* COSTO *

det (costo) il costo del calcolo del valore di una funzione è il numero di op elem effettuate.

Oss: def ragionevole se...

as: $x, y \in \mathbb{R}^n$; $x^T y$ costa ... n prodotti + + (n-1) somme ... costo (asintotico) $\sim 2n$.

1. Costo EGPP: $\sim \frac{2}{3} n^3$
2. Costo SI, SA: $\sim n^2$
3. Costo calcolo "buono" di fatt QR: $\sim \frac{4}{3} n^3$

⇒ costo soluz sist (fatt + SA + SI):

$\sim \frac{2}{3} n^3$ (EGPP) / $\sim \frac{4}{3} n^3$ (QR)

number_properties (determine floating-point parameters)

Calling sequence: $pr =$ `number_properties(`$prop$`)`[1]

Arguments:

prop
: string

pr
: real or boolean scalar

Description:

This function may be used to get the characteristic numbers/properties of the floating point set denoted here by $F(b, p, emin, emax)$ (usually the 64 bits float numbers set prescribed by IEEE 754 [1]). Numbers of F are of the form:

$$\texttt{sign} \cdot m \cdot b^e$$

e is the *exponent* and m the *mantissa*:

$$m = d_1 \cdot b^{-1} + d_2 \cdot b^{-2} + \cdots + d_p \cdot b^{-p}$$

the digits d_i are in $[0, b-1]$ and e in $[emin, emax]$, the number is said *normalized* if $d_1 \neq 0$. The following may be gotten:

$prop =$ `radix`
: then pr is the radix b of the set F

$prop =$ `digits`
: then pr is the number of digits p

$prop =$ `huge`
: then pr is the max positive float of F

$prop =$ `tiny`
: then pr is the min positive *normalized* float of F

$prop =$ `denorm`
: then pr is a boolean (`%t` if denormalized numbers are used)

$prop =$ `tiniest`
: then if `denorm` = `%t`, pr is the min positive *denormalized* number else $pr =$ `tiny`

[1] In queste pagine si fa riferimento alla versione 5.5.0 di Scilab. Notare la posizione della descrizione del comando nella struttura dell'*help* di Scilab (la riga immediatamente sopra il nome del comando): `Scilab Help >> Elementary Functions > Floating point > number_properties`.

prop = `eps`
then *pr* is the machine epsilon (generally $\frac{1}{2}\,b^{1-p}$ which is the relative max error between a real x such than $|x|$ in [`tiny`, `huge`] and $\mathrm{fl}(x)$, its floating point approximation in F)

prop = `minexp`
then *pr* is *emin*

prop = `maxexp`
then *pr* is *emax*

Remarks:

This function uses the *Lapack* routine `dlamch` to get the machine parameters (the names *radix*, *digits*, *huge*, etc... are those recommended by the *LIA 1* standard [2] and are different from the corresponding Lapack's ones).

⚠ **CAUTION**: sometimes you can see the following definition for the machine epsilon: $eps = b^{1-p}$ but in this function we use the traditional one (see *prop* = `eps` before) and so $eps = \frac{1}{2}\,b^{1-p}$ if normal rounding occurs and $eps = b^{1-p}$ if not.

(1) La *definizione* di *precisione di macchina* ("machine epsilon") è quella riportata sopra ("il massimo errore relativo ..."). *Il suo valore effettivo dipende dalla definizione della funzione* fl: se l'aritmetica adotta il modo di arrotondamento *round to nearest* (per ogni numero reale x il numero $\mathrm{fl}(x)$ è l'elemento di F più vicino ad x) allora la precisione di macchina vale $\frac{1}{2}\,b^{1-p}$; se, invece, l'aritmetica adotta uno qualsiasi dei restanti modi di arrotondamento prescritti dallo Standard IEEE 754-2008 (*directed rounding*[2]) allora la precisione di macchina vale b^{1-p}.

Examples:

```
-->b = number_properties("radix")
 b  =

    2.

-->eps = number_properties("eps")
 eps  =

    1.110D-16
```

See also:

- `nearfloat` – get previous or next floating-point number
- `frexp` – dissect floating-point numbers into base 2 exponent and mantissa

[2] Vedere: `http://en.wikipedia.org/wiki/IEEE_floating_point#Rounding_rules`.

