Parallel implementation of the ellipsoid method for optimization problems of large dimension*

Anatoly Panyukov, Valentin Golodov, and Vyacheslav Bezborodov

South Ural State University, 76, Lenin prospect, Chelyabinsk, 454080, Russia
{anatoly.panyukov,avaksa,vyacheslav.bezborodov}@gmail.com

Abstract. A parallel implementation of the ellipsoid method is discussed in the paper. The method is important in theoretical researches but for practical issues it has some lacks such as low rate of convergence and round-error sensitiveness.
Authors suggest reducing the influence of the low rate of convergence at the expense of parallel computations. To prevent the loss of significance authors use arbitrary precision floating point numbers.
Parallel matrix operations were implemented as C shared library using OpenMP interface. The MPFR library was used for multiple-precision floating-point computations with correct rounding.

Keywords: linear optimization, ellipsoids method, parallel programming, arbitrary precision

1 Introduction

Consider a method that uses approximation of the localization sets. The method is based on the following geometric observation.

Let H be a positive-definite symmetric $(n \times n)$-matrix. Consider the ellipsoid

$$E(H, \overline{x}) = \left\{ x \in \mathbb{R}^n | \langle H^{-1}(x - \overline{x}), x - \overline{x} \rangle \leq 1 \right\}.$$

Choose the direction $g \in \mathbb{R}^n$ and consider the half of the ellipsoid defined by corresponding hyperplane:

$$E_+ = \left\{ x \in E(H, \overline{x}) | \langle g, \overline{x} - x \rangle \geq 0 \right\}.$$

Obviously this set is contained in another ellipsoid having a volume less than a volume of ellipsoid $E(H, \overline{x})$.

Lemma 1. *Proof. [6] Lets define*

$$\overline{x}_+ = \overline{x} - \frac{1}{n+1} \cdot \frac{H_k g_k}{\langle H_k g_k, g_k \rangle^{1/2}},$$

* The work is supported by RFBR grant 16-31-00108\16 "Solutions to fundamental issues of usability of extended and arbitrary-precision arithmetics".

$$H_+ = \frac{n^2}{n^2-1}\left(H - \frac{2}{n+1}\cdot\frac{H_k g_k g_k^T H_k}{\langle H_k g_k, g_k\rangle}\right).$$

Then $E_+ \subset E(H_+, \overline{x}_+)$ *and*

$$vol_n E(H_+, \overline{x}_+) \le \left(1 - \frac{1}{(n+1)^2}\right)^{n/2} vol_n E(H, \overline{x}).$$

The volume of the ellipsoid $E(H_+, \overline{x}_+)$:

$$\frac{vol_n E(H_+, \overline{x}_+)}{vol_n E(H, \overline{x})} = \left[\frac{\det H_+}{\det H}\right]^{1/2} = \left[\left(\frac{n^2}{n^2-1}\right)^n \frac{n-1}{n+1}\right]^{1/2} =$$

$$= \left[\frac{n^2}{n^2-1}\left(1 - \frac{2}{n+1}\right)^{\frac{1}{n}}\right]^{n/2} \le \left[\frac{n^2}{n^2-1}\left(1 - \frac{2}{n(n+1)}\right)\right]^{n/2} =$$

$$= \left[\frac{n^2(n^2+n-2)}{n(n-1)(n+1)^2}\right]^{n/2} = \left[1 - \frac{1}{(n+1)^2}\right]^{n/2}.$$

So $E(H_+, \overline{x}_+)$ is an ellipsoid of minimal volume which contains a half of the original ellipsoid E_+.

These statements can be represented as *an algorithm of the ellipsoid method.*

Ellipsoid method

1. Choose $y_0 \in \mathbb{R}$ and $\mathbb{R} > 0$ so that euclidian ball $B_2(y_0, \mathbb{R}) \supseteq Q$. Let $H_0 = \mathbb{R}^2 \cdot I_n$, $y^* = 0$, $f^* = \inf$.
2. k-th iteration ($k \ge 0$):

$$g_k = \begin{cases} g(y_k), \text{ if } y_k \in Q, \\ \hat{g}(y_k), \text{ if } y_k \notin Q, \end{cases}$$

$$y_{k+1} = y_k - \frac{1}{n+1}\cdot\frac{H_k g_k}{\langle H_k g_k, g_k\rangle^{1/2}},$$

$$y^* = y_{k+1}, f^* = f(y^*), \text{ if } f(y_{k+1}) < f^*,$$

$$H_{k+1} = \frac{n^2}{n^2-1}\left(H_k - \frac{2}{n+1}\cdot\frac{H_k g_k g_k^T H_k}{\langle H_k g_k, g_k\rangle}\right).$$

