MGVB: a new proteomics toolset for fast and efficient data analysis

Metodi V. Metodiev[1,1*]

[1]School of Life Sciences, Genomics and Computational Biology Group, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom.
[2]Translational Oncology Unit, Medical University of Pleven, Pleven, 10587, Bulgaria.

Corresponding author(s). E-mail(s): mmetod@essex.ac.uk;

Abstract

MGVB is a collection of tools implemented in the programming language C. It covers proteomics data processing from in silico digestion of protein sequences to identification of postranslational modifications and solving the protein inference problem. The library is developed with efficiency in mind. It enables very fast analysis at a fraction of the resources cost typically required by existing commercial and free tools. MGVB, as it is a native application, can be faster than existing proteomics tools such as MaxQuant and MSFragger and, in the same time, finds very similar, in some cases even larger number of peptides at a chosen level of statistical significance. It implements a probabilistic scoring function to match spectra to sequences, and a novel combinatorial search strategy for finding post-translational modifications, and a Bayesian approach to locate modification sites. This report describes the algorithms behind the tools, present benchmarking data sets analysis results comparing MGVB performance to MaxQuant/Andromeda, and provides step by step protocols for using it in typical analytical scenarios. A static library of object files exposing the important functions and data structures is also provided. The toolset is provided free to download and use for academic research and in software projects, but is not open source at the present. It is the intention of the author that the library will be made open source in the near future—following rigorous evaluations and feedback from the proteomics research community.

Keywords: software for computational proteomics, mass spectrometry, MS/MS search engine, shotgun analysis, post-translational modifications

1 Introduction

Proteomics aims to identify and quantify the proteins at genome scale (reviewed in [**?**]). At the present the technology of choice—almost universally employed in large-scale proteomics studies—is high-resolution mass spectrometry of proteolytic digests. Modern hybrid mass spectrometers, when interfaced with nano-scale liquid chromatography, generate many thousands tandem spectra of peptide precursors per run; typical projects often generate more than a million spectra (see for example refs).

Raw mass spectra are processed by computational pipelines to identify and quantify the proteins. These pipelines utilise search engines that typically match peptide fragmentation spectra to the theoretically predicted sequence specific fragments, i.e. predicted from genomic sequences. Matching is a probabilistic process prone to false positive and false negative results (refs). To account for this, search engines such as Mascot and Andromeda, apply filters, most commonly based on the number of reverse database hits (refs).

With the advance of instrumentation data processing is becoming the bottleneck of proteomics workflows. To illustrate: practitioners of the art know that at the present a nano-LC-MS/MS experiment will generate 20,000 to 40,000 spectra in an hour or two but MaxQuant or Mascot or Sequest analysis of the file would take substantial time even on a powerful multicore workstation. Even when MaxQuant is run under Mono on a high-performance computing cluster, analyses programmed to search for post-translational modifications take longer than the time needed to generate the raw files.

Even more challenging computationally is the recently proposed open search approach, which attempts to identify peptides modified by unknown groups (refs). One implementation of this approach is the MSFagger algorithm (ref). It uses precomputed indexed database of peptide fragments to search the MS/MS spectra. MSFragger was publicised as an ultrafast algorithm but it also requires more time to process the data than the hardware needs to generate them.

Part of the reason for this level of performance is technical: most freely-available search engines are implemented as non native applications running in virtual machines, in part to avoid platform dependence. This puts substantial overheads on memory requirements and execution speed. Another reason is the relative ease of developing in Java and C#, as they are object oriented garbage-collected languages with automatic memory management. However the platform dependence is a less severe problem nowadays as it used to be in the past, and ease of development does not justify inferior performance in scientific applications. A native search engine could potentially provide much faster processing time and would come with the added benefit of much smaller

carbon footprint as it would use much less memory and processor time. This was the initial motivation behind the MGVB project: to develop a native search engine that could perform as well as the state of the art programs in terms of peptide and protein coverage, but do it faster and with less energy consumption. As it turned out—in the process of development—a new approach to post-translational modification analysis was conceived. It is a combinatorial search that combines the advantages of the open search but is much more effective as it can handle peptides modified on more that one site by more than one type of modification, a capability that open search algorithms such as the MSFragger lack by design.

