | SC2-1 | SC2-2 | SC3-1 |
| I YEAR | 60 | 1600 | 358 | 1242 | | | | | | | | | | |
| 1 SEMESTER | 30 | 800 | 221 | 579 | | | | | | | | | | |
| Management | 5 | 125 | 50 | 75 | | X | | | X | X | | | | X |

## References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorihms*. The MIT Press, 3rd edition, 2009.

[2] Valentina Dagienė, Gintautas Grigas, and Tatjana Jevsikova. Anglų–lietuvių kalbų kompiuterijos žodynas. Vilniaus universitetas. Matematikos ir informatikos institutas, 2012. http://ims.mii.lt/ALK%C5%BD/.

[3] dblp. Computer Science Bibliography. http://dblp.uni-trier.de/db/.

[4] European Commission. Eurostat. Browse / search database, 2014. http://epp.eurostat.ec.europa.eu/portal/page/portal/statistics/search_database.

[5] Gnuplot homepage, November 2014. http://www.gnuplot.info/.

[6] O'Reilly Media, Inc. *Linux in a Nutshell. Linux Command Directory*, 5th edition edition, 2014. http://www.linuxdevcenter.com/cmd/.

[7] Philippe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers, 2002.

[8] Darius Sidlauskas, Simonas Saltenis, and Christian S. Jensen. Processing of extreme moving-object update and query workloads in main memory. *VLDB J.*, 23(5):817–841, 2014.

[9] Laurynas Speicys, Christian S. Jensen, and Augustas Kligys. Computational data modeling for network-constrained moving objects. In *ACM-GIS 2003, Proceedings of the Eleventh ACM International Symposium on Advances in Geographic Information Systems, New Orleans, Louisiana, USA, November 7-8, 2003*, page 118–125, 2003.