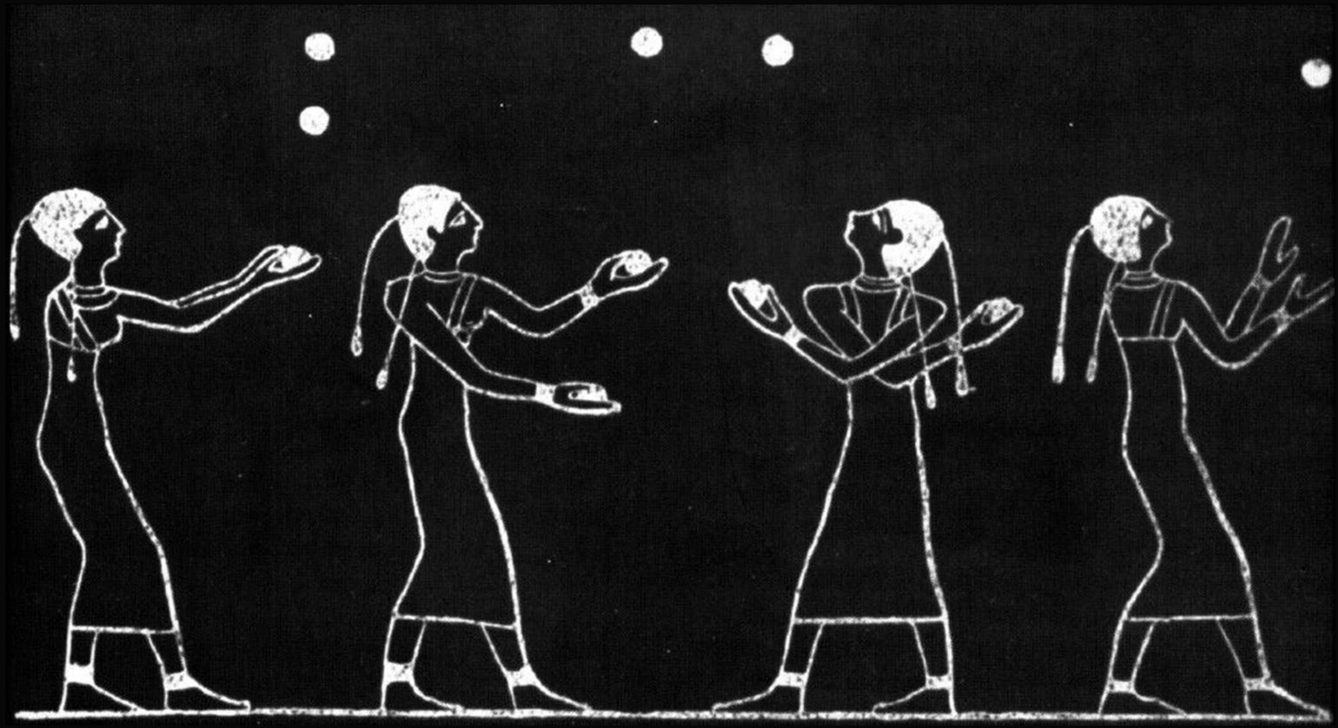
Symulacje komputerowe w fizyce

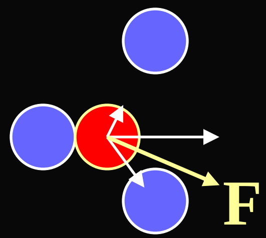


Wykład 3 – rozwiązywanie równań ruchu

Całkowanie równań Newtona

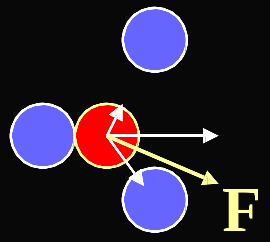
$$m_j \frac{d^2 \mathbf{r}_j}{dt^2} = \mathbf{F}_j$$

$$\mathbf{F}_j = \sum_{\substack{i=1 \\ i \neq j}}^{N} \mathbf{F}_{ij}$$



Wydaje się proste...

- obliczyć siły i przyspieszenia
- uaktualnić położenia i prędkości cząstek
- czynność powtórzyć



Jaki algorytm wybrać?

Cechy dobrego algorytmu:

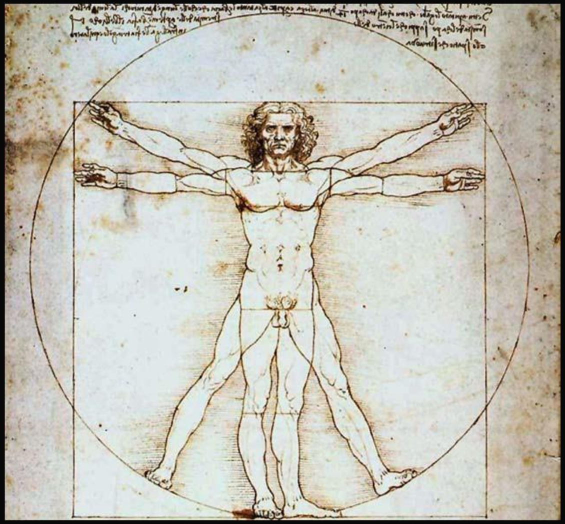
- jak najrzadsze obliczanie sił (operacja bardzo kosztowna)
- stabilność dla dużych kroków czasowych
- zachowanie energii

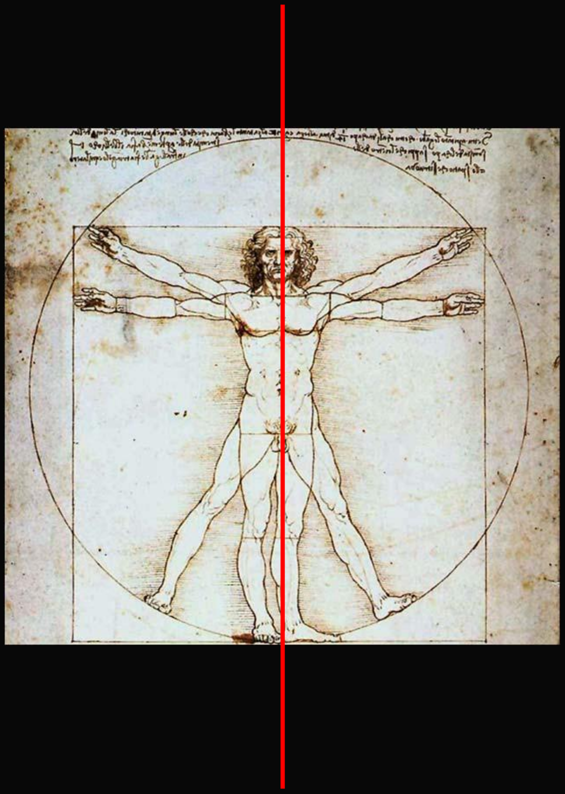
Głęboka prawda:

Te cechy najłatwiej uzyskać jeśli metoda numeryczna ma te same własności strukturalne (jak np. symetrie) co modelowany układ fizyczny

Symetria

Mówimy, iż obiekt jest symetryczny/niezmienniczy względem danego przekształcenia jeśli obrazem danego obiektu w tym przekształceniu jest on sam.



