Tutorials and worked examples for simulation, curve fitting, statistical analysis, and plotting.
https://simfit.org.uk

In order to appreciate the nature and use of scalable vector graphics files (*.SVG) it is useful to review the two document types, namely bitmap and vector, that are used to archive graphs and include them into computer generated documents or web pages.

Bitmaps

These files (*.BMP) contain the raw information for every pixel captured by a digital photograph or displayed on a computer screen. They have the following properties.

- The larger the number of pixels then the greater the detail recorded.
- Such files are very easy to display, include in documents, or print but, unless there is a large number of pixels, then lines and curves will appear stepped and fonts will pixelate.
- For complicated portraits, landscapes, capture of microscope fields, or 3D display of molecules etc. requiring shading such files are indispensable.
- Where there are appreciable homogeneous patches such as blue sky in a landscape then considerable compression into formats such those of the joint photographic expert group (*.JPG) or portable network graphics (*.PNG) can result in smaller files, but compression is not always lossless.
- Where a bitmap has been created using antialiasing to smooth out polygons and polylines as in text, curves, or lines then compression can lead to fuzzy curves or distorted characters. Traditional computer graphics hardly ever looked good on computer screens because hard edges at an angle would show as a series of steps (a distortion known as aliasing). This made even the simplest graph – such as a sine curve, look ugly. This problem has been largely eliminated by anti-aliasing (automatically employed by SIMFIT). However a bitmap of a specific size generated using anti-aliasing, never looks its best if it is ultimately displayed at another resolution. SVG completely sidesteps this problem because the necessary anti-aliasing is performed as an image is displayed (or printed), so that data are displayed correctly. This property of being device–independent is something that sets aside vector graphics from bitmap graphics.

Vector graphics

Scientific graphs largely consist of axes, curves, and plotting symbols, with small amounts of text, and vector graphic files simply contain the mathematical data such as coordinates necessary to reproduce the graph at any degree of magnification or compression without loss of information. Here is a summary of properties for the two main vector graphics files; encapsulated PostScript (*.EPS), and scalable vector graphics (*.SVG).

- The files are in text format, which means they can easily be edited retrospectively in text editors such as **notepad** in order to change, titles, legends, line types, plotting symbols, or colours.
- They are device independent so there is no loss of information on expansion or compression.
- They can be imported into LaTeX documents or include LaTeX code, so that high quality mathematical formulas and chemical structures can be incorporated.
- The free program **Ghostscript** can be used to convert EPS files into other formats, and similarly **Inkscape** can be used to visualize and transform SVG files.

- While EPS is the main import format for LaTeX documents, SVG is the recommended format for scientific graphs on the internet.

Bogus vector files

Note that many applications claim to transform bitmap and compressed bitmap files into vector files without loss of significant information, but this is almost impossible except for fairly simple images.

Such applications usually just exploit a weakness in vector files that allows bitmaps to be inserted giving bogus vector files that are wrappers containing bitmaps. Similarly portable document files (*.PDF) were developed from PostScript and retain many PostScript features that can be exploited. For instance, using the SimFIT interface to GhostScript to transform SimFIT EPS files into PDF files yields PDF files that are effectively device independent, whereas using Windows to distil files into PDF merely creates bitmap files.

Using SVG files in SimFIT

The SVG format is very comprehensive as it has been specifically developed to be versatile for web use. For that reason few applications implement the whole standard and SimFIT is no exception, so it must be emphasized that SimFIT will only accept and manipulate SVG files according to the graph plotting functionality provided by Silverfrost FTN95 Clearwin+. This interface was written by David Bailey and it provides the following SVG file functionality for SimFIT users.

- The SVG files can be used on the web and opened by browsers such as **firefox**.
- The SVG files can be displayed and edited retrospectively by the program **EditSVG**.
- The SVG files created by other applications cannot be used and cause warning messages. However, in some circumstances a SVG file may use an acceptable subset of SVG facilities. To explore this option you must open the SVG file with a text editor and splice the option Clearwin_output="1" into the line beginning <svg. It is easy to get this wrong, so it is probably best to file the result to a different file name.

