VP „Wissenschaftliches Arbeiten und Präsentation“
WS 2008/09

M. Held und A. Uhl
FB Computerwissenschaften
Universität Salzburg
A–5020 Salzburg, Austria

9. Oktober 2008

UNIV. SALZBURG
COMPUTERWISSENSCHAFTEN

Personalia: M. Held

Email-Adresse: `held@cosy.sbg.ac.at`.

Basis-URL: `http://www.cosy.sbg.ac.at/~held`.

WWW-Homepage: Basis-URL`/held.html`.

Büro: Computerwissenschaften, Zi. 1.20, Jakob-Haringer Str. 2, Salzburg-Itzling.

Telefonnummer (Büro): (0662) 8044-6304.

Telefonnummer (Sekretariat): (0662) 8044-6328.

Personalia: A. Uhl

Email-Adresse: uhl@cosy.sbg.ac.at.

Basis-URL: http://www.scicomp.sbg.ac.at/staff/andreas.uhl.html.

Büro: Computerwissenschaften, Zi. 1.11, Jakob-Haringer Str. 2, Salzburg-Itzling.

Telefonnummer (Büro): (0662) 8044-6303.

Telefonnummer (Sekretariat): (0662) 8044-6328.

Formalia (Gruppe Held)

LVA-URL: Basis-URL`/teaching/wiss_arbeiten/wiss_arbeiten.html`.

Allg. Information: Basis-URL`/for_students.html`.

Abhaltezeit der LVA: Freitag 9^{00}–11^{45}.

Abhalteort der LVA: T01, Computerwissenschaften, Jakob-Haringer Str. 2.

Formalia (Gruppe Uhl)

LVA-URL: `http://www.cosy.sbg.ac.at/˜uhl/student.html`.

Abhaltezeit der LVA: Freitag 9^{15}–12^{00}.

Abhalteort der LVA: T03, Computerwissenschaften, Jakob-Haringer Str. 2.

Preface

Welcome to an introduction to scientific working techniques and to some of the software packages that your CS faculty use for their own scientific work!

We will explain how “science” works, what scientific literature is and how it can be found, we will discuss software packages for

- preparing advanced scientific documents,
- drafting figures, plots, and similar illustrations,
- manipulating mathematical formulae,

and we will talk about theory and practice of giving scientific presentations.

Goals

The overall goal of this lecture is to familiarize you with science and those software packages to such an extent that

- you'll be able to use those packages for accomplishing simple tasks without consulting manuals repeatedly,
- you'll gain an understanding of the pros and cons of those packages, and know which package to use in order to accomplish a specific task,

and that

- you'll learn where to find additional information and help if needed.

It is our sincere hope that this will help you with getting the work needed for your studies done more effectively.

Topics Discussed

In this lecture we will discuss the following topics and corresponding software packages:

Scientific Literature: types, documentation, literature search.

Scientific Presentations: guidelines for oral presentations and written documents.

Scientific Text Processing: LaTeX, PostScript.

Drafting and Generating Plots: Ipe, tgif, xfig, gnuplot, xgraph.

Symbolical Mathematics: Mathematica.

DTP and WWW: PDF, ppower, LaTeX2HTML.

Obviously, time constraints do not allow us to discuss tons of software packages in detail. We have selected those packages according to whether we've personally found them useful for our own scientific work. With one single exemption, all packages discussed are freely available and can be installed on any PC running Linux. (Some of them may also be available for MS-based platforms.)

Electronic Slides and Online Material

In addition to these slides, you are encouraged to consult the WWW home-page of this lecture:

http://www.cosy.sbg.ac.at/~held/teaching/wiss_arbeiten/wiss_arbeiten.html.

In particular, this WWW page will contain links to online manuals.

A Few Words of Warning

We hope that these slides will help you to get acquainted with the software packages discussed. However, we would like to warn you explicitly not to regard these slides as the sole source of information on the topics of my lecture. It may and will happen that we'll use the lecture for talking about subtle details of some package that are not covered in these slides! In particular, by making these slides available to you I do not intend to encourage you to attend the lecture on an irregular basis.

Also, we hope that you will realize that most software packages dealt with in this lecture will only be fully appreciated after personally using them. It will be considerably more difficult to learn, say, LaTeX if you are not prepared and willing to get your hands on a computer and try it out personally.

Acknowledgments

We are happy to acknowledge that we benefited from material published by colleagues on diverse topics presented in this lecture. In particular, several sample codes (for LaTeX figures, PostScript, etc.) are borrowed from other publications. Similarly, some descriptions of software packages were copied from their respective user manuals. While some of the material used for this lecture was originally presented in traditional-style publications (such as textbooks), some other material has its roots in non-standard publication outlets (such as online documentations).

Salzburg, September 2008 Martin Held & Andreas Uhl

Table of Contents

Table of Contents: Scientific Literature, Oral and Written Presentations

Table of Contents: Scientific Text Processing

Table of Contents: Figures and Plots

Table of Contents: DTP and WWW

Table of Contents: Symbolical Mathematics

Table of Contents: Graphics and Visualization

Scientific Literature, Oral and Written Presentations

- What is Scientific Literature?
- Types of Scientific Literature,
- Bibliographic Data,
- Searching, Finding and Retrieving Literature,
- Oral Presentations,
- Written Presentations,
- Austrian Academic Careers.

What is Scientific Literature?

For many problems a lot of different solutions are already known and stored in the "literature". When working on the solution of a problem in computer science one should, of course, consult the literature in order to avoid spending time on developing solution approaches that are already known. Checking the literature and thinking about a solution are *alternating steps* that depend on each other and should be iterated:

- own attempts to solve a problem will clarify where a solution might be found in the literature,
- looking to the literature will provide new ideas on how to solve the problem.

For efficient use of the literature it is necessary to know *how literature is organized*.

What is Scientific Literature?

When a computer scientist finds a new solution to a problem he/she must know how “to store this solution in the literature” (“publish”) so that other people can make use of it. Technically perfect documentation of one’s findings by writing papers is important since results that are not well described will not be re-used but people will prefer to reinvent them.

In fact “literature” in computer science more and more also comprises “active” knowledge and methods in the form of

- software systems and applets,
- algorithm libraries,
- knowledge bases,
- WWW pages.

In the following, we will concentrate on classical written documents when using the term “literature”.

Types of Scientific Literature

- monographs (books),
- articles in journals,
- articles in collections,
- written versions of conference talks (proceedings papers),
- technical reports of research organizations,
- theses,
- patent descriptions.

Characteristics of Scientific Literature

These basic types of scientific literature have different characteristics with respect to

- authorship,
- contents,
- originality,
- production, organization,
- quality control.

Types of Scientific Literature: Monographs

- Authorship: a book has one or several authors who write the book. Subsequently, a book company has to be found for processing the book. In practice, usually a publisher is sought before the book is written. Often, scientists of high reputation are approached by book companies or editors of book series to write a book on a certain subject.

- Content: A monograph treats a specific area of computer science in a systematic and complete form. The area may be a traditional area seen under a new or specific perspective or a new area whose results were so far scattered in various other sources like journal articles and technical reports. The content is also determined by the level of background the author presupposes on the side of the reader (e.g. undergraduate texts, graduate texts, research monographs).

- Originality: Mostly, the results contained in a monograph are *not* new but were already published earlier in journal articles etc. However, explaining everything in one uniform context or filling gaps in a systematic treatment etc. may be quite a creative process although it is not considered to be original research in computer science.

Types of Scientific Literature: Monographs (cont'd)

- Production, organization: The author writes the book and the publishing company publishes the book and also owns the "copyright". The company pays "royalties" to the authors for each copy sold, about 5 - 10% of the retail price. Usually, the author receives complimentary copies or the right to obtain copies at a reduced price. A certain number of copies of the book (an "edition") is produced in one process and put on stock. Before a new edition is printed, authors are invited to update, improve and possibly extend the book. The number of copies of one edition may range from a few hundreds to several thousand copies.

- Quality control: At good publishing companies, the scientific quality of monographs is checked by an "editor" who normally is a renowned expert in the field of the book. Often, one or several editors are in charge of a book series in a specific area. Typically, additional scientists – so-called "referees" – are asked to assess the quality of some or all of the book's chapters. Furthermore, publishers employ a "copy editor" or "desk editor", who is in charge of correcting grammar and style. After the publication of the book on the market critical reviews about the book may be published in "review journals": short summary, assessment of value and criticism of the book.

Types of Scientific Literature: Journal Articles

- Authorship: Every scientist may be an author of an article in a journal. In fact, every scientist should strive for publishing his/her results in journals since this the type of scientific publication with the highest quality standard. Acceptance of articles in journals is essentially guaranteed if the paper is in the scope of the journal and the quality of the paper meets the scientific standard of the journal. In contrast to books, acceptance of articles in journals is not at all driven by economic considerations. In case of more than one author the authors may be ordered alphabetically or ordered in accordance to the importance of their contribution to the paper.

- Contents and Originality: Journal articles contain *new* results within the scope of the journal. (Exceptionally, journals also publish survey articles on emerging and topical fields. Usually, such articles are "by invitation", i.e., top scientists in the respective field are asked to submit a survey). Journal articles are directed towards the relatively small group of expert readers that work in the field covered by the journal. There are approximately 700 "refereed" journals in the area of mathematics and computer science.

Types of Scientific Literature: Journal Articles (cont'd)

- Production, Organization: Like books, journals are published by publishing companies. The author prepares a manuscript and sends ("submits") it to the editor (or to one of the editors) of the journal. Sometimes the "editorial board" of a journal may be quite big – ten to fifteen people – in order to represent the scope of the journal well. (The impulse to start a new journal is a joint effort of a group of scientists who want to open a publication forum for their field of expertise and of a publishing company which sees a niche in the market.)

  If the editor accepts a manuscript after the refereeing process (see below) then it is sent to the publishing company for printing. Also, the editor may suggest an "issue" into which the paper should go. The issues of a journal appear on a regular basis, for example quarterly, bimonthly or monthly. Typically, an issue has 50–150 pages and contains several articles ("papers"). Several issues are combined in a volume; usually, a volume comprises the issues that appear in one calendar year.

  The publisher owns the copyright for the articles and no royalties are paid to the authors. (Sometimes authors are even asked to share the printing costs.) Journals are sold to "subscribers" (such as scientific libraries). Often, publishers grant reduced rates for individual subscriptions.

Types of Scientific Literature: Journal Articles (cont'd)

- Quality control: The scientific quality of journal papers is checked by the following "refereeing procedure", which is fairly standardized for international journals in mathematics and computer science.

1. An author submits a manuscript to the editor of the journal.
2. The editor asks two or more anonymous "referees" – i.e., blind refereeing – to give a detailed assessment of the quality of the paper. The referees may be members of the editorial board but normally many more scientists outside of the editorial board are involved. There are several criteria that must be met by a paper that is "accepted" for publication (see below). "Double-blind refereeing" is a special kind of refereeing where also the identity of the authors is concealed – the goal is to provide a higher level of objectivity.
3. Often, a paper has to undergo several "revisions" before it is accepted for publication. The editor in charge for a paper supervises this process which usually involves communication between the anonymous referees and the author.
4. If the paper is finally accepted it is sent to the publisher. Otherwise, it is rejected.

Types of Scientific Literature: Journal Articles (cont'd)

The following items should be assessed by a referee report:

- Whether the paper is in the scope of the journal,
- Interest to the readers of the journal,
- Originality,
- Level of detail,
- Technical correctness and content,
- Language and clarity of presentation,
- Structural organization.

Also, it is common that the referee has to judge his own level of competence in refereeing the paper (e.g. specialist, familiar with the field, ...).

Sample Referee's Form ("Computer-Aided Design")

```
Referee's comments on a manuscript for CAD journal.



Please mark the boxes which best describe your view of the paper.
-----------------------------------------------------------------------

1. ORIGINALITY
[ ] Never been done before.
[ ] Never been done this way before.
[ ] Minor variation on a known technique.  (Can you cite a reference?)
[ ] Re-invention of a known technique.  (Can you cite a reference?)
-----------------------------------------------------------------------
2. SIGNIFICANCE
[ ] Important problem [ ] of current interest.
[ ] Part of a problem [ ] of current interest.
[ ] An interesting insight.
[ ] Recreational.
-----------------------------------------------------------------------
```

Sample Referee’s Form (cont’d)

```
3. SOUNDNESS
[ ] Obviously sound.
[ ] Probably sound.
[ ] Contains errors of detail.  (What sort of errors?)
[ ] Seriously flawed.  (Where are the flaws?)
------------------------------------------------------------------
4. DETAIL
[ ] Unnecessarily detailed.  (Which parts could be shortened?)
[ ] Enough for a graduate student to use the results.
[ ] Enough for the referee to use the results.
[ ] No-one could use the results.  (What's missing?)
------------------------------------------------------------------
5. REFERENCES
[ ] Too many background references of marginal value.
[ ] Virtually the same references the referee would have cited.
[ ] Out-of-date references: to old work only.
[ ] Shallow references: to new work only.
[ ] Totally inadequate references.  (What should be cited?)
------------------------------------------------------------------
```

Sample Referee’s Form (cont’d)

```
6. COMPREHENSIBILITY
[ ] Understood at first reading.
[ ] Several readings required.
[ ] It would take a week to understand this paper.
----------------------------------------------------------------
7. PRESENTATION
[ ] Paper is too long.  (What could be omitted?)

[ ] Paper is well-balanced.
[ ] Paper is too short.  (What’s missing?)
[ ] Rearrangement needed.  (How should the paper be arranged?)

[ ] Title not descriptive.  (Can you suggest a better title?)
[ ] Abstract not descriptive.  (What’s wrong with it?)
[ ] Poor figures.  (What’s wrong with them?)
----------------------------------------------------------------
```

Sample Referee’s Form (cont’d)

```
8. RECOMMENDATION
[ ] Accept as is.
[ ] Accept after minor revision.
[ ] Major revision and further refereeing.  (What changes are
    essential?)
     [ ] I am prepared to look at a revised version.

[ ] Reject.  (What is the main reason for this recommendation?)
-------------------------------------------------------------------

Please add any comments intended for the authors, which would explain
the problems with the manuscript and/or help them to improve it.
```

Types of Scientific Literature: Journal Articles (cont'd)

- Of course, the refereeing procedure takes time. Also, the printing process may be time consuming since many journals have "backlogs", i.e. there is a queue of accepted papers awaiting appearance in one of the next issues of the journals. Consequently, the time period between submission and appearance of journal articles may well be two years or longer. This is an obvious disadvantage especially in a rapidly evolving field like computer science.

- The refereeing process involves the subjective opinion of individuals, and hence, of course, it cannot be completely objective. Therefore, one might have "bad luck" with incompetent or uninterested referees. On the other hand, a carefully written referee report can improve the quality of a manuscript significantly. If a referee is not an expert in the field of the paper which (s)he was asked to referee, (s)he should decline! However, every scientist should feel obliged to invest serious effort into the refereeing process since the entire community relies on this technique.

Types of Scientific Literature: Articles in Collections

This is very similar to special issues of journals devoted to a specific topic.

- Authorship, Contents, Originality, Quality control: Similar to articles in journals.
- Production, Organization: A collection of articles is a single, independent event. A group of scientists in cooperation with a publishing company might want to publish independent articles in a topical field. Typically, an editor is asked to organize the volume, i.e., “solicit” papers from authors and write a “call for papers (CFP)” so that everybody who thinks he might make a valuable contribution to the volume can submit a paper. Furthermore, the editor organizes the refereeing process, guides authors in the revisions, and finally makes a decision about which papers to accept and which to reject.

Types of Scientific Literature: Conference Papers

- Authorship, originality: similar to journal articles or articles in collections
- Contents, production, organization: conference papers differ from journal articles in various respects that have to do with the specific way conferences are organized. A conference is organized for the purpose of quick exchange of new results in a particular area of computer science. Here is a typical procedure for organizing a conference and the written "proceedings" of a conference.
  - ⋆ Typically, a scientific organization – e.g., a scientific society such as IEEE or ACM, or a research institution – decides to organize a conference and determines scope, date, and place of the conference. They install a conference chairman, a program committee and an organization committee.
  - ⋆ The chairman presides and coordinates all people involved in the conference. In particular, (s)he is in charge of making the conference known in the scientific community and for getting sufficient support.
  - ⋆ The organizing committee is responsible for the local arrangements and organize all technical matters (lecture halls, equipment, lodging,....).

Types of Scientific Literature: Conference Papers (cont'd)

- Contents, production, organization (cont'd):
  - ⋆ The program committee (PC) consists of a couple of leading scientists in the field who are in charge of the scientific quality of the conference. The PC is led by the program chairman.
  - ⋆ The PC writes a CFP (i.e., sends an invitation to all scientists working in the field soliciting papers) and determines a "deadline" for submission. Also the PC will negotiate with a publishing company for eventual publication of the proceedings.
  - ⋆ Often, in addition, well known leading scientists are "invited" to present a talk at the conference on particularly important subjects (invited speakers, plenary talks).
  - ⋆ For all papers submitted the PC organizes the refereeing procedure. Of course, refereeing must take place within the deadline defined by the PC.
  - ⋆ After this deadline, in a session of the PC a decision is made about which papers are accepted and which papers are rejected.

Types of Scientific Literature: Conference Papers (cont'd)

- Contents, production, organization (cont'd):
  - ⋆ Accepted papers should be revised by the authors according to the suggestions of the referees (but this is not controlled!) and submitted by a specified deadline in their final form for publication.
  - ⋆ The revised and invited papers are included in the conference proceedings, which normally should be available at the time of the conference but sometimes are published somewhat later.
  - ⋆ The production, marketing and publication of the proceedings is organized by the publishing company. The PC chairman usually is also the editor of the proceedings.
  - ⋆ When the list of accepted papers has been fixed, an announcement of this conference is distributed to as many people as possible with an invitation to attend the conference. The authors of accepted papers will present their papers at the conference in the form of a "talk" or a "poster" with a possibility of discussion during or after the presentation. Conferences are organized in sessions which have dedicated chairpersons. Normally, more people take part in the conference than just the people who present a talk.

Types of Scientific Literature: Conference Papers (cont'd)

- Contents, production, organization (cont'd): Some conferences are organized on a regular basis at changing locations and changing PCs.

- Quality control: Obviously, quality control for conference papers cannot be as perfect as for journal papers due to the strict time schedule. Conference papers have the advantage of speedy publication and no backlog. Quality differs very much among the various conferences because the refereeing procedure may be quite different. As a rule of thumb, conferences organized in the frame of a large professional society like IEEE or ACM usually offer an excellent quality.

  Unfortunately, the main purpose of a few dubious conferences seems to be to make the organizers and some travel companies rich. Typically, such events are organized at spots of high touristic appeal, and fairly high registration fees and/or room rates for accommodation are charged.

Types of Scientific Literature: Technical Reports

The time elapsing between submitting a paper to a journal or a conference may be too long. Also, new results often do not yet satisfy all quality criteria for journals or conferences. Therefore, many scientific institutions publish their own “technical report series”.

- Authorship: Members or visitors of the scientific institution.
- Contents: New results in the special area of the author or preliminary versions of lecture notes.
- Originality: Often, highly original and topical material.
- Quality control: None. Often, revised versions of good technical reports are submitted to journals (“preprints”).
- Production, organization: Normally, technical reports are published in a series but irregularly. No publishing company is involved, reports are published by the scientific institution and often distributed over a web-page.

Types of Scientific Literature: Theses

- Master's thesis: Demonstrates the author's ability to work with scientific literature and scientific tools in general. It usually gives an overview of a field in computer science and the discussion and solution of/to a specific problem. It is of course desirable to have original results in a Master's thesis but this is not mandatory by law. The quality is controlled by the thesis advisor. A Master's thesis is normally not published but may contain parts that have been published by the author elsewhere.

- Dissertation/PhD thesis: Demonstrates the author's ability to achieve original scientific results. It is mandatory to have original results in a PhD thesis and parts of it should definitely be published in modified form. The quality is controlled by the thesis advisor and a second referee. (Quality control may differ among different schools.)

- Habilitation thesis: Demonstrates that the author is an established researcher in his field of expertise. A Habilitation thesis is either a collection of already published journal and/or conference papers ("cumulative thesis") or a monograph. The quality is controlled by several referees out of which at least two have to be members of the Habilitation committee. (Again, quality control may vary.) Habilitation is only known in Middle Europe and, to some extent, Eastern Europe; it corresponds to achieving tenure (at the level of associate professor) at US universities.

Bibliographic Data

The bibliographic data of a publication are the specification necessary

- for the unique identification of the publication, and
- for being able to find the publication in libraries or to order it from publishing companies, research institutions, remote libraries etc.

From this definition and from the descriptions of the various types of publications on the previous slides the information items required to provide a complete bibliographic identification of a publication are easily inferred.

International Standard Book Number

- The International Standard Book Number (ISBN) used to be a 10-digit number ("ISBN-10") that uniquely identifies books and book-like products world-wide. (The ISBN system has been adopted by more than 170 countries.)
- The ISBN scheme is based on the Standard Book Numbering (SBN) system introduced in Great Britain in 1968.
- Approved as ISO standard 2108 in 1970; International coordination by "Staatsbibliothek zu Berlin".
- The purpose of an ISBN is to uniquely identify one book or edition of a book from one specific publisher.
- It is specific for this particular title, and needs to be changed if the title undergoes a major revision or extension.
- When printed, the ISBN is always preceded by the Latin letters "ISBN".
- Once assigned, an ISBN can never be reused to denote a different book.

International Standard Book Number (cont’d)

- Since 01-Jan-2007, thirteen-digit ISBNs are in use.

- The shift from ISBN-10 to ISBN-13 was motivated by two main reasons:

  1. To expand the numbering capacity of the ISBN system and remedy numbering shortages in some areas of the world;
  2. To align the ISBN scheme with the global EAN.UCC identification system.

International Standard Book Number (cont'd)

- A new ISBN-13 consists of the following five elements:
  - ⋆ Prefix element: Three-digit number made available by EAN International. Currently, "978" is used as prefix.
  - ⋆ Registration group element: It identifies the country, geographical region, or language area. (E.g., "3" stands for "German".)
  - ⋆ Registrant element: It identifies a particular publisher within a registration group.
  - ⋆ Publication element: It identifies a specific (edition of a) publication.
  - ⋆ Check digit: It is calculated as follows (quoted from the ISBN User's Manual):

    > Each of the first 12 digits of the ISBN is alternately multiplied by 1 and 3. The check digit is equal to 10 minus the remainder resulting from dividing the sum of the weighted products of the first 12 digits by 10 with one exception. If this calculation results in an apparent check digit of 10, the check digit is 0.

    E.g.: 978-0-11-000222 is assigned the check digit 4, since $9+21+8+0+1+3+0+0+0+6+2+6=56$, and $(10-(56 \bmod 10)) \bmod 10 = 4$.

- These five elements are separated by hyphens or spaces when displayed in human-readable form. Note that the middle three elements are of variable length.

Digital Object Identifiers

- The main problem of using URLs for specifying electronic documents is their lack of persistency: Since the location rather than the actual document is specified, access to a document is lost once the location of the document changes.
- A Digital Object Identifier (DOI) provides a unique and persistent name for electronic documents.
- The document associated with a given DOI can be located by resorting to a DOI resolver, or by appending the DOI to the URL of the DOI System Proxy Server `http://dx.doi.org/`.
- DOI names can be used to identify free material as well as objects of commercial value.
- Standard publishers that have already established online publishing programs are among the main users of the DOI system.

Bibliographic Data: Monographs

- family name, first name (initials) of the author(s),
- title,
- number of edition,
- (number of pages,)
- name of publishing company, (location of publishing company,)
- year of publication,
- ISBN,
- name of series, number of book within series (e.g. LNCS),
- family name and first name (initials) of the editor(s).

Bibliographic Data: Journal Articles

- family name, first name (initials) of the author(s),
- title,
- name of journal,
- volume and number,
- year,
- first page and last page of the article,
- (name of publishing company, location of company).

Bibliographic Data: Articles in Collections

- family name, first name (initials) of the author(s),
- title,
- title of collection,
- family name and first name (initials) of the editor(s),
- name of publishing company, (location of publishing company,)
- year of publication,
- ISBN,
- first page and last page of the article.

Bibliographic Data: Conference Papers

- family name, first name (initials) of the author(s),
- title,
- title of proceedings,
- (name of conference, location of conference, date of conference),
- name of publishing company, (location of publishing company,)
- family name and first name (initials) of the editor(s),
- year of publication,
- first page and last page of the paper.

