Librerie preconfezionate

- Sono routine già scritte da altri, di solito ben collaudate;
- comprendono le definizioni delle funzioni e il codice eseguibile;
- Per librerie di pubblico dominio è disponibile anche il sorgente;
- Richiedono l'inclusione di altri file d'intestazione, come math.h all'interno del file sorgente
- È necessario linkare con opportune librerie extra come libm o libgsl
- Bisogna leggere molto attentamente la documentazione.

Un esempio: le librerie gsl

- GNU Scientific Library (GNU's Not Unix);
- includono file d'intestazione supplementari come gsl.h, gsl_matrix.h o gsl_sort.h e spesso altri file;
- il linking dei programmi va fatto con le opportune librerie, ovvero con le opzioni -lgsl, -lgslcblas. . . ;
- i nomi delle funzioni non possono essere scelti dal programmatore (ma vedremo che questo si può aggirare);
- le librerie sono scritte in linguaggio "C", non "C++", il che rende tutto più complicato.

Ma perché allora ho seguito questo corso?

- per guidare l'auto bisogna saper almeno cambiare una ruota;

- un programma che metta insieme cose sconosciute è facilmente sbagliato;

- Un algoritmo standard non va bene per la maggior parte delle applicazioni; i risultati originali hanno bisogno di programmi originali

- anche programmi con funzioni preconfezionate possonmo essere "adattati" al nostro modo di programmare.

Cosa si fa con le librerie gsl

- Valori di costanti matematiche
  $\texttt{M\_PI} = \pi$

- calcolo di funzioni elementari e funzioni speciali
  $\texttt{hypot}(3, 4) \quad \texttt{gsl\_pow\_4}(2)$

- valutazione di polinomi
  $p(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$
  $\texttt{gsl\_poly\_eval}(c, n, x)$

- ordinamento di un vettore
  $\texttt{gsl\_sort}(v, k, n) \quad v_0, v_k, v_{2k}, \ldots$

- ricerca di autovalori e autovettori di una matrice

- integrali

- generatori di numeri random

- . . . e molto altro ancora!

Inclusione e linking

- Oltre ai due file che vanno inclusi sempre, ogni categoria di routine richiede l'inclusione di altri file d'intestazione

- anche la compilazione è differente a seconda delle routine utilizzate

- infine, una data routine o categoria di routine può usare strutture dati proprie e aree di memoria proprie, da usare come spazio di lavoro, che devono essere allocate dal programmatore.

Esempi

- Se voglio generare numeri random, devo includere gsl/gsl_rng.h e usare le strutture gsl_rng e gsl_rng_type
- Il linking viene fatto collegando libgsl e, spesso, libgslcblas e libm. Un comando che dovrebbe funzionare sempre è quindi

  gcc -lgsl -lgslcblas -lm programma.c

- BLAS = Basic Linear Algebra Subroutines è una collezione di funzioni che permette di fare in modo ottimale alcune operazioni su vettori e matrici, come il prodotto di un vettore per uno scalare o la somma di due vettori;
- libm sono le librerie matematiche che includono funzioni trigonometriche, radici quadrate, potenze, etc.

Esempio: classe per i vettori

- Voglio usare le Gsl per la mia classe di vettori;
- Gsl è scritta in C, devo inserirla in un programma C++;
- i vettori hanno una propria struttura gsl_vector, che contiene un membro "data" con gli elementi del vettore e un membro "size" con la sua dimensione: devo includere un puntatore a questa struttura tra le variabili private;
- in "C" dovrei allocare esplicitamente lo spazio per il vettore con gsl_vector_alloc, in C++ posso farlo automaticamenmte inserendo questa funzione nel costruttore;
- se voglio scrivere la funzione get, che restituisce un elemento di un vettore uso la funzione analoga della libreria Gsl definita da gsl_vector_get(v,i) che restituisce l'elemento i-esimo;

Strutture dati, argomenti delle funzioni e workspace

- Le routine scritte da altri utilizzano strutture dati proprie: il programmatore deve adattarsi

- Le funzioni utilizzano parametri di chiamata propri e parametri di uscita per verificare esattezza e precisione del risultato

- Alle funzioni serve anche uno spazio di lavoro temporaneo per velocizzare il programma: questo spazio deve essere allocato dall'utente

Esempio: integrazione con quadratura gaussiana

gsl_integration_qag(&F, a,b,epsabs,epsrel,limit, key,workspace, & result, & abserror)

- F è una struttura che contiene la funzione integranda,
- a e b sono gli estremi di integrazione,
- epsabs è l'errore assoluto richiesto, epsrel l'errore relativo richiesto,
- key è collegato al numero di punti di integrazione gaussiana (va da 1 a 6 con numero crescente).

- L'algoritmo consiste nel dividere l'intervallo iniziale in sotto intervalli, i cui estremi sono memorizzati nel workspace. il massimo numero di intervalli è limit.

- Workspace è una struttura di tipo gsl_integration_workspace e ha proprie funzioni di allocazione e deallocazione

- F è una struttura gsl_function che ha due membri: il primo è la funzione integranda f, il secondo un array di parametri da passare alla funzione f oltre a quello da integrare

Esercizi

- Scrivere l'algoritmo di Wolff con le Gsl e provarlo con diversi generatori di numeri

- integrare delle equazioni differenziali con le Gsl