Introduzione a SAGE Math

Stefano Zambon

Esercitazione di Analisi Matematica I

Corsi di Laurea in Matematica Applicata e Informatica Multimediale
Università degli Studi di Verona

Marzo 2009

Sommario

Introduzione

Software matematici (alla buona):

Calcolo Simbolico

- Manipolano analiticamente espressioni matematiche
- . . . spesso non trovano la soluzione!
- $\int_0^\pi \sin(x) = \cos(0) - \cos(\pi) = 2$
- *Maple, Mathematica*

Calcolo Numerico

- Lavorano su rappresentazioni finite in virgola mobile
- Problemi di precisione numerica
- $\int_0^\pi \sin(x) \simeq 1.999998731$
- *Matlab*, C/FORTRAN + librerie

Introduzione

Software matematici (alla buona):

Calcolo Simbolico

- Manipolano analiticamente espressioni matematiche
- . . . spesso non trovano la soluzione!
- $\int_0^\pi \sin(x) = \cos(0) - \cos(\pi) = 2$
- *Maple, Mathematica*

Calcolo Numerico

- Lavorano su rappresentazioni finite in virgola mobile
- Problemi di precisione numerica
- $\int_0^\pi \sin(x) \simeq 1.999998731$
- *Matlab*, C/FORTRAN + librerie

Introduzione

Software matematici (alla buona):

Calcolo Simbolico

- Manipolano analiticamente espressioni matematiche
- . . . spesso non trovano la soluzione!
- $\int_0^\pi \sin(x) = \cos(0) - \cos(\pi) = 2$
- *Maple, Mathematica*

Calcolo Numerico

- Lavorano su rappresentazioni finite in virgola mobile
- Problemi di precisione numerica
- $\int_0^\pi \sin(x) \simeq 1.999998731$
- *Matlab*, C/FORTRAN + librerie

Introduzione

Software matematici (alla buona):

Calcolo Simbolico

- Manipolano analiticamente espressioni matematiche
- . . . spesso non trovano la soluzione!
- $\int_0^\pi \sin(x) = \cos(0) - \cos(\pi) = 2$
- *Maple, Mathematica*

Calcolo Numerico

- Lavorano su rappresentazioni finite in virgola mobile
- Problemi di precisione numerica
- $\int_0^\pi \sin(x) \simeq 1.999998731$
- *Matlab*, C/FORTRAN + librerie

Sage MATH

- `http://www.sagemath.org`
- Free Software (GPL) sviluppato inizialmente alla Washington University
- Interfaccia unica per diversi software free
- Basato su *Python*
- Molte potenzialità: algebra, calcolo combinatorio, teoria dei numeri. . .
- Noi lo useremo per calcolo simbolico elementare relativo agli argomenti del corso

Sage MATH

- `http://www.sagemath.org`
- Free Software (GPL) sviluppato inizialmente alla Washington University
- Interfaccia unica per diversi software free
- Basato su *Python*
- Molte potenzialità: algebra, calcolo combinatorio, teoria dei numeri...
- Noi lo useremo per calcolo simbolico elementare relativo agli argomenti del corso

Sage MATH

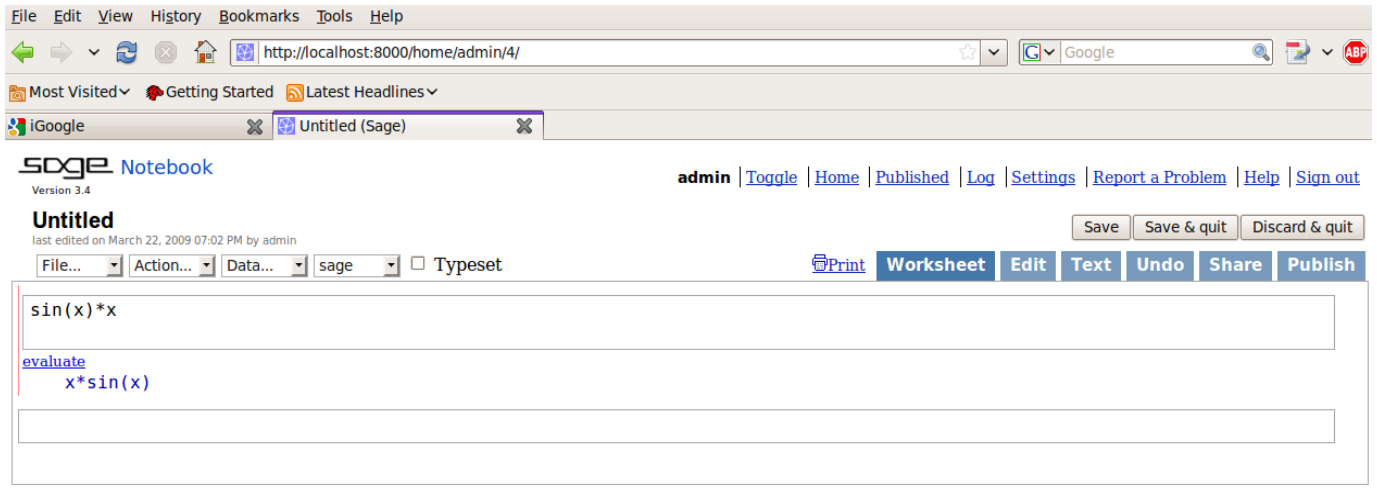
- `http://www.sagemath.org`
- Free Software (GPL) sviluppato inizialmente alla Washington University
- Interfaccia unica per diversi software free
- Basato su *Python*
- Molte potenzialità: algebra, calcolo combinatorio, teoria dei numeri...
- Noi lo useremo per calcolo simbolico elementare relativo agli argomenti del corso