Bibliographic Data: Technical Reports

- family name, first name (initials) of the author(s),
- title,
- title of technical report series,
- number of the technical report,
- name and address of the institution publishing the series,
- year of publication.

Bibliographic Data: Theses

- family name, first name (initials) of the author,
- title,
- name and address of research institution,
- type of thesis,
- year of publication.

Searching, Finding and Retrieving Literature

- WWW:
  - ⋆ Standard search engines (e.g., `http://www.google.at/`);
  - ⋆ Science-specific search engines.
- Searchable WWW libraries published by scientific organizations and publishers, e.g., ACM's Digital Library, Elsevier, Springer-Verlag, World Scientific Publishing.
- Libraries (e.g., ALEPH in Austria),
- Review journals,
- Science Citation Index,
- Bibliographies,
- Online compilations, e.g., The Directory of Computing Science Journals, or The Computing Research Repository (CoRR).
- Ordering at publishing companies, Subito, inter-library loan ("Fernleihe").

Science-Specific Search Engines

- Scirus at `http://www.scirus.com`.
- Google Scholar at `http://scholar.google.com/`.
- CiteSeer by NEC and PSU at `http://citeseer.ist.psu.edu/cs`.
- BibFinder by ASU at `http://kilimanjaro.eas.asu.edu/`.
- Scopus at `http://www.scopus.com`.

Typically, a bibliographic search can be extended both into the past and into the future relative to a publication known.

Review Journals and Bibliographies

Review journals are journals that systematically collect the publications appearing in a particular subject area and publish short summaries (“reviews”) of each publication. The summaries are ordered according to some keyword index. Also, various further indices help to find publications in one’s field of interest. A review also contains a critical evaluation of the results presented. Unfortunately, it may take several years until a new result finds its way into a review journal. Sample review journals:

- Mathematical Reviews,
- Zentralblatt für Mathematik,
- SIAM Reviews,
- ACM Computing Reviews,
- ACM Guide to Computing Literature.

A bibliography on a particular area typically consists of an author index, a keyword (subject index), and a survey on the area together with literature references.

How to Obtain a Paper

1. Try to find the author on the WWW and check his/her home page.

2. Send an email to the author and ask for an electronic copy of the paper. (You may also want to ask for any more recent paper on the same subject!)

3. Try to obtain an electronic copy of the paper via the electronic access "Elektronische Zeitschriften" of UBS Salzburg.

4. Send a letter to the author by conventional mail and ask for a "reprint" or a "preprint".

5. Ask your advisors, colleagues, and friends.

6. Shop around libraries.

7. Go for inter-library loan.

Science Citation Index

The Science Citation Index was developed by ISI Thompson (Institute for Scientific Information). By means of this research index it is possible to search for relevant literature starting from an available literature item of year y “into the future”, i.e., in the years $y+1,\dots,$ present year. This is possible since in the database all journal articles and refereed conference proceedings are analyzed with respect to their list of references.

The SCI may be used to

- search for literature, and
- rank journals according to their “quality” (Impact Factor).

This database can be accessed at UBS at Salzburg (CD-ROM und Onlinedatenbanken/Alphabetische Liste/S); two free services for Mathematics may be found at

- `http://www.emis.de/ZMATH/` (Zentralblatt Math.),
- `http://www.ams.org/mathscinet/search` (AMS).

Science Citation Index: Impact Factor

The Journal Impact Factor (JIF) is a measure of the average rate a journal is cited in the scientific literature and is published in the “Journal Citation Report”. Assume that we want to compute the JIF of a journal for 2001:

- S := number of papers and reviews published in the journal of interest in the years 1999 and 2000.
- R := number of citations in the year 2001 which refer to the journal of interest in the years 1999 and 2000.
- Then the JIF for this journal for 2001 is given by R/S.

The JIF is used

- to assess the quality of journals,
- to assess the qualification of scientists,
- as a decision criterion in which journal to publish, for granting funds, ...

Science Citation Index: Problems With the Impact Factor

- Not all publication outlets – and not even all journals – are covered; publishers have to pay to ISI to have their journals included!
- The differences among different articles in journals are hidden.
- The JIF depends on the subject and scientific field: High JIFs are found in Medicine and Biology, relatively low JIFs prevail in CS.
- New branches of science have severe disadvantages.

H-Index

- Suggested in 2005 by J.E. Hirsch to measure the productivity and scientific impact of scientists.

- According to Hirsch,

  a scholar has index h if h of his n papers have at least h citations each, and the other $n-h$ papers have at most h citations each.

- Thus, the h-index shall be higher for truly influential scientists as compared to those who simply feed the paper mill.

- Problems:
  - ⋆ The h-index tends to measure the life-time achievement of scholars. That is, it tends to increase with age!
  - ⋆ Scientists with short careers or few publications are strongly disadvantaged. (E.g., the h-indices of Kurt Gödel or Evariste Galois are disastrously low!)
  - ⋆ The computation of the h-indices requires accurate citation databases.
  - ⋆ Less than trivial to compute automatically for authors whose names are common.
  - ⋆ And, in any case, one can only compare scientists within the same discipline!

Oral Presentations

Before planning an oral presentation the type of audience needs to be assessed and the general setting of the presentation has to be clarified:

- Audience: scientists from your own discipline, scientists from related disciplines, students, politicians, ...
- Available time-slot: typical time for a conference presentation is 20–25 minutes plus 5 minutes of discussion.
- Availability of technical aids: video-beamer, overhead-projector, personal computer, Internet connection, flip-chart, blackboard, slide-projector, VCR and TV-set, ...
- Light conditions and structure of the lecture hall.

Oral Presentations: What is the Goal?

The first step in planning an oral presentation is to identify the goal you want to pursue with this presentation. In this context it is not sufficient that one is aware of the subjects which should be covered by the talk. The main question here is as follows:

- Which activity should the audience be able to perform after listening to your presentation?

This approach delivers automatically an intrinsic motivation for listening to your presentation.

Oral Presentations: Main Guidelines

- Scientific content: do only give an oral presentation if you have something interesting to say.
- Structure of the content: even the most brilliant scientific result cannot be communicated without structuring the information.
- Presentation of the content (mostly visual media): presentation media should help to transport the content of your talk but must not replace the content.
- Speaker: keep in mind that also a purely academic talk is a communication and consequently heavily influenced by emotions!

Oral Presentations: Structure

- Welcome (who am I, where do I come from, . . .)
- Introduction (what am I going to talk about);
- Structure/Contents (what is the outline of the talk, time plan);
- Main part:
  - ⋆ Problem statement;
  - ⋆ Problem solution;
  - ⋆ Correctness, implementation, experiments;
- Conclusion;
- Acknowledgments;

Oral Presentations: Visual Aids

It is very important to know that presentations which use pictorial information **a**nd textual information are significantly more efficient as compared to presentations which use only textual information.

- Use charts and/or graphs instead of tables.
- Graphics in presentations should be simple and clear.
- Be careful when using scanned images! (And make sure to include references when using somebody else's material.)
- Use color.
- Animated graphics are nice but they may distract the attention of your audience from the content of your talk.
- A graphic should not be considered to be self-explaining: your contribution is important!

Oral Presentations: Using Graphics

- Announce the graphics;
- Display the graphics;
- Explain the elements of the graphics:
  - ⋆ Touch;
  - ⋆ Turn;
  - ⋆ Talk;
- Interpretation and Conclusion.
- Note: Whatever can be seen has to be readable and understandable for the audience!

Oral Presentations: Mortal Sins When Producing Slides

- Too much information on one slide,
- Font size is too small,
- Lines are too thin,
- Only copied from text,
- No graphics,
- No color,
- Too many slides – more than one slide/minute is definitely too much.

Oral Presentations: Mortal Sins of the Speaker

- Speaks towards the wall, back to the audience.
- Covers the projection with her/his body.
- Shows too many slides or changes slides too fast.
- Points towards the projection without making clear what exactly is to pointed out.
- Talks towards the projector instead of towards the audience.
- Does not find a specific slide for a while.

Oral Presentations: Mortal Sins of the Speaker (cont'd)

- Does not look at the audience – eye-contact is important!
- Speaks with low, monotonous voice.
- Walks around without any purpose.
- Stands at the same position during the entire talk.
- Hands are moving without connection to the content of the talk.
- Speaks too fast and without pauses.
- Uses long, complicated sentence constructions: Do not read but speak without notes!!

Oral Presentations: Pros and Cons of Different Visual Media

- Transparencies:
  Pro – projectors are widely available, spontaneous;
  Con – old fashioned, boring if badly prepared.

- Flip-chart/Blackboard:
  Pro – interaction with audience, spontaneous;
  Con – preparation is hardly possible, limited graphics facility. (Do not underestimate the difficulty of drawing neat figures on the blackboard!)

- Video-beamer/PC:
  Pro – perfect preparation, perfect graphics facilities;
  Con – technical equipment may fail, presentation may give a “sterile” feeling, often presentations tend to be overloaded, careful a-priori planning of the schedule is needed.

Poster Presentations

A poster presentation is normally given during a poster session at a conference. During a fixed time frame authors are present at their posters and they give short presentations and explanations of their work, typically for less than five minutes. Since the atmosphere is more informal as compared to a talk, more lively interaction between the author and the audience may be expected.

- Pro: More interactive, more spontaneous, real discussions.
- Con: Often, poster sessions are abused to accommodate some low-quality contributions submitted to a conference. In this case, poster sessions tend to take place as a side event during coffee breaks.

It is important to show the main ideas on the poster only! If the poster is written using small fonts and very detailed graphics, nobody will take the time to study it thoroughly and the interest of the audience will be directed to the poster of your colleague just beside yourself. There is a hard competition at poster sessions to attract the attention of the potential audience.

Written Presentations

The most important issue with respect to written presentations is to produce a *well-structured* manuscript. (This is achieved by pursuing a top-down approach.) The reader should be able to find as soon as possible the parts of the manuscript which are of interest to her/him.

The classical sections of a scientific manuscript are:

- Title Block,
- Abstract and Keywords,
- Text,
- Bibliography.

Written Presentations: Title Block

- Title,
- Author(s),
- Address(es) of author(s), i.e., street address, e-mail address, URL.

The formulation of the title is a crucial issue: On the one hand, the reader should be able to decide on the basis of the title whether this manuscript is of any interest to her/him, on the other hand the title should be short and concise. Also, the title should neither be too general and nor too specific. In any case, it should be clear from the title whether this manuscript is, e.g., an experimental study or a theoretical contribution (or both).

Written Presentations: Abstract and Keywords

The abstract is a short description of the manuscript which should characterize the content of the paper as good as possible without the necessity to read the paper itself. Therefore, no references should be made into the manuscript and no citations to other literature must be made.

Keywords are used for indexing the manuscript and therefore facilitate efficient search for the paper. Hence, the keywords should reflect the content of the paper as closely as possible and should be neither too specific nor too general. (However, simply repeating the words of the title makes no sense!)

Written Presentations: Text

- Introduction,
- Exact formulation of the problem,
- Exact formulation of the solution,
- Correctness considerations (if applicable),
- Implementation (if applicable),
- Experimental results (if any),
- Summary, Conclusion,
- Acknowledgments.

Written Presentations: Introduction

Similar to the abstract, the introduction is a short description of the manuscript which contains a description of the problem and the corresponding solution. However, there are significant differences:

- The introduction is longer.
- Problem and solution should be described in a simple and understandable way; the introduction may occupy a significant amount of space.
- It is important to state what has been already known and what is really new in this manuscript. Usually, references to existing related literature are given in the introduction.
- The introduction explains the structure of the manuscript and therefore refers to the following parts of the paper.

Distinguishing between original parts of the manuscript and already existing results is important for two reasons: “intellectual and scientific honesty” and “intellectual property protection”!

Written Presentations: Formulation of the Problem and the Solution

The structure described below is somewhat idealistic and is not followed by all scientific papers. Often, the “Black box” is omitted.

- *Black box*: Parts of the manuscript for the “user”. Here, the problem and its solution are exactly described and sample applications are given. This is to give the user a timing advantage: (s)he should be able to take advantage of the content of the paper without necessarily going into all details that justify the solution of the problem.
- *White box*: Parts of the manuscript for the scientist working in the same field. Here, information is given on the basic ideas which lead to the solution and details about the solution approach (e.g., complexity, correctness), and possible corresponding implementations.

Written Presentations: Conclusion

Similar to the abstract, the conclusion is a short description of the manuscript. However, there are important additional features:

- We may suppose that the rest of the paper has been read.
- The conclusion refers to the results in the manuscript and thereby may “start the discussion”.
- Open problems are stated.
- Possible future work in the area of the manuscript is described.
- Specific details and cross connections to other work are emphasized.

Written Presentations: Style

- The Chicago Manual of Style contains detailed rules on how to write scientific papers that are widely followed by the publishing industry.
- Note that the term “style” refers to grammar, interpunctuation, italicizing, citing, and other related topics rather than to prose style.
- Although targeted at American English, the recommendations contained in The Chicago Manual of Style are widely used when writing scientific papers, no matter which language is used.

Written Presentations: Style (cont'd)

In technical manuscripts we face two situations which need to be treated differently with respect to style: parts which cover known facts and already published literature and parts which cover original own research.

- Personal pronoun: Try to avoid the use of "I/we" in the abstract. I/we helps to emphasize own original work. (However, don't overuse it.)
- Never use "I" or "we" when referring to somebody else's work! Rather, use passive voice or "one" constructs. Complicated constructions in passive voice should be avoided, though!
- Tense: Authors usually write about their own original work in the simple past tense. Other people's work is variously reported. Many authors use the present tense (which is better), some use the compound past tense.

Written Presentations: Style (cont'd)

- Watch the interplay of mathematical terms ("symbols") and normal text:
  - ⋆ Do not start a sentence with a symbol. That is, write
    "The point P is contained in ..."
    rather than
    "P is contained in ..."
    even if it is well-known that P denotes a point.
  - ⋆ Try to separate prose and symbols if the same font is used. E.g., when using LaTeX do not put italicized words and math symbols in consecutive order.
  - ⋆ Make sure to separate symbols that do not belong to the same mathematical term by more than only an interpunctuation character. That is, write
    "Since $p \in P$ we conclude that $q \notin A$ and ..."
    rather than
    "Since $p \in P$, $q \notin A$ and ..."
- Always use the same fonts for symbols in figures and in the running text!
- Never use a font smaller than the standard font size for symbols in figures.

Written Presentations: Grammar and Orthography

As a meta rule, keep in mind that a technical or scientific presentation is nothing but a standard write-up on a specific topic. Thus, technical presentations have to follow the same rules with respect to grammar, orthography and interpunctuation as any other prose! Itemized lists, figures grouped within the text, or mathematical formulae have to be treated like standard words or groups of words. In particular,

- Check for missing commas and periods;
- Check your spelling;
- When using English, decide whether to use British or American English and stick to your decision;
- When using English words within German, decide on the interplay of German and English words, and stick to your decision;
- Watch for missing left or right parentheses in parenthetical remarks;
- Do not use colloquial abbreviations like “he’d” or “it’s”. In any case, note the difference between “it’s” and “its”!

Austrian Academic Careers

The situation changed significantly when a new University Legislation was introduced in 2002 (UG 2002). Therefore, we will discuss pre-UG02 type careers and UG02 type careers.

The most important differences among jobs in academia are

- temporary vs. permanent employments: a temporary contract has a determined end date. In academia it is sometime the case that this type of job cannot be renewed at the same institution.
- tenured vs. non-tenured employments: tenured people cannot be dismissed (“Pragmatisierung”). There are no new tenured positions according to UG02.

Note that a permanent position is not necessarily tenured ! Academic careers follow a strictly hierarchical system which we describe in the following.

Pre-UG02 type Careers

Since no new personnel can enter this career model, we only describe the characteristics of those positions currently being held by members of the department. These positions are all permanent and tenured.

- Assistant Professors (“Universitätsassistenten, Ass.Prof.”): highest academic qualification is PhD/doctorate. Sample representative members of the department: Clemens Amstler, Bernhard Collini-Nocker.
- Associate Professors (“Universitätsdozenten, Ao.Univ.Prof.”): highest academic qualification is the habilitation; promoted from the rank of assistant professor after one’s habilitation is accepted. Sample representative members of the department: Elmar Eder, Helge Hagenauer, Martin Held, Helmut Mayer, Andreas Uhl.
- Full Professors (“Universitätsprofessoren, Univ.-Prof. or o.Univ.Prof.”): highest academic rank, but no habilitation needed. Full professors are selected by an appointment committee. Usually, they come from a different university. Sample representative members of the department: Jochen Pfalzgraf, Wolfgang Pree, Marian Vajtersic, Peter Zinterhof.

Only associate and full professors are eligible as advisors for MSc and PhD theses.

UG02 type Careers

There are no tenured positions according to UG02.

- Research Assistants (“Wissenschaftliche Mitarbeiter in Ausbildung”): highest academic qualification DI or Master, temporary position (max. 4 years, no renewal possible). Position holders usually write their dissertation / PhD-thesis during their appointment. Teaching is allowed only during the last two years of the contract. Sample representative members of the department: Dominik Engel, Thomas Soboll, Rainer Trummer, Gerald Stieglbauer.
- Post-Docs (“Universitätsassistenten”): highest academic qualification is PhD/doctorate, temporary position (max. 6 years, no renewal possible). Sample representative members of the department: Rade Kutil, Stefan Resmerita.
- Contract Professors (“Vertragsprofessoren, Univ.-Prof.”): highest academic rank, temporary position (max. 7 years, renewal possible). Selected by an appointment committee. Sample representative members of the department: Werner Pohlmann, Manfred Tschelligi.
- Full Professors (“Universitätsprofessoren, Univ.-Prof.”): highest academic rank, permanent position, but no habilitation needed. Full professors are selected by an appointment committee. Usually, they come from a different university. Sample representative member of the department: Christoph Kirsch.

Only contract and full professors are eligible as advisors for MSc and PhD theses.

Further Academic Positions

- External lecturers (“Lehrbeauftragte”): come from different institutions and this job is restricted to teaching specific courses (temporary position for a 1/2 year, renewal possible). Sample representative members of the department: Claudia-Lidia Badea, Roland Schwaiger, Josef Templ, Norbert Ulamec.
- Honorary Professors (“Honorarprofessoren”): similar privileges as associate and full professors on a temporary basis (max. 5 years, renewal possible) but are paid similar to external lecturers only for their teaching. Sample representative members of the department: Ulrich Hofmann, Gerhard Mayer, Carl-Herbert Rokitansky.
- Visiting Professors (“Gastprofessoren”): usually they are on leave from their institution for a sabbatical and have similar privileges as associate and full professors for the time of their visit. Currently, the department has no visiting professor.
- Project staff (“Projektmitarbeiter”): paid project co-workers with various academic degrees, temporary positions depending on the fundings available within research projects. No teaching.

Note that people can be both external lecturers and project staff. Only honorary and visiting professors are eligible as advisors for master and PhD-theses.

Scientific Text Processing

- What is TEX/LATEX?
- Getting Started with LATEX,
- Structuring Commands,
- Type Styles and Sizes,
- Mathematics and LATEX,
- Figures and Tables,
- Cross-referencing,
- Defining New Commands and Environments,
- Trouble Shooting.

What is $\TeX$?

“$\TeX$ is a new typesetting system intended for creation of beautiful books – and especially for books that contain a lot of mathematics. By preparing a manuscript in $\TeX$ format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world’s finest printers.”

Donald Knuth

- $\tau\varepsilon\chi$ = art.

Basics of TEX

- Professional-quality layout;
- Predefined layouts for standard text styles (article, book, letter,. . .);
- Tons of features for solving a wide array of layout problems;
- Particularly good at formatting mathematical formulae;
- Available for most computing platforms;
- The layout does not depend on the output device (monitor, laser printer,. . .);
- All the TEX source code is publicly available;
- Pretty much every individual layout option can be changed and adapted to specific needs – providing that one knows how to do it;
- TEX comes for free.

What is LaTeX?

“LaTeX adds to TeX a collection of commands that simplify typesetting by letting the user concentrate on the structure of the text rather than on formatting commands. In turning TeX into LaTeX, I have tried to convert a highly-tuned racing car into a comfortable family sedan. The family sedan isn’t meant to go as fast as a racing car or be as exciting to drive, but it’s comfortable and gets you to the grocery store with no fuss. However, the LaTeX sedan has all the power of TeX hidden under its hood, and the more adventurous driver can do everything with it that he can with TeX.”

Leslie Lamport

Basics of LaTeX

- Designed and implemented by Leslie Lamport in the early 80s;
- Lots of macros that are based on TeX;
- Lamport: “LaTeX is your typographic designer, and TeX is its typesetter”;
- WYSIWYG: “what you see is what you get”;
- WYSIWYG: “what you see is all you’ve got” (B. Reid, B. Kernighan);
- LaTeX is not a WYSIWYG program;
- LaTeX enables the author to concentrate on the logical structure of a text, rather than on details of its layout;
- LaTeX offers (and enforces) a “logical design”, contrary to the “visual design” of a conventional WYSIWYG program;
- We are in a migration phase from LaTeX 2.09 to LaTeX 3; the current version of LaTeX is called LaTeX 2$_\varepsilon$.

Advantages of LaTeX

- Professional layouts are readily available;
- Complex mathematical formulae are typeset neatly;
- Only a few structuring commands control the logical structure of a document;
- Footnotes, tables of contents, tables, figures, bibliographic data, and similar cross-referencing are easily incorporated into a document;
- All cross-references are updated automatically when the document changes;
- LaTeX is the most widely accepted standard for writing scientific papers in the fields of computer science;
- LaTeX files are plain ASCII files, and your favorite text editor suffices for preparing a LaTeX document;
- LaTeX is publicly available, including its source code, and it comes for free.

Disadvantages of LaTeX

- Not a WYSIWYG program;
- Minor modifications of the default layout are easily accomplished, but major changes require a thorough understanding of LaTeX;
- The support for non-English languages still ought to be improved;
- Complicated figures are hard to prepare using LaTeX, and require the use of some drafting package.

Books on LaTeX

- D.E. Knuth:
  TeX book.
  Addison-Wesley, 1988. ISBN 0-201-13448-9.

- L. Lamport:
  LaTeX. A Document Preparation System.
  Addison-Wesley, 2nd edition, Nov 1994. ISBN 0-201-52983-1.

- M. Goossens and F. Mittelbach and A. Samarin:
  The LaTeX Companion.
  Addison-Wesley, 2nd edition, 1994. ISBN 0-201-54199-8.

Books on LaTeX (cont'd)

- M. Goossens and S. Rahtz and F. Mittelbach:
  The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.
  Addison-Wesley, 1997. ISBN 0-201-85469-4.

- S. Rahtz and F. Mittelbach:
  The LaTeX Web Companion: Integrating TeX, HTML and XML.
  Addison-Wesley, 1999. ISBN 0-201-43311-7.

- H. Kopka and P.W. Daly:
  A Guide to LaTeX: Document Preparation for Beginners and Advanced Users.
  Addison-Wesley, 1999. ISBN 0-201-39825-7.

LaTeX Input Characters

- The input to LaTeX is an ASCII text file.
- Unless stated otherwise (in a so-called *Local Guide*, for instance), the following characters are the only ones that normally appear in a LaTeX input file.

letters: A,. . . ,Z; a,. . . ,z;
digits: 0,. . . ,9;
punctuation chars: `.  :  ;  .  ?  !  ` ' " ( ) [ ] - / * @`
special chars: `#  $  %  &  _  {  }  ~  ^  \`
math chars: + = | < >

LaTeX Input Characters (cont'd)

- Note that the percent sign (%) is interpreted by LaTeX as the start of a comment! (LaTeX will ignore the rest of a line after reading a % sign.)

- Similarly, all the other special characters have a special meaning for LaTeX.

- In order to produce any of the signs # $ % & _ { }, the sign itself has to be preceded by a back slash. That is, $ is produced by means of `\$`.

- Some installations of LaTeX may be able to handle German "umlaut" (and similar characters that do not belong to English) directly as part of the input. However, be warned that using those characters may render your LaTeX document less portable. Thus, I prefer to encode those characters as explained later on.

Basic LaTeX Document

- A LaTeX document starts with a `\begin{document}` command and ends with `\end{document}`.

- The part of the input file preceding the command `\begin{document}` is called the *preamble*.

