Metodi Computazionali della Fisica

26 gennaio 2024

Parte I
Introduzione

1 Obbiettivi del corso

Fin dal primo anno di corso in Fisica , o persino dal liceo, ci si accorge che lo studio della fisica presenta delle equazioni per la descrizione di fenomeni naturali, che possono essere anche differenziali o differenziali alle derivate parziali, e che in molti casi non porgono soluzioni analitiche. Per soluzione analitica intendiamo una formula matematica immediatamente valutabile.

Un buon esempio è dato dall'integrazione delle equazioni del moto del pendolo semplice (vedi Fig. 1):

$$\frac{d^2\theta\left(t\right)}{dt^2}+\frac{g}{l}\sin\left(\theta\left(t\right)\right)=0 \tag{1}$$

con le condizioni iniziali :

$$\begin{cases}\theta(t=0)=\theta_0\\ \frac{d\theta}{dt}\left(t=0\right)=0\end{cases} \tag{2}$$

noi sappiamo risolvere analiticamente Eq. 1 solo nel caso limite di piccolo θ_0 in tal caso espandendo in serie di Taylor e approssimando:

$$\sin\left(\theta\left(t\right)\right)\approx\theta\left(t\right) \tag{3}$$

andiamo a trovare la nota formula del moto armonico:

$$\theta\left(t\right)=\theta_o\cos\left(\sqrt{\frac{g}{l}}t\right) \tag{4}$$

Anche se l'Eq.1 non ammette nel caso generale soluzioni analitiche possiamo calcolare *numericamente* soluzioni particolari come, ad esempio, il periodo di oscillazione o l'angolo ad un instante di tempo dato. In principio tali soluzioni numeriche posso essere ottenute con precisione arbitraria. In pratica la precisione sarà limitata dei mezzi di calcolo disponibili.

Altri esempi fra quelli probabilmente già ben noti ai partecipanti al corso sono l'integrazioni delle equazioni del moto per un sistema di $N\geqslant 3$ corpi interagenti fra di loro o il calcolo del campo elettrico per una distribuzione di carica non banale.

In questo corso impareremo a risolvere le equazioni della fisica usando metodi numerici. E' doveroso notare che tali metodi sono stati introdotti ed ampiamente usati ben prima dell'avvento dei computer. Un esempio può essere l'uso dell'espansione di serie di Taylor per valutare le funzioni trigonometriche. Oggi però tali metodi numerici vengono sempre *implementati* in un software e i calcoli vengono svolti dai computers. Pertanto è opportuno parlare di *metodi computazionali* piuttosto che di metodi numerici.

Si tenga presente che l'utilizzo dei computers nella fisica non è limitato al pur vasto problema della soluzione di equazioni ma riguarda anche il largo settore dei sistemi di acquisizioni dati in laboratorio, della loro analisi, trattamento e visualizzazione. Essendo questo un corso introduttivo esso sarà focalizzato quasi esclusivamente sulla prima parte.

2 Il computer

2.1 Numeri in virgola mobile

Anche se questo corso presuppone delle basi di informatica, è opportuno trattare in forma un po' dettaglita gli aspetti del computer che influenzano maggiormente l'implementazione e l'uso dei metodi computazionali per la fisica. Un computer

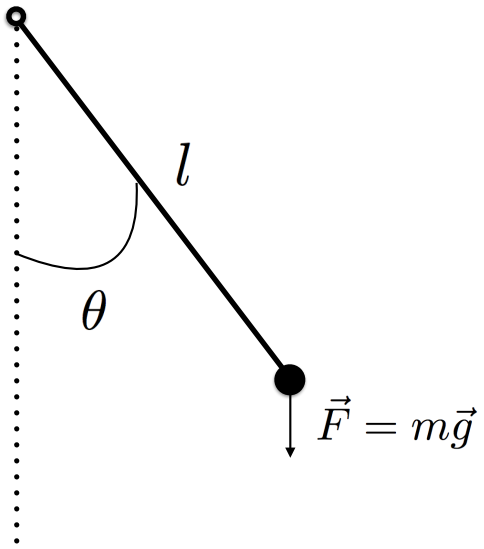


Figura 1: Il ben noto pendolo semplice

opera con dati espressi come serie di ***byte*** ogni *byte* è composto da 8 ***bit***. Un bit è una cifra in un sistema binario e pertanto può assumere solo due valori che vengono contrassegnati 0 e 1. Pertanto un byte può assumere $2^8 = 256$ valori diversi. Per rappresentare valori numerici i computers ricorrono alla rappresentazione in virgola mobile, come in:

$$f = \pm 0.c_1 c_2 c_3 c_4 c_5 \cdot 10^{\pm d_1 d_2} \tag{5}$$

dove le cifre $c_1 \dots c_5$ sono cifre da 0 a 9 che formano la *mantissa* e le cifre $d_1\, d_2$ individuano l'esponente. Al giorno d'oggi la rappresentazione in virgola mobile è regolata dallo standard **IEEE754**. Esso prevede in particolare due formati per i quali le operazioni fondamentali come comparazione, somma, sottrazione, moltiplicazione, sono implementate a *livello hardware*. Essi sono la **singola** e la d**oppia precisione**. Un numero in virgola mobile in singola precisione viene immagazzinato con 4 bytes, 8 bytes invece occorrono per la doppia precisione.

Nel caso di un numero a singola precisione uno dei 32 bits viene usato per immagazzinare il segno (+ o -) 8 bits per l'esponente e i restanti 23 per la mantissa. Questo vuol dire che passando alla rappresentazione decimale la mantissa contiene da 6 a 9 cifre (significative) e, attenzione, l'esponente varia da -126 a +127. Questo perché -127 e +128 vengono usati per contrassegnare quando il risultato di un'operazione matematica è anomalo: **underflow** per un numero il cui valore assoluto è più piccolo del minimo valore rappresentabile, + o - **infinity** per un numero il cui valore assoluto è maggiore del massimo valore rappresentabile, **NaN (not a number)** per il risultato di operazioni come divisione per zero.

Un numero a doppia precisione riserva 1 dei sui 64 bits per il segno 11 bits per l'esponente e i restanti 52 bits per la mantissa. Ciò risulta in 15-17 cifre significative decimali e un esponente da -1022 a +1023 oltre alla trattazione dei casi particolari vista prima.

2.2 Schema di base di un computer

Vogliamo ora passare in rassegna gli elementi di un computer che rivestono particolare importanza nel calcolo numerico soprattutto in termini di velocità di calcolo. Escludendo memorie di massa (hard-disk,ssd) scheda grafica, scheda di rete, ci concentriamo sulla memoria e la **CPU (central processing unit)** delle quali riportiamo in Fig. 2 riportiamo uno schema molto semplificato. La CPU si occupa di eseguire le istruzioni (*control unit)* operando su appositi registri dove vengono immagazzinati i dati che vengono processati nella *ALU* (unità logico aritmetica) e nella *FPU (floating point unit)* per i dati in virgola mobile. La memoria *RAM* è connessa alla CPU tramite un *memory bus.* La CPU dispone anche di una *cache memory*, memoria RAM ad accesso/lettura e scrittura rapida.

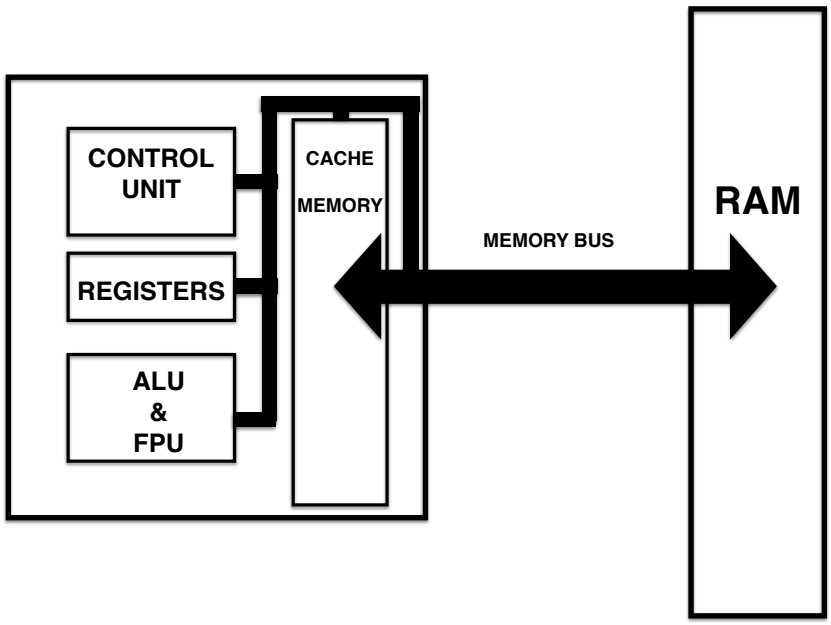


Figura 2: Schema delle connessioni tra CPU e RAM in computer

Una CPU moderna riesce a eseguire almeno un' operazione elementare su un numero a doppia precisione per ciclo di clock. Ciò significa che se un processore ha una «velocità» di 3GHz esso in teoria può eseguire almeno $3 \cdot 10^9$ operazioni in virgola mobile al secondo. La velocità di un computer può essere espressa in FLOPS ossia il numero di operazioni in virgola mobile per secondo. Quindi se ho un processore a 3 GHz mi aspetterei almeno 3 GFLOPS. Le situazione però non è sempre così favorevole, molto spesso i nostri calcoli richiedono il trasferimento di dati dalla RAM alla CPU. Si pensi ad esempio alla moltiplicazione di due matrici quadrate grandi. La velocità di tale trasferimento è limitata dalla velocità o larghezza di banda (**bandwidth)** del memory bus anch'essa è espressa in Hz ed una velocità , ad esempio, di 1.6 GHz significa che in un secondo vengono trasferiti $1.6 \cdot 10^9$ bytes, quindi in un secondo riusciamo a trasferire da memoria a CPU solo $1.6 \cdot 10^9/8 = 0.2 \cdot 10^9$ numeri in doppia precisione inoltre va considerato che nell'accedere a zone di RAM non attigue si verifica una latenza (tempo di accesso) dell'ordine dei ~ 10 ns. Questo determinerebbe un crollo notevole delle capacità di calcolo del nostro computer. Per rimediare a questo problema la CPU contiene una speciale memoria RAM , la cache memory, il cui contenuto è immediatamente utilizzabile (tipicamente in 1 ciclo di *clock)* dal resto della CPU. Quindi i nostri programmi per operare al meglio devono trasferire blocchi di dati dalla RAM alla cache per venire poi elaborati.

Da circa una decina d'anni ha preso piede la tecnologia a **molti core**: il singolo *microprocessore* contiene più di una CPU (*core*) , tipicamente da 2 a 8 per i personal computers/laptops, e fino a 64 o più per sistemi di calcolo avanzato. Come si vedi in Fig. 3 abbiamo una cache memory comune a tutti i cores più una o due cache memory interne a ciascun core. Lo sviluppo di tale tecnologia rende vantaggioso l'utilizzo di tecniche di programmazione i*n parallelo* in cui un solo programma è elaborato da più cores contemporaneamente.

Il calcolo parallelo è ormai il paradigma su si basano tutte le machine di calcolo ad alte prestazioni (*high performance computing*) . In esse più *nodi* ciascuno contente uno o più microprocessori a molti core e della memoria RAM sono interconnessi tra di loro.

Recentemente sono state sviluppate schede di calcolo esterne al processore derivate dalle schede grafiche. Tali schede sono denominate GP-GPU (general purpose graphical processing unit) o semplicemente GPU. La CPU scambia dati con la GPU attraverso una connessione (databus) ,solitamente di tipo PCI, che permette una larghezza di banda minore del memory bus. La GPU è strutturata come una CPU con moltissimi cores di calcolo (fino a qualche migliaia) che hanno accesso veloce ad una memoria RAM propria della scheda. La frequenza di clock dei cores della GPU non è, tipicamente, elevata come come quelli della CPU. Pertanto i programmi devono essere scritti in maniera da trasferire sulla GPU le parte che sono ben parallelizzabile.

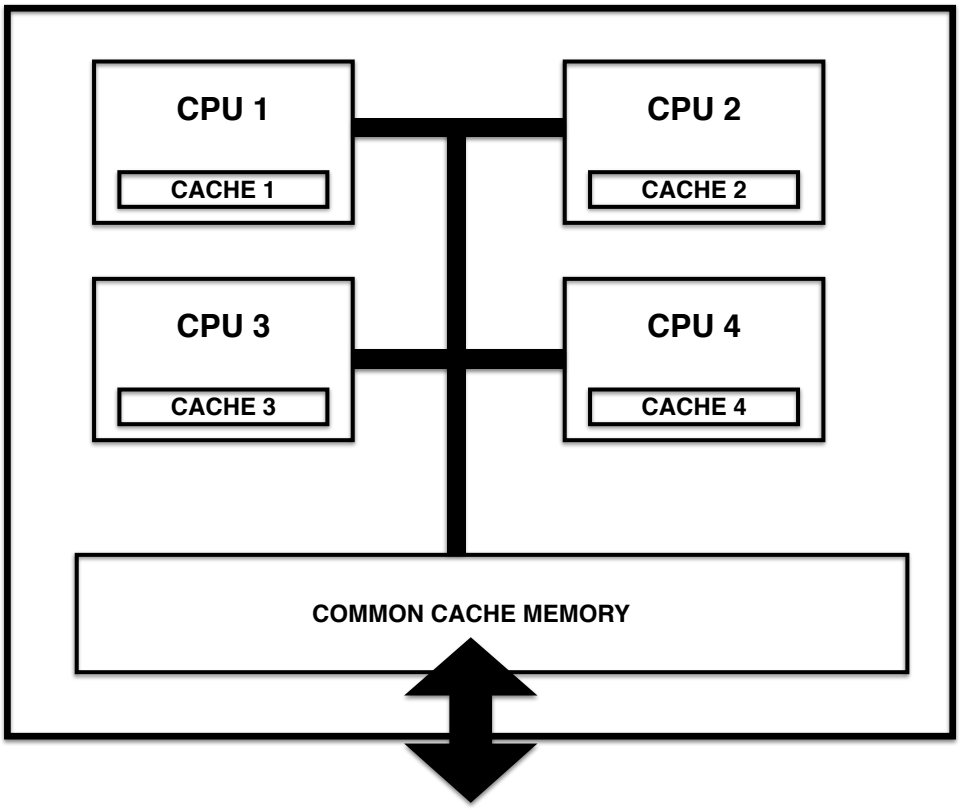


Figura 3: Schema di un microprocessore a 4 cores

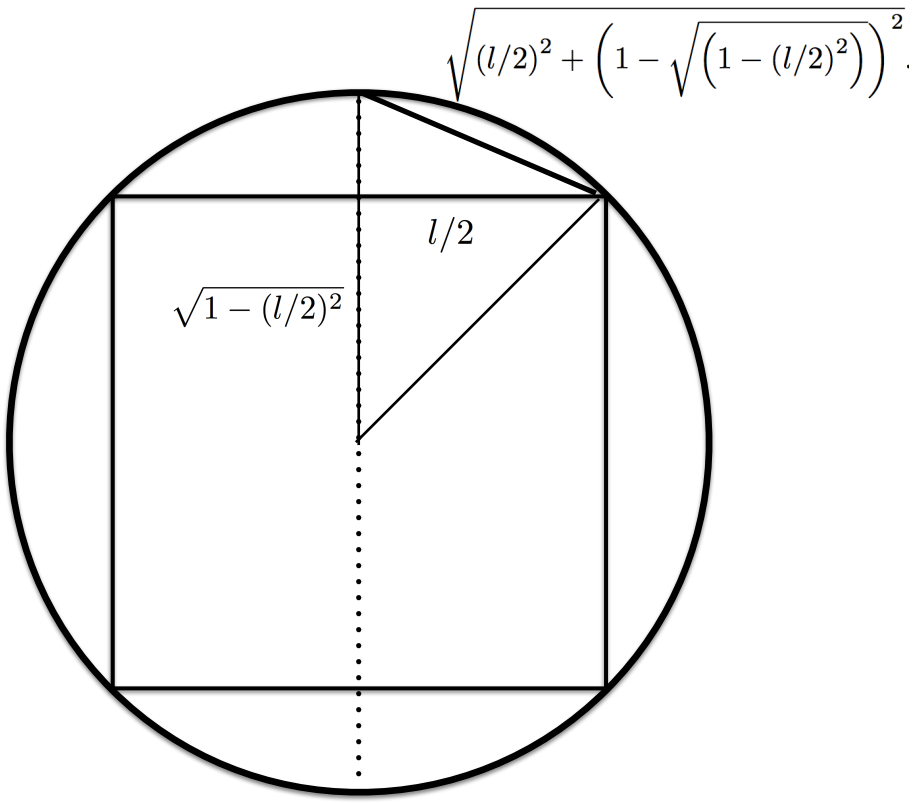


Figura 4: Calcolo del lato di un poligono regolare di numero di lati doppio al poligono regolare di lato l

Attualmente in Italia il sistema di calcolo più potente disponibile per la ricerca, Marconi-100 (www.cineca.it) consta di 980 nodi interconnessi tra di loro ciascuno con 32 cores, 256 GB di mermoria RAM e 4 schede GP-GPU NVIDIA A100.

2.3 Operazioni in virgola mobile: possibili problemi

Quando si lavora con numeri in virgola mobile possono presentarsi problemi (**roundoff** errors**)** dovuti al numero limitato di cifre significative disponibili. Il fatto di avere un limite di cifre significative implica che i nostri numeri sono definiti entro un *range* di valori possibili.

Quindi la sottrazione di due numeri molto vicini tra di loro (NB l'esponente è lo stesso) potrebbe essere calcolata con un notevole errore relativo.

Inoltre è importante considerare l'ordine con cui si eseguano le operazione matematiche, supponiamo di essere in doppia precisione e di calcolare:

1)$a = 1.0 + 1.0 \cdot 10^{-18} = 1.0$

2)$b = a - 1.0 = 0.0$

mentre se scambiamo l'ordine delle somme otteniamo:

1)$a = 1.0 - 1.0 = 0.0$

2)$b = a + 1.0 \cdot 10^{-18} = 1.0 \cdot 10^{-18}$

In taluni casi tali differenze possono portare ad esiti catastrofici. Tale è il caso del calcolo del valore di pigreco π come limite del perimetro di poligoni regolari inscritti in una circonferenza di raggio unitario.

Se la misura del lato di un poligono regolare di n lati è l, il teorema di Pitagora da per il lato l' del poligono di $2n$ lati: $l' = \sqrt{(l/2)^2 + \left(1 - \sqrt{\left(1 - (l/2)^2\right)}\right)^2}$. Vediamo cosa succede se implementiamo tale algoritmo usando:

a) formula CIOFECA $l' = \sqrt{2 - 2\sqrt{\left(1 - (l/2)^2\right)}}$ (dove abbiamo espanso il quadrato dentro la radice quadrata).

oppure

b)formula BUONA $l' = \sqrt{(l/2)^2 + \left(1 - \sqrt{\left(1 - (l/2)^2\right)}\right)^2}$

Usando la formula vediamo che per l molto piccolo la formula a) dà $l' = 0$ mentre la formula b) dà $l' = l/2$. Pertanto usando b) il calcolo del pigreco convergerà ad un valore vicino nel limite del roundoff error al valore esatto mentre a) darà risultati totalmente instabili. Ecco i risultati del metodo ciofeca in singola precisione:

```
1 3.06147
2 3.121444463729858398437 5
3 3.13654613494873046875
4 3.140333414077758789062 5
5 3.141285657882690429687 5
6 3.141518831253051757812 5
7 3.141207933425903320312 5
8 3.14245128631591796875
9 3.14245128631591796875
10 3.162277698516845703125
11 3.162277698516845703125
12 2.828427076339721679687 5
13 0
14 0
```

Questi invece sono i risultati del metodo buono ,

```
1 3.06146745892071869832307129399850965
2 3.12144515225805285751903284108266234
3 3.13654849054593976021010348631534725
4 3.14033115695475339990139218571130186
5 3.14127725093277332391039635695051402
6 3.14151380114430134327108135039452463
7 3.14157294036709178186583812930621207
8 3.14158772527716001476960627769585699
9 3.14159142151120018837673342204652727
10 3.14159234557011801669546002813149244
11 3.14159257658487289788240559573750943
12 3.14159263433856317249137646285817027
13 3.14159264877698607421052656718529761
14 3.14159265238659157759570916823577136
15 3.14159265328899328650891220604535192
16 3.14159265351459371373721296549774706
17 3.14159265357099393156659061787649989
18 3.14159265358509420806853995600249618
19 3.14159265358861894412711990298703313
20 3.14159265358950046120867227728012949
21 3.14159265358972117354596775840036571
22 3.14159265358977624060798916616477072
23 3.14159265358979000737349451810587198
24 3.14159265358979356008717331860680133
25 3.14159265358979444826559301873203367
26 3.14159265358979489235480286879464984
27 3.14159265358979489235480286879464984
```

dove le prime 15 cifre significative sono esatte: 3.14159265358979 come ci possiamo aspettare per un numero in doppia precisione. Nelle pagine moodle del corso trovate il semplice codice in C++ per ottenere tali risultati.

Nel resto del corso ci soffermeremo su tali problemi solo nel caso risultino rilevanti per il funzionamento degli algoritmi che svilupperemo.

2.4 Uso della memoria con C++: Stack e Heap

Durante l'esecuzione di un programma i dati vengono immagazzinati (allocati) nella memoria RAM. Distinguiamo due diversi tipi di allocazione: l'allocazione automatica nello stack e l'allocazione dinamica nello heap. Lo stack è un'area della RAM che viene assegnata ai dati temporanei allocatati in maniera automatica negli scopes di C/C++. Ad esempio:

```
{\\inizio scope
   double v;
   double a[100];
```

```
    double m[10,10];
}\\fine scope
```

La variabile m, l'array a e la matrice m vengono creati (allocati) riservando una zona dello stack. Quando lo scope finisce tale zona di memoria viene liberata automaticamente e i dati non sono più disponibili. Lo stack è basato sul principio di una pila (stack) di tipo LIFO (last in first out). Infatti man mano che si entra in scopes più interni la pila cresce mentre quando si esce dagli scopes la rispettiva parte di pila (ossia di memoria) viene rilasciata. Pertanto l'allocazione automatica nello stack risulta veloce anche se i dati sono disponibili solo all'interno dello scope in cui vengono definiti. La quantità di memoria massima riservata allo stack è determinata dal sistema operativo e di solito è relativamente limitata. Nei sistemi linux il comando *ulimit* permette di conoscere e in taluni casi di modificare la dimensione massima dello stack.

Lo heap, invece, è una zona della memoria riservata all'allocazione dinamica dei dati. I dati allocati in tale maniera sono disponibili anche al di fuori dello scope fintanto vengono liberati esplicitamente dal programma. In C++ possiamo allocare con *new* e deallocare con *delete*. Alternativamente possiamo usare come in C le funzioni *malloc* e *free*.

```
    double* v;
    double* a;

    v = (double*) new double;
    a = (double*) new double[100];

    delete [] v;
    delete [] a;
```

La dimensione massima dello heap è pari normalmente a tutta la RAM disponibile pertanto l'allocazione dinamica è consigliata per gestire grandi volumi di dati.

2.5 Array e Matrici in C++

Il C/C++ supporta l'allocazione automatica di vettori, chiamati arrays, e matrici. Basta scrivere semplicemente:

```
{\\inizio scope
    double v;
    double a[100];
    double m[10][10];
}\\fine scope
```

Inoltre dallo standard C++11 vengono supportati anche array e matrici di dimensione variabile ossia determinata durante l'esecuzione del programma:

```
{\\inizio scope
    int n;

    //qui viene determinato n

    double a[n];
    double m[n][n];
}\\fine scope
```

Purtroppo l'allocazione dinamica è limitata agli arrays:

```
{\\inizio scope
    int n;
    double* v;

    //qui viene determinato n

    v = (double*) new double[n];

    delete [] v;
}\\fine scope
```

Per allocare delle matrici riservando un unico blocco di RAM possiamo ricorrere alla strategia seguente

```
{\\inizio scope
   int n;
   double* bmat;
   double** b;

   //qui viene determinato n

   bmat = (double*) new double[n*m];
   b = (double**) new double*[n]

   for (int i=0; i<n; i++) {
        b[i]=&bmat[m*i];
   }


   delete [] v;
   delete [] mat;
   delete [] m;
}\\fine scope
```

In questo modo m si comporta esattamente come la matrice *double b[n][m]*. Inoltre ricordiamo che a differenza del Fortran in C/C++ gli le matrici sono immagazzinate in memoria riga dopo riga.

2.6 Importanza di un corretto uso della memoria

Mostriamo ora con un esempio quanto il modo in cui il nostro codice usa la memoria ed il trasferimento memoria/cache/-registri/fpu può influenzare il tempo di esecuzione. Consideriamo la moltiplicazione di due matrici quadrate di ordine (o dimensione) N, A e B:

$$C_{ij} = \sum_{k=1,N} A_{ik} B_{kj} \tag{6}$$

quindi per calcolare $C = A \cdot B$ dobbiamo compiere N^3 somme e addizioni per cui su un solo core ci aspettiamo un tempo di esecuzione paragonabile a $T_{ideal} = 2N^3/(N_{op} \cdot \text{clock})$, con clock la velocità di clock in cicli per secondo e N_{op} numero di operazioni in virgola mobile per ciclo di clock.

In questo corso gli esempi saranno dati in C/C++. Dato che N può essere un numero grande, le matrici A e B dovranno essere allocate in memoria (*heap*) sfruttando l'allocazione dinamica. In C però possiamo definire direttamente matrici sono nel caso di allocazione nello *stack*. Tipo:

```
double A[10][10];
```

per definire matrici di dimensioni arbitrarie in C possiamo utilizzare il seguente schema

```
....
long iN;
double *dA,*dB,*dC,*dD;
double **dAM,**dBM,**dCM;
.....
dA=(double*) malloc(iN*iN*sizeof(double));
dB=(double*) malloc(iN*iN*sizeof(double));
dC=(double*) malloc(iN*iN*sizeof(double));

dAM=(double**) malloc(iN*sizeof(double*));
dBM=(double**) malloc(iN*sizeof(double*));
dCM=(double**) malloc(iN*sizeof(double*));
........
 for(iI=0;iI<iN;iI++){
    dAM[iI]=&dA[iN*iI];
    dBM[iI]=&dB[iN*iI];
    dCM[iI]=&dC[iN*iI];
 }
........
```

ora possiamo scrivere l'operazione di moltiplicazione tra matrici come:

```
for(iI=0;iI<iN;iI++){
    for(iJ=0;iJ<iN;iJ++){
        for(iK=0;iK<iN;iK++){
            dCM[iI][iJ]+=dAM[iI][iK]*dBM[iK][iJ];
        }
    }
 }
```

Consideriamo $N = 1000$ e usiamo un MacBook Pro del 2014 con microprocessore Intel i5 (sandy bridge) che ha un clock=2.6 GHz e può eseguire fino a 8 somme e moltiplicazioni in virgola mobile per ciclo di clock. Ci aspettiamo un tempo di esecuzione ideale di $T_{ideale} = 2 \cdot 1000^3/(8 \cdot 2.6 \cdot 10^9) = 0.096$ s. Il codice sorgente è riportato in appendice. Se compiliamo senza attivare le ottimizzazioni:

```
g++ -lcblas   main.cpp
```

registriamo un tempo $T_{naive} = 6.8$ s ossia il nostro codice sta girando 68 volte più lentamente della velocità ideale. Ci accorgiamo che nel ciclo *for* più interno, quello su iK, compiamo l'operazione

```
dCM[iI][iJ]+=dAM[iI][iK]*dBM[iK][iJ];
```

quindi per iK e IK+1 andiamo ad usare due elementi di dBM che non sono contigui in memoria. Per usare solo elementi contigui basta considerare al posto di dBM la sua trasposta:

```
for(iI=0;iI<iN;iI++){
     for(iJ=0;iJ<iN;iJ++){
          dDM[iI][iJ]=dBM[iJ][iI];
     }
}
for(iI=0;iI<iN;iI++){
      for(iJ=0;iJ<iN;iJ++){
           for(iK=0;iK<iN;iK++){
                dCM[iI][iJ]+=dAM[iI][iK]*dDM[iJ][iK];
           }
      }
}
```

dove il calcolo della matrice trasposta dDM richiede N^2 operazioni di assegnazione. Ora per la moltiplicazione matriciale occorre un tempo $T_{trasposta} = 3.7$ s. Possiamo ora tentare di utilizzare al meglio le risorse del processore attivando una compilazione più ottimizzata:

```
g++ -lcblas   -Os main.cpp
```

questo consente di ridurre sia il tempo della strategia iniziale $T^*_{naive} = 2.8$ s sia di quella tramite trasposta $T^*_{trasposta} = 1.09$ s. Ci accorgiamo che ciascuna delle nostre matrici occupa uno spazio in memoria di $8 \cdot 1000^2 = 7.6$ MB più grande della cache memory (L3 di terzo livello). Per cercare di utilizzare al meglio la cache memory andiamo a scomporre l'operazione $C = A \cdot B$ nella moltiplicazione di matrici di dimensione ridotta:

```
for(iII=0;iII<iN;iII+=iNblock){
     for(iJJ=0;iJJ<iN;iJJ+=iNblock){
       for(iKK=0;iKK<iN;iKK+=iNblock){
                          //copy data
              for(iI=0;iI<iNblock;iI++){
                   pd1=&dAbl[iI*iNblock];
                   pd2=&dA[(iII+iI)*iN+iKK];
                        for(iJ=0;iJ<iNblock;iJ++){
                              pd1[iJ]=pd2[iJ];
                        }
               }
//of B take the transpose
               for(iI=0;iI<iNblock;iI++){
                   pd1=&dBbl[iI*iNblock];
                   pd2=&dB[(iKK+iI)*iN+iJJ];
                   for(iJ=0;iJ<iNblock;iJ++){
```

```
                        pd1[iJ]=pd2[iJ];
                    }
                }
                for(iI=0;iI<iNblock;iI++){
                    for(iJ=0;iJ<iNblock;iJ++){
                        dDbl[iI*iNblock+iJ]=dBbl[iJ*iNblock+iI];
                    }
                }
                for(iI=0;iI<iNblock*iNblock;iI++){
                    dCbl[iI]=0.;
                }
                for(iI=0;iI<iNblock;iI++){
                    for(iJ=0;iJ<iNblock;iJ++){
                        pd1=&dAbl[iI*iNblock];
                        pd2=&dDbl[iJ*iNblock];
                        pd3=&dCbl[iI*iNblock+iJ];
                        for(iK=0;iK<iNblock;iK++){
                            *pd3+=pd1[iK]*pd2[iK];
                        }
                    }
                }
            //sum up on global matrix
                for(iI=0;iI<iNblock;iI++){
                    pd1=&dC[(iII+iI)*iN+(iJJ)];
                    pd2=&dCbl[iI*iNblock];
                    for(iJ=0;iJ<iNblock;iJ++){
                        pd1[iJ]+=pd2[iJ];
                    }
                }
            }
        }
}
```

dove iNblock è la dimensione N_b della matrice ridotte. Il costo del calcolo dipende dalla scelta di N_b. Per $N_b = 25$ troviamo $T^*_{block} = 0.86$ s. Abbiamo ottenuto una diminuzione del tempo di esecuzione di circa il 20% ma siamo ancora lontani dal limite ideale. Inoltre il nostro codice è diventato molto più complesso.

Per sfruttare al meglio le capacità dei processori sono state create delle librerie matematiche standardizzate quali BLAS e LAPACK (www.netlib.org) che contengono routines per risolvere problemi di algebra lineare. La libreria BLAS provvede routines per prodotti e somme di matrici e vettori. LAPACK invece fornisce routines per operazioni quali la soluzione del problema agli autovalori e la diagonalizzazione di matrici. Queste librerie sono scritte in FORTRAN e possono contenere parti in scritte in ASSEMBLY. Per evitare le noie dovute al chiamare subroutines Fortran in un codice scritto in C/C++ possiamo usare le librerie CBLAS e CLAPACK che provvedono con delle interfacce. Proviamo allora ad usare la routine CBLAS: cblas_dgemm che calcola il il prodotto di due matrici. Essa usa gli stessi parametri della routine Fortran DGEMM descritti in http://www.netlib.org/lapack/explore-html/d7/d2b/dgemm_8f.html.Notiamo che usare cblas_dgemm richiede poca scrittura di codice:

```
cblas_dgemm( CblasRowMajor, CblasNoTrans, CblasNoTrans, iN, iN, iN, 1.0, dA, iN, dB, iN, 1.0, dC, iN);
```

inoltre non è richiesto alcun livello particolare di ottimizzazione durante la compilazione. Usando quest'ultima strategia troviamo $T_{dgemm} = 0.18$ s. Ben compatibile con la nostra stima per il limite ideale. I codici in C++ sono riportati sul moodle.

2.7 Moltiplicazione di matrici in python

Analizziamo ora il problema della moltiplicazione di due matrici usando python. Per definire delle matrici useremo la libreria *numpy*. Proviamo moltiplicare due matrici quadrate di ordine n nella seguente semplice maniera:

```
import numpy as np
import time
```

```
def main():
    n=input("Ordine della matrice :\n")
    n=int(n)

    ma=np.ones((n,n),dtype=np.float64)


    mb=np.ones((n,n),dtype=np.float64)

    mc=np.zeros((n,n),dtype=np.float64)

    print("Ora moltiplico")

    start_time = time.perf_counter()
    for i in range(n):
        for j in range (n):
            for k in range (n):
                mc[i,j]+=ma[i,k]*mb[k,j]

    end_time = time.perf_counter()

    elapsed_time = end_time - start_time

    print("Elapsed time: ", elapsed_time)


    print(str(mc[0,0])+ " "+str(mc[1,0])+"  "+ str(mc[2,0]))

if __name__ == "__main__":
    main()
```

Ho utilizzato il metodo *np.unos* per creare le matrici di tipo reale a doppia precisione (*dtype=np.float64*). Facciamo delle prove con un laptop che monta un processore Intel Core (11th Gen Intel(R) Core(TM) i7-1165G7) dotato di 4 cores. Ognuno di questi può svolgere fino a 10 operazioni in virgola mobile per ciclo di cloack. Per cui mi aspetto una performance teorica in doppia precisione di $4*8*4.8 = 192$ GFLOPS. Dove la velocità di cloack è stata posta a 4.8 GHz (valore di boost). Per matrici di ordine n ci mette 52.0 s invece dei, considerando $2*n^3$ operazioni in virgola mobile, $2*500^3/(192*10^9) = 0.0014$ s del tempo ideale. Questa grande differenza è dovuta sia al fatto che python è un linguaggio interpretato e non compilato, sia al cattivo utilizzo della memoria fatto dall'algoritmo usato per la moltiplicazione. Una semplice soluzione è usare il metodo *np.matmul* di *numpy* per moltiplicare due matrici. Tale metodo (nella maggior parte delle implementazioni) utilizza le librerie BLAS. Inoltre permetter di girare su più cori contemporaneamente tramite il protocollo OPENMP.

```
    import numpy as np
import time


def main():
    n=input("Ordine della matrice :\n")
    n=int(n)

    ma=np.ones((n,n),dtype=np.float64)
    mb=np.ones((n,n),dtype=np.float64)
    print("Ora moltiplico")

    start_time = time.perf_counter()
    mc=np.matmul(ma,mb)
    end_time = time.perf_counter()

    elapsed_time = end_time - start_time
```

```
print(str(mc[0,0])+ " "+str(mc[1,0])+"   "+ str(mc[2,0]))
print("Elapsed time: ", elapsed_time)
```

Ora ci vogliono solo 0.017 s. Proviamo ora un test più significativo scegliendo $n = 10000$. Ora *np.matmul* ci mette 12.5 s contro i $2(10^4)^3/(192 * 10^9) = 10.5$ s del tempo teorico ideale. Per cambiare il numero di *threads* impiegati, uso il comando linux, valido per la shell BASH :

```
export OMP_NUM_THREADS=4
```

in cui abbiamo impostato 4 threads corrispondenti a 4 cores fisici del processore.