Lets evaluate the effectiveness of the ellipsoid method. Let $Y = \{y_k\}_{k=0}^{\infty}$ and let X be a feasible part of Y subsequence:

$$X = Y \cap Q.$$

Let $f_k^* = \min_{0 \le j \le k} f(x_j)$.

Theorem 1. *Proof. [6] Let f be a Lipschitz function with domain $B_2(x^*, \mathbb{R})$ bounded by a Lipschitz constant M. Then for $i(k) > 0$ the following inequality is true:*

$$f^*_{i(k)} - f^* \leq M\mathbb{R}\left(1 - \frac{1}{(n+1)^2}\right)^{k/2} \cdot [vol_n B_0(x_0, \mathbb{R}) vol_n Q]^{1/n}.$$

2 Implementation

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion. By design, C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems [5].

Since C doesn't support complex data structures and provide low-level mechanism of grouping variables called plain old data structure (POD) for this, we need to define a data type which represents matrices in source code. It can be achieved by using C *struct* keyword (see Listing 1).

```
/**
 * Matrix structure.
 */
struct mtx
{
    mpfr_t* storage;     ///!< internal storage
    size_t nrows;        ///!< number of rows
    size_t ncols;        ///!< number of columns
};
```

Listing 1. Matrix representation: *storage* is a pointer to allocated memory block, *nrows* and *ncols* are used for pointer offset.

There are at least two methods to declare N by M 2D array in C. The first one involves N allocation requests, one for each row, plus one request for array of row arrays. The bucket locations in individual rows are contiguous, but rows are not necessarily contiguous in heap space. By using this method element accessing can be done with `array[i][j]` syntax which is more preferable for humans. The second one is considered to be memory efficient: there is the only one N by M array allocation. Really this is a large 1-dim array of values onto which we will map 2D accesses. In that case we cannot use `array[i][j]` syntax because the compiler has no idea where the next row starts within this chunk of heap space, so must use single index value that is calculated using row and column index values and the column dimension.

Note that the type of elements in internal storage is `mpfr_t`. It is a special data type from The GNU MPFR Library[1] – a portable library written in C

[1] http://www.mpfr.org/

for arbitrary precision arithmetic on floating-point numbers. It is based on the GNU MP library. It aims to provide a class of floating-point numbers with precise semantics. The main characteristics of MPFR, which make it differ from most arbitrary precision floating-point software tools, are:

- the MPFR code is portable, i.e., the result of any operation does not depend on the machine word size `mp_bits_per_limb` (64 on most current processors);
- the precision in bits can be set exactly to any valid value for each variable (including very small precision);
- MPFR provides the four rounding modes from the IEEE 754-1985 standard, plus away-from-zero, as well as for basic operations as for other mathematical functions.

In particular, with a precision of 53 bits, MPFR is able to exactly reproduce all computations with double-precision machine floating-point numbers (e.g., double type in C, with a C implementation that rigorously follows Annex F of the ISO C99 standard and `FP_CONTRACT` pragma set to OFF) on the four arithmetic operations and the square root, except the default exponent range is much wider and subnormal numbers are not implemented (but can be emulated).

In order to manipulate represented matrix POD structure, we provide simple basic interface to our library. Some methods are shown in Listing 2.

```
// memory handling
int mtx_init(struct mtx* const m,
        size_t rows, size_t columns, mpfr_prec_t prec);

int mtx_clear(struct mtx const m);

// matrix I/O
int mtx_fprint(FILE* stream, struct mtx const m);
int mtx_fscan(FILE* stream, struct mtx m, char const* delim);

// assignment
int mtx_fill(struct mtx m, mpfr_t val, mpfr_t diagval);
int mtx_fill_d(struct mtx m, double val, double diagval);

// copying
int mtx_copy(struct mtx rop, struct mtx const op);

// multiplication
int mtx_mul(struct mtx rop,
        struct mtx const op1, struct mtx const op2);

int mtx_mulval(struct mtx rop, struct mtx const op1, mpfr_t op2);

// addition
int mtx_add(struct mtx rop,
```

```
        struct mtx const op1, struct mtx const op2);

// transposition
int mtx_tr(struct mtx rop, struct mtx const op);
```

Listing 2. Matrix library interface.

