Langevin-Parisi 2D

JAVIER ZAMBRANO ZAMBRANO

LUNA MAY LÓPEZ GONZÁLEZ

INDICE

INTRODUZIONE

Paul Langevin

Paul Langevin (Parigi, 23/01/1872 - 19/12/1946) è stato un fisico francese, noto per la sua teoria del magnetismo e per l'organizzazione del Congresso Solvay.

Una delle sue contribuzioni più importanti in fisica è la sua teoria del magnetismo.

Useremo l'equazione di Langevin, un'equazione differenziale stocastica che descrive l'evoluzione di un sistema soggetto a una combinazione di forzanti deterministiche e casuali

Giorgio Parisi

Giorgio Parisi (Roma, 04/08/1948) è un influente fisico italiano, noto per i suoi studi sulla meccanica statistica e la teoria quantistica dei campi.

È stato uno dei vincitori del Premio Nobel per la Fisica nel 2021.

Parisi presenta un nuovo algoritmo che consente il calcolo diretto delle funzioni di correlazione connesse per sistemi continui, l'algoritmo è basato sull'**equazione di Langevin**

Obiettivi:

- Spiegare tutti le equazioni matematiche che saranno utilizzate
- Analizzare dettagliatamente tutti i programmi e le sue strutture
- Vedere un esempio pratico dei programmi LangPar2D e schr2D
- Confrontare i risultati e analizzare le grafiche
- Conclusione

Consideriamo l'Hamiltoniano $H[\phi], \phi_i (i = 1, N)$ N variabili continui). Il valore atteso di equilibrio della funzione è dato da:

$$\langle f[\phi] \rangle = \frac{\int d[\phi] \exp(-H[\phi]) f[\phi]}{\int d[\phi] \exp(-H[\phi])}$$

Dove imponiamo che $\beta = (kT)^{-1} = 1$

Una forma pratica di calcolare $\langle f[\phi] \rangle$ è basata nell'equazione di Langevin:

$$\dot{\phi}_i = -\frac{\partial H}{\partial \phi_i} + \eta_i(t), \qquad \langle \eta_i(t) \eta_j(t) \rangle = 2\delta_{ij}\delta(t - t') \tag{1}$$

$\phi_i(t)$ una funzione del "tempo" t e $\eta_i(t)$ è una variabile gaussiana aleatoria.

Infatti, abbiamo:

$$\langle f[\phi]\rangle = \lim_{\tau\to\infty}\frac{1}{\tau}\int_0^{\tau} dt\, f[\phi(t)]$$

$\phi(t)$ η (2)

dove è la soluzione dell'equazione (1) per una scelta aleatoria

Eqs (1), (2) sono gli strumenti per calcolare il valore atteso statistico, alternativo al metodi di Monte Carlo standard.

Prima di risolvere l'eq $$\phi_i^{k+1} = -\varepsilon\frac{\delta H}{\partial\phi_i}[\phi^k] + \sqrt{2\varepsilon}R_i^k$$

$t = \varepsilon k$ R_i^k

dove abbiamo fissato e $\langle R_i^k R_{i\prime}^{k\prime}\rangle = \delta^{kk\prime}\delta_{ii\prime}$ rie gaussiane:

Il tempo misurato è proporzionale a ε^{-1} e gli errori nel risultato finale sono proporzionale a ε. Se sono necessari risultati accurati, questo metodo è più lento che il metodo di Monte Carlo: l'equazione del tempo discreto di Langevin è molto simile al metodo di Monte Carlo con piccoli passi proporzionali a $\varepsilon^{1/2}$:

Del teorema di fluttuazione-dissipazione sappiamo che:

$$\langle \phi_i \phi_j \rangle_c = \frac{d}{d\lambda} \langle \phi_i \rangle_\lambda |_{\lambda=0}$$

dove $\langle \phi \rangle_\lambda$ è misurato con l'hamiltoniana $H_\lambda = H - \lambda \phi_j$. Possiamo introdurre l'equazione di Langevin dipendente da λ

$$\dot{\phi}_i(t,\lambda) = -\frac{\partial H}{\partial \phi_i} + \lambda \delta_{ij} + \eta_i(t)$$

Sviluppando $\phi_i(t,\lambda)$ n potenze da $\lambda(\phi_i(t,\lambda) = \phi_i^{(0)} + \lambda\phi_i^{(1)} + \frac{1}{2}\lambda^2\phi_i^{(2)} + \cdots)$ I identificando i termini, otteniamo:

$$\dot{\phi}_i^{(0)} = -\frac{\partial H}{\partial \phi_i} + \eta_i(t), \qquad \dot{\phi}_i^{(1)} = -\frac{\partial^2 H}{\partial \phi_i \partial \phi_1}\phi_1^{(1)} + \delta_{ij},$$

$$\dot{\phi}_i^{(1)} = -\frac{\partial^2 H}{\partial \phi_i \partial \phi_1}\phi_1^{(2)} - \frac{\partial^3 H}{\partial \phi_i \partial \phi_1 \partial \phi_n}\phi_1^{(1)}\phi_n^{(1)}$$

dove H e sua derivata sono calcolate solo come funzioni da $\phi^{(0)}$

Finalmente arriviamo a

$$\langle \phi_i \phi_j \rangle_c = \lim_{\tau\to\infty} \frac{1}{\tau}\int_0^\tau \phi_i^{(1)}(t)dt,$$

$$\langle \phi_i \phi_j \phi_j \rangle_c = \lim_{\tau\to\infty} \frac{1}{\tau}\int_0^\tau \phi_j^{(2)}(t)dt,$$

PROGRAMMA PRINCIPALE

Algoritmo di Parisi-Langevin per la quantizzazione stocastica del potenziale di Simon bidimensionale.

$$V(x,y) = y^2 x^2$$

Il codice realizza una simulazione stocastica utilizzando il metodo di Langevin per ottenere la correlazione <x(s)x(t)> dei coordinati x in funzione del tempo euclideo.