Parte II
Derivare ed integrare

Per introdurre dei metodi numerici per risolvere le equazioni della fisica che sono di natura differenziale, cominciamo con analizzare dei metodi per derivare ed integrare numericamente delle funzioni matematiche. Per semplicità la nostra trattazione sarà limitata a funzioni reali di numeri reali. Consideriamo la funzione $f(x)$ con $x \in \mathbb{R}$ e $f(x) \in \mathbb{R}$. Supponiamo di conoscere $f(x)$ su una griglia di valori discreti di x che indichiamo con $\{x_i\}$. Per semplicità consideriamo anche che tale griglia sia equispaziata:

$$x_i = hi + x_{\text{offset}} \tag{7}$$

con $i \in \mathbb{Z}$ e $h, x_{\text{offset}} \in \mathbb{R}$. Indichiamo anche con f_i:

$$f_i = f(x_i) \tag{8}$$

e per $\epsilon \in [0,1]$:

$$f_{i+\epsilon} = f(x_i + h\epsilon) \tag{9}$$

3 Derivare

L'esigenza di derivare numericamente una funzione non sorge solo dalla eventuale difficoltà per calcolare analiticamente la derivata di una formula nota ma anche dalla possibilità che possiamo conoscere la funzione derivanda $f(x)$ solo su di un insieme di punti. Cominciamo con l'introdurre la notazione:

$$f_i^n = \left.\frac{d^n f(x)}{dx^n}\right|_{x=x_i} \tag{10}$$

Dalla definizione stessa di derivata come limite del rapporto incrementale:

$$\left.\frac{df}{dx}\right|_{x_o} = \lim_{\Delta x \to 0} \frac{f(x_o + \Delta x) - f(x_0)}{\Delta x} \tag{11}$$

saremmo tentati di calcolare semplicemente $f_i^1 \approx \frac{f_{i+1} - f_i}{h}$ per poi scegliere un h sufficientemente piccolo da garantire l'accuratezza di f^1 cercata ma ci accorgiamo che a causa del *roundoff error* tale formula numericamente diviene mal definita per h piccoli. Per ovviare a questo problema dobbiamo analizzare l'errore nel calcolo della derivata rispetto al *grid-spacing* h. Sfruttando lo sviluppo in serie di Taylor fino all' ordine n, possiamo scrivere:

$$f_{i+1} = f_i + \sum_{m=1,n} \frac{1}{m!} f_i^m h^m + O\left(h^{m+1}\right) \tag{12}$$

allora per il nostro metodo naif:

$$f_i^{1,naif} = f_i^1 + \frac{O\left(h^2\right)}{h} = f_i^1 + O(h) \tag{13}$$

che preferiamo scrivere come:

$$f_i^1 = \frac{f_{i+1} - f_i}{h} + O(h) \tag{14}$$

analogamente possiamo scegliere il limite sinistro al posto del limite destro e scrivere:

$$f_i^1 = \frac{f_i - f_{i-1}}{h} + O(h) \tag{15}$$

Per ottenere una stima migliore dobbiamo ricorrere allo sviluppo in serie di Taylor fino al terzo ordine:

$$\begin{cases} f_{i+1} &= f_i + f_i^1 h + \frac{1}{2} f_i^2 h^2 + O\left(h^3\right) \\ f_{i-1} &= f_i - f_i^1 h + \frac{1}{2} f_i^2 h^2 + O\left(h^3\right) \end{cases} \tag{16}$$

sottraendo membro a membro possiamo trovare:

$$f_i^1 = \frac{f_{i+1} - f_{i-1}}{2h} + O\left(h^2\right) \tag{17}$$

quindi abbiamo ottenuto un errore di un ordine di h minore.

Proviamo ora a ricavare una formula per f^2. Usiamo lo sviluppo in serie di Taylor fino al terzo ordine:

$$\begin{cases} f_{i+1} &= f_i + f_i^1 h + \frac{1}{2} f_i^2 h^2 + \frac{1}{3!} f_i^3 h^3 + O\left(h^4\right) \\ f_{i-1} &= f_i - f_i^1 h + \frac{1}{2} f_i^2 h^2 - \frac{1}{3!} f_i^3 h^3 + O\left(h^4\right) \end{cases} \tag{18}$$

sommando membro a membro troviamo:

$$f_i^2 = \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} + O\left(h^2\right) \tag{19}$$

Ci poniamo ora il problema di come ottenere f^1 con accuratezza ancora maggiore e di come ottenere le derivate di ordine successivo. Consideriamo lo sviluppo in serie di taylor fino all'ordine n con n pari. Possiamo costruire il seguente sistema di equazioni lineari a $n+1$ variabili:

$$\begin{cases} f_{i+\frac{n}{2}} &= f_i + f_i^1 \left(\frac{n}{2} h\right) + \frac{1}{2} f_i^2 \left(\frac{n}{2} h\right)^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} \left(\frac{n}{2} h\right)^{n-1} + \frac{1}{(n)!} f_i^n \left(\frac{n}{2} h\right)^n + O\left(h^{n+1}\right) \\ f_{i+\frac{n}{2}-1} &= f_i + f_i^1 \left(\left(\frac{n}{2} - 1\right) h\right) + \frac{1}{2} f_i^2 \left(\left(\frac{n}{2} - 1\right) h\right)^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} \left(\left(\frac{n}{2} - 1\right) h\right)^{n-1} + \frac{1}{(n)!} f_i^n \left(\left(\frac{n}{2} - 1\right) h\right)^n + O\left(h^{n+1}\right) \\ \ldots & \\ f_{i+1} &= f_i + f_i^1 h + \frac{1}{2} f_i^2 h^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} h^{n-1} + \frac{1}{(n)!} f_i^n h^n + O\left(h^{n+1}\right) \\ f_i &= f_i \\ f_{i-1} &= f_i - f_i^1 h + \frac{1}{2} f_i^2 h^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} (-h)^{n-1} + \frac{1}{(n)!} f_i^n (-h)^n + O\left(h^{n+1}\right) \\ \ldots & \\ f_{i-\frac{n}{2}+1} &= f_i + f_i^1 \left(-\left(\frac{n}{2} - 1\right) h\right) + \frac{1}{2} f_i^2 \left(-\left(\frac{n}{2} - 1\right) h\right)^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} \left(-\left(\frac{n}{2} - 1\right) h\right)^{n-1} + \frac{1}{(n)!} f_i^n \left(-\left(\frac{n}{2} - 1\right) h\right)^n + O\left(h^{n+1}\right) \\ f_{i-\frac{n}{2}} &= f_i + f_i^1 \left(-\frac{n}{2} h\right) + \frac{1}{2} f_i^2 \left(-\frac{n}{2} h\right)^2 + \ldots + \frac{1}{(n-1)!} f_i^{n-1} \left(-\frac{n}{2} h\right)^{n-1} + \frac{1}{(n)!} f_i^n \left(-\frac{n}{2} h\right)^n + O\left(h^{n+1}\right) \end{cases} \tag{20}$$

che possiamo facilmente risolvere nelle incognite $f_i^1, f_i^2, \ldots, f_i^{n-1}, f_i^n$. Le soluzioni avranno accuratezze $O\left(h^n\right), O\left(h^{n-1}\right), O\left(h^2\right), O($ rispettivamente. A guisa d'esempio vediamo il caso $n = 2$, che fornisce il sistema:

$$\begin{cases} f_{i+1} &= f_i + f_i^1 h + \frac{1}{2} f_i^2 h^2 \\ f_i &= f_i \\ f_{i-1} &= f_i + f_i^1 h + \frac{1}{2} f_i^2 h^2 \end{cases} \tag{21}$$

che possiamo scrivere il forma matriciale come:

$$\begin{pmatrix} f_{i+1} \\ f_i \\ f_{i-1} \end{pmatrix} = \begin{pmatrix} 1 & h & \frac{1}{2}h^2 \\ 1 & 0 & 0 \\ 1 & -h & \frac{1}{2}h^2 \end{pmatrix} \begin{pmatrix} f_i \\ f_i^1 \\ f_i^2 \end{pmatrix} \tag{22}$$

che possiamo risolvere come:

$$\begin{pmatrix} f_i \\ f_i^1 \\ f_i^2 \end{pmatrix} = \begin{pmatrix} 1 & h & \frac{1}{2}h^2 \\ 1 & 0 & 0 \\ 1 & -h & \frac{1}{2}h^2 \end{pmatrix}^{-1} \begin{pmatrix} f_{i+1} \\ f_i \\ f_{i-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2h} & 0 & -\frac{1}{2h} \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{pmatrix} \begin{pmatrix} f_{i+1} \\ f_i \\ f_{i-1} \end{pmatrix} \tag{23}$$

da cui (ri-)troviamo le formule:

$$\begin{cases} f_i^1 &= \frac{f_{i+1} - f_{i-1}}{2h} + O\left(h^2\right) \\ f_i^2 &= \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + O\left(h^2\right) \end{cases} \tag{24}$$

Dove l'errore in $O\left(h^2\right)$ per f_i^2 è dovuto alla simmetria del problema come visto prima. L'inversione matriciale può essere anche fatta numericamente usando le routine BLAS/LAPACK. Applichiamo ora al caso $n = 4$:

$$\begin{pmatrix} f_{i+2} \\ f_{i+1} \\ f_i \\ f_{i-1} \\ f_{i-2} \end{pmatrix} = \begin{pmatrix} 1 & 2h & \frac{1}{2}(2h)^2 & \frac{1}{3!}(2h)^2 & \frac{1}{4!}(2h)^4 \\ 1 & h & \frac{1}{2}h^2 & \frac{1}{3!}h^3 & \frac{1}{4!}h^4 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & -h & \frac{1}{2}h^2 & -\frac{1}{3!}h^3 & \frac{1}{4!}h^4 \\ 1 & -2h & \frac{1}{2}(2h)^2 & -\frac{1}{3!}(2h)^2 & \frac{1}{4!}(2h)^4 \end{pmatrix} \begin{pmatrix} f_i \\ f_i^1 \\ f_i^2 \\ f_i^3 \\ f_i^4 \end{pmatrix} = \tag{25}$$

da cui invertendo:

$$\begin{pmatrix} f_i \\ f_i^1 \\ f_i^2 \\ f_i^3 \\ f_i^4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{12h} & \frac{2}{3h} & 0 & -\frac{2}{3h} & \frac{1}{12h} \\ -\frac{1}{12h^2} & \frac{4}{3h^2} & -\frac{5}{2h^2} & \frac{4}{3h^2} & -\frac{1}{12h^2} \\ \frac{1}{2h^3} & -\frac{1}{h^3} & 0 & \frac{1}{h^3} & -\frac{1}{2h^3} \\ \frac{1}{h^4} & -\frac{4}{h^4} & \frac{6}{h^4} & -\frac{4}{h^4} & \frac{1}{h^4} \end{pmatrix} \begin{pmatrix} f_{i+2} \\ f_{i+1} \\ f_i \\ f_{i-1} \\ f_{i-2} \end{pmatrix} \tag{26}$$

che ci permette di trovare:

$$\begin{cases} f_i^1 &= \frac{-f_{i+2}+8f_{i+1}-8f_{i-1}+f_{i-2}}{12h} + O\left(h^4\right) \\ f_i^2 &= \frac{-f_{i+2}+16f_{i+1}-30f_i+16f_{i-1}-f_{i-2}}{12h^2} + O\left(h^4\right) \\ f_i^3 &= \frac{f_{i+2}-2f_{i+1}+2f_{i-1}-f_{i-2}}{2h^3} + O\left(h^2\right) \\ f_i^4 &= \frac{f_{i+2}-4f_{i+1}+6f_i-4f_{i-1}+f_{i-2}}{h^4} + O\left(h^2\right) \end{cases} \tag{27}$$

Tale metodo può essere facilmente esteso al caso di derivate parziali.

4 Integrare

Consideriamo il problema dell'integrazione della funzione $f(x)$ di cui conosciamo solo i valori $\{f_i\}$ nei punti $\{x_i\}$. Supponiamo di voler integrare la funzione da x_0 a x_N. La stessa definizione di Riemann dell'integrale:

$$\int_a^b f(x)\,dx = \lim_{M\to+\infty} \sum_{l=0,M-1} f\left(a + \frac{(b-a)}{M}l\right)\frac{(b-a)}{M} \tag{28}$$

ci porta a scrivere:

$$\int_{x_0}^{x_N} f(x)\,dx \approx \sum_{i=0,N-1} f_i h \tag{29}$$

che prende anche il nome di *metodo dei rettangoli naif.* Si può facilmente intuirne il motivo guardando Fig. 5 (a) l'integrale corrisponde all' area in tratteggiato. E' interessante notare che possiamo considerare il metodo dei rettangoli (e gli altri metodi che vedremo) come l'integrale analitico di una funzione $\tilde{f}(x)$ che interpola la $f(x)$. In particolare nel metodo dei rettangoli tale funzione è discontinua. Andiamo a considerare l'errore associato al metodo dei rettangoli sfruttando lo sviluppo in serie di Taylor:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} \left(f_i + O(h)\right)dx = f_i h + O\left(h^2\right) \tag{30}$$

quindi l'accuratezza del metodo dei triangoli sarà:

$$\int_{x_0}^{x_N} f(x)\,dx = \sum_{i=0,N-1} f_i h + NO\left(h^2\right) \tag{31}$$

e ricordandoci che $h = (x_N - x_0)/N$ possiamo scrivere

$$\int_{x_0}^{x_N} f(x)\,dx = \sum_{i=0,N-1} f_i h + O\left(\frac{1}{N}\right) \tag{32}$$

Notiamo che nel metodo dei rettangoli gli estremi di integrazione x_0 e x_N sono anche punti della griglia $\{x_i\}$ tale metodo viene indicato quindi come metodo *chiuso.*

Possiamo pensare a vedere cosa succede al procedimento appena visto considerando estremi di integrazione non appartenenti alla griglia in x su cui andiamo a calcolare la funzione f. Supponiamo quindi di conoscere la funzione f nei punti $\left\{x_{i+\frac{1}{2}}\right\}$. Allora:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} \left(f_{i+\frac{1}{2}} + f_{i+\frac{1}{2}}^1\left(x - x_{i+\frac{1}{2}}\right) + O\left(h^2\right)\right)dx = f_{i+\frac{1}{2}}h + \left[\frac{1}{2}x^2 - x_{i+\frac{1}{2}}x\right]_{x_i}^{x_{i+1}} + O\left(h^3\right) \tag{33}$$

ora possiamo sfruttare la proprietà $x_{i+\frac{1}{2}} = \frac{x_{i+1}+x_i}{2}$ quindi:

$$\left[\frac{1}{2}x^2 - x_{i+\frac{1}{2}}x\right]_{x_i}^{x_{i+1}} = \left[\frac{1}{2}x^2 - \frac{x_{i+1}+x_i}{2}x\right]_{x_i}^{x_{i+1}} = 0 \tag{34}$$

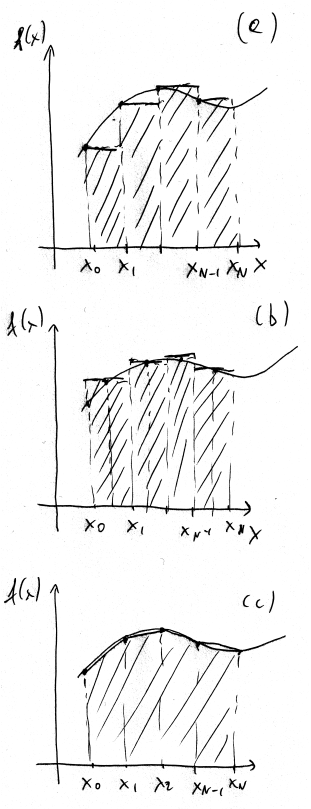


Figura 5: Integrazione con: (a) metodo dei rettangoli naif, (b) metodo dei rettangoli, (c) metodo dei trapezi

da cui:

$$\int_{x_0}^{x_N} f(x)\,dx = \sum_{i=0,N-1} f_{i+\frac{1}{2}} h + NO\left(h^3\right) = \sum_{i=0,N-1} f_{i+\frac{1}{2}} h + O\left(\frac{1}{N^2}\right) \tag{35}$$

tale metodo prende il nome di *metodo dei rettangoli e* come si vede in figura 5 (b) è *aperto* rispetto agli estremi di integrazione. Anche in questo caso il metodo di integrazione può essere visto come un'interpolazione della funzione f e anche in questo caso essa è discontinua. Andiamo ora a vedere un metodo corrispondente ad un'interpolazione continua. Consideriamo l'espansione in serie di Taylor fino al secondo ordine:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} \left(f_i + f_i^1 (x - x_i) + O\left(h^2\right)\right) dx = \int_{x_i}^{x_{i+1}} \left(f_i + \left(\frac{f_{i+1} - f_i}{h} + O(h)\right)(x - x_i) + O\left(h^2\right)\right) dx \tag{36}$$

dove abbiamo usato la formula *naif* per f_i^1. Si ricava:

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} f(x)\,dx &= \left[f_i + \left(\frac{f_{i+1} - f_i}{h}\right)\left(\frac{1}{2}x^2 - x_i x\right)\right]_{x_i}^{x_{i+1}} + O\left(h^3\right) \\
&= \left(\frac{f_{i+1} - f_i}{h}\right)\left(\frac{x_{i+1}^2}{2} - x_i x_{i+1} - \frac{x_i^2}{2} + x_{i^2}\right) + f_i (x_{i+1} - x_i) + O\left(h^3\right) \\
&= \left(\frac{f_{i+1} - f_i}{h}\right)\frac{1}{2}(x_{i+1} - x_i)^2 + f_i (x_{i+1} - x_i) = \frac{(f_{i+1} - f_i)}{2h} h^2 + f_i h = \frac{(f_{i+1} + f_i)}{2} h + O\left(h^3\right)
\end{aligned} \tag{37}$$

stando ora attenti ai punti estremi f_0 e f_1, Troviamo:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \sum_{i=0,N-1} \frac{f_i + f_{i+1}}{2} h + NO\left(h^3\right) = \frac{1}{2}(f_0 + f_N) h + \sum_{i=1,N-1} f_i h + O\left(\frac{1}{N^2}\right) \tag{38}$$

questo metodo prende il nome di *metodo dei trapezi* ed è *chiuso* rispetto agli estremi di integrazione. La sua accuratezza è analoga al metodo dei rettangoli. Questo non ci sorprende infatti il metodo dei trapezi è equivalente al metodo dei

rettangoli ponendo $f_{i+\frac{1}{2}} \approx \frac{1}{2}(f_{i+1}+f_i)$, vedi Fig. 5 (c). Ad ogni modo i due metodi non danno nel caso generale lo stesso risultato.

Vogliamo ora vedere di aumentare la nostra accuratezza, possiamo ancora ricorrere allo sviluppo in serie di Taylor ed al metodo delle differenze finite per trovare le derivate. Vediamo ora lo sviluppo fino al terzo ordine e integriamo tra x_{i-1} e x_{i+1}. Sviluppiamo attorno al punto centrale x_i. Termini di ordine dispari (primo e terzo ordine) integrano a zero per simmetria.

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx &= \int_{x_{i-1}}^{x_{i+1}} \left(f_i + f_i^1(x-x_i) + \frac{f_i^2}{2!}(x-x_i)^2 + \frac{f_i^3}{3!}(x-x_i)^3 + O\left(h^4\right)\right) dx \\ &= \int_{x_{i-1}}^{x_{i+1}} \left(f_i + \frac{1}{2}\left(\frac{f_{i+1}-2f_i+f_{i-1}}{h^2} + O\left(h^2\right)\right)(x-x_i)^2 + O\left(h^4\right)\right) dx \\ &= \int_{x_{i-1}}^{x_{i+1}} \left(f_i + \frac{1}{2}\left(\frac{f_{i+1}-2f_i+f_{i-1}}{h^2}\right)(x-x_i)^2 + O\left(h^4\right)\right) dx \\ &= f_i 2h + \frac{1}{6}\left(\frac{f_{i+1}-2f_i+f_{i-1}}{h^2}\right)2h^3 + O\left(h^5\right) = \left(\frac{1}{3}f_{i+1}+\frac{4}{3}f_i+\frac{1}{3}f_{i-1}\right)h + O\left(h^5\right)\end{aligned} \quad + \quad (39)$$

ora dobbiamo sommare sugli intervalli di lunghezza $2h$, poniamo quindi N pari, stando attenti agli estremi, troviamo:

$$\int_{x_0}^{x_N} f(x)\,dx = \frac{1}{3}hf_0 + \frac{4}{3}hf_1 + \frac{2}{3}hf_2 + \frac{4}{3}hf_3 + \frac{2}{3}hf_3 + \ldots + \frac{4}{3}hf_{N-1} + \frac{1}{3}hf_N + O\left(\frac{1}{N^4}\right) \quad (40)$$

tale regola d'integrazione prende il nome di *regola di Simpson* ed è chiusa rispetto agli estremi. E' interessante vedere a quale interpolazione corrisponde l'integrazione tramite la regola di Simpson. Essa è equivalente ad integrare tra x_{i-1} e x_{i+1} la funzione:

$$\widetilde{f}(x) = f_i + \frac{f_{i+1}-f_{i-1}}{2h}(x-x_i) + \frac{f_{i+1}-2f_i+f_{i-1}}{2h^2}(x-x_i)^2 \quad (41)$$

essa è un polinomio di secondo grado che passa per i punti f_{i-1}, f_i, f_{i+1}. Notiamo che di tali polinomi ne esiste uno solo. Quindi il metodo di Simpson equivale ad approssimare la funzione $f(x)$ tra x_{i-1} e x_{i+1} con un $p_2(x)$ polinomio di II grado definito dalle condizioni $p_2(x_{i-1}) = f_{i-1}$, $p_2(x_i) = f_i$,$p_2(x_{i+1}) = f_{i+1}$. Possiamo generalizzare questa idea considerando n punti da x_i a x_{i+n-1} ed andare ad integrare il polinomio di grado $n-1$ passante per i punti $(x_i, f_i), \ldots (x_{i+n-1}, f_{i+n-1})$. Tale metodo prende il nome di *regola di Newton-Cotes*. Se $n = 2$ il polinomio è un segmento di retta passante per i due punti e vado a ritrovare il metodo dei trapezi. Per un n arbitrario p_{n-1} è dato dalla formula di Lagrange:

$$p_{n-1}(x) = \sum_{j=1,n} f_j \prod_{\substack{k=1,n \\ k\neq j}} \frac{x-x_k}{x_j - x_k} \quad (42)$$

Sembrerebbe alletante usare tale formula per interpolare $f(x)$ da f_0 a f_N usando tutti i punti f_i, purtroppo la formula di Lagrange non è stabile per n dell'ordine di qualche decina e per n ancora più grandi può fornire catastrofici risultati divergenti. Illustriamo questo problema studiando la funzione:

$$f(x) = xsin(20x) \quad (43)$$

nell'intervallo $[0, 1]$. In Fig. 6 mostriamo che mentre per n piccolo l'interpolazione è soddisfacente per n grande essa diventa completamente erronea in prossimità degli estremi.

4.1 Integrazione con estremo infinito

Talvolta dobbiamo integrare su di un intervallo del tipo $\int_0^{+\infty} f(x)\,dx$, questo non permette l'introduzione di griglie equispaziate in x. Supponendo che f sia integrabile, andiamo a scegliere una funzione $g(y)$ monotona tale che $g(0) = 0$ e $\lim_{y\to 1^-} g(y) = +\infty$ allora possiamo porre $x = g(y)$ e scrivere:

$$\int_0^{+\infty} f(x)\,dx = \int_0^1 f(g(y))\left(\frac{dg(y)}{dy}\right)dy \quad (44)$$

dunque possiamo utilizzare le tecniche apprese nella precedente sezione integrando su y nell'intervallo $[0, 1]$. Un esempio di tale funzione è $g(x) = \frac{1}{1-x} - 1 = \frac{x}{1-x}$.

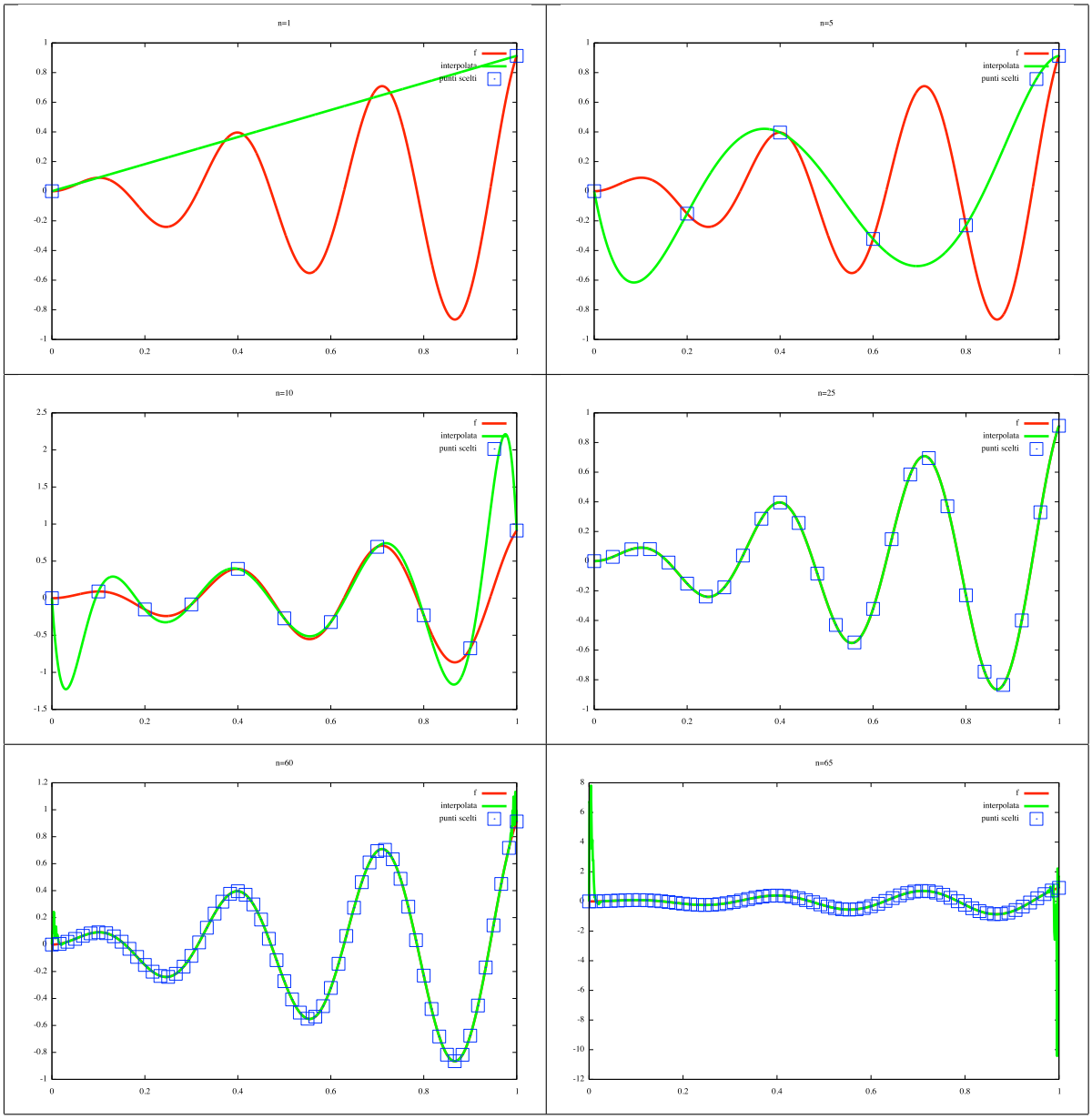


Figura 6: Funzione $x \sin(20x)$ (rosso) e interpolazione (verde) tramite formula di Lagrange con n punti. I $n+1$punti della funzione scelti sono visualizzati come quadrati blu.

4.2 Integrazione di funzioni a più dimensioni

Consideriamo la funzione $f\left(x^1, x^2, x^3, \ldots, x^M\right)$ definita su $\mathbb{R}^M$, supponiamo di volere integrare sul volume $\left[x_0^1, x_N^1\right]\left[x_0^2, x_N^2\right]\ldots\left[x_0^M, x_N^M\right]$ e supponiamo di conoscerla su di una griglia equispaziata rispetto ad ogni direzione tale che per ogni direzione la valuti in N punti. Ossia la funzione è conosciuta in N^M punti. Posso facilmente integrarla con il metodo dei rettangoli o dei trapezi (scegliamo qui quello dei rettangoli per semplicità di scrittura):

$$\begin{aligned}
&\int_{x_0^1}^{x_N^1} dx^1 \ldots \int_{x_0^M}^{x_N^M} dx^M f(x^1, \ldots, x^M) = \int_{x_0^1}^{x_N^1} dx^1 \ldots \int_{x_0^{M-1}}^{x_N^{M-1}} dx^{M-1} \left(\sum_{i=0,N-1} f\left(x^1, \ldots, f_{i+\frac{1}{2}}^M\right) h + O\left(\frac{1}{N^2}\right)\right) \\
&= \left\{ \int_{x_0^1}^{x_N^1} dx^1 \ldots \int_{x_0^{M-2}}^{x_N^{M-2}} dx^{M-2} \left(\sum_{j=0,N-1} \sum_{i=0,N-1} f\left(x^1, \ldots, f_{j+\frac{1}{2}}^{M-1}, f_{i+\frac{1}{2}}^M\right) h + O\left(\frac{1}{N^2}\right)\right)\right\} + O\left(\frac{1}{N^2}\right) \\
&= \sum_{i^1=0,N-1} \cdots \sum_{i^M=0,N-1} f\left(x_{i^1+\frac{1}{2}}^1, \ldots, f_{i^M+\frac{1}{2}}^M\right) h^M + MO\left(\frac{1}{N^2}\right)
\end{aligned} \tag{45}$$

dove M è generalmente molto più piccolo di N per cui l'accuratezza finale è dell'ordine $O\left(\frac{1}{N^2}\right)$ dove N è il numero di punti lungo la singola direzione. Per una dimensione M il numero totale di punti in cui devo valutare f è $N_{sampling} = N^M$ per cui l'accuratezza rispetto al numero di valutazioni della funzione è $O\left(\frac{1}{N_{sampling}^{2/M}}\right)$ per cui per M grande l'integrazione diventerà costosa dal punto computazionale.

5 Interpolare

Abbiamo visto come i metodo per l'integrazione corrispondono a metodi di interpolazione. Vediamo ora due metodi usati per la solo interpolazione. Supponiamo di voler interpolare la funzione $f(x)$ conosciuta nei punti x_i, $f_i = f(x_i)$ con $i = 0, .., N$.

5.1 Splines

In ogni intervallo $[x_i, x_{i+1}]$ interpoliamo con un polinomio di ordine m:

$$f(x) \approx p_i(x) = \sum_{k=0,m} c_{i,k} x^k \quad \text{con} \quad x \in [x_i, x_{i+1}] \tag{46}$$

I coefficienti $c_{i,k}$ vengono trovati imponendo che la funzione interpolante passi per i punti (x_i, f_i) e che essa sia continua con derivate continue fino all'ordine $m-1$:

$$p_i(x_i) = f_i \quad i = 0, N-1 \tag{47}$$

$$p_{N-1}(x_N) = f_N \tag{48}$$

$$p_i^{(l)}(x_{i+1}) = p_{i+1}^{(l)}(x_{i+1}) \quad i = 0, N-2 \;\; l = 0, m-1 \tag{49}$$

Dove *fisso* solo l'estremo sinistro per ogni intervallino; quello destro lo fissa la richiesta di continuità. Notiamo che ho un totale di $(m+1)N$ incognite ed un sistema di $N+1+(m)(N-1)$ equazioni. Il sistema per avere soluzioni richiede dunque ancora $m-1$ equazioni. Esse vanno cercate imponendo qualche relazione per $p_0^{(l)}(x_0)$ e per $p_{N-1}^{(l)}(x_N)$ con $l = m-1, m-2, ...$

Vediamo ora il caso (molto usato) della splines cubica $m = 3$ abbinata alla scelta *naturale* delle condizioni aggiuntive:

$$p_0^{(2)}(x_0) = 0 \tag{50}$$

$$p_{N-1}^{(2)}(x_N) = 0 \tag{51}$$

La derivata seconda della spline sarà una funzione lineare in ogni intervallo $[x_i, x_{i+1}]$ e continua. Pertanto possiamo trovarla andando ad interpolare in ogni intervallo, con $x \in [x_i, x_{i+1}]$:

$$p_i^{(2)}(x) = \frac{1}{h_i}((x - x_i)p_{i+1}^{(2)} - (x - x_{i+1})p_i^{(2)}) \tag{52}$$

dove $h_i = x_{i+1} - x_i$. Ora integriamo due volte e troviamo:

$$p_i(x) = \alpha_i(x - x_i)^3 + \beta_i(x - x_{i+1})^3 + \gamma_i(x - x_i) + \eta_i(x - x_{i+1}) \tag{53}$$

con:

$$\alpha_i = \frac{p_{i+1}^{(2)}}{6h_i} \tag{54}$$

$$\beta_i = -\frac{p_i^{(2)}}{6h_i} \tag{55}$$

$$\gamma_i = -\alpha_i h_i^2 + \frac{f_{i+1}}{h_i} = -\frac{p_{i+1}^{(2)} h_i}{6} + \frac{f_{i+1}}{h_i} \tag{56}$$

$$\eta_i = -\beta_i h_i^2 - \frac{f_i}{h_i} = \frac{p_i^{(2)} h_i}{6} - \frac{f_i}{h_i} \tag{57}$$

dove le espressioni per γ_i e η_i sono state trovate imponendo $p_i(x_i) = f_i$ e $p_i(x_{i+1}) = f_{i+1}$. Quindi per determinare la spline dobbiamo trovare le $p_i^{(2)}$. Applichiamo ora la condizione:

$$p_{i-1}^{(1)}(x_i) = p_i^{(1)}(x_i) \tag{58}$$

si trova:

$$3\alpha_{i-1}h_{i-1}^2 + \gamma_{i-1} + \eta_{i-1} = 3\beta_i h_i^2 + \gamma_i + \eta_i \tag{59}$$

$$\frac{p_i^{(2)}h_{i-1}}{2} - \frac{p_i^{(2)}h_{i-1}}{6} + \frac{f_i}{h_{i-1}} + \frac{p_{i-1}^{(2)}h_{i-1}}{6} - \frac{f_{i-1}}{h_{i-1}} = -\frac{p_i^{(2)}h_i}{2} - \frac{p_{i+1}^{(2)}h_i}{6} + \frac{f_{i+1}}{h_i} + \frac{p_i^{(2)}h_i}{6} - \frac{f_i}{h_i} \tag{60}$$

che possiamo scrivere come:

$$h_{i-1}p_{i-1}^{(2)} + 2(h_{i-1} + h_i)p_i^{(2)} + h_i p_{i+1}^{(2)} = 6\left(\frac{g_i}{h_i} - \frac{g_{i-1}}{h_{i-1}}\right) \tag{61}$$

dove abbiamo posto $g_i = f_{i+1} - f_i$. Ricordiamo che abbiamo scelto le condizioni aggiuntive $p_0^{(2)} = 0$ e $p_N^{(2)} = 0$. Il problema può essere dunque posto in forma matriciale:

$$\begin{pmatrix} d_1 & h_1 & 0 & \dots & \dots & 0 \\ h_1 & d_2 & h_2 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & h_{N-3} & d_{N-2} & h_{N-2} \\ 0 & \vdots & \vdots & 0 & h_{N-2} & d_{N-1} \end{pmatrix} \begin{pmatrix} p_1^{(2)} \\ p_2^{(2)} \\ \vdots \\ p_{N-2}^{(2)} \\ p_{N-1}^{(2)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N-2} \\ b_{N-1} \end{pmatrix} \tag{62}$$

con $d_i = 2(h_{i-1} + h_i)$ e

$$b_i = 6\left(\frac{g_i}{h_i} - \frac{g_{i-1}}{h_{i-1}}\right) \tag{63}$$

Quindi abbiamo riformulato il problema in:

$$\mathbf{A}\mathbf{p}^{(2)} = \mathbf{b} \tag{64}$$

Dove la matrice $\mathbf{A}$ è tridiagonale:

$$A_{ij} = \begin{cases} d_i \text{ se } i = j \\ h_{i-1} \text{ se } j = i - 1 \\ h_i \text{ se } j = i + 1 \\ \text{altrimenti} \end{cases} \tag{65}$$

Illustriamo ora un metodo iterativo per risolvere il problema $\mathbf{A}\mathbf{z} = \mathbf{b}$ con $\mathbf{A}$ matrice tridiagonale e $\mathbf{z}$ vettore incognita. Per prima cosa decomponiamo $\mathbf{A} = \mathbf{L}\mathbf{U}$ con $\mathbf{L}$ matrice triangolare inferiore $L_{ij} = 0$ per $j > i$ e $\mathbf{U}$ matrice triangolare superiore $U_{ij} = 0$ per $j < i$. Scegliamo $U_{ii} = 1$ chiamata fattorizzazione di Crout. Usiamo la notazione:

$$A = \begin{pmatrix} d_1 & e_1 & 0 & \dots & \dots & 0 \\ c_1 & d_2 & e_2 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & c_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \vdots & \vdots & 0 & c_{N-2} & d_{N-1} \end{pmatrix} \tag{66}$$

$$\mathbf{L} = \begin{pmatrix} w_1 & 0 & 0 & \dots & \dots & 0 \\ v_1 & w_2 & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & v_{N-3} & w_{N-2} & 0 \\ 0 & \vdots & \vdots & 0 & v_{N-2} & w_{N-1} \end{pmatrix} \tag{67}$$

e

$$\mathbf{U} = \begin{pmatrix} 1 & t_1 & 0 & \dots & \dots & 0 \\ 0 & 1 & t_2 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & 0 & 1 & t_{N-2} \\ 0 & \vdots & \vdots & 0 & 0 & 1 \end{pmatrix} \tag{68}$$

Troviamo dunque il sistema di equazioni:

$$\begin{cases} w_1 = d_1 \\ t_{i-1}v_{i-1} + w_i = d_i \\ w_i t_i = e_i \\ v_{i-1} = c_{i-1} \end{cases} \tag{69}$$

che possiamo risolvere in maniera iterativa: prima poniamo $v_1 = c_1$ e $w_1 = d_1$ poi usiamo $w_i = d_i - t_{i-1}v_{i-1}$ e $t_i = e_i/w_i$, dove si intende che $t_0 = v_0 = 0$.