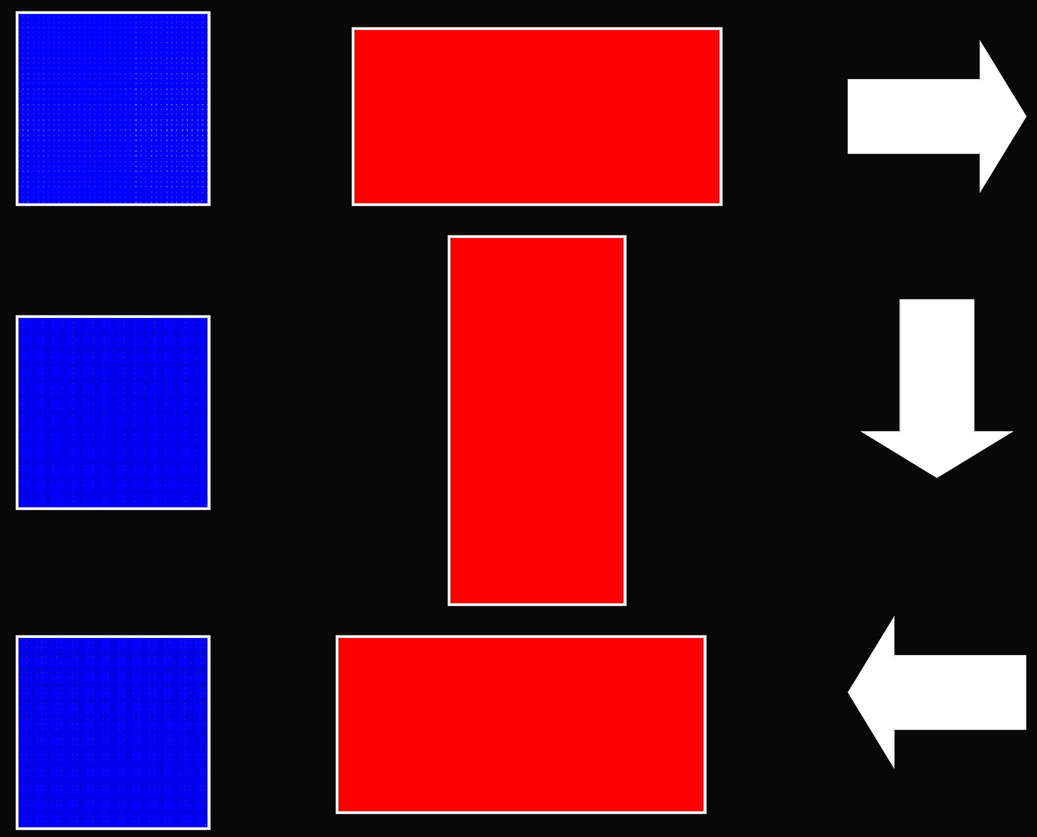


Proste symetrie

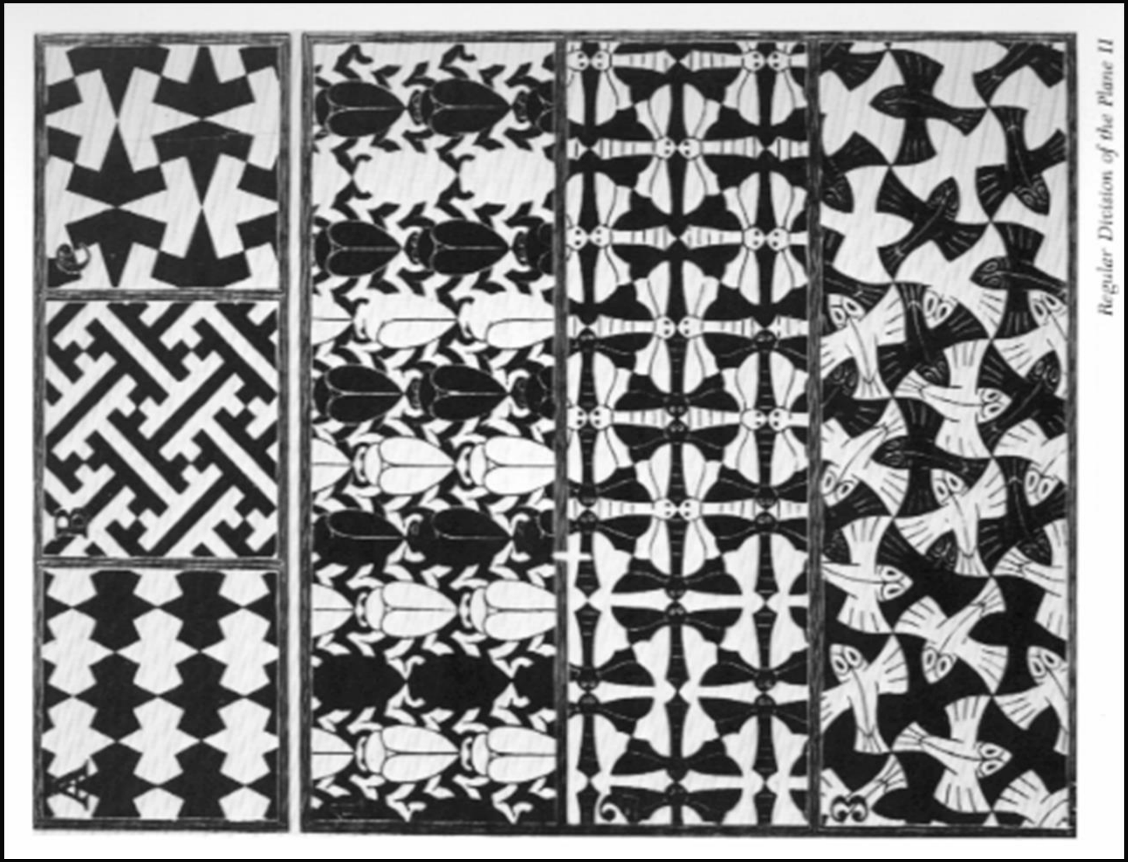
Wyjściowy kształt

obrót o 90^o

Odbicie względem pionowej osi



Skomplikowane...



Symetrie w fizyce

- W czasie
- Ze względu na przesunięcie
- Rotacyjna
- Symetria CPT…

Twierdzenie Noether – każda ciągła symetria generuje prawo zachowania

Symetrie równań Newtona

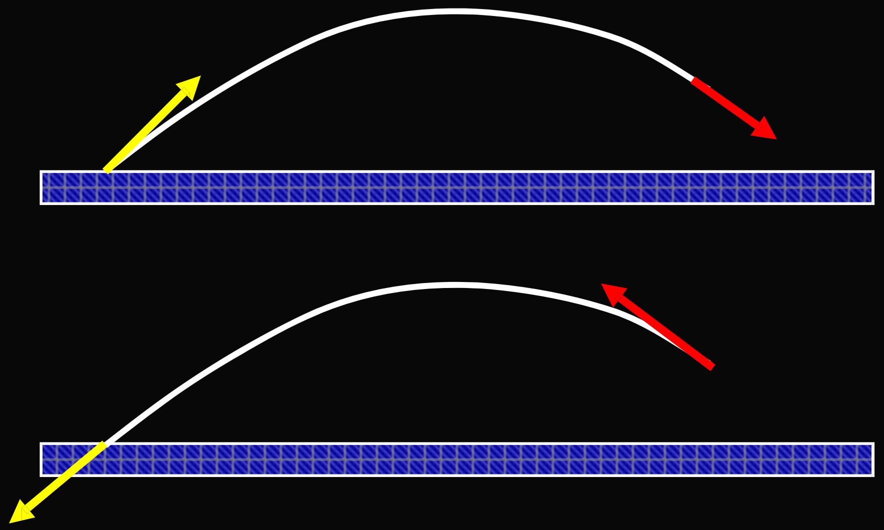
- odwracalność w czasie
- zachowanie objętości w przestrzeni fazowej

Odwracalność

trajektoria ulega odwróceniu przy zmianie znaku pędów

$$\frac{d\mathbf{r}_j}{dt} = \frac{\mathbf{p}_j}{m}$$

$$\frac{d\mathbf{p}_j}{dt} = \mathbf{F}_j$$



równania są niezmiennicze ze względu na transformację $t \to -t, \;\; p \to -p$

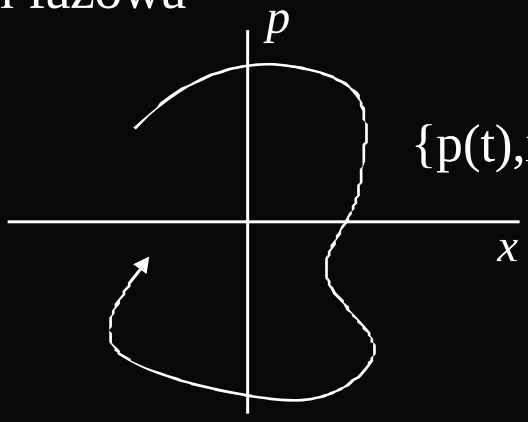
Ruch w przestrzeni fazowej

Rozważmy ruch pojedynczej cząstki w 1d

– opisywany przy użyciu 2d przestrzeni fazowej *(x,p)*
– Hamiltonian $H(p,x) = \frac{1}{2m} p^2 + V(x)$
– Równania ruchu

$$\frac{dx}{dt} = +\frac{\partial H}{\partial p} = \frac{p}{m} \qquad \frac{dp}{dt} = -\frac{\partial H}{\partial x} = F(x)$$