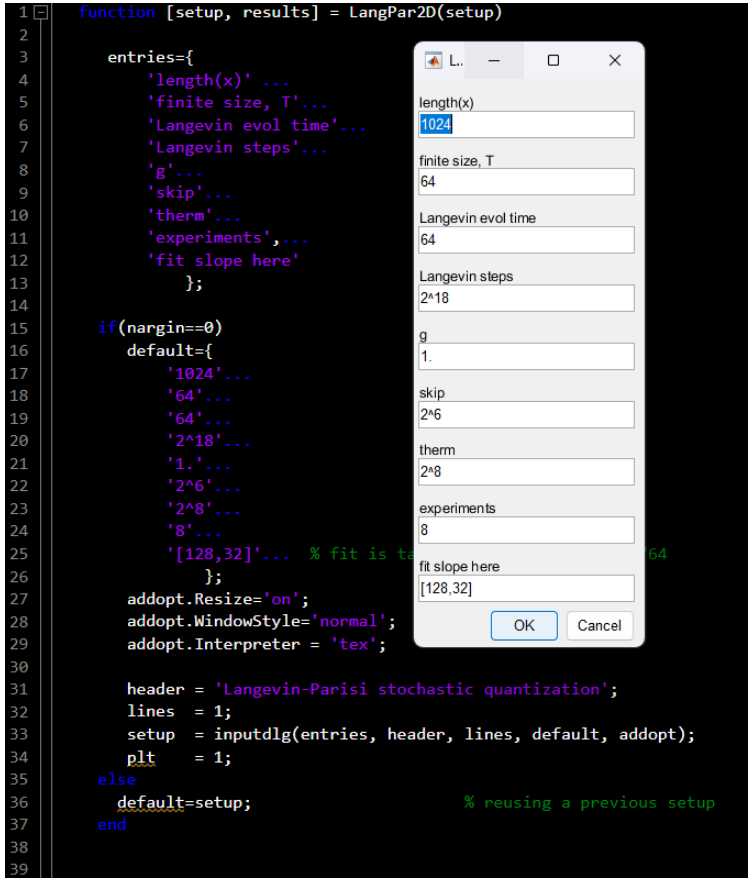
Funzione LangPar2D e variabili di entrata

Inizia la funzione principale

Variabili di entrata necessari per la implementazione del codice.

Scelta di valori predeterminati

L'obiettivo è aprire una finestra automaticamente quando si inizia il programma con i valori già introdotti, però è possibile modificarli manualmente



Variabili di entrata e configurazione iniziale

Dopo da introdurre i valori nella finestra, dobbiamo estrarre i parametri e salvarli.

hbar rappresenta la costante ridotta di Planck.

```
40      N     = str2num(setup{1});            % extract parameters
41      T     = str2num(setup{2});
42      Tlang = str2num(setup{3});
43      steps = str2num(setup{4});
44      g     = str2num(setup{5});
45      skip  = str2num(setup{6});
46      therm = str2num(setup{7});
47      exps  = str2num(setup{8});
48      trim  = str2num(setup{9});
49
50      hbar  = 1.0;
51
52      t  = linspace(0, T, N);
53      a  = T/N;             % Eucl.time lattice size
54
```

Differenza seconda discreta

Questo blocco stabilisce i condizioni iniziali e i parametri necessari per la simulazione di Langevin.

Quindi otterremo una discretizzazione dell'operatore Laplaciano con condizioni periodiche al contorno.

```
55      % diff.seconda discreta con condizioni periodiche al contorno
56      d = ones(N,1);
57      Delta = spdiags([-d, d],0:1,N,N);
58      Delta(1,N) = 1;
59      Delta = Delta+Delta';
60      Delta = Delta/a^2;
61      clear d
62
63      tau = Tlang/steps;            % Langevin time step
64      D = sqrt(2*tau*hbar/a);               % diffusion coefficient
65
66      Xcum = [];
67
68      src = 1;                              % location of source
69
70      tic                                   % start clock
71
```

Ciclo parfor (pararell for)

Questo ciclo permette l'esecuzione di cicli "for" in modo parallelo.

Dopo da iniziare abbiamo:

- *Linea 74*: mostra il numero dell'esperimento
- *Linea 76*: 4 vettori di lunghezza N riempiti con zeri
- *Linea 78*: Inizializza un vettore vuoto Xc

```
72      parfor r = 1:exps     % run "exps" independent experiments in parallel
73
74          disp(['Running ',num2str(r),'-th experiment'])
75
76          x0 = zeros(N,1); x1 = x0; y0 = x0; y1 = x0;
77
78          Xc = [];
79
```

Ciclo parfor (pararell for)

Questa parte implementa la fase di termalizzazione a partire da un ciclo.