- The preamble contains declarations which globally affect the appearance of the formatted text.

- Note that the number of spaces (or line breaks) in the input file does not matter. One space is as good as ten spaces.

- Also, LaTeX only cares about empty lines (that separate paragraphs), but does not care about how lines are broken between consecutive non-empty lines.

Basic LaTeX Document (cont'd)

```
\documentclass[12pt,fleqn]{article}
   % Specifies the document class and the type size.
   % Also, we do not want equations to be centered.
   % The preamble begins here.
\title{\textbf{\LaTeXe\ }}
   % Declares the document's title. We request a bold-face font.
\author{Martin Held}
   % Declares the author's name.
\date{February 24, 2003}
   % Deleting this command produces today's date.
\begin{document}
   % End of preamble and beginning of text.
\maketitle
   % Produces the title.
\section{Introduction}
   % Declares a section.
This is a short survey of the \LaTeXe\ typesetting system. [...]
\end{document}
   % End of document. LaTeX won't read beyond this line!
```

Document Classes and Options

- Standard classes for ordinary documents are `article`, `report`, `book`, `letter`, and `slides`.
- By default every document is formatted for `10pt` types.
- However, `11pt` and `12pt` types can be requested. (Larger type sizes can be defined, too.)
- Additional document-class options include `fleqn` and `twoside`, among many others. We refer to the LaTeX Book for details.
- User-defined options can be included, too. However, in this case the environment variable `TEXINPUTS` has to be set to the appropriate search path if a user-defined document-class option or package is not contained in the actual working directory. E.g., for `tcsh`:

```
setenv TEXINPUTS .:${HOME}/styles//:${HOME}/figures:${TEXINPUTS}.
```

Running LaTeX

1. Write or modify the document by means of an ASCII editor, and save it to a file with extension `.tex`.

2. Invoke LaTeX, e.g.: `latex foo.tex` in order to process the LaTeX file `foo.tex`.

3. In case of LaTeX errors go back to 1.

4. Run BibTeX if a (new) bibliographic data base is to be included.

5. Re-run LaTeX until all symbolic labels for cross-referencing are stable. (LaTeX will tell you whether any labels have changed.)

6. Use a previewer in order to view the DVI file. E.g., `xdvi foo.dvi` under the X11 windowing system.

7. Back to 1 if changes are to be carried out.

8. Use a device driver in order to convert the DVI file to a file that can be printed on your printer. E.g., `dvips -o foo.ps foo.dvi` in order to create PostScript.

Commands for Structuring (Sectioning)

- A sectional unit is begun by a sectioning command with the unit's title as its argument.

```
\section{Commands for Structuring}
\subsection{Sectioning}
```

- LaTeX automatically generates the (sub)section numbers — subsections are numbered within sections.

- The sectioning commands provided include `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph`. Note that the set of commands available depends on the document class.

- For omitting the numbers, add a `*` after the command.

- There is also an `\appendix` command, which does not directly produce text. Rather, it causes sectional units to be numbered properly for an appendix.

Structuring Displayed Material

- LaTeX uses a construction called *environment* in order to group portions of text that are subordinate to the surrounding text or that function as equal units.

- An environment is generated by typing the commands

  `\begin{name}...\end{name}`,

  where `name` denotes the name of the environment.

- The `\begin` and `\end` commands delimit the scope of the environment.

- Examples of environments are given by `quote`, for making quotations, `verse`, for doing poetry, and by `verbatim`, which is used for simulating typed text.

List-Making Environments

- LaTeX provides three predefined environments for making lists: `itemize`, `enumerate`, and `description`.
- In all three environments, every new list item is begun with an `\item` command.
- Here comes an `enumerate`d list:

```
\begin{enumerate}
 \item A single list item.
 \item And yet another one.
\end{enumerate}
```

1. A single list item.
2. And yet another one.

- The following example shows an `itemize`d list:

```
\begin{itemize}
 \item A single list item.
 \item And yet another one.
\end{itemize}
```

⋆ A single list item.

⋆ And yet another one.

List-Making Environments (cont'd)

- In the `description` environment, the `item` command takes an optional argument:

```
\begin{description}
 \item[Foo:] A single list item.
 \item[FooFoo:] And yet another one.
\end{description}
```

Foo: A single list item.
FooFoo: And yet another one.

List-Making Environments (cont'd)

1. Of course, LaTeX allows to nest lists, usually up to some fixed depth (such as 7).
2. If the same environments are nested, e.g. an `enumerate` environment within an `enumerate` environment, then LaTeX automatically chooses different kinds of labels for each list.
3. (a) There is a default numbering scheme for nested lists.
   (b) Of course, you are free to change the default scheme if you don't like it.
4. More customized lists can be generated by using the `list` environment. See the LaTeX Book for details.

Type Styles

- Most sentences, including this phrase, are printed in a type style called 'roman'. Roman is LaTeX's default type style.

- LaTeX distinguishes between three components that specify a type style:

  ⋆ *shape*,
  ⋆ *series*,
  ⋆ *family*,

  which can be combined in order to produce more elaborate effects.

- Quite a few more fancy typefaces and fonts are available; consult the LaTeX Companion for details.

- The old LaTeX 2.09 commands for changing type styles (e.g., `\tt`) still work, but the use of the new commands (e.g., `\texttt` instead of `\tt`) is encouraged.

Type Styles

- Shown below are the basic type styles, together with the declarations that turn them on.

```
\textrm{This is a roman type style.}
\textbf{This is a bold type style.}
\textsf{This is a sans serif type style.}
\textsl{This is a slanted type style.}
\textsc{This is a Small Caps type style.}
\texttt{This is a typewriter type style.}
\textit{This is an italic type style.}
```

This is a roman type style.
This is a bold type style.
This is a sans serif type style.
This is a slanted type style.
THIS IS A SMALL CAPS TYPE STYLE.
`This is a typewriter type style.`
This is an italic type style.

Type Sizes

- The following declarations select a type size; they are listed below in nondecreasing size.
  - ⋆ `\tiny`;
  - ⋆ `\scriptsize`;
  - ⋆ `\footnotesize`;
  - ⋆ `\small`;
  - ⋆ `\normalsize`;
  - ⋆ `\large`;
  - ⋆ `\Large`;
  - ⋆ `\LARGE`;
  - ⋆ `\huge`;
  - ⋆ `\Huge`.
- Note that the actual type size produced by one of these size declarations depends on the default type size of the document.

Type Sizes (cont'd)

- Note that some declarations may have the same effect, depending on the document class and default type size used.
- Of course, changes of type style and type size can be combined. For instance, the command `{\textit{\texttt{\LARGE word}}}` produced this *word*.
- Note, however, that you should not expect your LaTeX installation to provide all the fonts for all imaginable combinations of type styles at all possible type sizes.
- If the `mktexpk` program is installed, `dvips` will automatically invoke METAFONT to generate fonts that do not already exist, *provided* that a METAFONT source for this font is available.

Aligning Text in Columns

- For aligning text in columns, LaTeX offers the `tabbing` and the `tabular` environments.
- In the `tabbing` environment, text is aligned by explicitly setting tab stops, as it is done with an ordinary typewriter.
- Tab stops are set using the `\=` command, and `\>` moves to the next tab stop.
- Lines are separated by the `\\` command.

Aligning Text in Columns (cont'd)

- The following code produces the listing given below:

```
\begin{tabbing}
Bears: \= Kodiak Bear \= (Kodiak Island), \kill
Bears: \> Polar Bear  \> (Arctic Region),\\
       \> Kodiak Bear \> (Kodiak Island), \\
       \> Grizzly     \> (Western US, Canada).
\end{tabbing}
```

```
Bears: Polar Bear    (Arctic Region),
       Kodiak Bear  (Kodiak Island),
       Grizzly      (Western US, Canada).
```

Aligning Text in Columns (cont'd)

- The `tabular` environment is somewhat similar to the `tabbing` environment.
- Columns are separated by `&`, and an input line is ended by `\\`.
- Frames can be made by requesting horizontal and vertical lines to be drawn by means of specifying `\hline` and `|`.

Aligning Text in Columns (cont'd)

- The following code produces the listing given below:

```
\begin{tabular}{||l|c|r||} \hline
\multicolumn{3}{||c||}{Bears of the World} \\
\hline \hline
Bears & Polar Bear  & (Arctic Region) \\ \hline
      & Kodiak Bear & (Kodiak Island) \\ \cline{2-3}
      & Grizzly     & (Western US, Canada) \\
\hline \hline
\end{tabular}
```

Bears of the World		
Bears	Polar Bear	(Arctic Region)
	Kodiak Bear	(Kodiak Island)
	Grizzly	(Western US, Canada)

Aligning Text in Columns (cont'd)

- Note that the `@{`*string*`}` construct makes it possible to specify the column separator. Effectively, this command kills the intercolumn space and replaces it by *string*.
- The following LaTeX code is a standard example for explaining how to line up decimal numbers in one decimal-point-justified column:

```
\begin{tabular}{c r @{.} l} \hline
Symbolic Term & \multicolumn{2}{c}{Numerical Value} \\ \hline
$\pi$                   & 3&1416  \\
$\pi^{\pi}$             & 36&46   \\
$(\pi^{\pi})^{\pi}$  & 80662&7 \\
\end{tabular}
```

Symbolic Term	Numerical Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

Math Stuff

- LaTeX is especially good in displaying mathematical stuff.
- It provides the `displaymath` and `equation` environments for displaying formulae.
- These environments are the same except that `equation` numbers the formulae and `displaymath` doesn't.
- For shorthand, `\[ ...\]` may be typed instead of `\begin{displaymath} ...\end{displaymath}`.

$$x' + y^{2} = z_{1}^{2}$$

```
\[ x' + y^{2} = z_{1}^{2} \]
```

Math Stuff (cont'd)

- A numbered equation:

$$x' + y^2 = z_2^2 \tag{1}$$

```
\begin{equation} \label{eq:foo}
   x' + y^{2} = z_{2}^{2}
\end{equation}
```

- A formula that appears in the running text, a so-called *in-line formula*, is produced by the `math` environment.

- For shorthand, this environment can be invoked and delimited by `\( ...\)` or by `$ ...$`.

- Another way for producing an in-line formula is the `\ensuremath` command. It is especially useful for defining a command that can appear in both normal text and formulae.

Math Stuff (cont'd)

- Subscripts and superscripts are made with the _ and ^ commands.
- Fractions are denoted by the / symbol.
- Large fractions may also be displayed using the `\frac` command.
- As a rule of thumb, many mathematical symbols can be generated by typing commands that are related to the English names of the symbols.

$$x_{1}^{y^{2}}$$

```
\[ x_{1}^{y^{2}}  \]
```

$$\frac{x + y/2}{x - \frac{y}{z+1}}$$

```
\[ \frac{x + y/2}{x - \frac{y}{z+1}} \]
```

Sample Math

$$\sum_{i=1}^{n} \sqrt{x_{i}}$$

```
\[ \sum_{i=1}^{n} \sqrt{x_{i}} \]
```

$$\lim_{n \rightarrow \infty} 1/n = 0$$

```
\[ \lim_{n \rightarrow \infty} 1/n = 0 \]
```

$$\int_{0}^{1} x \sin 1/x \, dx$$

```
\[ \int_{0}^{1} x \sin 1/x \, dx \]
```

Sample Math (cont'd)

- All the previous formulae were generated as off-line formulae. The following example demonstrates the effect of replacing `$ ... $` by `\[ ... \]`: $\sqrt{\lim_{n \rightarrow \infty} \int_{-n}^{n} \frac{1}{x^{2}} \sin x \, dx}$; and off-line:

$$\sqrt{\lim_{n \rightarrow \infty} \int_{-n}^{n} \frac{1}{x^{2}} \sin x \, dx}.$$

- And this is the corresponding math code (without `$ ... $` or `\[ ... \]`):

```
\sqrt{ \lim_{n \rightarrow \infty}
       \int_{-n}^{n} \frac{1}{x^{2}} \sin x \, dx }
```

- Note that symbols like $\int$ are variable-sized. Their sizes do not only depend on the type size used but also on whether they are displayed in-line, i.e., within `$ ... $`, or off-line, i.e. within `\[ ... \]`.

Mathematical Symbols

- LaTeX supports a variety of special mathematical symbols. (See the LaTeX Book.) Symbols provided include
  - ⋆ *(binary) operation symbols*, e.g. $\pm$ (`$\pm$`), $\div$ (`$\div$`), $\cdot$ (`$\cdot$`), $\cap$ (`$\cap$`), $\cup$ (`$\cup$`);
  - ⋆ *relation symbols*, e.g. $\leq$ (`$\leq$`), $\subset$ (`$\subset$`), $\in$ (`$\in$`);
  - ⋆ *arrow symbols*, e.g. $\leftarrow$ (`$\leftarrow$`), $\Uparrow$ (`$\Uparrow$`), $\mapsto$ (`$\mapsto$`);
  - ⋆ *miscellaneous symbols*, e.g. $\aleph$ (`$\aleph$`), $\forall$ (`$\forall$`), $\exists$ (`$\exists$`);
  - ⋆ *delimiters*, e.g. $\{$ (`$\{$`), $\lfloor$ (`$\lfloor$`), $\rangle$ (`$\rangle$`).
- Observe that all those symbols can only be used in the so-called math mode, i.e., within the scope of `$` . . . `$` or `\[` . . . `\]`.

Mathematical Delimiters

- Delimiters can also be used in multi-line formulae. The commands `\left` and `\right` are used in order to make them “fit around”.

- This piece of code produces the following (nonsense) multi-line formula:

```
\[   \vec{a} + \vec{b} = \left(  \begin{array}{c}
                                   c_{x} \\
                                   c_{y}
                                \end{array}
                         \right\}                   \]
```

$$\vec{a}+\vec{b}=\left(\begin{array}{c} c_{x} \\ c_{y} \end{array}\right\}$$

- LaTeX will complain if no matching right delimiter is found – you may use `\right.` as a dummy right delimiter in this case.

Mathematical Equations

- For coding sequences of equations it is convenient to use the `eqnarray` environment, which is very much like a special `array` environment.

$$\begin{aligned} x &= 2y-3z \qquad (2)\\ 5x+7y &\geq a+b+c+d+e+f+g+h+i+ \\ & \quad j+k+l+m+n+o+p+q \qquad (3)\end{aligned}$$

```
\begin{eqnarray}
   x         &    =    &  2y - 3z \\
   5x + 7y   &   \geq  &  a + b + c + d + e + f + g + h +
                          i + \nonumber \\
             &         &  j + k + l + m + n + o + p + q
\end{eqnarray}
```

Mathematical Equations (cont'd)

- Note that the alignment is handled by LaTeX. You can put `\tiny` around the `eqnarray` construct, and it will again be aligned properly:

$$\begin{aligned} x &= 2y-3z && (4)\\ 5x+7y &\geq a+b+c+d+e+f+g+h+i+ \\ & \quad j+k+l+m+n+o+p+q && (5)\end{aligned}$$

- LaTeX is also good in producing Greek and other (foreign) letters. The command for producing a Greek letter is obtained by placing a \ in front of the name of the letter. For instance, `$\gammma$` produces a γ.

- Uppercase Greek letters are generated by capitalizing the first letter of the command name, as long as the uppercase Greek letter is not the same as its Roman equivalent. For instance, `$\Gamma$` produces Γ.

Non-Latin Characters

- LaTeX was originally designed for English. It does not support any other languages on its own.

- As far as German is concerned, a minimal subset of standardized commands for German has been agreed upon.

  - ⋆ `\"a` or `"a` produces ä;
  - ⋆ `\ss` or `"s` produces ß;
  - ⋆ ``"` `` and `"'` produce German left and right double quotes.

- One should note that LaTeX does not hyphenate German words correctly without being supplied with German hyphenation patterns!

- For extended language support – such as hyphenation patterns, names of document elements, etc. – LaTeX 2ε provides the package `babel`, with option `austrian` for replacing English dates (etc.) by the equivalent Austrian versions.

Non-Latin Characters (cont'd)

- Cyrillic, Hebrew and a lot of other special-language character sets can be produced similarly to producing Greek characters, provided that the fonts required for actually generating them are available.

- Special symbols of other languages are produced similarly to German symbols.

- For instance, `\'e` produces é, `\~n` results in ñ, and `\c{c}` yields ç.

- LaTeX also supports a variety of other special characters, such as © (`\copyright`), § (`\S`), and £ (`\pounds`).

Euro Symbol

- The European Commission defined the Euro symbol as a strictly geometric logo. That is, the official symbol is meant to be a sans serif character, always the same regardless of the font being used. This violates normal typesetting conventions. (E.g., the Dollar and Pound signs are different for different fonts.)

- When using an Euro symbol, care should be taken that the symbol is available as an outline-based Type 1 font (rather than as a pixel-based Type 3 font).

- As of 2001-Dec-05, one of the better options seems to be the use of Martin Vogel's *Marvosym Font Package*.

- After putting `\usepackage{marvosym}` into the preamble (and after installing the proper font descriptions), the Euro symbol can be created: € (`\EUR`).

- This package also provides quite a few other symbols, such as CE (`\CEsign`), ✂ (`\Rightscissors`), ⚽ (`\Football`), or ♑ (`\Capricorn`).

- The conventional resizing commands of LaTeX may be applied. E.g., `{\LARGE\EUR}` produces €.

Figures and Tables

- Since pictures and tables cannot be split at page breaks LaTeX provides two environments, `figure` and `table`, that can float to convenient places.

- The `figure` environment is generally used for pictures and the `table` environment for tabular information.

- The major difference between both environments is how they are captioned: the figure's caption is below the body of the figure whereas the table's caption goes above the table.

- All LaTeX cares about is to find suitable positions, as long as possible, for placing their contents without generating half-empty pages.

```
\begin{figure}[!tbph]
The body of the figure goes here. You may want to leave some space by using the
\vspace{...} command.
\caption{The caption goes here.}
\end{figure}
```

Figures and Tables (cont'd)

- LaTeX's decision where to place a floating object can be influenced by specifying any combination of the parameters `t, b, p` and `h`, where `t` means that you suggest to place the figure at the top of the (following) page, relative to the position of the text around the place where you have specified the figure in your input file.
- Similarly, `b` stands for bottom. A `p` indicates that LaTeX is allowed to generate an extra page of floats, which does not contain any text.
- If you are really keen on having the figure put exactly where you have specified it, you may want to try `h` — for 'here'; LaTeX sometimes even cares about your wishes.
- If you add a `!` to the location, LaTeX tries harder to satisfy your request.

LaTeX and PostScript Figures

- For creating simple pictures within a figure, the `picture` environment may be used.
- However, the creation of pictures is not a real highlight of LaTeX and it is usually better to import pictures created by some other system.
- As long as pictures are imported in ‘*Encapsulated PostScript*’ (EPS) style, LaTeX automatically takes care of the amount of height needed by the picture.
- An EPS figure is imported by the following commands, typically placed into the body of a figure.

```
\begin{center}
\includegraphics[width=8.3cm]{name_of_file}
\end{center}
```

LaTeX and PostScript Figures (cont'd)

- Note, however, that the `graphicx` package must be included by putting `\usepackage{graphicx}` into the preamble.
- For our example, the figure will be scaled to fit into a horizontal space with width 8.3cm.
- Similarly, it can be scaled to fit into a prescribed vertical space. The `width` or `height` command is optional; omitting it causes LaTeX to reproduce the figure at its original size.
- We will learn more about how to incorporate PostScript files after discussing packages for drawing figures . . .

Cross-referencing

- LaTeX can automatically generate a table of contents and similar cross-references if asked to do so.
- The command `\tableofcontents` tells LaTeX where to put the table of contents within the document.
- Note that it requires two runs in order to generate a correct table of contents.
- In the first run LaTeX extracts all necessary sectional information and writes it to a file with extension `.toc`.
- When invoked for the second time, it reads this file and generates a table of contents according to the layout arranged in the previous run. Besides, it issues a warning message if the actual sectional information does not correspond to the old table of contents read from the `.toc` file.

Cross-referencing (cont'd)

- The commands `\listoffigures` and `\listoftables` produce a list of figures and a list of tables, respectively. They work just like the `\tableofcontents` command, except that files with extensions `.lof` and `.lot` are involved.
- Nearly every numbered environment can be referred to after a *key* has been assigned to it.
- A key is assigned by means of the `\label{key}` command, which can be put anywhere within the scope of the environment to be referenced, and where `key` is the symbolic key.
- Reference is made by means of the `\ref{key}` command.
- As in the case of generating a table of contents, LaTeX needs two runs and one additional file, with extension `.aux`, for generating correct references.

Cross-referencing (cont'd)

- For instance, recall that our first numbered equation was Equation 1 on Slide 116.
- The label for this reference was generated by putting `\label{eq:foo}` within the environment of the equation to be referenced, and by referring to it as `\ref{eq:foo}`.
- Similarly, sections, pages and other numbered units can be referenced.
- However, for references to pages it is necessary to substitute the `\ref` command by a `\pageref` command.
- For establishing a reference to a figure or a table, make sure to put the `\label` command *after* the `\caption` command.

Bibliographic Citations

- A citation is a cross-reference to another publication, such as a book.
- With LaTeX you can use a separate program called BibTeX to generate bibliographical data from information stored in a bibliographical database, i.e., in a collection of files with extensions `.bib`.
- If the bibliographical database is not contained in your actual working directory then you may want to inform LaTeX where to find this database by setting the environment variable `BIBINPUTS` to the appropriate search path, e.g.,

```
setenv BIBINPUTS .:$HOME/papers/biblio//
```

- When calling BibTeX, the information requested by `\cite` commands is extracted from the bibliographical database and is stored in two files with extensions `.bbl` and `.blg`.

Bibliographic Citations (cont'd)

- The following example shows a sample entry to a BIB file:

```
@string{AW      = "Addison-Wesley"}
@book{Lamp94,
      author={L. Lamport},
      title={\LaTeX. A Document Preparation System},
      publisher=AW,
      note={ISBN 0-201-52983-1},
      edition={2nd},
      month=nov,
      year=1994}
```

- As long as the bibliographical database is not changed and no new `\cite` commands are added, the `.bbl` and `.blg` files correctly represent the bibliographical data needed for making citations.

- As with all other symbolic pointers LaTeX needs two runs in order to have all references established.

Bibliographic Citations (cont'd)

- For every cited reference, a bibliography entry is extracted from the BIB file, and neatly formated.
- The placement of the bibliography is controlled by the placement of the `\bibliography{bib_file}` command within the LaTeX file.
- Here, `bib_file.bib` is the name of a file containing the bibliographical data. (It is also possible to use several bib-files as arguments of the `\bibliography` command.)
- A detailed explanation of BibTeX is out of the scope of this survey. For additional information on LaTeX and BibTeX you may want to consult the LaTeX Book [1].

Bibliographic Citations (cont'd)

- Using the sample bib entry, this reference is produced by the command `\cite{Lamp94}`.

- Note that you will have to run BibTeX on the LaTeX document in order to prepare the bibliographic references.

- E.g., `bibtex foo` will run BibTeX on the file `foo.tex` and its corresponding 'auxiliary' file `foo.aux`. Then, you will have to re-run LaTeX twice in order to establish and confirm all citations.

Theorems and Similar Environments

- Theorems can be produced neatly, too.
- LaTeX provides a `\newtheorem` declaration in order to define environments for particular theorem-like environments.

  Hypothesis 1. [Murphy] *There is always one error lefft.*

  ```
  \newtheorem{hypothesis}{Hypothesis}
  \begin{hypothesis}[Murphy] \label{hyp:murphy}
  There is always one error lefft.
  \end{hypothesis}
  ```

- Like other numbered environments, theorems can also be referenced, and this sometimes even works in spite of Hypothesis 1, which has been referenced by means of `\ref{hyp:murphy}`.

Defining New Commands and Environments

- The layout of a document heavily depends on the document-class options and add-on packages used for formatting it.
- These optional packages contain a myriad of control parameters, environments, and the like, which all can be modified individually in order to fit special purposes.
- However, this is the hard way of forcing LaTeX to modify its formatting strategies, i.e., this is the domain of LaTeX wizards!
- And if all else fails, you can still use plain TeX commands — this is the really hard way and asking a TeX guru is recommended!
- The easier way to modify LaTeX's way of formatting a document is to use the `\newcommand` and `\newenvironment` commands, which allow to define new commands and environments based on already existing ones.
- Another easy alternative is to use one of the many existing add-on packages, see the LaTeX Book or the LaTeX Companion.

