Compiling LaTeX to computer algebra-enabled HTML5

Bernard Parisse
Institut Fourier
UMR 5582 du CNRS
Université de Grenoble I

2017

Abstract

This document explains how to create or modify an existing LaTeX document with commands enabling computations in the HTML5 output: when the reader opens the HTML5 output, he can run a computation in his browser, or modify the command to be executed and run it. This is done by combining different softwares: `hevea`[6] for compilation to HTML5, `giac.js` for the CAS computing kernel (itself compiled from the C++ Giac[10] library with `emscripten`[11]), and a modified version[9] of `itex2MML`[3] for fast and nice rendering in MathML in browsers that support MathML.

Contents

1

Index

1   Introduction

Combining LaTeX rendering quality and CAS computing is not new:

1. math softwares provide converters to export data to a LaTeX file, or provide automated computations in a way similar to the way bibtex provides bibliography, like sagetex ([2]).

2. some softwares handle both LaTeX-like rendering and computation, for example texmacs ([5]), lyx ([7]), Jupyter notebook ([8]).

However, in the first case, the reader can not modify the CAS commandlines, and in the second case the data format is not standard LaTeX (the writer can not start from an existing document) and requires additional software to be installed on the reader device or a net access to a server to run the computations.

The solution presented here is new in that the writer will edit a standard LaTeX file, add a few easy to learn commands like `\giacinputmath{factor(x^10-1)}` or `\giacinput{plot(sin(x))}` and compile it to produce a HTML5+MathML document. The reader can see the document in any browser (it's optimized for Firefox), without installation, and he can modify computation commandlines and run them on his own computer.

If you are reading this file in PDF format, it is highly recommended to open the HTML5/Mathml version[1] in order to test interactivity and look at the LaTeX source[2]

2   User manual

2.1   Installation on the writer computer

The writer must install

- version 2.30 (or greater) of `hevea`[3] ([6]) or a forked version `hevea-mathjax`[4] ([1])),

- `Giac/Xcas`[5] ([10]) for computing-enabled output

- `heveatomml`[6] ([9]) for MathML output

The files `giac.tex`[7] (or the French version `giacfr.tex`[8]) `giac.js`[9], `hevea.sty`[10], `mathjax.sty`[11] must be copied in the LaTeX working directory. On an Internet con-

---

[1] https://www-fourier.ujf-grenoble.fr/~parisse/giac/castex.html
[2] https://www-fourier.ujf-grenoble.fr/~parisse/giac/castex.tex
[3] http://hevea.inria.fr/
[4] https://github.com/YannickChevalier/hevea-mathjax
[5] https://www-fourier.ujf-grenoble.fr/~parisse/giac.html
[6] https://www-fourier.ujf-grenoble.fr/~parisse/giac/heveatomml.tgz
[7] https://www-fourier.ujf-grenoble.fr/~parisse/giac/giac.tex
[8] https://www-fourier.ujf-grenoble.fr/~parisse/giac/giacfr.tex
[9] https://www-fourier.ujf-grenoble.fr/~parisse/giac/giac.js
[10] https://www-fourier.ujf-grenoble.fr/~parisse/giac/hevea.sty
[11] https://www-fourier.ujf-grenoble.fr/~parisse/giac/mathjax.sty

nected linux box, the writer can run once the following shell script to install the tools
required for HTML5/MathML output :

```
#! /bin/bash
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/giac.tex
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/giacfr.tex
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/giac.js
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/hevea.sty
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/mathjax.sty
wget http://hevea.inria.fr/distri/unstable/hevea-2017-05-18.tar.gz
tar xvfz hevea-2017-05-18.tar.gz
cd hevea-2017-05-18
make
sudo make install
cd ..
wget https://www-fourier.ujf-grenoble.fr/~parisse/giac/heveatomml.tgz
tar xvfz heveatomml.tgz
cd heveatomml/src
make
sudo make install
cd ../..
```

2.2 On the writer side

We now assume that the installation is done. The writer opens a LaTeX file with his
usual editor. He must add in the preamble the following lines

```
\makeindex
\input{giac.tex}

\giacmathjax
```

For interactive CAS LaTeX commands support, the writer should add

```
\begin{giacjshere}
\tableofcontents
\printindex
```

just after `\begin{document}` and

```
\end{giacjshere}
```

just before `\end{document}`. Printing the table of contents and index before the
first LaTeX section command is recommended, otherwise the HTML output `Table`
and `Index` buttons will not link correctly.