Ora possiamo risolvere $\mathbf{LUz} = \mathbf{b}$. Procediamo in due passi entrambi iterativi. Cominciamo con trovare $\mathbf{y}$ che risolve $\mathbf{Ly} = \mathbf{b}$:

$$\begin{cases} w_1 y_1 = b_1 \\ v_{i-1}y_{i-1} + w_i y_i = b_i \end{cases} \tag{70}$$

poniamo dunque $y_1 = b_1/w_1$ e poi iteriamo con:

$$y_i = \frac{b_i - v_{i-1}y_{i-1}}{w_i} \tag{71}$$

Infine risolviamo $\mathbf{Uz} = \mathbf{y}$. Abbiamo:

$$\begin{cases} z_{N-1} = y_{N-1} \\ z_i + t_i z_{i+1} = y_i \end{cases} \tag{72}$$

poniamo dunque $z_{N-1} = y_{N-1}$ e poi iteriamo, con verso opposto, con:

$$z_i = y_i - t_i z_{i+1} \tag{73}$$

6 Serie di Fourier discreta e Fast Fourier Transform

Consideriamo ora una funzione periodica di periodo L:

$$f(x + L) = f(x) \tag{74}$$

che conosciamo su di una griglia di N punti equispaziati, per cui vale: $f_0 = f_N$. La serie di Fourier di f permettere di scrivere tale funzione come:

$$f(x) = \frac{1}{2}a_0 + \sum_{m=1,+\infty} \left(a_m \cos(\frac{2\pi}{L}mx) + b_m \sin(\frac{2\pi}{L}mx) \right) \tag{75}$$

Possiamo calcolare la corrispondente trasformata di Fourier discreta che possiamo poi usare per interpolare la funzione f. I coefficienti a_m e b_m sono dati da:

$$a_m = \frac{2}{L}\int_0^L f(x)\cos(\frac{2\pi}{L}mx)dx \tag{76}$$

$$b_m = \frac{2}{L}\int_0^L f(x)\sin(\frac{2\pi}{L}mx)dx \tag{77}$$

$$\tag{78}$$

Infatti, se consideriamo a_m

$$\int_0^L f(x)\cos(\frac{2\pi}{L}mx) = \frac{1}{2}a_0\int_0^L \cos(\frac{2\pi}{L}mx)dx \tag{79}$$

$$+\int_0^L \sum_{m'=1,+\infty} a'_m \cos(\frac{2\pi}{L}m'x)\cos(\frac{2\pi}{L}mx)dx \tag{80}$$

$$+\int_0^L \sum_{m'=1,+\infty} b'_m \sin(\frac{2\pi}{L}m'x)\cos(\frac{2\pi}{L}mx)dx \tag{81}$$

$$\tag{82}$$

Possiamo usare i seguenti integrali indefiniti:

$$\int \cos(\alpha x)\cos(\beta x)dx = \frac{\sin((\alpha-\beta)x)}{2(\alpha-\beta)} + \frac{\sin((\alpha+\beta)x)}{2(\alpha+\beta)} \quad \text{per} \quad |\alpha| \neq |\beta| \tag{83}$$

$$\int \cos^2(\alpha)dx = \frac{1}{2}x + \frac{1}{4\alpha}\sin(2\alpha x) \tag{84}$$

$$\int \sin(\alpha x)\cos(\beta x)dx = -\frac{\cos((\alpha+\beta)x)}{2(\alpha+\beta)} - \frac{\cos((\alpha-\beta)x)}{2(\alpha-\beta)} \quad \text{per} \quad |\alpha| \neq |\beta| \tag{85}$$

$$\tag{86}$$

Da cui si ottiene facilmente le formule di Eq. 76.

Spesso è conveniente, anche per una funzione f reale, scrivere:

$$f(x) = \sum_{m=-\infty,\infty} c_m e^{i\frac{2\pi}{L}mx} \tag{87}$$

dove:

$$c_m = \frac{1}{L}\int_0^L f(x)e^{-i\frac{2\pi}{L}mx}dx \tag{88}$$

Infatti:

$$\frac{1}{L}\int_0^L f(x)e^{-i\frac{2\pi}{L}mx}dx = \frac{1}{L}\int_0^L \left(\sum_{m'=-\infty,\infty} c_{m'}e^{i\frac{2\pi}{L}m'x}\right)e^{-i\frac{2\pi}{L}mx}dx \tag{89}$$

$$= \frac{1}{L}\sum_{m'=-\infty,\infty}\int_0^L e^{i\frac{2\pi}{L}(m'-m)x}dx \tag{90}$$

$$= \frac{1}{L}\sum_{m'=-\infty,\infty} Lc_{m'}\delta_{m',m} \tag{91}$$

$$= c_m \tag{92}$$

Se f è reale possiamo ricavare le seguente relazione:

$$f(x) = f^*(x) \tag{93}$$

$$\sum_m c_m e^{i\frac{2\pi}{L}mx} = \sum_{m'} c^*_{m'}e^{-i\frac{2\pi}{L}m'x} \tag{94}$$

da cui, per f reale:

$$c_m = c^*_{-m} \tag{95}$$

E' interessante il caso in cui la funzione periodica f è conosciuta su di N punti equispaziati x_i con $i = 0, N-1$ e con $x_N - x_0 = L$. La condizione di periodicità viene scritta con: $f_{i+N} = f_i$. La serie di Fourier discreta viene definita con:

$$f_i = \frac{1}{N}\sum_{j=0,N-1} g_j e^{i\frac{2\pi}{L}jx_i} \tag{96}$$

con:

$$g_i = \sum_{j=0,N-1} f_j e^{-i\frac{2\pi}{L}ix_j} \tag{97}$$

Usando la relazione $x_i = iL/N$ possiamo scrivere:

$$f_i = \frac{1}{N} \sum_{j=0,N-1} g_j e^{i\frac{2\pi}{N}ji} \tag{98}$$

$$g_i = \sum_{j=0,N-1} f_j e^{-i\frac{2\pi}{N}ji} \tag{99}$$

Infatti:

$$\sum_{j=0,N-1} f_j e^{-i\frac{2\pi}{N}ji} = \sum_{j=0,N-1} \frac{1}{N} \sum_{j'=0,N-1} g_{j'} e^{i\frac{2\pi}{N}j'j} e^{-i\frac{2\pi}{N}ji} \tag{100}$$

$$= \frac{1}{N} \sum_{j'=0,N-1} \sum_{j=0,N-1} g_{j'} e^{i\frac{2\pi}{N}(j'-i)j} \tag{101}$$

$$= \frac{1}{N} \sum_{j'=0,N-1} g_{j'} N \delta_{j',i} \tag{102}$$

$$= g_i \tag{103}$$

In Eq. 98 abbiamo usato la stessa convenzione di segni e del fattore $1/N$ usata nella libreria di python *scipy*. Notiamo che sia la f che la funzione inversa g sono periodiche, pertanto:

$$g_{-i} = g_{N-i} \tag{104}$$

A prima vista il calcolo della serie discreta di Fourier scala con il numero N di punti come $O(N^2)$. Fortunatamente è possibile raccogliere i termini entranti le sommatorie in Eq. 98 in maniera tale da ottenere un costo di ordine $O(N\ln(N))$. Tale algoritmo prende il nome Fast Fourier Transform (FFT).

6.1 Derivare una trasformata di Fourier

E' interessante notare come sia semplice l'operazione di derivata nel caso la funzione $f(x)$ sia espressa come serie di Fourier. Sia:

$$f(x) = \sum_m c_m e^{i\frac{2\pi}{L}mx} \tag{105}$$

allora

$$\frac{df(x)}{dx} = \sum_m \left(ic_m \frac{2\pi}{L} m \right) e^{i\frac{2\pi}{L}mx} \tag{106}$$

quindi in questo caso la derivata è esatta. Nel caso della trasformata discreta, la derivata si ottiene tramite la medesima corrispondenza:

$$g_m \to i g_m \frac{2\pi}{L} m \tag{107}$$

7 Risolvere equazioni

vogliamo risolvere la generica equazione:

$$f(x) = 0 \tag{108}$$

valutando f e, in caso la sua derivata, numericamente. Supponiamo che la funzione f sia continua.

7.1 Metodo della bisezione

Cerchiamo soluzioni nell'intervallo $[x_0, x_N]$ e consideriamo la solita griglia equispaziata h. Se tra i e $i+1$ il segno di f cambia, allora c'è almeno una soluzione nell'intervallo $[x_i, x_{i+1}]$. Notiamo che la condizione del cambio di segno può essere scritta come $f_i f_{i+1} < 0$. Supponiamo ora di voler trovare una soluzione con accuratezza ϵ. Usiamo il seguente algoritmo:

1. poniamo $a = x_i$ e $b = x_{i+1}$
2. poniamo $c = \frac{b-a}{2}$
3. se $f(c)f(a) < 0$ poniamo $b = c$ altrimenti poniamo $a = c$
4. se $b - a < \epsilon$ il risultato è $x_{sol} = \frac{b-a}{2}$ altrimenti torniamo al punto 2

7.2 Metodo di Newton-Raphson

Questo metodo sfrutta sia il calcolo di f sia di $f^{(1)} = \frac{df}{dx}$. L'idea è di approssimare f con una funzione lineare e di trovare una soluzione approssimata e di iterare. Cominciamo dal punto x_{trial}:

1. considero $f(x) \approx f(x_{trial}) + f^{(1)}(x_{trial})(x - x_{trial})$
2. risolvo e pongo $x_{sol} = x_{trial} - \frac{f(x_{trial})}{f^{(1)}(x_{trail})}$
3. se $|x_{trial} - x_{sol}| < \epsilon$ prendo la soluzione x_{sol} altrimenti pongo $x_{trial} = x_{sol}$ e riparto dal punto 1

Parte III
Equazioni differenziali ordinarie con condizioni iniziali

8 Propagare le equazioni del moto

Supponiamo di voler trovare la legge oraria di un punto materiale di massa m, vincolato a muoversi lungo la direzione x, su cui agisce la forza (diretta anch'essa lungo x) $F(x, v, t)$, dove $v = dx/dt$. Inoltre, supponiamo di conoscere all'istante iniziale t_0 i valori $x_0 = x(t_0)$, $v_0 = v(t_0)$. Ossia dobbiamo risolvere il sistema:

$$\begin{cases} \frac{dx}{dt}(t) & = v(t) \\ \frac{dv}{dt}(t) & = \frac{F(x(t),v(t),t)}{m} \end{cases} \tag{109}$$

con le condizioni iniziali:

$$\begin{cases} x_0 & = x(t_0) \\ v_0 & = v(t_0) \end{cases} \tag{110}$$

Per volontà di generalizzazione introduciamo il vettore $\mathbf{y}(t)$:

$$\mathbf{y}(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \tag{111}$$

cosicché il nostro problema viene scritto come:

$$\begin{cases} \frac{d\mathbf{y}}{dt} & = \mathbf{f}(\mathbf{y}(t), t) \\ \mathbf{y_0} & = \mathbf{y}(t_0) \end{cases} \tag{112}$$

dove abbiamo posto:

$$\mathbf{f}(\mathbf{y}(t), t) = \begin{pmatrix} v(t) \\ \frac{F(x(t),v(t),t)}{m} \end{pmatrix} \tag{113}$$

e

$$\mathbf{y}_0 = \begin{pmatrix} x_0 \\ v_0 \end{pmatrix} \tag{114}$$

Tale notazione ci permette non solo una facile generalizzazione nel caso di più gradi di libertà (es. dimensioni spaziali e numero di punti materiali) ma anche ci permette di generalizzare al caso di equazioni differenziali ordinarie di ordine arbitrario. Infatti, l'equazione generica:

$$\frac{d^n y}{dt^n}(t) = f\left(y(t), \frac{dy}{dt}(t), \frac{d^2y}{dt^2}(t), \ldots, \frac{d^{n-1}y}{dt^{n-1}}(t), t\right) \tag{115}$$

può essere posta nella stessa forma di Eq. 112 definendo:

$$\mathbf{y}(t) = \begin{pmatrix} y(t) \\ \frac{dy}{dt}(t) \\ \frac{d^2y}{dt^2}(t) \\ \cdots \\ \frac{dy^{n-1}}{dt^{n-1}}(t) \end{pmatrix} \tag{116}$$

e definendo

$$\mathbf{f}(\mathbf{y}(t), t) = \begin{pmatrix} \frac{dy}{dt}(t) \\ \frac{d^2y}{dt^2}(t) \\ \frac{d^3y}{dt^3}(t) \\ \cdots \\ f\left(y(t), \frac{dy}{dt}(t), \frac{d^2y}{dt^2}(t), \ldots, \frac{d^{n-1}y}{dt^{n-1}}(t), t\right) \end{pmatrix} \tag{117}$$

Per risolvere numericamente Eq. 112 discretizziamo l'asse temporale ponendo:

$$t_i = t_0 + i\Delta t \quad i \in \mathbb{Z} \tag{118}$$

e usiamo anche la notazione:

$$\mathbf{y}_i = \mathbf{y}\left(t_i\right) \tag{119}$$

Anche in questo caso cominciamo con l'individuare un metodo intuitivo per poi razionalizzarlo in metodi più accurati. Supponiamo di conoscere $\mathbf{y}_i$ e di voler calcolare $\mathbf{y}_{i+1}$, possiamo pensare che con una certa approssimazione si possa considerare $\mathbf{f}\left(\mathbf{y}\left(t\right), t\right)$ costante tra t_i e t_{i+1}, allora possiamo porre:

$$\mathbf{y}_{i+1} \approx \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\Delta t \tag{120}$$

tale metodo prende il nome di **METODO DI EULERO ESPLICITO** andiamo ora a studiarne l'accuratezza. Possiamo vedere la funzione $\mathbf{f}\left(\mathbf{y}\left(t\right), t\right)$ come funzione di solo t, allora:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}\left(\mathbf{y}\left(t\right), t\right) dt \tag{121}$$

e se applichiamo il metodo dei rettangoli naif, troviamo proprio il metodo di Eulero esplicito:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\Delta t + O\left(\Delta t^2\right) \tag{122}$$

Avremmo potuto anche applicare il metodo dei rettangoli naif scegliendo l'estremo di integrazione opposto, in questo caso si trova il **METODO DI EULERO IMPLICITO** :

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_{i+1}, t_{i+1}\right)\Delta t + O\left(\Delta t^2\right) \tag{123}$$

notiamo che il vettore $\mathbf{y}_{i+1}$ compare in ambo i membri dell'uguaglianza. Ciò significa che il problema di risolvere Eq. 123 è un problema *autoconsistente* e andrà risolto con tecniche iterative. Mentre la formula del metodo di Eulero esplicito, in Eq. 122 , è di immediata valutazione.

Come fatto nella Sezione 4 per ottenere un accuratezza maggiore andiamo a considerare più punti per l'integrazione. Il **METODO LEAP-FROG O METODO DI STÖRMER-VERLET** si ottiene applicando la regola dei rettangoli nell'intervallo $[t_{i-1}, t_{i+1}]$:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i-1}^{t_{i+1}} \mathbf{f}\left(\mathbf{y}\left(t\right), t\right) dt = \mathbf{y}_{i-1} + 2\mathbf{f}\left(\mathbf{y}_i, t_i\right)\Delta t + O\left(\Delta t^3\right) \tag{124}$$

come si vede tale metodo è *esplicito* ossia il termine a destra dell'uguale non dipende da termini in $i+1$ ed è quindi di immediata valutazione. Attenzione dovrà essere posta nel trovare il punto $\mathbf{y}_1$ visto che non conosciamo $\mathbf{y}_{-1}$.

Il **METODO DI CRANK-NICOLSON** si trova applicando il metodo dei trapezi all'intervallo $[t_i, t_{i+1}]$:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}\left(\mathbf{y}\left(t\right), t\right) dt = \mathbf{y}_i + \frac{1}{2}\left(\mathbf{f}\left(\mathbf{y}_{i+1}, t_{i+1}\right) + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\right)\Delta t + O\left(\Delta t^3\right) \tag{125}$$

esso è quindi un metodo *implicito*.

Per ottenere accuratezze ancora maggiori sono state elaborate diverse strategie. Di particolare diffusione sono quelle dei metodi di **RUNGE-KUTTA**. L'idea è di calcolare $\mathbf{y}_{i+1}$ usando valori (approssimati) per un insieme di punti $\{\mathbf{y}_{i+\epsilon}\}$ per tempi compresi tra t_i e t_{i+1}. A guisa d'esempio vediamo il metodo **EXPLICIT MIDPOINT EULERO**. Cominciamo con l'utilizzare la regola dei rettangoli:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}\left(\mathbf{y}\left(t\right), t\right) dt = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_{i+\frac{1}{2}}, t_{i+\frac{1}{2}}\right)\Delta t + O\left(\Delta t^3\right) \tag{126}$$

che offre un'accuratezza di $O\left(\Delta t^3\right)$. Però noi non conosciamo $\mathbf{y}_{i+\frac{1}{2}}$, per calcolarlo con precisione $O\left(\left(\frac{\Delta t}{2}\right)^2\right)$ possiamo usare il metodo di Eulero esplicito:

$$\mathbf{y}_{i+\frac{1}{2}} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\frac{\Delta t}{2} + O\left(\left(\frac{\Delta t}{2}\right)^2\right) \tag{127}$$

cosicché si trova:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\frac{\Delta t}{2}, t_{i+\frac{1}{2}}\right)\Delta t + O\left(\Delta t^2\right) \tag{128}$$

Come si vede l'accuratezza è dell'ordine $O\left(\Delta t^2\right)$ invece di $O\left(\Delta t^3\right)$ come potremmo aspettarci in prima battuta. Lo si può capire analizzando la dipendenza da Δt della funzione errore:

$$E\left(\Delta t\right) = \mathbf{y}_{i+1}\left(\Delta t\right) - \left(\mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_i, t_i\right)\frac{\Delta t}{2}, t_i + \frac{\Delta t}{2}\right)\Delta t\right) \tag{129}$$

dove, in questo caso, $\mathbf{y}_{i+1}\left(\Delta t\right)$ è inteso come il valore esatto. Espandiamo $E\left(\Delta t\right)$ in funzione di Δt:

$$E\left(\Delta t\right) = \mathbf{y}_{i+1}\left(0\right) + \frac{d\mathbf{y}_i}{dt}\Delta t + O\left(\Delta t^2\right) - \mathbf{y}_i - \left(\mathbf{f}\left(\mathbf{y}_i, t_i\right) + O\left(\frac{\Delta t}{2}\right)\right)\Delta t \tag{130}$$

dal fatto che $\mathbf{y}_{i+1}\left(0\right) = \mathbf{y}_i$ e che $\frac{d\mathbf{y}_i}{dt} = \mathbf{f}\left(\mathbf{y}_i, t_i\right)$, troviamo:

$$E\left(\Delta t\right) = O\left(\Delta t^2\right) \tag{131}$$

Nonostante l'esempio precedente i metodi di Runge-Kutta permettono di ottenere accuratezze migliori rispetto ai metodi visti prima. Il metodo di questo tipo più conosciuto ed usato è il metodo di **RUNGE-KUTTA** a 4 punti (normalmente chiamato semplicemente metodo di RUNGE_KUTTA). E' definito dal seguente schema che è di ordine $O\left(\Delta t^5\right)$:

$$\begin{cases} \mathbf{Y}_1 & = \mathbf{y}_i \\ \mathbf{Y}_2 & = \mathbf{y}_i + \mathbf{f}\left(\mathbf{Y}_1, t_i\right)\frac{\Delta t}{2} \\ \mathbf{Y}_3 & = \mathbf{y}_i + \mathbf{f}\left(\mathbf{Y}_2, t_i + \frac{\Delta t}{2}\right)\frac{\Delta t}{2} \\ \mathbf{Y}_4 & = \mathbf{y}_i + \mathbf{f}\left(\mathbf{Y}_3, t_i + \frac{\Delta t}{2}\right)\Delta t \\ \mathbf{y_{i+1}} & = \mathbf{y}_i + \left[\mathbf{f}\left(\mathbf{Y}_1, t_i\right) + 2\mathbf{f}\left(\mathbf{Y}_2, t_i + \frac{\Delta t}{2}\right) + 2\mathbf{f}\left(\mathbf{Y}_3, t_i + \frac{\Delta t}{2}\right) + \mathbf{f}\left(\mathbf{Y}_4, t_i\right)\right]\frac{\Delta t}{6} \end{cases} \tag{132}$$

8.1 Risolvere equazione auto-consistenti

Negli schemi impliciti compaiono equazioni di tipo auto-consistente in cui i coefficienti dell'equazione dipendono dalla soluzione stessa. Ad esempio nel metodo di Eulero implicito dobbiamo risolvere:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_{i+1}, t_{i+1}\right)\Delta t$$

Una possibilità è usare il seguente metodo iterativo: cominciamo con la prima stima di $\mathbf{y}_{i+1}$ che chiamiamo $\mathbf{y}_{i+1,0}$ e poniamo $\mathbf{y}_{i+1,0} = \mathbf{y}_i$, poi troviamo la seconda stima come $\mathbf{y}_{i+1,1} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_{i+1,0}, t_{i+1}\right)$, e la terza come $\mathbf{y}_{i+1,2} = \mathbf{y}_i + \mathbf{f}\left(\mathbf{y}_{i+1,1}, t_{i+1}\right)$, e continuiamo in questo modo finché $\mathbf{y}_{n+1} = \mathbf{y}_n$ a meno di un opportuno valore soglia.

8.2 Errore Metodologico

Come abbiamo visto nel caso di derivate, integrali, soluzione di equazioni differenziali, i vari metodi presentati offrono accuratezze diverse. Per errore metodologico si intende la differenza tra la soluzione esatta e la soluzione numerica, supponendo per questa di non avere errori dovuti alla precisione di calcolo (come il roundoff error). In pratica l'indicazione dell'accuratezza come in $O\left(\Delta t^n\right)$ è un'indicazione dell'errore metodologico.

8.3 Esempio: oscillatore armonico

Applichiamo ora a guisa d'esempio i metodi descritti nelle sezioni precedenti al caso di un oscillatore armonico unidimensionale. Le corrispondenti funzioni $\mathbf{y}\left(t\right)$ e $\mathbf{f}\left(\mathbf{y}, t\right)$ sono:

$$\mathbf{y}\left(t\right) = \begin{pmatrix} x\left(t\right) \\ v\left(t\right) \end{pmatrix} \tag{133}$$

e

$$\mathbf{f}\left(t\right) = \begin{pmatrix} v\left(t\right) \\ -\frac{k}{m}x\left(t\right) \end{pmatrix} \tag{134}$$

Per la nostra simulazione numerica scegliamo $m = 1$, $k = 1$, $x\left(t_0 = 0\right) = 0.1$, $v\left(t_0 = 0\right) = 0$, e $\Delta t = 0.1$. In Figura 7 (a) mostriamo l'evoluzione temporale di $x\left(t\right)$ calcolata con i metodi di Eulero esplicito e di Runge Kutta (ordine 4) per $N = 500$ punti sull'asse t. Ci si accorge che dopo le prime oscillazioni i due metodi differiscono e chiaramente l'energia meccanica del sistema non è conservata (come ci si aspetta invece per il sistema fisico) con il metodo di Eulero, infatti si vede che con tale metodo l'energia diverge (panello b).

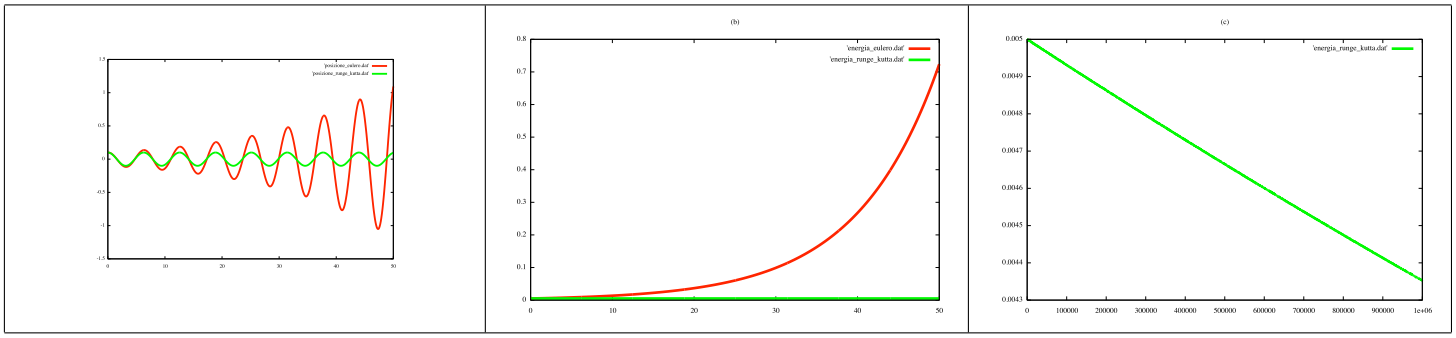


Figura 7: Evoluzione temporale di un oscillatore armonico (vedi testo). (a) funzione posizione calcolata con il metodo di Eulero esplicito (rosso) e con il metodo di Runge-Kutta (verde). (b) funzione energia meccanica calcolata con il metodo di Eulero esplicito (rosso) e con il metodo di Runge-Kutta (verde). (c) funzione energia meccanica calcolata con il metodo di Runge Kutta.

Dal panello (c) si vede che l'energia meccanica su scale temporali più grandi non è conservata nemmeno nel caso di Runge Kutta. In questo caso la variazione dell'energia è lineare rispetto al tempo sebbene tali variazioni siano molto piccole.

Nel caso ci interessino le proprietà di un sistema fisico (conservativo) su scali di tempo grandi dovrò utilizzare metodi di integrazione che garantiscano una sufficiente conservazione dell'energia.

9 Evoluzione temporale di un sistema Hamiltoniano

Per analizzare le proprietà dell'evoluzione temporale dei sistemi fisici consideriamo sistemi conservativi che quindi possono essere descritti tramite l'approccio di Hamilton.Ci ricordiamo che la funzione di Hamilton:

$$H\left(\{q_i\},\{p_i\}\right) = T\left(\{q_i\},\{p_i\}\right) + V\left(\{q_i\},\{p_i\}\right) \tag{135}$$

è pari all'energia E del sistema e valgono le relazioni

$$\frac{d}{dt}q_i = \frac{\partial}{\partial p_i}H \tag{136}$$

e

$$\frac{d}{dt}p_i = -\frac{\partial}{\partial q_i}H \tag{137}$$

per una funzione generica $A\left(\{q_i\},\{p_i\},t\right)$, che generalmente corrisponde ad una quantità osservabile, vale la relazione:

$$\frac{dA}{dt} = \{A,H\} + \frac{\partial A}{\partial t} \tag{138}$$

dove abbiamo usato le parentesi di Poisson:

$$\{A,B\} = \sum_i \left(\frac{\partial A}{\partial q_i}\frac{\partial B}{\partial p_i} - \frac{\partial A}{\partial p_i}\frac{\partial B}{\partial q_i}\right) \tag{139}$$

da cui si ricava subito la conservazione dell'energia. Il teorema di Liuville ci dice che se consideriamo ad un tempo t_0 un certo volume nello spazio delle fasi $\{q_i\}\times\{p_i\}$ tale volume si conserva se consideriamo la trasformazione $\{q_i\}\times\{p_i\} \to \{q_i'\}\times\{p_i'\}$, corrispondente all'evoluzione temporale del sistema da t_0 a t_1, allora il volume nello spazio delle fasi resta constante.

Anche i metodi per propagare le equazioni del moto che abbiamo visto precedentemente costituiscono delle mappe $\{q_i\}\times\{p_i\} \to \{q_i'\}\times\{p_i'\}$ se tali mappe preservano il volume vengono dette *simplettiche*. Quindi parleremo di metodi *simplettici* e metodi *non-simplettici*. Si è visto che i metodi *simplettici* conservano l'energia molto meglio degli altri metodi anche se in genere si osservano delle oscillazioni attorno ad un valor medio. I metodi: Eulero esplicito, Eulero implicito, midpoint explicit Euler, Runge-Kutta 4, sono tutti non-simplettici e quindi non conservano l'energia. Il metodo di Störmer- Verlet invece è simplettico per questa ragione è il metodo usato nelle applicazioni di dinamica molecolare.

Possiamo anche rendere simplettici i metodi visti precedentemente. In particolare il **metodo di Eulero simplettico** è definito dalla regola:

$$\begin{cases} q_{i,n+1} &= q_{i,n} + \frac{\partial H(\mathbf{q}_n,\mathbf{p}_{n+1})}{\partial p_i}\Delta t \\ p_{i,n+1} &= p_{i,n} - \frac{\partial H(\mathbf{q}_n,\mathbf{p}_{n+1})}{\partial q_i}\Delta t \end{cases} \tag{140}$$

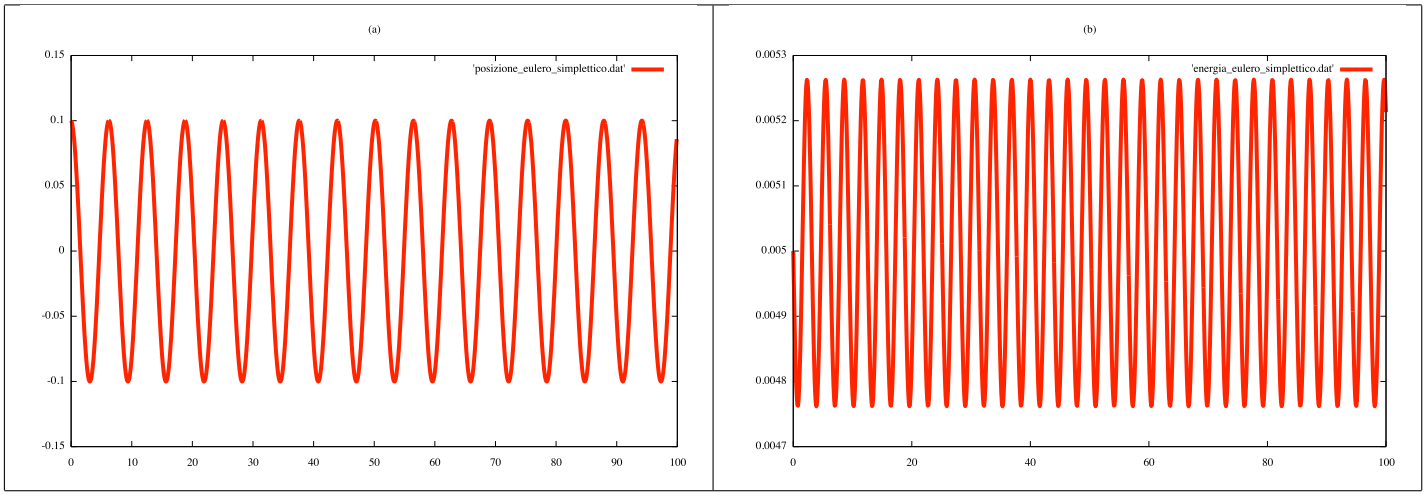


Figura 8: Oscillatore armonico calcolato con il metodo di Eulero simplettico: (a) funzione posizione (b) funzione energia

dove con n indichiamo l'indice di tempo, con i, l'indice di coordinata e con $\mathbf{q}$ e $\mathbf{p}$ l'insieme delle coordinate e dei relativi momenti generalizzati. E'interessante vedere l'applicazione di tale metodo nel caso di un sistema descritto da coordinate cartesiane in cui l'energia potenziale dipende solo dalle coordinate:

$$H(\mathbf{q},\mathbf{p}) = \sum_{i=1,N} \frac{p_i^2}{m_i} + V(\mathbf{q}) \tag{141}$$

dove m_i è la massa relativa al punto materiale a cui appartiene la i-esima coordinata e N è il numero di coordinate. Vale $p_i = m_i v_i = m_i \frac{dq_i}{dt}$. Applicando Eq. 140, troviamo:

$$\begin{cases} q_{i,n+1} & = q_{i,n} + \frac{p_{i,n+1}}{m_i}\Delta t + O\left(\Delta t^2\right) \\ p_{i,n+1} & = p_{i,n} + F_i(\mathbf{q}_n)\Delta t + O\left(\Delta t^2\right) \end{cases} \tag{142}$$

dove la forza agente sulla coordinata i è data da $F_i(\mathbf{q}) = -\frac{\partial V(\mathbf{q})}{\partial q_i}$. L'uso di tale metodo risulta semplice: prima computo tutti i $p_{i,n+1}$ e successivamente calcolo i $q_{i,n+1}$. In Figura 8 riportiamo i risultati per l'oscillatore armonico usando gli stessi parametri usati nella sezione precedente. Si vede chiaramente che l'energia meccanica oscilla attorno ad un valore costante.