Let us give a short brief. All the functions are grouped according to a certain kind of operations:

- **memory handling** – matrix creation/destroying with memory allocation/freeing (note the precision parameter; it is the number of bits used to represent the significand of a floating-point number; the precision can be any integer between `MPFR_PREC_MIN` and `MPFR_PREC_MAX`);
- **input/output facilities** – reading/writing matrices from/to different sources;
- **assignment operations** – filling matrices by certain value (note `diagval` parameter which is used for diagonal elements filling if matrix is square);
- **basic operations** – multiplication (matrix product), multiplication by value, addition, transposition and copying.

Note the return value of each function. The library is designed to report about a problem by returning 0 on success and non-zero otherwise.

Due to specific nature, the most matrix operations can be performed in parallel without any data races, resource acquiring and minimal data sharing. That's why the major part of the operations from Listing 2 are parallelized to be efficient.

There are a lot of approaches to organize parallelism. One of the most popular, modern and flexible is OpenMP (Open Multi-Processing). OpenMP is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran,[4] on most platforms, processor architectures and operating systems, including Solaris, AIX, HP-UX, Linux, OS X, and Windows. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer.

An application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and Message Passing Interface (MPI), such that OpenMP is used for parallelism within a (multi-core) node while MPI is used for parallelism between nodes. There have also been efforts to run OpenMP on software distributed shared memory systems,[3] to translate OpenMP into MPI[1, 2] and to extend OpenMP for non-shared memory systems.

Let us to show a way how our matrix library encapsulates creation a number of workers by using OpenMP API behind the scene. Lets look through the matrix multiplication method. The interface of the method is shown below.

```
int mtx_mul(struct mtx rop,
        struct mtx const op1, struct mtx const op2)
```

The function accepts two operands `op1` and `op2` and saves the result of the operation in `rop`. Firstly, we need to ensure that dimensions of the accepted values is correct.

```
if (rop.nrows != op1.nrows || rop.ncols != op2.ncols)
    return -1;

if (op1.ncols != op2.nrows)
    return -1;
```

Next, we marks internal loops as *OpenMP parallel region*. Work-sharing constructs used to specify how to assign independent work to one or all of the threads. Directives `omp for` or `omp do` are used to split up loop iterations among the threads, also called loop constructs.

```
size_t i, j, k;

#pragma omp parallel for
    shared(rop) private(i,j,k) schedule(static)

    for (i = 0; i < op1.nrows; ++i)
    {
        for (j = 0; j < op2.ncols; ++j)
        {
            mpfr_t* const prop = rop.storage + i * rop.ncols + j;
            mpfr_set_ui(*prop, 0, MPFR_RNDN);

            for (k = 0; k < op1.ncols; ++k)
            {
                // do multiplication ...
            }
        }
    }
```

Since OpenMP is a shared memory programming model, most variables in OpenMP code are visible to all threads by default. But sometimes private variables are necessary to avoid race conditions and there is a need to pass values between the sequential part and the parallel region (the code block executed in parallel), so data environment management is introduced as data sharing attribute clauses by appending them to the OpenMP directive. Some types of clauses are:

- **shared** – the data within a parallel region is shared, which means visible and accessible by all threads simultaneously. By default, all variables in the work sharing region are shared except the loop iteration counter.
- **private** – the data within a parallel region is private to each thread, which means each thread will have a local copy and use it as a temporary variable.

A private variable is not initialized and the value is not maintained for use outside the parallel region. By default, the loop iteration counters in the OpenMP loop constructs are private.

One of the *scheduling clauses* is `schedule(type, chunk)`. This is useful if the work sharing construct is a do-loop or for-loop. The iteration(s) in the work sharing construct are assigned to threads according to the scheduling method defined by this clause. By using the `static` schedule, all the threads are allocated iterations before they execute the loop iterations. The iterations are divided among threads equally by default. However, specifying an integer for the parameter chunk will allocate chunk number of contiguous iterations to a particular thread.