The rest of the source file is standard LaTeX except that

- References to numbered equations should be inside additional backslash-ed paren-
  thesis, for example

4

```
\begin{equation} \label{eq:test}
 \frac{2}{x^2-1}=\frac{1}{x-1}-\frac{1}{x+1}
\end{equation}
From equation (\(\ref{eq:test}\)) ...
```

$$\frac{2}{x^2-1} = \frac{1}{x-1} - \frac{1}{x+1} \tag{1}$$

From equation (1) ...

- `\mathbb{}` should be explicit, commands like `\R` where `\R` is defined by `\newcommand{\R}{\mathbb{R}}` will not work.

- New commands are available for interactive CAS support

  - `\giacinputmath{commandline}` will output an inline `commandline` that the user can modify and execute, the answer will be displayed in MathML (or SVG for 2-d graph output).
    Example : `\giacinputmath{factor(x^10-1)}`

    `factor(x^10-1)`

    $$(x-1) \cdot (x+1)(x^4 - x^3 + x^2 - x + 1)(x^4 + x^3 + x^2 + x + 1)$$

    Warnings, if your command contains < or >, you must replace them by `<` or `>`, otherwise they will be interpreted as HTML delimiters. You can also use the `giacprog` and `giaconload` environments explained below.
    If the output is a 2-d graph, do not skip a line with `\\` after the command for PDF output

  - `\giaccmdmath{command}{arguments}` will output `command` in a button following the `arguments`, the reader can only modify the arguments:
    `\giaccmdmath{factor}{x^4-1}`

    `factor(x^4-1)`

    $$(x-1) \cdot (x+1)(x^2 + 1)$$

  - These commands may take an optional HTML style argument, for example
    `\giacinputmath[style="width:200px;"]{factor(x^10-1)}`

    `factor(x^10-1)`

    $$(x-1) \cdot (x+1)(x^4 - x^3 + x^2 - x + 1)(x^4 + x^3 + x^2 + x + 1)$$

5

```
\giaccmdmath[style="font-size:x-large"]{factor}{x^4-1}
```

```
factor(x^4-1)
```

$$(x - 1) \cdot (x + 1)(x^2 + 1)$$

– There are similar commands for outlined output `\giacinputbigmath{}`
  or `\giaccmdbigmath{}{}`:
  For example `\giacinputbigmath{factor(x^25-1)}`

```
factor(x^25-1)
```

$$(x - 1)(x^4 + x^3 + x^2 + x + 1)(x^{20} + x^{15} + x^{10} + x^5 + 1)$$

Example with an optional style argument `\giacinputbigmath[style="width:600px;height:2`

```
factor(x^25-1)
```

$$(x - 1)(x^4 + x^3 + x^2 + x + 1)(x^{20} + x^{15} + x^{10} + x^5 + 1)$$

```
\giaccmdbigmath{factor}{x^25-1}
```

```
factor(x^25-1)
```

$$(x - 1)(x^4 + x^3 + x^2 + x + 1)(x^{20} + x^{15} + x^{10} + x^5 + 1)$$

```
\giaccmdbigmath[style="width:600px;height:20px;"]{factor}{x^25-1}
```

```
factor(x^25-1)
```

$$(x - 1)(x^4 + x^3 + x^2 + x + 1)(x^{20} + x^{15} + x^{10} + x^5 + 1)$$

– Similar commands with text (or plot) output `\giacinput` and `\giacinputbig`
  and `\giaccmd`, example:
  `\giacinput{factor(x^4-1)}`:

```
factor(x^4-1)
```

