SAGE, the open source CAS
to end up all CASs?

Thomas Risse
Institute of Informatics & Automation, IIA
Faculty E&I, Hochschule Bremen, HSB
University of Applied Sciences

SEFI MWG, June 21\textsuperscript{th} – 23\textsuperscript{rd}, 2010
Agenda

1. Introduction – Criteria for Computer Algebra Systems
2. SAGE = System for Algebraic & Geometric Experimentation
3. (Symbolic) Calculus
4. Linear Algebra
5. Numerics
6. Ordinary Differential Equations – IVP
7. Ordinary Differential Equations – BVP
8. Programming
9. Comparison
10. Conclusion – Try SAGE
Criteria for CASs in Education & Research

CASs are problem solving tools: get to know, use, solve
CASs allow the investigation of algorithms: implement, analyse, perform
deployment of CASs helps to understand mathematics: visualise, animate, illustrate concepts
analysis of the CAS itself reveals the underlying math: investigate, extend, compare

requirements that determine the role of CASs in education

- **power** both for learning/teaching and research
- **usability** intuitively usable, problem oriented
- **availability** available, easy to install, ready to use
SAGE = System for Algebraic and Geometric experimentation
vs MATLAB or Octave&GinaC resp.

**Similarities**
both are CASs (incl. *Symbolic Math Toolbox*), command line oriented, programmable, problem oriented, extendable etc
both feature graphical capabilities, different workspaces, interpreted, precompiled, object orientation etc

**Differences**

<table>
<thead>
<tr>
<th>SAGE</th>
<th>MATLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>local implementation</td>
<td>local implementation</td>
</tr>
<tr>
<td>&amp; web interface</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>m-files</td>
</tr>
<tr>
<td>many predefined data types</td>
<td>few predefined data types</td>
</tr>
<tr>
<td>objects</td>
<td>handles</td>
</tr>
<tr>
<td>modules</td>
<td>toolboxes</td>
</tr>
<tr>
<td>open source</td>
<td>proprietary/open source</td>
</tr>
</tbody>
</table>
(Symbolic) Calculus: visualisation

\begin{align*}
\text{plot}(\sin(x)) & \quad \text{ezplot('sin(x)')} \\
\text{plot}(\sin(x),-3,3) & \quad \text{ezplot('sin(x)',-3,3)} \\
p=\text{plot}(\sin(x),-3,3, \text{color='red'}); & \quad \text{hd1=ezplot('sin(x)',-3,3);} \\
q=\text{plot}(\cos(x),-3,3, \text{color='blue'}); & \quad \text{set(hdl,'color','r');} \\
\text{show}(p+q); & \quad \text{hold on;}
\end{align*}

\begin{align*}
hd1=\text{ezplot('cos(x)',-3,3)}; & \quad \text{hd1=ezplot('cos(x)',-3,3);}
\text{set(hdl,'color','b');} & \quad \text{hold off;}
\end{align*}

\begin{align*}
\text{plot}(1/x, \text{detect_poles=True}) & \quad \text{ezplot('1/x',-1,1)}
\end{align*}
Differentiation and Integration

\[
\text{diff}(\sin(x)/x, x) \quad \text{diff}(\sin(x)/x, x)
\]

\[
\text{integral}(x*\sin(x), x) \quad \text{int}(x*\sin(x), x)
\]

\[
\text{integral}(\exp(-x^2/2), x, 0, 1) \quad \text{int}(\exp(-x^2/2), x, 0, 1)
\]

of course, with numerical evaluation, e.g.

\[
\text{N(integral(}\sin(x)/x, x, 0, 1)) \quad \text{N(int(}\sin(x)/x, x, 0, 1))
\]
Solving equations symbolically