MGVB consists of several program, which prepare peptide sequences, extract raw spectral data to proprietary binary files, match spectra, infer proteins and compute spectral counts to quantify the identified proteins. The scorer program implements probabilistic algorithms for spectra assignment to sequences, similar to MaxQuant/Andromeda (ref). To speed up modified peptides identification, for each candidate precursor sequence, predicted fragments are packaged in a balanced binary search tree, which is used for matching the spectral peaks. A binomial probability score is assigned based on the number of fragments matched. Modifications sites are determined by a Bayesian updating algorithm, which considers assigned fragments as experimental evidences of possible PTM localisation models to compute the final posterior probabilities of localisation. The combinatorial search algorithm, named scorer_mpi, uses a precomputed database of combinations of up to 3 different post-translational modifications masses called mod_comb. The current compilation of mod_comb consists of more than 100,000 combinations of $ different modifications from the Unimod database (Unimod is described in ref). Scorer_mpi executes an open search to identify a set of candidate precursors. It then calculate the delta mass for each of the candidates and searches the mod-comb database for combinations matching the delta-mass. Once such combination is identified, scorer_mpi uses the same Bayesian network algorithm as the socrer program to accurately determine the location of each modification from the combination.

An additional advantage of this approach is that only the initial precursor search needs to be restricted to high-mass accuracy. The search for MS/MS fragments matches does not need such accuracy. This makes the large amount of raw data acquired in the High/Low mode of analysis amenable to processing by scorer_mpi.

2 Results and Discussion

MGVB was tested with a collection of raw data files obtained from experiments with human cell lines. An LTQ/Orbitrap Velos isntrument was used to generate the data as described in refs. The results were obtained by analysing data from immunoprecipitation experiments using GFP-tagged Scribble as bait expressed in human HEK293 cells as described in Metodieva et al (2016). Both, high/low and high/high acquisition modes were used to generate the

data. The performance of MGVB was compared to MaxQuant, version XXX running on the same hardware.

The following performance metrics were compared: number of peptides identified at 1% FDR, number of proteins identified at 1% FDR, total number of MS/MS spectra assigned at 1% FDR, number of PTM identified for the target protein Scribble (see below for details of experimental context), speed: time for completing the different steps of the analysis, memory consumption, CPU time.

The results from these comparisons are presented in figures 1- 3 and in tables 1- 3. Fig. 1 shows the peptide and protein identification performance of MGVB compared to MaxQuant/Andromeda. MGVB and MaxQuant find very similar numbers of peptides and proteins at the chosen FDR. The number of MS/MS spectra assigned to the bait protein—which is the most abundant protein in the sample by a large margin—is also very similar. Similarly, the spectral counts for the known Scribble-interacting proteins GIT1 and ARHGEF7 are very close (shown in Table 1).

The quantitative agreement between MaxQuant and MGVB is illustrated on Fig. 2. In Fig 2A the spectral counts for Scribble, GIT1, ARHGEF7 and several other proteins identified in the immunoprecipitation experiments described in Metodieva et al (2016) are used to compute correlation between MaxQuant and MGVB. Figure 2B shows a scatterplot and correlation coefficient for the peptide score for more than XXXX MS/MS spectra acquired in the immunoprecipitation experiments. As the figure shows there is excellent agreement between MaxQuant and MGVB in both comparisons, the quantitative estimation of protein abundance by spectral counting, and the peptide score.

Fig. 3 compares the performance of MGVB and MaxQuant in terms of execution speed, memory consumption and CPU usage. MaxQuant was run under Mono on the same hardware configuration as MGVB. Speed of execution was measured for two specific tasks: very stringent search, typically employed in projects aiming to quantify protein abundance but not interested in post-translational modification analysis, and PTM-focused projects. In the later case MaxQuant and MGVB were set to search for up to 3 PTMs per peptide and the modifications were set to N-terminal acetylation, methionine oxidation and STY phosphorylation. MGVB outperformed MaxQuant in speed and consumed much less memory and CPU time per run in all benchmarking experiments.