Trouble Shooting

- Always remember that LaTeX is nothing but a type setting system that has to rely on your commands.
- For instance, it cannot guess where you meant to insert a parenthesis but forgot to do so!
- Thus, it will bark about any syntactical error that it can detect.
- Also, note that syntactical correctness need not imply a logical correctness.
- For instance, LaTeX will be perfectly happy to set an entire book in `\tiny` type size, which may be different from what you intended to do.

Trouble Shooting Guidelines

1. Consult the LaTeX Book and the LaTeX Companion. (Yes, indeed: RTFM!)
2. Make sure that all parentheses occur in matching pairs. It is good practice to enter `{}` prior to entering anything between the parentheses. (Some editors support this and will automatically re-position the cursor.)
3. Make sure that all `\begin` and `\end` commands occur in matching pairs.
4. Similarly, all math delimiters need to occur in matching pairs.
5. Rerun LaTeX frequently. The load that it will place on the CPU is no issue with modern computers, but it will tremendously help you with locating problems.
6. At all positions where one space or empty line is allowed, several spaces and empty lines are allowed. It will help your first attempts to locate a problem if your LaTeX file is formatted neatly!
7. Recall that a % sign starts a comment for LaTeX, and that it will ignore the rest of the line.

Drafting Figures and Generating Plots

- Drawing Figures in LaTeX;
- PostScript;
- Drafting packages:
  - ⋆ tgif,
  - ⋆ xfig,
  - ⋆ Ipe,
  - ⋆ Dia;
- Utilities:
  - ⋆ pstoedit,
  - ⋆ psfrag;
- Plotting:
  - ⋆ xgraph,
  - ⋆ gnuplot.

Drawing Figures with LaTeX

- Simple figures can be generated using the *picture* environment of LaTeX:
  `\begin{picture}(`*width*`,`*height*`)(`*x-lower_left*`,`*y-lower_left*`)`
  `\end{picture}`
  with all coordinates being expressed in terms of `\unitlength`.

- The unit length can be set using the command `\setlength`. E.g., the following command sets the unit length to $5mm$:
  `\setlength{\unitlength}{5mm}`.

- `(`*x-lower_left*`,`*y-lower_left*`)` specifies the coordinates of the lower-left corner of the picture. If absent, the lower-left corner has coordinates `(0,0)`.

- Two standard line widths are available within the picture environment: `\thinlines` and `\thicklines`.

Drawing Figures with LaTeX (cont'd)

- The `\begin{picture}` command puts LaTeX into *picture mode*. The only things that can appear inside the picture environment are the commands `\put`, `\multiput`, `\qbezier`, and `\graphpaper`, and declarations such as `\thicklines`.

- The basic command for drawing is the `\put` command:
  `\put(`*x-coord*`,`*y-coord*`){`*picture object*`}`.

- Valid picture objects are text, (dashed) boxes, lines, arrows, (filled) circles, ovals:
  `\put(`*x-coord*`,`*y-coord*`){`*my_text*`}`
  `\put(`*x-coord*`,`*y-coord*`){\framebox(`*width*`,`*height*`){`*my_text*`}}`
  `\put(`*x-coord*`,`*y-coord*`){\line(`*x-dir*`,`*y-dir*`){`*length*`}}`
  `\put(`*x-coord*`,`*y-coord*`){\vector(`*x-dir*`,`*y-dir*`){`*length*`}}`
  `\put(`*x-coord*`,`*y-coord*`){\circle{`*radius*`}}`
  `\put(`*x-coord*`,`*y-coord*`){\oval(`*width*`,`*height*`)}`

Drawing Figures with LaTeX (cont'd)

- The reference point of a box is its lower-left corner.
- The box-drawing commands take one or two additional optional arguments for specifying the position of the text relative to the box: `l` (left), `r` (right), `t` (top), `b` (bottom). The default is to center the text horizontally and vertically within the box.
- For a line with *x-dir* $\neq 0$, the x-coordinate of its endpoint is given by *x-coord* $+$ *length*, and the y-coordinate of its endpoint is given by *y-coord* $+$ (*y-dir*/*x-dir*) $*$ *length*.
- For a line with *x-dir* $= 0$, the y-coordinate of its endpoint is given by *y-coord* $+$ *length*.
- Note that LaTeX can only draw lines for a few fixed slopes: *x-dir* and *y-dir* have to be integers between -6 and 6, and their greatest common divisor has to be one.

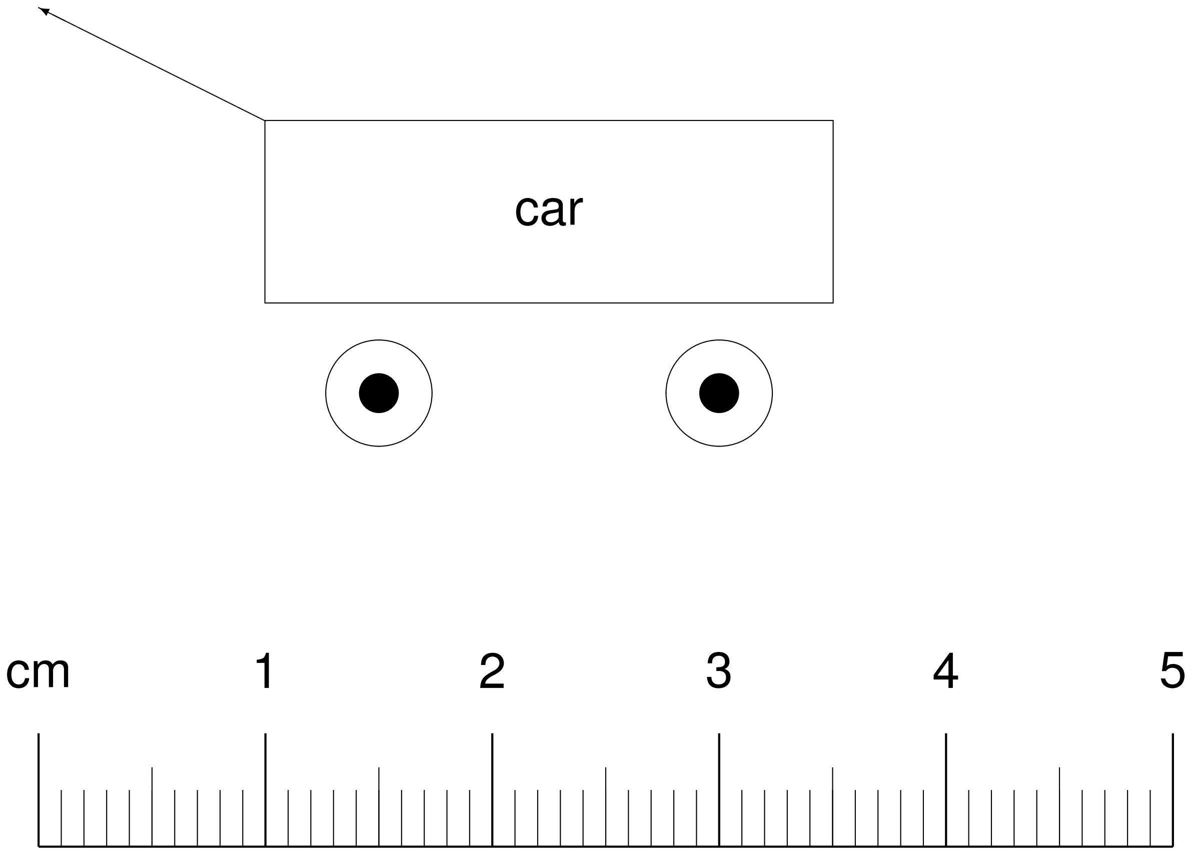
Drawing Figures with LaTeX (cont'd)

- Similarly, LaTeX supports only a fixed collection of radii of circles and disks.
- These restrictions on the slopes and radii are waived with the `pict2e` package.
- Objects can be saved by means of the `\savebox` and reused with the `\usebox` command.
- Repeated patterns can be generated with the `\multiput` command. See the LaTeX Book for details.

Sample LaTeX Picture

```
\newcounter{cms}
\setlength{\unitlength}{3mm}
\begin{center}
  \begin{picture}(50,39)
    \put(0,7){\makebox(0,0)[b]{cm}}
    \multiput(10,7)(10,0){5}
      {\addtocounter{cms}{1}\makebox(0,0)[b]{\arabic{cms}}}
    \put(15,20){\circle{6}}
    \put(30,20){\circle{6}}
    \put(15,20){\circle*{2}}
    \put(30,20){\circle*{2}}
    \put(10,24){\framebox(25,8){car}}
    \put(10,32){\vector(-2,1){10}}
    \multiput(1,0)(1,0){49}{\line(0,1){2.5}}
    \multiput(5,0)(10,0){5}{\line(0,1){3.5}}
    \thicklines
    \multiput(0,0)(10,0){6}{\line(0,1){5}}
    \put(0,0){\line(1,0){50}}
  \end{picture}
\end{center}
```

Sample LaTeX Picture (cont'd)



PostScript

- Designed by Adobe, Inc. in 1982, and documented in the PostScript Language Reference Manual (“the red book”) in 1985.
- PostScript (PS) is a device-independent *Page Description Language* (PDL) and has become a de-facto industrial standard. (It also has many elements of a Printer Control Language.)
- It is a *stack-oriented* programming language that relies on *reverse Polish notation* (RPN):

C:	sqrt ((3 $*$ 3) + (4 $*$ 4))
Lisp:	(sqrt (+ ($*$ 3 3) ($*$ 4 4)))
PostScript:	3 3 mul 4 4 mul add sqrt

PostScript (cont'd)

- The following function takes a Fahrenheit temperature and returns the corresponding Celsius temperature:

C:	int f2c (int t) { return ((t - 32) $*$ 5 / 9); }
Lisp:	(defun f2c (t) (/ ($*$ 5 (- t 32)) 9))
PostScript:	/f2c { 32 sub 5 mul 9 div } def

- Standard procedural (e.g., C, Ada) or functional (e.g., LISP) programming languages need parentheses in order to specify the order of execution of the clauses. PS needs no parentheses since its stack accumulates intermediate results, and the order of execution is always defined by the order in which the operations pushed onto the stack.
- Most programming languages use stacks internally. However, in general a user has no direct access to them.
- PS features polymorphic operators and late binding. It is weakly typed.

Sample PostScript Code

- The following PS code produces a shadowed PS logo:

```
/Times-Italic findfont 100 scalefont setfont

/PrintPS
{ 0 0 moveto
  (PostScript) show
} def

100 400 translate
.95 -0.05 0
{ setgray PrintPS -1.5 0.7 translate} for
1 setgray PrintPS

showpage
```

Sample PostScript Output

PostScript

Sample PostScript Code (cont'd)

- The following PS subroutine and main code produce a set of rays clipped to a text:

```
/Rays
{ 0 1 180
  { gsave
    rotate
    0 0 moveto
    340 0 rlineto
    stroke
    grestore
  } for
} def
```

Sample PostScript Code (cont’d)

- And here comes the main code for the PS “rays”:

```
/Times-BoldItalic findfont
120 scalefont setfont
50 400 translate
.24 setlinewidth
newpath
0 0 moveto
(StarLines) true charpath clip
newpath
240 -15 translate
Rays

showpage
```

PostScript Document Structuring Conventions

- A raw PS file lacks any easy-to-understand logical structure.
- Adobe specified the "*PostScript Document Structuring Conventions*" (DSC) for providing additional structural data in a PS file.
- A PS file is called conforming if it adheres to Adobe's DSC.
- In general, every application that generates PS output is expected to conform to Adobe's DSC.
- A line of DSC data is marked by %% or %! in the first two characters of the line. (%! is the so-called "file magic"; it may only appear in the very first line of a PS file.)
- The DSC data is partitioned into header comments, body comments, and trailer comments.

PostScript Document Structuring Conventions (cont'd)

- Here comes the header of a PS file generated by LaTeX and `dvips`:

```
%!PS-Adobe-2.0
%%Creator: dvipsk 5.58f Copyright 1986 ...
%%Title: drafting.dvi
%%Pages: 11
%%PageOrder: Ascend
%%BoundingBox: 0 0 596 842
%%DocumentPaperSizes: A4
%%EndComments
```

- Individual pages of a multi-page PS document are marked by `%%Page:` (followed by the page number).

Encapsulated PostScript

- Encapsulated PostScript files (EPS) are used for including PS data into an other PS applications (such as LaTeX).
- What turns an ordinary PS file into an EPS file is the BoundingBox, a box that describes where the figure sits on the page.
- The BoundingBox information typically resides in the first few lines of an EPS file.
- It is described by four numbers: the x, y-coordinates of the lower-left corner of the figure, followed by the x, y-coordinates of the upper-right corner of the image. E.g.,
  `%%BoundingBox: 0 0 453 216`.
- In this example, the figure sits right down in the bottom left-hand corner of the page. The numbers are points with 1pt = 1/72 inches. So, this figure is about 6 inches wide and 3 inches high.
- An EPS need not necessarily include the PS command “showpage”, which is the cue to a printer to actually print the page. Thus, an EPS file may not print by itself!

Encapsulated PostScript (cont'd)

- If you view a PS image with `ghostview`, the x, y-coordinates are displayed as you move the mouse to point at different parts of the image. Also, `ghostview` will display only the portion of the page described by the file's BoundingBox line. Thus, you can use `ghostview` to help you edit the BoundingBox line and to view the results.

- The scope of an EPS file included into another PS file is delimited by `%%BeginDocument:` and `%%EndDocument:`.

- An EPS file that contains a preview map in the format specified by Adobe is said to be in Encapsulated PostScript Interchange Format. When using four-letter extensions its default extension is `.epsi`.

- Preview maps are used for displaying PS data by applications that do not employ a complete PS machine for parsing a PS file.

- According to Adobe, a preview map is a bitmap encoded as hexadecimal ASCII values. A preview map is bounded by `%%BeginPreview` and `%%EndPreview`.

LaTeX and Encapsulated PostScript

- When TeX was implemented, PS and other graphics formats (like JPEG) did not exist. Thus, TeX does not have direct support for importing graphics.
- However, TeX allows DVI files to contain `\special` commands directed at programs that use DVI files.
- Since DVI files are most often converted to PS, the best supported format for imported graphics is EPS.
- With the release of LaTeX 2ε, the "LaTeX graphics bundle" was also released.
- The graphics bundle contains the "standard" `graphics` package and the "extended" `graphicx` package.
- Both packages offer roughly the same functionality, although the `graphicx` package is widely regarded as more user-friendly and slightly more efficient.

LaTeX and Encapsulated PostScript (cont'd)

- A file `foo.eps` can be included into a LaTeX document by using the `\includegraphics` command as follows:

  `\includegraphics[`options`]{`foo.eps`}`

  Typical options are the specification of a *height* or *width* of the graphics. E.g., `[width=3in]` requests the graphics to be scaled such that its total width is three inches.

- Any of the units accepted by LaTeX can be used for specifying dimensions: pt, in, cm, mm, . . .

- Instead of making the width be a fixed length (such as three inches), it may be better to make the width dependent upon the the `\textwidth` (or upon `\em`).

- Other options allow to rotate the graphics about a specified origin, to clip it to a viewing area, and to specify a bounding box of the graphics.

LaTeX and Encapsulated PostScript (cont'd)

- Note that `\includegraphics` does not end a paragraph. Thus, small symbols can be included into the running text.
- Single-page PS files can be converted to an EPS file by means of the `ps2epsi` utility distributed with Ghostscript. In particular, it will create information on the bounding box of the PS graphics.
- Note, however, that any such PS file may not contain instructions that change the global appearance of the document that includes it. E.g., commands like `erasepage`, `stop` or `a4` are not permitted in an EPS file.
- Watch for non-standard EPS files! For instance, Mathematica developed its own "improved" flavor of PS.
- Some of those trouble makers, including Mathematica output, can be cleaned with the `psfix` utility (on Unix systems).

LaTeX and Encapsulated PostScript (cont'd)

- Also, note that the proper inclusion of EPS files into LaTeX requires the use of compatible DVI drivers and previewers.
- Normally, `xdvi`, `dvips`, and Ghostscript/Ghostview do not cause any troubles when handling EPS files included into a LaTeX document.
- On systems that support pipes, the `graphicx` package can also be used to include compressed and non-EPS graphics files.

Basics of TGIF

- `Tgif` is an Xlib-based interactive 2D *drawing tool* that allows the user to draw and manipulate objects under the X Window System.
- `Tgif` is free for non-commercial applications.
- `Tgif` supports the hierarchical construction of drawings, and an easy navigation between sets of drawings.
- It is also a hyper-graphics editor/browser on the WWW.
- `Tgif` is purely based on Xlib. It requires a three-button mouse.
- The source code for `tgif` is freely available on the WWW.
- `Tgif` supports a variety of primitive objects.
- Objects can be grouped together to form a *grouped* object.

Basics of TGIF (cont'd)

- Commands applied to a grouped object are applied to all sub-objects of the group.
- Typically, `tgif` objects are stored in files with an .obj extension (referred to as an *object file*). (So-called "building-block" objects are stored in files with a `.sym` extension (referred to as a *symbol file*).)
- Both types of files are stored in the form of Prolog facts. Prolog code can be written to interpret the drawings!

Tgif Window Layout

Top Window: Displays the current domain and the name of the file `tgif` is looking at. Mouse clicks and key presses have no effect.

Menu-bar Window: This window is directly under the Top Window. Pull-down menus can be activated from it with any mouse button. Key presses have no effect. Some menus are cascading menus.

Message Window: This is directly under the Menu-bar Window and to the right. It displays `tgif` messages.

Canvas Window: This is the drawing area. The effects of the actions of the mouse is determined by the current drawing mode. Holding down the “SHIFT” key and clicking the left mouse on an object which is not currently selected will add the object to the list of already selected objects. The same action applied to an object which is already selected will cause it to be de-selected. Pressing the middle mouse button always generates the main `tgif` pop-up menu.

Tgif Window Layout (cont'd)

Scrollbars: Clicking the left mouse button in the vertical/horizontal scrollbar causes the canvas window to scroll down/right by a small distance; clicking the right mouse button has the reverse effect.

Status Window: This window is below the horizontal scrollbar. It shows what action will be taken if a mouse button is pressed. By default, a right-handed mouse is assumed.

Rulers: They track the mouse location. Mouse clicks and key presses have no effect. When the page reduction/magnification is set at 100%, the markings in the rulers correspond to centimeters when the metric grid is used, and they correspond to inches when the English grid is used.

Tgif Window Layout (cont'd)

Panel Window: This is the window to the left of the Message Window. It contains a collection of icons reflecting the current state of `tgif`.

In top/bottom, left/right order, it displays the page style (portrait or landscape), print/export mode, zoom factor, move and stretch mode (constrained or unconstrained), radius for rounded-corner rectangles, text rotation, page number or row/column, page layout mode (stacked or tiled), horizontal alignment, vertical alignment, font, text size, vertical spacing between lines of text within the same text object, text justification, shape, stretchable or non-stretchable text mode, dash pattern, arrow style, line type (polyline, spline, interpolated spline), line width, fill pattern, pen pattern, color, and transparency.

In addition to displaying the current state of `tgif`, the icons in the Panel Window can also be used to change the current state. Each icon is associated with a particular state variable of `tgif`. Clicking the left mouse button on top of an icon cycles the state variable associated with the icon forward. If there are objects selected in the canvas window, then the action of the mouse will cause the selected objects to change to the newly selected mode.

Objects and Object Transformations

- Primitive objects supported by `tgif` are rectangles, ovals, rounded-corner rectangles, arcs, polylines, polygons, open/closed splines, text, some specific forms of X11 pixmaps, and EPS.
- All objects in `tgif` can be moved, duplicated, deleted, rotated, flipped, and sheared.
- Most commands in `tgif` can either be activated by a pop-up menu or by typing an appropriate non-alphanumeric key.
- All operations that change any object can be undone and then redone.
- Commands such as zoom and scroll are not undoable.
- The size of the undo/redo history buffer can be set in the .Xdefaults file.

Object Attributes

- `Tgif` supports
  - ⋆ 32 fill patterns,
  - ⋆ 32 pen patterns,
  - ⋆ 7 default line widths,
  - ⋆ 4 line styles (plain, head arrow, tail arrow, double arrows) for polylines and open-splines,
  - ⋆ 9 dash patterns,
  - ⋆ 3 types of text justifications,
  - ⋆ 4 text styles (roman, italic, bold, bold-italic),
  - ⋆ 11 default text sizes (8, 10, 12, 14, 18, and 24 for the 75dpi fonts and 11, 14, 17, 20, 25, and 34 for the 100dpi fonts),
  - ⋆ 5 default fonts (Times, Courier, Symbol, New-Century-Schoolbook, Helvetica), and
  - ⋆ 11 default colors.
- Additional line widths, text sizes, etc., can be added in the .Xdefaults file.

File Input/Output

- `Tgif` can generate output in several different formats:
  - ⋆ PS,
  - ⋆ EPS,
  - ⋆ PDF (needs `ps2pdf` from the `ghostscript` package),
  - ⋆ X11 bitmap (XBM), or XPM for color output),
  - ⋆ plain ASCII text.
- X11 bitmap files, certain forms of X11 pixmap files (such as the one generated by `tgif`), and EPS can be imported into `tgif` and can be represented as `tgif` primitive objects.
- Files in other raster formats (e.g, PNG, JPEG, TIFF, etc.) can also be imported into and exported from `tgif` if external tools can be used to convert them into X11 XBM/XPM files.
- By default, `tgif` drawings are formatted for printing on letter-size paper. `Tgif` offers a compile-time flag in order to make DIN A4 the default paper size.
- `Tgif` can capture (portions of) a screen and input it as a `tgif` object.

Xfig

- `Xfig` is an interactive, menu-driven drawing tool which runs under the X Window System (X11R4 or later), on most Unix-compatible platforms.
- It is freeware, and available via anonymous ftp.
- Detailed instructions for using `xfig` can be found by typing `man xfig`. You can get pretty far just playing around.
- It requires a two- or three-button mouse.
- When using a two-button mouse use the `meta` key and the right button at the same time to effect the action of the middle button.
- Note that the behavior of the mouse buttons depends on what tool you currently have selected. You can see what this behavior is in the top right-hand corner of the `xfig` window.

Xfig Command-line Options

- Start it as `xfig [`*options*`] [`*file*`]`.
- `Xfig` offers several command-line options.
- Here comes a brief list of some command-line options; check its manual page for details:
  - ⋆ **-P[ortrait]:** Make it come up in portrait mode. (Landscape mode is default.)
  - ⋆ **-pw[idth] XXX:** Make it come up *XXX* units wide (where units are either *cm* or `in`).
  - ⋆ **-ph[eight] YYY:** Make it come up *YYY* units high.
  - ⋆ **-inc[hes]:** Make inches the unit of choice. (Default.)
  - ⋆ **-me[tric]:** Make centimeters the unit of choice.
  - ⋆ **-startf[ontsize] pointsize:** Set the default font size for text objects. (Default: 12pt.)
  - ⋆ **-sp[ecialtext]:** Start it with the special text mode for text objects (such as LaTeX strings).

Xfig Main Menus

- The following menus are at the top of the `xfig` window:
  - ⋆ File,
  - ⋆ Edit,
  - ⋆ View,
  - ⋆ Help.
- The *Help menu* contains links to `xfig` references, man pages, and a how-to manual in PDF format.
- The *View menu* contains functions for redrawing the canvas, switching between portrait and landscape mode, and for zooming out and zooming in on the canvas.
- The View menu also contains toggle switches for controlling the display of informative entities, e.g., of the info balloons (aka "mouse explainers").

Xfig Main Menus: File Menu

New: Deletes all objects from the canvas and erases the current filename to make a new figure. This operation may be undone by `undo`.

Open: Popup panel to open a FIG file.

Merge: Popup panel to merge one or more FIG files with the current figure.

Save: Save current the figure in the current filename.

Save As: Popup panel to save the current figure in a new filename.

Export: Popup panel to export the current figure to various formats such as PS, EPS, PDF, GIF, JPEG, etc.

Print: Popup panel to print current figure to a PS device.

Exit: Exit from `xfig`. If the figure has been modified and not saved, a popup panel will appear to ask the user if she wants to save the figure first and then quit, quit without saving, or cancel the quit altogether.

Xfig Main Menus: Edit Menu

Undo: Undo the last operation such as object creation, deletion or modification. Multi-level undo is not supported!