Si può vedere che il valore medio dell'energia resta costante anche sui tempi grandi.

9.1 Sistemi sottoposti a vincoli

Consideriamo un sistema che sia sottoposto a uno (o più) vincoli olonomi riguardanti le sole coordinate spaziali, ossia scrivibile come:

$$f(\mathbf{q}) = a \tag{143}$$

ad esempio potrei studiare un pendolo semplice in coordinate cartesiane e imporre che la distanza tra massa e un punto fisso resti sempre costante. Posso inserire il vincolo nell'Hamiltoniana tramite l'introduzione di un moltiplicatore di Lagrange:

$$H(\mathbf{q},\mathbf{p}) \to H(\mathbf{q},\mathbf{p}) - \lambda\left(f(\mathbf{q}) - a\right) \tag{144}$$

questo risulta in un termine forza aggiuntivo:

$$\frac{d}{dt}p_i = -\frac{\partial}{\partial q_i}H' = F_i(\mathbf{q}) + G_i \tag{145}$$

dove la forza G_i dovuta al vincolo è data da:

$$G_i = \lambda\frac{\partial f(\mathbf{q})}{\partial q_i} \tag{146}$$

Per utilizzare il metodo di Eulero Simplettico nel caso di uno (o più) vincoli, possiamo usare il seguente algoritmo:

1. Calcolo i p_i in funzione di λ: $p_{i,n+1}(\lambda) = p_i + F_i(\mathbf{q})\Delta t + \lambda\frac{\partial f(\mathbf{q})}{\partial q_i}$

2. Calcolo i q_i in funzione dei $p_{i,n+1}(\lambda)$: $q_{i,n+1} = q_i + \frac{p_{i+1}(\lambda)}{m_i}\Delta t$

3. Scelgo λ in modo tale che il vincolo sia soddisfatto per i punti $q_{i,n+1}$

10 Dinamica molecolare

La possibilità di ricavare il comportamento fisico su scala macroscopica di un sistema a partire dal comportamento microscopico dei sui componenti (ad esempio il comportamento di un gas a partire dalle molecole che lo compongono) è uno dei grandi obbiettivi della ricerca fisica moderna. Grazie ai calcolatori moderni possiamo simulare sistemi contenenti fino a qualche milione di atomi. Anche se questo numero è molto minore del numero di Avogadro è lo stesso sufficiente per calcolare una grande quantità di grandezze interessanti.

10.1 L'approssimazione di Born-Oppenheimer

Consideriamo un sistema quantistico composto da N_{at} nuclei atomici puntiformi, di carica Z_I e massa M_I con I indice di nucleo, e da N_{el} elettroni. Nel limite non relativistico il sistema è descritto dalla seguente eq. di Schroedinger tempo indipendente (tralasciamo qui i gradi di libertà degli spin elettronici):

$$\left(\sum_{I=1,N_{at}} -\frac{\hbar^2}{2M_I}\nabla^2_{\mathbf{R}_I} + \sum_{i=1,N_{el}} -\frac{\hbar^2}{2m}\nabla^2_{\mathbf{r}_i} + \sum_{Ii}\frac{eZ_i}{4\pi\epsilon_0|\mathbf{R_I}-\mathbf{r_i}|} + \frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0|\mathbf{r_i}-\mathbf{r_j}|} + \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{4\pi\epsilon_0|\mathbf{R_I}-\mathbf{R_J}|}\right)\times \tag{147}$$

$$\Psi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}},\mathbf{r}_1,..,\mathbf{r}_{N_{el}}) = E_{tot}\Psi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}},\mathbf{r}_1,..,\mathbf{r}_{N_{el}}) \tag{148}$$

dove e è la carica ed m la massa dell'elettrone. Grazie alla grande differenza di massa tra neutrone/protone ed elettrone, è possibile separe i gradi di libertà elettronici da quelli dei nuclei e scrivere:

$$\Psi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}},\mathbf{r}_1,..,\mathbf{r}_{N_{el}}) = \Phi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}})\psi(\mathbf{r}_1,..,\mathbf{r}_{N_{el}};\mathbf{R}_1,..,\mathbf{R}_{N_{at}}) \tag{149}$$

dove i gradi di libertà dei nuclei entrano come parametri nella funzione d'onda elettronica ψ:

$$\left(\sum_{i=1,N_{el}} -\frac{\hbar^2}{2m}\nabla^2_{\mathbf{r}_i} + \sum_{Ii}\frac{eZ_i}{4\pi\epsilon_0|\mathbf{R_I}-\mathbf{r_i}|} + \frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0|\mathbf{r_i}-\mathbf{r_j}|}\right)\psi(\mathbf{r}_1,..,\mathbf{r}_{N_{el}};\mathbf{R}_1,..,\mathbf{R}_{N_{at}}) = E_{el}(\mathbf{R}_1,..,\mathbf{R}_{N_{at}})\psi(\mathbf{r}_1,..,\mathbf{r}_{N_{el}};\mathbf{R}_1,..,\mathbf{R}_{N_{at}} \tag{150}$$

Il termine $E_{el}(\mathbf{R}_1,..,\mathbf{R}_{N_{at}})$ è quindi l'energia potenziale per il sistema dei gradi di libertà nucleari:

$$\left(\sum_{I=1,N_{at}} -\frac{\hbar^2}{2M_I}\nabla^2_{\mathbf{R}_I} + \sum_{IJ}\frac{Z_I Z_J}{4\pi\epsilon_0|\mathbf{R_I}-\mathbf{R_J}|} + E_{el}(\mathbf{R}_1,..,\mathbf{R}_{N_{at}})\right)\Phi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}}) = E_{tot}\Phi(\mathbf{R}_1,..,\mathbf{R}_{N_{at}}) \tag{151}$$

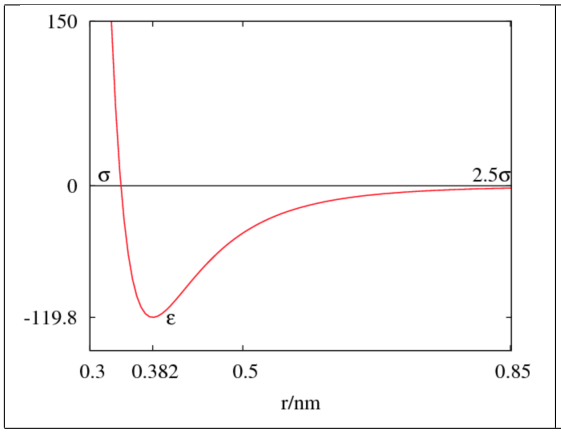
In molti casi è possibile fare un'ulteriore approssimazione e trattare i nuclei atomici come particelle classiche. In tale caso la funzione E_{el} diviene la funzione energia potenziale del sistema. Le equazioni del moto diventano:

$$M_I\frac{d^2}{dt^2}R_I(t) = -\nabla_I E_{el}(\mathbf{R}_1,..,\mathbf{R}_{N_{at}}) + \sum_{J\neq I}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|^2}\frac{4\pi\epsilon_0\mathbf{R}_I-\mathbf{R}_J}{|\mathbf{R}_I-\mathbf{R}_J|} \tag{152}$$

10.2 Density functional theory

La teoria del funzionale della densità o *density functional theory* (DFT) è una della approssimazioni più usate per trattare il problema quantistico elettronico e trovare E_{el}. Infatti una trattazione diretta del problema presenterebbe un costo di calcolo che varia esponenzialmente con il numero di elettroni. Questo è dovuto che per rappresentare, o meglio immagazzinare nel computer, una funzione d'onda generica a N elettroni interagenti abbiamo bisogno di M^{3N} punti dove M sono i punti di griglia che usiamo in una singola direzione Cartesiana. La DFT è una teoria rigorosa per trasformare il problema in quello di N elettroni *non*-interagenti. In questo caso ho da rappresentare un N volte una funzione d'onda per un singolo elettrone. Quindi, in totale, ho NM^3 punti da immagazzinare. Le N funzioni d'onda sono poi tra di loro ortogonali. Esse sono soluzioni delle equazioni dette di Kohn-Sham:

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + \sum_{I=1,N_{at}}\frac{eZ_I}{4\pi\epsilon_0\,|\mathbf{r}-\mathbf{R}_I|} + \int d\mathbf{r}'\frac{en(\mathbf{r})}{4\pi\epsilon_0\,|\mathbf{r}-\mathbf{r}'|} + V_{xc}[n](\mathbf{r})\right)\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}) \tag{153}$$



Compound	(ε/k) (K)	σ (10⁻¹² m)	Compound	(ε/k) (K)	σ (10⁻¹² m)
Ar	111.84	362.3	CH_4	140.42	401.5
Kr	154.87	389.5	CD_4	138.16	402.3
Xe	213.96	426.0	C_2H_6	216.12	478.2
O_2	113.27	365.4	nC_3H_8	255.18	547.1
N_2	91.85	391.9	nC_4H_{10}	287.20	608.1
F_2	104.29	357.1	nC_5H_{12}	309.75	670.9
Cl_2	296.27	448.5	nC_6H_{14}	327.47	731.9
CF_4	155.31	491.1	nC_7H_{16}	340.97	790.2
CCl_4	378.86	624.1	cC_3H_6	277.46	509.2
SiH_4	193.65	453.9	cC_5H_{10}	346.11	610.0
SiF_4	140.14	669.2	C_6H_6	377.46	617.4
$SiCl_4$	338.01	668.2	C_2H_2	209.11	463.5
$TiCl_4$	417.49	672.5	C_2H_4	200.78	458.9
SF_6	206.85	578.3	CO_2	201.71	444.4
UF_6	323.61	637.9	CS_2	388.81	498.3

Figura 9: Potenziale di Lennard-Jones (sinistra) e alcuni valori tipici per i parametri ϵ e σ (destra)

dove n è la densità elettronica:

$$n(\mathbf{r}) = \sum_{i=1,N_{el}} \psi_i^*(\mathbf{r})\psi(\mathbf{r}) \tag{154}$$

$V_{xc}[n]$ è un funzionale (funzione di funzione) della densità detto di *correlazione e scambio*. Questo significa che le equazioni di Kohn-Sham sono auto-consistenti. Sebbene la forma esatta di V_{xc} non sia conosciuta esistono approssimazioni molto efficaci. Le più semplici sono locali, ossia possono essere scritte come: $V_{xc}[n(\mathbf{r})](\mathbf{r})$ o $V_{xc}[n(\mathbf{r}), \nabla n(\mathbf{r})](\mathbf{r})$. L'energia del sistema è poi ottenuta da:

$$E_{el}(\mathbf{R}_1, .., \mathbf{R}_{N_{at}}) = \sum_{i=1,N_{el}} \langle\psi_i| - \frac{\hbar^2}{2m}\nabla^2|\psi_i\rangle + \sum_I \int d\mathbf{r} \frac{eZ_I n(\mathbf{r}')}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{R}_I|} + \frac{1}{2}\int d\mathbf{r} \int d\mathbf{r}' \frac{e^2 n(\mathbf{r})n(\mathbf{r}')}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \tag{155}$$

10.3 Force fields empirici

Invece di calcolare E_{el} tramite metodi di meccanica quantistica è anche possibile usare delle forme approssimate molto semplici ed empiriche detti campi di forza o *force fields*. Il più semplice è stato introdotto nel 1931 da Lennard-Jones. Consideriamo per semplicità un sistema di N atomi (o molecole) di una sola specie di massa m. Sia $\mathbf{r}_i$ la posizione dell' i-esimo atomo e $\mathbf{v}_i$ la sua velocità. **Lennard-Jones** introdusse una semplice approssimazione per l'energia potenziale data da una coppia di atomi i, j che dipende solamente dalla loro distanza $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$:

$$U_{ij} = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right] \tag{156}$$

tale energia potenziale è riportata in Figura 9. Il termine di potenza 6 è attrattivo e agisce sulle distanze di separazione grandi e orgina la forza di Van der Waals. Il termine di potenza 12 è repulsivo ed agisce sulle corte distante. L'energia ϵ corrisponde alla profondità della buca di potenziale, mentre σ definisce il raggio d'azione della componente repulsiva. In Figura 9 riportiamo anche dei valore tipici usati nelle simulazioni.

L'energia totale del sistema viene quindi scritta come:

$$U = \frac{1}{2}\sum_{i \neq j} U_{ij} \tag{157}$$

dove il fattore $\frac{1}{2}$ evita di contare due volte la stessa coppia di atomi. Siccome per ogni coppia di atomi i, j il loro contributo all'energia potenziale dipende solo dalla distanza relativa r_{ij} allora la corrispondente forza sarà centrale e quindi diretta lungo l'asse $\mathbf{r}_{ij} = \mathbf{r_i} - \mathbf{r}_j$. Si può facilmente mostrare che tale contributo (agente sull'atomo i) assume la forma seguente:

$$\mathbf{f}_i = \frac{24\epsilon}{r_{ij}^2} \left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right] (\mathbf{r_i} - \mathbf{r}_j) \tag{158}$$

11 I propagatori di Verlet

Per integrare (propagare) le equazioni del moto nei problemi di dinamica molecolare sono richiesti metodi simplettici che quindi conservino l'energia. I metodi più usati sono quelli di Verlet.

METODO LEAP FROG: usiamo l'integrazione con il metodo dei rettangoli come in Eq. 124 ma consideriamo i tempi sui cui calcoliamo le velocità traslati di $\Delta t/2$:

1. Inizializzo l'algoritmo calcolando le velocità, con l'algoritmo di Eulero, $\mathbf{v}_{i,\frac{1}{2}} = \mathbf{v}_{i,0} + \frac{1}{2m_i}\mathbf{f}_{i,0}\Delta t + O\left(\Delta t^2\right)$
2. Calcolo le posizioni $\mathbf{r}_{i,n+1} = \mathbf{r}_{i,n} + \mathbf{v}_{i,n+\frac{1}{2}}\Delta t + O\left(\Delta t^3\right)$
3. Calcolo le velocità:$\mathbf{v}_{i,n+\frac{1}{2}} = \mathbf{v}_{i,n-\frac{1}{2}} + \frac{1}{m_i}\mathbf{f}_{i,n}\Delta t + O\left(\Delta t^3\right)$
4. Torno al punto 2 e continuo il ciclo.

Si vede che il primo passo ha un'accuratezza $O\left(\Delta t^2\right)$, normalmente questo non è un problema perché nella dinamica molecolare di solito interessano le medie su tempi grandi, ad ogni modo per migliorare l'accuratezza del primo passo potrei usare un' algoritmo come Crank-Nicolson

METODO STÖRMER-VERLET: questo algoritmo coinvolge il calcolo delle sole posizioni:

Si parte dalla formula di Eq. 24 per $\frac{d^2\mathbf{r}_{i,n}}{dt^2}$:

$$\frac{d^2\mathbf{r}_{i,n}}{dt^2} = \frac{\mathbf{r}_{i,n+1} - 2\mathbf{r}_{i,n} + \mathbf{r}_{i,n-1}}{\Delta t^2} + O\left(\Delta t^2\right) = \frac{1}{m_i}\mathbf{f}_{i,n} \tag{159}$$

risolvendo per $\mathbf{r}_{i,n+1}$, troviamo:

$$\mathbf{r}_{i,n+1} = 2\mathbf{r}_{i,n} - \mathbf{r}_{i,n-1} + \frac{1}{m}\mathbf{f}_{i,n}\Delta t^2 + O\left(\Delta t^4\right) \tag{160}$$

l'algoritmo risultante sarà:

1. Inizializzo calcolando, tramite Taylor, $\mathbf{r}_{i,1} = \mathbf{r}_{i,0} + \mathbf{v_{i,0}}\Delta t + \frac{1}{2}\frac{1}{m_i}\mathbf{f}_{i,0}\Delta t^2 + O\left(\Delta t^3\right)$
2. Calcolo $\mathbf{r}_{i,n+1} = 2\mathbf{r}_{i,n} - \mathbf{r}_{i,n-1} + \frac{1}{m_i}\mathbf{f}_{i,n}\Delta t^2 + O\left(\Delta t^4\right)$
3. Torno a 2 e continuo il ciclo

Questo algoritmo non involve il calcolo delle velocità, queste possono essere calcolate come $\mathbf{v}_{i,n} = \frac{\mathbf{r}_{i,n+1} - \mathbf{r}_{i,n-1}}{2\Delta t}$ ma con accuratezza solo di $O\left(\Delta t^2\right)$.

METODO VELOCITY-VERLET: questo algoritmo permette maggior accuratezza per la velocità e risulta essere il più usato. Si parte dallo sviluppo in serie di Taylor:

$$\mathbf{r}_{i,n+1} = \mathbf{r}_{i,n} + \mathbf{v}_{i,n}\Delta t + \frac{1}{2m_i}\mathbf{f}_{i,n}\Delta t^2 + O\left(\Delta t^3\right) \tag{161}$$

e poi si ricava la velocità tramite l'integrazione con il metodo dei trapezi:

$$\mathbf{v}_{i,n+1} = \mathbf{v}_{i,n} + \frac{1}{2m_i}\left(\mathbf{f}_{i,n} + \mathbf{f}_{i,n+1}\right)\Delta t + O\left(\Delta t^3\right) \tag{162}$$

In pratica l'algoritmo segue il seguente schema:

1. Calcolo $\mathbf{r}_{i,n+1} = \mathbf{r}_{i,n} + \mathbf{v}_{i,n}\Delta t + \frac{1}{2m_i}\mathbf{f}_{i,n}\Delta t^2 + O\left(\Delta t^3\right)$
2. Calcolo $\mathbf{v}_{i,n+1} = \mathbf{v}_{i,n} + \frac{1}{2m_i}\left(\mathbf{f}_{i,n} + \mathbf{f}_{i,n+1}\right)\Delta t + O\left(\Delta t^3\right)$
3. Ritorno a 1 e continuo il ciclo

12 Simulazione di sistemi estesi: condizioni al contorno

Se la nostra simulazione di dinamica molecolare riguarda un sistema isolato, come ad esempio una (nano)-goccia o un *cluster* di atomi non ci sono particolari accorgimenti riguardo il trattamento delle coordinate spaziali. Diverso il caso in cui vogliamo studiare un sistema esteso, come ad esempio un liquido o un solido. In questo caso visto che siamo limitati ad un numero relativamente piccolo di atomi (rispetto al numero di Avogadro) se confinassimo il nostro sistema in un volume dello spazio, che per semplicità considereremo sempre cubico, avremmo che gli effetti di superficie sarebbero troppo rilevanti da non permettere la simulazione del sistema esteso (o *bulk*).

Un metodo semplice per ovviare a questo problema è l'introduzione di **CONDIZIONI AL CONTORNO PERIODICHE**: consideriamo un sistema di N atomi in una cella di simulazione cubica di lato L. Gli atomi hanno coordinate $\mathbf{r}_i$ entro il cubo. Allora andiamo a considerare anche tutte le *repliche periodiche* del nostro sistema:

$$\mathbf{r}_{i,(i_x,i_y,i_z)} = \mathbf{r}_i + i_x L\hat{\mathbf{x}} + i_y L\hat{\mathbf{y}} + i_z L\hat{\mathbf{z}} \quad i_x, i_y, i_z \in \mathbb{Z} \tag{163}$$

in pratica è come se un atomo uscisse da una faccia del cubo per rientrare dalla faccia opposta. La generazione di un numero in principio infinito di repliche porta però a dei problemi nel calcolo delle forze e dell'energia. Nel caso di un'energia potenziale del tipo di Lennard-Jones, è stato mostrato che per $L >\sim 6\sigma$, possiamo limitarci a considerare per ogni coppia di indici $i, j = 1, N$ solo la particolare scelta di repliche (i_x, i_y, i_z) (j_x, j_y, j_z) , dette *primi* vicini, che minimizza la distanza tra

$$\left|\mathbf{r}_{i,(i_x,i_y,i_z)} - \mathbf{r}_{j,(j_x,j_y,j_z)}\right| \tag{164}$$

tale distanza minima la denotiamo allora di nuovo con r_{ij} e quindi possiamo trattare solo $\frac{1}{2}N \times (N-1)$ coppie di atomi e usare per l'energia potenziale del sistema e per le forze le formule in Eq. 156 e Eq. 158. Analogamente il termine cinetico verrà calcolato considerando solo gli N atomi di partenza senza le repliche.

13 Medie termodinamiche

Lo schema che abbiamo introdotto per le dinamiche molecolari conserva il volume della cella di simulazione, il numero di atomi e l'energia meccanica del sistema. Quindi stiamo descrivendo l'insieme *microcanonico*. Talvolta tale insieme termodinamico viene indicato come **NVE**. Siccome per grandi N, V, E (numero atomi, volume, energia meccanica) l'insieme microcanonico diventa equivalente a quello canonico, è sensato introdurre una funzione *temperatura* che in maniera analoga alla definizione di temperatura nel caso del sistema canonico associa un'energia $\frac{1}{2}k_B T$ ad ogni grado di libertà del sistema. Nel nostro caso avendo N atomi il numero di gradi di libertà spaziali è $3N-3$ visto che abbiamo 3 costanti del moto ossia il vettore quantità di moto del sistema. Quindi poniamo:

$$E_{kin} = (3N-3)\,\frac{1}{2}k_B T \tag{165}$$

dove l'energia cinetica E_{kin} è pari a:

$$E_{kin} = \sum_{i=1,N} \frac{1}{2} m_i \left|\mathbf{v}_i\right|^2 \tag{166}$$

da cui ricavo la temperatura T:

$$T = \frac{1}{(3N-3)\,k_B}\left(\sum_{i=1,N} m_i \left|\mathbf{v}_i\right|^2\right) \tag{167}$$

Durante la nostra simulazione NVE osserveremo che la temperatura non sarà costante ma oscillerà attorno ad un valore (costante nel caso di un lungo tempo di simulazione). Quindi mediando sul tempo otterremo $T \pm \Delta T$ con ΔT incertezza sulla temperatura. Da cui potremo poi calcolare altre medie termodinamiche relative a tale temperatura.

14 Pair correlation function

Anche un sistema in una fase non cristallina come nel caso di un gas o di un liquido presenta precise proprietà strutturali. Una funzione che permette di quantificarle è la *pair correlation function*:

$$g\left(r\right) = \left\langle \frac{1}{4\pi r^2}\frac{V}{N^2}\left(\sum_{i=1,N}\sum_{\substack{j=1,N\\ j\neq i}} \delta\left(r - r_{ij}\right)\right)\right\rangle \tag{168}$$

dove le parentesi $\langle \ldots \rangle$ indicano la media temporale. In pratica, poi, la funzione δ Dirac-delta viene approssimata con una funzione Gaussiana normalizzata. La funzione $g\left(r\right)$ ci dice data la posizione di un atomo del sistema la densità media di altri atomi ad una distanza r da esso. Infatti:

$$\int_{R_1}^{R_2} dr 4\pi r^2 \frac{N}{V} g(r) = \#\text{medio di atomi tra } R_1 \text{ e } R_2 \tag{169}$$

Nel caso di un sistema cristallino la $g\left(r\right)$ presenta una struttura a picchi anche per grandi r, mentre nel caso di un sistema in fase liquida o gassosa (o amorfa) tale funzione ha picchi solo per r piccoli e poi tende a 1.

La funzione $g\left(r\right)$ può essere misurata tramite analisi della diffrazione di neutroni o di raggi X. In figura 10 riportiamo le funzione di pair correlation misurate per Zolfo solido e liquido.

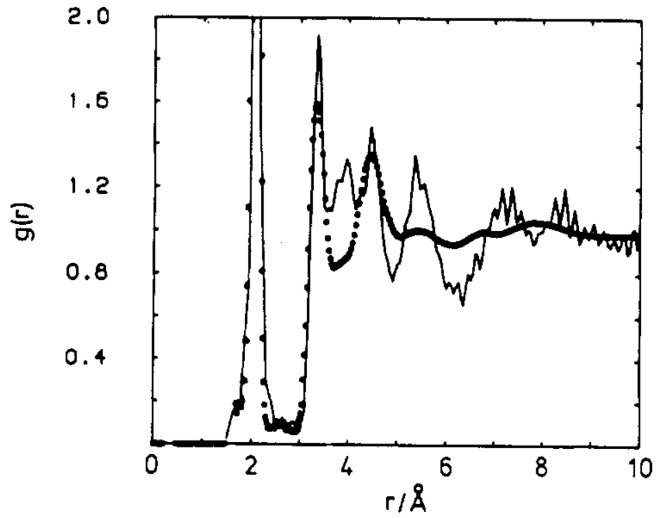


Figura 10: Funzione $g(r)$ misurate per Zolfo: (dischi) liquido (423 K) e (linea continua) solido (300 K). R Winter et al. J. Phys. Condens. Matt. 2, SA125 (1990).

15 La costante di diffusione

In un sistema liquido o gassoso variazioni $\rho'(\mathbf{r},t)$ dalla densità media ρ_0 sono descritte dall' equazione di diffusione:

$$\frac{\partial \rho'(\mathbf{r},t)}{\partial t} = D\nabla^2 \rho'(\mathbf{r},t) \tag{170}$$

dove D prende il nome di costante di diffusione e può venir misurata in laboratorio. Possiamo valutare D dall'analisi della nostra simulazione:

$$D = \lim_{t\to\infty} \frac{d}{dt}\frac{1}{6}\frac{1}{N}\sum_{i=1,N} |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2 \tag{171}$$

stando attenti a non aggiornare le coordinate $\mathbf{r}_i$ quando l'atomo i attraversa i lati della cella di simulazione. Tale formula deriva infatti dalla relazione:

$$\left\langle r^2 \right\rangle = 6Dt \tag{172}$$

Dall'analisi di D possiamo anche capire se stiamo descrivendo una fase liquida o una fase solida che potrebbe essere anche amorfa. Infatti la $g(r)$ è simile nei due casi. Se il sistema è solido troverò che D sarà nulla.

16 La configurazione di partenza

Per avviare una simulazione di dinamica molecolare di un sistema liquido o gassoso, devo fornire le posizioni iniziali e le velocità iniziali degli atomi. Supponiamo un gas o liquido atomico (es. Argon). Le coordinate possono essere generate in maniera casuale utilizzando uno dei generatori di numeri casuali che studieremo in seguito. Per velocizzare l'equilibrazione del sistema e per sceglierne (almeno parzialmente) la temperatura, le velocità degli atomi vengono generate secondo la distribuzione di Maxwell-Boltzmann:

$$f(v) = 2^{\frac{1}{2}}\pi^{-\frac{1}{2}} m^{\frac{3}{2}} (k_B T)^{-\frac{3}{2}} v^2 e^{-\frac{mv^2}{2k_B T}} \tag{173}$$

In pratica viene generato un vettore velocità $\mathbf{v}$ in maniera casuale. Poi viene generato un numero casuale c nell' intervallo $[0, \max(f)]$. Se $c \leq f(v)$ assegniamo all' atomo tale vettore velocità, altrimenti riproviamo generando casualmente un nuovo vettore $\mathbf{v}$. Tale algoritmo, che rivedremo in seguito, assicura che per un numero di atomi N grande la distribuzione delle loro velocità sia descritta da $f(v)$. Tale proprietà viene facilmente dimostrata considerando il numero di assegnazioni dati ad intervallini discretizzanti l'asse v ed usando la legge dei grandi numeri.

Una volta generata la configurazione iniziale ed avviata la dinamica, dovremmo considerare che il sistema avrà bisogno di un certo tempo per equilibrarsi. Una possibilità per capire se il sistema è in condizioni di equilibrio è valutare la costante di diffusione e vedere se essa resta stabile o meno.

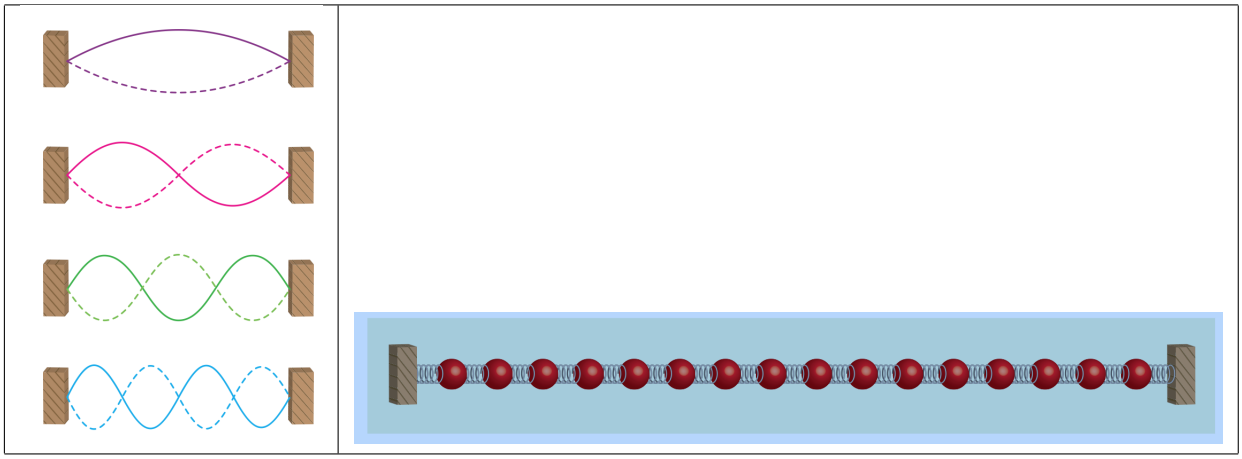


Figura 11: (sinistra) Modi vibrazionali in una corda tesa. (destra) catena di punti materiali come nel problema di Fermi Pasta Ulam

Parte IV

Il problema di Fermi Pasta Ulam

La formulazione teorica della termodinamica viene spesso espresso tramite i postulati di Gibbs:

Postulato 0 esistenza degli stati microscopici: un sistema macroscopico è formato da un insieme di costituenti elementari che possono in principio essere descritti con le leggi fondamentali della fisica.

Postulato 1 equilibrio: all'equilibrio in un sistema chiuso tutte le configurazioni possibili (ossia con stessa energia E) sono equiprobabili

Postulato 2 evoluzione fuori dall'equilibrio: fuori dall'equilibrio un sistema tende ad evolvere verso l'equilibrio.

Dai postulati 0 e 1 si trova facilmente la legge dei gas perfetti nel caso di un sistema di particelle non interagenti. E' interessante notare che la situazione di equilibrio del postulato 1 non può essere raggiunta se non c'è alcuna interazione tra le particelle.

Nel 1955 Fermi, Pasta e Ulam ebbero l'idea di usare uno dei primi computer a valvole (il Maniac I) per provare a verificare il raggiungimento della situazione di equilibrio, anche chiamata *equipartizione dell'energia*, nel caso di un sistema semplificato. La loro idea era di analizzare il moto di una corda elastica tesa posta in oscillazione. Per una tale corda fissata agli estremi nel caso di termini di forza solamente elastici l'equazioni del moto ammette modi normali di oscillazione come raffigurato in Fig. 11. Se ci limitiamo ad una corda ideale elastica e se al tempo t_0 il moto è descritto solo da un certo modo (ad esempio quello fondamentale) , allora tutti gli altri modi non contribuiranno mai al moto in istanti di tempo successivi. Viceversa se termini di forza non lineari (non-elastici) sono presenti allora ci si aspetta, dal secondo postulato, che dopo un certo tempo tutti i modi siano equiprobabili.

Per semplicità Fermi Pasta Ulam considerarono una catena unidimensionale di N punti materiali di massa m connessi tra di loro con delle molle ideali di costante elastica k. I punti 1 e N sono connessi con molle dello stesso tipo a due punti fissi. Indichiamo con x_i lo *spostamento* del punto materiale i rispetto alla sua posizione di equilibrio. Nel caso non ci siano termini non lineari le equazioni del moto sono date da:

$$\begin{cases} m\frac{d^2x_i(t)}{dt^2} = k\left(x_{i+1}-x_i\right)+k\left(x_{i-1}-x_i\right) = k\left(x_{i+1}-2x_i+x_{i-1}\right) \\ x_0 = 0 \\ x_{N+1} = 0 \end{cases} \tag{174}$$

cerchiamo ora soluzioni armoniche del tipo:

$$x_i\left(t\right) = Ae^{i\left(\tilde{k}i-\omega t\right)} \tag{175}$$

dove assumiamo che la componente immaginaria abbia valore fisico. Sostituiamo e troviamo

$$
\begin{aligned}
-m\omega^2 A e^{i\left(\tilde{k}i-\omega t\right)} &= kA\left(e^{i\left(\tilde{k}i-\omega t+\tilde{k}\right)}-2e^{i\left(\tilde{k}i-\omega t\right)}+e^{i\left(\tilde{k}i-\omega t-\tilde{k}\right)}\right) \\
-m\omega^2 &= k\left(e^{i\tilde{k}}-2+e^{-i\tilde{k}}\right) \\
-m\omega^2 = 2k\left(\cos\left(\tilde{k}\right)-1\right) &= 2k\left(\cos^2\left(\frac{\tilde{k}}{2}\right)-\sin^2\left(\frac{\tilde{k}}{2}\right)-1\right) \\
\omega &= \sqrt{\frac{4k}{m}}\sin\left(\frac{\tilde{k}}{2}\right)
\end{aligned}
\tag{176}
$$

i numeri d'onda $\tilde{k}$ devono essere scelti in maniera da far sì che le condizioni agli estremi (o al contorno) siano soddisfatte: costruiamo un'onda stazionaria relativa al modo n:

$$
x_{i,n}(t) = \frac{A_n}{2}\left[\left(e^{i\left(\tilde{k}i-\omega t\right)}-e^{i\left(-\tilde{k}i-\omega t\right)}\right]\right. = A_n\sin\left(\tilde{k}i\right)e^{-i\omega t} \tag{177}
$$

che possiamo scrivere come

$$
x_{i,n}(t) = A_n\sin\left(\frac{\pi n}{N+1}i\right)\sin\left(-\omega t\right) \tag{178}
$$

allora per soddisfare le condizioni al contorno secegliamo:

$$
\tilde{k} = \frac{\pi n}{N+1}\ n = 1,\ldots,N \tag{179}
$$

dove la frequenza del modo è data da:

$$
\omega_n = \sqrt{\frac{4k}{m}}\sin\left(\frac{\pi n}{2\left(N+1\right)}\right) \tag{180}
$$

il modo di oscillazione con $n = 1$ viene chiamato modo fondamentale.