### e.g. linear (systems)

<table>
<thead>
<tr>
<th>SAGE</th>
<th>Calculus</th>
<th>Linear Algebra</th>
<th>Numerics</th>
<th>ODEs - IVP</th>
<th>ODEs - BVP</th>
<th>Programming</th>
<th>Comparison</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

```python
vrs=var('a b c d e f x y');
solve([a*x+b*y==c, d*x+e*y==f],x,y)
syms a b c d e f x y;
[x y] = solve( ’a*x+b*y-c’, ’d*x+e*y-f’,x,y)
```

### e.g. non-linear

```python
vars=var(’a b c d x’);
solve( a*x^3+b*x^2+c*x+d==0,x)
syms a b c d x;
solve( ’a*x^3+b*x^2+c*x+d’)  
```

### e.g. differential

```python
t = var(’t’);
x = function(’x’,t);
desolve(diff(x,t)+x==1,[x,t])
dsolve(’Dx+x=1’)
```
Linear Algebra – vectors & matrices

```python
pnts = [(random(),random())
for _ in range(5)];
p = line(pnts);
c = cos(pi/4);
s = sin(pi/4);
R=N(matrix([[c,s],[-s,c]]));
pnts = matrix(pnts)*R;
q=line([(pnts[i,0],pnts[i,1])
for i in range(5)],
color='red');
(p+q).show()
```

```python
plgn = rand(2,5);
plot(plgn(1,:), ... plgn(2,:));
hold on;
c=cos(pi/4);
s=sin(pi/4);
R = [c s;-s c];
plgn = R*plgn;
plot(plgn(1,:), ... plgn(2,:),’r’); 
hold off;
```
linear equations

vars = var('a,b,c,d,e,f');
A = matrix([[a,b],[c,d]]);
b = vector([e,f]); A\b

or again, also symbolically

A = matrix([[1,2,3], [3,2,1],[1,1,1]]);
b = vector([0,-4,-1]); A\b

eigenvalues & eigenvectors

A = Matrix([[1,2,3], [3,2,1],[1,1,1]]);
A.eigenvalues(); # or
A.eigenvectors_right();

A = sym([[1,2,3; ... 3,2,1;1,1,1]]);
b = sym([0;-4;-1]); A\b

A = sym([1,2,3; ... 3,2,1;1,1,1]);
eig(A) % or
[V,D] = eig(A);
Numerics

both CASs based on based on BLAS or LAPACK routines

- solving linear equations by decomposition: LU, Cholesky, Schur, SVD
- least squares, eigenvalues & eigenvectors
- solving non-linear equations
- quadrature, double/tripple quadrature
- solving ordinary differential equations

```
x = var('x');
find_root(cos(x)==sin(x),0,pi/2)
```

% x0 is start guess
% or start intervall
fzero(...
@(x)(cos(x)-sin(x)), x0)
Ordinary Differential Equations – IVP

Volterra-Lottka IVP model \( y' = \begin{pmatrix} y_1(a - by_2) \\ y_2(-c + dy_1) \end{pmatrix} \) for \( t \in [0, 25] \) and for two populations \( y_1(t) \) and \( y_2(t) \).

```python
# initialize
# a,b,c,d,y0[0],y0[1]
def odefun(y,t):
    return [
        y[0]*(a-b*y[1]),
        y[1]*(-c+d*y[0])
    ];
t = srange(0,25,0.01);
y = odeint(odefun,y0,t)
```

```latex
% initialize
% a,b,c,d,y0(1),y0(2)
function fvl = odef(t,y)
fvl=[y(1)*(a-b*y(2));
    y(2)*(-c+d*y(1))];
end
tspan = [0 25];
[t y_ode45] = ...
ode45(@odef,tspan,y0);
```
Some BVP \[ u'' + 2u' + u = 0 \] for \( 0 \leq x \leq \frac{\pi}{2} \) with \( u(0) = 3, u\left(\frac{\pi}{2}\right) = 2 \) is equivalent to \[ y' = \begin{pmatrix} y_2 \\ -2y_2 - y_1 \end{pmatrix} \]

```python
vars = var('x');
de = diff(y,x,2) + 2*diff(y,x) + y == 0;
sln = desolve(de, y, [0,3,pi/2,2]);
plot(sln(x),(x,0,pi/2));

# there are packages
# like the numerical
# scikits.bvp_solver
# also ...
```

```python
f = @(x,y) [y(2); ...
    -y(2)-y(1)];
g = @(ya,yb)[ya(1)-3; ...
    yb(1)-2];
x = linspace(0,pi/2,10)';
yguess=@(x)[3-2*x/pi; ...
    -2/pi];
init = bvpinit(x,yguess);
sln = bvp4c(f,g,init);
xp = 0:0.01:pi/2;
plot(xp,deval(sln,xp,1));
```
Programming

