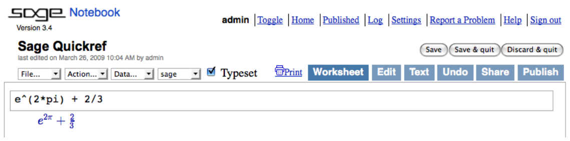
Sage: Guia de Referència Ràpida

William Stein (basada en treball de P. Jipsen)

Adaptació al català: Mauro Viader Olivé i Joaquim Puig

Notebook



Avaluar cel·la: ⟨shift-enter⟩

Avaluar cel·la creant-ne una de nova: ⟨alt-enter⟩

Partir cel·la: ⟨control-;⟩

Ajuntar cel·les: ⟨control-backspace⟩

Inserir cel·la matemàtica: clicar la línia blava entre les cel·les

Inserir cel·la text/HTML : shift-clicar la línia blava entre les cel·les i això obrirà un editor WYSIWYG

Esborrar cel·la: esborrar el contingut i després backspace

Línia de comandes

com⟨tab⟩ completa *command*

`*`*bar*`*?` llista les comandes que contenen “bar”

command`?`⟨tab⟩ mostra la documentació

command`??`⟨tab⟩ mostra el codi font

`a.`⟨tab⟩ mostra els mètodes per a un objecte `a` (més: `dir(a)`)

`a._`⟨tab⟩ mostra els mètodes ocults de per a un objecte `a`

`search_doc("`*string o regexp*`")` cerca tot el text dels documents

`search_src("`*string o regexp*`")` cerca el codi font, surt `_` abans de la sortida

Nombres

Enters: $\mathbf{Z}$ = `ZZ` ex. `-2 -1 0 1 10^100`

Racionals: $\mathbf{Q}$ = `QQ` ex. `1/2 1/1000 314/100 -2/1`

Reals: $\mathbf{R} \approx$ `RR` ex. `.5 0.001 3.14 1.23e10000`

Complexos: $\mathbf{C} \approx$ `CC` ex. `CC(1,1) CC(2.5,-3)`

Precisió de double: `RDF` i `CDF` ex. `CDF(2.1,3)`

Mòdul n: $\mathbf{Z}/n\mathbf{Z}$ = `Zmod` ex. `Mod(2,3) Zmod(3)(2)`

Cossos finits: $\mathbf{F}_q$ = `GF` ex. `GF(3)(2) GF(9,"a").0`

Polinomis: $R[x,y]$ ex. `S.<x,y>=QQ[] x+2*y^3`

Sèries: $R[[t]]$ ex. `S.<t>=QQ[[]] 1/2+2*t+O(t^2)`

Nombres p-àdics: $\mathbf{Z}_p \approx$`Zp`, $\mathbf{Q}_p \approx$`Qp` ex. `2+3*5+O(5^2)`

Clausura algebraica: $\overline{\mathbf{Q}}$ = `QQbar` ex. `QQbar(2^(1/5))`

Interval aritmètic: `RIF` ex. `sage: RIF((1,1.00001))`

Cos numèric: `R.<x>=QQ[];K.<a>=NumberField(x^3)`

Aritmètica

ab = `a*b` $\frac{a}{b}$ = `a/b` a^b = `a^b` $\sqrt{x}$ = `sqrt(x)`

$\sqrt[n]{x}$ = `x^(1/n)` $|x|$ = `abs(x)` $\log_b(x)$ = `log(x,b)`

Sumes: $\sum_{i=k}^{n} f(i)$ = `sum(f(i) for i in (k..n))`

Productes: $\prod_{i=k}^{n} f(i)$ = `prod(f(i) for i in (k..n))`

Constants i funcions

Constants: π = `pi` e = `e` i = `i` ∞ = `oo`

ϕ = `golden_ratio` γ = `euler_gamma`

Aproximació: `pi.n(digits=18)` = 3,14159265358979324

Funcions: `sin cos tan sec csc cot sinh cosh tanh sech csch coth log ln exp` ...

Funció en Python: `def f(x): return x^2` (els blocs en Python síndenten amb 3 espais després de :)

Funcions Interactives

Escriu `@interact` abans de la funció (les vars determinen el control)

```
@interact
def f(n=[0..4], s=(1..5), c=Color("red")):
  var("x");show(plot(sin(n+x^s),-pi,pi,color=c))
```