The open search capabilities of MGVB are demonstrated with data in Table 2. MGVB can perform combinatorial PTM searches to identify hundreds of different modifications in two modes of operation: in an unrestricted mode it searches against the entire genome of the organism under study. This is challenging and requires a high-performance computer. Typically, in such experiments MGVB was run on up to 40 cpus using its inbuilt high-performance message-passing functionality. An alternative mode, the focused search, restricts the analysis to a subset of sequences selected from a preceding

ultrafast stringent search. Typically, MGVB was set to initially search without any modification allowed and no missed cleavages. Such searches complete under a minute on a multicore workstation (8 cores). Proteins identified in such searches are then subjected to a focused combinatorial search. Table 2 shows that the focused approach correctly identifies all phosphorylation sites known for Scribble and suggests several novel modifications.

Compared to MGVB, recently published open search algorithms such as MSFragger (ref) suffer from two fundamental limitations. First, they require high mass accuracy for fragment masses. MSFragger, for example, uses a precomputed index of y and b fragment masses which have to be matched with very high mass accuracy. This imposes strict restriction on the analytical platforms that can be used in the process of generating the data and makes the analysis of legacy high/low data difficult. Second, existing open search tools identify candidate peptide modifications by matching the delta mass of the identified peptide to candidate modification in a process that is inherently limited to recognising the modification as a single added group. A lengthy post processing employing machine learning is required to make sense of the initial assignments. For example, a delta mass of W units cannot be immediately assigned to a modification of type A on residue X plus a modification of type B on residue Y because the algorithm has no way of learning what the individual masses A and B are from their sum. It will report the delta mass and the peptide sequence but it would be up to the researcher to hunt for the identities of A and B. Even more limiting, to the extend of making open search engine unusable for such cases, is the fact that if there are two modification on two different residues, many of the spectral peaks corresponding to y and b fragments will not be assigned by the open search engine.

In contrast, MGVB would use the scorer_mpi module to execute a combinatorial search, which would directly identify the two modifications, A and B, and will localise them to the correct residues—all in a single step analysis.

This is illustrated in Fig. 4 with the example of Scribble and its post-translational modifications. An open search would have not identified the doubly phosphorylated peptides and the phospho/MetOx modified peptides as such because all singly modified fragment ions would have been missed.

3 Methods

This section summarises the implementation and some of the algorithms used in MGVB.

3.1 Implementation

As stated earlier MGVB is implemented in C and compiled in a collection of binary executable files and a library of objective files, which can be used on hardware running the Linux operating system. There is one exception, the program used to extract spectra from the proprietary raw files is implemented in C# in order to be able to use the API provided by Thermo Fisher Scientific,

but availability of .Net is not required to run it as the program is packaged into an executable file. Table **??** lists the different programs, and gives brief specifications of their typical use. Appendix 1 list in more details the various functions utilised by the programs and provides APIs for calling them in custom code.

Table 1 List of MGVB modules

Name	Description
digest	Performs in silico digestion of sequences from FASTA files.
mod_pep	Generate modified peptides sequences.
toSQL_proteins	Create sqlite table with protein sequences from FASTA files.
toSQL_peptides	Create sqlite table with peptide sequences from digested FASTA files.
toSQL_mod_pep	Creates sqlite table with modified peptide sequences.
extractRaw	Extracts from raw files and writes MS2 and MS1 files to disk[1] .
parseMS	Parses MS2 files to proprietary binary mms files.
scorer	Uses mms files and sqlite database to search for peptide matches[2].
scorer_mpi	Uses mms and two sqlite databases to search for combinations of PTM[3].
select_by_prob	Filters candidate peptides based on score and other criteria.
mgvb	Runs the above to perform automatic analysis using a config.rms file.

[1]MS1 and MS2 files are text files that can be used by various proteomics search engines (ref).

[2]Scorer uses the OpenMP library for shared memory parallel processing.

[3]Scorer_mpi uses the OpenMPI library for message passing parallel computations.

In addition to the binary files included in the table above, there is a collection of shell scripts, which automate the various modes of analysis possible with MGVB. These are described in details in tutorial section in Appendix 2.

The following external libraries were used to develop MGVB: the arbitrary precision number libraries from GNU, GMP and MPFR (refs); OpenMP (for shared memory parallelism) (ref); OpenMPI (for message-passing parallelism) (ref); sqlite3 (for storing and querying data) (ref). These libraries are statically linked and do not need to be installed on the user's computer.