Finally, the innermost loop do the multiplication routine.

```
mpfr_t* const pop1 = op1.storage + i * op1.ncols + k;
mpfr_t* const pop2 = op2.storage + k * op2.ncols + j;

mpfr_t tmp;
mpfr_init2(tmp, prec);

mpfr_mul(tmp, *pop1, *pop2, MPFR_RNDN);
mpfr_add(*prop, *prop, tmp, MPFR_RNDN);

mpfr_clear(tmp);
```

There are a lot of specific functions from MPFR library here, such as multiplication and addition. Note using the `MPFR_RNDN` parameter. It is called *rounding mode*. The following five rounding modes are supported by GNU MPFR Library:

1. **MPFR_RNDN** – round to nearest (roundTiesToEven in IEEE 754-2008),
2. **MPFR_RNDZ** – round toward zero (roundTowardZero in IEEE 754-2008),
3. **MPFR_RNDU** – round toward plus infinity (roundTowardPositive in IEEE 754-2008),
4. **MPFR_RNDD** – round toward minus infinity (roundTowardNegative in IEEE 754-2008),
5. **MPFR_RNDA** – round away from zero.

That is the way how the most matrix library functions handle matrices of large dimension. Of course, not all the functions can be performed in parallel (matrix I/O facilities is still consequent for obvious reasons).

2.1 Checking the PDF File

Kindly assure that the Contact Volume Editor is given the name and email address of the contact author for your paper. The Contact Volume Editor uses these details to compile a list for our production department at SPS in India. Once the files have been worked upon, SPS sends a copy of the final pdf of

each paper to its contact author. The contact author is asked to check through the final pdf to make sure that no errors have crept in during the transfer or preparation of the files. This should not be seen as an opportunity to update or copyedit the papers, which is not possible due to time constraints. Only errors introduced during the preparation of the files will be corrected.

This round of checking takes place about two weeks after the files have been sent to the Editorial by the Contact Volume Editor, i.e., roughly seven weeks before the start of the conference for conference proceedings, or seven weeks before the volume leaves the printer's, for post-proceedings. If SPS does not receive a reply from a particular contact author, within the timeframe given, then it is presumed that the author has found no errors in the paper. The tight publication schedule of LNCS does not allow SPS to send reminders or search for alternative email addresses on the Internet.

In some cases, it is the Contact Volume Editor that checks all the final pdfs. In such cases, the authors are not involved in the checking phase.

2.2 Additional Information Required by the Volume Editor

If you have more than one surname, please make sure that the Volume Editor knows how you are to be listed in the author index.

2.3 Copyright Forms

The copyright form may be downloaded from the "For Authors" (Information for LNCS Authors) section of the LNCS Website: `www.springer.com/lncs`. Please send your signed copyright form to the Contact Volume Editor, either as a scanned pdf or by fax or by courier. One author may sign on behalf of all of the other authors of a particular paper. Digital signatures are acceptable.

3 Paper Preparation

Springer provides you with a complete integrated LaTeX document class (`llncs.cls`) for multi-author books such as those in the LNCS series. Papers not complying with the LNCS style will be reformatted. This can lead to an increase in the overall number of pages. We would therefore urge you not to squash your paper.

Please always cancel any superfluous definitions that are not actually used in your text. If you do not, these may conflict with the definitions of the macro package, causing changes in the structure of the text and leading to numerous mistakes in the proofs.

If you wonder what LaTeX is and where it can be obtained, see the "*LaTeX project site*" (`http://www.latex-project.org`) and especially the webpage "*How to get it*" (`http://www.latex-project.org/ftp.html`) respectively.

When you use LaTeX together with our document class file, `llncs.cls`, your text is typeset automatically in Computer Modern Roman (CM) fonts. Please

do *not* change the preset fonts. If you have to use fonts other than the preset fonts, kindly submit these with your files.

Please use the commands `\label` and `\ref` for cross-references and the commands `\bibitem` and `\cite` for references to the bibliography, to enable us to create hyperlinks at these places.

For preparing your figures electronically and integrating them into your source file we recommend using the standard LaTeX `graphics` or `graphicx` package. These provide the `\includegraphics` command. In general, please refrain from using the `\special` command.

Remember to submit any further style files and fonts you have used together with your source files.

Headings. Headings should be capitalized (i.e., nouns, verbs, and all other words except articles, prepositions, and conjunctions should be set with an initial capital) and should, with the exception of the title, be aligned to the left. Words joined by a hyphen are subject to a special rule. If the first word can stand alone, the second word should be capitalized.