$$(x - 1) \cdot (x + 1)(x^2 + 1)$$

```
\giaccmd{print}{"Hello world"}:
```
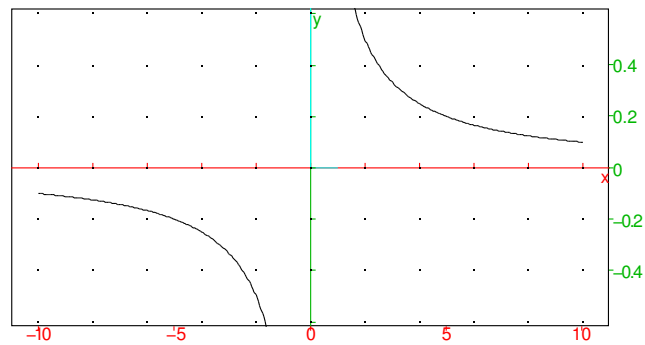
```
print("Hello world")
```

$$0$$

With optional style argument
```
\giacinput[style="font-size:x-large"]{plot(1/x)}
```

```
plot(1/x)
```



```
\giaccmd[style="font-size:x-large"]{factor}{x^4-1}
```

```
factor(x^4-1)
```

$$(x - 1) \cdot (x + 1)(x^2 + 1)$$

– The `giacprog` environment should be used for programs or multi-line
commands
```
\begin{giacprog}...\end{giacprog}
```
Inside this environment, you can keep < and >. The program will be parsed
once the user press the `ok` button. After parse, the program may be modi-
fied and parsed again.
Warning: Do not use the `giacprog` environment in another environment
(like itemize or enumerate).
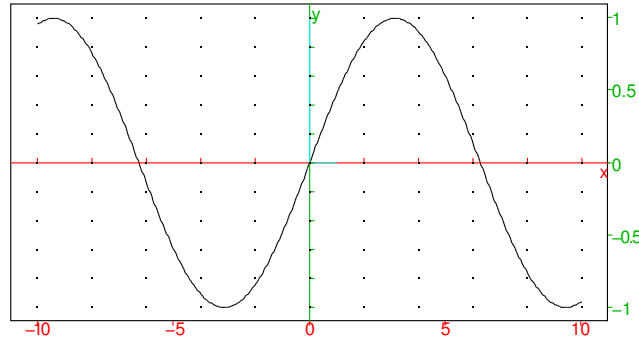If you want the program to be parsed at load-time, replace `giacprog` with
`giaconload`:
```
\begin{giaconload}...\end{giaconload}
```
– The `\giacslider{idname}{min}{max}{step}{value}{command}`
command will add a slider. When the user modifies the slider interactively,

```

the new `value` is stored in `idname` and the `command` (depending on `idname`) is executed. Example:

```
\giacslider{a}{-5}{5}{0.1}{0.5}{plot(sin(a*x))}
```

```
a:=0.5;plot(sin(a*x))
```


– The `\giachidden` command behaves like `\giaccmd` except that the default HTML5 style is "hidden" until the command button has been pressed.

– The `\giaclink` command will add a link in the HTML version and nothing in PDF/DVI. The links open in a new tab, and the corresponding text may be specified as optional argument (default is Test online). Note that `hevea.sty` provides similar commands (`\ahref`, `\footahref`, `\ahrefurl`) with output in PDF/DVI.
Example with a link to Xcas for Firefox with a few commands