3.2 Outline of algorithms

The detailed description of the scoring algorithm implemented in the scorer module is not provided as it closely follows the published Andromeda scoring algorithm: the same binomial score and the same approach of filtering the top q spectral peaks as MaxQuant/Andromeda are used.

Where MGVB differs is the algorithm for PTM localisation. Andromeda uses a scoring algorithm that is similar to the one used to match spectra to sequences (ref). A binomial probability derived score is computed for each possible PTM assignment model and the one with the highest score is selected. MGVB departs from this approach and uses a simple Bayesian updating algorithm to assign localisation probabilities.

The process starts with generation of all possible PTM localisation models, which are encoded as binary arrays. These are assigned equal prior probabilities. These priors are then updated using the detected fragments as experimental evidences to obtain the posterior probabilities of localisation.

The Bayesian updating algorithm is described in Algorithm 1. MGVB assumes a likelihood of 0.7 if a fragment is compatible with a localisation model and likelihood of 0.2 if not (the likelihood is the probability that the fragment will be observed given that the model in question is true).

Algorithm 1 Calculate PTM localisation models probabilities

```
models ← computeModels                          ▷ generates N models
p ← {1/N, 1/N ... 1/N}                          ▷ assigns uniform prior
F ← masses of matched fragments in spectrum
for all f in F do
    for all m in models do
        if f, m are compatible then
            p_m ← p_m × λ_1                     ▷ if compatible, lambda is set to 0.7
        else
            p_m ← p_m × λ_0                     ▷ if not compatible it is usually 0.2
        end if
    end for
end for
Renormalise p to sum to 1
```

The Algorithm **??** calculates posterior probabilities for the possible PTM localisation models. Model probabilities p_j are then converted to site probabilities P_i using the following equations:

$$P_i = \sum_{j=1}^{N} p_j \times \gamma_j^i. \tag{1}$$

where,

$$\gamma_j^i = \begin{cases} 1 & \text{if site i is modified under model j} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The combinatorial PTM search algorithm implemented by scorer_mpi is presented in Algorithm **??**. It uses a precompiled database of combinations of up to 3 different PTMs and covers 125 (???) different modifications derived from Unimod.

Algorithm **??** is implemented in select_by_prob (see Table **??**).This module filters peptide hits using decoy database hits. Importantly, it solves the protein

Algorithm 2 Combinatorial PTM search implemented by scorer_mpi

```
mod_comb ← generate mod_comb                          ▷ generates delta mass db
peptides ← generate peptide db                        ▷ generates peptides db
for all s in mms file do                              ▷ mms file contains spectra
    M ← peptides matching parent of s                 ▷ usually ±500 Da
    for all m in M do
        δ ← compute delta mass for m
        ptm ← PTM combinations from mod_comb matching δ
        for all c in ptm do
            score ← score s against m modified by c, save to results
        end for
    end for
end for
```

inference problem: given a set of matched peptides passing the score thresholds, what is the optimal peptides to proteins assignment? This is not a trivial problem as many proteins encoded by distinct genes share sequence similarities, which cause peptides to be shared across groups of proteins. MGVB solves the protein inference problem by implementing a recursive algorithm, which assigns peptides to the protein with the highest protein score in the protein group that share these peptides. Two data structures are involved: a linked list of protein groups, each node containing a list of identities of the proteins in the group, count of spectra matching the group, and a pointer to the next element of the list. In addition, an array of protein data structures is used. This array is sorted by protein score to allow efficient searching.

Algorithm 3 Recursive protein inference from peptide matches

```
pr_groups ← generate linked list of protein_group structs
proteins ← generate sorted array of tuples of protein IDs with scores
function PROCESS_PROT_GROUPS(pr_groups, proteins)
    if pr_groups is empty then
        return
    end if
    names ← split protein IDs in first node of pr_groups into array
    Pr ← the element of names with the highest score in proteins
    delete all pr_groups nodes containing Pr summing their counts to c
    save Pr, c to results
    return PROCESS_PROT_GROUPS(pr_groups, proteins)
end function
PROCESS_PROT_GROUPS(pr_groups, proteins)
```

Algorithm **??** outputs to a file containing spectral counts and protein IDs. MGVB also contains facilities for combining such files into an aggregated

report file of comma separated values, which can be further analysed in R, Python, Excel or any other environment for machine learning and statistical analysis. In addition, MGVB creates sqlite database files for each raw file analysed, which contain plethora of information about scans, peptides, modification etc. These can be analysed by sqlite functions or other software operating on SQL databases. Some simple but useful report functions are given in Appendix 1.