Paste Objects: Paste the FIG object previously copied into the `xfig` cut buffer into the current figure. The object will appear on the canvas under or near the mouse where it may then be moved and placed by pressing mouse button 1. If you want to place it where it originally came from press mouse button 2.

This function can be used to copy part of another figure into the figure being edited. The cut buffer can be shared between `xfigs` if a user runs two or more xfig programs at the same time, and it is possible to copy objects between those `xfig` programs. Normally, the ``.xfig'' file in a user's home directory is used as the cut buffer.

Paste Text: Paste text from the X11 cut buffer onto the canvas where a text object has been started. Note that you must already have started a text object by clicking the Text mode and clicking on the canvas where you want the text pasted.

Xfig Main Menus: Edit Menu (cont'd)

Search/Replace: Popup panel to search and/or replace strings in FIG text objects.

Spell Check: Popup panel to check FIG text objects for spelling errors.

Global settings: Popup panel showing global settings such as the HTML browser, spelling checker, mouse tracking in rulers, etc.

Set units: Popup the unit panel to change drawing/scaling units.

Xfig Graphical Objects

- The objects in `xfig` are divided into *primitive objects* and *compound objects*.
- The primitive objects are:
  - ⋆ ARC,
  - ⋆ CIRCLE,
  - ⋆ (CLOSED) SPLINE,
  - ⋆ ELLIPSE,
  - ⋆ POLYLINE,
  - ⋆ POLYGON,
  - ⋆ PICTURE OBJECT (i.e., imported image files),
  - ⋆ RECTANGULAR BOX,
  - ⋆ ARC BOX, and
  - ⋆ TEXT.
- A primitive object can be moved, rotated, flipped vertically or horizontally, scaled, copied, aligned within a compound object, or erased.
- The TEXT primitive may not be flipped. It may be rotated but in the canvas only the markers show the rotation. TEXT *is* rotated on PostScript output!

Xfig Graphical Objects (cont'd)

- `Xfig` supports the use of ISO Latin-1 characters.
- The attributes of any primitive object can be edited using a popup panel. E.g., you can set the precise coordinates of an object manually.
- A *compound object* is composed of primitive objects. It is generated by *grouping* primitive objects together. Compounds may not be nested hierarchically.
- The primitive objects that constitute a compound cannot be individually modified, but they can be manipulated as an entity: a compound can be moved, rotated, flipped vertically or horizontally, scaled, copied or erased.
- A compound that contains any boxes, arc-boxes, ellipses or circles may only be rotated by 90 degrees.
- Regular polygons may be created using a special drawing mode, but `xfig` will actually create a general POLYGON, which may then be modified, e.g., the individual vertices may be moved if desired.

Xfig Indicator Panel

- The indicator panel contains buttons to set certain drawing parameters, such as line thickness, canvas grid, rotation angle etc.
- Not all of these are always visible unless the `showallbuttons` command-line option was invoked or the X11 resource set.
- All of the buttons use the same mouse buttons for setting values.
- Pressing the left mouse button on the indicator will pop up a panel in which either a value may be typed (e.g., for a line thickness) or the mouse may be clicked on one of several buttons (e.g. for grid style or font name).
- Pressing the middle mouse button on an indicator will decrement the value (e.g., for line thickness) or cycle through the options in one direction (e.g., font names).
- Pressing the right mouse button will increment the value or cycle through the options in the other direction.

Xfig Drawing Mode Panel

- Icons in the drawing (end editing) mode panel represent object manipulation functions, modes and other drawing or modification aids.
- Manipulation functions are selected by positioning the cursor over it and clicking the left mouse button.
- The selected icon is highlighted, and a message describing its function appears.
- The drawing mode panel contains buttons used to create the various objects.
- Once a drawing mode is selected, the object is created by moving the mouse to the point on the canvas where the object is to be placed and pressing and releasing the left button.
- For objects which may have more than two points (e.g., a polyline), the left button may be pressed for each successive point, and the middle button must be pressed to finish the object.
- At any time the right button may be pressed to cancel the creation of the object.

Xfig Editing Mode Panel

- When a button in the editing mode panel is pressed, any objects that may be affected by that editing operation will show their corner markers. Only those objects may be affected by the particular edit mode.
- When multiple objects have points in common, e.g., two boxes that touch at one corner, only one object can be selected by clicking on that point.
- To select other objects, hold down the *shift* key while pressing the left mouse button: the markers of one object will be temporarily highlighted. By repeatedly clicking the l eft button while holding down the *shift* key, it is possible to cycle through all candidates for selection at that point.
- To perform the selected action, e.g., deleting one box, click on the point without holding down the *shift* key. The operation will be performed on the highlighted object.
- If the mouse is not clicked near enough to an object marker or if (for whatever reason) `xfig` cannot ”find” the object the user is trying to select, a black square will temporarily appear above the mouse cursor.

Exporting Xfig Graphics

- The `TransFig` package is used when printing or exporting the output from `xfig`.
- The `fig2dev` program from the `TransFig` package is automatically called by `xfig` as a back-end processor to produce various types of output, such as
  - ⋆ LaTeX picture, LaTeX `epic` and `eepic`,
  - ⋆ combined PS/LaTeX,
  - ⋆ PS (PostScript, color PostScript is supported),
  - ⋆ EPS (Encapsulated PostScript,suitable for inclusion in LaTeX),
  - ⋆ PDF (Portable Data Format),
  - ⋆ XBM (X11 Bitmap) and XPM (X11 Pixmap),
  - ⋆ PBM (Portable Bitmap) and PPM (Portable Pixmap),
  - ⋆ MF (METAFONT),
  - ⋆ SLD (AutoDesk Slide Format),
  - ⋆ GIF (Graphic Interchange Format) and PNG (Portable Network Graphics),
  - ⋆ JPEG (Joint Photographic Expert Group), TIFF (Tag Image File Format).
- See the manual page of `fig2dev` for all options.

Importing Xfig Graphics Into LaTeX

- Among the different export options of `xfig`, the ones listed below work well in conjunction with LaTeX:
  - ⋆ LaTeX picture,
  - ⋆ EPS file,
  - ⋆ Combined PS/LaTeX format,
  - ⋆ PDF file (if PDFTeX/PDFLaTeX is to be used).

Importing Xfig Graphics in LaTeX Picture Mode

- Obviously, exporting as a LaTeX picture is the weakest option, as one is bound by the drawing capabilities of LaTeX. However, strings may make full use of all LaTeX commands.

- To include a LaTeX picture `foo.tex`, just use the `input` command of LaTeX: `\input{foo}`.

- In order to scale such a LaTeX picture within a LaTeX document, use the LaTeX command `\scalebox`. E.g., `\scaleboc{2}{\input{foo}}`.

- The scaling factor is just a number: reduction if less than 1, enlargement if larger than 1.

- Note that scaling bitmap fonts may produce ugly results, so try and avoid them!

- The LaTeX command `\resizebox` scales an object (such as a picture) to make it fit a width given. E.g., `\resizebox{5cm}{!}{\input{foo}}` will make `foo` appear 5cm wide, and suitably high. (Using `!` as an argument retains the aspect ratio of the box.)

Importing Xfig Graphics as EPS File

- When exporting as an EPS file, there are no limitations in drawing figures, except that one cannot use LaTeX command strings in this format.
- However all of the many fonts of PS are available when this format is used.
- A file `foo.eps` can be included into a LaTeX document by using the `\includegraphics` command as follows:

  `\includegraphics[`options`]{`foo`}`.

Importing Xfig Graphics as Combined PS/LaTeX File

- Exporting `xfig` data in combined PS/LaTeX format will create a file with a `.pstex_t` extension which you can then include directly into your LaTeX document, and a file with a `.pstex` extension which contains PS part of the figure.
- The `.pstex_t` file automatically calls the `.pstex` file and you do not need to include it explicitly in your LaTeXfile.
- To include a figure `foo.pstex_t`, just use the `input` command of LaTeX: `\input{foo.pstex_t}`.
- When you call `xfig`, make sure to set the `-spec[ialtext]` flag.
- If the `-spec` flag is set, special characters (such as `\`, for example) in a text will not be specially processed but passed to the output "as is" when exporting.
- This may be used to put LaTeX commands in a text. E.g., the string `$\int_0^9 f(x) dx$` would be processed by LaTeX.

Using Xfig to Generate HTML Image Maps

- `Xfig` can be used for generating clickable HTML image maps.
- To make an object clickable, use *comments* on the `edit` panel, and insert data like `HREF="`*url*`"   ALT="`*string*`"`.
- Here, *url* is the URL of the target of the link, and *string* is an alternative string for browsers which will not display images.
- Note that `xfig` cannot use text objects for links.
- Curved objects, such as circles or splines, will be approximated with polygons.

Basics of Ipe

- Ipe is a drawing editor that generates drawings in XML, PDF or EPS format.
- Ipe is particularly geared towards making sophisticated 2D figures that serve as illustrations of geometric concepts and algorithms. It offers most standard features of a drafting package, plus a few “CAD-like” features.
- The Ipe interface allows keyboard shortcuts which (mostly) are equivalent to the shortcuts of Emacs.
- Objects supported include (poly)lines, polygons, splines, splinegons, circles and ellipses, circular arcs, rectangles, and marks. Bitmaps are supported, too.
- Users can provide Ipelets (Ipe plug-ins) to add functionality to Ipe. This way, Ipe can be extended for each task at hand.
- Ipe is written in standard C++ using the STL.
- Ipe is free software.

Ipe’s Main Features

- Entry of text as LaTeX source code. In the display, text is displayed as it will appear in the figure.
- Produces pure Postscript/PDF, including the text.
- It is easy to align objects with respect to each other using various snapping modes.
- The text model is based on Unicode, and has been tested with Korean, Chinese, and Japanese.
- The GUI is implemented using the portable toolkit Qt, and, thus, can be compiled for Unix, Windows, and Mac OS X.
- Ipe 5.0 had gained a reputation of being difficult to compile. Those problems became irrelevant when Ipe 6.0 made its debut.
- Note: Ipe is upwards compatible, but it is not downwards compatible! (That is, Ipe 5.0 cannot input Ipe 6.0 files!)

Inclusion of Ipe Files Into LaTeX

- Ipe 6.0 outputs EPS/PDF documents, which are readily included into LaTeX by the same means as any other EPS/PDF document is included.
- An Ipe 5.0 figure `snapping.ipe` can be included into a LaTeX figure environment as follows:

```
\IpeFit{10.0cm} \Ipe{./drafting/snapping.ipe}
```

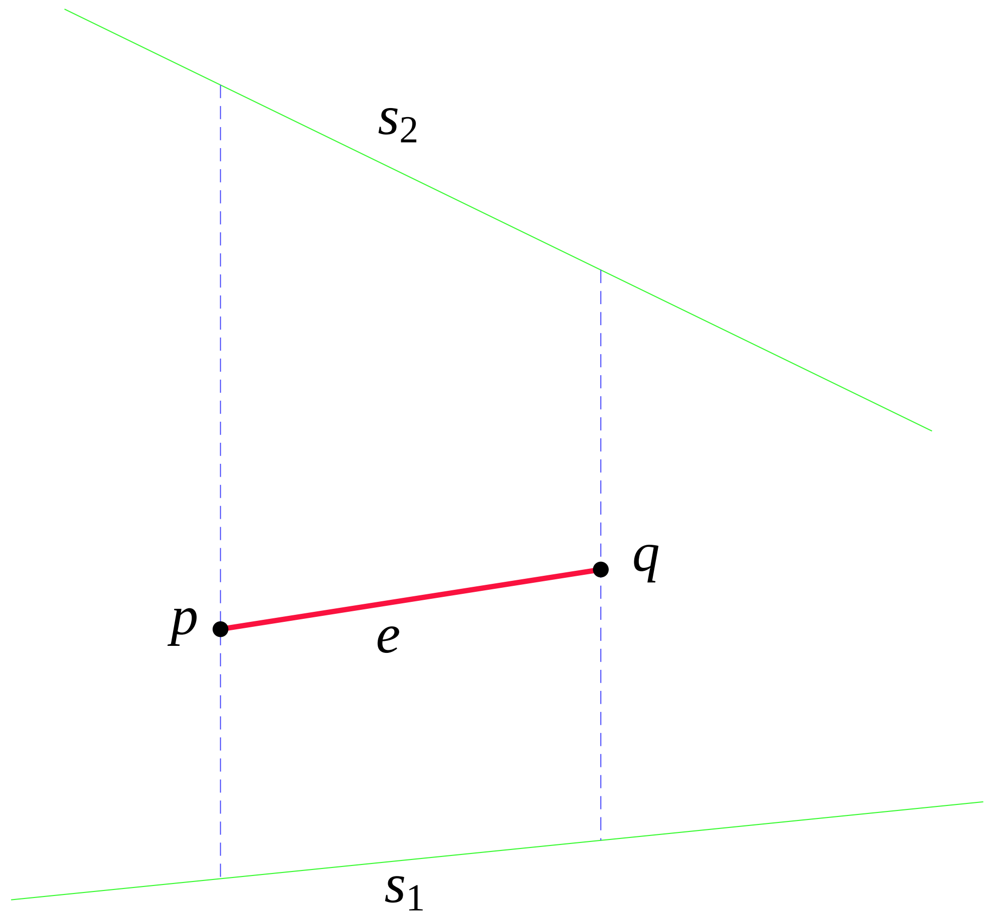
- Note that this requires the inclusion of the package `ipe.sty`.
- An `.ipe` file can be scaled to xx percent of its size by using the command `IpeScale{`xx`}`.

Special Features of Ipe

- Ipe is extendible. One can easily interface personal editing functions, so-called *Ipe extensions* or *Ipe User Macros* (IUMs), with Ipe.
- One of the nicest features of Ipe is the possibility to have the mouse *snap* to other objects. That is, the user can make certain objects in the drawing canvas *magnetic*, which makes it very easy to align an object under construction to other objects.
- If the cursor is too far away from the nearest interesting object then the cursor will not snap. The snapping threshold can be changed in Ipe's configuration window.
- Ipe supports three types of snapping: *grid snapping* (to grid points), *context snapping* (to vertices, boundaries, intersections), and *directional/angular snapping*.

Sample Snapping in Ipe

- Suppose we are given the segments s_1, s_2 and e, with end points p and q, and want to add vertical extensions through p and q between s_1 and s_2.



Sample Snapping in Ipe (cont'd)

- Snapping action:

1. Turn on vertex snapping (`.`), e.g., by pressing F4, and go into polyline mode.
2. Move cursor near p and set the coordinate origin by pressing F1.
3. Turn on directional snapping ($-$) by pressing F5 and angular snapping by pressing F8.
4. Go near s_1 and click left mouse button, then go near s_2 and click right mouse button.
5. Go near q, re-set the coordinate origin by pressing F1, and draw the second vertical line.
6. Turn off the coordinate system by pressing SHIFT-F1.

- Note that pressing F1 at point p and then pressing F2 at point q will set the coordinate origin at p and will align one coordinate axis with the line through p and q.

Dia – A Tool for Generating Diagrams and Flowcharts

- Tool to draw various kinds of diagrams.
- Many functions are similar to xfig or tgif.
- Pro: extensible library of predefined objects.
- Pro: connections (e.g., arrows) follow objects that are moved or resized..
- WWW home page: http://www.gnome.org/projects/dia/.

Basics of Dia

- An object can be moved or scaled by dragging one of the green spots at its bounding box.
- Objects can be connected: possible docking points are marked with a tiny line. E.g., place an arrow on the canvas, move an end point to a docking point. A connected point is drawn red instead of green.
- Most actions have to be initiated via a context menu: select the object with the left mouse button, then use the middle button for the context menu (e.g., to add corners to a polyline).
- Other features can be changed using the right-mouse menu entry Object->Properties. Normally, it can also be accessed using the middle mouse button. In the properties dialog it is possible to change line styles, color, dots, and also properties special to the object (such as class names in UML objects).

Dia's Object Library

- Dia provides predefined objects for various purposes:
  - ⋆ generic symbols like boxes, triangles,. . .
  - ⋆ symbols used to draw electronic circuits,
  - ⋆ symbols for sketching network environments,
  - ⋆ symbols used in flowchart diagrams,
  - ⋆ symbols necessary for UML,
  - ⋆ and many others.
- Objects which are not included in the library are easy to create by using Dia itself to draw them and export the drawing as a *Dia shape file*.

Dia — Input and Output

- Dia uses its own file format which is based on XML.
- Dia allows to export diagrams using many other file formats, but it cannot read most of them.
- Besides the aforementioned Dia shape file format, Dia is able to export to Computer Graphics Metafile, Encapsulated Postscript, SVG, xfig, . . .

Pstoedit

- The utility `pstoedit` translates PS (and PDF) graphics into other vector formats.
- Currently, `pstoedit` can generate the following formats (among many others):
  - ⋆ PDF,
  - ⋆ OBJ (for `tgif`),
  - ⋆ FIG (for `xfig`),
  - ⋆ MP (for METAPOST and TEX/LATEX usage),
  - ⋆ DXF (CAD exchange format),
  - ⋆ gnuplot format.
- Note that you will need a PS interpreter to get `pstoedit` to work.
- Also, your PS interpreter needs to capable of processing PDF if you'd like to use `pstoedit` to convert PDF to other vector formats.

Pstoedit: How Does It Work?

- `Pstoedit` works by redefining the two basic painting operators of PS, `stroke` and `show`. (Bitmaps drawn by the `image` operator are currently only supported by the `xfig` back-end.)
- After redefining these operators, the PS file that needs to be converted is processed by a PS interpreter, e.g., Ghostscript (`gs`).
- The output that is written by the interpreter due to the redefinition of the drawing operators is a sort-of “flat” PS file that contains only simple operations like `moveto`, `lineto`, `show`, etc.
- You can look at this intermediate file using the `-f` debug option.
- This intermediate file is read by end-processing functions of `pstoedit`, and it triggers the drawing functions in the selected back-end driver.

PSfrag for Generating LaTeX Symbols

- While `Tgif` can generate PS output that is suitable for inclusion into a LaTeX document, it cannot generate all the (mathematical) symbols that LaTeX supports.
- PSfrag is a set of LaTeX macros for overlaying PS figures with fragments of LaTeX.
- More precisely, the PSfrag macros allow specific pieces of PS text (so-called “tags”) in a PS figure to be replaced with arbitrary fragments of LaTeX. When the document is latex’ed and dvips’ed, each piece of PS text is replaced by the properly sized, aligned, and rotated LaTeX text.
- In this way, Greek letters, super- and subscripts and mathematical symbols can be used in PS files with a typography that is consistent with the rest of the LaTeX document.

PSfrag and LaTeX

- For each tag word in the EPS file, one adds a command to the LaTeX document to specify how this tag is to replaced, as follows:

`\psfrag{`tag`}[`posn`][`psposn`][`scale`][`angle`]{`LaTeX text`}`

- All data given in brackets `[]` is optional and is used to specify the exact position and orientation of the LaTeX text with respect to the bounding box of the tag string. (See the manual for details.)

- Any text that is not mentioned in a `\psfrag` command will not be replaced; hence, PS and LaTeX text can be freely mixed.

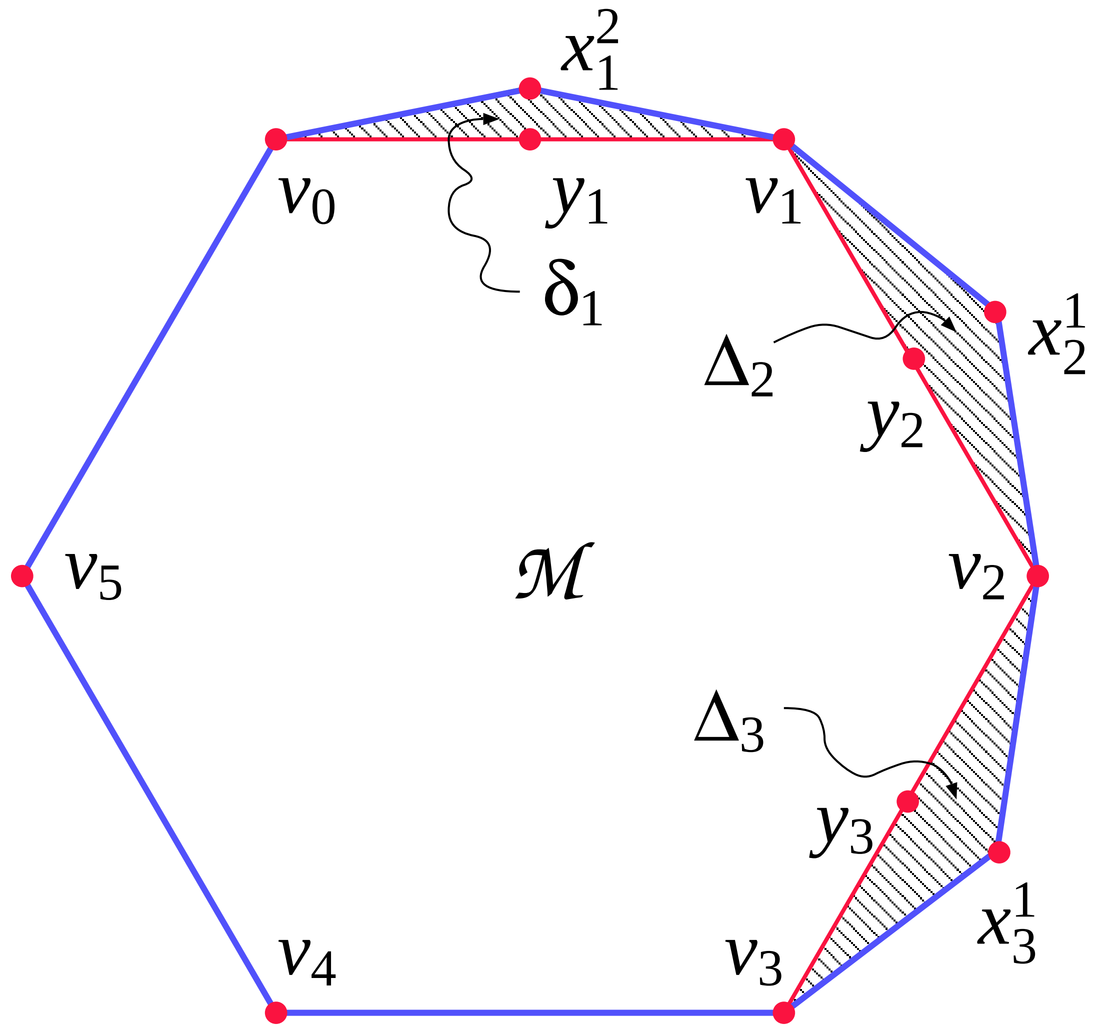
- Most DVI previewers (such as `xdvi`) are incapable of displaying the replaced text correctly.

- Note that `psfrag` relies on the PostScript `\special` command.