Riferimenti

[1] *IEEE Std 754-2008, IEEE Standard for Floating-Point Arithmetic*, 2008.[3]

[2] *ISO/IEC 10967-1:2012, Language Independent Arithmetic, part 1: Integer and Floating Point Arithmetic*, 2012.[4]

[3] Per approfondire: `http://en.wikipedia.org/wiki/IEEE_floating_point`.

[4] Per approfondire: `http://en.wikipedia.org/wiki/ISO/IEC_10967`.

backslash (\): left matrix division

Calling sequence: `X = A\B`[1]

Description:

Backslash is the left matrix division: `X = A\B` is a solution to `A*X = B`.

(1) L'equazione $AX = B$ potrebbe avere *più di una* soluzione.

(2) Come evidente dall'esecuzione della prima parte degli esempi, l'assegnamento fornisce *un'approssimazione* di una soluzione dell'equazione.

♢

If `A` is square and non-singular, `X = A\B` is equivalent to `X = inv(A)*B` in exact arithmetic, but the computations are more accurate and cheaper in floating point arithmetic.

(3) I due assegnamenti, *quando si opera in aritmetica esatta*, sono equivalenti (posto che, in tal caso, `inv(A)` $= A^{-1}$) *ma*, quando si opera in aritmetica floating point, *non lo sono* quasi mai.

(4) Il primo assegnamento fornisce *un'approssimazione della soluzione* del sistema $AX = B$ (quasi sempre) più accurata ed ha *costo aritmetico* (quasi sempre) inferiore rispetto al secondo.

Hence, to compute the solution of the linear system of equations `A*X = B`, the backslash operator should be used, and the `inv` function should be avoided.

(5) Si ricordi che l'assegnamento fornisce *un'approssimazione* della soluzione. Ometteremo di ripetere nel seguito questa puntualizzazione.

♢

In the case where `A` is square, the solution `X` can be computed either from LU factorization or from a linear least squares solver. If the condition number of `A` is smaller than `1/(10 * %eps)` (i.e. if `A` is well conditioned), the LU factorization with row pivoting is used. If not (i.e. if `A` is poorly conditioned), then `X` is the minimum-norm solution which minimizes ‖`A*X - B`‖ using a complete orthogonal factorization of `A` (i.e. `X` is the solution of a linear least squares problem).[2]

[1] In queste pagine si fa riferimento alla versione 5.5.0 di Scilab. Notare la posizione della descrizione del comando nella struttura dell'*help* di Scilab (la riga immediatamente sopra il nome del comando): `Scilab Help >> Scilab > Scilab keywords > backslash`.

[2] Vedere [3] pag. 142–143, [2] sezioni "Driver Routines, Linear Least Squares (LLS) Problems", "Computational Routines, Orthogonal Factorizations and Linear Least Squares Problems" e [6] pag. 144–148.

(6) *(numero di condizionamento)$^{-1}$ = (misura relativa della più piccola perturbazione additiva che rende la matrice singolare, [6] pag. 62)*.

♢

If `A` is not square, `X` is a least square solution, i.e. norm(`A*X - B`) is minimal (Euclidean norm). If `A` is full column rank, the least square solution, `X = A\B`, is uniquely defined (there is a unique `X` which minimizes norm(`A*X - B`)). If `A` is not full column rank, then the least square solution is not unique, and `X = A\B`, in general, is not the solution with minimum norm (the minimum norm solution is `X = pinv(A)*B`).

(7) Sistemi di equazioni lineari con matrice *non quadrata*: teoria (vedere [1]).

(8) *Pseudoinversa di una matrice* A: definizione, uso per cercare la soluzione del sistema $Ax = b$ nel senso dei minimi quadrati di norma minima e calcolo: vedere [1].

♢

`A.\B` is the matrix with `(i,j)` entry `A(i,j)\B(i,j)`. If `A` (or `B`) is a scalar, `A.\B` is equivalent to `A*ones(B).\B` (or `A.\(B*ones(A))`).