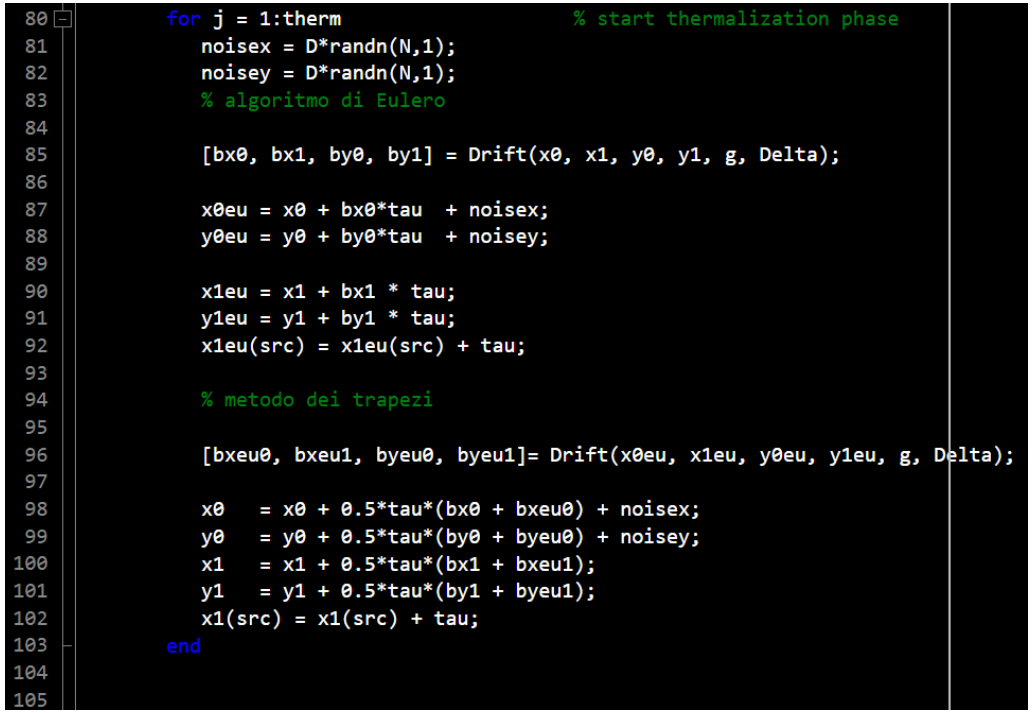
- *Linee 81 e 82*: Questo simula le fluttuazioni termiche nelle coordinate "*x*" e "*y*"
- *Linee 83-93*: Implementa l'algoritmo di Eulero per avanzare le posizioni "*x*" e "*y*" nel tempo "*tau*". Viene aggiunto il rumore stocastico e viene applicata una correzione alla posizione "*x1*" nella posizione "*src*"

```
80        for j = 1:therm                    % start thermalization phase
81           noisex = D*randn(N,1);
82           noisey = D*randn(N,1);
83           % algoritmo di Eulero
84
85           [bx0, bx1, by0, by1] = Drift(x0, x1, y0, y1, g, Delta);
86
87           x0eu = x0 + bx0*tau  + noisex;
88           y0eu = y0 + by0*tau  + noisey;
89
90           x1eu = x1 + bx1 * tau;
91           y1eu = y1 + by1 * tau;
92           x1eu(src) = x1eu(src) + tau;
93
94           % metodo dei trapezi
95
96           [bxeu0, bxeu1, byeu0, byeu1]= Drift(x0eu, x1eu, y0eu, y1eu, g, Delta);
97
98           x0   = x0 + 0.5*tau*(bx0 + bxeu0) + noisex;
99           y0   = y0 + 0.5*tau*(by0 + byeu0) + noisey;
100          x1   = x1 + 0.5*tau*(bx1 + bxeu1);
101          y1   = y1 + 0.5*tau*(by1 + byeu1);
102          x1(src) = x1(src) + tau;
103       end
104
105
```

Ciclo parfor (pararell for)

- *Linee 94-103*: Applica il metodo dei trapezi per aggiornare le posizioni "*x*" e "*y*" in modo più preciso.

Quindi, questo frammento simula l'evoluzione del sistema durante la fase di termalizzazione utilizzando l'algoritmo di Langevin. Questo approccio aiuta il sistema a raggiungere un equilibrio termico prima di effettuare misurazioni o esperimenti significativi.



Ciclo parfor (pararell for)

Questa parte del codice implementa la simulazione di un processo Langevin bidimensionale.

Possiamo spiegare passo dopo passo:

- *Primo ciclo*: Ogni iterazione simula il processo Langevin per un passo temporale.
- *Secondo ciclo interno*: è praticamente la stessa struttura che nella fase di termalizzazione (metodo di Eulero e dei trapezi), ma applicando il metodo di Langevin.

```
106        for j = 1:skip:steps              % start Langevin run
107           for m = 1:skip
108              noisex = D*randn(N,1);
109              noisey = D*randn(N,1);
110
111              [bx0, bx1, by0, by1]= Drift(x0, x1, y0, y1, g, Delta);
112
113              x0eu = x0 + bx0 * tau  + noisex;
114              y0eu = y0 + by0 * tau  + noisey;
115
116              x1eu = x1 + bx1 * tau;
117              y1eu = y1 + by1 * tau;
118              x1eu(src) = x1eu(src) + tau;
119
120
121              [bxeu0, bxeu1, byeu0, byeu1]= Drift(x0eu, x1eu, y0eu, y1eu, g, Delta);
122
123              x0   = x0 + 0.5*tau*(bx0 + bxeu0) + noisex;
124              y0   = y0 + 0.5*tau*(by0 + byeu0) + noisey;
125              x1   = x1 + 0.5*tau*(bx1 + bxeu1);
126              y1   = y1 + 0.5*tau*(by1 + byeu1);
127              x1(src) = x1(src) + tau;
128           end
129
130           Xc =[Xc, x1];          % correlazione < x(s).x(t) >
131        end
132
133        Xcorr = mean(Xc,2);
134        Xcum  = [Xcum, Xcorr];    % average on steps*exps/skip...
135     end
136
137     toc
138
```