Sample PSfrag Code

```
\psfrag{v0}{{\Large $v_0$}}
\psfrag{v1}{{\Large $v_1$}}
\psfrag{v2}{{\Large $v_2$}}
\psfrag{v3}{{\Large $v_3$}}
\psfrag{v4}{{\Large $v_4$}}
\psfrag{v5}{{\Large $v_5$}}
\psfrag{x12}{{\Large $x_1^2$}}
\psfrag{x21}{{\Large $x_2^1$}}
\psfrag{x31}{{\Large $x_3^1$}}
\psfrag{y1}{{\Large $y_1$}}
\psfrag{y2}{{\Large $y_2$}}
\psfrag{y3}{{\Large $y_3$}}
\psfrag{delta1}{{\Large $\delta_1$}}
\psfrag{Delta1}{{\Large $\Delta_1$}}
\psfrag{Delta2}{{\Large $\Delta_2$}}
\psfrag{Delta3}{{\Large $\Delta_3$}}
\psfrag{hex}{{\Large $\cal{M}$}}
\includegraphics[width=12cm]{grasp_hex}
```

Sample PSfrag/TGIF Picture



PSfrag and LaTeX(cont'd)

- PSfrag requires a recent version of LaTeX.
- A compatible DVI-to-PS driver is required, too. PSfrag works best with `dvips`, the DVI-to-PS driver from Radical Eye Software.
- Note that the file `psfrag.sty` has to be installed in a location searched by the LaTeX search path for macros. For `kpathsea`-based systems such as `teTeX`, this path is determined by the `TEXINPUTS` environment variable.
- Also, the DVI-to-PS driver has to be able to find the file `psfrag.pro`.
- A `\psfrag` replacement will remain in effect until its surrounding environment is exited.
- Thus, one can define global `\psfrag` commands which will apply to every figure of a LaTeX file, or one can define `\psfrag` commands inside an environment (e.g., a `figure` environment) which will apply to only one EPS file.

xwd – Making X11 Screen Dumps

- `Xwd` is a utility for storing X11 window images in a specially formatted dump file.
- This file can then be read by various other X11 utilities (such as `xv` and `xpr`) for redisplay, printing, editing, formatting, archiving, image processing, etc.
- The target window is selected by clicking the mouse pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.
- `Xwd` is part of the standard X11 distribution.
- Sample command sequence for dumping a window into a file `foo.xwd`, and for converting it into a PostScript file:

```
xwd > foo.xwd
xpr -device ps -portrait  -psfig foo.xwd > foo.ps
```

- Several packages for drafting and image manipulation also support capturing part or all of an X11 display.

xv – Display and Conversion of Graphics Files

- `Xv` is a utility for the interactive display of images, under the X11 Window System.
- It can display images in a variety of formats, including GIF, JPEG, TIFF, XBM, XPM, XWD, possibly PS.
- `Xv` can also generate output in most of these formats. In particular, it is a simple-to-use interactive conversion program between different graphics formats.
- `Xv` provides several functions for editing an image. It can translate, rotate, and clip images, perform an edge enhancement or blur the image, improve the contrast, etc.
- Note that `xv` is shareware, and a license is required for anything but personal use.
- Other packages like `gimp` or `ImageMagick` provide more sophisticated means for displaying and manipulating images.

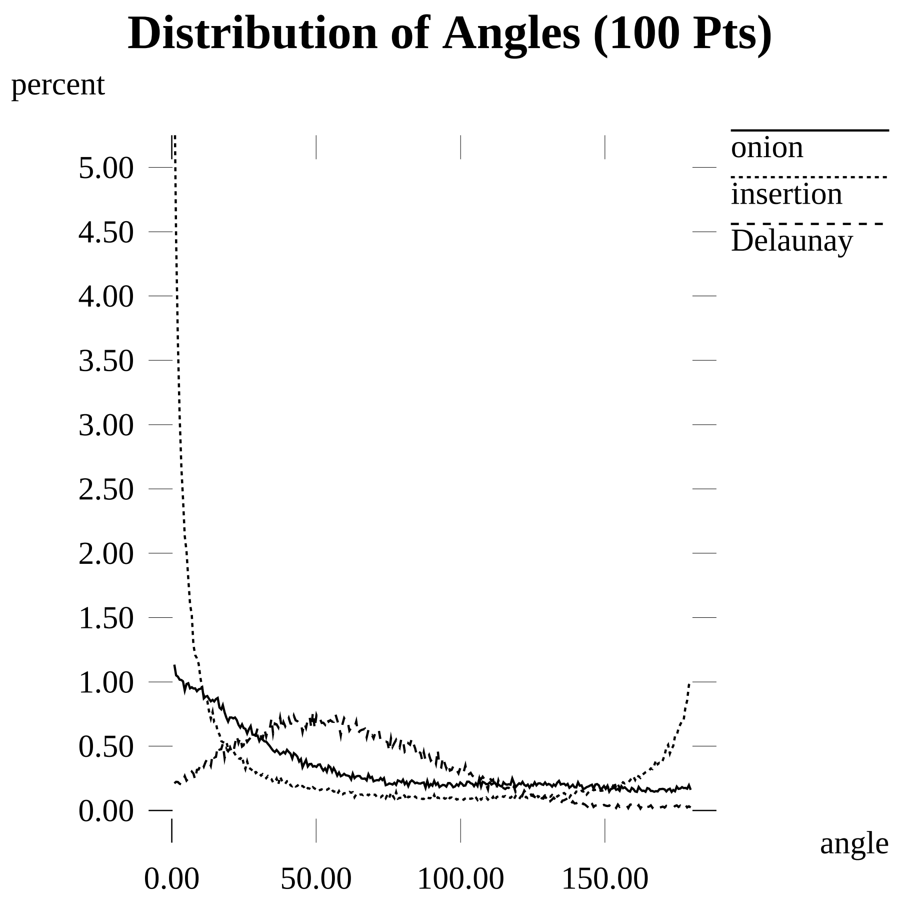
Xgraph – A Simple Plotting Program

- Xgraph plots simple graphs, displaying them in a window that it creates.
- In its simplest form it is invoked as `xgraph foo`, where `foo` is a data file containing coordinates of points, one per line, with the x- and y-coordinates separated by spaces. These points will be drawn connected by lines, with axes that are automatically scaled to the range of the x- and y-coordinates.
- Xgraph can plot several graphs superimposed by specifying more than one datafile, or by putting several datasets into one file, separated by blank lines.
- Various options allow one to plot points instead of or as well as lines, to plot on a log scale, and to change titles, etc.
- One can zoom into a plot interactively.
- One can convert plots to PS, and prepare them for inclusion into a LaTeX document.

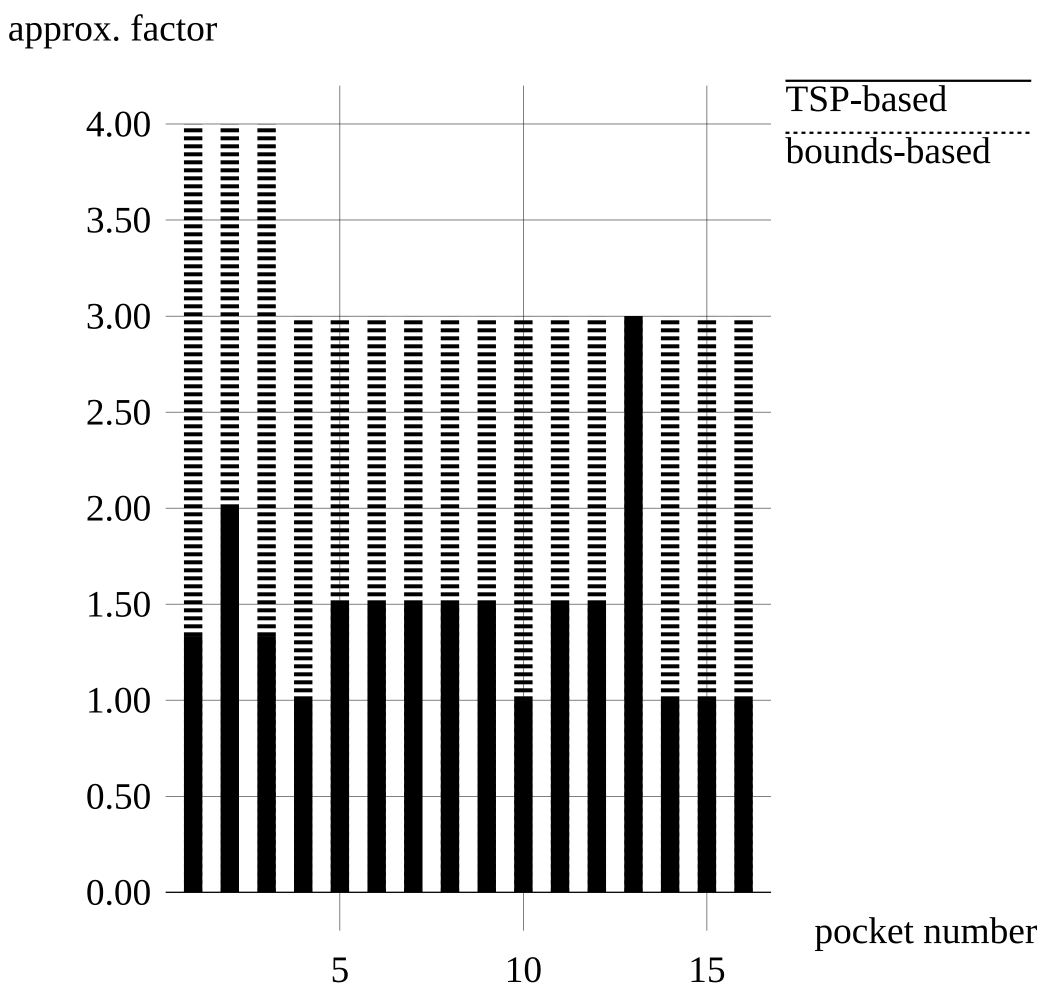
Xgraph (cont'd)

- Xgraph supports the construction of multiple bar graphs.
- A '93 revised version of xgraph supports a crude animation of the data set, and provides an interface for incorporating xgraph output into `tgif`.
- The main advantage of xgraph is that it is convenient to use for simple tasks.
- Its main disadvantage, besides its somewhat limited functionality, is that it was developed several years ago on a Vax VMS system under X11R3. Compiling it under X11R7 may constitute a serious challenge, unless appropriate patches are used.

Sample Xgraph Plot



Sample Xgraph Plot (cont'd)



The Gnuplot Plotting Program

- Gnuplot is a command-line driven interactive plotting tool.
- It can plot 2D and 3D graphs, and can handle plots of built-in or user-defined functions.
- Input for LaTeX can be generated by instructing gnuplot to output its plot in the EPS format: `set term postscript eps` tells gnuplot to generate the plot in EPS format.
- The command `set term pslatex` instructs gnuplot to generate a LaTeX picture of the plot, i.e., a LaTeX .tex file. The advantage of using the LaTeX picture environment is that all LaTeX commands can be used for making labels, etc. Its main disadvantage is that any LaTeX picture is limited by the small number of slopes that lines can be drawn with. Thus, the appearance of a plot is likely to be less than satisfying.

Gnuplot (cont'd)

- If the `epic.sty` and `eepic.sty` style files for the extended picture environment of LaTeX are available then LaTeX can handle more general pictures, and gnuplot can be instructed to set its terminal type to `set term eepic`. Note, however, that this currently does not support dashed lines in the plots!

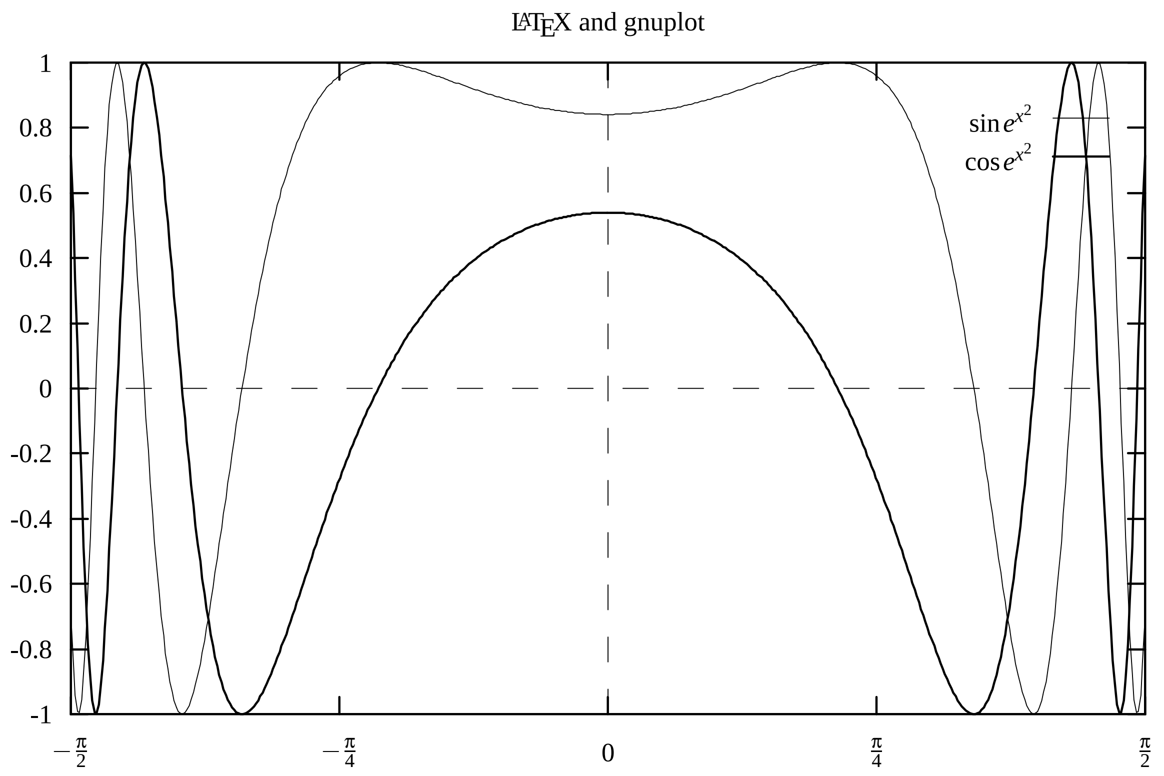
- Another alternative for incorporating LaTeX text into a gnuplot plot is to `set term eps`, and to use `\psfrag` commands for replacing tag strings by actual LaTeX text.

Sample Gnuplot Plot

- The following plot was generated by means of gnuplot, and included into LaTeX by using the `eepic` environment:

```
f(x)=sin(exp(x**2))
g(x)=cos(exp(x**2))
set samples 500
set term eepic
set output 'gnuplot.tex'
set title '\LaTeX\ and gnuplot'
set xrange [-pi/2:pi/2]
set xtics ('$-\frac{\pi}{2}$' -pi/2, \
           '$-\frac{\pi}{4}$' -pi/4, '0' 0, \
           '$\frac{\pi}{4}$' pi/4, \
           '$\frac{\pi}{2}$' pi/2)
plot f(x) title '$\sin e^{x^2}$', \
     g(x) title '$\cos e^{x^2}$'
```

Sample Gnuplot Plot (cont'd)



DTP and WWW

- Portable Data Format (PDF),
- LaTeX-to-PDF Conversion,
- Generating PDF Slides:
  - ⋆ PPower4,
  - ⋆ LaTeX Beamer Class,
- LaTeX-to-HTML Conversion,
- MathML.

Basics of PDF

- PDF (Portable Document Format) is a cross-platform high-resolution universal document exchange format created by Adobe.
- PDF is a condensed version of the page description language PostScript, with added Hypertext and multi-media functionality.
- Virtually any PS file can be *distilled* into a PDF file.
- Typically, a PDF file is much smaller than its corresponding PS file.
- PDF files are not HTML web pages.
- PDF files are a fast way to publish existing documents on the WWW without having to recreate them in HTML and without compromising the printed image quality.

Basics of PDF (cont'd)

- The creator of a `PDF` document can block a user from copying of text or graphics, making changes, and printing the document. (This feature needs encryption.)
- `PDF` is a pure data format. Contrary to PostScript, it does not require complex operations to be performed prior to output.
- `PDF` is an object-oriented data format; individual `PDF` objects/pages can easily be extracted from a `PDF` file.
- De-facto industrial standard; ISO is currently working on an appropriate ISO standard based on the `PDF` file format.
- `PDF` files can be viewed by a variety of tools that are freely available, e.g., Adobe's Acrobat Reader (`acroread`), Aladdin's Ghostscript (`gs`), or `xpdf`.

Adobe's Commercial PDF Tools

Acrobat Distiller converts a PS file into a PDF file. It first applies a "normalization" to the PS file in order to free it from non-standard features and flavors. Supposedly, it can handle virtually any flavor of PostScript.

Acrobat Exchange supports a minimal amount of editing and formatting for final customization. Its input is a distilled PDF file. In Acrobat Exchange, one can supply hypertext links to other portions of the document (e.g., to a table of contents) or to other PDF files or WWW sites. Sounds and Quicktime movies can be included, too.

Acrobat Catalog features extensive indexing and searching capabilities. It can handle hundreds of PDF files, and produces a search data structures that can be searched very efficiently.

Acrobat Reader, which is provided free of charge by Adobe, lets you display and print PDF files.

Pros of PDF

- PDF supports “byte-serving”: a PDF browser does not need to have an entire document in its local memory in order to display its first page.
- PDF supports a non-linear flow of reading.
- PDF provides efficient search and retrieval capabilities.
- Based on PostScript, which is today’s standard page description language.
- Little rework needed in order to provide CD/WWW distributable documents.
- Better resolution than with HTML.
- Better document security than with HTML.
- PDF files can be magnified up to 800% without loss of clarity in text or graphics.
- Fonts can be embedded in a PDF file.

Pros of PDF (cont'd)

- JPEG images can be included into a PDF file very compactly.
- PDF has a built-in per-page compression.
- PDF is truly platform-independent, with support for reading PDF documents being available on all major platforms and operating systems (Unix/Linux, Windows, Mac-OS).
- MS Word can be instructed to output a document in PDF format, which, likely, is the simplest approach to making Word documents readable for Unix users without loss of visual quality.

Cons of PDF

- Adobe does not really strive to help third-party developers.
- Support by non-commercial tools still is rather limited.
- While a PDF file generally is smaller than a PS file, a gzipped PS file generally is much smaller than a gzipped PDF file.
- Unless care is taken, the embedding of fonts may cause huge PDF files.
- The conversion to PDF often is anything but straightforward.

Creating a PDF File

- Adobe Acrobat: allows to “web capture” web pages and convert them to PDF documents.
- Acrobat Distiller: converts PS to PDF.
- Acrobat PDFWriter: converts GDI or QickDraw graphics to PDF.
- Acrobat Capture: batch conversion of text or image files to PDF.
- Framemaker can generate PDF directly.
- Commercial third-party software; check the WWW.
- Public-domain software for converting PS files to PDF files, and for translating LaTeX documents into PDF.
- As a meta remark, be warned that this still is a rapidly changing field, and any “how-to-do-it” advice that is accurate as of today may be out of date tomorrow.

LaTeX-to-PDF Conversion Using Acrobat Distiller

- If Acrobat Distiller is available, then the best (and most painless) work-flow seems to be TEX $\longrightarrow$DVI $\longrightarrow$PS $\longrightarrow$PDF, where
  - ⋆ the DVI file is generated by any standard LaTeX system,
  - ⋆ the DVI-to-PS conversion is done by means of Rokicki's `dvips`, and
  - ⋆ the resulting PS file is converted to PDF using Acrobat Distiller.
- For the PS-to-PDF conversion you'll have to make sure to use PostScript Type 1 fonts rather than bit-map fonts (pk fonts).
- PostScript Type 1 fonts are outline fonts that look good at arbitrary resolutions. (Bitmap fonts are classified as Type 3 fonts by Adobe.)
- Since most PostScript Type 1 fonts that you may want to use are not known to Acrobat, you will have to get `dvips` to embed the fonts into the PostScript output.
- Of course, it is wise to embed only those glyphs that are actually used.

LaTeX-to-PDF Conversion Using Acrobat Distiller

- You may want to change Distiller's maximum subsetting threshold to 99%. (Its default maximum subsetting threshold is only 10%! In other words, if more than 10% of the characters in a font are included in the document, the entire font is embedded in the document.)

- Since `dvips` performs some resolution-dependent fine tuning, it is recommend to set the `dvips` resolution to a very high value. (E.g., set to 8000 dpi.)

- Be sure to instruct Distiller to use compression. Otherwise, the embedded fonts will result in huge PDF files.

PostScript-to-PDF Conversion Using PD-Software

- The PostScript utility `ps2pdf` converts a PS file to a PDF file.
- It is based on Aladdin Ghostscript (`gs` 6.0).
- Currently, `ps2pdf` does a reasonable job on filled/stroked graphics, and on text in the 14 built-in PDF fonts in the intersection of Windows and ISO Latin-1 encodings.
- It converts all other text in the PS file to bitmaps in the PDF file, which are resolution-dependent. (It does only write the bitmap for each character once per page, though, and only on pages where the character is actually used.)
- It does not compress the output at all, except for character bitmaps.
- The PERL script `epstopdf` does a similar job for EPS files, and it also relies on Ghostscript.

LaTeX-to-PDF Conversion Using PDFTeX

- The PDFTeX package is an extension of LaTeX/TeX that can create PDF directly from TEX source files and improve/enhance the result of TEX typesetting with the help of PDF.

- When PDF output is not selected, `pdftex` produces normal DVI output, otherwise it produces PDF output that looks (virtually) identical to the DVI output.

- The typical use of the PDFTeX-package is with pre-generated formats for which PDF output has been enabled.

- The `pdftex` command uses the equivalent of the plain TeX format, and the `pdflatex` command uses the equivalent of the LaTeX format.

- Currently, `pdftex`/`pdflatex` generate PDF Level 1.4 output.

Macro Packages Supported by PDFTEX

- Currently, all main-stream macro packages offer PDFTEX support in some way.
- When using such a package, it makes sense to turn on this support in the appropriate way, otherwise one cannot be sure whether things will be set up correctly.
- For instance, the `hyperref` package has substantial support for PDFTEX, and provides access to most of its features. In the simplest case, the user merely needs to load `hyperref` with the `pdftex` option, and all cross-references will be converted to PDF hypertext links. PDF output is automatically selected, compression is turned on, and the page size is set up correctly. Bookmarks are created to match the table of contents.
- Similarly, the LATEX packages `graphics` and `color` have options for PDFTEX, which allow the use of the standard commands for color, text rotation, and graphics inclusion.
- PDFTEX supports the inclusion of pictures in PNG, JPEG, TIFF and PDF format.

PDFTEX and PostScript Figures

- When producing DVI output, part of the job is delegated to the DVI postprocessor, either by directly providing this program with commands, or by means of `\specials`.
- Since PDFTEX directly produces the final format, it has to do everything itself, from handling color, graphics, hyperlink support, font inclusion, up to page imposition and page manipulation.
- Thus, `\specials` make no sense at all.
- This means that (Encapsulated) PostScript figures cannot be included directly into a document that is to be processed by PDFTEX, and commands like `\psfrag` will not work.

PDFTEX and PostScript Figures (cont'd)

- EPS files can be converted to PDF by Ghostscript, Acrobat Distiller, or by the PERL script `epstopdf`.
- To get the right *BoundingBox* from an EPS file, before converting to PDF, it is necessary to transform the EPS file so that the start point is at the (0,0) coordinate and the page size is set exactly corresponding to the BoundingBox.
- Do not use `epsfig` in your LATEX file, or any other old image inclusion package; use `graphics` or `graphicx`.
- If you want to be able to build PS and PDF files from the same source, leave off the file extensions from the image filenames in the `\includegraphics` calls.
- In any case, make sure to use Type 1 fonts!
- If the PS-to-PDF conversion of an EPS figure creates poor results, convert the image to JPEG or PNG, and use that for the PDF version.

Converting PostScript Figures to PDF

- No big deal if the PS figure does not contain any text: use `ps2pdf` or `epstopdf`.
- The situation gets more tricky once the use of fonts is required. In particular, the use of `\psfrag` or IPE 5.0 files constitutes a problem.
- According to my experience, it seems best to include the PS figure in a LaTeX "container" file, with appropriate instructions for using the correct fonts, to convert this file to a PS file, and to run `epstopdf` on it.
- Note that the use of `\psfrag` sometimes results in an odd problem: you might end up with a visible remark in your PDF file telling you that `\psfrag` replacements were used.
- So far, my best solution to this problem has been to edit the PS file, and to manually delete the offending lines, and to proceed as normal afterwards . . .

Sample PS-to-PDF LaTeX Container File

```
\documentclass[12pt]{article}
\usepackage{pslatex}
\usepackage[T1]{fontenc}
\usepackage{graphicx,psfrag}
\usepackage{epic,eepic}
\usepackage{ipe}

\begin{document}

\pagestyle{empty}

\begin{center}
     \psfrag{hex}{{\LARGE $\cal{M}$}}
     \includegraphics{grasp_hex}
\end{center}

\thispagestyle{empty}

\end{document}
```

Sample Shell Script for PS-to-PDF Conversion

```
#!/bin/sh

latex convert
dvips -E  -Pcmz -Pamz -o $1 convert
epstopdf $1
```

Checking for Type 1 Fonts in a PDF File

- Acrobat Reader (`acroread`) allows you to list all fonts used in your document. Click on
  `File > Document Info > Fonts ...`
- If you use `dvips`, the Type 3 bitmapped fonts may also be named T1, T2, etc., while Type 1 fonts will display their correct names, such as `cmr10`, `cmbx10`, and so on.
- Fonts that have a six-character prefix added to the font name (such as `KICOLA+cmr7`) are Type 1 fonts that have been subsetted.
- A visual inspection (by zooming into a PDF document) typically will also quickly tell you whether Type 3 fonts are used.