`A\.B` is an operator with no predefined meaning. It may be used to define a new operator (see *overloading*) with the same precedence as `*` or `/.`

Examples:

```
-->A = [  9.   -36.    30.
        -36.   192.  -180.
         30.  -180.   180. ];

-->b = [  3.
        -24.
         30. ];

-->x = A\b

x  =

    1.
    1.
    1.

-->A*x - b  // close to zero

 ans  =

 10^(-14) *
```

```
  - 0.7105427
    0.
    0.
```

(9) Si constata che il vettore $(1,1,1)^{\mathsf{T}}$ è soluzione (*l'unica* – si constata che `A` è invertibile) del sistema $\mathtt{A}x = \mathtt{b}$;

(10) Come si può constatare eseguendo il comando `x == [1;1;1]`, *nessuna* delle componenti di `x` vale 1. Quindi il vettore fornito è *un'approssimazione* della soluzione, come rilevato anche dall'ultima istruzione che mostra il *residuo* associato al vettore `x`.

```
-->A = rand(3,2);

-->b = [1;1;1];

-->x = A\b;

-->y = pinv(A)*b;

-->x-y

 ans  =

 10^(-14) *

    0.1304512
  - 0.1110223
```

(11) Si considera un sistema di tre equazioni in due incognite con matrice dei coefficienti `A` generata in modo "casuale" (vedere la descrizione della funzione `rand`). Salvo casi improbabili, le due colonne di `A` sono *linearmente indipendenti* (la matrice `A` *ha* "full column rank"), dunque il sistema $\mathtt{A}x = \mathtt{b}$ ha una sola soluzione nel senso dei minimi quadrati, e `x` ed `y` sono due approssimazioni di tale soluzione.

(12) Come si constata dall'essere `x-y` $\neq 0$, le due approssimazioni sono diverse e quindi devono essere state calcolate utilizzando *procedure diverse*.[3] Si osservi che, a causa del funzionamento dell'interfaccia utente/Scilab, le istruzioni `x` (ovvero: "mostra il valore di `x`") e `y` (ovvero: "mostra il valore di `y`") generano risultati *identici*.

```
-->A = rand(2,3); b = [1;1];
```

[3] Il valore effettivo della differenza può non coincidere con quello mostrato nel testo perché la matrice `A` ha elementi "casuali."

```
-->x = A\b;

-->y = pinv(A)*b;

-->x-y

 ans  =

    0.8855461
  - 1.5367794
    0.8354957

-->A*x - b

 ans  =

 10^(-15) *

    0.2220446
    0.2220446

-->A*y - b

 ans  =

 10^(-15) *

  - 0.5551115
    0.
```

(13) Si considera un sistema di due equazioni in tre incognite con matrice dei coefficienti `A` generata in modo "casuale." Le tre colonne di `A` sono *linearmente dipendenti* (la matrice `A` *non ha* "full column rank"), dunque il sistema $\mathtt{A}x = \mathtt{b}$ ha infinite soluzioni nel senso dei minimi quadrati.

(14) Come si constata dal risultato generato dall'istruzione `x-y`, i vettori `x` ed `y` sono approssimazioni di due soluzioni nel senso dei minimi quadrati *distinte*, come esplicitato nel testo. Inoltre, si può constatare che $\mathtt{norm(x)} > \mathtt{norm(y)}$, risultato anche questo suggerito nel testo.

(15) Salvo casi improbabili, la matrice `A` ha rango due, quindi l'insieme delle soluzioni del sistema $\mathtt{A}x = \mathtt{b}$ *non è vuoto* e coincide con l'insieme delle soluzioni nel senso dei minimi quadrati. Ne segue che i vettori `x` ed `y` sono approssimazioni di due *soluzioni* (distinte) del sistema. È quindi ragionavole il risultato generato dalle istruzioni `A*x - b` e `A*y - b` che mostra come i vettori siano *poco* diversi da zero (si ricordi che *se* un vettore v è soluzione del sistema $\mathtt{A}x = \mathtt{b}$, *allora* $\mathtt{A}v - \mathtt{b} = 0$).

$\diamondsuit$

```
// if rank is deficient

-->A = rand(3,1)*rand(1,2);

-->b = [1;1;1];

-->x = A\b;
Rango insufficiente. rango = 1

-->y = pinv(A)*b;

-->A*x - b

 ans  =

    0.2519766
  - 0.0752298
  - 0.5640432

-->A*y - b

 ans  =

    0.2519766
  - 0.0752298
  - 0.5640432
```

(16) Si considera un sistema di tre equazioni in due incognite con matrice dei coefficienti `A` generata moltiplicando una colonna con tre componenti "casuali" ed una riga con due componenti "casuali." Le due colonne di `A` sono *linearmente dipendenti* (la matrice `A` *non ha* "full column rank"), dunque il sistema $\mathtt{A}x = \mathtt{b}$ ha infinite soluzioni nel senso dei minimi quadrati.