Ciclo parfor (pararell for)

- *Aggiornamento e accumulo dei risultati*: Le coordinate "*x1*" vengono memorizzate in una matrice "*Xc*", che rappresenta la correlazione <x(s).x(t)> ad ogni passo del ciclo esterno.
  Dopo che il ciclo esterno è terminato, viene calcolata la media di "*Xc*" lungo la seconda dimensione (media su tutti gli esperimenti).

```
106        for j = 1:skip:steps              % start Langevin run
107           for m = 1:skip
108              noisex = D*randn(N,1);
109              noisey = D*randn(N,1);
110
111              [bx0, bx1, by0, by1]= Drift(x0, x1, y0, y1, g, Delta);
112
113              x0eu = x0 + bx0 * tau  + noisex;
114              y0eu = y0 + by0 * tau  + noisey;
115
116              x1eu = x1 + bx1 * tau;
117              y1eu = y1 + by1 * tau;
118              x1eu(src) = x1eu(src) + tau;
119
120
121              [bxeu0, bxeu1, byeu0, byeu1]= Drift(x0eu, x1eu, y0eu, y1eu, g, Delta);
122
123              x0   = x0 + 0.5*tau*(bx0 + bxeu0) + noisex;
124              y0   = y0 + 0.5*tau*(by0 + byeu0) + noisey;
125              x1   = x1 + 0.5*tau*(bx1 + bxeu1);
126              y1   = y1 + 0.5*tau*(by1 + byeu1);
127              x1(src) = x1(src) + tau;
128           end
129
130           Xc =[Xc, x1];          % correlazione < x(s).x(t) >
131        end
132
133        Xcorr = mean(Xc,2);
134        Xcum  = [Xcum, Xcorr];    % average on steps*exps/skip...
135     end
136
137     toc
138
```

Ciclo parfor (pararell for)

- *Risultati finali*: La media calcolata ("*Xcorr*") viene aggiunta a una matrice cumulativa "*Xcum*". Alla fine, "*Xcum*" conterrà la correlazione cumulativa media su tutti gli esperimenti e passi temporali.

```
106        for j = 1:skip:steps              % start Langevin run
107           for m = 1:skip
108              noisex = D*randn(N,1);
109              noisey = D*randn(N,1);
110
111              [bx0, bx1, by0, by1]= Drift(x0, x1, y0, y1, g, Delta);
112
113              x0eu = x0 + bx0 * tau  + noisex;
114              y0eu = y0 + by0 * tau  + noisey;
115
116              x1eu = x1 + bx1 * tau;
117              y1eu = y1 + by1 * tau;
118              x1eu(src) = x1eu(src) + tau;
119
120
121              [bxeu0, bxeu1, byeu0, byeu1]= Drift(x0eu, x1eu, y0eu, y1eu, g, Delta);
122
123              x0   = x0 + 0.5*tau*(bx0 + bxeu0) + noisex;
124              y0   = y0 + 0.5*tau*(by0 + byeu0) + noisey;
125              x1   = x1 + 0.5*tau*(bx1 + bxeu1);
126              y1   = y1 + 0.5*tau*(by1 + byeu1);
127              x1(src) = x1(src) + tau;
128           end
129
130           Xc =[Xc, x1];          % correlazione < x(s).x(t) >
131        end
132
133        Xcorr = mean(Xc,2);
134        Xcum  = [Xcum, Xcorr];    % average on steps*exps/skip...
135     end
136
137     toc
138
```

Operazioni

Questa sezione del codice esegue operazioni sulla simulazione, come la selezione di un intervallo di tempo, l'adattamento di correlazioni ed energie e calcoli aggiuntivi sulla media di "*Xcum*". Viene anche eseguito un ulteriore adattamento per ottenere la pendenza (slope).

```
139     interv = round(N/trim(1)):4:round(N/trim(2));
140     Xcum   = Xcum(interv,:);
141     trim   = t(interv);
142
143     E=zeros(exps,1);
144
145     if(~isnan(Xcum)),
146        for n=1:exps,
147           s1 = fitcorr(Xcum(:,n), setup, interv);
148           E(n) = hbar*s1;
149        end
150     end
151
152     X  = mean(Xcum,2);
153     dX = std(Xcum,0,2)/sqrt(exps-1);
154
155     [s1,f,Xf1] = fitcorr(X, setup, interv); % fit the exponential decay
156
157     slope = log(X(1)/X(3))/(trim(3)-trim(1));
158
```

Risultati

In questa sezione vengono eseguiti i calcoli finali che portano ai risultati mostrati.

Verrà visualizzata la differenza di energia calcolata dal programma *LangPar2D.m* e la differenza di energia fornita da *schr2D.m* (insieme ad altri valori).