```
\giaclink{http://www-fourier.ujf-grenoble.fr/\%7eparisse/xcasen.html#+fa
```

Once the source file is written, it is compiled to HTML5 with the command
`hevea2mml sourcefile.tex`
The HTML output and the `giac.js` files should be in the same directory on the web server. Index and bibliography should be processed with `makeindex` and `bibhva`.

If a PDF output is desired, the command `icas` from a Giac/Xcas installation should be used instead of `pdflatex` because it will run all CAS commands, output them in a temporary LaTeX file, and run `pdflatex` on the output (this was inspired by the pgiac script[12] from Jean-Michel Sarlat [4]). The temporary file name is obtained by adding a _ at the end of the initial file name (without the `.tex` extension). Therefore, if you have an index and or citations, you should run `makeindex` and `bibtex` on the file name with _ appended. For `bibtex` citations in the HTML files, you should run `bibhva`. For example, the PDF version of this document is available here[13].

---

[12]`http://melusine.eu.org/syracuse/giac/pgiac/`
[13]`https://www-fourier.ujf-grenoble.fr/~parisse/giac/castex.pdf`

2.3 On the reader side

The reader's browser opens an HTML5+MathML file (linking to the JavaScript `giac.js`). The MathML is rendered natively on Firefox or Safari, while Chrome or Internet Explorer will automatically load MathJax to render MathML (this is of course noticeably slower if the document is large). Computations are run by the reader's browser (the CAS is JavaScript code). This is slower than native code but faster than net access to a server and it does not require setting up a specific server for computations.

2.4 More examples

2.4.1 Trace (2-d graph)

This example illustrates with a slider that the evolute of a curve is the envelope of the normals to the curve, here the curve is an ellipsis and the envelop an astroid. The list of normals `L` is initialized empty at load-time.

```
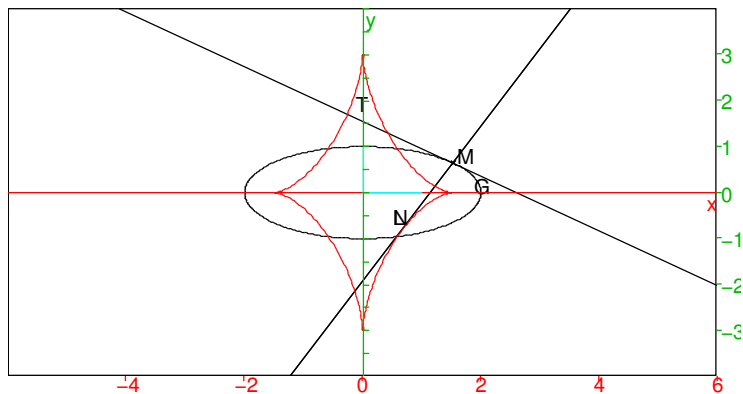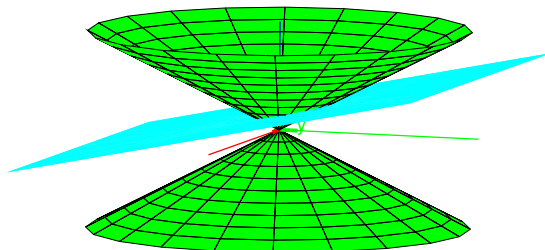L:=[]
```

Now move the slider:

```
t0:=0.7;gl_x=-6..6;gl_y=-4..4;G:=plotparam
([2*cos(t),sin(t)],t=0..2*pi);M:=element
(G,evalf(t0));T:=tangent(M);N:=perpendicular
(M,T);L:=append(L,N);evolute(G,color=red)
```


2.4.2 Cone section (3-d graph)

C is a cone of center the origin, axis of direction $(0, 0, 1)$, and angle $\frac{\pi}{6}$, P is a plane of equation $z = my + 3$. m is controlled by the slider, when m moves the intersection is an ellipsis or hyperbola (limit value is a parabola).

```
m:=0.7;C:=cone([0,0,0],[0,0,1],pi/6, display=green+filled
);P:=plane(z=evalf(m)*y+3,display=cyan+filled);
```

2.4.3 Dunford decomposition (CAS)

A program computing the Dunford decomposition of a matrix with Newton method. It is parsed at load-time (`giaconload` environment).