(17) Come si può constatare eseguendo l'istruzione `x-y`, i vettori `x` ed `y` sono approssimazioni di due soluzioni nel senso dei minimi quadrati *distinte*, come esplicitato nel testo. Inoltre, si può constatare che $\mathtt{norm(x)} > \mathtt{norm(y)}$, risultato anche questo suggerito nel testo.

(18) Salvo casi improbabili, la matrice `A` ha rango uno (come correttamente rilevato dall'istruzione `A\b`) e l'insieme delle soluzioni del sistema $\mathtt{A}x = \mathtt{b}$ *è vuoto*. Infatti, le istruzioni `A*x - b` e `A*y - b` mostrano come il residuo sia, in entrambi i casi, *decisamente* diverso da zero. Si osservi, però, che se v e w sono soluzioni del sistema $\mathtt{A}x = \mathtt{b}$ nel senso dei minimi quadrati allora: $\mathtt{A}v = \mathtt{A}w =$ *la* proiezione ortogonale di `b` sullo spazio generato dalle colonne di `A` e quindi i vettori $\mathtt{A}v - \mathtt{b}$ e $\mathtt{A}w - \mathtt{b}$ sono *uguali*. È quindi ragionevole che l'interfaccia utente/Scilab mostri i vettori `A*x - b` e `A*y - b` identici. In effetti i due vettori *non* sono uguali, come si può constatare eseguendo l'istruzione `(A*x - b) - (A*y - b)`, perché, si ricordi, `x` ed `y` sono *approssimazioni* di due soluzioni nel senso dei minimi quadrati.

```
-->A = rand(2,1)*rand(1,3);

-->b = [1;1];

-->x = A\b;
Rango insufficiente. rango = 1

-->y = pinv(A)*b;

-->A*x - b

 ans  =

  - 0.1684007
    0.1245334

-->A*y - b

 ans  =

  - 0.1684007
    0.1245334
```

(19) Si considera un sistema di due equazioni in tre incognite con matrice dei coefficienti `A` generata moltiplicando una colonna con due componenti "casuali" ed una riga con tre componenti "casuali." Le tre colonne della matrice sono *linearmente dipendenti* (la matrice `A` *non ha* "full column rank"), dunque il sistema $\mathtt{A}x = \mathtt{b}$ ha infinite soluzioni nel senso dei minimi quadrati.

(20) Come si può constatare eseguendo l'istruzione `x-y`, i vettori `x` ed `y` sono approssimazioni di due soluzioni nel senso dei minimi quadrati *distinte*, come esplicitato nel testo. Inoltre, si può constatare che `norm(x)` $>$ `norm(y)`, risultato anche questo suggerito nel testo.

(21) Salvo casi improbabili, la matrice `A` ha rango uno (come correttamente rilevato dall'istruzione `A\b`) e l'insieme delle soluzioni del sistema $\mathtt{A}x = \mathtt{b}$ *è vuoto*. Infatti le istruzioni `A*x - b` e `A*y - b` mostrano come il residuo sia, in entrambi i casi, *decisamente* diverso da zero. Anche in questo caso l'interfaccia utente/Scilab mostra i vettori `A*x - b` e `A*y - b` identici anche se in effetti *non lo sono*, come si può constatare eseguendo l'istruzione `(A*x - b) - (A*y - b)`.

$\diamondsuit$

```
// A benchmark of several linear solvers

-->[A,descr,ref,mtype] = ReadHBSparse(SCI + ...
              "/modules/umfpack/examples/bcsstk24.rsa");

-->b = zeros(size(A,1),1);
```

```
-->tic();
   res = umfpack(A,'\',b);
   mprintf('\nTime with umfpack: %.3f\n',toc());

Time with umfpack: 0.081

-->tic();
   res = linsolve(A,b);
   mprintf('\nTime with linsolve: %.3f\n',toc());

Time with linsolve: 48.661

-->tic();
   res = A\b;
   mprintf('\nTime with backslash: %.3f\n',toc());

Time with backslash: 7.971
```