Generating PDF Slides

- PPower4,
- LaTeX Beamer Class.

PPower4

- `PPower4` is a Java package used to post-process LaTeX files formatted with `pdflatex` such that PDF pages can be built step by step during a presentation.

- When using `PPower4`, the work-flow is TEX →PDF →PDF, where

  ⋆ the first PDF file is generated using `pdflatex` (or `pdftex`),
  ⋆ the second PDF file is generated by running the post-processor, `ppower4`, on the first PDF file.

- Acrobat Reader, `acroread` can be used for giving the actual presentation. (Note that `acroread` can be run in full-screen mode!)

- `PPower4` offers (among other features)

  ⋆ partial page builds (by means of the command `\pause`), ▮
  ⋆ colored backgrounds, ▮
  ⋆ and fancy page transitions.

PDF Slide Shows (cont'd)

- These slides were prepared using `pdflatex` and `ppower4`, and are displayed using `acroread`.
- See a demo for a demonstration and explanation of more advanced presentation techniques.
- Since `PPower4` slides are based on standard LaTeX code, virtually all features of LaTeX can be used to create sophisticated technical slides.
- With `PPower4`, the time-consuming re-work of LaTeX publications into PowerPoint slides is obsolete.
- Furthermore, `PPower4` only needs `acroread` to give the actual presentation — and `acroread` is available for every major platform, including Unix/Linux and Windows.
- Best of all, `PPower4` is free under a GNU GPL.

LaTeX Beamer Class

- The LaTeX beamer class allows to create slides directly within LaTeX, with no need to resort to other software packages such as `PPower4`.
- It can be used with `pdflatex`, but also with `dvips`.
- Professional layouts and sophisticated overlays can be achieved.
- See a demo for a demonstration of some of the features of the LaTeX beamer class.

Basics of LaTeX2HTML

- LaTeX2HTML is a conversion tool that allows documents written in LaTeX to be translated into HTML (Hypertext Mark-up Language).

- Its development started in 1993. For several years the development efforts were coordinated and spear-headed by Nikos Drakos (U. of Leeds).

- The current project leader is Ross Moore (U. of Sidney). Work on LaTeX2HTML has turned into a truly international effort.

- LaTeX2HTML is freely available. It is covered by a license and warranty agreement that is similar in terms and spirits to GNU's license agreement.

- LaTeX2HTML consists entirely of scripts to run other pieces of software (e.g., Perl, LaTeX, Ghostscript, netpbm) and standard Unix utilities (e.g. cp, rm, make, ln) as well as the operating system shell.

Basics of LaTeX2HTML (cont'd)

- Note that HTML is an evolving standard, with different versions supporting different features. Currently, the most advanced version of HTML is HTML 4.01, which is a subversion of HTML 4.

- XHTML 1.0 is W3C's recommendation for the latest version of HTML, following on from earlier work on HTML. With a wealth of features, XHTML 1.0 is a reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML.

- LaTeX2HTML replicates the basic structure of a LaTeX document as a set of interconnected HTML files which can be explored using automatically generated navigation panels.

- It is applied to a file *foo.tex* by typing `latex2html foo.tex`. This will create a directory called *foo*, which will contain the generated HTML files plus (likely) some images.

Basics of LaTeX2HTML (cont'd)

- All cross-references, citations, footnotes, and the table-of-contents are translated into hypertext links. (Same for the lists of figures and tables.)
- Formatting instructions that have equivalent HTML "tags" – e.g., lists, paragraphs, quotes – are converted as appropriate.
- All formatting instructions that cannot be directly expressed in HTML are converted to images that are placed automatically at appropriate positions in the HTML document.
- In particular, most mathematical equations are converted to images!

List of Features of LaTeX2HTML

The following list, which was taken from the LaTeX2HTML Manual, provides a survey of the main features of LaTeX2HTML:

- It breaks up a document into one or more components as specified by the user.
- It provides optional, customizable iconic navigation panels on every page which contain links to other parts of the document, or other documents.
- It handles in-lined equations ($\sum_{i=1}^{n} x_i = \int_0^1 f$), handles equation alignment ($A_{B_{C+D}}$), right-justified numbered equations, tables, figures, and any arbitrary environment.
- Either the complete environment or sub-parts thereof are passed to LaTeX for conversion to images, which are then either included in the document or are made available through hypertext links.

List of Features of LaTeX2HTML (cont'd)

- Figures and tables can be arbitrarily scaled and oriented, and shown either as in-lined images or “thumbnail” sketches. Alternatively, their contents are displayed within a table constructed using the `<TABLE>` tags of HTML 4.
- Theorem-like environments are supported, along with automatic numbering and counter dependencies.
- It can produce output suitable for browsers that support in-lined images or character-based browsers (as specified by the user). In particular the TeX or LaTeX code for mathematical expressions and formulae will be displayed in character-based browsers, such as lynx.
- Colored text and/or background is fully supported, as is the ability to use an image to create a tiled backdrop.
- It handles definitions of new commands, environments and counters even when these are defined in external files for input.

List of Features of LaTeX2HTML (cont'd)

- It handles footnotes, tables of contents, lists of figures and tables, bibliographies and can generate an index. By including hyper-links between index entries, simple navigation aids can be built into the index, for easy browsing.

- It automatically translates cross-references and citations into hyper-links, and extends the LaTeX cross-referencing mechanism to work not just within a document but between documents which may reside in remote locations.

- It translates LaTeX accent and special character commands (e.g., ø, ö, ©, $) to the equivalent ISO Latin-1 or Unicode character set, else an image can be created.

- It recognizes hypertext links (to multi-media resources or arbitrary Internet services such as sound, video, ftp, http, news) and links which invoke arbitrary program scripts – all expressed as LaTeX commands.

List of Features of LaTeX2HTML (cont'd)

- It recognizes conditional text which is intended only for the hypertext version, or only for the paper (.DVI) version – see the `\hyperref` command, and the environments `htmlonly` and `latexonly`.

- It provides the environment `rawhtml` for including raw HTML in a LaTeX document (e.g. in order to specify interactive forms).

- It can deal sensibly with virtually all of the concepts and commands described in the LaTeX Book where there is a meaningful interpretation appropriate to an HTML document. Also many other LaTeX constructions are handled, including many described in the LaTeX Companion and LaTeX Graphics Companion (e.g., XY-pic).

List of Features of LaTeX2HTML (cont'd)

- It can be configured to translate equations either as GIF images or as HTML mark-up (as browsers become available which are suitable for the task), or by making images of subparts of equations, as required.
- It links symbolic references across document segments which have been independently processed – see the `\segment` command and the segmentation example in the LaTeX2HTML Manual. (Note: Standard Unix Makefiles can be used for complicated segmentations.)
- It will try to translate any document with embedded LaTeX commands, irrespective of whether it is complete or syntactically legal.
- Dozens of command-line options can be used to customize the layout of the HTML pages. (Type `latex2html -h` in order to get a listing of all options.)
- Frequently used command-line options can be stored in a personal initialization file, `.latex2html-init`.

MathML

- HTML cannot handle complicated formulas but has to resort to ugly an workaround based on images.
- MathML is an XML-derived standard to publish formulas.
- Based on ideas from many existing products, e.g., LaTeX.
- WWW home page: http://www.w3.org/Math/.

Sample MathML Code

- The code fragment

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msup>
    <msqrt>
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </msqrt>
    <mn>27</mn>
  </msup>
</math>
```

should be rendered similar to: $\sqrt{a+b}^{27}$

Generating MathML Code

- One can produce MathML with a variety of tools:
  - ⋆ your favorite editor (the do-it-yourself approach),
  - ⋆ starting with Version 4, Mathematica can handle MathML,
  - ⋆ TEX4ht: http://www.cse.ohio-state.edu/~gurari/TeX4ht/.
  - ⋆ many more, see http://www.w3.org/Math/iandi/.

Symbolical Mathematics

- Basics of Mathematica,
- Lists, Vectors, and Matrices in Mathematica,
- Symbolic Computation in Mathematica,
- Elementary Calculus with Mathematica,
- Symbolic Solution of Equations with Mathematica,
- Numerical Math in Mathematica,
- Mathematica and Graphics,
- Import and Export of Mathematica Data,
- Symbolic Computation – Caveats.

Basics of Mathematica

- Mathematica is a software package – "*computer algebra system*" (CAS) – for use in mathematical applications that require symbolic computation.
- It was designed by Stephen Wolfram, starting in late 1986.
- Mathematica is an excellent tool for *symbolic* and *numerical* computation.
- It also provides graphics capabilities, and can produce two- and three-dimensional graphs.
- It can be used as a scientific calculator, but can also perform operations on functions, manipulate algebraic formulae, and do calculus.
- It is widely used in science and engineering.
- It can be used for designing "interactive" documents.

Basics of Mathematica (cont'd)

- Mathematica is based on its own *high-level programming language*.
- It can be interfaced with external programs: it can invoke external programs, and it can be invoked by external programs.
- It provides an interface to LaTeX, and it can output data in a variety of graphics formats such as encapsulated PostScript, GIF, etc.
- Mathematica is a commercial product, and it is available for a variety of platforms.
- Other Packages for Symbolic Computation: Axiom, Derive, Macsyma, Maple, Reduce, . . .

User Interface

- Mathematica can run in an *ASCII terminal mode*, or it can display *notebooks* as an *X11 client*.
- To run Mathematica from an ASCII terminal, enter `math`. To run Mathematica as an X11 client, enter `mathematica`.
- Both display variants are front ends to Mathematica's *kernel*, which takes care of the actual computations.
- Note: one has to press `SHIFT RET` in order to invoke a computation after keying in a command!
- Mathematica can handle very large numbers. If a number is so large that it cannot be displayed on one line, Mathematica places a backslash (\) at the end of the line to show that the number continues on the next line.
- Calculations that take too long can be aborted by typing `ALT ,` (in a notebook environment), or `CTRL c` (in terminal mode).

User Interface (cont'd)

- The primary structural principle of Mathematica notebooks is their division into *cells*.
- The *scope of cells* is indicated by vertical brackets on the right-hand side of the Mathematica window.
- *Text cells* are marked by brackets that have a second small horizontal bar on top. Text cells cannot be interpreted by Mathematica.
- In order to interprete a cell, the cursor needs to be placed anywhere within the cell. Pressing the left mouse key will generate a a blinking cursor. Then press `SHIFT RET`, and Mathematica will evaluate the cell.
- Consecutive cells in Mathematica can be *grouped* together.
- Double-clicking on a bracket that has a peculiar lower arrow will *open* the corresponding cell. Double-clicking on a bracket which does not have a lower arrow will *close* this cell.

Mathematica Palettes

- Mathematica offers several *palettes* for facilitating the input of characters and symbols.
- Functions can often be entered via *templates*, and `TAB` can be used for moving among *placeholders*.
- Many symbols can also be entered directly. E.g., `ESC p ESC` will generate π.

Basic Math in Mathematica

- Mathematica labels its input and output as `In[`n`]` and `Out[`n`]`, where n is a counter that is stepped for every input/output pair.

- One can refer to the last output generated as `%`. A string of k percent signs refers to the k-th previous output, and `%`n refers to the output numbered n.

`In[1] :=` $3+5$
`Out[1] =` 8

`In[2] :=` $\% + 3^2$
`Out[2] =` 17

`In[3] :=` $\% - 2 * \%1$
`Out[3] =` 1

Basic Math in Mathematica (cont'd)

- Mathematica distinguishes between two types of values, *exact* and *approximate*. Exact values may either be (a) integers or fractions, in which case Mathematica keeps as many digits as necessary to express the value exactly, or (b) symbolic names for constants such as e, π, $\sqrt{2}$, for which Mathematica knows how to find as many digits as necessary in any computation.

- Approximate values are most typically numeric expressions containing a decimal point.

- An exact value of x can be converted to an n-digit approximate value by using the function `N[`x,n`]`.

```
In[4]  :=  N[% + π, 20]
Out[4]  =  4.1415926535897932385
```

- A semicolon (`;`) after an expression instructs Mathematica not to print the result.

Basic Math in Mathematica (cont'd)

- The *arithmetic operators* of Mathematica have standard calculator form ("$+$", "$-$", "$*$", "$/$", and "^") and have standard mathematical precedence. For instance, multiplication and division are executed before addition and subtraction.
- Mathematica accepts some non-standard input forms for arithmetic. E.g., the $*$ may be omitted so that the multiplication is implied.
- Note, however, that spaces are required if the $*$ is omitted: $x\ 2$ is different from $x2$!
- The expression $x =$ *value* assigns *value* to x.
- Note that this is a *permanent assignment*, and Mathematica will substitute *value* in all subsequent occurrences of x, until or unless explicitly told otherwise.

Basic Math in Mathematica (cont'd)

- Any value assigned to x can be removed via $x =$. or `Clear[`x`]`.

```
In[5] := x = 4
Out[5] = 4
```

```
In[6] := 3 * √x
Out[6] = 6
```

```
In[7] := Clear[x]
In[8] := 3 * √x
Out[8] = 3 √x
```

- In order to avoid mistakes, it is advisable to clear assignments as soon as they are no longer needed.

- By convention, all names of built-in objects of Mathematica start with upper-case letters. Note that names can never start with a number.

Mathematica as a Scientific Calculator

```
In[9]  :=  123456789123456789 + 987654321987654321
Out[9]  =  1111111111111111110
```

```
In[10]  :=  40!
Out[10]  =  815915283247897734345611269596115894272000000000
```

```
In[11]  :=  x = N[π]
Out[11]  =  3.14159
```

```
In[12]  :=  Sin[π]
Out[12]  =  0
```

```
In[13]  :=  Sin[x]
```

Out[13] = $1.22465\,10^{-16}$

Mathematica as a Scientific Calculator (cont'd)

In[14] := $1.0\ /\ (\mathrm{Sin}[x]\ \hat{}\ 1000)$
Out[14] = $9.75322579165\ 10^{15911}$

In[15] := $x = 1\ /3 + 1\ /5$
Out[15] = $\frac{8}{15}$

In[16] := $(15\ x)\ /\ 8$
Out[16] = 1
In[17] := $\mathrm{Clear}[x]$

In[18] := $\sqrt[6]{64}$
Out[18] = 2

In[19] := $\pi\hat{}2\ /\ 6.$
Out[19] = 1.64493

Lists as Mathematica Objects

- Many Mathematica objects are based on *lists*. Also, most operations can be applied to whole lists, which get treated as single objects.

In[20] := $x = \{2, 3, 4\}$
Out[20] = $\{2,3,4\}$

In[21] := x^2
Out[21] = $\{4,9,16\}$

- The commands Part[x,i] and x[[i]] extract the i-th element of the list x.

In[22] := x
Out[22] = $\{2,3,4\}$

In[23] := $x[[2]] = 10$
Out[23] = 10

Lists as Mathematica Objects (cont'd)

- The commands `Part[`x,i`]` and x`[[`i`]]` extract the i-th element of the list x.

```
In[24]  :=  x
Out[24]  =  {2,10,4}


In[25]  :=  x[[1]] + x[[2]] + x[[3]]
Out[25]  =  16
```

- Typically, parentheses `()` are used for grouping, brackets `[]` enclose function arguments, curly braces `{}` delimite lists, and double brackets `[[]]` are used for indexing.

Vectors and Matrices

- Vectors and matrices are lists and lists of lists, respectively.

In[26] := $m[x\_] \; := \; \{\{\mathrm{Cos}[x],\ -\mathrm{Sin}[x],\ 0\},\ \{\mathrm{Sin}[x],\ \mathrm{Cos}[x],\ 0\},\ \{0,\ 0,\ 1\}\}$

In[27] := $\mathrm{MatrixForm}[m[x]]$

Out[27] = $\begin{pmatrix} \mathrm{Cos}[x] & -\mathrm{Sin}[x] & 0 \\ \mathrm{Sin}[x] & \mathrm{Cos}[x] & 0 \\ 0 & 0 & 1 \end{pmatrix}$

In[28] := $\mathrm{Simplify}[\mathrm{Det}[m[x]]]$

Out[28] = 1

In[29] := $\mathrm{Transpose}[m[x]]$

Out[29] = $\{\{\mathrm{Cos}[x],\mathrm{Sin}[x],0\},\{-\mathrm{Sin}[x],\mathrm{Cos}[x],0\},\{0,0,1\}\}$

Vectors and Matrices (cont'd)

- Vectors and matrices are lists and lists of lists, respectively.

In[30] := $v = \{1, 0, 1\}$
Out[30] = $\{1,0,1\}$

In[31] := $\mathrm{Dimensions}[m[x]]$
Out[31] = $\{3,3\}$

In[32] := $w = m[\pi/2] \,.\, v$
Out[32] = $\{0,1,1\}$

In[33] := $\mathrm{Cross}[v, w]$
Out[33] = $\{-1,-1,1\}$

In[34] := $v.\, w$
Out[34] = 1

Symbolic Computation in Mathematica

- `Expand` and `Factor` can be used for transforming algebraic expressions.

```
In[35]  :=  1 + x^2 - 2x
Out[35]  =  1 - 2 x + x^2

In[36]  :=  % * (2 + x)
Out[36]  =  (2 + x) (1 - 2 x + x^2)

In[37]  :=  Expand[%]
Out[37]  =  2 - 3 x + x^3

In[38]  :=  Factor[%]
Out[38]  =  (-1 + x)^2 (2 + x)

In[39]  :=  % /. {x -> 1}
Out[39]  =  0
```

Symbolic Computation in Mathematica (cont'd)

- `Expand` and `Factor` can be used for transforming algebraic expressions.

`In[40] :=` $\%\% \;/.\; \{x -> \{a + 2\}\}$
`Out[40] =` $\left\{(1+a)^2\,(4+a)\right\}$

`In[41] :=` $\%\%\% * x/\,((x - 1)\hat{}3 * (x + 1))$

`Out[41] =` $\frac{x\;(2+x)}{(-1+x)\;(1+x)}$

`In[42] :=` $\mathrm{Expand}[\%]$

`Out[42] =` $\frac{2\;x}{(-1+x)\;(1+x)}+\frac{x^2}{(-1+x)\;(1+x)}$

`In[43] :=` $\mathrm{ExpandAll}[\%]$

`Out[43] =` $\frac{2\;x}{-1+x^2}+\frac{x^2}{-1+x^2}$

Simplifying Algebraic Terms with Mathematica

- Getting expressions into a simple form sometimes is an art, and may require a bit of experimenting with `Simplify` and similar commands.

`In[44] :=` $\mathrm{Simplify}[\%]$

`Out[44] =` $\frac{x\ (2+x)}{-1+x^2}$

`In[45] :=` $\%\ *\ (y\ +\ 1)\ /\ (y\ -1)$

`Out[45] =` $\frac{x\ (2+x)\ (1+y)}{(-1+x^2)\ (-1+y)}$

`In[46] :=` $\mathrm{ExpandAll}[\%]$

`Out[46] =` $\frac{2\ x}{1-x^2-y+x^2\ y}+\frac{x^2}{1-x^2-y+x^2\ y}+\frac{2\ x\ y}{1-x^2-y+x^2\ y}+\frac{x^2\ y}{1-x^2-y+x^2\ y}$

Simplifying Algebraic Terms with Mathematica (cont'd)

- Getting expressions into a simple form sometimes is an art, and may require a bit of experimenting with `Simplify` and similar commands.

In[47] := $\% * (1 - x\hat{}2)\,(1 - y)$

Out[47] = $$\left(\left(\frac{2\ x}{1-x^2-y+x^2\ y}+\frac{x^2}{1-x^2-y+x^2\ y}+\frac{2\ x\ y}{1-x^2-y+x^2\ y}+\frac{x^2\ y}{1-x^2-y+x^2\ y}\right)\right.$$
$$\left.(1-x^2)\,(1-y)\right)$$

In[48] := $\mathrm{Simplify}[\%]$
Out[48] = $x\,(2+x)\,(1+y)$

In[49] := $\mathrm{Expand}[\%]$
Out[49] = $2\,x+x^2+2\,x\,y+x^2\,y$

Simplifying Algebraic Terms with Mathematica (cont'd)

- Getting expressions into a simple form sometimes is an art, and may require a bit of experimenting with `Simplify` and similar commands.

In[50] := $\text{FactorTerms}[\%,\, y]$
Out[50] = $\left(2\,x+x^2\right)\,(1+y)$

In[51] := $\text{Collect}[\text{Expand}[\%],\, y]]$
Out[51] = $2\,x+x^2+\left(2\,x+x^2\right)\,y$

Elementary Calculus with Mathematica

- Mathematica can handle differentiation and integration symbolically.

In[52] := $D[x\,(1\ +\ x\hat{}4),\ x]$
Out[52] = $1+5\,x^4$

In[53] := $D[2x\ +\ x\hat{}2\ +\ (2\,x\ +\ x\hat{}2)\,y,\ y]$
Out[53] = $2\,x+x^2$

In[54] := $\mathrm{Integrate}[\%\ /\ (x\ +1),\ x]$
Out[54] = $x+\frac{x^2}{2}-\log[1+x]$

In[55] := $D[\%,\ x]$
Out[55] = $1+x-\frac{1}{1+x}$

Elementary Calculus with Mathematica (cont'd)

- Mathematica can handle differentiation and integration symbolically.

```
In[56]  :=
```
$\mathrm{Factor}[\%]$

```
Out[56]  =
```
$\frac{x\ (2+x)}{1+x}$

```
In[57]  :=
```
$\mathrm{D}[f[x]\ /\ x,\ x]$

```
Out[57]  =
```
$-\frac{f[x]}{x^2}+\frac{f'[x]}{x}$

```
In[58]  :=
```
$\mathrm{Integrate}[\%,\ x]$

```
Out[58]  =
```
$\frac{f[x]}{x}$

```
In[59]  :=
```
$\mathrm{D}[x\hat{\ }y,\ x]$

```
Out[59]  =
```
$x^{-1+y}\ y$

Elementary Calculus with Mathematica (cont'd)

- Of course, an integral need not always exist. Still, one may be able to get a numerical approximation of a corresponding definite integral.

```
In[60]  :=  % /. {y -> x}
Out[60]  =  x^x

In[61]  :=  %
Out[61]  =  x^x

In[62]  :=  Integrate[%, x]
Out[62]  =  ∫ x^x dx

In[63]  :=  Integrate[%%, {x, 0, 1}]
Out[63]  =  ∫_0^1 x^x dx

In[64]  :=  N[%]
Out[64]  =  0.783431
```

Elementary Calculus with Mathematica (cont'd)

- Mathematica can handle differentiation and integration symbolically.

In[65] := $\mathrm{D}[\%\%\%,\ x]$
Out[65] = x^x

- Mathematica can also handle sums and products.

In[66] := $\mathrm{D}[x\ \mathrm{Sum}[1\ /\ 2\hat{\ }i,\ \{i,\ 0,\ \infty\}],\ x]$
Out[66] = 2

In[67] := $\mathrm{Sum}[\mathrm{Product}[x+i,\ \{i,\ 0,\ j\}],\ \{j,\ 0,\ 3\}]$
Out[67] = $x+x\,(1+x)+x\,(1+x)\,(2+x)+x\,(1+x)\,(2+x)\,(3+x)$

In[68] := $\mathrm{Expand}[\%]$
Out[68] = $10\,x+15\,x^2+7\,x^3+x^4$

Elementary Calculus with Mathematica (cont'd)

- Mathematica can handle limits.

In[69] := $\mathrm{Sin}[x]\ /\ x$

Out[69] = $\frac{\mathrm{Sin}[x]}{x}$

In[70] := $\%\ /.\ x\ {-}{>}\ 0$

Out[70] = *Indeterminate*

In[71] := $\mathrm{Limit}[\%\%,\ x{-}{>}\ 0]$

Out[71] = 1

- And it can handle ordinary differential equations.

In[72] := $\mathrm{DSolve}[\{\ y'[x]\ ==\ a\ y[x]\ +\ 1,\ y[0]\ ==\ 0\},\ y[x],\ x]$

Out[72] = $\{\{y[x] \rightarrow \frac{-1+e^{a\ x}}{a}\}\}$

- Mathematica can also handle multi-dimensional calculus if Mathematica's package `VectorAnalysis` is loaded: `<<Calculus`VectorAnalysis``.

Symbolic Solutions for Equations

- Mathematica provides the function `Solve` for computing symbolic solutions for equations.

In[73] := $\text{Solve}[a\,x\hat{}2 + b\,x + c == 0,\ x]$

Out[73] = $\left\{\left\{x \to \frac{-b-\sqrt{b^2-4\ a\ c}}{2\ a}\right\}, \left\{x \to \frac{-b+\sqrt{b^2-4\ a\ c}}{2\ a}\right\}\right\}$

In[74] := $\% \,/.\, \{a -> 2,\ b -> 3,\ c -> 1\,/\,2\}$

Out[74] = $\left\{\left\{x \to \frac{1}{4}\left(-3-\sqrt{5}\right)\right\}, \left\{x \to \frac{1}{4}\left(-3+\sqrt{5}\right)\right\}\right\}$

In[75] := $x\,/.\,\%$

Out[75] = $\left\{\frac{1}{4}\left(-3-\sqrt{5}\right), \frac{1}{4}\left(-3+\sqrt{5}\right)\right\}$

In[76] := $\%[[1]] * 4$

Out[76] = $-3-\sqrt{5}$

Symbolic Solutions for Equations (cont'd)

- Mathematica provides the `Solve` function for computing symbolic solutions for equations.

`In[77] :=` $\mathrm{Solve}[\{x - y == 2,\ x + y == 0\},\ \{x,\ y\}]$
`Out[77] =` $\{\{x \to 1, y \to -1\}\}$

`In[78] :=` $\mathrm{Eliminate}[\{x - y == 2,\ x + y == 0\},\ y]$
`Out[78] =` $x == 1$

`In[79] :=` $\mathrm{Solve}[\mathrm{Sin}[x]\hat{}2 == a,\ x]$
`Out[79] =` $\left\{\left\{x \to -\mathrm{ArcSin}\left[\sqrt{a}\right]\right\}, \left\{x \to \mathrm{ArcSin}\left[\sqrt{a}\right]\right\}\right\}$

Numerical Mathematics in Mathematica

- Mathematica provides functions for computing numerical approximations of sums, products, and integrals.