– Przestrzeń fazowa



{p(t),x(t)} - trajektoria

Przykład: oscylator harmoniczny

$$H(p,x) = \tfrac{1}{2}p^2 + \tfrac{1}{2}x^2$$

$(m=1,\ \omega=1)$

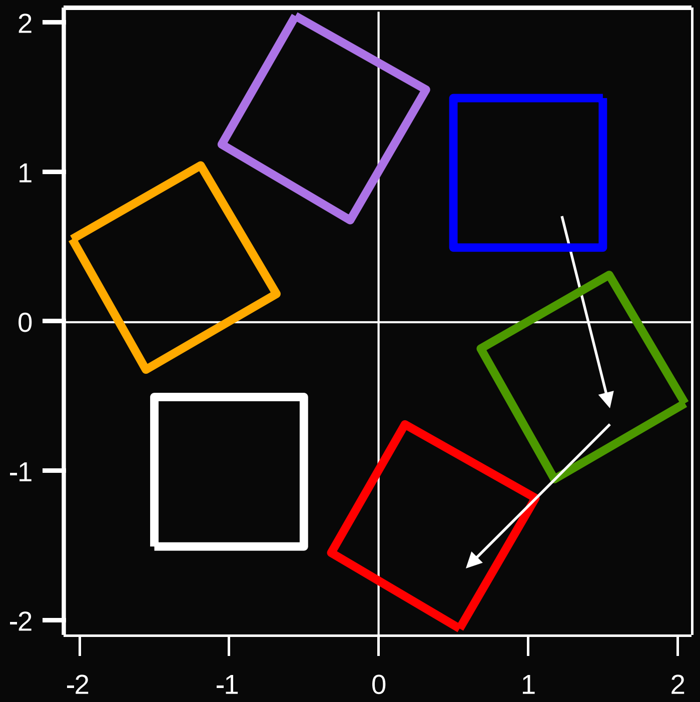
$$\frac{dx}{dt} = +p \qquad x(0) = x_0$$

$$\frac{dp}{dt} = -x \qquad p(0) = p_0$$

$$x = +x_0 \cos t + p_0 \sin t$$

$$p = -x_0 \sin t + p_0 \cos t$$

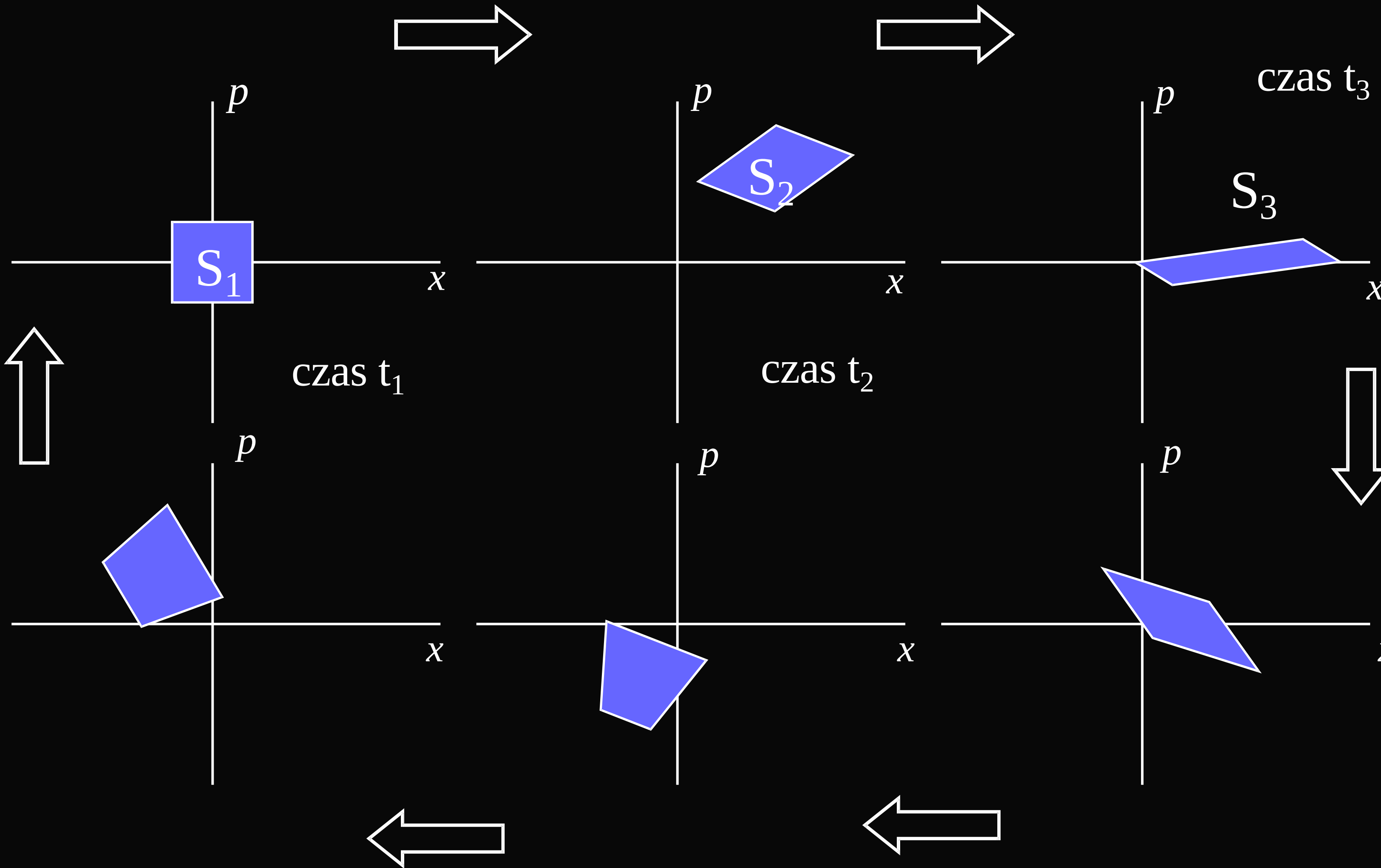
(obrót w przestrzeni fazowej)



$$E(t) = \tfrac{1}{2}p_0^2 + \tfrac{1}{2}x_0^2 = \text{constant}$$

(równanie okręgu w przestrzeni fazowej)

Zachowanie objętości w przestrzeni fazowej



$$S_1 = S_2 = S_3 = \ldots$$

Algorytmy zachowujące objętość w przestrzeni fazowej są bardziej efektywne

- stabilniejsze
- energia nie rośnie nieograniczenie

Algorytm Eulera...?

oparty na rozwinięciu Taylora:

$$\mathbf{r}(\mathrm{t}+\delta t) \approx \mathbf{r}(\mathrm{t}) + \dot{\mathbf{r}}(\mathrm{t})\delta t + \frac{1}{2!}\ddot{\mathbf{r}}(\mathrm{t})\delta t^2$$

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{p}(t)\delta t + \tfrac{1}{2}\mathbf{F}(t)\delta t^2 \qquad (m=1)$$

$$\mathbf{p}(t+\delta t) = \mathbf{p}(t) + \mathbf{F}(t)\delta t$$

- nieodwracalny
- nie zachowuje objętości w przestrzeni fazowej

Euler jest nieodwracalny...

Zacznijmy w t, zróbmy krok δt

$$r(t+\delta t) = r(t) + p(t)\delta t + \frac{1}{2}F(t)\delta t^2$$

(masa jednostkowa)

$$v(t+\delta t) = v(t) + F(t)\delta t$$

Odwróćmy czas...

$$r(t-\delta t) = r(t) - p(t)\delta t + \frac{1}{2}F(t)\delta t^2$$

Przekształćmy to, żeby uzyskać $r(t)$:

$$r(t) = r(t-\delta t) + p(t)\delta t - \frac{1}{2}F(t)\delta t^2$$

Przeskalujmy czas: $t \rightarrow t + \delta t$

$$r(t+\delta t) = r(t) + p(t+\delta t)\delta t - \frac{1}{2}F(t+\delta t)\delta t^2$$

Wzór zupełnie inny niż wyjściowy, korzystający z innych pędów i sił do aktualizowania położenia.