Fermi Pasta Ulam aggiunsero dei termini non lineari alle forze cosicché le equazioni del moto diventarono, nel caso di aggiunta di termini quadratici :

$$
\begin{cases}
m\frac{d^2x_i(t)}{dt^2} = k\left(x_{i+1}-2x_i+x_{i-1}\right) & +\alpha\left[\left(x_{i+1}-x_i\right)^2-\left(x_i-x_{i-1}\right)^2\right] \\
x_0 = 0 & \\
x_{N+1} = 0 &
\end{cases}
\tag{181}
$$

o nel caso di aggiunta di termini cubici:

$$
\begin{cases}
m\frac{d^2x_i(t)}{dt^2} = k\left(x_{i+1}-2x_i+x_{i-1}\right) & +\beta\left[\left(x_{i+1}-x_i\right)^3-\left(x_i-x_{i-1}\right)^3\right] \\
x_0 = 0 & \\
x_{N+1} = 0 &
\end{cases}
\tag{182}
$$

Se al tempo t gli spostamenti sono dati da $x_i(t)$, il contributo $C_n(t)$ del modo n è dato da:

$$
C_n(t) = \sum_{i=1,N} x_i(t)\frac{1}{\sqrt{N}}\sin\left(\frac{\pi n}{N+1}i\right) \tag{183}
$$

dove abbiamo usato:

$$
\sum_{i=1,N}\sin^2\left(\frac{\pi n}{N+1}i\right) = N \tag{184}
$$

questo ci permette di scrivere:

$$
x_i(t) = \sum_{n=1,N} C_n(t)\frac{1}{\sqrt{N}}\sin\left(\frac{\pi n}{N+1}i\right) \tag{185}
$$

Fermi Pasta Ulam integrarono le equazioni del moto imponendo che il solo modo fondamentale contribuisca al tempo iniziale e si aspettavano di trovare che tutti i modi diventassero attivi dopo un certo periodo di equilibrazione. Invece trovarono che solo taluni modi contribuivano al moto e apparentemente dopo un certo tempo il moto tornava ad essere descritto principalmente dal solo modo iniziale. Per analizzare la partizione dell'energia, andarono a considerare il contributo energetico di ciascun modo tenendo conto solo del termini cinetici ed elastici. Al tempo t l'energia del modo n è data da :

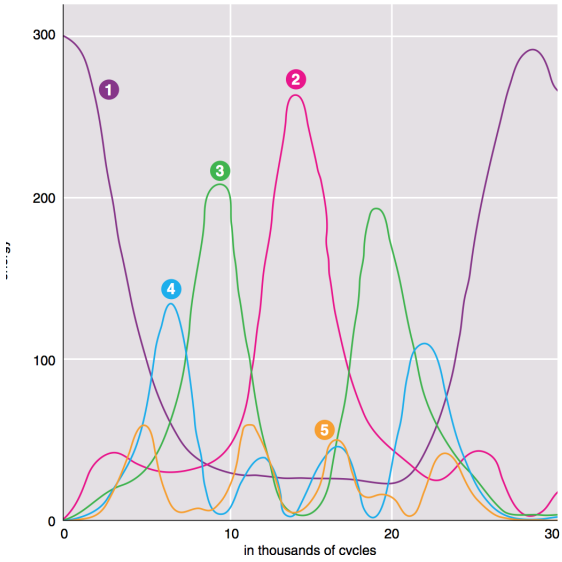


Figura 12: Evoluzione temporale della distribuzione di energia per il problema di Fermi Pasta Ulam. Sistema di 31 punti materiali, $m = 1$, $k = 1$,$\alpha = \frac{1}{4}$.

$$
\begin{aligned}
E_n(t) &= \frac{1}{2}m\sum_{i=1,N}\left(\frac{dC_n(t)}{dt}\right)^2\frac{1}{N}\sin^2\left(\frac{\pi n}{N+1}i\right)+\sum_{i=0,N}\frac{1}{2}k\frac{1}{N}\left(C_n(t)\sin\left(\frac{\pi n}{N+1}(i+1)\right)-C_n(t)\sin\left(\frac{\pi n}{N+1}i\right)\right)^2\\
E_n(t) &= \frac{1}{2}m\left(\frac{dC_n(t)}{dt}\right)^2+\frac{1}{2}k\left(C_n(t)\right)^2\frac{1}{N}\times\\
&\quad\left[\left(\sum_{i=0,N}\sin^2\left(\frac{\pi n}{N+1}(i+1)\right)\right)+\left(\sum_{i=0,N}\sin^2\left(\frac{\pi n}{N+1}i\right)\right)-2\left(\sum_{i=0,N}\sin\left(\frac{\pi n}{N+1}(i+1)\right)\sin\left(\frac{\pi n}{N+1}i\right)\right)\right]\\
E_n(t) &= \frac{1}{2}m\left(\frac{dC_n(t)}{dt}\right)^2+\frac{1}{2}k\left(C_n(t)\right)^2\times\\
&\quad\left[2-2\frac{1}{N}\left(\sum_{i=0,N}\sin^2\left(\frac{\pi n}{N+1}i\right)\cos\left(\frac{\pi n}{N+1}\right)+\sum_{i=0,N}\cos\left(\frac{\pi n}{N+1}i\right)\sin\left(\frac{\pi n}{N+1}i\right)\sin\left(\frac{\pi n}{N+1}\right)\right)\right]\\
E_n(t) &= \frac{1}{2}m\left(\frac{dC_n(t)}{dt}\right)^2+k\left(C_n(t)\right)^2\left(1-\cos\left(\frac{\pi n}{N+1}\right)\right)\\
E_n(t) &= \frac{1}{2}m\left(\frac{dC_n(t)}{dt}\right)^2+2k\left(C_n(t)\right)^2\sin^2\left(\frac{\pi n}{2(N+1)}\right)
\end{aligned}
\tag{186}
$$

dove abbiamo usato la proprietà:

$$
\sum_{i=0,N+1}\cos\left(\frac{\pi n}{N+1}i\right)\sin\left(\frac{\pi n}{N+1}i\right)=0 \tag{187}
$$

In Fig. 7 si vede che invece di equipartirsi, l'energia è data solo da alcuni modi. Inoltre si vede che apparentemente dopo un certo tempo il sistema ritorna (quasi) nelle condizioni iniziali. Tale comportamento continua in maniera ciclica per tempi successivi. Cosicché apparentemente sembra non si arrivi mai all'equipartizione. Studi successivi indicano che l'equilibrio viene raggiungo solo per tempo ancora maggiori. Il tempo per giungere all'equipartizione è più grande se il numero di punti materiali N è piccolo e se l' energia totale iniziale è piccola.

Parte V
Equazioni differenziali ordinarie con condizioni al contorno

17 Equazioni che studieremo

Vogliamo introdurre dei metodi che ci permettano, ad esempio, di trovare le soluzioni dell'equazione di Schrödinger stazionaria in una sola dimensione:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + V(x)\psi(x) = E\psi(x) \tag{188}$$

Per ottenere un metodo più generale consideriamo equazioni differenziali del secondo ordine del tipo:

$$\begin{aligned} a(x)\frac{d^2}{dx^2}y(x) + b(x)\frac{d}{dx}y(x) + c(x)y(x) &= f(x) \\ \begin{cases} \alpha_0 y(a) + \alpha_1 \frac{d}{dx}y(a) & = \lambda_1 \\ \beta_0 y(b) + \beta_1 \frac{d}{dx}y(b) & = \lambda_2 \end{cases} & \end{aligned} \tag{189}$$

dove la funzione incognita $y(x)$ è definita nell'intervallo $[a, b]$ e le condizioni al contorno sono disaccoppiate. Le condizioni al contorno vengono poi chiamate **di Dirichlet** se hanno la forma:

$$\begin{aligned} y(a) &= \alpha \\ y(b) &= \beta \end{aligned} \tag{190}$$

sono invece chiamate di **Neumann** se hanno la forma:

$$\begin{aligned} \frac{d}{dx}y(a) &= \alpha \\ \frac{d}{dx}y(b) &= \beta \end{aligned} \tag{191}$$

18 Metodo delle differenze finite:

Vogliamo trasformare il problema della soluzione dell'equazione differenziale in un problema matriciale. Cominciamo col definire sull'intervallo $[a, b]$ dell'asse x una griglia di N punti equispaziati, chiamiamo h il parametro di spaziatura e usiamo la notazione $y_i = y(x_i)$ e $y_i^{(n)} = \frac{d^n}{dx^n}y(x_i)$. Notiamo che con tale scelta risulta:

$$\begin{aligned} y_1 &= y(a) \\ y_N &= y(b) \\ h &= \frac{b-a}{N-1} \end{aligned} \tag{192}$$

Possiamo quindi usare le formule di Eq. 24. Per $i = 2, \ldots, N-1$ abbiamo:

$$\begin{aligned} y_i^{(2)} &\approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \\ y_i^{(1)} &\approx \frac{y_{i+1} - y_{i-1}}{2h} \end{aligned} \tag{193}$$

allora per ogni $i = 2, \ldots, N-1$ otteniamo un'equazione lineare:

$$a_i\left(\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}\right) + b_i\left(\frac{y_{i+1} - y_{i-1}}{2h}\right) + c_i y_i = f_i \tag{194}$$

che scriviamo come:

$$\left(\frac{a_i}{h^2} + \frac{b_i}{2h}\right)y_{i+1} + \left(-\frac{2a_i}{h^2} + c_i\right)y_i + \left(\frac{a_i}{h^2} - \frac{b_i}{2h}\right)y_{i-1} = f_i \tag{195}$$

Ora dobbiamo occuparci dei punti $i = 1$ e $i = N$. Introduciamo i punti x_0 e x_{N+1} e abbiamo le equazioni per le condizioni al contorno:

$$\alpha_0 y_1 + \alpha_1 \left(\frac{y_2 - y_0}{2h} \right) = \lambda_1$$
$$\beta_0 y_N + \beta_1 \left(\frac{y_{N+1} - y_{N-1}}{2h} \right) = \lambda_2 \tag{196}$$

nel caso in cui $\alpha_1 \neq 0$ risolvendo troviamo:

$$y_0 = \frac{\alpha_0 y_1 + \frac{\alpha_1 y_2}{2h} - \lambda_1}{\frac{\alpha_1}{2h}} = y_2 + 2h \frac{\alpha_0 y_1 - \lambda_1}{\alpha_1} \tag{197}$$

che possiamo utilizzare in Eq: 195:

$$\left(\frac{a_1}{h^2} + \frac{b_1}{2h} \right) y_2 + \left(-\frac{2a_1}{h^2} + c_i \right) y_1 + \left(\frac{a_1}{h^2} - \frac{b_1}{2h} \right) \left(y_2 + 2h \frac{\alpha_0 y_1 - \lambda_1}{\alpha_1} \right) = f_1 \tag{198}$$

che scriviamo come:

$$2\frac{a_1}{h^2} y_2 + \left(-\frac{2a_1}{h^2} + c_1 + \frac{\alpha_0}{\alpha_1} \left(\frac{2a_1}{h} - b_1 \right) \right) y_1 = f_1 + \frac{\lambda_1}{\alpha_1} \left(\frac{2a_1}{h} - b_1 \right) \tag{199}$$

mentre nel caso $\alpha_1 = 0$ possiamo porre direttamente:

$$y_1 = \frac{\lambda_1}{\alpha_0} \tag{200}$$

Per $i = N$ e $\beta_1 \neq 0$ otteniamo:

$$y_{N+1} = y_{N-1} + 2h \frac{(-\beta_0 y_N + \lambda_2)}{\beta_1} \tag{201}$$

che ci permette di scrivere:

$$\left(\frac{a_N}{h^2} + \frac{b_N}{2h} \right) \left(y_{N-1} + 2h \frac{(-\beta_0 y_N + \lambda_2)}{\beta_1} \right) + \left(-\frac{2a_N}{h^2} + c_N \right) y_N + \left(\frac{a_N}{h^2} - \frac{b_N}{2h} \right) y_{N-1} = f_N \tag{202}$$

che scriviamo come:

$$\left(-\frac{2a_N}{h^2} + c_N - \frac{\beta_0}{\beta_1} \left(\frac{2a_N}{h} + b_N \right) \right) + 2\frac{a_N}{h^2} y_{N-1} = f_N - \frac{\lambda_2}{\beta_1} \left(\frac{2a_N}{h} + b_N \right) \tag{203}$$

mentre nel caso $\beta_1 = 0$ possiamo porre direttamente:

$$y_N = \frac{\lambda_2}{\beta_0} \tag{204}$$

E' utile *pensare* al problema come a un prodotto di matrici, definiamo il vettore $\mathbf{Y}$:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_{N-1} \\ y_N \end{pmatrix} \tag{205}$$

ed il vettore $\mathbf{F}$, nel caso $\alpha_1, \beta_1 \neq 0$:

$$\mathbf{F} = \begin{pmatrix} f_1 + \frac{\lambda_1}{\alpha_1} \left(\frac{2a_1}{h} - b_1 \right) \\ f_2 \\ \dots \\ f_{N-1} \\ f_N - \frac{\lambda_2}{\beta_1} \left(\frac{2a_N}{h} + b_N \right) \end{pmatrix} \tag{206}$$

mentre nel caso $\alpha_1 = \beta = 0$ definiamo:

$$\mathbf{F} = \begin{pmatrix} \frac{\lambda_1}{\alpha_0} \\ f_2 \\ \dots \\ f_{N-1} \\ \frac{\lambda_2}{\beta_0} \end{pmatrix} \tag{207}$$

allora il nostro problema può essere scritto come il problema di trovare il vettore $\mathbf{Y}$, tale che:

$$\mathbf{M} \cdot \mathbf{Y} = \mathbf{F} \tag{208}$$

dove la matrice $\mathbf{M}$ è definita, nel caso $\alpha_1, \beta_1 \neq 0$, come:

$$\mathbf{M} = \begin{pmatrix} \left(-\frac{2a_1}{h^2} + c_1 + \frac{\alpha_0}{\alpha_1}\left(\frac{2a_1}{h} - b_1\right)\right) & 2\frac{a_1}{h^2} & & & & & \\ L_2 & D_2 & U_2 & & & & \\ & \cdots & \cdots & \cdots & & & \\ & & L_i & D_i & U_i & & \\ & & & \cdots & \cdots & \cdots & \\ & & & & L_{N-1} & D_{N-1} & U_{N-1} \\ & & & & & 2\frac{a_N}{h^2} & \left(-\frac{2a_N}{h^2} + c_N - \frac{\beta_0}{\beta_1}\left(\frac{2a_N}{h} + b_N\right)\right) \end{pmatrix} \tag{209}$$

mentre nel caso $\alpha_1, = \beta = 0$ come:

$$\mathbf{M} = \begin{pmatrix} 1 & & & & & & \\ L_2 & D_2 & U_2 & & & & \\ & \cdots & \cdots & \cdots & & & \\ & & L_i & D_i & U_i & & \\ & & & \cdots & \cdots & \cdots & \\ & & & & L_{N-1} & D_{N-1} & U_{N-1} \\ & & & & & & 1 \end{pmatrix} \tag{210}$$

con:

$$L_i = \left(\frac{a_i}{h^2} - \frac{b_i}{2h}\right) \tag{211}$$

$$D_i = \left(-\frac{2a_i}{h^2} + c_i\right) \tag{212}$$

e

$$U_i = \left(\frac{a_i}{h^2} + \frac{b_i}{2h}\right) \tag{213}$$

Quindi la soluzione numerica del nostro problema è data formalmente da

$$\mathbf{Y} = \mathbf{M}^{-1} \cdot \mathbf{F} \tag{214}$$

dove la matrice $\mathbf{M}$ è una matrice tridiagonale. Per risolvere il problema di Eq. 208 esistono vari metodi di algebra lineare che sfruttano la tridiagonalità di $\mathbf{M}$. Si veda la scomposizione in forma LU nella parte II della dispensa. Inoltre, diversi metodi sono implementati all'interno della libreria $LAPACK$ disponibile in *Fortran* e *C++*. In *python* possiamo usare la libraria *numpy.linalg* oppure la libreria *scipy*.

18.1 Condizioni al contorno periodiche

Consideriamo il problema che si presenta quando le condizioni al contorno sono del tipo:

$$y\left(a\right) = y\left(b\right) \tag{215}$$

con:

$$f\left(a\right) = f\left(b\right) \tag{216}$$

Ponendo $L = b - a$ posso pensare a $y(x)$ definita su tutto l'asse reale e con i vincoli:

$$\begin{aligned} y(x+L) &= y(x) \\ a(x+L) &= a(x) \\ b(x+L) &= b(x) \\ c(x+L) &= c(x) \\ f(x+L) &= f(x) \end{aligned} \tag{217}$$

che scriverò come

$$\begin{aligned} y_N &= y_1 \\ a_N &= a_1 \\ b_N &= b_1 \\ c_N &= c_1 \\ f_N &= f_1 \end{aligned} \tag{218}$$

e che posso anche pensare come:

$$\begin{aligned} y_{i+N-1} &= y_i \\ a_{i+N-1} &= a_i \\ b_{i+N-1} &= b_i \\ c_{i+N-1} &= c_i \\ f_{i+N-1} &= f_i \end{aligned} \tag{219}$$

allora definisco:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_{N-1} \end{pmatrix} \tag{220}$$

e

$$\mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ \cdots \\ f_{N-1} \end{pmatrix} \tag{221}$$

l'equazione da risolvere $\mathbf{M} \cdot \mathbf{Y} = \mathbf{F}$ è definita dalla matrice:

$$\mathbf{M} = \begin{pmatrix} D_1 & U_1 & & & & L_1 \\ L_2 & D_2 & U_2 & & & \\ & \cdots & \cdots & \cdots & & \\ & & L_i & D_i & U_i & \\ & & & \cdots & \cdots & \cdots \\ U_{N-1} & & & & L_{N-1} & D_{N-1} \end{pmatrix} \tag{222}$$

in questo caso possiamo utilizzare le routine LAPACK per le matrici a bande.

19 Il problema agli autovalori

Nel caso della soluzioni dell'equazione di Schödinger stazionaria e di problemi analoghi dal punto di vista formale vogliamo trovare per $x \in [a, b]$ gli (auto-)valori E e le (auto-)funzioni $y(x)$ tali che:

$$a(x) \frac{d^2}{dx^2} y(x) + b(x) \frac{d}{dx} y(x) + c(x) y(x) = E y(x) \tag{223}$$

consideriamo condizioni al contorno del tipo:

$$\begin{cases} y(a) &= 0 \\ y(b) &= 0 \end{cases} \tag{224}$$

possiamo trasformare il problema in un problema matriciale come visto prima, chiamiamo:

$$\mathbf{Y} = \begin{pmatrix} y_2 \\ \cdots \\ \cdots \\ y_{N-1} \end{pmatrix} \tag{225}$$

e poniamo

$$\mathbf{M} = \begin{pmatrix} D_2 & U_2 & & & \\ \cdots & \cdots & \cdots & & \\ & L_i & D_i & U_i & \\ & & \cdots & \cdots & \cdots \\ & & & L_{N-1} & D_{N-1} \end{pmatrix} \tag{226}$$

con:

$$L_i = \left(\frac{a_i}{h^2} - \frac{b_i}{2h} \right) \tag{227}$$

$$D_i = \left(-\frac{2a_i}{h^2} + c_i \right) \tag{228}$$

e

$$U_i = \left(\frac{a_i}{h^2} + \frac{b_i}{2h} \right) \tag{229}$$

quindi il problema agli autostati-autovalori che dobbiamo risolvere è:

$$\mathbf{M} \cdot \mathbf{Y} = E\mathbf{Y} \tag{230}$$

questo problema è facilmente risolvibile con i metodi di algebra lineari per le matrici tridiagonali. Oltretutto non è necessario salvare in memoria tutta la matrice ma soltanto la diagonale e la diagonale superiore ed inferiore.

19.1 Il metodo Shooting e quello di Numerov

Consideriamo il problema agli autovalori:

$$a(x) \frac{d^2}{dx^2} y(x) + b(X) \frac{d}{dx} y(x) + c(x) y(x) = E y(x) \tag{231}$$

con le condizioni al contorno:

$$\begin{cases} y(a) &= 0 \\ y(b) &= 0 \end{cases} \tag{232}$$

il metodo di shooting per trovare le coppie autostati-autovalori consiste nello scegliere un certo valore per $y^{(1)}(a)$ e poi risolvere un'equazione differenziale ordinaria alle condizioni iniziali. Usiamo il formalismo di Sezione 8:

$$\mathbf{y}(x) = \begin{pmatrix} y(x) \\ \frac{d}{dx} y(x) \end{pmatrix} \tag{233}$$

e

$$\mathbf{f}(x) = \begin{pmatrix} \frac{d}{dx} y(x) \\ -\frac{b(x)}{a(x)} \frac{d}{dx} y(x) + \frac{(E - c(x))}{a(x)} y(x) \end{pmatrix} \tag{234}$$

questo ci permette dato un valore per E di di propagare $y(x)$ da a a b. L' E usato nella propagazione sarà un buon autovalore solo se la condizione $y(b) = 0$ è soddisfatta. In tal caso E sarà un autovalore e $y(x)$ sarà la corrispondente auto-funzione. E' facile mostrare che la scelta del $y^{(1)}(a)$ non incide sugli E trovati ma solo sulla norma delle autofunzioni $y(x)$.

Nel caso particolare di un'equazione analoga a quella di Schrödiger:

$$\frac{d^2}{dx^2} y(x) + c(x) y(x) = E y(x) \tag{235}$$

che posso scrivere come:

$$y_i^{(2)} = (E - c_i) y_i \tag{236}$$

Il metodo di Numerov permette di propagare $y(x)$ in maniera veloce e precisa. Si parte dallo sviluppo in serie di Taylor

$$\begin{cases} y_{i+1} &= y_i + y_i^{(1)} h + \frac{1}{2} y_i^{(2)} h^2 + \frac{1}{6} y_i^{(3)} h^3 + \frac{1}{24} y_i^{(4)} h^4 + \frac{1}{120} y_i^{(5)} h^5 + O\left(h^6\right) \\ y_{i-1} &= y_i - y_i^{(1)} h + \frac{1}{2} y_i^{(2)} h^2 - \frac{1}{6} y_i^{(3)} h^3 + \frac{1}{24} y_i^{(4)} h^4 - \frac{1}{120} y_i^{(5)} h^5 + O\left(h^6\right) \end{cases} \tag{237}$$

ora sommo le due equazione membro a membro:

$$y_{i+1} + y_{i-1} = 2y_i + y_i^{(2)} h^2 + \frac{1}{12} y_i^{(4)} h^4 + O\left(h^6\right) \tag{238}$$

da cui ricavo:

$$y_i^{(2)} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{1}{12} y_i^{(4)} h^2 + O\left(h^4\right) \tag{239}$$

ora uso Eq.236 e trovo:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{1}{12} y_i^{(4)} h^2 + O\left(h^4\right) = \left(E - c_i\right) y_i \tag{240}$$

ora scrivo $y_i^{(4)}$ come:

$$y_i^{(4)} = \frac{y_{i+1}^{(2)} - 2y_i^{(2)} + y_{i-1}^{(2)}}{h^2} = \frac{\left(E - c_{i+1}\right) y_{i+1} - 2\left(E - c_i\right) y_i + \left(E - c_{i-1}\right) y_{i-1}}{h^2} + O\left(h^2\right) \tag{241}$$

e inserisco nella precedente equazione:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{\left(E - c_{i+1}\right) y_{i+1} - 2\left(E - c_i\right) y_i + \left(E - c_{i-1}\right) y_{i-1}}{12} + O\left(h^4\right) = \left(E - c_i\right) y_i \tag{242}$$

ora risolvo per y_{i+1} e trovo il propagatore cercato:

$$y_{i+1} = \frac{1}{12 - \left(E - c_{i+1}\right) h^2} \left[\left(10h^2 \left(E - c_i\right) + 24\right) y_i + \left(\left(E - c_{i-1}\right) h^2 - 12\right) y_{i-1}\right] + \left(h^4\right) \tag{243}$$

Parte VI
Equazioni differenziali alle derivate parziali

Questa parte del corso è dedicata ad alcune delle equazioni differenziali alle derivate parziali di solo tipo lineare che rivestono particolare importanza nello studio della fisica.

20 Equazione di Poisson

Uno dei casi più frequenti in cui siamo richiesti di risolvere un'equazione di Poisson è quello del calcolo di un campo elettrico data una distribuzione di carica $\rho(\mathbf{r})$. Le equazioni di Maxwell porgono:

$$\nabla \cdot \mathbf{E}(\mathbf{r}) = \frac{1}{\epsilon_0}\rho(\mathbf{r}) \tag{244}$$

invece di risolvere direttamente questo problema andiamo a trovare il potenziale elettrostatico $\phi(\mathbf{r})$ tale che $\mathbf{E}(\mathbf{r}) = -\nabla\phi(\mathbf{r})$, tramite la soluzione dell'equazione di Poisson:

$$\Delta\phi(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0} \tag{245}$$

dove per il potenziale dobbiamo considerare delle condizioni al contorno.

Per avere un formalismo più generale consideriamo la soluzione del problema:

$$\Delta\phi(\mathbf{r}) = f(\mathbf{r}) \tag{246}$$

con $\mathbf{r}$ all'interno di un volume $\Omega = [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$. Le condizioni al contorno sono del tipo $\phi(\mathbf{r}) = g(\mathbf{r})$ per $\mathbf{r}$ sul bordo del volume Ω che indichiamo con $\partial\Omega$. Scriviamo ora il problema in termini della coordinate di $\mathbf{r}$, (x, y, z):

$$\begin{cases} \frac{\partial^2}{\partial x^2}\phi(x,y,z) + \frac{\partial^2}{\partial y^2}\phi(x,y,z) + \frac{\partial^2}{\partial z^2}\phi(x,y,z) = f(x,y,z) & (x,y,z) \in \Omega \\ \phi(x,y,z) = g(x,y,z) & (x,y,z) \in \partial\Omega \end{cases} \tag{247}$$

Ora discretizziamo i tre intervalli $[a_x, b_x]$,$[a_y, b_y]$,$[a_z, b_z]$, con un numero di punti N_x,N_y,N_z e distanza tra punti h_x, h_y, h_z, rispettivamente.

$$\begin{cases} x_1 & = a_x \\ x_{N_x} & = b_x \\ h_x = \frac{b_x - a_x}{N_x - 1} & \end{cases}$$
$$\begin{cases} y_1 & = a_y \\ y_{N_x} & = b_y \\ h_y = \frac{b_y - a_y}{N_y - 1} & \end{cases} \tag{248}$$
$$\begin{cases} z_1 & = a_z \\ z_{N_x} & = b_z \\ h_z = \frac{b_z - a_z}{N_z - 1} & \end{cases}$$

Indichiamo con $\phi_{ijk} = \phi(x_i, y_i, z_k)$ Abbiamo che le condizioni al contorno vengono espresse da:

$$\begin{cases} \phi_{1ij} & = g_{1ij} \\ \phi_{N_x ij} & = g_{N_x ij} \\ \phi_{i1j} & = g_{i1j} \\ \phi_{iN_y j} & = g_{iN_y j} \\ \phi_{ij1} & = g_{ij1} \\ \phi_{ijN_z} & = g_{ijN_z} \end{cases} \quad \forall i, j \tag{249}$$

l' equazione di Poisson viene allora scritta usando le note formule per le derivate:

$$\frac{\phi_{i+1jk} - 2\phi_{ijk} + \phi_{i-1jk}}{h_x^2} + \frac{\phi_{ij+1k} - 2\phi_{ijk} + \phi_{ij-1k}}{h_y^2} + \frac{\phi_{ijk+1} - 2\phi_{ijk} + \phi_{ijk-1}}{h_z^2} = f_{ijk} \tag{250}$$

Possiamo facilmente trasformare questo problema in un problema matriciale. Per prima cosa introduciamo un nuovo indice unico $l \in [0, N_x N_y N_z - 1]$ per individuare i punti (i, j, k):

$$(i, j, k) \to l = (k-1)\, N_x N_y + (j-1)\, N_x + (i-1) \tag{251}$$

la trasformazione inversa è data da:

$$l \to (i, j, k) = \begin{cases} k & = \text{floor}\left(l/\left(N_x N_y\right)\right) + 1 \\ j & = \text{floor}\left(\left(l - (k-1) N_x N_y\right)/N_x\right) + 1 \\ i & = l - (k-1) N_x N_y - (j-1) N_x + 1 \end{cases} \tag{252}$$

Dove con $\text{floor}\,(x)$ con $x > 0$ intendiamo il numero intero minore o uguale a x più grande possibile. Quindi possiamo scrivere la funzione cercata nei punti del nostro reticolo come ϕ_l e analogamente scriveremo f_l. Introduciamo la matrice M, che definiamo in questo modo, sia $l \longleftrightarrow (i, j, k)$ e $m \longleftrightarrow (i', j', k')$ se l non appartiene al bordo poniamo:

$$M_{lm} = \begin{cases} \frac{1}{h_x^2} & (i', j', k') = (i+1, j, k) \\ -2\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right) & (i', j', k') = (i, j, k) \\ \frac{1}{h_x^2} & (i', j', k') = (i-1, j, k) \\ \frac{1}{h_y^2} & (i', j', k') = (i, j+1, k) \\ \frac{1}{h_y^2} & (i', j', k') = (i, j-1, k) \\ \frac{1}{h_z^2} & (i', j', k') = (i, j, k+1) \\ \frac{1}{h_z^2} & (i', j', k') = (i, j, k-1) \\ 0 & \text{altrimenti} \end{cases} \tag{253}$$

inoltre se l individua un punto sul bordo poniamo

$$\begin{cases} M_{ll} = 1 \\ M_{lm} = 0 \qquad m \neq l \end{cases} \tag{254}$$

In questa maniera il mio problema viene scritto come:

$$\sum_m M_{lm} \phi_m = (f_l + b_l) \tag{255}$$

dove il vettore $\mathbf{b}$ è definito da:

$$\begin{cases} b_l = g_l \qquad l \in \partial\Omega \\ b_l = 0 \qquad l \notin \partial\Omega \end{cases} \tag{256}$$

per risolverlo una possibilità è quella di invertire la matrice M:

$$\phi_l = \sum_m M_{lm}^{-1} \left(f_l + b_l\right) \tag{257}$$

Per invertire la matrice posso usare le routines di algebra lineare come quelle della libreria LAPACK. Tale metodo però potrebbe non essere fattibile nel caso che la dimensione di M ossia $N_x N_y N_z$ sia grande. Ad esempio se $N_x = N_y = N_z = 100$ per allocare in memoria la matrice in doppia precisione ho bisogno di $100^6 * 8$ bytes $= 7.45$ TB.

Per ovviare a questo problema vengono usati **algoritmi iterativi**. Questi sono basati sul fatto che anche se immagazzinare M in memoria è non possibile è (ancora) possibile calcolare $\sum_m M_{lm}\phi_m$.

20.1 Metodo di Jacobi e di Gauss-Seidel

Un metodo largamente usato è quello di Gauss-Seidel. Partiamo dal problema:

$$M \cdot \phi = \mathbf{f} \tag{258}$$

in cui ϕ è l'incognita e introduciamo la matrice A definita come:

$$\begin{cases} A_{ll} = \frac{1}{M_{ll}} \\ A_{lm} = 0 \qquad m \neq l \end{cases} \tag{259}$$

allora otteniamo:

$$AM \cdot \phi = A \cdot \mathbf{f} \tag{260}$$

dove:

$$(AM)_{lm} = \begin{cases} 1 & l = m \\ \frac{M_{lm}}{M_{ll}} & l \neq m \end{cases} \tag{261}$$

introduciamo la matrice $\tilde{M}$:

$$\tilde{M}_{lm} = \begin{cases} 0 & l = m \\ -\frac{M_{lm}}{M_{ll}} & l \neq m \end{cases} \tag{262}$$

ed il vettore

$$\tilde{f}_l = \frac{f_l}{M_{ll}} \tag{263}$$

Il nostro problema allora diventa:

$$\phi = \tilde{M}\phi + \tilde{\mathbf{f}} \tag{264}$$

possiamo allora provare a risolvere il problema iterando. Parto da:

$$\phi^{(0)} = \tilde{\mathbf{f}} \tag{265}$$

poi pongo:

$$\phi^{(1)} = \tilde{M}\phi^{(0)} + \tilde{\mathbf{f}} \tag{266}$$

e così via per le iterazioni successive:

$$\phi^{(i+1)} = \tilde{M}\phi^{(i)} + \tilde{\mathbf{f}} \tag{267}$$

se ad una certa iterazione viene raggiunta la *convergenza* ossia $\phi^{(i+1)} = \phi^{(i)}$ (in pratica considererò un valore soglia per il modulo del vettore differenza $\phi^{(i+1)} - \phi^{(i)}$) allora tale vettore è proprio la soluzione che cerchiamo. Formalmente possiamo anche scrivere:

$$\phi = \frac{1}{\mathbb{I} - \tilde{M}}\tilde{\mathbf{f}} = \left(\mathbb{I} + \tilde{M} + \tilde{M}^2 + \tilde{M}^3 + \ldots\right)\tilde{\mathbf{f}} = \tilde{\mathbf{f}} + M\tilde{\mathbf{f}} + M^2\tilde{\mathbf{f}} + M^3\tilde{\mathbf{f}} + M^4\tilde{\mathbf{f}} + \ldots \tag{268}$$

quindi ritrovando i termini che calcoliamo con il procedimento iterativo. Questo algoritmo prende il nome di metodo di Jacobi.

Si può renderlo più veloce aggiornando un elemento di matrice dopo l'altro ossia rimpiazzando Eq. 267:

$$\phi_j^{(i+1)} = \sum_{k<j} \tilde{M}_{jk}\phi_k^{(i+1)} + \sum_{l \geq j} \tilde{M}_{jl}\phi_l^{(i)}\tilde{\mathbf{f}}_l \tag{269}$$

Tale variante prende il nome di algoritmo di Gauss-Seidl.

20.2 Il Metodo di Richardson

Un altro metodo iterativo frequentemente usato è quello di Richardson, come nel caso precedente vogliamo risovere:

$$M \cdot \phi = \mathbf{f} \tag{270}$$

Iniziamo l'algoritmo ciclico scegliendo un coefficiente $\alpha > 0$ e scegliamo $\phi^{(0)}$ in maniera casuale. Poi iteriamo con la regola:

$$\phi^{(i+1)} = \phi^{(i)} + \alpha\left(\mathbf{f} - M \cdot \phi^{(i)}\right) \tag{271}$$

Si vede subito che nel caso sa giunga a convergenza ossia se vale $\phi^{(i+1)} = \phi^{(i)}$ allora anche l'equazione 270 è soddisfatta.

20.3 Relazione con problema di minimizzazione

Risolvere Eq. 270 è equivalente a trovare il minimo del funzionale che scriviamo usando la notazione di Dirac per i vettori $|\phi)$:

$$E\left(|\phi\rangle\right) = \langle\phi| M |\phi\rangle - \left(\langle f| \phi\rangle + \langle\phi| f\rangle\right) \tag{272}$$

infatti la condizione di minimo viene scritta come

$$\frac{\partial E\left(|\phi\rangle\right)}{\partial \langle\phi|} = M |\phi\rangle - f\rangle = 0 \tag{273}$$

Per il problema generale di trovare il minimo di una funzione reale generica $e(\mathbf{x})$, con $\mathbf{x}$ appartenente ad un opportuno spazio vettoriale, uno dei metodi più semplici da implementare anche se generalmente alquanto lento è quello del *steepest descent.* Cominciamo con una prima scelta $\mathbf{x}^{(0)}$ poi applichiamo il seguente algoritmo

1. calcoliamo il gradiente $\mathbf{g}^{(i)} = -\frac{\partial e\left(\mathbf{x}^{(i)}\right)}{\partial \mathbf{x}}$
2. consideriamo la funzione reale $f(\lambda) = e\left(\mathbf{x}^{(i)} + \lambda \mathbf{g}^{(i)}\right)$ con $\lambda \in \mathbb{R}$
3. calcolo λ_{min} che minimizza $f(\lambda)$ imponendo $\frac{df}{d\lambda} = 0$
4. troviamo il nuovo vettore $\mathbf{x}$ con $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \lambda_{min}\mathbf{g}^{(i)}$
5. Torniamo al punto 1

Vediamo che Richardson corrisponde alla minimizzazione steepest-descent con la scelta di porre sempre $\lambda_{min} = \alpha$.

