

## Uso di librerie preconfezionate

Cosa sono:

- pezzi di programma già scritto da altri; di solito ben collaudati;
- comprendono le definizioni delle funzioni e il codice eseguibile;
- per librerie di pubblico dominio è disponibile anche il sorgente;

Diponibili pubblicamente sono

- GNU scientific libraries (GSL)
- CERNLIB
- Netlib
- LAPACK (matrici)
- fftw (trasformate di Fourier)
- e molto altro ancora...

## Cosa calcolano

- costanti fondamentali
- funzioni speciali
- polinomi
- radici
- matrici
- trasformate di Fourier
- etc. etc.

D. Perché non l'ha detto dall'inizio? Non avrei frequentato questo corso!

R. Avete mai provato a far benzina se l'auto è elettrica? Anche chi non vuole diventare meccanico deve conoscere un poco le automobili. Questo corso è un'introduzione alla Fisica Computazionale, non un trattato!

D. Io però voglio occuparmi di fisica e non di programmazione. Ho già progettato di scoprire il settimo quark

R. Per gli esperimenti di fisica delle alte energie è necessario avere dei buoni programmi di simulazione Montecarlo. Sono di solito non-standard, e ve li dovete quindi scrivere da soli.

D. Nel mio gruppo di ricerca fanno calcoli perturbativi all'ordine  $3/2$ . Nel corso non ne ho sentito parlare

R. Ora che conoscete i rudimenti del mestiere, potete studiare da soli le equazioni differenziali stocastiche o i problemi che più vi interessano

Cosa richiedono:

- inclusione di un file d'intestazione. Se la libreria ci chiama *gsl* si includerà [gsl.h](#) e/o altri file d'intestazione all'interno del file sorgente, per definire le funzioni che usiamo;
- link con opportune librerie extra [-lgsl](#). il corrispondente file si chiamerà [libgsl.la](#) o [libgsl.so](#)
- *Importante*: leggete molto attentamente la documentazione con: [gnome-help info://gsl-ref](#).

## Un esempio: le librerie gsl

- GNU Scientific Library (GNU's Not Unix);
- includono sempre `gsl.h`;
- link sempre `-lgsl` e spesso `-lgslcblas`;
- a seconda della particolare routine che si usa si devono includere altri file d'intestazione: ad esempio, se si usano generatori di numeri pseudocasuali può essere necessario includere anche `gsl/gsl_rng.h` mentre se faccio un integrale `gsl/gsl_integration.h`;
- i file d'intestazione definiscono i prototipi delle funzioni ma anche alcune costanti e molti tipi di dati.

## Programmi con le librerie gsl

Per programmare con le librerie gsl occorre ricordare bene tre cose

1. includere gli opportuni file d'intestazione: per esempio, per le costanti numeriche, oltre a `gsl.h` andrà incluso anche `gsl_math.h`
2. linkare con le librerie opportune: per molti funzioni sarà necessario non solo `-lgsl` ma anche `-lgslcblas` (attenzione al linking: statico o dinamico?)
3. attenersi alla struttura dei dati usata dalla libreria: i vettori e le matrici possono essere immagazzinati in modi particolari, che bisogna necessariamente utilizzare per usufruire delle librerie gsl.

## Esempi

- costanti matematiche

`M_PI =  $\pi$`

- funzioni elementari e funzioni speciali

`hypot(3, 4)`

`gsl_pow_4(2)` potenza

`gsl_sf_bessel_J0` funzione di Bessel cilindrica

- valutazione di polinomi

$p(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1}$

`gsl_poly_eval(c, n, x)`

- ordinamento di un vettore

`gsl_sort(v, k, n)`

`v(0), v(k), v(2k), ...`

per un vettore di n elementi

## In pratica

Includo i file header

```
#include < gsl/gsl_xxx.h >
```

```
xxx = math, poly, sort_vector
```

Linking

```
gcc -lgsl -lgslcblas programma.c (-static)
```

BLAS = Basic Linear Algebra Subroutines

Documentazione

Mi documento con info gsl guardando soprattutto gli esempi.

# Quattro esempi

## 1. Integrazione

- Il file da includere è `<gsl/gsl_integration.h>`
- è necessario fornire dello spazio “extra” di lavoro usato dalla routine
- la funzione da integrare è parte di una struttura
- la funzione può dipendere da altri parametri oltre alla variabile su cui integro. Devo passare questa informazione alla funzione in qualche modo: in pratica si mettono tutti i parametri in un vettore e si passa un puntatore ad esso.

## 2. Sorting

È il più semplice: occorre solo passare la funzione che fa il confronto tra elementi da ordinare (vedi anche “qsort” nella libreria standard). Il file da includere è `<gsl/gsl_sort_double.h>`

## 3. Generazione di numeri pseudocasuali

- Il file da includere è `<gsl/gsl_rng.h>`.
- il programma fornisce una funzione che ritorna un numero pseudorandom tra zero e uno. Il metodo da usare è determinato dall'utente che deve fornire il tipo con la chiamata ad una opportuna funzione di inizializzazione.
- come per i numeri Fibonacci lagged, e' necessario avere a disposizione della memoria extra. Un'apposita funzione si occupa di allocarla: chiaramente la quantità di memoria allocata dipenderà dal tipo di generatore usato.

## 4. Autovalori e autovettori

- Il file da includere è `<gsl/gsl_eigen.h>`;
- è molto importante qui il modo scelto dai programmatori delle gsl per memorizzare vettori e matrici; si usano strutture in linguaggio C dove, se  $m$  è la matrice,  $m \rightarrow data$  punta all'array che contiene gli elementi dell'array. Allo stesso modo per un vettore  $v$  si ha che  $v \rightarrow data$  punta agli elementi dell'array. Altri elementi della struttura contengono la dimensione dell'array, etc.
- anche qui occorre allocare memoria sia per il workspace che per vettori e matrici. Per gli uni e le altre ci sono funzioni dedicate che tengono conto della loro struttura particolare.
- Alla fine del programma è naturalmente possibile deallocare questa memoria con altre funzioni dedicate.