4 Appendix 1

4.1 How-to-do guide for using the MGVB pipeline

4.1.1 Differential expression and protein interaction analysis

MGVB is distrinuted as a single archive file. Download and expand the file in a convenient location on your system. For example: /proteomics. When placed in this directory and expanded, the archive will create a subdirectory tree containing mgvb/bin. To install the executables and run the pipeline do:

1. Put mgvb/bin to the path: PATH=$PATH: [path to mgvb/bin]
2. Copy raw files to a new project directory
3. Edit config.rms. Change only the necessary entries: fasta files and experiment names. If not interested in phosphorylations comment out the entry. Make sure precTol and tol are appropriate for the type of data—High/Low or High/High
4. Execute omp_auto.sh

4.1.2 Focused analysis of modifications

1. Execute steps 1 to 3 above
2. Create config_focused.rms by copying config.rms and changing it: make precTol 500. Comment out all modificatons and all fasta entries. Create a new fasta entry as focused.fasta
3. Execute auto_focused.sh
4. Examine results for a specific protein by: (i) opening the corresponding *.sig_proteins_ccounts.txt file and noting the protein number; (ii) looking up all relevant entries in unnested in the corresponding individual db file and then extracting all entries from sig_scans with matching sequence:

For a protein with number 921: sqlite3 select * from sig_scans where Sequence in (select sequence from unnested where Protein = "921");

NB: Use exp_ prefix for experiment names in config.rms. This solves problems with sqlite table names

4.1.3 Automated focused analysis

1. Make sure MGVB is installed (executables and sh scripts are on the path)

2. Start with a clean directory containing raw files, db1_min.db file, config.rms and config_focused.rms

3. Make sure table proteins in db1_min.db is deleted: execute sqlite3 db1_min.db drop table proteins; and then vacuum

4. Execute auto_focused_mgvb.sh ¡number of processors¿

5. Interrogate results using sqlite functionality

Topical subheadings are allowed. Authors must ensure that their Methods section includes adequate experimental and characterisation data necessary for others in the field to reproduce their work. Authors are encouraged to include RIIDs where appropriate.

Ethical approval declarations (only required where applicable) Any article reporting experiment/s carried out on (i) live vertebrate (or higher invertebrates), (ii) humans or (iii) human samples must include an unambiguous statement within the methods section that meets the following requirements:

The Introduction section, of referenced text [**?**] expands on the background of the work (some overlap with the Abstract is acceptable). The introduction should not include subheadings.

Springer Nature does not impose a strict layout as standard however authors are advised to check the individual requirements for the journal they are planning to submit to as there may be journal-level preferences. When preparing your text please also be aware that some stylistic choices are not supported in full text XML (publication version), including coloured font. These will not be replicated in the typeset article if it is accepted.

5 This is an example for first level head—section head

5.1 This is an example for second level head—subsection head

5.1.1 This is an example for third level head—subsubsection head

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

6 Equations

Equations in LaTeX can either be inline or on-a-line by itself ("display equations"). For inline equations use the `$...$` commands. E.g.: The equation $H\psi = E\psi$ is written via the command `$H \psi = E \psi$`.

For display equations (with auto generated equation numbers) one can use the equation or align environments:

$$\|\tilde{X}(k)\|^2 \leq \frac{\sum\limits_{i=1}^{p}\left\|\tilde{Y}_i(k)\right\|^2 + \sum\limits_{j=1}^{q}\left\|\tilde{Z}_j(k)\right\|^2}{p+q}. \tag{3}$$

where,

$$\begin{aligned} D_\mu &= \partial_\mu - ig\frac{\lambda^a}{2}A^a_\mu \\ F^a_{\mu\nu} &= \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + gf^{abc}A^b_\mu A^a_\nu \end{aligned} \tag{4}$$

