Lo scopo è di risolvere numericamente le equazioni del moto di Newton

$$m_i \mathbf{a}_i = F_i(\mathbf{r}_i) = -\partial V(\mathbf{r})/\partial \mathbf{r}_i. \tag{1}$$

Vista l'impossibilità per un computer di trattare il continuo, è necessario discretizzare l'asse temporale; prenderemo intervalli uguali di ampiezza δ, che sarà quindi il *passo temporale* (timestep) del nostro algoritmo. I valori del tempo che consideriamo quindi sono quelli, ammesso di partire a $t = 0$, corrispondenti a $t_i = i\delta$, $i \in \{0, 1, \ldots, n\}$. Il tempo simulato, quindi, sarà $\tau = n\delta$. È quindi ovvio che è nostro interesse scegliere un valore δ il più grande possibile compatibilmente con una accettabile accuratezza dell'integrazione dell'equazione, al fine di minimizzare il numero di passi n a parità di τ. Per semplicità qui prenderemo in considerazione un sistema con un solo grado di libertà, e chiameremo x_0 la posizione corrente, x_+ quella allo step successivo eccetera. Analogamente F_0 sarà la forza al tempo corrente e così via per tutte le grandezze.

Se ipotizziamo forze conservative, abbiamo in linea di principio un sistema ad energia costante (*ensemble microcanonico*). La soluzione numerica evidentemente potrà non rispettare, date le inevitabili approssimazioni e troncamenti, questo criterio. Dal punto di vista del calcolo di proprietà di equilibrio, tuttavia, una violazione moderata (entro pochi punti percentuali) di questa condizione può essere accettata; ricordiamo che comunque il sistema simulato è con ogni probabilità caotico e non possiamo comunque sperare di seguirne l'evoluzione in modo perfetto. La condizione realmente importante è l'assenza di una variazione secolare (drift) dell'energia, visto che questa porterebbe a non avere una simulazione in un ensemble ben definito né a identificare con certezza la posizione del sistema simulato nello spazio delle fasi termodinamico.

Come è noto dalla Meccanica Teorica, la conservazione dell'energia è strettamente legata all' assenza di una dipendenza della Lagrangiana/Hamiltoniana del sistema dal tempo, quindi alla proprieà di reversibilità temporale. È inoltre legata (Teorema di Liouville) alla conservazione del volume dei punti rappresentativi del sistema nello spazio delle fasi. Si può quindi pensare che gli algoritmi di soluzione delle equazioni del moto dovranno appartenere a una categoria (algoritmi simplettici) che rispetta queste importanti proprietà. Possiamo immaginarci dunque che un algoritmo apparentemente adeguato come quello di Eulero

$$\begin{aligned} x_+ &= x_0 + v_0\delta + F_0\delta^2/2m \\ v_+ &= v_0 + F_0\delta/m, \end{aligned}$$

ottenuto sviluppando in serie all'ordine più basso possibile in δ le soluzioni delle equazioni del moto (in questo caso in forma Hamiltoniana) non funzioni bene. Certamente le soluzioni tendono a quelle corrette per $\delta \to 0$, con errore $O(\delta^3)$, ma la non reversibilità temporale dell'algoritmo risultante causa un drift nell'energia che risulta inaccettabile a meno di scegliere δ estremamente piccolo. Infatti l'algoritmo è "sbilanciato" dal lato del futuro; ovvero le derivate non sono centrate all'istante corrente. Questo rende l'algoritmo tale per cui invertendo il segno del tempo ($\delta \to -\delta$) non si ripercorrono esattamente gli stessi punti, e quindi viola la proprietà di inversione temporale.

Possiamo però facilmente crearne uno che rispetti esattamente l'inversione temporale. Scrivendo la derivata seconda di x rispetto a t in modo centrato, l'equazione di Newton diventa, discretizzata:

$$F_0/m = \left[(x_+ - x_0)/\delta - (x_0 - x_-)/\delta\right]/\delta = (x_+ - 2x_0 + x_-)/\delta^2,$$

ovvero

$$x_+ = 2x_0 - x_- + F_0\delta^2/m, \tag{2}$$

che per come è stato ottenuto è invariante per inversione temporale. Per convincersene si cambi il segno a δ e si invertano x_+ e x_-. Questo algoritmo è noto come *algoritmo di Verlet*. Si può ottenerlo anche sviluppando in serie x_+ "in avanti" e x_- "all'indietro":

$$\begin{aligned} x_+ &= x_0 + v_0\delta + a_0\delta^2/2 + \partial^3 x/\partial t^3|_0\delta^3/6 + O(\delta^4) \\ x_- &= x_0 - v_0\delta + a_0\delta^2/2 - \partial^3 x/\partial t^3|_0\delta^3/6 + O(\delta^4) \end{aligned}$$

e sommandole. Il risultato sarà evidentemente invariante per inversione temporale. I termini dispari si cancellano tutti e si rimane con l'algoritmo 2. Si vede anche come, in aggiunta alla proprietà di inversione temporale, l'ordine dell'algoritmo sia più alto di uno rispetto a Eulero, visto che i termini trascurati sono $O(\delta^4)$ e non $O(\delta^3)$.