Questi valori devono essere molto simili.

```
159     dE = std(E)/sqrt(exps-1)
160     E  = mean(E);
161     disp(['<E1-E0>=',num2str(E),' +/- ',num2str(dE)])
162
163     disp('Running schr2D')
164     Eqm = schr2D(g);        % gap ottenuto dalla diagonalizzazione dell'Hamilt.
165     gap = Eqm(2)-Eqm(1)
166     dev = ceil(100*(gap - E)/dE);
167
168     disp(['Dev is ', num2str(abs(dev)), '% of std(E)/sqrt(exps-1)'])
169
```

```
dE =
    0.0262
<E1-E0>=0.80982 +/- 0.026247
Running schr2D
gap =
    0.8003
Dev is 36% of std(E)/sqrt(exps-1)
```

Risultati

Questa parte è necessaria per creare il grafico che relaziona <x(s).x(t)> col tempo euclideo. Verrà visualizzato un grafico in cui saranno riflessi i dati ottenuti dai programmi *LangPar2D.m* e *schr2D.m*. Infine, verranno mostrati gli ultimi risultati e successivamente il programma sarà completato.

```
170     errorbar(trim, X, dX, 'bo:','LineWidth',1.5, ...
171             'MarkerSize',6, 'MarkerFaceColor','r','MarkerEdgeColor','b');
172     hold on
173     Xeigs1 = exp(-trim*gap/hbar);
174     Xslope = exp(-trim*E/hbar);
175     Xslopep = exp(-trim*(E+2*dE)/hbar);
176     Xslopem = exp(-trim*(E-2*dE)/hbar);
177
178
179     plot(trim, Xeigs1*X(1)/Xeigs1(1),'g-.','LineWidth',1.5); % QM correlation
180     plot(trim, Xslopep*X(1)/Xslopep(1),'b:','LineWidth',1.); % from slope
181     plot(trim, Xslopem*X(1)/Xslopem(1),'b:','LineWidth',1.); % from slope
182
183     set(gca,'YScale','log')
184
185     xlabel('Euclidean time', 'FontSize',12, 'FontAngle','it')
186     ylabel('<x(s) x(t)>_c', 'FontSize',12, 'FontAngle','it')
187
188     title(['Stoch. quantization - 2D-anharmonic potential - En-gap at g=', num2str(g)],...
189           'FontSize',14,'FontAngle','it','FontWeight','normal')
190
191     grid on; grid minor
192
193     print -dpdf -bestfit LangPar2D.pdf
194
195     save LangPar2D.mat
196
197     results = struct('E',E,'dE',dE,'Eqm',Eqm,...
198                 'X',X,'dX',dX,'trim',trim,'tau',tau);
199
200 end
```

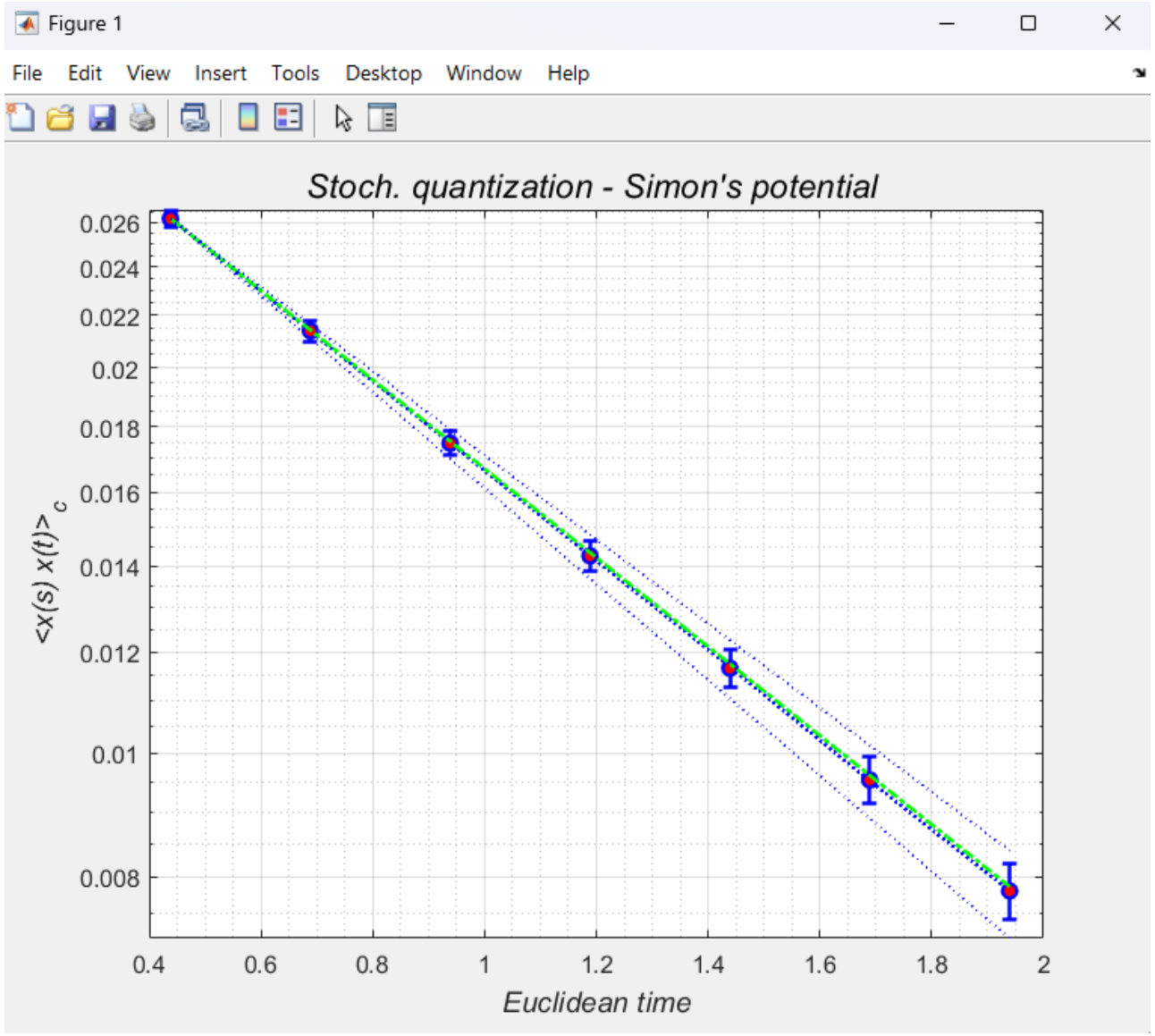
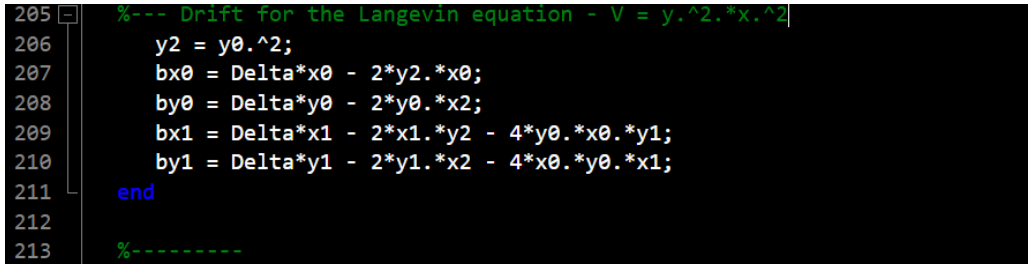