Notice the use of `\nonumber` in the align environment at the end of each line, except the last, so as not to produce equation numbers on lines where no equation numbers are required. The `\label{}` command should only be used at the last line of an align environment where `\nonumber` is not used.

$$Y_\infty = \left(\frac{m}{\text{GeV}}\right)^{-3}\left[1 + \frac{3\ln(m/\text{GeV})}{15} + \frac{\ln(c_2/5)}{15}\right] \tag{5}$$

The class file also supports the use of `\mathbb{}`, `\mathscr{}` and `\mathcal{}` commands. As such `\mathbb{R}`, `\mathscr{R}` and `\mathcal{R}` produces $\mathbb{R}$, $\mathscr{R}$ and $\mathcal{R}$ respectively (refer Subsubsection **??**).

7 Tables

Tables can be inserted via the normal table and tabular environment. To put footnotes inside tables you should use `\footnotetext[]{...}` tag. The footnote appears just below the table itself (refer Tables **??** and **??**). For the corresponding footnotemark use `\footnotemark[...]`

Table 2 Caption text

Column 1	Column 2	Column 3	Column 4
row 1	data 1	data 2	data 3
row 2	data 4	data 5[1]	data 6
row 3	data 7	data 8	data 9[2]

Source: This is an example of table footnote. This is an example of table footnote.

[1]Example for a first table footnote. This is an example of table footnote.

[2]Example for a second table footnote. This is an example of table footnote.

The input format for the above table is as follows:

```
\begin{table}[<placement-specifier>]
\begin{center}
\begin{minipage}{<preferred-table-width>}
\caption{<table-caption>}\label{<table-label>}%
\begin{tabular}{@{}llll@{}}
\toprule
Column 1 & Column 2 & Column 3 & Column 4\\
\midrule
row 1 & data 1 & data 2  & data 3 \\
row 2 & data 4 & data 5\footnotemark[1] & data 6 \\
row 3 & data 7 & data 8  & data 9\footnotemark[2]\\
\botrule
\end{tabular}
\footnotetext{Source: This is an example of table footnote.
This is an example of table footnote.}
\footnotetext[1]{Example for a first table footnote.
This is an example of table footnote.}
\footnotetext[2]{Example for a second table footnote.
This is an example of table footnote.}
\end{minipage}
\end{center}
\end{table}
```

Table 3 Example of a lengthy table which is set to full textwidth

	Element 1[1]			Element 2[2]		
Project	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote. This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]Example for a first table footnote.

[2]Example for a second table footnote.

In case of double column layout, tables which do not fit in single column width should be set to full text width. For this, you need to use `\begin{table*} ... \end{table*}` instead of `\begin{table} ... \end{table}` environment. Lengthy tables which do not fit in textwidth should be set as rotated table. For this, you need to use `\begin{sidewaystable} ... \end{sidewaystable}` instead of `\begin{table*} ... \end{table*}` environment. This environment puts tables rotated to single column width. For tables rotated to double column width, use `\begin{sidewaystable*} ... \end{sidewaystable*}`.

Table 4 Tables which are too long to fit, should be written using the "sidewaystable" environment as shown here

	Element 1[1]			Element[2]		
Projectile	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40
Element 5	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 6	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]This is an example of table footnote.

8 Figures

As per the LaTeX standards you need to use eps images for LaTeX compilation and pdf/jpg/png images for PDFLaTeX compilation. This is one of the major difference between LaTeX and PDFLaTeX. Each image should be from a single input .eps/vector image file. Avoid using subfigures. The command for inserting images for LaTeX and PDFLaTeX can be generalized. The package used to insert images in LaTeX/PDFLaTeX is the graphicx package. Figures can be inserted via the normal figure environment as shown in the below example:

```
\begin{figure}[<placement-specifier>]
\centering
\includegraphics{<eps-file>}
\caption{<figure-caption>}\label{<figure-label>}
\end{figure}
```

Fig. 1 This is a widefig. This is an example of long caption this is an example of long caption this is an example of long caption this is an example of long caption

In case of double column layout, the above format puts figure captions/images to single column width. To get spanned images, we need to provide `\begin{figure*} ... \end{figure*}`.

For sample purpose, we have included the width of images in the optional argument of `\includegraphics` tag. Please ignore this.