Algorytm Verleta

PHYSICAL REVIEW VOLUME 159, NUMBER 1 5 JULY 1967

Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*

LOUP VERLET†
Belfer Graduate School of Science, Yeshiva University, New York, New York
(Received 30 January 1967)

The equation of motion of a system of 864 particles interacting through a Lennard-Jones potential has been integrated for various values of the temperature and density, relative, generally, to a fluid state. The equilibrium properties have been calculated and are shown to agree very well with the corresponding properties of argon. It is concluded that, to a good approximation, the equilibrium state of argon can be described through a two-body potential.

lepszy (choć wciąż prosty) sposób całkowania równań Newtona

Loup Verlet

Algorytm Verleta

Krok n+1 dostajemy przez rozwinięcie Taylora:

$$r_{n+1} = r_n + v_n \Delta t + \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2 + O\left(\Delta t^3\right)\ldots$$

Podobnie wstecz w czasie, dla kroku n-1:

$$r_{n-1} = r_n - v_n \Delta t + \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2 - O\left(\Delta t^3\right)\ldots$$

Dodajemy oba wyrażenia, człon prędkościowy ulega skróceniu:

$$r_{n+1} = 2r_n - r_{n-1} + \left(\frac{F_n}{m}\right)\Delta t^2 + O\left(\Delta t^4\right)\ldots$$

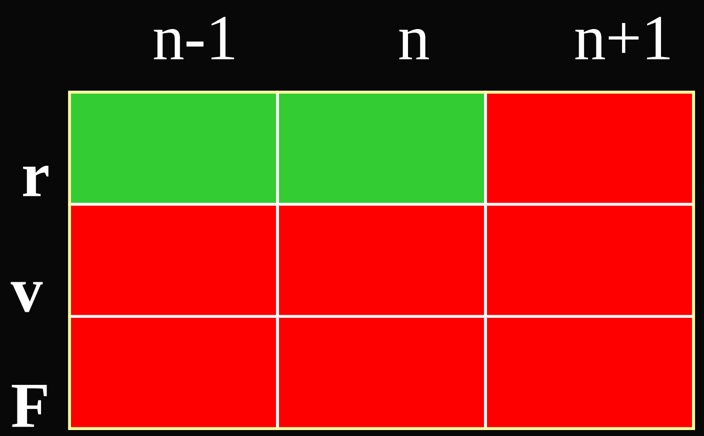
1. Użyj r_n aby obliczyć F_n
2. Użyj r_n, r_{n-1} oraz F_n (krok 1) aby uzyskać r_{n+1}

Wyliczanie prędkości nie jest konieczne, lecz można ją oszacować np. poprzez:

$$v_n = \frac{r_{n+1} - r_{n-1}}{2\Delta t} + O\left(\Delta t^2\right)$$

(przydatna do obliczenia energii bądź temperatury)

Algorytm Verleta - diagram

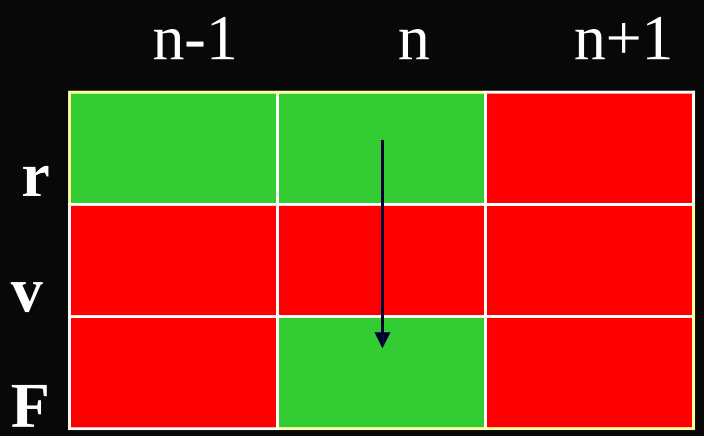


Znamy położenie w chwili

n oraz n-1

$$r_{n-1}, r_n$$

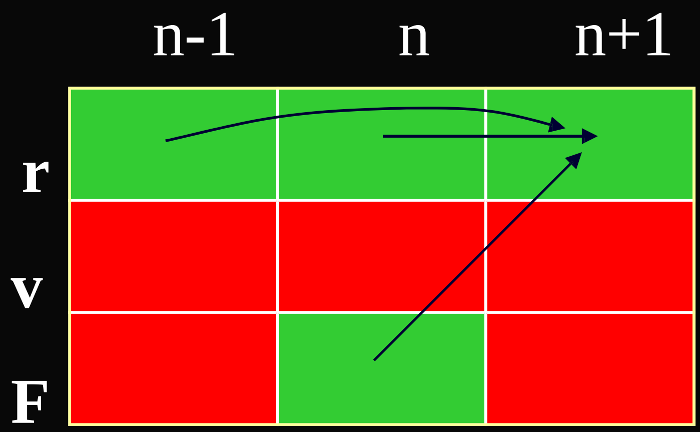
Algorytm Verleta - diagram



Obliczamy siłę odpowiadającą aktualnemu położeniu

$$r_n \rightarrow F_n$$

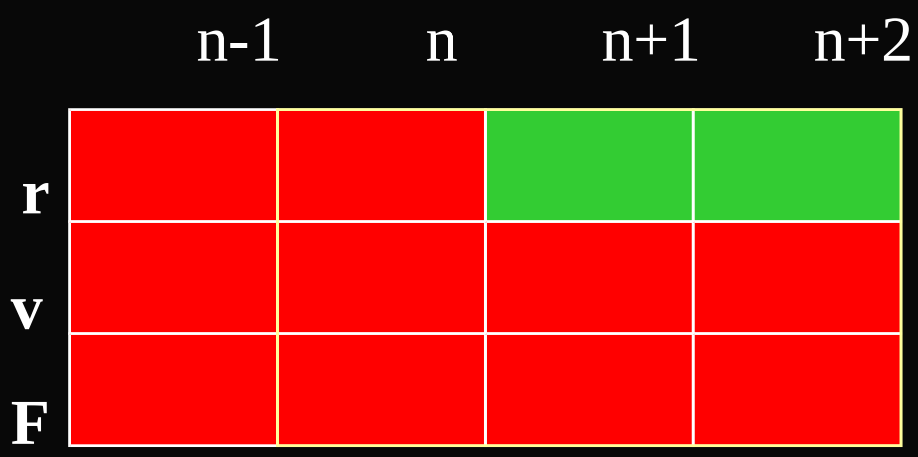
Algorytm Verleta - diagram



Na podstawie aktualnego i poprzedniego położenia oraz aktualnej siły obliczmy położenie w następnym kroku

$$r_{n+1} = 2r_n - r_{n-1} + \left(\frac{F_n}{m}\right)\Delta t^2 + O\left(\Delta t^4\right)\ldots$$

Algorytm Verleta - diagram



Przesuwamy się o jeden krok czasowy i wszystkie czynności powtarzamy

$$r_n, r_{n+1}$$

Zalety Verleta

- odwracalny
- zachowuje objętość w przestrzeni fazowej
- nie wykazuje stałego wzrostu energii
- tylko jedno obliczenie sił na cykl

Odwracalność

– krok wprzód w czasie

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \tfrac{1}{m}\mathbf{F}(t)\delta t^2$$

– zastąpmy δt poprzez –δt

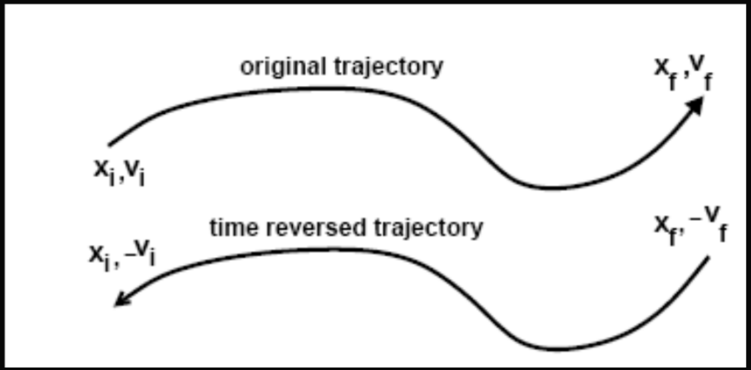
$$\mathbf{r}(t+(-\delta t)) = 2\mathbf{r}(t) - \mathbf{r}(t-(-\delta t)) + \tfrac{1}{m}\mathbf{F}(t)(-\delta t)^2$$

$$\mathbf{r}(t-\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t+\delta t) + \tfrac{1}{m}\mathbf{F}(t)\delta t^2$$

– przekształćmy powyższe, żeby otrzymać $r(t+\delta t)$:

$$r(t+\delta t) = 2r(t) - r(t-\delta t) + \frac{1}{m}F(t)\delta t^2$$

Ten sam algorytm, wykorzystujący te same położenia i siły.