```
In[80]  :=  Sum[1 / i^2, {i, 1, ∞}]
Out[80]  =  π^2/6
```

$$\frac{\pi^2}{6}$$

```
In[81]  :=  N[%]
Out[81]  =  1.64493
```

```
In[82]  :=  NSum[1 / i^2, {i, 1, ∞}]
Out[82]  =  1.64493
```

```
In[83]  :=  NIntegrate[ Sin[xy], {x, 0, 1}, {y, 0, x}]
Out[83]  =  0.119906
```

Numerical Mathematics in Mathematica (cont'd)

- It can also solve a (system of) polynomial equation(s) numerically, or search for an approximate solution of an arbitrary equation.

In[84] := $\text{Solve}\left[\, x\hat{}3 \;-\; \sqrt{\pi}x\hat{}2 \;==\; 0,\; x\right]$
Out[84] = $\{\{x \to 0\}, \{x \to 0\}, \{x \to \sqrt{\pi}\}\}$

In[85] := $\text{NSolve}\left[\, x\hat{}3 \;-\; \sqrt{\pi}x\hat{}2 \;==\; 0,\; x\right]$
Out[85] = $\{\{x \to 0.\}, \{x \to 0.\}, \{x \to 1.77245\}\}$

In[86] := $\text{FindRoot}[\,\text{Sin}[x] \;==\; x, \{x,\; 0.001\}]$
Out[86] = $\{x \to 0.\}$

Defining Functions in Mathematica

- Mathematica lets one define functions that can then be used similar to built-in functions.

In[87] := $\mathrm{Expand}[\,\mathrm{Product}[\,x + i,\ \{i\ ,\ 1,\ 3\}]]$
Out[87] = $6+11\,x+6\,x^2+x^3$

In[88] := $\mathrm{exprod}[n\_]\ :=\ \mathrm{Expand}[\,\mathrm{Product}[\,x + i,\ \{i\ ,\ 1,\ n\}]]$
In[89] := $\mathrm{exprod}[3]$
Out[89] = $6+11\,x+6\,x^2+x^3$

In[90] := $\mathrm{D}[\mathrm{exprod}[3],\ x]$
Out[90] = $11+12\,x+3\,x^2$

In[91] := $\mathrm{cex}[n\_,\ i\_]\ :=\ (\,t\ =\ \mathrm{exprod}[n];\ \mathrm{Coefficient}[t,\ x\hat{}i]\,)$
In[92] := $\mathrm{cex}[3,\ 2]$
Out[92] = 6

Defining Functions in Mathematica (cont'd)

- Mathematica lets one define functions that can then be used similar to built-in functions.

In[93] := $\text{Clear}[\textit{cex}]$
In[94] := t
Out[94] = $6+11\,x+6\,x^2+x^3$

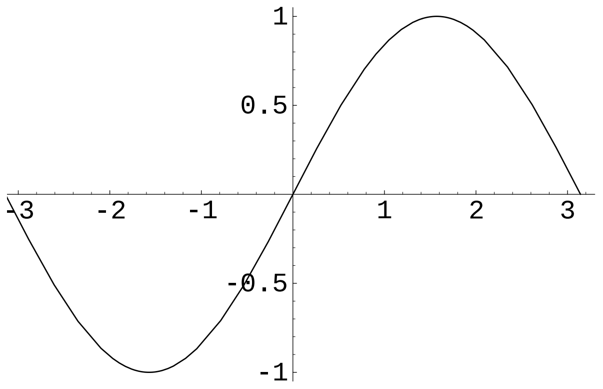
In[95] := $\text{Clear}[t]$
In[96] := $\text{cex}[n\_, i\_] := \text{Module}[\{t\}, t = \text{exprod}[n]; \text{Coefficient}[t, x\hat{\ }i]]$
In[97] := $\text{cex}[3,\ 2]$
Out[97] = 6

In[98] := t
Out[98] = t

2D Graphics in Mathematica

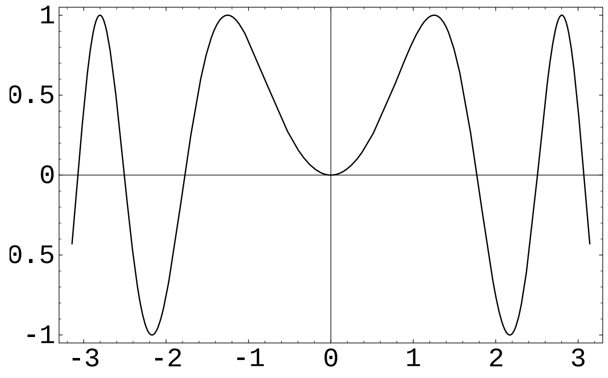
- `Plot` offers many features for drawing 2D graphs.

`In[99] :=` $p1 = \text{Plot}[\,\text{Sin}[x],\ \{x,\ -\pi,\ \pi\}]$

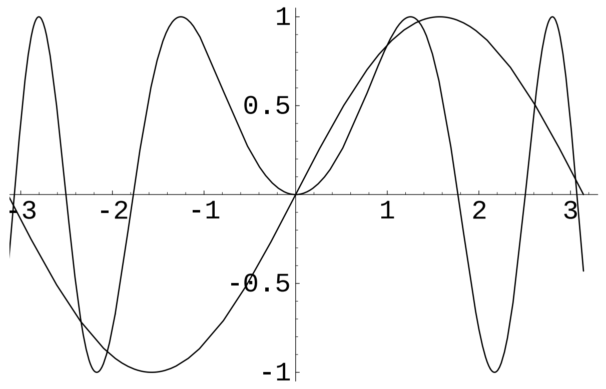


2D Graphics in Mathematica (cont'd)

- `In[100] :=` $p2 = \mathrm{Plot}[\,\mathrm{Sin}[x\hat{}2],\ \{x,\ -\pi,\ \pi\},$ Frame −> True]



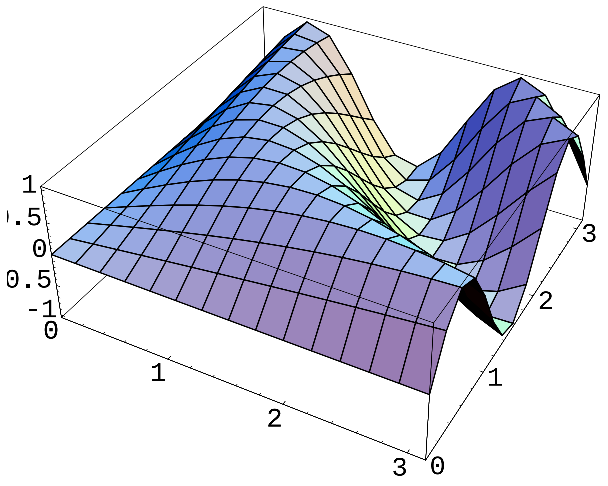
`In[101] :=` $p3 = \mathrm{Show}[p1, p2]$



3D Graphics in Mathematica

- Mathematica can also handle 3D plots.

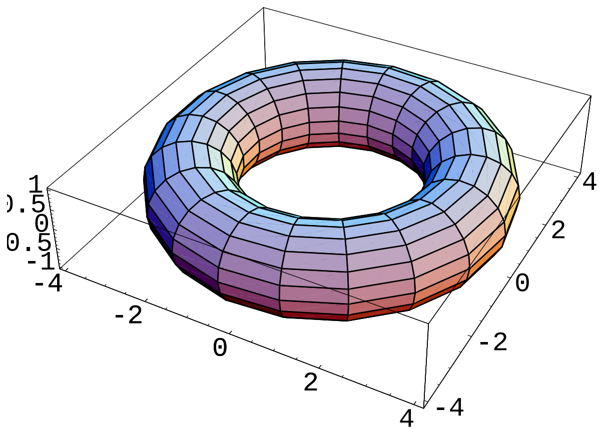
```
In[102] :=  Plot3D[ Sin[x y], {x, 0, π}, {y, 0, π}]
```



3D Graphics in Mathematica (cont'd)

- Mathematica can also handle 3D plots.

In[103] := torus = ParametricPlot3D[$\{\mathrm{Cos}[t]\,(3+\mathrm{Cos}[u]),\ \mathrm{Sin}[t]\,(3+\mathrm{Cos}[u]),\ \mathrm{Sin}[u]\},\ \{t,0,2\pi\},\ \{u,0,2\pi\}]$



Exporting Mathematica Output

- Mathematica can export expressions in C or Fortran format. (C macros are defined in Mathematica's file `mdefs.h`.)

In[104] := $t = (x\hat{}2 - 1) \,/\, \sqrt{x - 1}$

Out[104] = $\frac{-1+x^2}{\sqrt{-1+x}}$

In[105] := $\mathrm{CForm}[t]$

```
Out[105] = (-1 + Power(x,2))/Sqrt(-1 + x)
```

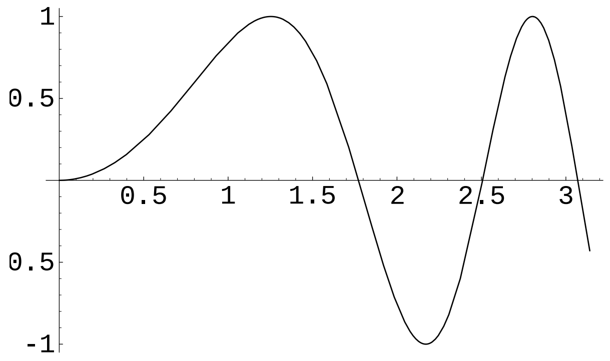
In[106] := $\mathrm{FortranForm}[t]$

```
Out[106] = (-1 + x**2)/Sqrt(-1 + x)
```

Exporting Mathematica Output (cont'd)

- Mathematica can also export a plot as a graphics file. Supported formats include, among others, EPS, PDF, GIF, TIFF, PBM.

In[107] := Plot[Sin [x^2], {x, 0, π}]



In[108] := Display["*foo.eps*", %, "*EPS*"]

- (Portions of) Mathematica notebooks can also be printed as PostScript files. See Mathematica's `print` menu.

Interfacing Mathematica with Other Programs

- Mathematica can export expressions in TEX-format.

In[109] := t

Out[109] = $\frac{-1+x^2}{\sqrt{-1+x}}$

In[110] := $\mathrm{TeXForm}[t]$

```
Out[110] = \frac{-1 + x^2}{\sqrt{-1 + x}}
```

- Also, Mathematica can export a notebook (or portions thereof) as a LATEX-file. This LATEX-file contains macros defined in Mathematica's style file `notebook.sty`. (This is the way all the Mathematica expressions of this document were generated.) See the `TeXSave` command for details.

- Similarly, the `HTMLSave` command saves a notebook in HTML form.

Interfacing Mathematica with Other Programs

- Personal experience tells me that the LaTeX-output generated by Mathematica needs a bit of manual polishing in order for LaTeX to digest it, and to format it neatly.

- Bi-directional communication between Mathematica and an application program is supported by the *MathLink* standard. See the manual for details.

- Mathematica can be instructed to generate output suitable for display by Geomview. (The file `OOGL.m` is provided by Geomview.)

```
In[111] :=  << OOGL.m
In[112] :=  WriteOOGL["m_torus.off", torus]
```

- One can also use Geomview directly for displaying Mathematica graphics. The `Geomview` command invokes Geomview and sends the graphics to Geomview as an OOGL object. (For some reason, this does not work in our environment!)

Symbolic Computation – Caveats

- Consider the class of terms generated from one variable x, constants for the rationals, π, and the function symbols $+, *, \sin,$ abs. Caviness et alii proved that the simplification problem with respect to functional equivalence is undecidable for this class of terms.

- Similarly, Risch proved that the problem of integration in finite terms is undecidable for transcendental functions. Risch also described the first complete integration algorithm for elementary transcendental functions.

- Several important algorithms of computer algebra have an exponential complexity, e.g., Collins' cylindrical algebraic decomposition for quantifier elimination.

- The bit complexity may grow substantially during a computation, thus potentially requiring a large main memory. In particular, the bit complexity of intermediate results may be significantly larger than the complexity of the input and the output. Some algorithms are well known to be memory hoggers – consult textbooks prior to waiting for hours/days just in order to see the system crash due to lack of memory.

Graphics and Visualization

- Basics of Geomview,
- Manipulation and Appearance of Geomview Objects,
- Geomview I/O,
- Geomview and External Applications.

Basics of Geomview

- Geomview is an interactive program for *viewing and manipulating 3D geometric objects*.
- Geomview was written by staff members of the Geometry Center (UMN).
- Unfortunately, in an attempt to save money, the US administration scrapped the Geometry Center in 1998, and development efforts for Geomview have come to a near stand-still.
- Geomview is in a mature and stable state, though! And it is still used widely.
- Geomview is free software, released under a GNU license.
- It runs on SGI workstations, and on a variety of systems using generic OpenGL or X11 graphics and a Motif interface.

Basics of Geomview (cont'd)

- The simplest way to use Geomview is as a *standalone viewer* to see and manipulate objects. It can display objects described in a variety of file formats.
- One can also use Geomview to handle the display of data coming from another program (*external module*) that is running simultaneously. As the other program changes the data, the Geomview image reflects the changes.
- Geomview allows multiple independently controllable objects and cameras. It provides interactive control for motion, appearances (including lighting, shading, and materials), picking on an object, edge or vertex level, snapshots, and adding or deleting objects.
- Objects can be manipulated through direct mouse manipulation, control panels, and keyboard shortcuts.

Basics of Geomview (cont'd)

- Geomview supports the following simple data types: polyhedra with shared vertices, quadrilaterals, rectangular meshes, vectors, and Bézier surface patches of arbitrary degree including rational patches.
- Object hierarchies can be constructed with lists of objects and instances of object(s) transformed by one or many 4×4 matrices.
- Geomview can also display 3D graphics output from Mathematica and Maple.
- Geomview supports hyperbolic and spherical geometry.

Object Manipulations

- Objects can be selected by clicking at the name of the object in the `Targets` browser of the `Main` panel. If `world` is selected, then any motion/transformation is applied to all objects currently drawn.
- The object selected is called the *target object*.
- Geomview lets you manipulate objects with the mouse. There are six different mouse motion modes: Rotate, Translate, Cam Fly, Cam Zoom, Geom Scale, and Cam Orbit. The `Tools` panel has a button for each of these modes; to switch modes, click on the corresponding button.
- Most of the motion modes have *inertia*, which means that if one lets go of the button while moving the mouse, the motion will continue.
- Generally, the left mouse button controls motion in the screen plane, while the middle mouse controls motion along or around the forward direction.

Object Manipulations (cont'd)

- Pressing the shift key while dragging with left or middle mouse buttons in most motion modes gives slow-speed motions, useful for fine control of object placements.

- The `Center` button undoes the target object's transformation, moving it back to its home position, which is where it was when it was originally loaded into Geomview.

- Geomview uses the *glass sphere model* for mouse-based rotations. Think of the object as being inside an invisible sphere, and regard the mouse cursor as a gripper outside the sphere. When one presses the left mouse button, the gripper grabs the sphere; when one releases the left mouse button, the gripper releases the sphere.

- Moving the mouse while holding the button down causes the sphere (and hence the object) to move in the same direction as the mouse.

Object Manipulations (cont'd)

- Specifically, in `Rotate` mode the axis of rotation passes through the origin of the center object, is parallel to the camera view plane, and is perpendicular to the direction of motion of the mouse. When the center is ”target”, this means that the target object rotates about its own origin.
- Press the middle mouse button in order to rotate the target object about an axis perpendicular to the view plane.
- One can pick any point on an object (not just its origin) as the center of motion by holding down the shift key while clicking the right mouse button; this chooses a point of interest.
- In order to translate the target object, hold the left mouse button down (after selecting the `Translate` mode). The middle mouse button translates the target along an axis perpendicular to the view plane.
- `Cam Fly` is a crude flight simulator that lets one fly around the scene. It works by moving the camera.

Object Manipulations (cont'd)

- `Cam Orbit` mode lets one rotate the current camera around the current center.
- `Cam Zoom` lets one change the current camera's field of view with the mouse.
- `Geom Scale` mode lets one enlarge or shrink an object.
- The `Stop` button causes all motion to stop.
- The `Look At` button causes the current camera to be moved to a position such that it is looking at the target object, and such that the target object more or less fills the window.
- The `Reset` button stops all motion and causes all objects to move back to their home positions.

Modifying the Appearance of Objects

- Geomview uses a hierarchy of appearances to control the way things look. An *appearance* is a specification of information about how something should be drawn.
- There is an appearance associated with ”World”, which serves as the parent of each individual object’s appearance. Also, there is a global ”base” appearance, which is the parent of the World appearance.
- Appearances work in a hierarchical manner: if a certain appearance property, for example a face color, is not specified in a particular object’s appearance, that object is drawn using that property from the parent appearance. If both the parent and the child appearance specify a property, the child’s setting takes precedence unless the parent appearance is set to override.
- The `Appearance` panel controls various things about the way Geomview draws objects. For instance, the `[ae] Edges` button allows to toggle between having the object displayed with or without edges.

Modifying the Appearance of Objects (cont’d)

- The `Appearance` panel also lets one select colors (in RGB or HSV) and shading information (constant, flat, smooth).
- The `Materials` panel controls material properties such as the degree of opacity, diffuse and specular reflection, and ambient light.
- The `Lighting` panel controls the number, position, and color of the light sources used in shading.
- The `Cameras` panel controls certain aspects of the target camera (such as its field of view). The use of multiple cameras is supported.
- The *Geomview command language* (gcl) provides complete control of all appearance data, including data that cannot be changed via the panels.

I/O Control

- The `Save` panel offers several possibilities for storing Geomview objects and other information in files.
- One can store gcl commands, geometric data, input data for RenderMan, SGI snapshots (on SGIs), PPM (software) snapshots, snapshots in PostScript format, and data for restoring all windows and panels in a subsequent session of Geomview.
- Commands in gcl format, which uses the syntax of lisp, can be entered via the `Commands` panel.
- Most panel interaction can be replaced by commands that have keyboard shortcuts. For instance, the keyboard shortcut for switching to `Rotate` mode is `r`.
- Some keyboard shortcuts consist of more than one key. In these cases one types the keys one after the other, with no `RET` afterwards. For instance, `g1ae` toggles the edge drawing for object (“geom”) `g1`.

OOGL Files

- Geomview reads objects in the format of the *Object Oriented Graphics Library* (OOGL).
- Examples for many OOGL objects can be found in Geomview's `data/geom` directory.
- Most OOGL files are are free-format ASCII. (Binary formats are also defined for several objects types.)
- Typical OOGL objects begin with a key word designating the object type, possibly with modifiers indicating the presence of additional data (such as color).
- Most key words are case sensitive.

OOGL Files (cont'd)

- When OOGL objects are input, the OOGL library uses the file suffix to guess at the file type.
- Geomview supports inhomogeneous and homogeneous coordinates.
- Transformation matrices are given in a 4×4 row-vector representation, for multiplication on the right of vectors. That is, a row vector p (of a point in homogeneous coordinates) is transformed by a matrix $\mathbf{M}$ to a point p' as follows: $p' = p\,\mathbf{M}$.
- Appearances and texture maps can be specified; see the manual for details.

OOGL Objects

QUAD: a collection of quadrilaterals. The default file suffix is `.quad`.

MESH: a rectangularly-connected mesh of dimension $n \times m$. The default file suffix is `.mesh`.

Bézier: a Bézier surface. The default file suffix is `.bez`.

OFF: an object in *object file format*. It is used for representing collections of planar polygons, possibly with shared vertices. This is a convenient way to describe polyhedra. The polygons may be concave but polygons with holes are not supported. The default file suffix is `.off`.

VECT: strings of connected line segments, possibly closed. The default file suffix is `.vect`.

SKEL: collections of points and polylines, possibly with shared vertices. The default file suffix is `.skel`.

OOGL Objects (cont'd)

SPHERE: a sphere, drawn as a collection of rational Bézier patches. The default file suffix is `.sph`.

INST: a 4×4 transformation, to be applied to another OOGL object. The default file suffix is `.inst`.

LIST: a list of OOGL objects. The default file suffix is `.list`.

TLIST: a list of 4×4 transformations. The default file suffix is `.grp`.

External Modules

- An *external module* is a program that interacts with Geomview. It communicates with Geomview through gcl commands and can control any aspect of Geomview that one can control through Geomview's user interface. Typically, Geomview acts as a *display engine* for the external module.

- External modules known to Geomview are listed in the `Modules` browser in Geomview's `Main` panel. An external module can be invoked by clicking on its entry in the browser.

- In order to make an external module `foo` known to Geomview, include the following line in your local initialization file, `.geomview`:

  ```
  (emodule-define "Foo" "./foo")
  ```

  Here, `Foo` is the name of the external module that will appear in the `Modules` browser of Geomview. One can also execute this command on-line from the `Commands` panel.

External Modules (cont'd)

- When starting an external module, Geomview creates *pipes* connected to the module's standard input and output. (Pipes are like files except they are used for communication between programs rather than for storing data on a disk.)
- Geomview interprets everything that the module writes to its standard output as a gcl command. Likewise, if the exernal module requests any data from Geomview, Geomview writes that data to the module's standard input.
- Note that this implies that the module cannot use standard I/O for communicating with the user!
- Sample external modules (`example*.c`) are available in Geomview's subdirectory `/geomview/doc`.

Using Geomview as an External Display Engine

- It is also possible to invoke Geomview from an application program, and to direct graphics output produced by the application to Geomview, thus using Geomview as an external display engine.

- Communication between the application program and Geomview is again carried out via *pipes*.

- In a C environment, an input file `geomview_in` for Geomview is opened by the application program as follows:

```
geomview_in = popen("togeomview", "w");
```

- Then, the application writes any input for Geomview to `geomview_in`.

- Since pipes may be buffered, it is advisable to flush the pipe after data has been written to the pipe: `fflush(geomview_in)`.

Using Geomview as an External Display Engine (cont'd)

- Typically, we will want Geomview to create objects that can later on be modified (e.g., translated or rotated). The gcl command

  ```
  (geometry Foo { : foo })
  ```

  instructs Geomview to create an object ("geom") named `Foo` as an instance of the *handle* `foo`, where `Foo` is the name of the object as it will appear in the object browser of Geomview, and `foo` is the internal reference for Geomview.

- Handles allow one to name a piece of geometry whose value can be specified elsewhere, and which can be updated repeatedly. See the manual for details.

- If multiple objects are to be passed to Geomview, it is a good idea to turn off any scaling of the individual objects:

  ```
  (normalization Foo none)
  ```

The End!

I hope that you enjoyed this course, and I wish you all the best for your future studies.