9 Algorithms, Program codes and Listings

Packages `algorithm`, `algorithmicx` and `algpseudocode` are used for setting algorithms in LaTeX using the format:

```
\begin{algorithm}
\caption{<alg-caption>}\label{<alg-label>}
\begin{algorithmic}[1]
. . .
\end{algorithmic}
\end{algorithm}
```

You may refer above listed package documentations for more details before setting `algorithm` environment. For program codes, the "program"

package is required and the command to be used is `\begin{program} ... \end{program}`. A fast exponentiation procedure:

```
begin
  for i := 1 to 10 step 1 do
      expt(2, i);
      newline() od                Comments will be set flush to the right margin
where
proc expt(x, n) ≡
  z := 1;
  do if n = 0 then exit fi;
      do if odd(n) then exit fi;
          comment: This is a comment statement;
          n := n/2; x := x * x od;
      {n > 0};
      n := n − 1; z := z * x od;
  print(z).
end
```

Algorithm 4 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$

```
1:  y ⇐ 1
2:  if n < 0 then
3:      X ⇐ 1/x
4:      N ⇐ −n
5:  else
6:      X ⇐ x
7:      N ⇐ n
8:  end if
9:  while N ≠ 0 do
10:     if N is even then
11:         X ⇐ X × X
12:         N ⇐ N/2
13:     else[N is odd]
14:         y ⇐ y × X
15:         N ⇐ N − 1
16:     end if
17: end while
```

Similarly, for `listings`, use the `listings` package. `\begin{lstlisting} ... \end{lstlisting}` is used to set environments similar to `verbatim` environment. Refer to the `lstlisting` package documentation for more details.