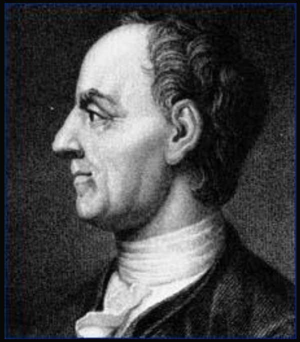
Odwracalność (2)

W praktyce, z uwagi na błędy numeryczne, nie jest możliwe dokładne odwrócenie trajektorii

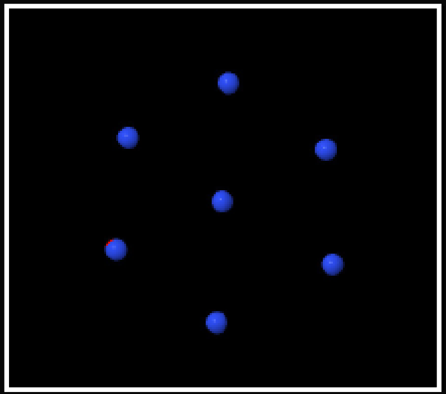
Levesque, Verlet (1993) – algorytm w pełni odwracalny:

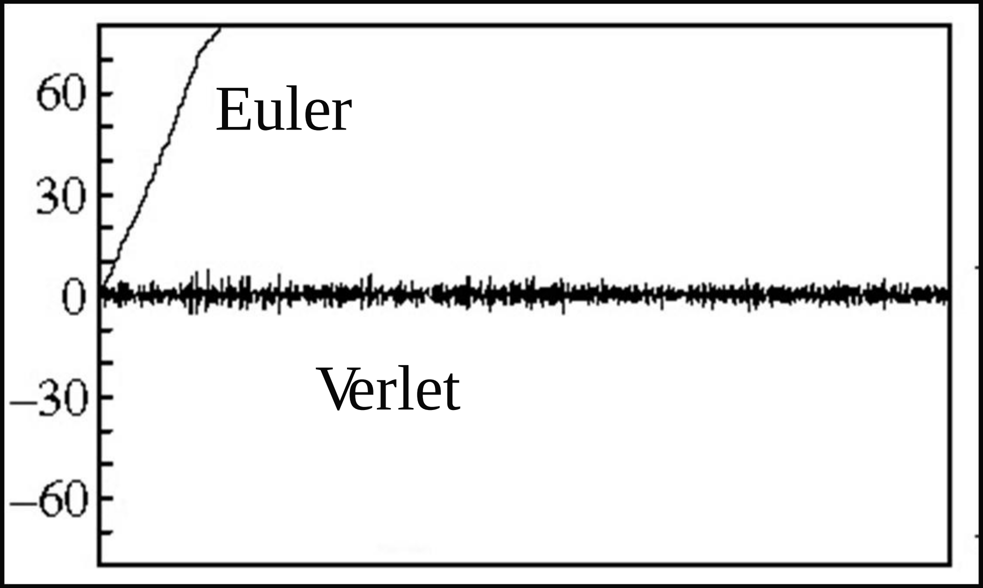
$$r_{n+1} = 2r_n - r_{n-1} + \left[\left(\frac{F_n}{m} \right) \Delta t^2 \right]_{\mathrm{int}}$$

Verlet vs Euler



Układ 7 atomów argonu oddziałujących siłami Lennarda-Jonesa z krokiem czasowym 10fs





Verlet – dodatkowe szczegóły

Inicjalizacja

- Jak dostać położenie “przy poprzednim kroku” na samym początku?
- proste przybliżenie

$$\mathbf{r}(t_0 - \delta t) = \mathbf{r}(t_0) - \mathbf{v}(t_0)\delta t$$

Otrzymywanie prędkości

- nie wyliczane w normalnym trybie pracy algorytmu
- niezbędne do uzyskania szeregu wielkości charakteryzujących układ, np.
  - temperatury
  - stałej dyfuzji
- wzór na prędkość:

$$\mathbf{v}(t) = \frac{1}{2\delta t}\left[\mathbf{r}(t+\delta t) - \mathbf{r}(t-\delta t)\right] + O(\delta t^2)$$

wady Verleta

- Prędkości nie są włączone bezpośrednio do algorytmu

– trudniej jest kontrolować energię czy temperaturę

- Położenia i prędkości nie są znane w tej samej chwili czasu, co komplikuje analizę wyników

$$\mathbf{v}_n = \frac{\mathbf{r}_{n+1} - \mathbf{r}_{n-1}}{2\Delta t}$$

(prędkość w kroku **n** nie jest osiągalna przed wykonaniem n+1 kroku w położeniu)

- Postać algorytmu wprowadza pewien dodatkowy błąd numeryczny

Szum numeryczny przy dodawaniu liczb małych do dużych

$$\mathbf{r}(t+\delta t) = 2\boxed{\mathbf{r}(t)} - \boxed{\mathbf{r}(t-\delta t)} + \frac{1}{m}\boxed{\mathbf{F}(t)\delta t^2}$$

$O(\delta t^0)$ $O(\delta t^0)$ $O(\delta t^2)$

dodawanie bardzo małego członu do różnicy dwóch dużych

Algorytm skokowy (‘Leapfrog’)

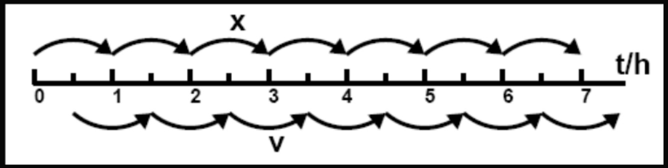
Prędkości :

$$v_{n-\frac{1}{2}} = \frac{r_n - r_{n-1}}{\Delta t} \qquad v_{n+\frac{1}{2}} = \frac{r_{n+1} - r_n}{\Delta t}$$

$$v_{n+\frac{1}{2}} - v_{n-\frac{1}{2}} = \frac{r_{n+1} - 2r_n + r_{n-1}}{\Delta t} = \frac{F_n}{m}\Delta t$$

Stąd:

$$\boxed{v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + \left(\frac{F_n}{m}\right)\Delta t \qquad r_{n+1} = r_n + v_{n+\frac{1}{2}}\Delta t}$$



1. Użyj r_n aby wyliczyć F_n
2. Użyj F_n i $v_{n-1/2}$ aby wyliczyć $v_{n+1/2}$
3. Użyj r_n i $v_{n+1/2}$ aby wyliczyć r_{n+1}

Szacowanie prędkości w chwili t $\qquad v_n = \frac{1}{2}\left(v_{n+\frac{1}{2}} + v_{n-\frac{1}{2}}\right)$

(niekonieczna dla działania algorytmu)

Matematycznie równoważny Verletowi...

$$r_{n+1} = r_n + v_{n+1/2}\Delta t$$

$$v_{n+1/2} = \boxed{v_{n-1/2} + \left(\frac{F_n}{m}\right)\Delta t}$$

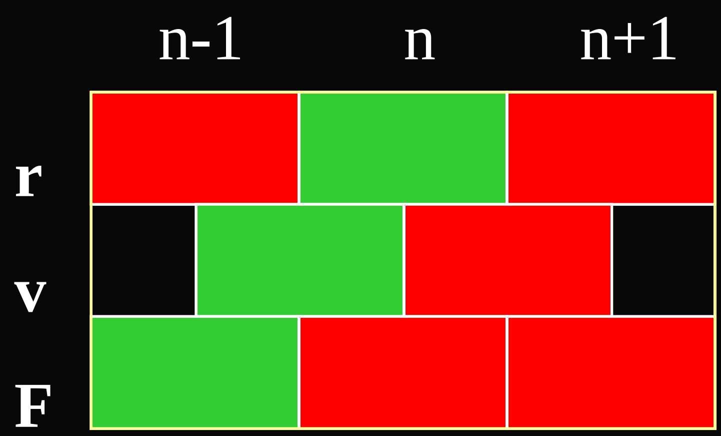
$$r_{n+1} = r_n + \left[v_{n-1/2} + \left(\frac{F_n}{m}\right)\Delta t\right]\Delta t$$

r$_n$ wyliczone z poprzedniego kroku czasowego

$$r_n = r_{n-1} + v_{n-1/2}\Delta t$$

$$r_{n+1} = r_n + \left[r_n - r_{n-1} + \left(\frac{F_n}{m}\right)\Delta t^2\right]$$