Here are some examples of headings: "Criteria to Disprove Context-Freeness of Collage Language", "On Correcting the Intrusion of Tracing Non-deterministic Programs by Software", "A User-Friendly and Extendable Data Distribution System", "Multi-flip Networks: Parallelizing GenSAT", "Self-determinations of Man".

Lemmas, Propositions, and Theorems. The numbers accorded to lemmas, propositions, and theorems, etc. should appear in consecutive order, starting with Lemma 1, and not, for example, with Lemma 11.

3.1 Figures

For LaTeX users, we recommend using the *graphics* or *graphicx* package and the `\includegraphics` command.

Please check that the lines in line drawings are not interrupted and are of a constant width. Grids and details within the figures must be clearly legible and may not be written one on top of the other. Line drawings should have a resolution of at least 800 dpi (preferably 1200 dpi). The lettering in figures should have a height of 2 mm (10-point type). Figures should be numbered and should have a caption which should always be positioned *under* the figures, in contrast to the caption belonging to a table, which should always appear *above* the table; this is simply achieved as matter of sequence in your source.

Please center the figures or your tabular material by using the `\centering` declaration. Short captions are centered by default between the margins and typeset in 9-point type (Fig. 1 shows an example). The distance between text and figure is preset to be about 8 mm, the distance between figure and caption about 6 mm.

To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

If screenshots are necessary, please make sure that you are happy with the print quality before you send the files.

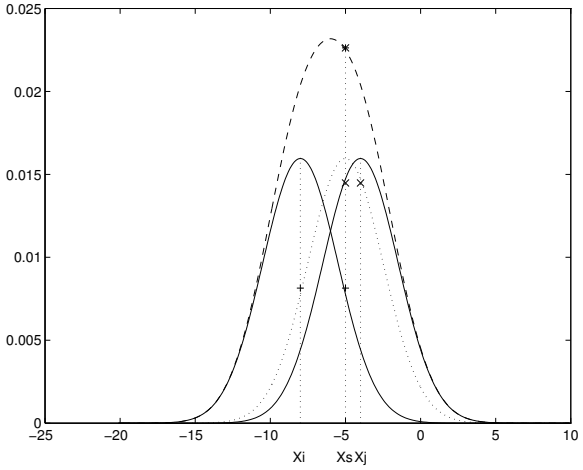

Fig. 1. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

Please define figures (and tables) as floating objects. Please avoid using optional location parameters like "`[h]`" for "here".

Remark 1. In the printed volumes, illustrations are generally black and white (halftones), and only in exceptional cases, and if the author is prepared to cover the extra cost for color reproduction, are colored pictures accepted. Colored pictures are welcome in the electronic version free of charge. If you send colored figures that are to be printed in black and white, please make sure that they really are legible in black and white. Some colors as well as the contrast of converted colors show up very poorly when printed in black and white.

3.2 Formulas

Displayed equations or formulas are centered and set on a separate line (with an extra line or halfline space above and below). Displayed expressions should be numbered for reference. The numbers should be consecutive within each section or within the contribution, with numbers enclosed in parentheses and set on the

right margin – which is the default if you use the *equation* environment, e.g.,

$$\psi(u) = \int_o^T \left[\frac{1}{2}\left(\Lambda_o^{-1}u, u\right) + N^*(-u)\right] dt\,. \tag{1}$$

Equations should be punctuated in the same way as ordinary text but with a small space before the end punctuation mark.

3.3 Footnotes

The superscript numeral used to refer to a footnote appears in the text either directly after the word to be discussed or – in relation to a phrase or a sentence – following the punctuation sign (comma, semicolon, or period). Footnotes should appear at the bottom of the normal text area, with a line of about 2 cm set immediately above them.[2]

3.4 Program Code

Program listings or program commands in the text are normally set in typewriter font, e.g., CMTT10 or Courier.