Figure 1
File Edit View Insert Tools Desktop Window Help
Stoch. quantization - Simon's potential
0.026
0.024
0.022
0.02
0.018
0.016
0.014
0.012
0.01
0.008
<x(s) x(t)>c
0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Euclidean time

Funzione Drift

In questa funzione viene calcolato il "Drift" per l'equazione di Langevin; nel nostro caso, verrà utilizzato il potenziale di Simon.

Per eseguire queste operazioni si utilizzano i metodi matematici precedentemente spiegati.



Funzione fitcorr

Questa funzione è utilizzata per adattare la correlazione ottenuta in un processo Langevin.

Ecco il funzionamento:

- Vengono estratti i parametri chiave del sistema dalla configurazione fornita.
- Viene definito il tempo t in base alla lunghezza e al numero di punti nella simulazione.

```
function   [s, f, Xf] = fitcorr(Xc, setup, interv)
% Extract the slope of the correlation produces in LangCluster
%              E(x(t)x(s)) ~f*exp(-s|t-s|)
% using fminsearch; Xf returns the fit
%  © E. Onofri, U.Parma, LabTeo-2005

   N     = str2num(setup{1});            % extract parameters
   T     = str2num(setup{2});

   hbar = 1. ;
   t = linspace(0,T,N)';

   opt = optimset('TolFun', 1e-4,...
                  'TolX', 1e-4,...
                  'MaxIter', 1e6,...
                  'MaxFunEval', 1e6,...
                  'Display','on');
   ti = t(interv);

   z = fminsearch(@corr , [1., 1.], opt, ti, Xc, T);

   s = z(1);
   f = z(2);

   Xf = f*(exp(-s*t)+exp(-s*(T-t)));

end

%--------------------
```

Funzione fitcorr

- Vengono configurate le opzioni per l'algoritmo di ottimizzazione fminsearch mediante l'oggetto opt
- Viene selezionato l'intervallo di tempo desiderato utilizzando interv.
- Viene utilizzato l'algoritmo fminsearch per adattare i parametri s e f minimizzando una funzione di costo definita dalla funzione annidata corr.

```
function   [s, f, Xf] = fitcorr(Xc, setup, interv)
% Extract the slope of the correlation produces in LangCluster
%              E(x(t)x(s)) ~f*exp(-s|t-s|)
% using fminsearch; Xf returns the fit
%  © E. Onofri, U.Parma, LabTeo-2005

   N     = str2num(setup{1});            % extract parameters
   T     = str2num(setup{2});

   hbar = 1. ;
   t = linspace(0,T,N)';

   opt = optimset('TolFun', 1e-4,...
                  'TolX', 1e-4,...
                  'MaxIter', 1e6,...
                  'MaxFunEval', 1e6,...
                  'Display','on');
   ti = t(interv);

   z = fminsearch(@corr , [1., 1.], opt, ti, Xc, T);

   s = z(1);
   f = z(2);

   Xf = f*(exp(-s*t)+exp(-s*(T-t)));

end

%--------------------
```

Funzione fitcorr

- I parametri adattati vengono restituiti come s e f.
- Viene generata una funzione adattata Xf utilizzando i parametri ottenuti.

Quindi, questa funzione viene utilizzata per adattare la correlazione ottenuta in un processo Langevin a una forma funzionale specifica, fornendo così informazioni sulla relazione tra le variabili nel sistema studiato.

```
function   [s, f, Xf] = fitcorr(Xc, setup, interv)
% Extract the slope of the correlation produces in LangCluster
%              E(x(t)x(s)) ~f*exp(-s|t-s|)
% using fminsearch; Xf returns the fit
%  © E. Onofri, U.Parma, LabTeo-2005

   N     = str2num(setup{1});            % extract parameters
   T     = str2num(setup{2});

   hbar = 1. ;
   t = linspace(0,T,N)';

   opt = optimset('TolFun', 1e-4,...
                  'TolX', 1e-4,...
                  'MaxIter', 1e6,...
                  'MaxFunEval', 1e6,...
                  'Display','on');
   ti = t(interv);

   z = fminsearch(@corr , [1., 1.], opt, ti, Xc, T);

   s = z(1);
   f = z(2);

   Xf = f*(exp(-s*t)+exp(-s*(T-t)));

end

%--------------------
```

Funzione corr e finale del codice

Questa funzione, chiamata corr, è utilizzata come funzione di costo nel processo di ottimizzazione eseguito da fminsearch all'interno della funzione principale fitcorr.