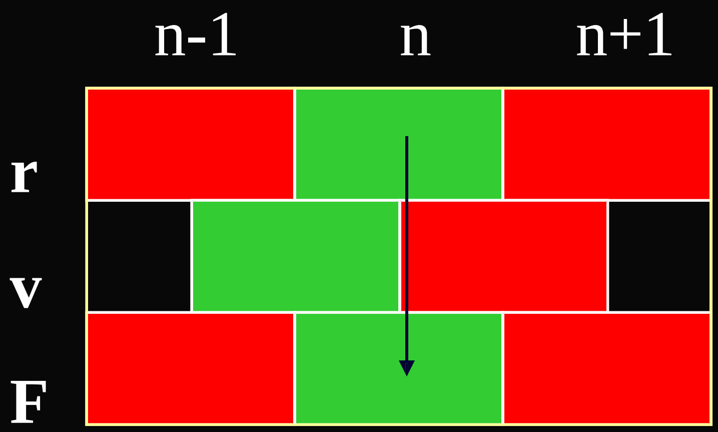
Algorytm skokowy - diagram



Znamy aktualne położenie i prędkość pół kroku wcześniej

$$F_{n-1}, v_{n-1/2}, r_n$$

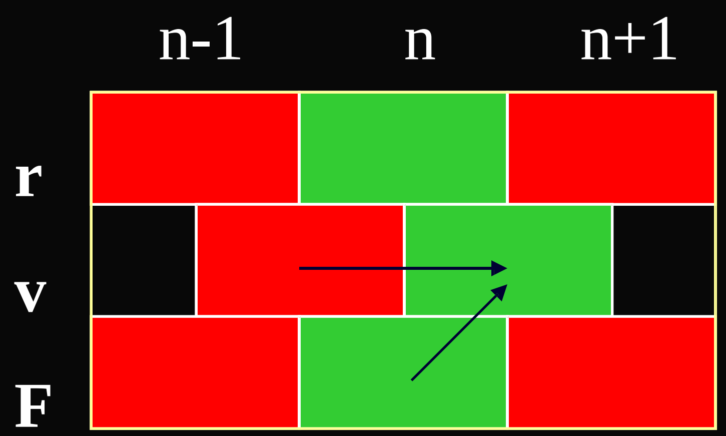
Algorytm skokowy - diagram



Obliczamy aktualną siłę

$$r_n \rightarrow F_n$$

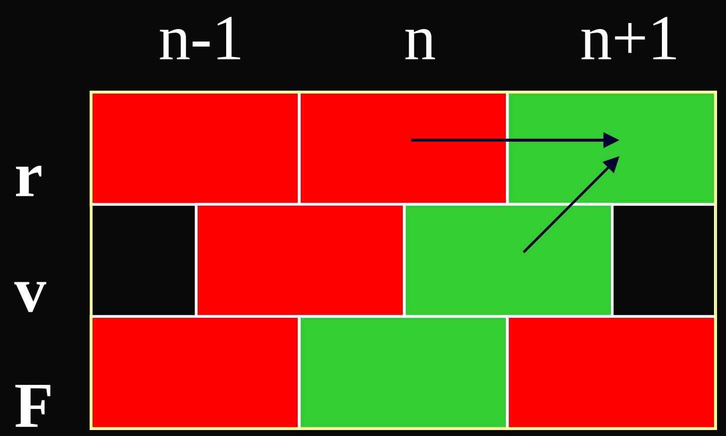
Algorytm skokowy - diagram



Obliczamy prędkość za następne pół kroku

$$v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + \left(\frac{F_n}{m}\right)\Delta t$$

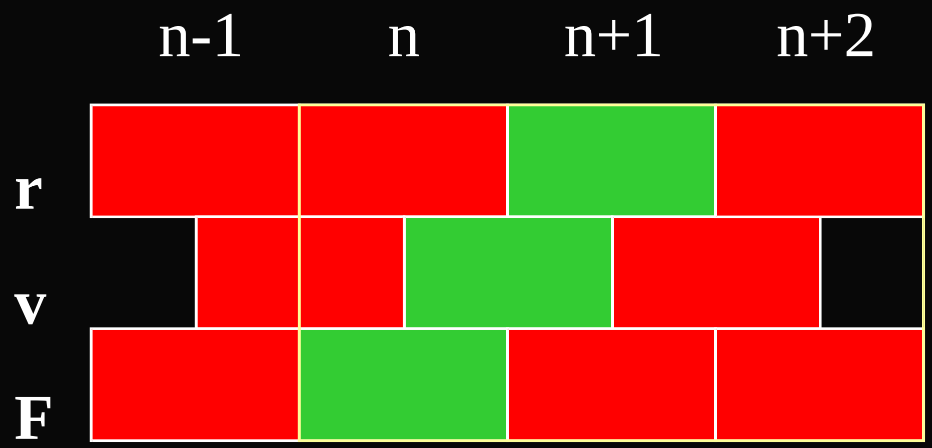
Algorytm skokowy - diagram



Obliczamy następne położenie

$$r_{n+1} = r_n + v_{n+\frac{1}{2}} \Delta t$$

Algorytm skokowy - diagram



Przesuwamy wszystko o jeden krok i powtarzamy

$$F_n, v_{n+1/2}, r_{n+1}$$

Algorytm skokowy – różności

Inicjalizacja

–Jak dostać prędkość “przy poprzednim kroku” na samym początku?

–proste przybliżenie:

$$\mathbf{v}(t_0 - \delta t / 2) = \mathbf{v}(t_0) - \tfrac{1}{m}\mathbf{F}(t_0)\tfrac{1}{2}\delta t$$

Otrzymywanie prędkości w pełnych krokach czasowych

–interpolacja:

$$\mathbf{v}(t) = \frac{1}{2}\left[\mathbf{v}(t + \tfrac{1}{2}\delta t) + \mathbf{v}(t - \tfrac{1}{2}\delta t)\right]$$

Zalety żabki

- Prędkości obliczane z większą dokładnością
- Bezpośrednie obliczanie prędkości ułatwia kontrolę temperatury w symulacji
- Zredukowany problem szumu numerycznego obecny w algorytmie Verleta. W żabce człony $O(dt^1)$ są dodawane do członów $O(dt^0)$

Wady żabki

- Położenia i prędkości wciąż wyliczane w różnych chwilach czasu, co komplikuje analizę wyników
- Trochę bardziej kosztowny numerycznie niż Verlet

ok – wygląda nieźle, ale...

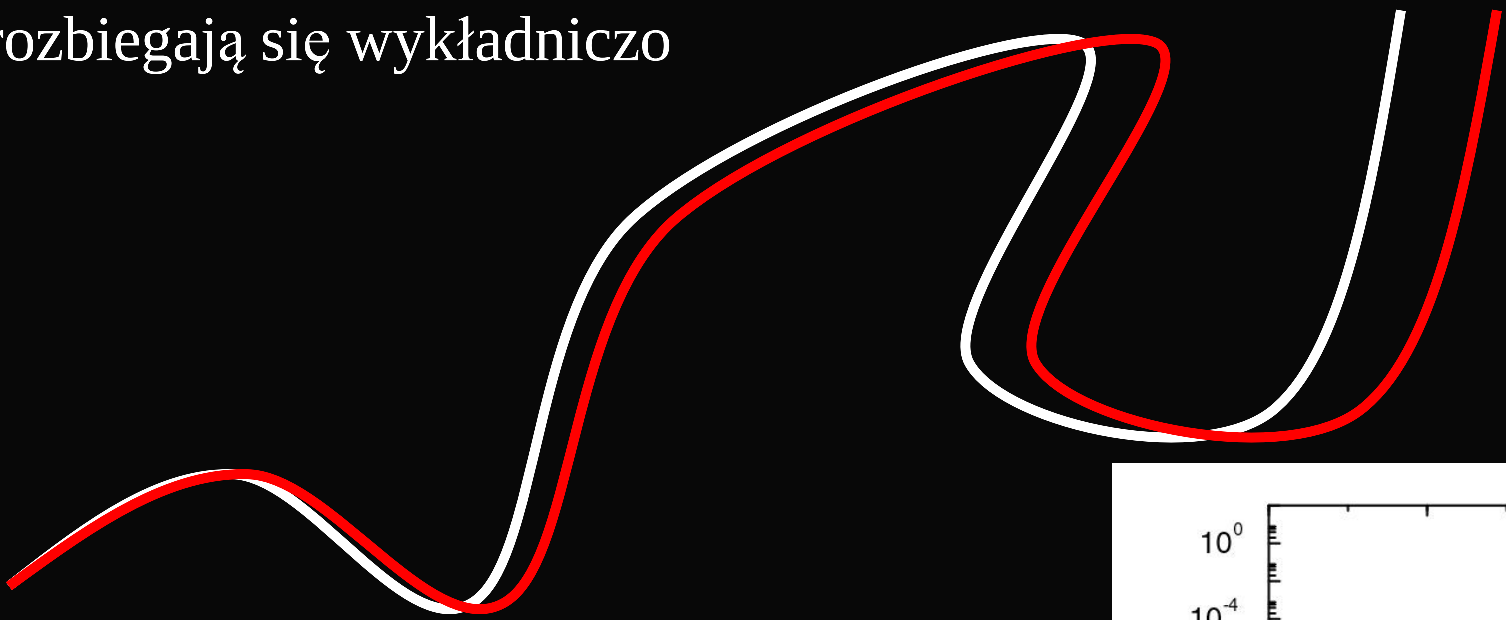
...czy wiernie odtwarza trajektorię układu?