Installazione

- Varie opzioni. In ordine dalla più semplice alla più difficile:
  1. Da un browser, connessione con server remoto, es. `https://sagenb.kaist.ac.kr:8022/`
  2. Scaricare un *Live CD*
  3. Distribuzione binaria per Linux ("grossa", niente pacchetti...)
  4. Dentro una macchina virtuale sotto Windows
  5. Compilazione da sorgente
- La documentazione è accessibile da: `http://www.sagemath.org/doc`
- Il "Reference Manual" mostra diversi esempi (ripresi in queste slides).

Uso

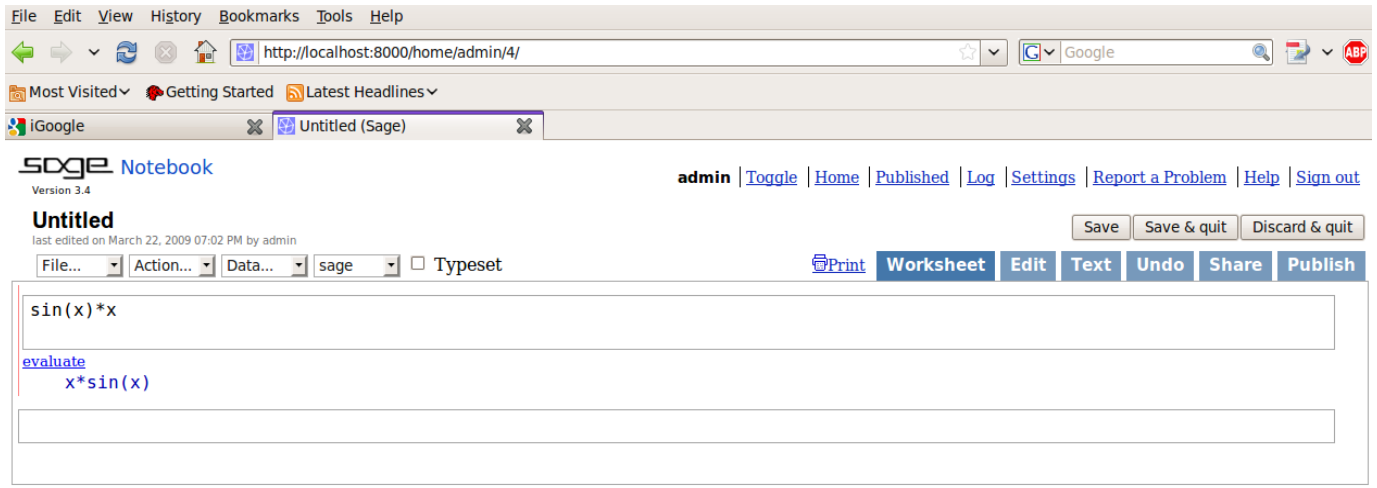
- Usiamo l'interfaccia *Notebook*:



- Nella casella di testo si possono inserire più comandi o espressioni
- Vengono valutate con il tasto "evaluate" o con <SHIFT>+<ENTER>
- È presente un help interattivo (vediamo dopo un esempio...)