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
```

10 Cross referencing

Environments such as figure, table, equation and align can have a label declared via the `\label{#label}` command. For figures and table environments use the `\label{}` command inside or just below the `\caption{}` command. You can then use the `\ref{#label}` command to cross-reference them. As an example, consider the label declared for Figure **??** which is `\label{fig1}`. To cross-reference it, use the command `Figure \ref{fig1}`, for which it comes up as "Figure **??**".

To reference line numbers in an algorithm, consider the label declared for the line number 2 of Algorithm **??** is `\label{algln2}`. To cross-reference it, use the command `\ref{algln2}` for which it comes up as line **??** of Algorithm **??**.

10.1 Details on reference citations

Standard LaTeX permits only numerical citations. To support both numerical and author-year citations this template uses `natbib` LaTeX package. For style guidance please refer to the template user manual.

Here is an example for `\cite{...}`: [**?**]. Another example for `\citep{...}`: [**?**]. For author-year citation mode, `\cite{...}` prints Jones et al. (1990) and `\citep{...}` prints (Jones et al., 1990).

All cited bib entries are printed at the end of this article: [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**] and [**?**].

11 Examples for theorem like environments

For theorem like environments, we require `amsthm` package. There are three types of predefined theorem styles exists—`thmstyleone`, `thmstyletwo` and `thmstylethree`

`thmstyleone`	Numbered, theorem head in bold font and theorem text in italic style
`thmstyletwo`	Numbered, theorem head in roman font and theorem text in italic style
`thmstylethree`	Numbered, theorem head in bold font and theorem text in roman style

For mathematics journals, theorem styles can be included as shown in the following examples:

Theorem 1 (Theorem subhead) *Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proposition 2 *Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Example 1 Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Remark 1 Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Definition 1 (Definition sub head) Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text.

Additionally a predefined "proof" environment is available: `\begin{proof} ... \end{proof}`. This prints a "Proof" head in italic font style and the "body text" in roman font style with an open square at the end of each proof environment.

Proof Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proof of Theorem **??** Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

For a quote environment, use `\begin{quote}...\end{quote}`

> Quoted text example. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text (refer Figure **??**). Sample body text. Sample body text. Sample body text (refer Table **??**).

12 Methods

Topical subheadings are allowed. Authors must ensure that their Methods section includes adequate experimental and characterization data necessary for others in the field to reproduce their work. Authors are encouraged to include RIIDs where appropriate.

Ethical approval declarations (only required where applicable) Any article reporting experiment/s carried out on (i) live vertebrate (or higher invertebrates), (ii) humans or (iii) human samples must include an unambiguous statement within the methods section that meets the following requirements:

1. Approval: a statement which confirms that all experimental protocols were approved by a named institutional and/or licensing committee. Please identify the approving body in the methods section
2. Accordance: a statement explicitly saying that the methods were carried out in accordance with the relevant guidelines and regulations

3. Informed consent (for experiments involving humans or human tissue samples): include a statement confirming that informed consent was obtained from all participants and/or their legal guardian/s

If your manuscript includes potentially identifying patient/participant information, or if it describes human transplantation research, or if it reports results of a clinical trial then additional information will be required. Please visit (https://www.nature.com/nature-research/editorial-policies) for Nature Portfolio journals, (https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/publishing-ethics/14214) for Springer Nature journals, or (https://www.biomedcentral.com/getpublished/editorial-policies#ethics+and+consent) for BMC.

13 Discussion

Discussions should be brief and focused. In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Some journals prefer a section 'Results and Discussion' followed by a section 'Conclusion'. Please refer to Journal-level guidance for any specific requirements.

14 Conclusion

Conclusions may be used to restate your hypothesis or research question, restate your major findings, explain the relevance and the added value of your work, highlight any limitations of your study, describe future directions for research and recommendations.

In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Please refer to Journal-level guidance for any specific requirements.

Supplementary information. If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

Acknowledgments. Acknowledgments are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
- Ethics approval
- Consent to participate
- Consent for publication
- Availability of data and materials
- Code availability
- Authors' contributions

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

Editorial Policies for:

Springer journals and proceedings:
https://www.springer.com/gp/editorial-policies

Nature Portfolio journals:
https://www.nature.com/nature-research/editorial-policies

Scientific Reports:
https://www.nature.com/srep/journal-policies/editorial-policies

BMC journals:
https://www.biomedcentral.com/getpublished/editorial-policies

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

[1] Campbell, S.L., Gear, C.W.: The index of general nonlinear DAES. Numer. Math. **72**(2), 173–196 (1995)

[2] Slifka, M.K., Whitton, J.L.: Clinical implications of dysregulated cytokine production. J. Mol. Med. **78**, 74–80 (2000). https://doi.org/10.1007/s001090000086

[3] Hamburger, C.: Quasimonotonicity, regularity and duality for nonlinear systems of partial differential equations. Ann. Mat. Pura. Appl. **169**(2), 321–354 (1995)

[4] Geddes, K.O., Czapor, S.R., Labahn, G.: Algorithms for Computer Algebra. Kluwer, Boston (1992)

[5] Broy, M.: Software engineering—from auxiliary to key technologies. In: Broy, M., Denert, E. (eds.) Software Pioneers, pp. 10–13. Springer, New York (1992)

[6] Seymour, R.S. (ed.): Conductive Polymers. Plenum, New York (1981)

[7] Smith, S.E.: Neuromuscular blocking drugs in man. In: Zaimis, E. (ed.) Neuromuscular Junction. Handbook of Experimental Pharmacology, vol. 42, pp. 593–660. Springer, Heidelberg (1976)

[8] Chung, S.T., Morris, R.L.: Isolation and characterization of plasmid deoxyribonucleic acid from Streptomyces fradiae. Paper presented at the 3rd international symposium on the genetics of industrial microorganisms, University of Wisconsin, Madison, 4–9 June 1978 (1978)

[9] Hao, Z., AghaKouchak, A., Nakhjiri, N., Farahmand, A.: Global integrated drought monitoring and prediction system (GIDMaPS) data sets. figshare https://doi.org/10.6084/m9.figshare.853801 (2014)

[10] Babichev, S.A., Ries, J., Lvovsky, A.I.: Quantum scissors: teleportation of single-mode optical states by means of a nonlocal single photon. Preprint at https://arxiv.org/abs/quant-ph/0208066v1 (2002)

[11] Beneke, M., Buchalla, G., Dunietz, I.: Mixing induced CP asymmetries in inclusive B decays. Phys. Lett. **B393**, 132–142 (1997) https://arxiv.org/abs/0707.3168 [gr-gc]

[12] Stahl, B.: DeepSIP: Deep Learning of Supernova Ia Parameters, 0.42, Astrophysics Source Code Library (2020), https://ascl.net/2006.023