Example of a Computer Program

```
program Inflation (Output)
  {Assuming annual inflation rates of 7%, 8%, and 10%,...
   years};
   const
     MaxYears = 10;
   var
     Year: 0..MaxYears;
     Factor1, Factor2, Factor3: Real;
   begin
     Year := 0;
     Factor1 := 1.0; Factor2 := 1.0; Factor3 := 1.0;
     WriteLn('Year  7% 8% 10%'); WriteLn;
     repeat
       Year := Year + 1;
       Factor1 := Factor1 * 1.07;
       Factor2 := Factor2 * 1.08;
       Factor3 := Factor3 * 1.10;
       WriteLn(Year:5,Factor1:7:3,Factor2:7:3,Factor3:7:3)
     until Year = MaxYears
end.
```

(Example from Jensen K., Wirth N. (1991) Pascal user manual and report. Springer, New York)

[2] The footnote numeral is set flush left and the text follows with the usual word spacing.

3.5 Citations

For citations in the text please use square brackets and consecutive numbers: [7], [8], [10] – provided automatically by LaTeX's `\cite ... \bibitem` mechanism.

3.6 Page Numbering and Running Heads

There is no need to include page numbers. If your paper title is too long to serve as a running head, it will be shortened. Your suggestion as to how to shorten it would be most welcome.

4 LNCS Online

The online version of the volume will be available in LNCS Online. Members of institutes subscribing to the Lecture Notes in Computer Science series have access to all the pdfs of all the online publications. Non-subscribers can only read as far as the abstracts. If they try to go beyond this point, they are automatically asked, whether they would like to order the pdf, and are given instructions as to how to do so.

Please note that, if your email address is given in your paper, it will also be included in the meta data of the online version.

5 BibTeX Entries

The correct BibTeX entries for the Lecture Notes in Computer Science volumes can be found at the following Website shortly after the publication of the book:
`http://www.informatik.uni-trier.de/~ley/db/journals/lncs.html`

Acknowledgments. The heading should be treated as a subsubsection heading and should not be assigned a number.

6 The References Section

In order to permit cross referencing within LNCS-Online, and eventually between different publishers and their online databases, LNCS will, from now on, be standardizing the format of the references. This new feature will increase the visibility of publications and facilitate academic research considerably. Please base your references on the examples below. References that don't adhere to this style will be reformatted by Springer. You should therefore check your references thoroughly when you receive the final pdf of your paper. The reference section must be complete. You may not omit references. Instructions as to where to find a fuller version of the references are not permissible.

We only accept references written using the latin alphabet. If the title of the book you are referring to is in Russian or Chinese, then please write (in Russian) or (in Chinese) at the end of the transcript or translation of the title.

The following section shows a sample reference list with entries for journal articles [7], an LNCS chapter [8], a book [9], proceedings without editors [10] and [11], as well as a URL [12]. Please note that proceedings published in LNCS are not cited with their full titles, but with their acronyms!

References

1. Wang, J., Hu, C., Zhang, J., Li, J.: OpenMP compiler for distributed memory architectures. Science China Information Sciences. Springer Publishing. 53 (5): 932944. (2010)
2. Basumallik, A., Min, S., Eigenmann, R.: Programming Distributed Memory Sytems [sic] using OpenMP. Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium. New York: IEEE Press. (2007)
3. Costa, J.J.; et al.: Running OpenMP applications efficiently on an everything-shared SDSM. Journal of Parallel and Distributed Computing. Elsevier. 66 (5), 647658 (2006)
4. Gagne, A.S., Peter Baer Galvin, G.: Operating system concepts (9th ed.). Hoboken, N.J.: Wiley. pp. 181182.
5. Kernighan, B.W., Ritchie, D.M.: The C Programming Language (1st ed.). Englewood Cliffs, NJ: Prentice Hall. (1978)
6. Nesterov Ju. E. Metody vypukloj optimizacii. M.: Izdatel'stvo MCNMO, 2010. 281 s.
7. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
8. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
9. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
10. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
11. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
12. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`

Appendix: Springer-Author Discount

LNCS authors are entitled to a 33.3% discount off all Springer publications. Before placing an order, the author should send an email, giving full details of his or her Springer publication, to `orders-HD-individuals@springer.com` to obtain a so-called token. This token is a number, which must be entered when placing an order via the Internet, in order to obtain the discount.

7 Checklist of Items to be Sent to Volume Editors

Here is a checklist of everything the volume editor requires from you:

- ☐ The final LaTeX source files
- ☐ A final PDF file
- ☐ A copyright form, signed by one author on behalf of all of the authors of the paper.
- ☐ A readme giving the name and email address of the corresponding author.