Uso

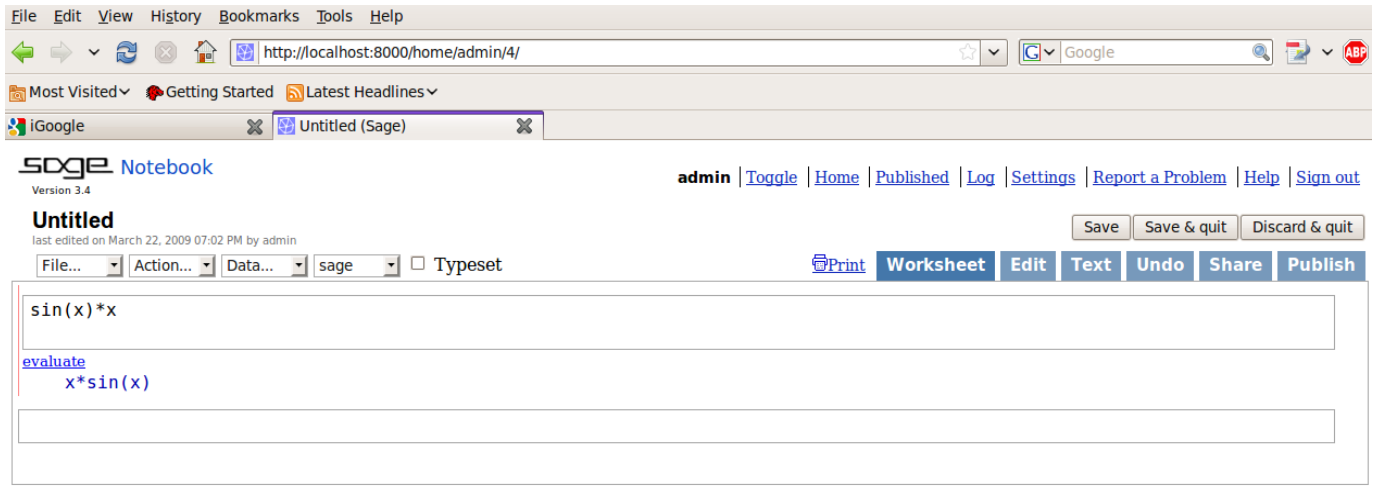
- Usiamo l'interfaccia *Notebook*:



- Nella casella di testo si possono inserire più comandi o espressioni
- Vengono valutate con il tasto "evaluate" o con <SHIFT>+<ENTER>
- È presente un help interattivo (vediamo dopo un esempio. . .)

Uso

- Usiamo l'interfaccia *Notebook*:



- Nella casella di testo si possono inserire più comandi o espressioni
- Vengono valutate con il tasto "evaluate" o con <SHIFT>+<ENTER>
- È presente un help interattivo (vediamo dopo un esempio. . .)

Variabili Simboliche

- Bisogna dichiarare in anticipo le variabili che intendiamo usare:

```
x = var('x')
```

- **Attenzione**: non sono le classiche "variabili" in un linguaggio di programmazione...
- Si possono dichiarare contemporaneamente più variabili:

```
x, y, t = var('x, y, t')
```

- La variabile `x` è predefinita nell'ambiente

Variabili Simboliche

- Bisogna dichiarare in anticipo le variabili che intendiamo usare:

```
x = var('x')
```

- **Attenzione**: non sono le classiche "variabili" in un linguaggio di programmazione...
- Si possono dichiarare contemporaneamente più variabili:

```
x, y, t = var('x, y, t')
```

- La variabile `x` è predefinita nell'ambiente

Variabili Simboliche

- Bisogna dichiarare in anticipo le variabili che intendiamo usare:

```
x = var('x')
```

- **Attenzione**: non sono le classiche "variabili" in un linguaggio di programmazione...
- Si possono dichiarare contemporaneamente più variabili:

```
x, y, t = var('x, y, t')
```

- La variabile `x` è predefinita nell'ambiente

Variabili Simboliche

- Bisogna dichiarare in anticipo le variabili che intendiamo usare:

```
x = var('x')
```

- **Attenzione**: non sono le classiche "variabili" in un linguaggio di programmazione...
- Si possono dichiarare contemporaneamente più variabili:

```
x, y, t = var('x, y, t')
```

- La variabile `x` è predefinita nell'ambiente

Espressioni

- È possibile costruire espressioni che coinvolgono le variabili dichiarate ed assegnarle ad un oggetto, es.:

```
f = (x^2) * (log(x) - 1)
```

- La sintassi per le espressioni è molto intuitiva
- Operazioni binarie : `+, -, *, /, ^`
- Funzioni predefinite : `log, cos, sin, exp ...`
- Costanti predefinite : `pi, e, I ...`

Espressioni

- È possibile costruire espressioni che coinvolgono le variabili dichiarate ed assegnarle ad un oggetto, es.:

```
f = (x^2) * (log(x) - 1)
```