NIE!

Niestabilność Liapunowa

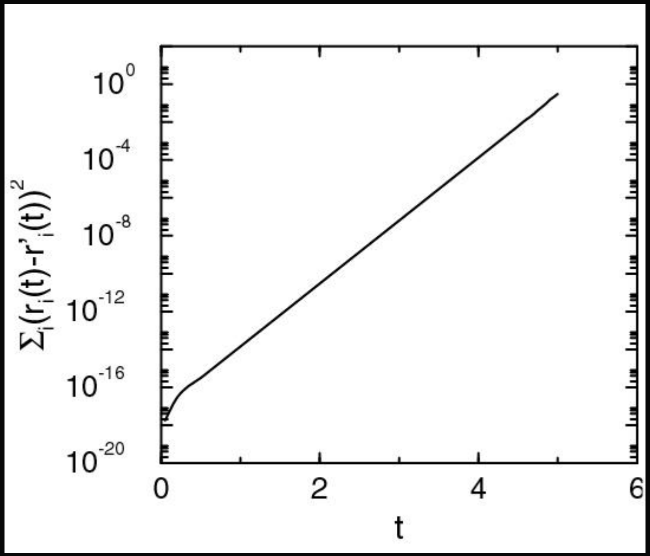
Trajektorie o minimalnie różnych warunkach początkowych rozbiegają się wykładniczo

$t = \tau$

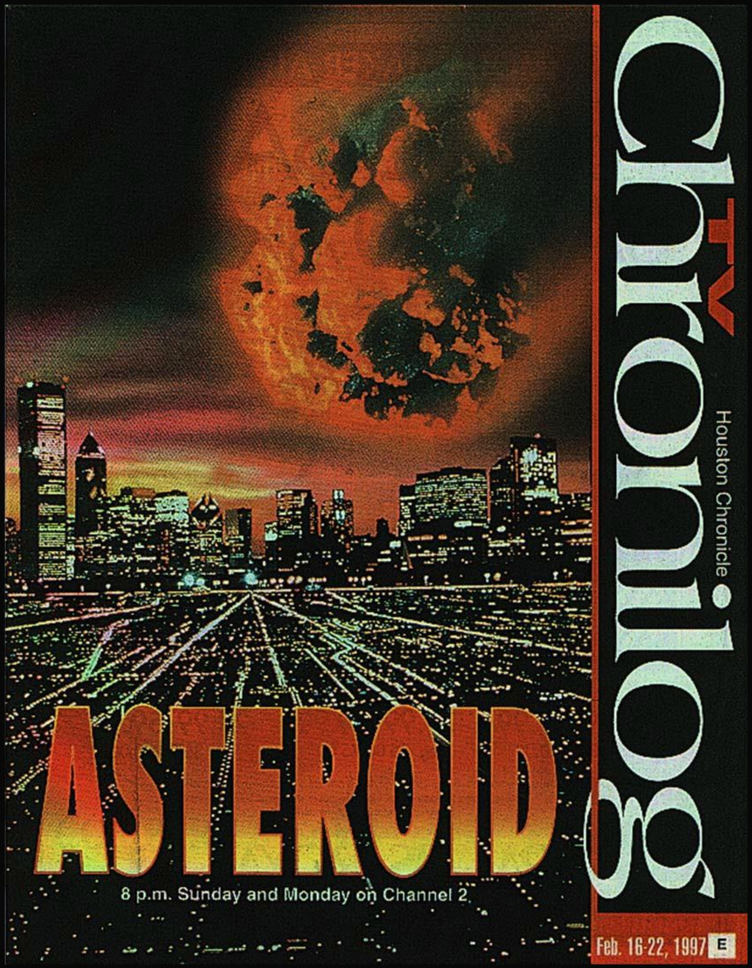


$t=0$

Każdy, nawet najmniejszy błąd numeryczny będzie narastał wykładniczo – dla każdego algorytmu!



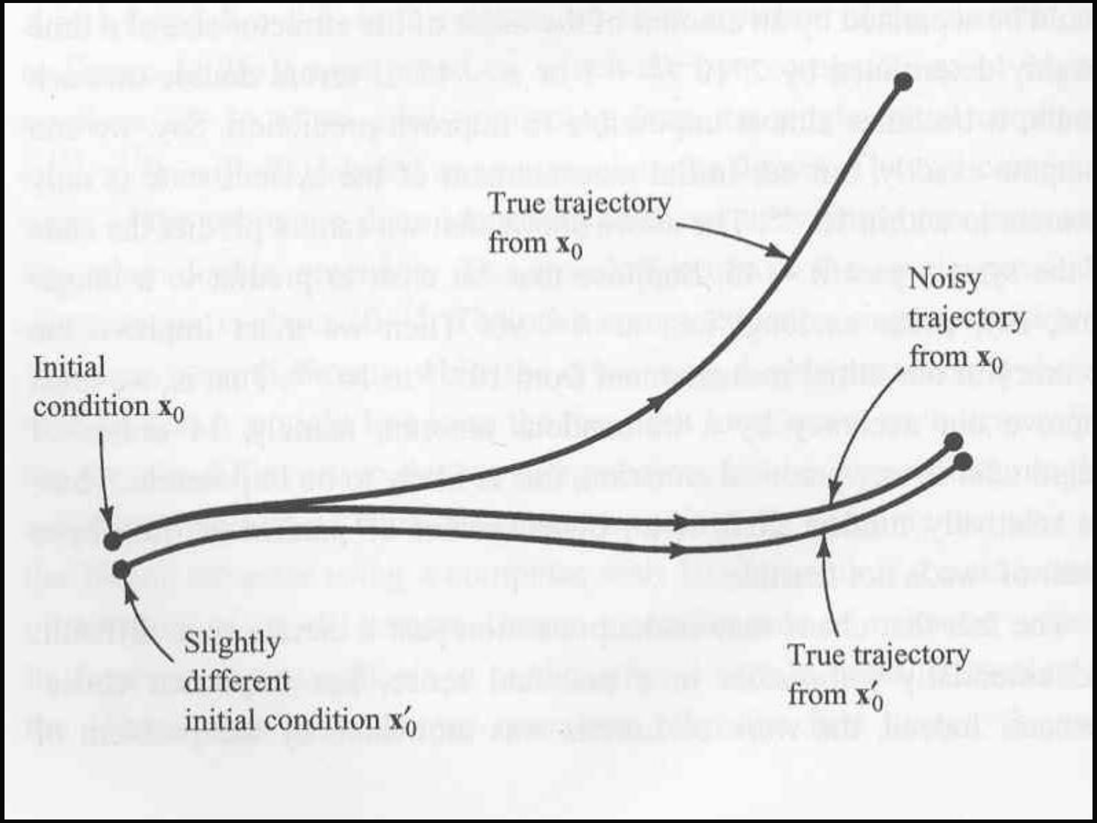
Duże znaczenie w przypadku apokalipsy...



Czy sprawa jest beznadziejna?

Twierdzenie o cieniach

Chociaż wyliczona numerycznie trajektoria układu chaotycznego rozbiega się wykładniczo od ‘prawdziwej’ trajektorii o tych samych warunkach początkowych, to istnieje inna trajektoria układu z nieco innymi warunkami początkowymi która pozostaje w bezpośredniej bliskości trajektorii numerycznej



Z punktu widzenia badania dynamiki układu często nie ma znaczenia którą z tych trajektorii podążamy

Algorytm prędkościowy Verleta

Położenia, prędkości i siły przechowywane są w tej samej chwili czasu t:

$$r_{n+1} = r_n + v_n \Delta t + \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2$$