21 Equazione di Schrödinger stazionaria a tre dimensioni

L'equazione di Schrödinger stazionaria ha la forma:

$$\Delta\psi(\mathbf{r}) + c(\mathbf{r})\,\psi(\mathbf{r}) = E\psi(\mathbf{r}) \tag{274}$$

consideriamo il caso in cui $\mathbf{r}$ appartenga al volume Ω descritto in Sezione 20 con condizioni al contorno $\psi(\mathbf{r}) = 0 \;\; \mathbf{r} \in \partial\Omega$, trasformiamo il problema in un problema matriciale usando le definizione della Sezione 20 definiamo la matrice $H_{lm} \;\; l, m \notin \partial\Omega$ di dimensione $(N_x - 2)(N_y - 2)(N_z - 2)$, con $i \in [2, N_x - 1]$, $j \in [2, N_y - 1]$, $k \in [2, N_z - 1]$ e abbiamo $l \longleftrightarrow (i, j, k)$ e $m \longleftrightarrow (i', j', k')$

$$H_{lm} = \begin{cases} \frac{1}{h_x^2} & (i', j', k') = (i+1, j, k)\,, \;\; i+1 \neq N_x \\ -2\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right) + c_l & (i', j', k') = (i, j, k) \\ \frac{1}{h_x^2} & (i', j', k') = (i-1, j, k) \;\; i+1 \neq 0 \\ \frac{1}{h_y^2} & (i', j', k') = (i, j+1, k) \;\; j+1 \neq N_y \\ \frac{1}{h_y^2} & (i', j', k') = (i, j-1, k) \;\; j-1 \neq 0 \\ \frac{1}{h_z^2} & (i', j', k') = (i, j, k+1) \;\; k+1 \neq N_z \\ \frac{1}{h_z^2} & (i', j', k') = (i, j, k-1) \;\; k-1 \neq 0 \\ 0 & \text{altrimenti} \end{cases} \tag{275}$$

il nostro problema diventa:

$$\sum_m H_{lm}\psi_m = E\psi_l \tag{276}$$

dove voglio trovare le coppie (ψ, E) di autofunzioni-autovalori soluzioni del problema tali che le ψ trovate siano ortogonali tra di loro. Per la soluzione di questo problema posso usare gli algoritmi di algebra lineare per diagonalizzare la matrice Hnel caso la sua dimensione non sia proibitiva (ad esempio potrò usare le routine LAPACK) altrimenti potrò ricorrere ad algoritmi iterativi.

22 Formulazione del problema agli autovalori/autostati come problema di minimo

Trovare lo stato fondamentale di H risolvendo l' Eq. 276 è equivalente a trovare il minimo rispetto a $|\psi\rangle$ del seguente funzionale che scriviamo usando la notazione di Dirac:

$$E(|\psi\rangle) = \langle\psi|H|\psi\rangle \tag{277}$$

soggetto alla condizione di (orto-) normalizzazione

$$\langle\psi|\psi\rangle = 1 \tag{278}$$

Possiamo trattare tale vincolo inserendo un moltiplicatore di Lagrange λ e minimizzando rispetto a $|\psi\rangle$ e λ il funzionale:

$$E'(|\psi\rangle) = \langle\psi|H|\psi\rangle - \lambda(\langle\psi|\psi\rangle - 1) \tag{279}$$

Infatti nel minimo devono essere soddisfatte le seguenti equazioni:

$$\begin{cases} \frac{\partial E'}{\partial\langle\psi|} = 0 \\ \frac{\partial E'}{\partial\lambda} = 0 \end{cases}$$
$$\begin{cases} H|\psi\rangle - \lambda|\psi\rangle = 0 \\ \langle\psi|\psi\rangle - 1 = 0 \end{cases} \tag{280}$$

quindi nel minimo il valore del moltiplicatore di Lagrange è pari all'energia dello stato fondamentale. Per trovare il minimo l'algoritmo più semplice è lo steepest-descent, poniamo α parametro reale costante (es. 0.1):

1. Partiamo da guess iniziale $\psi^{(0)}$ che soddisfa $\langle\psi^{(0)}|\psi^{(0)}\rangle = 1$
2. Calcoliamo per la i-esima iterazione il gradiente cambiato di segno $|G(\lambda)>= -H|\psi^{(i)}> +\lambda|\psi^{(i)}>$
3. Scegliamo λ in maniera che $|\psi^{(i)}\rangle + \eta|G(\lambda)\rangle|^2 = 1 + O(\eta^2)$ che porge $\lambda^{(i)} = \langle\psi^{(i)}|H|\psi^{(i)}\rangle$
4. Poniamo $|\tilde{\psi}^{(i+1)}\rangle = |\psi^{(i)}\rangle + \alpha|G(\lambda^{(i)})\rangle$
5. Normalizziamo $|\psi^{(i+1)}\rangle = \frac{|\tilde{\psi}^{(i)}\rangle}{\sqrt{\langle\tilde{\psi}^{(i)}|\tilde{\psi}^{(i)}\rangle}}$
6. Torniamo al punto 2

Tale procedura permette il calcolo dello stato fondamentale. Possiamo trovare gli autostati di autoenergia via via maggiore proiettando ad ogni passo i vettori coinvolti sul sottospazio ortogonale al sottospazio degli autostati che sono già stati calcolati.

23 Equazione di diffusione

Consideriamo il solo caso dell'equazione di diffusione con costante di diffusione costante:

$$\frac{\partial}{\partial t}\rho(\mathbf{r},t) = D\Delta\rho(\mathbf{r},t) \tag{281}$$

per semplicità ci limitiamo al caso unidimensionale:

$$\frac{\partial}{\partial t}\rho(x,t) = D\frac{\partial^2}{\partial x^2}\rho(x,t) \tag{282}$$

con $x \in [a,b]$ e con le condizioni al contorno $\rho(a,t) = g_a$ e $\rho(b,t) = g_b$ e con le condizioni iniziali $\rho(x,t_0) = \rho^0(x)$. Come al solito introduciamo una griglia di N punti su $[a,b]$ con spaziatura h, in maniera tale che $x_1 = a$, $x_N = b$ e $h = (b-a)/(N-1)$. Inoltre introduciamo una griglia sull'asse temporale di istanti di tempo con spaziatura Δt, e poniamo:

$$t_n = t_0 + n\Delta t \tag{283}$$

e

$$\rho_i^{(n)} = \rho(x_i, t_n) \tag{284}$$

Per discretizzare il termine a sinistra dell'uguale in Eq. 282 possiamo scegliere tra i propagatori che abbiamo introdotto in Sezione 8. La scelta più semplice è quella di utilizzare il metodo di Eulero esplicito:

$$\begin{aligned}\rho_i^{(n+1)} &= \rho_i^{(n)} + D\frac{\partial^2}{\partial x^2}\rho\left(x,t\right)\Big|_{\substack{x=x_i \\ t=t_n}} \times \Delta t \\ \rho_i^{(n+1)} &= \rho_i^{(n)} + D\left(\frac{\rho_{i+1}^{(n)} - 2\rho_i^{(n)} + \rho_{i-1}^{(n)}}{h^2}\right)\Delta t\end{aligned} \tag{285}$$

Si vede e si può mostrare analiticamente che tale algoritmo diventa instabile se

$$\frac{D\Delta t}{h^2} > \frac{1}{2} \tag{286}$$

limitando quindi la scelta del time-step a valori piccoli. Tale relazione può essere argomentata (e dimostrata) nel seguente modo:

la propagazione temporale di $\rho(\mathbf{r},t)$ può essere scritta come:

$$\rho(\mathbf{r},t') = \int d\mathbf{r}' G(\mathbf{r},\mathbf{r}';t'-t)\rho(\mathbf{r}',t) \tag{287}$$

con G funzione di Green per l'equazione di diffusione:

$$G(\mathbf{r},\mathbf{r}',t'-t) = \left(\frac{1}{4\pi D(t'-t)}\right)^{\frac{N}{2}} e^{-\frac{|\mathbf{r}-\mathbf{r}'|^2}{4D(t'-t)}} \tag{288}$$

con N dimensione dello spazio Cartesiano di $\mathbf{r}$. Tale funzione è quindi una Gaussiana, normalizzata nello spazio N dimensionale, la cui ampiezza varia come $\sigma = \sqrt{2D(t'-t)}$. Con l'algoritmo di Eulero esplicito (vediamo qui il caso unidimensionale) il valore di $\rho_i^{(n)}$ dipende solamente dalla funzione ρ al tempo t_{n-1} entro una distanza h. Precisamente dipende solo da $\rho_i^{(n-1)}, \rho_{i-1}^{(n-1)}, \rho_{i+1}^{(n-1)}$. Se in Δt la nostra Gaussiana 'avanza' per più di h allora il valore di $\rho_i^{(n)}$ dovrà dipendere anche dal valore di $\rho^{(n-1)}$ in punti più distanti. Per avere convergenza bisognerà che:

$$\sqrt{2D\Delta t} < h \tag{289}$$

, elevando al quadrato si trova Eq. 289

Per ovviare a questo problema si può usare come propagatore il metodo di Eulero implicito, infatti con questo propagatore il valore di $\rho_i^{(n)}$ dipende da tutta la $\rho^{(n-1)}$:

$$\rho_i^{(n+1)} = \rho_i^{(n)} + D\left(\frac{\rho_{i+1}^{(n+1)} - 2\rho_i^{(n+1)} + \rho_{i-1}^{(n+1)}}{h^2}\right)\Delta t \tag{290}$$

ora però risulta meno immediato calcolare $\rho^{(n+1)}$ e dobbiamo ricorrere ad una formulazione matriciale. Introduciamo la matrice :

$$M_{ij} = \begin{cases} \frac{D}{h^2}\Delta t & j = i+1, i \neq 1, i \neq N \\ -\frac{2D}{h^2}\Delta t & j = i, i \neq 1, i \neq N \\ \frac{D}{h^2}\Delta t & j = i-1, i \neq 1, i \neq N \\ 0 & \text{altrimenti} \end{cases} \tag{291}$$

che risulta essere tridiagonale. In forma matriciale il nostro problema diviene:

$$\rho^{(n+1)} = \rho^{(n)} + M \cdot \rho^{(n+1)}\Delta t \tag{292}$$

che posso scrivere come:

$$(\mathbb{I} - M) \cdot \rho^{(n+1)} = \rho^{(n)} \tag{293}$$

risolvo questo equazione con i metodo dell'algebra lineare o usando metodi iterativi.

24 Equazione delle onde

Vogliamo trovare soluzioni all'equazione delle onde in una dimensione in un mezzo non dispersivo:

$$\frac{\partial^2}{\partial t^2} u(x,t) = c^2 \frac{\partial^2}{\partial x^2} u(x,t) \tag{294}$$

nel caso siano date le condizioni iniziali $u(x,t_0)$ e $\frac{\partial}{\partial t} u(x,t_0)$. Cominciamo col introdurre una griglia equispaziata sull' asse x dove h è la spaziatura ed una griglia equispaziata sull' asse t dove Δt è la spaziatura. Indichiamo con $u_i^{(n)} = u(x_i, t_n)$. La formula di Eq. 24 permette di scrivere

$$\frac{u_i^{(n+1)} - 2u_i^{(n)} + u_i^{(n-1)}}{\Delta t^2} = c^2 \frac{u_{i+1}^{(n)} - 2u_i^{(n)} + u_{i-1}^{(n)}}{h^2} \tag{295}$$

dove il termine incognito è $u_i^{(n+1)}$ che diviene:

$$u_i^{(n+1)} = 2u_i^{(n)} - u_i^{(n-1)} + \frac{c^2 \Delta t^2}{h^2} \left(u_{i+1}^{(n)} - 2u_i^{(n)} + u_{i-1}^{(n)} \right) \tag{296}$$

Quindi per propagare la funzione d'onda $u(x,t)$ dal tempo t_0 abbiamo bisogno non solo di $\left\{ u_i^{(0)} \right\}$ ma anche di $\left\{ u_i^{(1)} \right\}$ che possiamo calcolare utilizzando lo sviluppo in serie di Taylor:

$$u_i^{(1)} = u_i^{(0)} + \left(\frac{\partial}{\partial t} u_i^{(0)} \right) \Delta t + \frac{1}{2} \left(\frac{\partial^2}{\partial t^2} u_i^{(0)} \right) \Delta t^2 + O\left(\Delta t^3\right) \tag{297}$$

dove $\frac{\partial}{\partial t} u_i^{(0)} = g_i$ è una funzione data e sostituiamo $\left(\frac{\partial^2}{\partial t^2} u_i^{(0)} \right) = c^2 \left(\frac{\partial^2}{\partial x^2} u_i^{(0)} \right)$. Quindi si arriva alla formula:

$$u_i^{(1)} = u_i^{(0)} + g_i \Delta t + \frac{c^2}{2} \frac{u_{i+1}^{(0)} - 2u_i^{(0)} + u_{i-1}^{(0)}}{h^2} \Delta t^2 \tag{298}$$

Si vede numericamente e si può mostrare numericamente che tale algoritmo è stabile solo se:

$$\frac{c \Delta t}{h} \leq 1 \tag{299}$$

che viene chiamata condizione di *Courant-Friedrichs-Lewy*.

Infatti, con il propagatore che abbiamo trovato il valore di $u_i^{(n)}$ dipende solo da $u_i^{(n-1)}, u_{i+1} + 1^{(n-1)}, u_{i-1} + 1^{(n-1)}$. Però, se in Δt l'onda 'avanza' più di h , $u_i^{(n)}$ sarà determinato, nella soluzione esatta, anche da valore di $u^{(n-1)}$ più lontani.

Inoltre come fatto nelle sezione precedenti possono essere facilmente scelte delle condizioni al contorno.

25 Equazione di Schrödinger tempo dipendente

Consideriamo la soluzione dell'equazione di Schrödinger tempo dipendente:

$$H\psi(x,t) = i\hbar \frac{\partial}{\partial t} \psi(x,t) \tag{300}$$

con

$$H = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) \tag{301}$$

dove H non dipende dal tempo e $x \in [a,b]$ con le condizioni al contorno $\psi(a,t) = 0$ e $\psi(b,t) = 0$. Inoltre le condizioni iniziali $\psi(x,t_0)$ sono date. Come nella sezione precedente consideriamo una griglia sull' asse x di N_x punti spaziati h e time steps di Δt. La soluzione formale di Eq. 300 viene scritta come:

$$\psi(x,t) = e^{-\frac{i}{\hbar} H t} \psi_o(x) \tag{302}$$

che incorporando le condizioni iniziali scriviamo come:

$$\psi(x,t) = e^{-\frac{i}{\hbar} H(t-t_0)} \psi(x,t_0) = U(t,t_0)\, \psi(x,t_0) \tag{303}$$

dove $U(t,t_0) = e^{-\frac{i}{\hbar} H(t-t_0)}$ è l'operatore di evoluzione temporale. Esso è un operatore unitario e questo garantisce la conservazione della norma di ψ: $\int |\psi(x,t)|^2 \, dx = 1$.

Il metodo più semplice per risolvere numericamente è approssimare:

$$\psi\left(x,t+\Delta t\right)=\left(1-\frac{i}{\hbar}H\Delta t\right)\psi\left(x,t\right) \tag{304}$$

con la solita notazione $\psi_i^{(n)}=\psi\left(x_i,t_n\right)$ la formula diventa:

$$\psi_i^{(n+1)}=\psi_i^{(n)}-\frac{i\Delta t}{\hbar}\left(-\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2}\psi_i^{(n)}\right)+V_i\psi_i^{(n)}\right) \tag{305}$$

che approssimiamo come:

$$\psi_i^{(n+1)}=\psi_i^{(n)}-\frac{i\Delta t}{\hbar}\left(-\frac{\hbar^2}{2m}\left(\frac{\psi_{i+1}^{(n)}-2\psi_i^{(n)}+\psi_{i-1}^{(n)}}{h^2}\right)+V_i\psi_i^{(n)}\right) \tag{306}$$

tale metodo è equivalente a quello di Eulero esplicito quindi equivalente all'integrazione con il metodo dei rettangoli naif. Per una soddisfacente conservazione della norma della funzione d'onda bisogna però ricorrere a Δt piccoli visto che l'operatore approssimato $U\left(t+\Delta t,t\right)\approx\left(1-\frac{i}{\hbar}H\Delta t\right)$ non è unitario. Per usare un Δt maggiore possiamo basarci sul metodo di Crank Nicolson ossia sul metodo di integrazione con i trapezi. Formalmente procediamo nel seguente modo:

$$\psi\left(t_n+\frac{1}{2}\Delta t\right)=U\left(t_{n+\frac{1}{2}},t_n\right)\psi\left(t_n\right)=U\left(t_{n+1-\frac{1}{2}},t_{n+1}\right)\psi\left(t_{n+1}\right) \tag{307}$$

che discretizziamo con:

$$\psi_i^{(n)}-\frac{i\Delta t}{2\hbar}\left(-\frac{\hbar^2}{2m}\left(\frac{\psi_{i+1}^{(n)}-2\psi_i^{(n)}+\psi_{i-1}^{(n)}}{h^2}\right)+V_i\psi_i^{(n)}\right)=\psi_i^{(n+1)}+\frac{i\Delta t}{2\hbar}\left(-\frac{\hbar^2}{2m}\left(\frac{\psi_{i+1}^{(n+1)}-2\psi_i^{(n+1)}+\psi_{i-1}^{(n+1)}}{h^2}\right)+V_i\psi_i^{(n)}\right) \tag{308}$$

Anche in questo caso il problema può essere posto come un'equazione matriciale e per ottenere una forma più snella dividiamo ambo i membri per $-i\hbar\Delta t/\left(4mh^2\right)$:

$$\sum_{\substack{i=2,\ N_x-1\\ j=2,\ N_{x-1}}} M_{ij}\psi_j^{(n+1)}=F_i \tag{309}$$

con:

$$M_{ij}=\begin{cases} i\frac{4mh^2}{\hbar\Delta t}-2-\frac{2mh^2}{\hbar^2}V_i & j=i\\ 1 & j=i+1\\ 1 & j=i-1\\ 0 & \text{altrimenti}\end{cases} \tag{310}$$

e con:

$$F_i=-\psi_{i+1}^{(n)}+2\psi_i^{(n)}-\psi_{i-1}^{(n)}+i\frac{4mh^2}{\hbar\Delta t}\psi_i^{(n)}+\frac{2mh^2}{\hbar^2}V_i\psi_i^{(n)} \tag{311}$$

26 Metodo risolutivo per matrici tridiagonali

Consideriamo il problema generico di calcolare il vettore $\mathbf{a}$ tale che :

$$M\mathbf{a}=\mathbf{b} \tag{312}$$

con M matrice di tridiagonale di dimensione N. Il problema è equivalente ad un sistema di Nequazioni lineari. La prima di queste è:

$$M_{11}a_1+M_{12}a_2=b_1 \tag{313}$$

che possiamo scrivere come:

$$a_2=\alpha_1a_1+\beta_1 \tag{314}$$

dove:

$$\alpha_1 = -\frac{M_{11}}{M_{12}} \tag{315}$$

e

$$\beta_1 = \frac{b_1}{M_{12}} \tag{316}$$

per le altre equazioni possiamo sempre scrivere:

$$a_{i+1} = \alpha_i a_i + \beta_i \tag{317}$$

dove i coefficienti α_i e β_i vengono trovati tramite il seguente procedimento iterativo, dove usiamo $a_{i-1} = \frac{a_i}{\alpha_{i-1}} - \frac{\beta_{i-1}}{\alpha_{i-1}}$:

$$\begin{aligned} M_{i,i-1}a_{i-1} + M_{ii}a_i + M_{i,i_{+1}}a_{i+1} &= b_i \\ M_{i,i-1}\left(\frac{a_i}{\alpha_{i-1}} - \frac{\beta_{i-1}}{\alpha_{i-1}}\right) + M_{ii}a_i + M_{i,i_{+1}}a_{i+1} &= b_i \\ M_{i,i+1}a_{i+1} &= \left(-\frac{M_{i,i-1}}{\alpha_{i-1}} - M_{ii}\right)a_i + b_i + \frac{M_{i,i-1}}{\alpha_{i-1}}\beta_{i-1} \end{aligned} \tag{318}$$

da cui:

$$\alpha_i = \left(-\frac{M_{i,i-1}}{M_{i,i+1}\alpha_{i-1}} - \frac{M_{ii}}{M_{i,_{i+1}}}\right) \tag{319}$$

e

$$\beta_i = \frac{b_i}{M_{i,i+1}} + \frac{M_{i,i-1}}{M_{i,i+1}}\frac{\beta_{i-1}}{\alpha_{i-1}} \tag{320}$$

questo ci permette di trovare tutti i coefficienti fino a:

$$a_N = \alpha_{N-1}a_{N-1} + \beta_{N-1} \tag{321}$$

l'ultima equazione del sistema invece porge:

$$M_{NN}a_N + M_{N,N-1}a_{N-1} = b_N \tag{322}$$

metto a sistema le ultime due equazioni e trovo:

$$\frac{a_N}{\alpha_{N-1}} - \frac{\beta_{N-1}}{\alpha_{N-1}} = -\frac{M_{N,N}}{M_{N,N-1}}a_N + \frac{b_N}{M_{N,N-1}} \tag{323}$$

da cui trovo:

$$a_N = \left(\frac{1}{\alpha_{N-1}} + \frac{M_{N,N}}{M_{N,N-1}}\right)^{-1}\left(\frac{b_N}{M_{N,N-1}} + \frac{\beta_{N-1}}{\alpha_{N-1}}\right) \tag{324}$$

trovato a_N posso trovare tutti gli altri $\{a_i\}$ utilizzando Eq. 317

Parte VII
Metodi stocastici

In quest'ultima parte del corso vediamo come possiamo usare metodi basati sulla generazione di numeri casuali per risolvere alcune delle equazioni della fisica. Vedremo che non saremo limitati alla trattazione di soli processi stocastici (o casuali).

27 Generazione di numeri casuali

Ci interessa generare dei numeri casuali nell' intervallo $[0, 1[$. Per un numero casuale r si intende che la densità lineare di probabilità di generare tale numero è:

$$p(r) = 1 \tag{325}$$

Anche se in pratica è possibile la generazione di *veri* numeri casuali, ad esempio tirando un dado o misurando il decadimento radioattivo di un isotopo, essa è fuori portata per le applicazioni di fisica computazionale. Quindi dobbiamo ricorrere a metodi *pseudo-casuali* in cui generiamo dei numeri con una probabilità simile a quella ideale. Tali algoritmi generano serie di numeri $\{r_i\}$ nell'intervallo $[0, 1[$. Il numero di $\{r_i\}$ diversi che vengono generati è determinato dall'algoritmo usato e prende il nome di *periodo massimo*. Vediamo ora alcuni algoritmi:

27.1 Generatori LCG

Gli algoritmi (LCG) (linear congruential generators) sono basati sul seguente algoritmo:

1. Inizializzo $x_0 \in \mathbb{N}$ detto seme o *seed* .
2. scelgo due parametri $a, c \in \mathbb{N}$ e $m \in \mathbb{N}$ con $m > a, c$
3. uso l'iterazione $x_{i+1} = (ax_i + c) \mod m$
4. I numeri finali sono ottenuti come $r_i = \frac{x_i}{m}$

Di solito i parametri a, c, m sono tenuti costanti mentre il seme x_0 viene cambiato tra un *run* del programma ed il successivo in maniera da non generare la stessa sequenza di numeri. Per cambiare x_0 un metodo frequentemente usato è quello di ricavarlo dall'orologio di sistema del computer.

Una scelta di parametri usata frequentemente è quella di PARK-MILLER: $a = 7^5,\ c = 0,\ m = 2^{31} - 1$ che ha quindi un periodo di 2^{31}. Per ottenere periodi maggiori e quindi numeri pseudo-casuali di qualità maggiore è frequentemente usata le seguente variante che prende il nome di *shuffling*:

1. Scelgo un parametro $N > 0 \in \mathbb{N}$ e i quattro parametri $x_0,\ a,\ c,\ m$
2. con l'algoritmo LCG calcolo i primi N numeri $x_i \ \ i = 1, N$
3. pongo $k = N,\ l = 1$
4. Calcolo il numero intero $j = 1 + \text{floor}\,(N * x_k/M)$ nell'intervallo $[1, N]$
5. Pongo $r_l = x_j/m$
6. Pongo $x_j = x_k = (ax_i + c)$
7. Pongo $k = j$ e $l = l + 1$
8. Chiudo il loop tornando al punto 4

27.2 Test statistici sui numeri casuali

Per un insieme di numeri distribuiti secondo la distribuzione di Eq. 325 mi aspetto che il calcolo dei loro momenti sia in accordo con quello per la funzione $p\,(x)$:

$$\left\langle x^k \right\rangle = \int_0^1 x^k p(x) dx = \int_0^1 x^k dx = \frac{1}{k+1} \tag{326}$$

questi li confronto con i valori medi ottenuti dai miei numeri pseudo casuali:

$$\overline{x^k} = \frac{1}{N} \sum_{i=1,N} r_i^k \tag{327}$$

inoltre posso andare ad investigare se i numeri pseudo-casuali sono *scorrelati* ossia se, dato $l > 0$:

$$\langle r_i r_{i+l} \rangle = \langle r_i \rangle \langle r_{i+l} \rangle = \frac{1}{2}\frac{1}{2} = \frac{1}{4} \tag{328}$$

per calcolare la funzione di correlazione per i miei numeri uso:

$$\overline{r_i r_{i+l}} = \frac{1}{N-l} \sum_{i=1,N-l} r_i r_{i+l} \tag{329}$$

Infine un test grafico della correlazione può essere effettuo plottando i punti (r_i, r_{i+1}) su di un grafico cartesiano e andando a vedere se ci sono delle periodicità evidenti.

28 Sampling di funzioni densità di probabilità

Ora vogliamo che i nostri numeri casuali siano distribuiti secondo una funzione densità di probabilità $p(x)$ con $x \in [a, b]$ normalizzata

$$\int_a^b p(x)\, dx = 1 \tag{330}$$

ossia vogliamo che dati $y_1 < y_2$ se generiamo N numeri casuali $\{x_i\}$ e N_{y_1,y_2} è il numero di numeri $\in [y_1, y_2]$ nel limite di grande N si verifichi:

$$\frac{N_{y_1,y_2}}{N} \to \int_{y_1}^{y_2} p(x)\, dx \tag{331}$$

inoltre vogliamo generare i numeri (pseudo-)casuali $\{x_i\}$ a partire dai numeri $\{r_i\}$ uniformemente distribuiti nell' intervallo $[0, 1]$. Per questo compito sono disponibili alcuni metodi:

28.1 Metodo della trasformata inversa

Cominciamo con il calcolare la funzione comulante:

$$P(x) = \int_a^x p(x)\, dx \tag{332}$$

essa è una funzione monotona (p è positiva) con $p(a) = 0$ $p(b) = 1$ quindi (almeno in principio) invertibile. Supponiamo ora di generare N numeri casuali distribuiti uniformemente tra 0 e 1 e sia N_{r_1,r_2} il numero di numeri casuali tra $0 \leq r_1 < r_2 \leq 1$. Consideriamo ora i numeri casuali ottenuti dalla trasformazione $x_i = P^{-1}(r_i)$ corrispondenti a quelli $\in [r_1, r_2]$ essi saranno tutti compresi nell'intervallo $\left[x_1 = P^{-1}(r_1), x_2 = P^{-1}(r_2)\right]$ dove $r_1 = P(x_1)$ e $r_2 = P(x_2)$ da cui:

$$N_{x_1,x_2} = N_{r_1,r_2} = \frac{r_2 - r_1}{1} N = [P(x_2) - P(x_1)]\, N = \left[\int_a^{x_2} p(x)dx - \int_a^{x_1} p(x)dx\right] N = N \int_{x_1}^{x_2} p(x)dx \tag{333}$$

quindi i numeri $x_i = P^{-1}(r_i)$ sono distribuiti secondo la funzione densità di probabilità $p(x)$. Il metodo della trasformata inversa può essere applicato però solo nel caso in cui la funzione P è invertibile.

28.2 Sampling della funzione Gaussiana

Vediamo ora come trovare numeri casuali distribuiti secondo la densità di probabilità Gaussiana: $p(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$. In questo caso l'applicazione diretta del metodo della precedente sezione non è possibile visto che non riusciamo ad avere una formula analitica per la funzione comulante. Riformulando il problema si arriva alla seguente ricetta: si prendono due numeri casuali r_1 e r_2. Poi viene ottenuto un numero casuale z distribuito secondo la densità di probabilità Gaussiana usando la seguente formula:

$$z = \cos(2\pi r_2) \sqrt{-2 \ln(r_1)} \tag{334}$$

28.3 Metodo accetta/nega

Abbiamo già visto questo metodo nella Sezione 16. Consideriamo una densità di probabilità $p(x)$ con $x \in [a,b]$ indichiamo con $p_{max} = \max(p)$. L'algoritmo è il seguente:

1. Generiamo un numero casuale r_1 nell'intervallo $[0,1]$
2. Consideriamo il corrispondente numero casuale $x = (b-a) r_1 + a$ distribuito uniformemente in $[a,b]$
3. Generiamo un numero casuale r_2 nell'intervallo $[0,1]$
4. Solo se $r_2 < p(x)/p_{max}$, aggiungiamo il numero casuale x ai numeri generati dall'algoritmo
5. Torna al numero 1

Si vede facilmente che i numeri generati dall' algoritmo soddisfano la distribuzione richiesta

29 Integrazione con numeri casuali o Monte Carlo

Cominciamo con considerare l'integrazione di una funzione generica $f(x)$ sull'intervallo $[a,b]$:

$$\int_a^b f(x)\,dx = \overline{f}\,(b-a) \tag{335}$$

dove il valore medio $\overline{f}$ della funzione è definito come:

$$\overline{f} = \frac{\int_a^b f(x)\,dx}{b-a} \tag{336}$$

Però noi possiamo calcolare $\overline{f}$ con una certa precisione generando N numeri casuali x_i distribuiti uniformemente su $[a,b]$ e mediando gli $f_i = f(x_i)$, dalle formule di teoria della probabilità si trova:

$$\overline{f} = \frac{1}{N}\sum_{i=1,N} f_i \pm \sqrt{\frac{\mathrm{var}(f)}{N}} \tag{337}$$

dove la varianza è definita come:

$$\mathrm{var}(f) = \overline{(f-\overline{f})^2} = \overline{f^2} - 2\overline{f\overline{f}} + \overline{f}^2 = \overline{f^2} - \overline{f}^2 \tag{338}$$

con

$$\overline{f^2} = \frac{1}{N}\sum_{i=1,N} f_i^2 \tag{339}$$

la varianza $\mathrm{var}(f)$ può quindi essere calcolata esattamente come:

$$\mathrm{var}(f) = \frac{\int_a^b (f(x))^2\,dx}{b-a} - \left(\frac{\int_a^b f(x)\,dx}{b-a}\right)^2 \tag{340}$$

quindi l'accuratezza di integrazione è di ordine $O\left(\frac{1}{\sqrt{N}}\right)$. Ci ricordiamo che il metodo dei rettangoli naif ha un'accuratezza $O\left(\frac{1}{N}\right)$ mentre quello dei rettangoli e quello dei trapezi hanno un'accuratezza $O\left(\frac{1}{N^2}\right)$. Quindi per integrare funzioni definite su $\mathbb{R}$ usare numeri casuali non è conveniente.

Il discorso cambia invece se consideriamo funzioni definite su $\mathbb{R}^M$. Supponiamo quindi di usare il metodo dei rettangoli (o trapezi) e di valutare la funzione in N punti, distribuiti in maniera regolare, cosicché lungo ogni dimensione di $\mathbb{R}^M$ io ho $N^{\frac{1}{M}}$ punti. L'accuratezza con cui integriamo (coi rettangoli-trapezi) è determinata dal numero di punti lungo le singole dimensioni quindi risulta essere $O\left(\frac{1}{N^{2/M}}\right)$.

Usando invece il metodo dei numeri casuali, l'accuratezza è sempre di ordine $O\left(\frac{1}{\sqrt{N}}\right)$. Quindi per $M > 4$ il metodo dei numeri casuali (o Monte Carlo) risulta essere più accurato a parità di costo computazionale.

29.1 Integrazione di funzione composte

Consideriamo ora l'integrazione (per semplicità di sola notazione consideriamo funzioni su $\mathbb{R}$) di una funzione del tipo $f(x)\,p(x)$ su $[a,b]$ dove $p(x)$ è una densità di probabilità definita sullo stesso intervallo $[a,b]$. E' facile vedere che posso calcolare:

$$\int_a^b f(x)\,p(x)\,dx = (b-a)\left(\frac{1}{N}\sum_{i=1,N} f_i \pm \sqrt{\frac{\mathrm{var}(f)}{N}}\right) \tag{341}$$

dove gli N numeri casuali x_i sono distribuiti secondo la distribuzione $p(x)$ e $f_i = f(x_i)$

Infatti consideriamo un numero N grande di numeri casuali distribuiti secondo $p(x)$. Il numero dN di numeri casuali in dx è pari a:

$$dN = Np(x)\frac{dx}{b-a} \tag{342}$$

e il contributo al termine di destra in Eq. 341 è:

$$(b-a)\frac{dN}{N}f(x) = (b-a)f(x)\frac{Np(x)\frac{dx}{b-a}}{N} = f(x)p(x)dx \tag{343}$$

quindi pari al termine che compare nell'integrale nel menbro di sinistra di Eq. 341.

30 L'algoritmo di Metropolis

Spesso è richiesto di calcolare integrali del tipo:

$$\int dx O(x)\,p(x) \tag{344}$$

dove la densità di probabilità $p(x)$ è definita come:

$$\begin{cases} p(x) & = \frac{q(x)}{Z} \\ Z & = \int dx q(x) \end{cases} \tag{345}$$

dove con x indichiamo un elemento di uno spazio generico, O è una genrica funzione corrispondente ad un osservabile e $q(x)$ e una funzione densità di probabilità NON normalizzata. La costante di normalizzazione è Z cosicchè $p(x)$ è una buona funzione densità di probabilità.

Ad esempio per un sistema di particella classiche, come quelli che abbiamo visto in Sezione 10, il valore medio di un osservabile O in condizioni di equilibrio termico alla temperatura T viene calcolato come:

$$\langle O\rangle_T = \int dq_1 \ldots \int dq_N \int dp_1 \ldots \int dp_N O(q_1,\ldots,q_N,p_1,\ldots,p_N)\,\frac{e^{-\frac{1}{k_BT}E(q_1,\ldots,q_N,p_1,\ldots,p_N)}}{Z} \tag{346}$$

con la funzione di partizione Z definita come:

$$Z = \int dq_1 \ldots \int dq_N \int dp_1 \ldots \int dp_N e^{-\frac{1}{k_BT}E(q_1,\ldots,q_N,p_1,\ldots,p_N)} \tag{347}$$

Quindi per calcolare $p(x)$ abbiamo bisogno di Z ma per calcolare Z dobbiamo calcolare un integrale della stessa complessità. Quindi integrare con il metodo di Monte Carlo le formule di Eqs. 344,346 non è possibile a meno di generare i punti casuali $\{x_i\}$ secondo la distribuzione $p(x)$ e poi usare il metodo di Sezione 29.1. Siccome non conosciamo $p_{max} = \max(p(x))$ il metodo accetta-nega può essere utilizzato solo usando un valore massimo p'_{max} sufficientemente grande da essere sicuri che $p'_{max} > p_{max}$. Questo può risultare altamente inefficiente.

Metropoils invece introdusse un algoritmo efficiente per generare i punti casuali $\{x_i\}$ secondo la distribuzione $p(x)$ usando solo la funzione $q(x)$. L'algoritmo è il seguente:

1. Generiamo casualmente il primo punto x_1, poniamo $i = 1$
2. Generiamo casualmente il punto x_{trial}
3. Se $p(x_{trial}) > p(x_i)$ ossia se $q(x_{trial}) > q(x_i)$, aggiungiamo il punto ai numeri generati: $x_{i+1} = x_{trial}$, $i = i+1$, TORNIAMO A 2
4. Altrimenti generiamo casulamente un numero $r \in [0,1]$

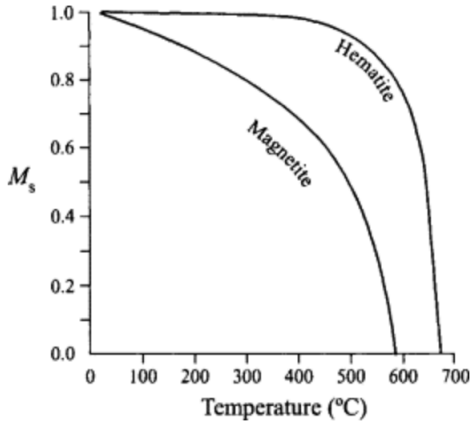


Figura 13: Andamento della magnetizzazione relativa rispetto alla temperatura.