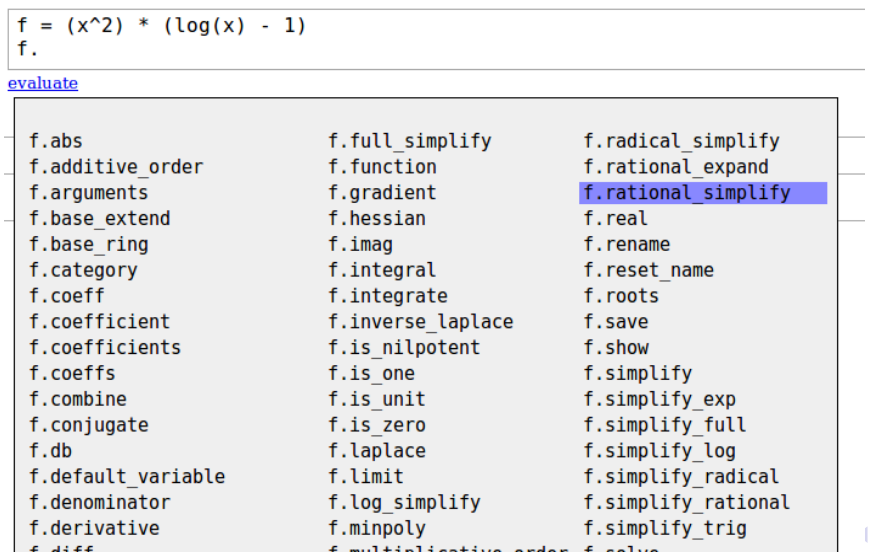
- La sintassi per le espressioni è molto intuitiva
- Operazioni binarie : `+, -, *, /, ^`
- Funzioni predefinite : `log, cos, sin, exp ...`
- Costanti predefinite : `pi, e, I ...`

Espressioni - 2

- Possiamo passare argomenti a `f` come se fosse una funzione, es. :

```
f(e^3)
      2*e^6
```

- In generale, `f` è un *oggetto*. Le operazioni possibili su `f` sono accessibili tramite la notazione punto `f.op(..)`
- È possibile vedere l'elenco delle operazioni digitando `f.` seguito da <TAB>:



Approssimazioni numeriche

- Possiamo trovare l'approssimazione numerica di una valore con l'operatore `.numerical_approx()`
- Abbreviato anche con `.n()`, es.:

```
sage:          f(e^3).n()
                  806.857586985470
```

Sostituzioni

- È possibile sostituire intere espressioni ad una variabile
- Uso pratico : comporre funzioni..
- Sintassi : `f.subs(var = expr)` Esempi:

```
sage:    t = var('t')
sage:    print f.subs(x = 3*t)
                          2
                    9 t  (log(3 t) - 1)

sage:    g = 4*t^2 - 1
sage:    print f.subs(x = g(t))
                     2      2           2
                (4 t  - 1)  (log(4 t  - 1) - 1)
```

- **Nota**: usando esplicitamente il comando print, otteniamo una formattazione più leggibile.

Semplificazioni

- Sage offre diversi comandi (anche troppi!) per semplificare espressioni.
- Grosso modo, tre classi di semplificazioni:
  1. `simplify_rational` usa regole per funzioni razionali
  2. `simplify_radical, simplify_log, exp_simplify` sono *alias* per lo stesso comando: semplificazioni usando regole per potenze, logaritmi ecc.
  3. `simplify_trig` viene usato per espressioni trigonometriche
- Esempi - `simplify_trig()` :

```
sage: f = sin(x)^2 + cos(x)^2; f
            sin(x)^2 + cos(x)^2
sage: f.simplify_trig()
            1
```

Semplificazioni - 2

- Esempi - `simplify_exp()` :

```
sage: f = (log(x+x^2)-log(x))^a/log(1+x)^(a/2)
sage: f.simplify_radical()

            log(x + 1)^(a/2)

sage: f = (e^x-1)/(1+e^(x/2))
sage: f.simplify_exp()

            e^(x/2) - 1
```

Semplificazioni - 3

- Esempi - `simplify_rational()` :

```
sage: f = ((x - 1)^(3/2)
          - (x + 1)*sqrt(x - 1))/sqrt((x - 1)*(x + 1))
sage: print f
                        3/2
                 (x - 1)    - sqrt(x - 1) (x + 1)
                -------------------------------
                     sqrt((x - 1) (x + 1))
sage: print f.simplify_rational()
                          2 sqrt(x - 1)
                        - -------------
                                 2
                          sqrt(x  - 1)
```

- **Note**:
  1. Esiste un comando `simplify_full()` che applica, nell'ordine, `simplify_trig()`, `simplify_rational()`, `simplify_radical(`
  2. Attenzione a ricordarsi le parentesi `()` ! Vale per tutti i comandi che non prendono argomenti