```
function dunford(A)
  local U,p,q,q1,j,d,n;
  U:=A;
  n:=nrows(U);
  p:=charpoly(U);
  q:=p/gcd(p,p'); // square free part
  q1:=q';
  for (j:=1;j<=n;j:=2*j){
    d:=inv(horner(q1,U))*horner(q,U); // Newton step
    if (d==0*d) return U,A-U;
    U:=U-d;
  }
  return U,A-U;
end:;
```

Example : we define J an almost diagonal matrix and A a similar matrix and we check the Dunford decomposition of A.

$$J = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & 0 \\ 3 & 4 & 1 \end{pmatrix}, \quad A = PJP^{-1}$$

```
J:=[[2,0,0],[0,1,1],[0,0,1]]; P:=trn([[1,2,3
],[0,-1,4],[0,0,1]]); A:=P*J*inv(P)
```

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & 0 \\ 3 & 4 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 & 0 \\ 13 & -3 & -1 \\ -41 & 16 & 5 \end{pmatrix}$$

```
D,N:=dunford(A); N^2; P*diag(diag(J))*inv(P)
```

$$\begin{pmatrix} 2 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 11 & -4 & -1 \\ -44 & 16 & 4 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}$$

2.4.4 Slopefield

This will display the slopefield of an ordinary differential equation

$$\frac{dy}{dt} = -y + cos(t)$$

and one solution corresponding to an initial condition $y(0)$ that the user may modify with the slider.

```
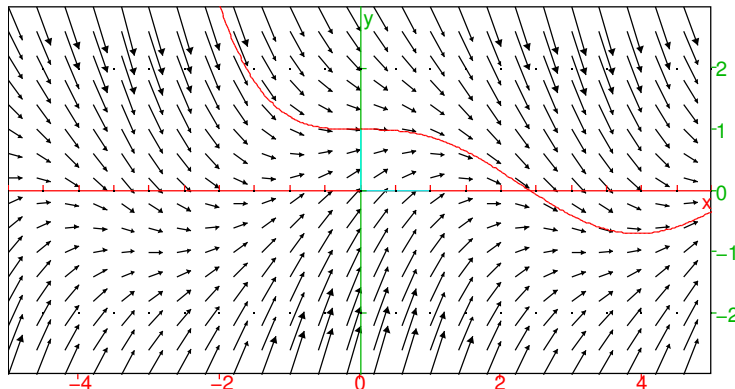y0:=1.0;gl_x=-5..5; gl_y=-3..3;plotfield
(-y+cos(t),[t=-5..5,y=-3..3],xstep=0.4,ystep=0.4
);plotode(-y+cos(t),[t=-5..5,y],[0,y0],tstep=0.1,color=red
)
```


2.4.5 Gröbner basis (CAS)

The CAS kernel can compute non-trivial Gröbner basis. Of course, the JavaScript version is significantly slower than the native Giac/Xcas kernel.

```
kat7:=[-x1+2*x8^2+2*x7^2+2*x6^2+2*x5^2+2*x4^2+2*x3^2+2*x2^2+x1^2,
 -x2+2*x8*x7+2*x7*x6+2*x6*x5+2*x5*x4+2*x4*x3+2*x3*x2+2*x2*x1,
 -x3+2*x8*x6+2*x7*x5+2*x6*x4+2*x5*x3+2*x4*x2+2*x3*x1+x2^2,
 -x4+2*x8*x5+2*x7*x4+2*x6*x3+2*x5*x2+2*x4*x1+2*x3*x2,
 -x5+2*x8*x4+2*x7*x3+2*x6*x2+2*x5*x1+2*x4*x2+x3^2,
 -x6+2*x8*x3+2*x7*x2+2*x6*x1+2*x5*x2+2*x4*x3,
 -x7+2*x8*x2+2*x7*x1+2*x6*x2+2*x5*x3+x4^2,
 -1+2*x8+2*x7+2*x6+2*x5+2*x4+2*x3+2*x2+x1]:;
```

Basis over $\mathbb{Z}/16777213$

```
G:=gbasis(kat7 mod 16777213,[x1,x2,x3,x4,x5,x6,x7,x8
]):; size(G); G[20];
```

Done, 74, $x5 \cdot x7 \cdot x8^4 + 6710886 \cdot x4 \cdot x8^5 - 3938997 \cdot x5 \cdot x8^5 + 5106109 \cdot x6 \cdot x8^5 - 5543774 \cdot x7 \cdot x8^5 + 4960220 \cdot x8^6 - 1622886$

Basis over $\mathbb{Q}$