Expresions Simbòliques

Defineix una nova variable simbòlica: `var("t u v y z")`

Funció Simbòlica: ex. $f(x) = x^2$ `f(x)=x^2`

Relacions: `f==g f<=g f>=g f<g f>g`

Resol $f = g$: `solve(f(x)==g(x), x)`

`solve([f(x,y)==0, g(x,y)==0], x,y)`

`factor(...)` `expand(...)` `(...).simplify_...`

`find_root(f(x), a, b)` troba $x \in [a,b]$ s.t. $f(x) \approx 0$

Càlculs

$\lim_{x\to a} f(x)$ = `limit(f(x), x=a)`

$\frac{d}{dx}(f(x))$ = `diff(f(x),x)`

$\frac{\partial}{\partial x}(f(x,y))$ = `diff(f(x,y),x)`

`diff` = `differentiate` = `derivative`

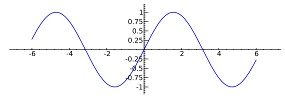
$\int f(x)dx$ = `integral(f(x),x)`

$\int_a^b f(x)dx$ = `integral(f(x),x,a,b)`

$\int_a^b f(x)dx \approx$ `numerical_integral(f(x),a,b)`

Polinomi de Taylor, grau n respecte a: `taylor(f(x),x,`a`,`n`)`

Gràfics 2D



`line([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*opcions*`)`

`polygon([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*opcions*`)`

`circle((`x`,`y`),`r`,`*opcions*`)`

`text("txt",(`x`,`y`),`*opcions*`)`

les *options* són a `plot.options`, ex. `thickness=`*pixel*, `rgbcolor=(`r`,`g`,`b`)`, `hue=`h on $0 \le r,b,g,h \le 1$

`show(`*gràfic*`, `*opcions*`)`

usa `figsize=[w,h]` per ajustar el tamany

usa `aspect_ratio=`*número* per ajustar la relació dáspecte

`plot(f(`x`),(`x`,`$x_{\min}$`,`$x_{\max}$`),`*opcions*`)`

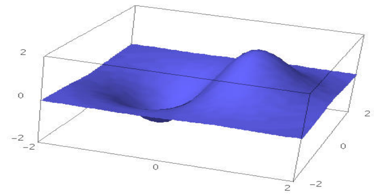
`parametric_plot((f(`t`),g(`t`)),(`t`,`$t_{\min}$`,`$t_{\max}$`),`*opcions*`)`

`polar_plot(f(`t`),(`t`,`$t_{\min}$`,`$t_{\max}$`),`*options*`)`

combinar: `circle((1,1),1)+line([(0,0),(2,2)])`

`animate(`*llista de gràfics, opcions*`).show(delay=20)`

Gràfics 3D



`line3d([(`x_1`,`y_1`,`z_1`),...,(`x_n`,`y_n`,`z_n`)],`*opcions*`)`

`sphere((`x`,`y`,`z`),`r`,`*opcions*`)`

`text3d("txt", (`x`,`y`,`z`), `*opcions*`)`

`tetrahedron((`x`,`y`,`z`),`*size*`,`*opcions*`)`

`cube((`x`,`y`,`z`),`*tamany*`,`*opcions*`)`

`octahedron((`x`,`y`,`z`),`*tamany*`,`*opcions*`)`

`dodecahedron((`x`,`y`,`z`),`*tamany*`,`*opcions*`)`

`icosahedron((`x`,`y`,`z`),`*tamany*`,`*opcions*`)`

`plot3d(f(`x`,`y`),(`x`,`x_b`,`x_e`), (`y`,`y_b`,`y_e`),`*opcions*`)`

`parametric_plot3d((f,g,h),(`t`,`t_b`,`t_e`),`*opcions*`)`

`parametric_plot3d((f(`u,v`),g(`u,v`),h(`u,v`)), (`u`,`u_b`,`u_e`),(`v`,`v_b`,`v_e`),`*opcions*`)`

opcions: `aspect_ratio=`$[1,1,1]$`, color=''red"`
`opacity=0.5, figsize=6, viewer="tachyon"`

Matemàtica Discreta

$\lfloor x \rfloor$ = `floor(x)` $\lceil x \rceil$ = `ceil(x)`

La resta de n dividit per k = `n%k` $k|n$ sii `n%k==0`

$n!$ = `factorial(n)` $\binom{x}{m}$ = `binomial(x,m)`

$\phi(n)$ = `euler_phi(`n`)`

Strings: ex. `s = "Hello" = "He"+'llo'`

`s[0]="H"  s[-1]="o"  s[1:3]="el"  s[3:]="lo"`

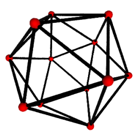
Llistes: ex. `[1,"Hello",x] = []+[1,"Hello"]+[x]`

Tuples: ex. `(1,"Hello",x)` (immutable)

Conjunts: ex. $\{1,2,1,a\}$ = `Set([1,2,1,"a"])` $(= \{1,2,a\})$

Comprensió de llistes ≈ notació constructiva per a conjunts, ex.
$\{f(x) : x \in X, x > 0\}$ = `Set([f(x) for x in X if x>0])`

Teoria de grafs



Graf: `G = Graph({0:[1,2,3], 2:[4]})`

Graf dirigit: `DiGraph(`*dictionary*`)`

Famílies de grafs: `graphs.`⟨tab⟩

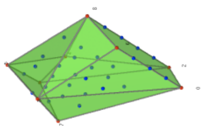
Invariants: `G.chromatic_polynomial()`, `G.is_planar()`

Camins: `G.shortest_path()`

Visualitza: `G.plot()`, `G.plot3d()`

Automorfismes: `G.automorphism_group()`,
`G1.is_isomorphic(G2)`, `G1.is_subgraph(G2)`