Animation

```python
anmt = animate( \\
    [point([ \\
        2*sin(pi/8*cos(t)), \\
        2*(1-cos(pi/8*cos(t)))]]\n    for t in \n    srange(0,11,0.1)],\n    xmin=-1,xmax=1, \n    ymin=0,ymax=2)\nanmt.show()

plot([-2,2,0,0], ... 
     [0,0,0,2],’k’);
hold on;
h=plot(0,0,’.k’);
while true
    set(h,’xData’, ... 
        2*sin(pi/8*cos(t))); 
    set(h,’yData’, ... 
        2-2*cos(pi/8*cos(t))); 
    drawnow;
end; hold off;
```
**Interaction**

```python
@interact
def NN(a=(0,2), \ 
b=1, c=[0,2,4])
p=plot( \ 
a*x^2+b*x+c,\ 
-2,2);
p.show();
```

```matlab
% begin code snippet
a = get(ha,'Value');
b=double(get(hb,'String'));
c = get(hc,'Value');
f = @(x) a*x.^2+b*x+c;
ezplot(f);
% end code snippet
```
Comparison is difficult [14].

**e.g. symbolical engines**

```python
x=var('x'); bool(x<x+1)  # SAGE
sym x; expr = (x<x+1)  # MATLAB
```

SAGE returns True whereas MATLABs Symbolic Math Toolbox with the Maple kernel complains about "Undefined function or method 'lt' for input arguments of type 'sym'".

```python
vars=var('a b'); expr= \sqrt(a*b)-\sqrt(a)\sqrt(b);
expr.simplify_full()  # SAGE
syms a b; expr = ... sqrt(a*b)-sqrt(a)*sqrt(b);
simple(expr)
simplify(expr)  # MATLAB
```

SAGE duly returns 0 whereas MATLABs simple and simplify both return the original expr unchanged.

**e.g. discrete math**

SAGEs strong point is discrete mathematics [13]: polynomial rings, groups, elliptic curves, and lots of algebra. Hence, SAGE supports applications in combinatorics, cryptography, coding, and graph theory for example. MATLAB natively does not offer equivalents.
Conclusion – Try SAGE

SAGE allows

- to tackle problems in many areas of mathematics
- to natively solve problems numerically and symbolically
- to access specialized computer algebra systems

SAGE is ready to be used

- without the need to install any software (SAGE runs on a server like e.g. www.sagenb.org or http://SAGE.informatik.hs-bremen.de)
- without the need to worry about licences and license fees
- with supportive user community.

Try SAGE and tap the big potential of SAGE.
References I

www.network-theory.co.uk/docs/octave3/


University of Washington, 2007
http://sage.math.washington.edu/home/wdj/sigsam/opensource_math2.html

Cambridge University Press, 2010

[5] MATLAB, the Language of Technical Computing;
www.mathworks.com

www.scipy.org/NumPy_for_Matlab_Users?action=print

References II

[8] Open source mathematics;
www.opensourcemath.org/opensource_math.html


[10] SAGE – web interface login page;
www.sagenb.org and http://nb.sagemath.org

www.sagemath.org/tour-benchmarks.html


www.weblearn.hs-bremen.de/risse/papers/MathEng8

www.math.unm.edu/~wester/cas_review.html