5. Se $\frac{p(x_{trial})}{p(x_i)} = \frac{q(x_{trial})}{q(x_i)} > r$ aggiungiamo il punto ai numeri generati: $x_{i+1} = x_{trial}$, $i = i + 1$, TORNIAMO A 2
6. Altrimenti aggiungiamo x_i ai numeri generati: $x_{i+1} = x_i$, $i = i + 1$, TORNIAMO A 2

L'algoritmo di Metropolis definisce una funzione probabilità $c(x)$ che un punto venga generato dato un certo numero di passi. Inoltre soddisfa la condizione di bilancio dettagliato ossia il numero medio di x_j generato subito dopo (o a partire da) x_i è pari al numero di punti x_i generati a partire da x_j. Questa proprietà è indicata nel lavoro orginale di Metropolis e collaboratori come dovuta alla semplice condizione di equilibrio in un sistema ergodico (e invariante per inversione temporale) e si può dimostare in maniera rigorosa tramite la teoria delle catene di Markov. Ora dimostriamo che il bilancio dettagliato implica il fatto che $c(x)$ è proporzionale a $p(x)$ come auspicato. Indichiamo con $p(x_j|x_i)$ la probabiltà di generare x_j da x_i:

$$p(x_j|x_i) = \begin{cases} 1 & p(x_j) > p(x_i) \\ \frac{p(x_j)}{p(x_i)} = \frac{q(x_j)}{q(x_i)} & p(x_j) \leq p(x_i) \end{cases} \tag{348}$$

allora il bilancio dettagliato viene scritto come:

$$p(x_j|x_i)\, c(x_i) = p(x_i|x_j)\, c(x_j) \tag{349}$$

se siamo nel caso $p(x_j) > p(x_i)$, il bilancio dettagliato diviene:

$$c(x_i) = \frac{p(x_i)}{p(x_j)} c(x_j) \tag{350}$$

mentre se $p(x_j) \leq p(x_i)$, il bilancio dettagliato diviene:

$$\frac{p(x_j)}{p(x_i)} c(x_i) = c(x_j) \tag{351}$$

in entrambi i casi si ottiene:

$$\frac{c(x_i)}{c(x_j)} = \frac{p(x_i)}{p(x_j)} \tag{352}$$

quindi la funzione $c(x)$ che posso scrivere come $c(x) = \frac{p(x)}{p(x_1)} c(x_1)$ è come voluto proporzionale a $p(x)$.

31 Modello di Ising per il magnetismo

Il magnetismo nei materiali è dovuto ai gradi di libertà elettronici e in particolare alla loro componente di spin.I materiali chiamati ferromagnetici esibiscono magnetizzazione spontanea (ossia presente anche in assenza di campo magnetico esterno) fino ad una certa temperatura critica chiamata temperatura di Curie. Il tipico andamento temperatura magnetizzazione è raffigurato in Figura 13 Alla temperatura di Curie i materiali esibiscono una transizione di fase di secondo ordine, ossi la magnetizzazione è continua ma con derivata discontinua. La magnetizzazione è dovuta alla presenza nel materiale di atomi di metalli di transizione (es. Fe) che contengo una shell di orbitali d incompleta con un numeri di elettroni dispari. Siccome tali orbitali sono localizzati è sensato introdurre un semplice modello per descrive l'energia (o

meglio l'operatore di Hamilton) del sistema. Gli unici gradi di libertà sono l'orientazione degli spin che indichiamo che con σ_l dove l indica il sito atomico di appartenenza. La variabile σ_l può assumere solo i valori ± 1. La funzione Hamiltoniana modello che descrive il sistema è data da:

$$H = -\frac{1}{2}\sum_{l \neq l'} J_{ll'}\sigma_l\sigma_{l'} - h\sum_l \sigma_l \tag{353}$$

dove i coefficienti $J_{ll'}$ rendono conto del termine di scambio tra il sito l e quello l' e abbiamo $J_{ll'} = J_{l'l}$. L'ultimo termine nel membro di destra tiene conto dell'accoppiamento con un eventuale campo magnetico esterno e abbiamo $h = -\mu_B g B/2$ con μ_B momento magnetico dell'elettrone e dove il fattore g, detto di Landé rende conto del fatto della perturbazione introdotta dagli altri elettroni del cristallo.

Se $O(\{\sigma\})$ è un generico osservabile, il suo valore di aspettazione alla temperatura T è dato da:

$$\langle O\rangle_T = \frac{\sum_{\{\sigma\}} O\{\sigma\}\, e^{-\frac{1}{k_B T}H(\{\sigma\})}}{\sum_{\{\sigma\}} e^{-\frac{1}{k_B T}H(\{\sigma\})}} \tag{354}$$

dove con $\{\sigma\} = (\sigma_1, \ldots, \sigma_{N_{siti}})$ indichiamo una configurazione di spin per tutti gli N atomi o siti del nostro sistema. Gli osservabili che ci interesseranno maggiormente sono:

l'energia per unità di sito:

$$E(\{\sigma\}) = \frac{H(\{\sigma\})}{N} \tag{355}$$

e la magnetizzazione per sito:

$$M(\{\sigma\}) = \frac{1}{N}\sum_l \sigma_l \tag{356}$$

Inoltre si può mostare che i termini di scambio che contribuiscono maggiormente sono quelli le coppie di siti più vicini (detti *primi-vicini*) . Allora possiamo ulteriormente semplificare il nostro modello, che diviene il noto modello di Ising:

$$H = -\frac{1}{2}\widetilde{\sum_{l \neq l'}} J\sigma_l\sigma_{l'} - h\sum_l \sigma_l \tag{357}$$

dove con $\widetilde{\sum}$indichiamo che la somma è limitata ai primi vicini. Si vede subito che se $J > 0$ l'ordine ferromagnetico è preferito ossia con gli spin con medesimo orientazione mentre se $J < 0$ l'ordine antiferromagnetico è preferito ossia con gli spin con orientazioni opposte.

Il modello di Ising amette soluzioni analitiche nei casi 1D e 2D per il limite di numero di siti infiniti.Nel caso 1D non è osservata alcuna transizione di fase mentre il modello 2D mostra una transizione di fase analoga a quella misurata.

Nel caso 3D invece non è stato possibile ancora trovare soluzioni analitiche. Pertanto il modello di Isig (e modelli simili) vegono studiati con metodi Monte Carlo ed in particolare con l'algoritmo di Metropolis.

Notiamo che per $h = 0$ il sistema è simmetrico (ossia l'energia è invariante) per inversione di tutti gli spin. Quindi ci aspettiamo che per ogni temperatura T:$\langle M\rangle_{T,h=0} = 0$. In questa maniera però non riusciamo studiare la magnetizzazione permanente. Il problema si risolve andando a definire e poi a studiare:

$$\langle M\rangle_{T,h=0^+} = \lim_{h\to 0^+}\langle M\rangle_{T,h} \tag{358}$$

Nel applicare il metodo di Metropolis al modello di Ising di solito ci si limita la generazione della configurazione di prova (σ_{trial}) all'inversione di uno solo spin scelto a caso rispetto alla configurazione precedente. Questo da un lato rende l'algoritmo molto più efficiente dall'altro *tende a bloccare* il sistema in un sottospazio dello spazio delle configurazioni con maggior parte degli spin aventi medesima orientazione tanto che per temperature basse e per numero di configurazioni generate relativamente limitato si trova $\overline{M}_{h=0} \neq 0$ e $\left|\overline{M}_{h=0}\right| \cong \langle M\rangle_{T,h=0^+}$.

Nelle Figure 14,15,16 riportiamo l'energia per sito e la magnetizzazione per sito nel caso del modello 2D con condizioni al contorno periodiche e una griglia di 50×50 spin. Notiamo che lo stato fondamentale ad energia più bassa corrisponde a tutti gli spin aventi la medesimo orientazione ed è quindi doppiamente degenere. Per tale stato l'energia per sito è $-2J$ e la magnetizzazione per sito è ± 1.

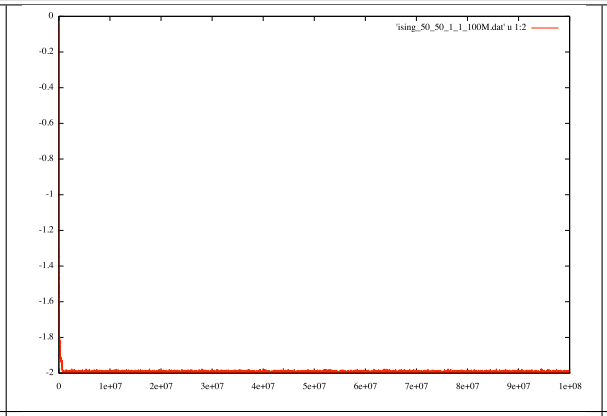
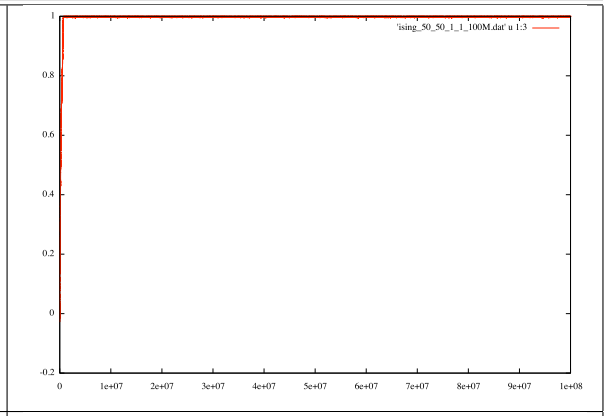
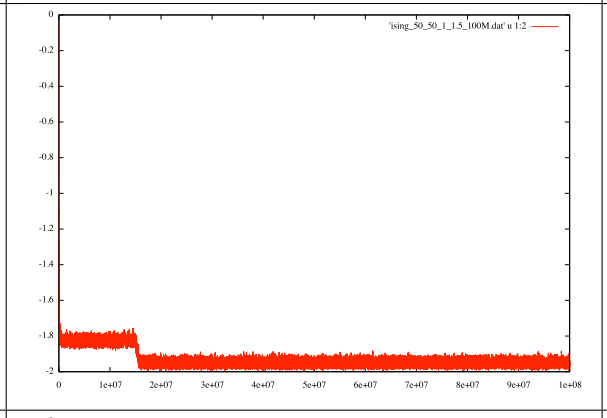
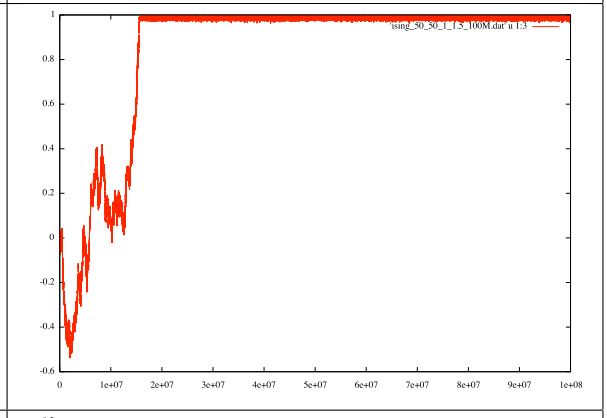
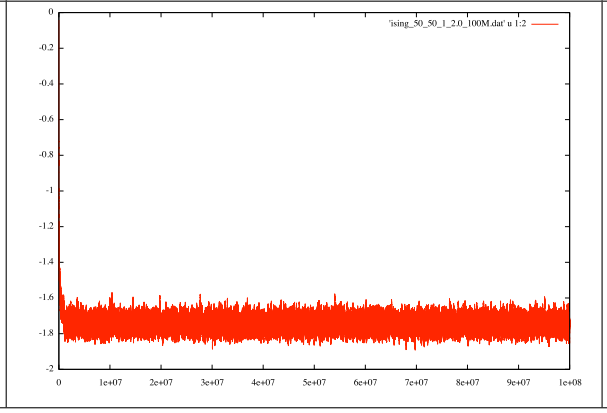
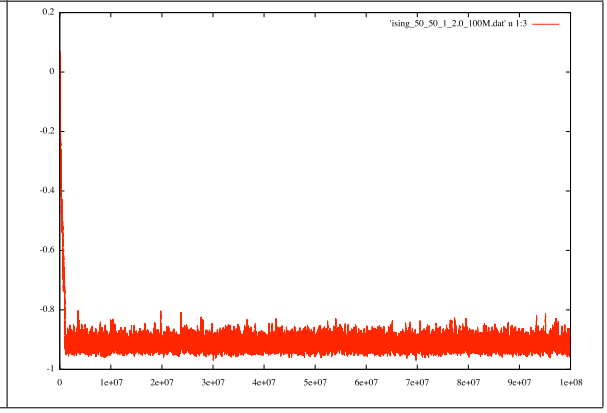
T	Energia per sito	Magnetizzazione per sito
1		
1.5		
2.0		

Figura 14: Simulazione Monte Carlo con l'algoritmo di Metropolis per il modello di Ising 2D con dimensioni 50x50 e condizini al contorno periodiche. $J = 1$ e $h = 0$. Ogni simulazione consta di $100 \cdot 10^6$ di passi.

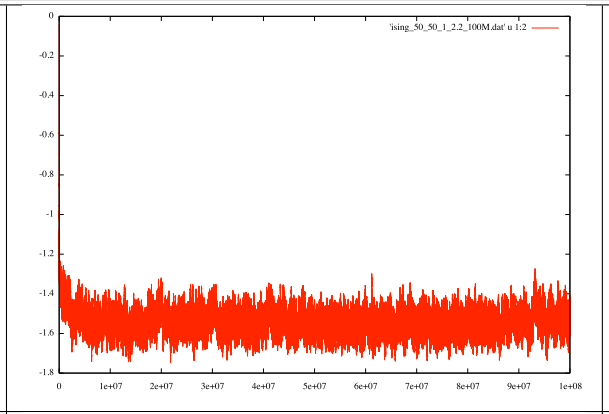
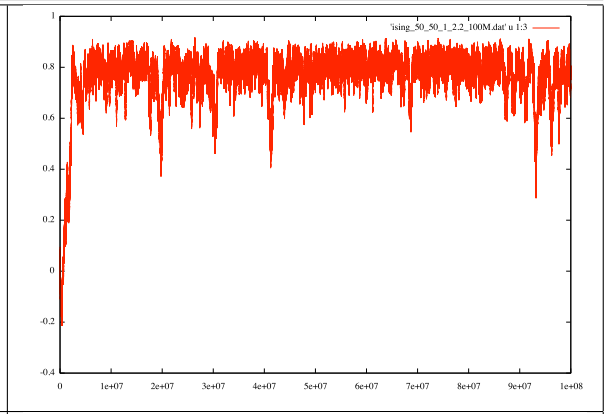
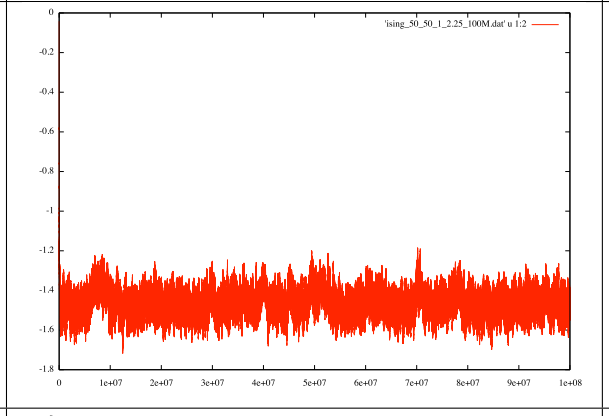
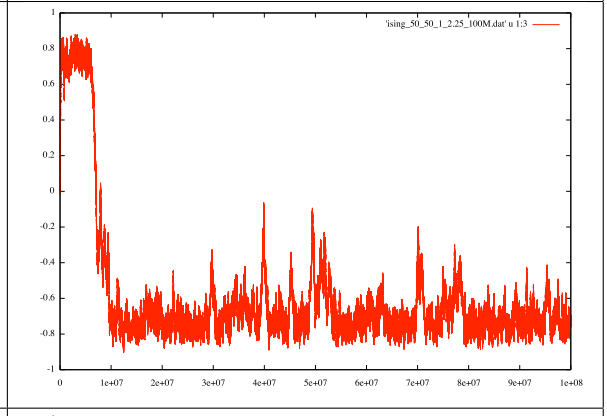
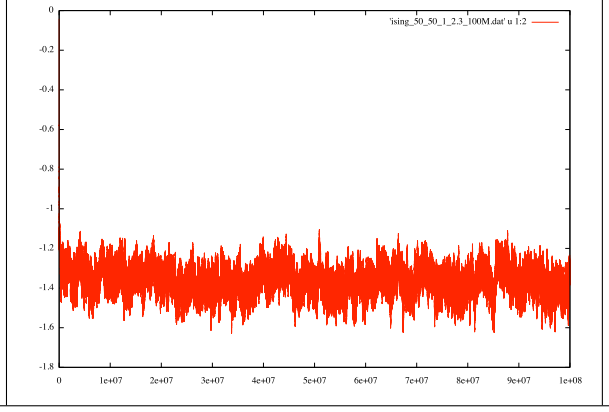
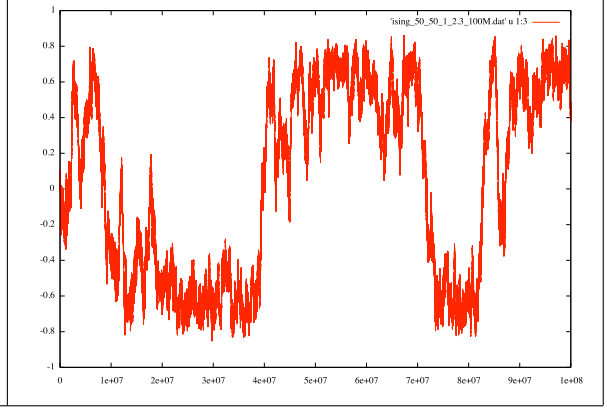
T	Energia per sito	Magnetizzazione per sito
2.2		
2.25		
2.3		

Figura 15: Simulazione Monte Carlo con l'algoritmo di Metropolis per il modello di Ising 2D con dimensioni 50x50 e condizini al contorno periodiche. $J = 1$ e $h = 0$. Ogni simulazione consta di $100 \cdot 10^6$ di passi.

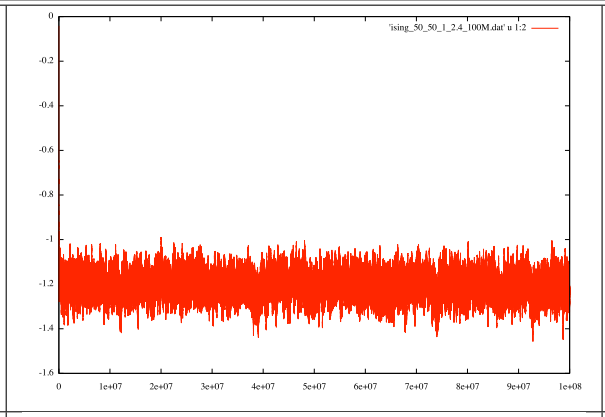
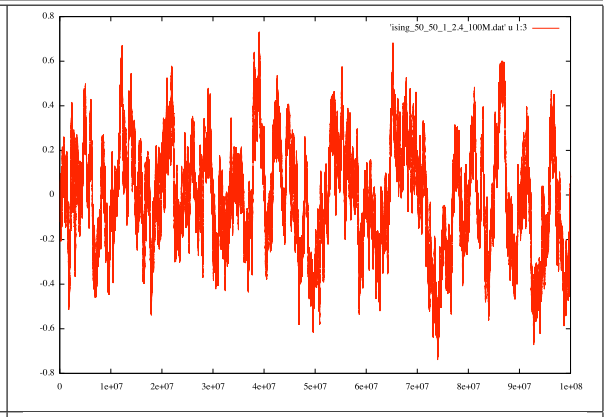
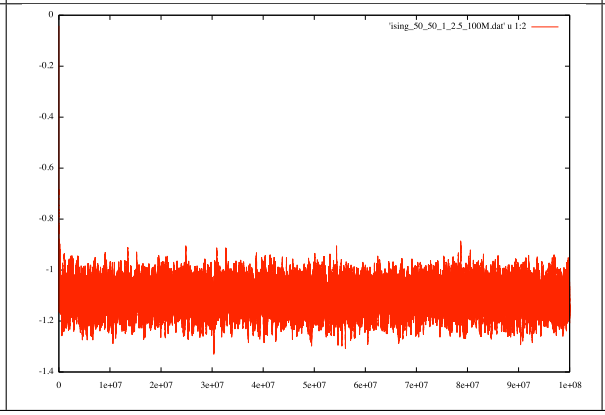
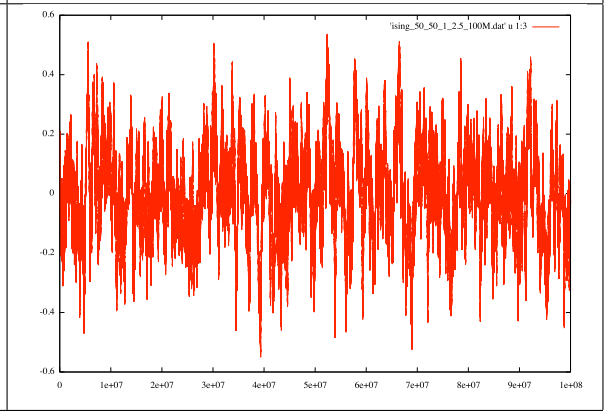
T	Energia per sito	Magnetizzazione per sito
2.4		
2.5		

Figura 16: Simulazione Monte Carlo con l'algoritmo di Metropolis per il modello di Ising 2D con dimensioni 50x50 e condizini al contorno periodiche. $J = 1$ e $h = 0$. Ogni simulazione consta di $100 \cdot 10^6$ di passi.

Parte VIII
Algoritmi di ottimizzazione

Cominciamo con il problema di trovare il minimo (o il massimo) di una funzione $f(x)$ $\Re \to \Re$. Qui parleremo sempre di minimi. Basta moltiplicare f per -1 per passare da un problema di minimo a uno di massimo.

32 Metodo della sezione aurea

Consideriamo $f(x)$ $\Re \to \Re$ continua e tre punti lungo x : $a < b < c$. Supponiamo valga: $f(b) < f(a)$ e $f(b) < f(c)$ allora c'è sicuramente (almeno) un minimo di f in $[a, c]$. Possiamo trovarlo con il seguente algoritmo iterativo (vedi Fig. 17) :

1. prendiamo $a < b < c$ tali che $f(b) < f(a)$ e $f(b) < f(c)$
2. se $(c - b) > (b - a)$ prendiamo $x_0 \in [b, c]$:
   (a) se $f(x_0) > f(b)$ allora ho un minimo in $[a, x_0]$, pongo $c = x_0$ e torno a (1)
   (b) altrimenti ho un minimo in $[b, c]$, pongo $a = b$, $b = x_0$ e torno a (1)
3. se $(c - b) > (b - a)$ prendiamo $x_0 \in [a, b]$:
   (a) se $f(x_0) > f(b)$ allora ho un minimo in $[x_0, c]$, pongo $a = x_0$ e torno a (1)
   (b) altrimenti ho un minimo in $[a, b]$, pongo $b = x_0$, $c = b$ torno a (1)

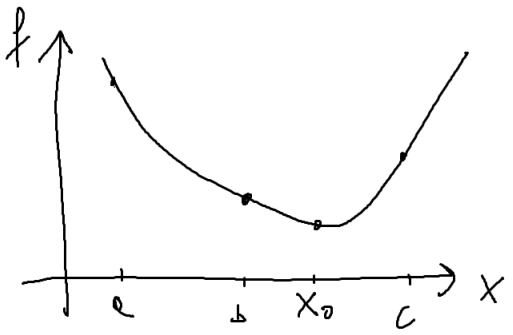


Figura 17: Minimizzazione di una funzione tramite il metodo della sezione aurea

Vogliamo trovare un metodo per scegliere x_0 in maniera tale che i due possibili nuovi intervalli di ricerca abbiano lunghezza uguale. Supponiamo di essere nel caso $(c-b) > ((b-a)$ e poniamo:

$$w = \frac{b-a}{c-a} \tag{359}$$

$$z = \frac{x_0 - b}{c-a} \tag{360}$$

Se $f(x_0) > b$ allora il nuovo intervallo di ricerca è $[a, x_0]$ con lunghezza $(w+z)(c-a)$, altrimenti l'intervallo è $[b, c]$ con lunghezza $(1-w)(c-a)$. I due intervalli sono uguali se:

$$z = 1 - 2w \tag{361}$$

Quindi dato w possiamo ricavare z e quindi x_0. Dobbiamo però ancora determinare il primo w. Supponiamo $f(x_0) < f(b)$. Imponiamo di avere sempre le stesse proporzioni tra i punti a, b, x_0 e b, x_0, c, ossia:

$$\frac{b-a}{c-a} = \frac{x_0 - b}{b - x_0} \tag{362}$$

$$w = \frac{z}{1-w} \tag{363}$$

Sostituiamo Eq. 361 e troviamo:

$$w = \frac{1-2w}{1-w} \tag{364}$$

$$w^2 - 3w + 1 = 0 \tag{365}$$

Prendiamo la sola soluzione soluzione entro $[0, 1]$:

$$w = \frac{3-\sqrt{5}}{2} \simeq 0.38197 \tag{366}$$

Vediamo ora in pratica come funziona. Cominciamo con scegliere a, c e poi poniamo $b = a + w(c-a)$. Abbiamo un minimo se $f(b) < f(a)$ e $f(b) < f(c)$ in tal caso, scegliamo $x_0 = b + z(c-a)$. Ora abbiamo due casi:

Se $f(x_0) < b$ ricominciamo con b, x_0, c e si verifica facilmente che $(x_0 - b)/(c-b) = w$.

Se $f(x_0) > b$ ricominciamo con a, x_0, b, in tal caso la situazione è *ribaltata* infatti si verifica facilmente che $(x_0 - b)/(x_0 - a) = w$ Quindi sceglieremo il nuovo punto di prova $x_0' = b - z(x_0 - a)$. Notiamo che w è proprio la sezione aurea.

33 Approssimazione con una parabola

In molti casi $f(x)$ è approssimabile con una parabola in prossimità del minimo con una parabola. Valutiamo f in $a < b < c$. e approssimiamo:

$$f(x) \approx \alpha + \beta x + \gamma x^2 \tag{367}$$

I coefficienti $alpha, \beta, \gamma$ sono univocamente determinati dalla richiesta che la funzione approssimante passi per $(a, f(a)), (b, f(b)), (c, f($
Poi il minimo approssimato di f è nel punto:

$$x_0 = -\frac{\beta}{2\gamma} \tag{368}$$

svolgendo i calcoli si trova:

$$x_0 = b - \frac{1}{2}\frac{(b-a)^2(f(b)-f(c)) - (b-c)^2(f(b)-f(a))}{(b-a)(f(b)-f(c)) - (b-c)(f(b)-f(a))} \tag{369}$$

In diversi casi è disponibile anche la derivata di f, $f'(x) = df(x)/dx$. In tal caso mi bastano due punti $a < b$ per trovare la parabola. Mi basta conoscere f' in uno dei due punti che supponiamo sia b. Allora, i parametri β e γ sono dati da:

$$\beta = f'(b) - 2\gamma b \tag{370}$$

$$\gamma = \frac{(f(a)-f(b)) - f'(b)(a-b)}{(a^2-b^2) - 2b(a-b)} \tag{371}$$

34 Metodi basati sul gradiente

Consideriamo il problema di trovare il minimo (o i minimi) di una generica funzione $F(x_1, \ldots, x_N)$ rispetto ai parametri $x_1, \ldots, x_{N.}$

Supponiamo sia possibile calcolare il gradiente:

$$\nabla F(x_1, \ldots, x_N) = \begin{pmatrix} \frac{\partial}{x_1} F(x_1, \ldots, x_N) \\ \frac{\partial}{\partial x_2} F(x_1, \ldots, x_N) \\ \cdots \\ \frac{\partial}{\partial x_N} F(x_1, \ldots, x_N) \end{pmatrix} \tag{372}$$

allora possiamo trovare il minimo spostandoci lungo la direzione opposta al gradiente. Secondo il seguente algoritmo detto *Steepest descend*.

1. Dati i valori iniziali $x_1^i, \ldots, x_{N.}^i$ calcolo $\nabla F\left(x_1^i, \ldots, x_N^i\right)$
2. Calcolo il parametro λ tale che $F\left(x_1^i - \lambda\frac{\partial}{\partial x_1}F\left(x_1^i, \ldots, x_N^i\right), \ldots, x_N^i - a\frac{\partial}{\partial x_N}F\left(x_1^i, \ldots, x_N^i\right)\right)$ sia minimo
3. Pongo per il valore di λ trovato $x_1^{i+1} = x_1^i - \lambda\frac{\partial}{\partial x_1}F\left(x_1^i, \ldots, x_N^i\right), \ldots, x_N^{i+1} = x_N^i - \lambda\frac{\partial}{\partial x_N}F\left(x_1^i, \ldots, x_N^i\right)$ e ritorno al punto 2

Talvolte è conveniente usare un valore di λ prefissato senza ricavarlo ad ogni iterazione.

Una metodo affine basato sui gradiente è quello del *Gradiente Coniugato* che permette di scegliere la direzione lineare di minizzazione in maniera più efficiente. Indichiamo con $\mathbf{u}$ il vettore direzione di ricerca. Il metodo steepest-descent pone:

$$\mathbf{u}^i = -\frac{\nabla f(\mathbf{x}^i)}{|\nabla f(\mathbf{x}^i)|} \tag{373}$$

Il metodo del gradiente coniugato genera delle direzioni di ricerca in maniera tale che la minimizzazione lungo la direzione successiva non rovini le minimizzazioni lungo le direzioni precedenti. Supponiamo di approssimare f in serie di Taylor attorno all'origine del sistema di riferimento:

$$f(\mathbf{x}) =\approx f(0) + \sum_{i=1,N}\frac{\partial f(0)}{\partial x_i}x_i + \frac{1}{2}\sum_{i,j=1,N}\frac{\partial^2 f(0)}{\partial x_i \partial x_j}x_i x_j \tag{374}$$

che scriviamo in forma vettoriale come:

$$f(\mathbf{x}) = c - \mathbf{b}\cdot\mathbf{x} + \frac{1}{2}\mathbf{x}\cdot\hat{A}\mathbf{x} \tag{375}$$

con:

$$\mathbf{b} = -\nabla f(0) \tag{376}$$

$$A_{ij} = \frac{\partial^2 f(0)}{\partial x_i \partial x_j} \tag{377}$$

Supponiamo di minimizzare lungo $\mathbf{u}$ fino al punto di minimo $\mathbf{x}_u$ in quel punto il gradiente di f sarà ortogonale a $\mathbf{u}$. Vogliamo ora trovare la direzione di minimizzazione $\mathbf{v}$ tale che spostandomi da $\mathbf{x}_u$ lungo $\mathbf{v}$ il gradiente di f non acquisti componenti lungo $\mathbf{u}$. Tale gradiente lo scrivo come:

$$\nabla f(\mathbf{x}_u + \epsilon\mathbf{v}) = -\mathbf{b} + \hat{A}(\mathbf{x}_u + \epsilon\mathbf{v}) \tag{378}$$

quindi richiediamo che

$$\mathbf{u} \cdot (-\mathbf{b} + \hat{A}(\mathbf{x}_u + \epsilon\mathbf{v})) = 0 \tag{379}$$

Siccome la relazione

$$\mathbf{u} \cdot (-\mathbf{b} + \hat{A}(\mathbf{x}_u)) = 0 \tag{380}$$

è soddisfatta per dalla condizione di minimo di $\mathbf{x}_u$ allora la direzione $\mathbf{v}$ deve soddisfare la relazione:

$$\mathbf{u} \cdot \hat{A}\mathbf{v} = 0 \tag{381}$$

Il metodo del gradiente coniugato genera una serie di direzioni $\mathbf{h}^i$ tutte coniugate tra di loro. Poniamo per la prima iterazione:

$$\mathbf{g}^0 = \mathbf{h}^0 \tag{382}$$

e per le iterazioni successive usiamo la regola:

$$\mathbf{g}^{i+1} = \mathbf{g}^i - \lambda_i \hat{A}\mathbf{h}^i \tag{383}$$

$$\mathbf{h}^{i+1} = \mathbf{g}^{i+1} + \gamma_i \mathbf{h}^i \tag{384}$$

con:

$$\lambda_i = \frac{\mathbf{g}^i \cdot \mathbf{g}^i}{\mathbf{h}^i \hat{A} \mathbf{h}^i} = \frac{\mathbf{g}^i \cdot \mathbf{h}^i}{\mathbf{h}^i \hat{A} \mathbf{h}^i} \tag{385}$$

$$\gamma_i = \frac{\mathbf{g}^{i+1} \cdot \mathbf{g}^{i+1}}{\mathbf{g}^i \cdot \mathbf{g}^i} \tag{386}$$

Si può mostrare che questo assicura che per $j < i$

$$\mathbf{g}^i \cdot \mathbf{g}^j = 0 \tag{387}$$

$$\mathbf{h}^i \hat{A} \mathbf{h}^j = 0 \tag{388}$$

$$\mathbf{g}^i \cdot \mathbf{h}^j = 0 \tag{389}$$

Ora ci poniamo il problema che la matrice $\hat{A}$ non è conosciuta. Possiamo usare il seguente teorema: Se per un i ho che $\mathbf{g}^i = -\nabla f(\mathbf{x}^i)$ e minimizzando lungo $\mathbf{h}^i$ trovo il minimo di f nel punto $\mathbf{x}^{i+1}$ allora risulta $\mathbf{g}^{i+1} = -\nabla f(\mathbf{x}^{i+1})$.

Dimostrazione: Abbiamo:

$$\mathbf{g}^i = \mathbf{b} - \hat{A}\mathbf{x}^i \tag{390}$$

$$\mathbf{g}^{i+1} = \mathbf{b} - \hat{A}(\mathbf{x}^i + \lambda\mathbf{h}^i) = \mathbf{g}^i - \lambda\hat{A}\mathbf{h}^i \tag{391}$$

con λ determinato dalla relazione di minimo $\mathbf{g}^{i+1} \cdot \mathbf{h}^i = 0$ ossia $0 = \mathbf{g}^i \cdot \mathbf{h}^i - \lambda \mathbf{h}^i \cdot \hat{A}\mathbf{h}^i$ da cui si ritrova:

$$\lambda = \frac{\mathbf{g}^i \cdot \mathbf{h}^i}{\mathbf{h}^i \cdot \hat{A}\mathbf{h}^i} \tag{392}$$

Come si voleva dimostrare.

La scelta di γ in Eq. 385 prende il nome di Fletcher-Reeves. Spesso viene preferita la variante di Polak-Ribiere

$$\gamma_i = \frac{(\mathbf{g}^{i+1} - \mathbf{g}^i) \cdot \mathbf{g}^{i+1}}{\mathbf{g}^i \cdot \mathbf{g}^i} \tag{393}$$

che risulta più stabile.