```
G:=gbasis(kat7,[x1,x2,x3,x4,x5,x6,x7,x8]
):; size(G); G[20];
```

Done, 74, $1848962411667810716195758327488000 \cdot x5 \cdot x7 \cdot x8^4 + 14791699293342485729566066619904000 \cdot x4 \cdot x8^5 +$

3 How this is done

The LaTeX \giac... commands are defined in giac.tex. For example \giacinput is defined like this:

```
\newcommand{
\verb||\\
$$\,\mathrm{undef}\,$$
[2][style="width:400px;font-size:large"]{
\ifhevea
\@print{<textarea onkeypress="UI.ckenter(event,this,1)" }
\@getprint{#1>#2}
\@print{</textarea><button onclick="previousSibling.style.display='inherit';var
\else
\lstinline@#2@
\fi
}
```

If hevea compiles the command, the \ifhevea part is active, and the command will output an HTML5 <textarea> element and a OK <button>, with a callback to JavaScript code that will evaluate the CAS command inside the textarea
var tmp=UI.caseval(previousSibling.value)
and fill the next HTML5 <span> field with the result of the CAS command.

The CAS evaluation is performed by a call to giaceval in the UI.caseval code (defined in giac.tex), where giaceval is a global JavaScript variable assigned at page load-time from the Module interface created by compiling Giac/Xcas with the C++ to JavaScript compiler emscripten[14]. The CAS code being in JavaScript, it can be run on every JavaScript-enabled browser. It will be faster on browsers that have support for asm.js (asmjs.org) like Mozilla Firefox: numerical computations are 1 to 2 times slower than native code, while exact computations are 2 to 10 times slower than native code (the main reason being that JavaScript has currently no 64 bits integer type).

---

[14]http://kripken.github.io/emscripten-site/

For a PDF output, if `pdflatex` is run on the tex file, giac commands will be written verbatim, but they will not be processed. The `icas` command from the Giac/Xcas package will filter all giac commands, process them and output the result in math mode in a temporary LaTeX file. If the answer is a 2-d graph output, `icas` will output a pdf file on the hard disk and output a corresponding `\includegraphics` command in the temporary LaTeX file. After that, the temporary file will be processed by `pdflatex`.

4 Conclusion

The current version of `icas` and `giac.tex` are already usable to easily produce HTML interactive CAS-enabled document from LaTeX documents. They may be completed in future versions depending on user requests. For example, online courses might have commands to enable student exercises answers auto-check.

Acknowledgements

References

[1] Yannick Chevallier. Hevea: LaTeX to HTML5 compiler, fork for MathJax support. `https://github.com/YannickChevalier/hevea-mathjax`, 2017.

[2] Dan Drake. SageTex. `https://www.ctan.org/pkg/sagetex`, 2009.

[3] Jacques Distler. LaTeX to MathML converter. `golem.ph.utexas.edu/˜distler/blog/itex2MML.html`, 2016.

[4] Jean-Michel Sarlat. pgiac. `http://melusine.eu.org/syracuse/giac/pgiac/`, 2011.

[5] Joris van der Hoeven. Texmacs. `http://www.texmacs.org/`, 2017.

[6] Luc Maranget. Hevea: LaTeX to HTML5 compiler (unstable version). `http://hevea.inria.fr/distri/unstable/`, 2017.

[7] Matthias Ettrich. Lyx. `https://ww.lyx.org`, 2012.

[8] NumFOCUS Foundation. The Jupyter Notebook. `jupyter.org`, 2017.

[9] Bernard Parisse. LaTeX to Mathml converter, fork for hevea output support. `www-fourier.ujf-grenoble.fr/˜parisse/giac/heveatomml.tgz`, 2017.

[10] Bernard Parisse and Renée De Graeve. Giac/Xcas Computer Algebra System. `www-fourier.ujf-grenoble.fr/˜parisse/giac.html`, 2017.

[11] Alon   Zakai.   Emscripten: A   C/C++   to   Javascript   compiler. `kripken.github.io/emscripten-site/`, 2017.