(22) Si considera il sistema (omogeneo!) $\mathtt{A}x = \mathtt{b}$ con `A` matrice 3562×3562 ad elementi reali, *sparsa*. Quest'ultimo termine si riferisce al *tipo* di struttura di `A` (*matrice sparsa*, appunto). Se M è una matrice (più tecnicamente: un *oggetto di tipo matrice*), Scilab ne memorizza *tutte* le componenti e tutte le manipolazioni di M avvengono in modo "ovvio." Invece, se M è una matrice sparsa (un *oggetto di tipo matrice sparsa*), Scilab memorizza soltanto *la posizione* (riga e colonna) e *il valore* di ciascun elemento *non nullo*, e le manipolazioni di M sono eseguite utilizzando procedure *ad hoc* per matrici sparse. L'uso della struttura matrice sparsa è tanto più vantaggioso (rispetto alla struttura matrice) quanto più elevata è la percentuale di elementi nulli in M. Per ulteriori informazioni sulle matrici sparse in Scilab vedere [4]. La matrice sparsa è creata dall'istruzione `ReadHBSparse`, che legge la matrice dal file specificato. Per dettagli su come la matrice è codificata nel file, vedere [5].

(23) Si determinano approssimazioni della soluzione del sistema con *tre* procedure differenti e si misurano *i tempi* di esecuzione. La prima approssimazione è ottenuta dall'istruzione `umfpack(A,'\',b)`. Come specificato nella descrizione, questa istruzione utilizza una procedura *specifica* per sistemi con matrice sparsa. La seconda approssimazione è ottenuta dall'istruzione `linsolve(A,b)`. Questa istruzione, come specificato nella descrizione, *può* operare su sistemi con matrice sparsa e, in tal caso, utilizza procedure specifiche.[4] Si osservi che la funzione determina *l'intera struttura* dell'insieme delle soluzioni del sistema, un compito *di complessità più grande* rispetto alla sola determinazione di una soluzione. Infine, la terza approssimazione è ottenuta dall'istruzione `A\b`. Nella descrizione precedentemente riportata dell'istruzione, *non si fa cenno* alla possibilità che essa possa operare su matrici

[4] Per approfondire, vedere il codice sorgente della funzione (tecnicamente una *compiled macro*), accessibile con l'istruzione `editor(get_function_path('linsolve'))`, [3], Capitolo 4.

sparse (cosa, invece, esplicitamente menzionata nella descrizione della funzione `linsolve`). Il risultato ottenuto indica come la funzione *sia in grado* di operare su matrici sparse (non viene generato alcun errore a causa del *tipo* di operandi) e, coerentemente con quanto riportato nella Sezione 3.2 di [4], che anch'essa utilizza procedure specifiche per sistemi con matrice sparsa.[5]

(24) I tempi riportati, in *secondi*, corrispondono al tempo di volta in volta trascorso tra l'esecuzione dell'istruzione `tic()` e l'esecuzione dell'istruzione `toc()` (vedere la descrizione delle istruzioni). Si osservi che ripetendo le istruzioni ciascuno dei tempi potrà risultare *diverso*.

Riferimenti

[1] Aceto L. e Ciampa M, *Decomposizione ai valori singolari*, 2009.[6]

[2] Anderson E. ecc, *LAPACK Users' Guide*, Third Edition, 1999.[7]

[3] Baudin M, *Programming in Scilab*, 2011.[8]

[4] Baudin M, *Introduction to sparse matrices in Scilab*, 2011.[9]

[5] Duff I, *User's guide for the Harwell-Boeing Sparse Matrix Collection*, 1992.[10]

[6] Lambers J, *MAT 610: Numerical Linear Algebra*, 2014.[11]

[5] Per l'operatore `backslash` è definita una *overloading function*: `%sp_l_sp` (vedere la descrizione della funzione `overloading` per capire l'origine dello strano *nome* della funzione). Il codice sorgente (accessibile con l'istruzione `editor(get_function_path('%sp_l_sp'))`, conferma quanto espresso nella Sezione 3.2 di [4].

[6] `http://users.dma.unipi.it/ciampa/Didattica_10-11/x-AlgLinFondGeo_Materiale.php`

[7] `http://www.netlib.org/lapack/lug/lapack_lug.html`

[8] `http://forge.scilab.org/index.php/p/docprogscilab/downloads/`

[9] `http://forge.scilab.org/index.php/p/docscisparse/downloads/`

[10] `http://www.cs.colostate.edu/ mroberts/toolbox/c++/sparseMatrix/hbsmc.pdf`

[11] `http://www.math.usm.edu/lambers/mat610/book610.pdf`