Ecco il funzionamento:

- Vengono estratti i parametri s e f dal vettore z.

```
245   function err = corr(z, t, Xc, T)
246
247     s=z(1);
248     f=z(2);
249
250     err = norm( Xc - f*(exp(-s*t) + exp(-s*(T-t))));
251
252
253   end
254
255
256
```

Funzione corr e finale del codice

- Viene calcolata la funzione di costo come norma della differenza tra la correlazione misurata (Xc) e la correlazione adattata utilizzando la forma funzionale f*exp(-s*t) + f*exp(-s*(T-t)).
- La norma utilizzata qui è la norma euclidea, che misura la distanza tra due punti in uno spazio.

```
245   function err = corr(z, t, Xc, T)
246
247     s=z(1);
248     f=z(2);
249
250     err = norm( Xc - f*(exp(-s*t) + exp(-s*(T-t))));
251
252
253   end
254
255
256
```

Funzione corr e finale del codice

In sintesi, questa funzione corr valuta la discrepanza tra la correlazione misurata e quella teorica adattata, e questo valore viene minimizzato durante il processo di ottimizzazione per trovare i parametri s e f ottimali che si adattano meglio ai dati osservati.

```
245   function err = corr(z, t, Xc, T)
246
247     s=z(1);
248     f=z(2);
249
250     err = norm( Xc - f*(exp(-s*t) + exp(-s*(T-t))));
251
252
253   end
254
255
256
```

Programma secondario

- Creato per calcolare lo spettro di autovalori e autofunzione per l'operatore di Schrödinguer in 2D con un potenziale specifico definito per V.

- Potenziale di Simon

Parametri d'entrata

- Inizializziamo i parametri d'entrata e stabiliamo i suoi valori predeterminati:
  - N: Numeri di punti nello spazio di coordinati
  - L: Longitude nello spazio di coordinati
  - Neig: Numero di autovalori a calcolare
  - Prec: Tolleranza per il metodo di Lanczos
  - lanczv: Parametro per il metodo di Lanczos
  - verbose:Controlla la quantità di informazione stampata durante la esecuzione.

```
12
13    if nargin < 7, verbose = 0; end
14    if nargin < 6, lanczv = 60; end
15    if nargin < 5, prec = 1e-6;end
16    if nargin < 4, Neig = 12; end
17    if nargin < 3, L = 10.0; end
18    if nargin < 2, N = 128; end
19    if nargin < 1, g=1;    end
20
```

Parametri d'entrata

- Configuriamo gli opzioni 'opts' per la funzione 'eigs' che e utilizzata per la diagonalizzazione:
  - tol: Tolleranza per essere utilizzata per 'eigs'
  - disp: Controlla la verbosità di 'eigs'
  - isreal e issym: Indicano che la matrice a diagonalizzare è reale e simmetrica.
  - p: Parametro per il metodo di Lanczos

```
21
22    opts.tol  = prec; % to be used by eigs (JacobiSpectrum)
23    opts.disp = verbose;    % eigs works quietly - alternatively =1 , =2.
24    opts.isreal = true;
25    opts.issym = true;
26    opts.p = lanczv;
```

Reticolo spaziale

Queste righe di codice stabiliscono un reticolo spaziale bidimensionale per il sistema fisico che vogliamo analizzare.

- Calcoliamo la spaziatura di ogni cuadricula
- Creiamo un reticolo spaziale bidimensionale attraverso 'meshgrid'
- Calcoliamo il radio 'r' e il suo quadrato

```
28    %% -------- space Lattice ----------------------
29    dx = 2*L/N;
30    [x,y]=meshgrid(-L*(1-1/N) : dx : L*(1-1/N)); % symmetrized instead
31    r = sqrt(x.^2+y.^2);                         % of choosing [-L,L-a]
32    r2= r.^2;
```

Momento del reticolo nello spazio di Fourier

Configuriamo il spazio di momento di Fourier:

- Creiamo il vettore che rappresenta le coordinate di momento attraverso ‘fftshift’. Questo vettore genera valori equispaziati e li regola in modo che siano centrati attorno allo zero.

- Generiamo le matrici bidimensionali che rappresentano le coordinati nello spazio di momento.

```
33    %% -------- momentum lattice in Fourier space --
34    fourier_space = fftshift(pi*(-N/2:N/2-1)/L);
35    [kx,ky]=meshgrid(fourier_space);
36
```

Derivate parziali nello spazio di Fourier

Calcoliamo il laplaciano nello spazio di Fourier

- Le derivate parziali quadratiche sono calcolate rispetto alle componenti della quantità di moto kx e ky

```
37    %% Partial derivatives in Fourier space --------
38
39    kxx = kx.^2;                          % Partial derivatives
40    kyy = ky.^2;
41    k2 = kxx+kyy;                         % Laplacean
42
```

- Sumamio kxx e kyy per ottenere il laplaciano, che è essenziale esprimere il termine cinetico nell'equazione di Schrödinger nella rappresentazione della quantità di moto.

Potenziale d'energia

- Definiamo il potenziale di energia che sarà utilizzato nella equazione di Schrödinguer.

- Inizializziamo la funzione d'onda iniziale come una gaussiana centrata nell'origine.