Combinatòria



Seqüències enteres: `sloane_find(`*list*`)`, `sloane.`⟨tab⟩

Particions: `P=Partitions(`n`)` `P.count()`

Combinacions: `C=Combinations(`*llista*`)` `C.list()`

Producte Cartesià: `CartesianProduct(P,C)`

Taula: `Tableau([[1,2,3],[4,5]])`

Paraules: `W=Words("abc"); W("abca")`

Posets: `Poset([[1,2],[4],[3],[4],[]])`

Root systems: `RootSystem(["A",3])`

Crystals: `CrystalOfTableaux(["A",3], shape=[3,2])`

Lattice Polytopes: `A=random_matrix(ZZ,3,6,x=7)`
`L=LatticePolytope(A)` `L.npoints()` `L.plot3d()`

Àlgebra de matrius

$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ = `vector([1,2])`

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ = `matrix(QQ,[[1,2],[3,4]], sparse=False)`

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ = `matrix(QQ,2,3,[1,2,3, 4,5,6])`

$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$ = `det(matrix(QQ,[[1,2],[3,4]]))`

Av = `A*v` A^{-1} = `A^-1` A^t = `A.transpose()`

Resol $Ax = v$: `A\v` o `A.solve_right(v)`

Resol $xA = v$: `A.solve_left(v)`

Forma reduïda escalonada: `A.echelon_form()`

Rang i dimensió del nucli: `A.rank()` `A.nullity()`

Forma de Hessenberg: `A.hessenberg_form()`

Polinomi característic: `A.charpoly()`

Valors propis: `A.eigenvalues()`

Vectors propis: `A.eigenvectors_right()` (també left)

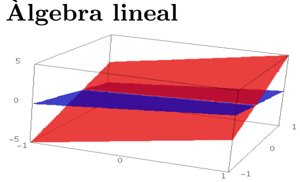
Gram-Schmidt: `A.gram_schmidt()`

Visualitza: `A.plot()`

Reducció: LLL `matrix(ZZ,...).LLL()`

Forma Hermite: `matrix(ZZ,...).hermite_form()`

Àlgebra lineal



Espais de vectors K^n = `K^n` ex. `QQ^3` `RR^2` `CC^4`

Subespai: `span(vectors, `*field*` )`

Ex., `span([[1,2,3], [2,3,5]], QQ)`

Kernel: `A.right_kernel()` (també left)

Suma i intersecció: `V + W` i `V.intersection(W)`

Base: `V.basis()`

Matriu de la base: `V.basis_matrix()`

Restringeix la matriu al subespai: `A.restrict(V)`

Vector en termes de base: `V.coordinates(`*vector*`)`

Matemàtica numèrica

Paquets: `import numpy, scipy, cvxopt`

Minimització: `var("x y z")`
`minimize(x^2+x*y^3+(1-z)^2-1, [1,1,1])`

Ajustament: `var("a b c")`

```
dadesx=range(100)
dadesy=[1.2*sin(.5*i+.1*random()) for i in dadesx]
model(x)=model(x)=a+b*sin(x+c)
find_fit(zip(dadesx,dadesy),model)
```

Teoria de nombres

Primers: `prime_range(n,m)`, `is_prime`, `next_prime`

Factor: `factor(n)`, `qsieve(n)`, `ecm.factor(n)`

Símbol de Kronecker: $\left(\frac{a}{b}\right)$ = `kronecker_symbol(`a,b`)`

Fraccions continues: `continued_fraction(x)`

Nombres de Bernoulli: `bernoulli(n)`, `bernoulli_mod_p(p)`

Corbes el·líptiques: `EllipticCurve([`a_1, a_2, a_3, a_4, a_6`])`

Caràcters de Dirichlet : `DirichletGroup(`N`)`

Formes modulars: `ModularForms(`*nivell*`, `*pes*`)`

Símbols modulars: `ModularSymbols(`*nivell*`, `*pes*`, `*sign*`)`

Mòduls de Brandt: `BrandtModule(`*nivell*`, `*pes*`)`

Varietats abelianes modulars: `J0(`N`)`, `J1(`N`)`

Teoria de grups

`G = PermutationGroup([[(1,2,3),(4,5)],[(3,4)]])`

`SymmetricGroup(`n`)`, `AlternatingGroup(`n`)`

Abelian groups: `AbelianGroup([3,15])`

Grups de matrius: `GL, SL, Sp, SU, GU, SO, GO`

Funcions: `G.sylow_subgroup(p)`, `G.character_table()`,
`G.normal_subgroups()`, `G.cayley_graph()`

Anells no commutatius

Quaternions: `Q.<i,j,k>= QuaternionAlgebra(a,b)`

Àlgebres Lliures: `R.<a,b,c>= FreeAlgebra(QQ, 3)`

Mòduls de Python

`import` *nom_del_mòdul*

`module_name.`⟨tab⟩ i `help(module_name)`

Perfils i debugging

`time` *commanda*: mostra la informació temporal

`timeit("command")`: mostra el temps amb més precisió

`t = cputime(); cputime(t)`: temps transcurregut de CPU

`t = walltime(); walltime(t)`: temps transcurregut de wall

`%pdb`: engega el debugger interactiu (només línia de comandes)

`%prun command`: comanda de perfil (només línia de comandes)