Editing SVG files in SimFIT

The functionality provided by procedure **EditSVG** is now listed.

1. The only file types that can be used in this program are:

   (A) *.SVG files created by SimFIT, and other Clearwin+ output from Silverfrost FTN95 programs.

   (B) *.TEX files describing mathematical equations or chemical formulas. Such *.TEX files can be used to generate internal *.SVG files if **latex.exe**, **dvips.exe**, and **dvisvgm.exe** are on the path. For instance if users have a recent version of MikTeX available.

2. Files can be input from the console, by drag and drop, or by using library files.
3. Images can be enlarged or reduced by "right clicking" on the image.
4. Images can be freely positioned by dragging a window from any point on its surface. Using the view menu it is possible to add a graticule with optional "snap to nearest graticule intersection" facility. The graticule does not remain on the finished image.
5. After manipulating a file or a set of files the resulting composite image can be written out to a *.SVG, or *.PNG, file, or even to a *.ISVG rebuild–image file.
6. It can be used to create strict collages where every image is snapped to the nearest grid point, freestyle collages where images can be arranged in arbitrary positions, or overlays where smaller images can be inserted into larger ones.

Using LaTeX

Some examples of how to use these procedures within the SimFIT package from version 7.5.0 onwards using the test files provided follow. However the procedure used to create the SVG files using LaTeX should be noted.

To enlarge on the use of LaTeX it must be emphasized that the procedure to use LaTeX depends on whether the user has a fully functioning LaTeX installation on their machine. So there are three distinct cases.

1. **The direct method**

   There is a LaTeX installation so the user prefers to input a *.TEX file directly into program **EditSVG** whereupon the *.TEX file will be processed and the image will appear in the main window.

2. **The indirect method**

   There is a LaTeX installation but the user prefers to use the command line technique described subsequently to transform the *.TEX file into *.DVI then use **dvisvgm** to transform this into a stand alone *.SVG file. Note that filenames used in this procedure must be local files with no spaces in the file name.

3. **The remote user method**

   There is no LaTeXinstallation so a known LaTeX user will have to perform the transformation. Alternatively, a stand–alone *.SVG file, such as the demonstration files distributed with SimFITand SimDEM, will have to be used.

LaTeX is designed to create documents and, because of this, care is needed to remove much of the header information in order to create simple images that can be imported into **EditSVG**. This is much the same as the steps required to create a *.EPS from from a *.PS file but using the following commands.

To make DVI: use `latex myfile.tex` to create `myfile.dvi`.

To make PS: use `dvips myfile.dvi` to create `myfile.ps`.

To make SVG: use `dvisvgm -E myfile.ps` to create `myfile.svg`.

The argument `-E` indicates that a PostScript file is to be input. Note that white space can be trimmed from the resulting SVG file by **EditSVG**, or alternatively by using **inkscape** or **GSview** (e.g. Version 5) to transform `myfile.ps` into `myfile.eps`, where the `BoundingBox` will automatically remove white space.

Important differences between EPS and SVG files

From within any SimFIT graph it is possible to create *.EPS files by first selecting [PS] then choosing [File], or to create *.SVG files by first selecting [Win] then choosing the [SVG] option. Any *.SVG file created in this way will be a fairly accurate representation of the display, as will any *.EPS files, except that there will be small but significant differences between them. That is unavoidable because the fonts used may differ slightly as the *.SVG file will use Windows fonts whereas the *EPS file will use PostScript fonts. In addition SimFIT allows users to set different global line thickness for EPS and SVG files as line thickness do not scale in exactly the same way in EPS and SVG files. Nevertheless it is useful to know how to create a *.SVG file from a *.EPS file and vice versa.

Fortunately such transformations can readily be carried out due to the widespread availability of Open Source programs such as **Inkscape** and **Cairo**. The usefulness of such transformations will be explained in subsequent tutorial sections.