- Per il metodo di diagonalizzazione 'eigs' viene impostato il vettore iniziale 'vo'.

```
43    %% Potential Energy
44    %V = r2/2 + g * r2.^2/4 ; % Oscillatore anarmonico
45    % V = r2/2; % Oscillatore anarmonico - test accuratezza
46    % V = g*r;
47    % V = -g./r;  % Coulomb
48     V = y.^2.*x.^2; % Simon's potential
49    %disp('here ok')
50
51    psi0 = exp(-r2/2);
52    opts.v0 = reshape(psi0,N^2,1);
53
```

Diagonalizzazione attraverso 'eigs'

- La funzione 'eigs' esegue la diagonalizzazione dell'operatore di Schrödinger.

```
54    %% Diagonalization using "eigs"
55    [Psi, E] = eigs(@Hamiltonian, N^2, Neig, 'SA', opts);
56    E = diag(E);
57    phi=zeros(N,N,Neig);
58    for j=1:Neig
59       phi(:,:,j) = reshape(Psi(:,j),N,N);
60    end
```

- Estraiamo i autovalori, li immagazziniamo in la matrice 'E' e convertiamo la matrice in un vettore.

- L'insieme delle autofunzioni è organizzato nella matrice tridimensionale 'phi'

Funzione Hamiltonian

Questa funzione implementa l'operatore hamiltoniano nel contesto dell'equazione di Schrödinger bidimensionale. L'azione di questa funzione su una funzione d'onda fornisce l'energia totale del sistema quantistico in quello specifico stato.

- Prende un vettore di input xin e restituisce un vettore di output xout.

```
65    function xout = Hamiltonian(xin)
```

Funzione Hamiltonian

- Trasformiamo il vettore di entrata in una matrice bidimensionale di misure 'NxN', la funzione 'eigs' lavora con vettori.

- Calcoliamo la trasformata di Fourier bidimensionale della funzione d'onda 'psi' attraverso 'fft2'. Ciò porta la funzione d'onda dallo spazio reale allo spazio della quantità di moto.

```
65  function xout = Hamiltonian(xin)
66  % Embedded function  - CAUTION! All var
67  % Insert here the definition of the Ham
68
69  psi = reshape(xin(1:N^2),N,N);
70  fpsi=fft2(psi);
71  Deltapsi = real(ifft2(k2.*fpsi));
72
73  Hpsi  = 0.5* Deltapsi +  V .* psi;
74
75  xout   = reshape(Hpsi, N^2, 1);
76
77  end
```

Funzione Hamiltonian

- Calcoliamo il termine spaziale dell'operatore laplaciano nello spazio di fourier e trasformato nuovamente nello spazio reale.
- Combiniamo il termino cinetico e il termino potenziale per formare il Hamiltoniano che rappresenta l'energia totale del sistema quantico in termine della funzione d'onda.
- Trasformiamo la matrice risultante 'Hpsi' di nuovo in un vettore.

```
65  function xout = Hamiltonian(xin)
66  % Embedded function  - CAUTION! All var
67  % Insert here the definition of the Ham
68
69  psi = reshape(xin(1:N^2),N,N);
70  fpsi=fft2(psi);
71  Deltapsi = real(ifft2(k2.*fpsi));
72
73  Hpsi  = 0.5* Deltapsi +  V .* psi;
74
75  xout   = reshape(Hpsi, N^2, 1);
76
77  end
```

Risultati

- La funzione ritorna le autovalori ‘E’, le autofunzioni ‘phi’, sia come le coordinati spaziali (‘x’ e ‘y’) e il potenziale ‘v’.

- Questo risultato ci aiuta a calcolare il gap secondo l'equazione di Schrödinger nel programma principale, e poter così acquistare con il valore ottenuto tramite il metodo di Langevin-Parisi.

Programma runLangPar

- Questo ultimo programma utilizza le due programme scorse per fare 3 serie di esperimenti con diversi tau per ottenere diversi risultati di ogni serie di esperimenti e fare una rappresentazione grafica.

Conclusione

Analizzeremo i risultati ottenuti utilizzando il metodo Langevin-Parisi e li confronteremo con il valore dell'operatore Schrödinguer.

- Attraverso il metodo di Langevin-Parisi non otterremo sempre lo stesso risultato, poiché è un processo stocastico, ma dovrà essere sempre intorno al valore ottenuto tramite Schrödinger.

Conclusione

- Tramite Schrodinger abbiamo ottenuto un valore fisso di gap = 0,8003
- Dopo aver eseguito il programma principale più volte, il valore più vicino ottenuto attraverso il metodo di Langevin-Parisi era 0,80982 con un errore standard di 0,02647.

Questo valore è un valore molto vicino a quello ottenuto utilizzando l'operatore di Schrödinger e l'errore ottenuto è piccolo rispetto al valore di gap ottenuto ciò significa che abbiamo un'elevata precisione nella stima del gap.

Conclusione

- dev ci fornisce il numero di volte in cui il gap energetico calcolato (gap) è superiore all'errore standard stimato nell'energia media (dE)
- Il valore ottenuto è del 36%, che è un valore basso.

Il gap energetico è significativamente maggiore dell'errore nella stima dell'energia media, che è un buon indicatore dell'affidabilità dei risultati del calcolo.

Conclusione

- Il metodo di Langevin-Parisi è un buon metodo per ottenere il gap di un sistema fisico in due dimensioni:

- Ci fornisce un valore abbastanza preciso.

- I risultati hanno un’elevata affidabilità.

FINE