Questo algoritmo ha molte buone proprietà. È time-reversible, semplice da implementare, e non richiede né più memoria né più tempo di calcolo dello stretto necessario. Si sarà osservato, tuttavia, che strada facendo è scomparsa la velocità, che ora è implicita e non figura tra le variabili. Questo può essere scomodo, visto che molte cose, a partire dall'energia cinetica, dipendono dalle velocità, e come si vedrà avere un modo per manipolare facilmente le velocità è indispensabile per varie applicazioni. Cerchiamo quindi un algoritmo che comprenda esplicitamente v.

Scegliendo di discretizzare simmetricamente le equazioni in forma hamiltoniana:

$$\begin{aligned} \dot{x} &= v \\ m\dot{v} &= F \end{aligned}$$

possiamo ottenere due equazioni come le seguenti:

$$\begin{aligned} (x_+ - x_0)\delta &= v_{1/2} \\ (v_{1/2} - v_{-1/2})\delta &= F_0/m \end{aligned}$$

dove la prima è centrata intorno al punto $t+\delta/2$, mentre $1/2$ e $-1/2$ indicano punti a metà tra t e $t+\delta$ e $t-\delta$ rispettivamente, e la seconda è centrata attorno a t. Ovvero le velocità vengono calcolate a punti temporali sfalsati di $\delta/2$ rispetto alle coordinate. L'algoritmo risultante è

$$\begin{aligned} v_{1/2} &= v_{-1/2} + F_0/m\delta \\ x_+ &= x_0 + v_{1/2}\delta \end{aligned} \tag{3}$$

ed è chiamato *leapfrog*, termine che in inglese indica il "gioco della cavallina", perché posizioni e velocità in un certo senso si "scavalcano" a turno. Si noti che l'uso di memoria è uguale all'algoritmo di Verlet, visto che è necessario ricordarsi le velocità da un passo all'altro ma non è più necessario ricordarsi le posizioni precedenti.

Il fatto di avere due discretizzazioni temporali distinte porta a qualche scomodità. Per esempio, energie potenziali e cinetiche vengono calcolate a istanti diversi e questo porta ad avere una apparente non conservazione dell'energia, che, pur non grave, e volendo correggibile calcolando ad ogni

istante $v_0 = (v_{1/2} + v_{-1/2})/2$, è fastidiosa. È però possibile sviluppare un algoritmo (*velocity Verlet*) che usa esplicitamente le velocità negli stessi istanti delle posizioni:

$$\begin{aligned} x_+ &= x_0 + v_0\delta + F_0\delta^2/2m \\ v_+ &= v_0 + (F_+ + F_0)\delta/2m. \end{aligned} \tag{4}$$

La seconda equazione è evidentemente tempo-simmetrica rispetto a $t + \delta/2$. Non è immediatamente evidente che lo sia l'intero algoritmo, visto che la prima delle equazioni 4 è identica a quella, insoddisfacente, di Eulero. Tuttavia è possibile far vedere che l'algoritmo è equivalente al Verlet originale. Scriviamo infatti l'equazione 2 per lo step successivo e inseriamo la seconda delle 4:

$$\begin{aligned} x_{++} &= 2x_+ - x_0 + F_+\delta^2/m = 2x_+ - x_0 + F_+\delta^2/2m - F_0\delta^2/2m + (v_+ - v_0)\delta \\ x_{++} &\quad -x_+ - v_+\delta - F_+\delta^2/2m = x_+ - x_0 - v_0\delta - F_0\delta^2/2m. \end{aligned}$$

Ma se uno soddisfa anche la prima delle 4 l'ultima equazione scritta sarà identicamente soddisfatta, quindi gli algoritmi sono equivalenti.

Si noti che la seconda delle 4 prevede l'uso delle forze sia all'istante corrente sia all'istante successivo. Questo non significa che venga raddoppiato il calcolo delle forze, che sarebbe un aggravio computazionale inaccettabile visto che questo calcolo è la parte più gravosa di tutto l'algoritmo: infatti la forza F_0 non è altro che la forza F_+ del passo precedente e basterà quindi tenerla in memoria. Bisogna però aggiornare le velocità "a pezzi": prima si calcola $F_0\delta/2m$ assegnandola a v_+, poi si aggiornano le posizioni, infine si calcola F_+ e si aggiunge il termine $F_+\delta/2m$ a v_+, ottenendo i valori definitivi.

Si è detto che il passo temporale δ è il risultato di un compromesso tra l'esigenza di integrare accuratamente l'equazione 1, che richiede in linea di principio $\delta \to 0$, e quella di limitare il costo computazionale, che richiede δ lunghi. Gli algoritmi simplettici sono abbastanza robusti sotto questo aspetto: il drift nell'energia appare solitamente quando le fluttuazioni attorno a un valore di equilibrio, innocue come detto all'inizio, assumono valori sensibili, a volte del 10% o anche più. Il valore ottimale deve essere ricavato tramite sperimentazione su ogni specifico sistema, ma possiamo in generale considerare che il passo temporale dovrà essere molto più breve dei tempi

tipici di oscillazione del sistema. Si può dire che, se ω è la massima frequenza presente nel sistema, allora $\delta << 1/\omega$, tipicamente qualche punto percentuale del periodo di oscillazione. Si noti che queste frequenze cambiano a secondo dello stato del sistema, in quanto un sistema molto caldo o molto compresso avrà avvicinamenti maggiori tra atomi che uno più freddo o meno denso, e questo porterà ad avere forze che variano rapidamente. Bisognerà dunque verificare l'adeguatezza di δ non solo a ogni cambio di sistema, ma anche a ogni cambio significativo di coordinate termodinamiche.