Sviluppi

- I comandi
  `expand_trig(), expand_rational(), expand_radical()`
  sviluppano espressioni anziché semplificarle
- Esempio:

```
sage: x,y = var('x,y')
sage: a = (x-y)^5
sage: a.expand_rational()
-y^5 + 5*x*y^4 - 10*x^2*y^3 + 10*x^3*y^2 - 5*x^4*y + x^5
```

Operazioni su polinomi

- È possibile fattorizzare un polinomio (o, in genere, un'espressione razionale) con `factor()`
- Esempio:

```
sage: f = x^6 + x^5/2 - 5*x^4 - 23*x^3/2
    - 25*x^2/2 - 7*x - 3/2
sage: print f.factor()
                          1          2   2
        (x - 3) (x + -) (x + 1)  (x  + x + 1)
                          2
```

- Per polinomi semplici, è possibile calcolare le radici esatte con `roots()`
- Esempio:

```
sage: f.roots()
    [(-1/2, 1), (3, 1), ((-sqrt(3)*I - 1)/2, 1),
    ((sqrt(3)*I - 1)/2, 1), (-1, 2)]
```

- Il formato dell'output è una lista di coppie (radice, molteplicità)

Risoluzione di equazioni

- Alcune equazioni non lineari possono essere risolte simbolicamente da Sage.
- Per le altre, esistono i comandi per metodi numerici (che non vediamo).
- Sintassi:
  `solve(x, multiplicities=False, explicit_solutions=Fal`

  1. `x` è l'incognita nell'equazione
  2. Gli altri parametri sono opzionali: se impostati a `True`, mostrano rispettivamente le molteplicità delle soluzioni e forzano soluzioni in forma esplicita
- Esempio:

```
sage: f = log((x+1)^2) - 2
sage: f.solve(x)
    [x == e - 1]
```

Derivate

- Possiamo differenziare una funzione con il comando `diff`
- Sintassi : `f.diff(var [,ord])`
- Calcola la derivata di ordine `ord` rispetto alla variabile `var`
- Esempio:

```
sage : f = log(sin(x)) / sqrt(x)
sage : print f.diff(x).simplify_full()

          2 x cos(x) - sin(x) log(sin(x))
          -------------------------------
                       3/2
                    2 x    sin(x)

sage : (sin(2*x)).diff(x,5)
        32*cos(2*x)
```

Polinomi di Taylor

- Comando: `taylor(var, cent, n)`
- Calcola il polinomio di Taylor di ordine `n` rispetto a `var` centrato in `cent`
- Esempio:

```
sage: f = arctan(log(1+x))
sage: f.taylor(x, 0, 4)
    x - x^2/2 + x^4/4
```

- Funziona anche per espansioni non polinomiali:

```
sage: f = cos(log(1 + sqrt(x)))
sage: print f.taylor(x, 0, 2)
                     2    3/2
                  5 x     x       x
                - ---- + ---- - - + 1
                   12      2       2
```

Limiti

- Comando: `limit(var = val [,taylor=True])`
- Calcola il limite per `var` tendente a `val` (usare `oo` per ∞)
- È consigliato impostare l'opzione `taylor=True`
- È possibile calcolare limiti destri e sinistri dando prima un comando `assume(expr)`, es.: `assume(x > 0)`, `assume(x < 1)`
- Esempi (tratti da esercizi-temi d'esame):

```
sage: f = ( (log(1+x^3))^2 - x^6 ) / ( (sin(2*x^3))^3 )
sage: f.limit(x=0,taylor=True)
        - 1 / 8
sage: f = ( x^2 + sin(x)*log(x) ) / ( 3*x^2 + 2*x + 5 )
sage: f.limit(x=+oo,taylor=True)
        1 / 3
```

Integrali

- Comando: `integral(var = val [,a,b])`
- Integra rispetto a `var`
- Se `a, b` sono specificati, calcola l'integrale definito tra questi estremi
- Altrimenti, trova una primitiva della funzione
- Esempi (tratti da esercizi-temi d'esame):

```
sage: print (sin(log(x))).integral(x)
          x (sin(log(x)) - cos(log(x)))
          -----------------------------
                          2
sage: f = sin(x)
sage: f.integral(x,0,pi)
            2
```

- Riesce anche a calcolare qualche integrale improprio

Plotting

- Comando: `plot(expr, xmin, xmax)`
- Disegna la funzione definita da `expr` nell'intervallo `[xmin, xmax]`

```
plot(x^2 * sin(3*x) / (2*e^(-x)), -1, 1)
```