35 Dynamic relaxation

Introduciamo un sistema dinamico fittizio associando ad ogni parametro λ_i un' energia cinetica fittizia $\frac{1}{2}\left(\frac{d\lambda_i}{dt}\right)^2$ e una forza frenante (frizione) $-f(t)\frac{d\lambda_i}{dt}$ quindi andremo a propagare le equazioni del moto fittizie:

$$\begin{cases} \frac{d^2\lambda_1(t)}{dt^2} & = -f(t)\frac{d\lambda_1}{dt} - \frac{\partial F(\lambda_1,\ldots,\lambda_N)}{\partial \lambda_1} \\ \ldots & \ldots \\ \frac{d^2\lambda_N(t)}{dt^2} & = -f(t)\frac{d\lambda_1}{dt} - \frac{\partial F(\lambda_1,\ldots,\lambda_N)}{\partial \lambda_1} \end{cases} \tag{394}$$

abbassando il termine di frizione fino al suo annullarsi in modo da trovare un punto di equilibrio in cui i parametri restano costanti nel tempo. Tale punto è un punto di minimo perché avremo $\frac{\partial F\left(\lambda_1^i,\ldots,\lambda_N^i\right)}{\partial \lambda_i} = 0$ per tutti gli i.

36 Simulated annealing

L'idea è quella di considerare il sistema dinamico fittizio dal punto di vista termodinamico. Consideriamo una distribuzione canonica per i parametri $\lambda_1,\ldots,\lambda_N$:

$$p(\lambda_1,\ldots\lambda_N) = \frac{1}{Z}e^{-\frac{F(\lambda_1,\ldots,\lambda_N)}{T}} \tag{395}$$

dove T è una temperatura fittizia e Z è una costante di normalizzazione che non conosciamo. L'idea è quella di usare un algoritmo come quello di Metropolis per campionare lo spazio dei parametri ad una certa temperatura che verrà poi abbassata in maniera da portare il sistema nel suo stato di energia (o meglio di F) più basso.

Parte IX
Machine Learing

37 Il problema della classificazione

Supponiamo di avere dei campioni (*samples*) che appartengono a classi diverse. Ogni classe è contraddistinta da un nome o etichetta (*label*). I campioni sono definite come un insieme di dati numerici (*features*). Un esempio famoso è la classificazione dei fiori Iris. Le features sono le lunghezze e larghezze di petalo e sepalo e le classi sono le specie di Iris (Viriginica, Versicor,Setosa). Il database contiene dati di 150 Iris 50 per specie. In generale per ogni sample i le features sono date dal vettore reale $\mathbf{x}^i$, il vettore ha dimensione N con N numero delle features, e la classe è data dal numero intero y^i. Ci poniamo il problema di creare un algoritmo capace di predire l'appartenenza di un campione ad una certa classe dopo essere stato *addestrato* su un certo numero di campioni detto insieme di addestramento. Per capire la qualità di un classficatore vado a vedere come si comporta su un insieme di campioni, detto di test, diverso da quello di addestramento.

38 Il Perceptron

Nel 1957 è stato proposto il primo classificatore che ha la struttura di un neurone artificiale. Per ogni feature $j = 1, N$ viene introdotto un peso w_j. Il percettrone può essere visto come una funzione di $\mathbf{x}$ dipendente dai parametri $\mathbf{w}$, detti pesi, e da un ulteriore parametro reale θ che restituisce 1 se il campione appartiene ad una certa classe e -1 in caso contrario. Tale funzione viene scritta come:

$$\tilde{\sigma}(z) = \begin{cases} 1 \text{ z} > \theta \\ -1 \text{ z} \leq \theta \end{cases} \tag{396}$$

con

$$z = \sum_{i=1,N} w_i x_i \tag{397}$$

Di solito si preferisce introdurre una feature aggiuntiva sempre paria 1: $x_0 = 1$ in tale maniera il parametro θ viene assorbito dal peso aggiuntivo w_0. L'algoritmo diventa

$$\sigma(z) = \begin{cases} 1 \text{ z} > \text{O} \\ -1 \text{ z} \leq 0 \end{cases} \tag{398}$$

con

$$z = \sum_{i=0,N} w_i x_i \tag{399}$$

dove σ è ora la funzione gradino. Rosenblatt introdusse un algoritmo (*learining rule*) per trovare i pesi $\mathbf{w}$ da un insieme di campioni di addestramento $\mathbf{x}^i$.

1. Parto con $\mathbf{w}$ vicino a 0
2. ciclo sui campioni i , calcolo il risultato attuale $\hat{y}^i = \sigma(\mathbf{w} \cdot \mathbf{x}^i)$
3. aggiorno i pesi secondo la regola $\mathbf{w} = \mathbf{w} + \eta(y^i - \hat{y}^i)\mathbf{x}^i$ con η di solito $\in [0, 1]$

Dopo l'iniziale entusiasmo ci si accorse che il Perceptron funziona solo per problemi che sono linearmente separabili ossia se esiste una iperpiano di dimensione $N - 1$ che divide i campioni delle due classi. Infatti nel caso $N = 2$ abbiamo che:

$$z = w_0 + w_1 x_1 + w_2 x_2 \tag{400}$$

è la condizione $z = 0$ è definita dalla retta:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} \tag{401}$$

Per problemi non separabili l'algoritmo di Rosenblatt non converge e diventa instabile.

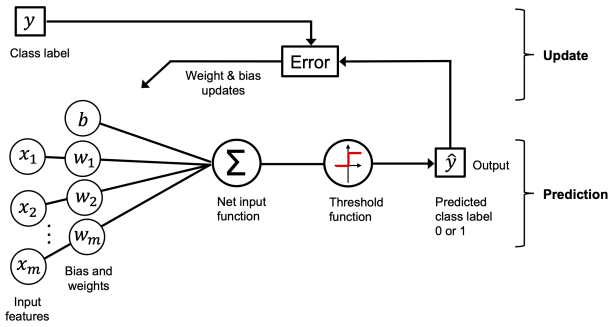


Figura 18: Schema di un perceptron

39 L'Adaline

L' *Adaptative linear neuron* introdotto nel 1960 permette la convergenza anche nel caso di problemi non completamente separabili inoltre presenta una struttura di neurone artificiale ripresa da modelli più recenti ed avanzati. Infatti viene sia introdotta una activation function $\phi(\mathbf{w}\cdot\mathbf{x})$ sia introdotta una funzione di errore $J(\mathbf{w}$ la minimizzazione di tale funzione permette di trovare i pesi. Poi il risultato del classificatore è dato,al solito, dalla funzione soglia $\sigma(\phi(\mathbf{w}\cdot\mathbf{x}))$

Nell'adaline la funzione di attivazione è semplicemente l'identita: $\phi(z) = z$ mentre la funzione di errore è definita come:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{1=1,N_{samples}} (y^i - \phi(\mathbf{w}\cdot\mathbf{x}^i))^2 \tag{402}$$

Per la minimizzazione devo calcolare il gradiente ∇J:

$$\frac{\partial J}{\partial w_j} = \sum_{1=1,N_{samples}} (y^i - \mathbf{w}\cdot\mathbf{x}^i)(-x_j^i) \tag{403}$$

Spesso come algoritmo di minimizzazione viene utilizzato lo steepest descent (semplice) ossia utilizzando la regola:

$$\mathbf{w} = \mathbf{w} - \eta\nabla J \tag{404}$$

con η chiamato *learning rate*.

Siccome spesso capita di avere a che fare con insiemi di addestramento molto numerosi viene sovente scelto l'algoritmo detto *stochastic gradient descent* in cui si sceglie in maniera casuale , a turno, un solo sample i e si aggiornano i pesi secondo la regola:

$$\mathbf{w} = \mathbf{w} + \eta(y^i - \mathbf{w}\cdot\mathbf{x}^i)\mathbf{x}^i \tag{405}$$

40 One versus all

Sia il Perceptron che l'Adaline forniscono l'informazione se un sample appartiene o meno ad una certa classe. Molte volte però ho a che fare con più di due classi. In tal caso per l'assegnazione si usa la strategia OVA (one versus all). Prima si addestrano tanti classificatori quante classi ci sono. Poi si assegna l'appartenenza ad una classe al classificatore che dà la funzione di attivazione $\phi(\mathbf{w}\cdot\mathbf{x})$ più elevata .

41 Standardizzazione

Succede che le features possono essere distribuite su scale di grandezza differenti. In tal caso è più efficiente rinormalizzarle in maniera che abbiano tutte lo stesso centro (che poniamo a 0) e la stessa deviazione standard che poniamo a 1. Se la feature j ha media u_j e devizione standard σ_j, usiamo la trasformazione:

$$x'_j = \frac{x_j - u_j}{\sigma_j} \tag{406}$$

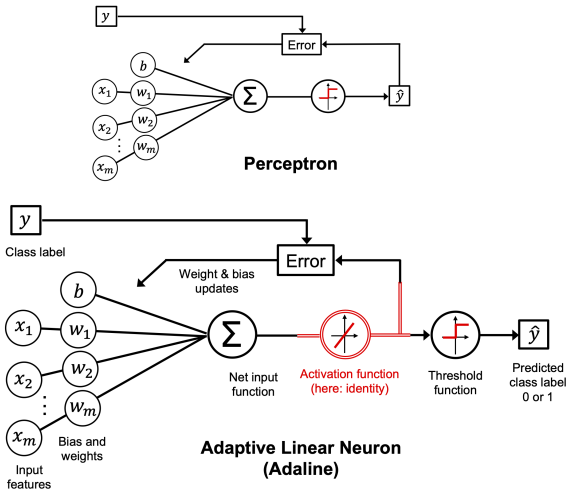


Figura 19: Schema di una adaline

42 Regressione Logistica

Talvolta oltre l'assegnazione ad una classe vogliamo anche che il nostro classificatore associ una probabilità a tale assegnazione. Questo viene usato ad esempio nelle previsioni meteoreologiche per la classe *oggi piove*.

Cominiciamo col mappare la probabilità $p \in [0,1]$ nell'intervallo $[-\infty, +\infty]$:

$$f(p) = \log(\frac{p}{1-p}) \tag{407}$$

Tale funzione prende il nome di funzione logistica *logit*.

La *logistic regression* è analoga alla Adelina ma con funzione di attivazione ϕ e funzione di errore J differenti. Richiediamo che $\phi(z)$ dia una probabilità:

$$\text{logit}(\phi(\mathbf{w} \cdot \mathbf{x})) = \mathbf{w} \cdot \mathbf{x} \tag{408}$$

Poniamo $z = \mathbf{w} \cdot \mathbf{x}$ e risolvendo troviamo:

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{409}$$

Tale funzione prende il nome di *sigmoidea*.

Dobbiamo ora fornire la funzione errore. Per prima cosa ridefiniamo le y^i in maniera che $\in 0, 1$ e usiamo una funzione a scalino:

$$\sigma'(p) = \begin{cases} 1 \; p > 0.5 \\ 0 \; p \leq 0.5 \end{cases} \tag{410}$$

Cominciamo con una prima funzione performance:

$$L(\mathbf{w}) = \prod_{i=1,N_{samples}} (\phi(z^i))^{y^i} (1-\phi(z^i))^{1-y^i} \tag{411}$$

dove la funzione nella produttoria per $y^i = 1$ è pari a $\phi(z^i)$ altrimenti è pari a $(1-\phi(z^i))$ tale funzione va massimizzata. Il problema è equivalente a massimizzare $\log(L)$. Scegliamo quindi la funzione errore da minimizzare:

$$J(\mathbf{w}) = -\log L(\mathbf{w}) \tag{412}$$

$$= \sum_{i=1,N_{samples}} (-y^i \log(\phi(z^i) - (1-y^i) \log(1-\phi(z^i)) \tag{413}$$

La minimizzazione richiede il calcolo di ∇J:

$$\frac{\partial J}{\partial w_j} = \sum_{i=1,N_{samples}} (-\frac{y^i}{\phi(z^i)} \frac{\partial \phi(z^i)}{\partial w_j} - \frac{1-y^i}{1-\phi(z^i)} (-\frac{\partial \phi(z^i)}{\partial w_j})) \tag{414}$$

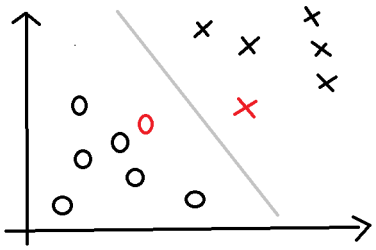


Figura 20: Samples di classe 1 (x), di classe -1 (o) e support vectors (rosso)

dove:

$$\frac{\partial \phi(z^i)}{\partial w_j} = \frac{\partial}{\partial w_j}(\frac{1}{1+e^{-z^i}}) = -\frac{1}{(1+e^{-z^i})^2}(-e^{-z^i})x_j^i \tag{415}$$

$$= \frac{1}{(1+e^{-z^i})}\frac{e^{-z^i}+1-1}{(1+e^{-z^i})}x_j^i \tag{416}$$

$$= \frac{1}{(1+e^{-z^i})}(1-\frac{1}{1+e^{-z^i}})x_j^i \tag{417}$$

$$= \phi(z^i)(1-\phi(z^i))x_j^i \tag{418}$$

Inserendo troviamo:

$$\frac{\partial J}{\partial w_j} = \sum_{i=1,N_{samples}} (-y^i + \phi(z^i))x_j^i \tag{419}$$

43 Regolarizzazione

Un inconveniente tipico della classificazione è il problema del *overfitting*. Questo è di solito accompagnato,se si usano features rinormalizzate, da una distribuzione irregolare dei pesi $\mathbf{w}$. Per alleviare tale problema si regolarizza la funzione errore, dato $C > 0$:

$$J(\mathbf{w}) = CJ(\mathbf{w}) + \frac{1}{2}\mathbf{w}\cdot\mathbf{w} \tag{420}$$

il termine $\mathbf{w}\cdot\mathbf{w}$ è minimo per una distribuzione uniforme dei pesi. Facendo diventare C piccolo si induce uniformità in $\mathbf{w}$.

44 Support Vector Machine

Supponiamo, inizialmente, di lavorare con un problema linearmente separabile. Useremo la notazione vettoriale per rappresentare samples $\mathbf{x}^i$ nello spazio delle features. Non includeremo però la dimensione 0-esima. Le Support Vector Machines forniscono una classificazione $-1, 1$. L'idea è di trovare un iper-piano nello spazio delle features che separi le soluzioni positive 1 da quelle negative -1. Se il problema è separabile tale iperpiano esiste ed inoltre ci saranno (praticamente sicuramente) più piani con tale proprietà. Dobbiamo quindi scegliere l'iperpiano migliore. Vogliamo massimizzare la distanza tra l'iperpiano e i samples dell'insieme di addestramento più vicini ad esso. Tali samples vengono chiamati support vectors (vedi Fig. 20)

Chiamiamo $\mathbf{x}^{pos}$ il support vector della classe $+1$ e $\mathbf{x}^{neg}$ il support vector della classe -1.Consideriamo la seguente funzione:

$$\phi(\mathbf{x}) = w_0 + \mathbf{w}\cdot\mathbf{x} \tag{421}$$

e scegliamo i pesi in maniera tale che $\phi \geq 1$ nella parte dello spazio a cui appartengo i samples positivi e $\phi \leq -1$ nella parte dello spazio a cui appartengo i samples negativi. Chiamiamo margini gli iperpiani $\phi(\mathbf{x})) = 1$ e $\phi(\mathbf{x})) = 1$ e scegliamo

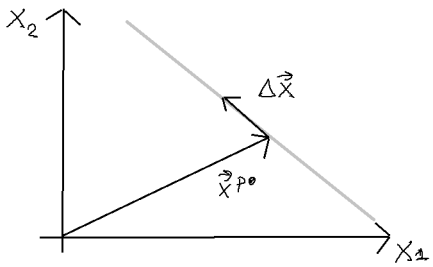


Figura 21: Margine passante per $\mathbf{x}^{pos}$

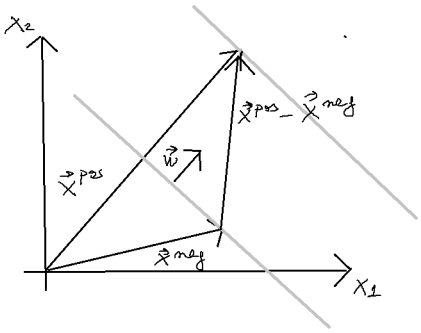


Figura 22: Distanza tra i margini

i pesi in maniera che i support vectors stiano sui margini:

$$w_0 + \mathbf{w} \cdot \mathbf{x}^{pos} = 1 \tag{422}$$

$$w_0 + \mathbf{w} \cdot \mathbf{x}^{neq} = -1 \tag{423}$$

$$https://www.overleaf.com/project/651ea46cf149d0d6b0350bb3 \tag{424}$$

Dimostriamo ora che $\mathbf{w}$ è perpendicolare ai margini. Infatti se prendiamo $\mathbf{x}^{pos} + \Delta\mathbf{x}$ appartenente al margine di o $\mathbf{x}^{pos}$ (vedi Fig. 21) risulta:

$$w_0 + \mathbf{w} \cdot (\mathbf{x}^{pos} + \Delta\mathbf{x}) = 0 \tag{425}$$

$$(w_0 + \mathbf{w} \cdot \mathbf{x}^{pos}) + \mathbf{w} \cdot \Delta\mathbf{x} = 0 \tag{426}$$

$$\mathbf{w} \cdot \Delta\mathbf{x} = 0 \tag{427}$$

Il nostro obbiettivo è massimizzare la distanza d tra i due margini. Essa può essere facilmente calcolata (vedi Fig. 22):

$$d = (\mathbf{x}^{pos} - \mathbf{x}^{neg}) \cdot \left(\frac{\mathbf{w}}{|\mathbf{w}|}\right) \tag{428}$$

$$= (\mathbf{x}^{pos} + w_0 - (\mathbf{x}^{neg} - w_0)) \cdot \left(\frac{\mathbf{w}}{|\mathbf{w}|}\right) \tag{429}$$

$$= \frac{2}{|\mathbf{w}|} \tag{430}$$

Si preferisce però minimizzare la funzione costo o errore:

$$J(\mathbf{w}, w_0) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} \tag{431}$$

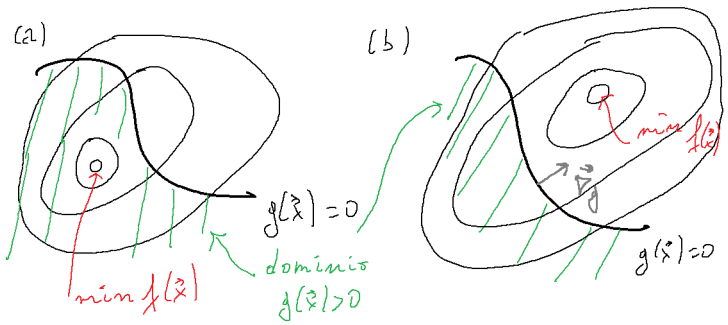


Figura 23: Minimizzazione di una funzione soggetta a condizione di diseguaglianza

J va minimizzata con la condizione che i samples siano classificati correttamente ossia con l'insieme di condizioni:

$$y^i(w_0 + \mathbf{w} \cdot \mathbf{x}^i) \geq 1 \tag{432}$$

con $i = 1, N_{samples}$ Notiamo che sovente non si riescono a separare linearmente in maniera completa le due classi di samples. Per estendere la SVM a questo caso introduciamo un insieme di valori soglia $\xi^i \geq 0$ per $i = 1, N_{samples}$ e usiamo condizioni *rilassate*:

$$\begin{cases} w_0 + \mathbf{w} \cdot \mathbf{x}^i \geq 1 - \xi^i \ \text{ per } \ y^i = 1 \\ w_0 + \mathbf{w} \cdot \mathbf{x}^i \leq -1 + \xi^i \ \text{ per } \ y^i = -1 \end{cases} \tag{433}$$

Poi minimizziamo la seguente funzione costo:

$$J(\mathbf{w}, w_0, \xi^1, .., \xi^{N_{samples}}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \left(\sum_{i=1, N_{samples}} \xi^i \right) \tag{434}$$

con C parametro fissato. Tale funzione va minimizzata sotto le condizioni di Eq. 433.

45 Moltiplicatori di Lagrange per condizioni di disuguaglianza

Supponiamo di cercare il minimo di $f(\mathbf{x})$ con la condizione $g(\mathbf{x}) \geq 0$. Abbiamo due casi.

(I) Nel primo il minimo di f sta nella parte di spazio dove $g > 0$ (vedi Fig. 23 (a)). In tal caso il minimo è lo stesso con o senza condizioni.

(II) Altrimenti il minimo deve trovarsi sul luogo dei punti (iper-superficie) $g(\mathbf{x}) = 0$ (vedi Fig. 23 (b)). Infatti se per assurdo il minimo si trovasse in un punto $\mathbf{x}'$ con $g(\mathbf{x}) > 0$ allora potrei muovermi lungo la direzione $\nabla f(\mathbf{x}')$ ed il valore di f diminuirebbe. Inoltre abbiamo che sulla iper-superficie $g(\mathbf{x}) = 0$ $\nabla g(\mathbf{x})$ è localmente ortogonale ad essa. Nel punto di minimo anche $\nabla f(\mathbf{x})$ è ortogonale a tale iper-superficie . Inoltre $\nabla g(\mathbf{x}) \parallel \nabla f(\mathbf{x})$ quindi posso scrivere $\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$ con $\lambda > 0$.

i due casi possono essere raggruppati insieme tramite la ricerca del minimo di:

$$\min_{\mathbf{x}} \max_{\lambda} \ f(\mathbf{x}) - \lambda g(\mathbf{x}) \quad \text{con} \quad \lambda \geq 0 \tag{435}$$

Infatti nella parte di spazio $g(\mathbf{x} > 0$ avremo che $\lambda = 0$ (caso (I)); altrimenti la soluzione $\mathbf{x}'$ soddisferà $\nabla(f(\mathbf{x}') - \lambda g(\mathbf{x}')) = 0$ e $g(\mathbf{x}'=0$. Pertanto $\mathbf{x}'$ starà sulla ipersuperficie $g(\mathbf{x}) = 0$. Notiamo che nel caso (I), in cui il minimo non sta sul bordo, l'equazione $\nabla(f(\mathbf{x}') - \lambda g(\mathbf{x}')) = 0$ è soddisfatta solo per $\lambda < 0$.

46 Lagrangian duality

Ritrattiamo ora il problema della minimizzazione di una funzione $f(\mathbf{x})$ sottoposta al vincolo $g(\mathbf{x}) \geq 0$. Chiamiamo p^* la soluzione:

$$p^* = \min_{\mathbf{x}} f(\mathbf{x}) \quad \text{con} \quad g(\mathbf{x}) \geq 0 \tag{436}$$

definiamo:

$$J(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & g(\mathbf{x}) \geq 0 \\ +\infty & g(\mathbf{x}) < 0 \end{cases} \tag{437}$$

allora:

$$p^* = \min_{\mathbf{x}} J(\mathbf{x}) \tag{438}$$

Definiamo:

$$H(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x}) \tag{439}$$

allora:

$$\max_{\lambda} H(\mathbf{x}, \lambda) = \begin{cases} f(\mathbf{x}) & g(\mathbf{x}) \geq 0 \\ +\infty & g(\mathbf{x}) < 0 \end{cases} \tag{440}$$

$$= J(\mathbf{x}) \tag{441}$$

Quindi:

$$p^* = \min_{\mathbf{x}} \max_{\lambda} H(\mathbf{x}, \lambda) \tag{442}$$

Proviamo ad invertire l'ordine di max e min:

$$H(\mathbf{x}, \lambda) \leq J(\mathbf{x}) \tag{443}$$

$$\min_{\mathbf{x}} H(\mathbf{x}, \lambda) \leq \min_{\mathbf{x}} J(\mathbf{x}) \tag{444}$$

$$\min_{\mathbf{x}} H(\mathbf{x}, \lambda) \leq p^* \tag{445}$$

$$\max_{\lambda} \min_{\mathbf{x}} H(\mathbf{x}, \lambda) \leq p^* \tag{446}$$

Allora $d^* = \max_\lambda \min_{\mathbf{x}} H(\mathbf{x}, \lambda)$ è un limite inferiore per p^*. Se $d^* = p^p$ si parla di *strong duality*, in tal caso il vincolo è strettamente soddisfatto $g(\mathbf{x}_{min}) > 0$ (caso (I) della sezione precedente). Altrimenti si parla di *weak duality*.

47 Minimizzazione della funzione costo di una SVM

Consideriamo la minimizzazione della funzione costo J di Eq. 431 soggetta ai vincoli di Eq. 432. Introduciamo i moltiplicatori di Lagrange λ_i per $i = 1, N_{samples}$ e la funzione:

$$J'(\mathbf{w}, w_0, \lambda_1, .., \lambda_{N_{samples}}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1,N_{samples}} \lambda_i y^i (w_0 + \mathbf{w} \cdot \mathbf{x}^i - 1) \tag{447}$$

Dobbiamo prima massimizzare J' rispetto alle λ_i con la condizione $\lambda_i \geq 0$ e poi dobbiamo minimizzare rispetto a $\mathbf{w}$ e w_0. Supponiamo strong duality e invertiamo le due operazioni. Consideriamo le condizione di minimo rispetto a $\mathbf{w}$ e w_0:

$$\frac{\partial J'}{\partial \mathbf{w}} = 0 \tag{448}$$

$$\mathbf{w} - \sum_{i=1,N_{samples}} \lambda_i y^i \mathbf{x}^i = 0 \tag{449}$$

e

$$\frac{\partial J'}{\partial w_0} = 0 \tag{450}$$

$$- \sum_{i=1,N_{samples}} \lambda_i y^i = 0 \tag{451}$$

Da cui ricaviamo:

$$\mathbf{w} = \sum_{i=1,N_{samples}} \lambda_i y^i \mathbf{x}^i \tag{452}$$

$$\sum_{i=1,N_{samples}} \lambda_i y^i w_0 = 0 \tag{453}$$

$$\tag{454}$$

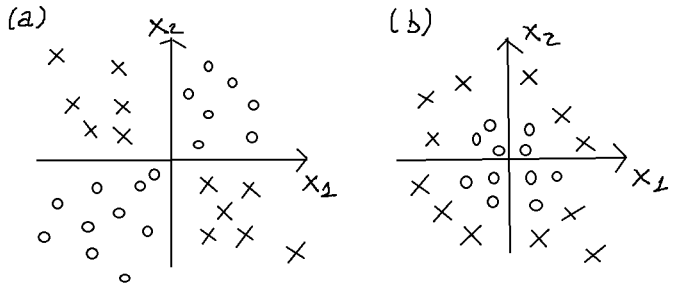


Figura 24: Due esempi di distribuzioni non classificabili: XOR (a) e circolare (b)

che sostituiamo in Eq. 447:

$$J''(\lambda_1, .., \lambda_{N_{samples}}) = \frac{1}{2} \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j \mathbf{x}^i \mathbf{x}^j - \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j \mathbf{x}^i \mathbf{x}^j \tag{455}$$

$$= -\frac{1}{2} \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j \mathbf{x}^i \mathbf{x}^j \tag{456}$$

$$\tag{457}$$

J'' va ora massimizzata rispetto alle λ_i con i vincoli $\lambda_i \geq 0$.

48 Estensione a problemi non linearmente separabili

In. Fig.24 mostriamo due esempi di distribuzioni non classificabili linearmente. Ci accorgiamo che possiamo scegliere delle trasformazioni univoche $\phi : \mathbf{x} \to \mathbf{x}'$ che trasformano lo spazio delle features in uno spazio di dimensione maggiore. Esistono delle trasformazioni particolari per cui la distribuzione trasformata risulta linearmente separabile. Ad esempio per la trasformazione XOR (panello (a)) una tale trasformazione è:

$$(x_1, x_2) \to (x_1, x_2, x_1 x_2) \tag{458}$$

mentre per la trasformazione circolare (panello (b)) una tale trasformazione è:

$$(x_1, x_2) \to (x_1, x_2, x_1^2 + x_2^2) \tag{459}$$

Nello spazio di dimensione maggiore cerchiamo il minimo della funzione:

$$J = \frac{1}{2} \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j \mathbf{x}^{i\prime} \mathbf{x}^{j\prime} \tag{460}$$

dove compaiono i vettori delle features trasformati. Posso scrivere:

$$J = \frac{1}{2} \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j \phi(\mathbf{x}^i) \cdot \phi(\mathbf{x}^j) \tag{461}$$

Chiamiamo kernel il seguente operatore:

$$K(\mathbf{x^i}, \mathbf{x^j}) = \phi(\mathbf{x}^i) \cdot \phi(\mathbf{x}^j) \tag{462}$$

allora:

$$J = \frac{1}{2} \sum_{i,j=1,N_{samples}} \lambda_i \lambda_j K(\mathbf{x^i}, \mathbf{x^j}) \tag{463}$$

In genere K è una funzione complicata. Possiamo però rimpiazzarlo con delle forme semplificate. Una molto usata prende il nome di *radial basis function*:

$$K(\mathbf{x^i}, \mathbf{x^j}) = e^{-\gamma |\mathbf{x}^i - \mathbf{x}^j|^2} \tag{464}$$

Si vede che tale kernel riesce a separare in maniera efficente le distribuzione (a) e (b).

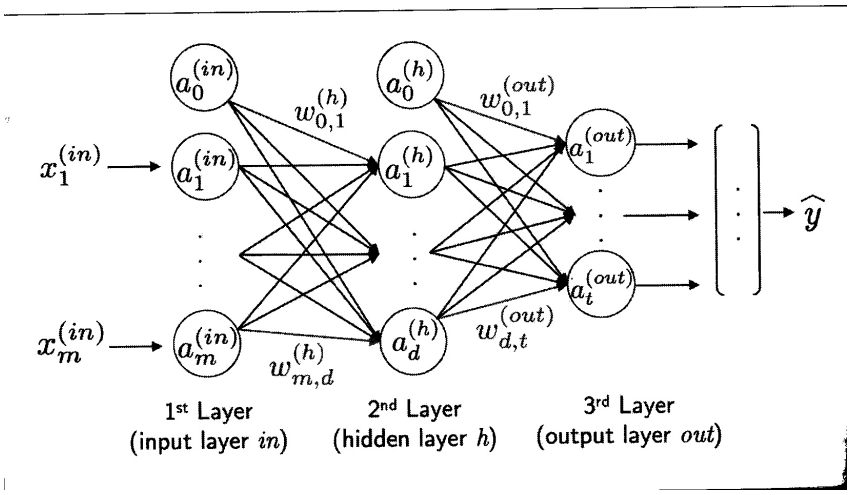


Figura 25:

Schema di una rete neurale

49 Reti Neurali

Per superare i limiti delle macchine a singolo neurone artificiale che possono risolvere solo problemi linearmente separabili sono state introdotte reti di neuroni artificiali a più strati (*layers*).In Fig. 25 i singoli neuroni artificiali sono indicati con i cerchi. Consideriamo m features di input $x_1^{(in)}, .., x_1^{(in)}$ e t possibili classificazioni $y_1, .., y_t$. Ogni neurone ha la struttura che abbiamo visto per l'adaline e la logistic regression.Qui ci limitiamo al caso della logistic regression. In questo caso l'activation function ϕ è la funzione sigmoide e σ la funzione soglia finale. Inoltre ci limitiamo ad un solo layer nascosto (*hidden*) che consta di d neuroni. La funziona nel seguente modo:

1. Copia $a_i^{(in)} = x_i^{(in)}$ per $i = 1, m$ e metti $a_0^{(in)} = 1$
2. Calcolo $z_i^{(h)} = \sum_{j=0,m} W_{ij}^{(h)} a_j^{(in)}$ per $i = 1, d$
3. Calcolo $a_i^{(h)} = \phi(z_i^{(h)})$ e metti $a_0^{(h)} = 1$
4. Calcola $z_i^{(out)} = \sum_{j=0,d} W_{ij}^{(out)} a_j^{(in)}$ per $i = 1, t$
5. Calcolo $a_i^{(out)} = \phi(z_i^{(h)})$ per $i = 1, t$
6. Trovo i_m tale che $a_{i_m}^{(out)}$ sia massimo
7. pongo $y_i = 1$ per $i = i_m$, $y_i = 0$ altrimenti

Le due matrice dei pesi $\hat{W}^{(h)}$ e $\hat{W}^{(out)}$ vanno trovate minimizzando la seguente funzione errore:

$$J(\hat{W}^{(h)}, \hat{W}^{(out)}) = -\sum_{n=1,N_{samples}} \sum_{i=1,t} y_i^n \log(a_i^{(out),n}) + (1 - y_i^n) \log(1 - a_i^{(out),n}) \tag{465}$$

con n che scorre sui campioni di addestramento. Per minimizzare J dobbiamo calcolare i gradienti rispetto ai pesi. Cominciamo con la derivata della funzione sigmoide:

$$\frac{d\phi(z)}{dz} = \frac{d}{dz}\frac{1}{1+e^{-z}} \tag{466}$$

$$= \frac{1}{(1+e^{-z})^2} e^{-z} \tag{467}$$

$$= \frac{1}{1+e^{-x}} \frac{e^{-z}+1-1}{1+e^{-z}} = \phi(z)(1-\phi(z)) \tag{468}$$

Ora calcoliamo:

$$\frac{\partial J}{\partial W_{ij}^{(out)}} = -\sum_n \left[\frac{y_i^n}{a_i^{(out),n}} a_i^{(out),n}(1-a_i^{(out),n}) \frac{\partial z_i^{(out),n}}{\partial W_{ij}^{(out)}} + \frac{1-y_i^n}{1-a_i^{(out),n}}(-1) a_i^{(out),n}(1-a_i^{(out),n}) \frac{\partial z_i^{(out),n}}{\partial W_{ij}^{(out)}} \right] \tag{469}$$

$$= -\sum_n (y_i^n - a_i^{(out),n}) a_j^{(h),n} \tag{470}$$

$$= -\sum_n \delta_i^{(err),n} a_j^{(h)} \tag{471}$$

dove abbiamo chiamato $\delta_i^{(err),n} = (y_i^n - a_i^{(out),n})$ l'errore relativo al termine i-esimo del layer di uscita.

Calcoliamo ora la parte di gradiente di J relativa ai pesi del layer interno:

$$\frac{\partial J}{\partial W_{ij}^{(h)}} = -\sum_n \sum_{k=1,t} \left[\frac{y_k^n}{a_k^{(out),n}} a_k^{(out),n}(1-a_k^{(out),n}) \frac{\partial z_k^{(out),n}}{\partial W_{ij}^{(h)}} + \frac{1-y_k^n}{1-a_k^{(out),n}}(-1) a_k^{(out),n}(1-a_k^{(out),n}) \frac{\partial z_k^{(out),n}}{\partial W_{ij}^{(h)}} \right] \tag{472}$$

Abbiamo che

$$\frac{\partial z_k^{(out),n}}{\partial W_{ij}^{(h)}} = \sum_{l=0,d} \frac{\partial W_{kl}^{(out)} a_d^{(h),n}}{\partial W_{ij}^{(h)}} \tag{473}$$

$$= W_{kl}^{(out)} \frac{\partial a_d^{(h),n}}{\partial W_{ij}^{(h)}} \tag{474}$$

$$= W_{kl}^{(out)} a_i^{(h),n}(1-a_i^{(h),n}) a_j^{(in),n} \tag{475}$$

Sostituendo, troviamo:

$$\frac{\partial J}{\partial W_{ij}^{(h)}} = \sum_n \sum_{k=1,t} (y_k^n - a_k^{(out),n}) W_{ki}^{(out)} a_i^{(h),n}(1-a_i^{(h),n}) a_j^{(in),n} \tag{476}$$

Se aggiungiamo più layers nascosti possiamo trovare i gradienti iterando il procedimento riportato sopra. In pratica ogni layer nascosto comporterà un termine del tipo $Wa(1-a)$. Questo algoritmo prende il nome di *backward propagation*.