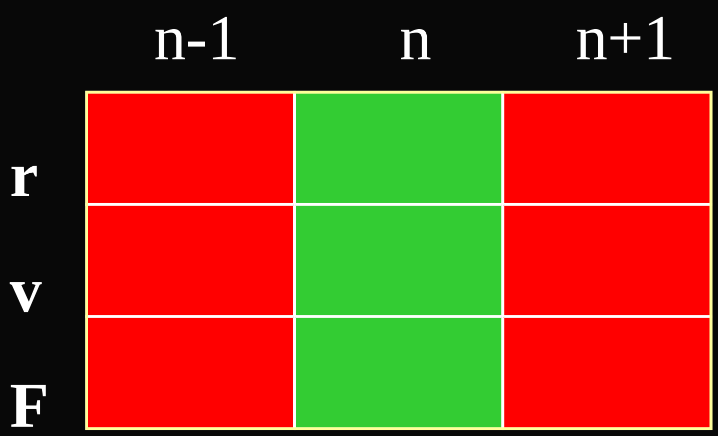
W algorytmie wykorzystuje się też prędkość w połowie kroku:

$$v_{n+1/2} = v_n + \frac{1}{2}\frac{F_n}{m}\Delta t$$

Po czym oblicza się F_{n+1}, a potem kolejne *v*:

$$v_{n+1} = v_{n+1/2} + \frac{1}{2}\frac{F_{n+1}}{m}\Delta t$$

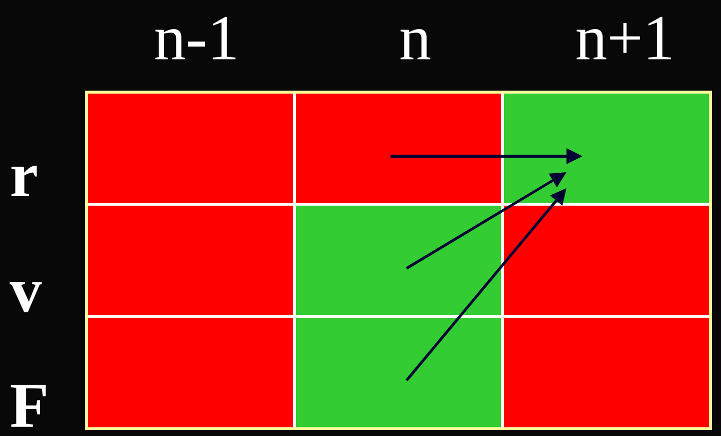
Algorytm prędkościowy - diagram



Znamy aktualne położenie, prędkość i siłę

$$F_n, v_n, r_n$$

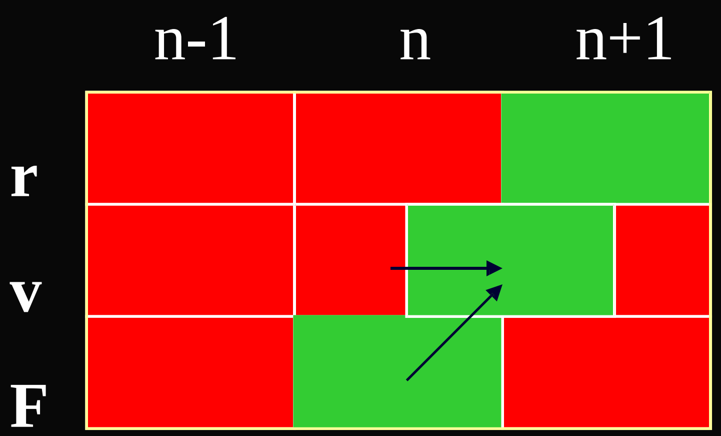
Algorytm prędkościowy - diagram



Obliczamy następne położenie

$$r_{n+1} = r_n + v_n \Delta t + \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2$$

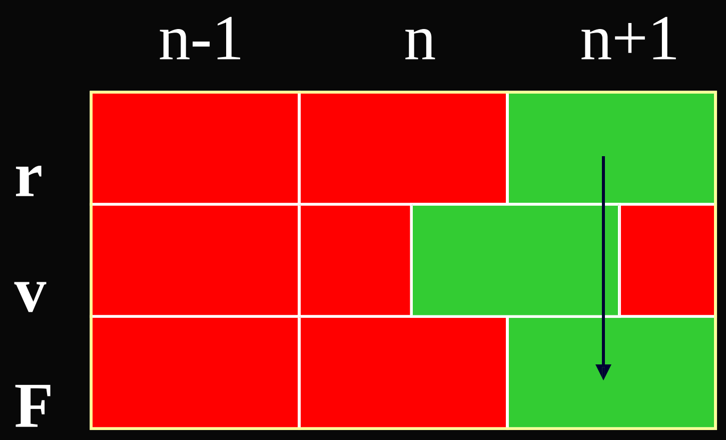
Algorytm prędkościowy - diagram



Obliczamy prędkość za pół kroku

$$v_{n+1/2} = v_n + \frac{1}{2}\frac{F_n}{m}\Delta t$$

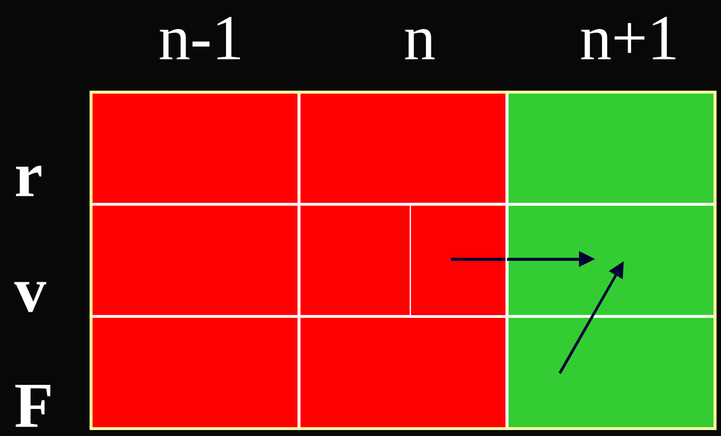
Algorytm prędkościowy - diagram



Obliczamy siłę w nowym położeniu

$$r_{n+1} \rightarrow F_{n+1}$$

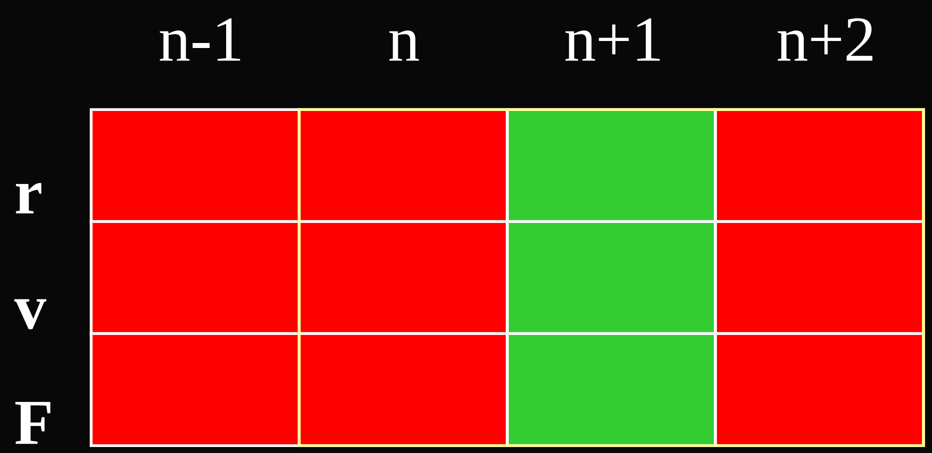
Algorytm prędkościowy - diagram



Obliczamy prędkość w następnym (pełnym) kroku

$$v_{n+1} = v_{n+1/2} + \frac{1}{2}\frac{F_{n+1}}{m}\Delta t$$

Algorytm prędkościowy - diagram



Przesuwamy się o jeden krok czasowy i powtarzamy

$$F_{n+1}, v_{n+1}, r_{n+1}$$

I znowu matematycznie równoważny Verletowi...

$$r_{n+1} = r_n + v_n \Delta t + \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2 \quad \Rightarrow \quad r_n = r_{n+1} - v_n \Delta t - \frac{1}{2}\left(\frac{F_n}{m}\right)\Delta t^2$$

$$r_{n+2} = r_{n+1} + v_{n+1}\Delta t + \frac{1}{2}\left(\frac{F_{n+1}}{m}\right)\Delta t^2$$

dodając

$$r_{n+2} + r_n = 2r_{n+1} + \left(v_{n+1} - v_n\right)\Delta t + \frac{1}{2}\left(\frac{F_{n+1} - F_n}{m}\right)\Delta t^2$$

$$v_{n+1} = v_{n+1/2} + \frac{1}{2}\frac{F_{n+1}}{m}\Delta t = v_n + \frac{1}{2}\left(\frac{F_{n+1} + F_n}{m}\right)\Delta t$$

$$v_{n+1/2} = v_n + \frac{1}{2}\frac{F_n}{m}\Delta t$$

$$\Downarrow$$

$$r_{n+2} + r_n = 2r_{n+1} + \frac{F_{n+1}}{m}\Delta t^2$$

Verlet!

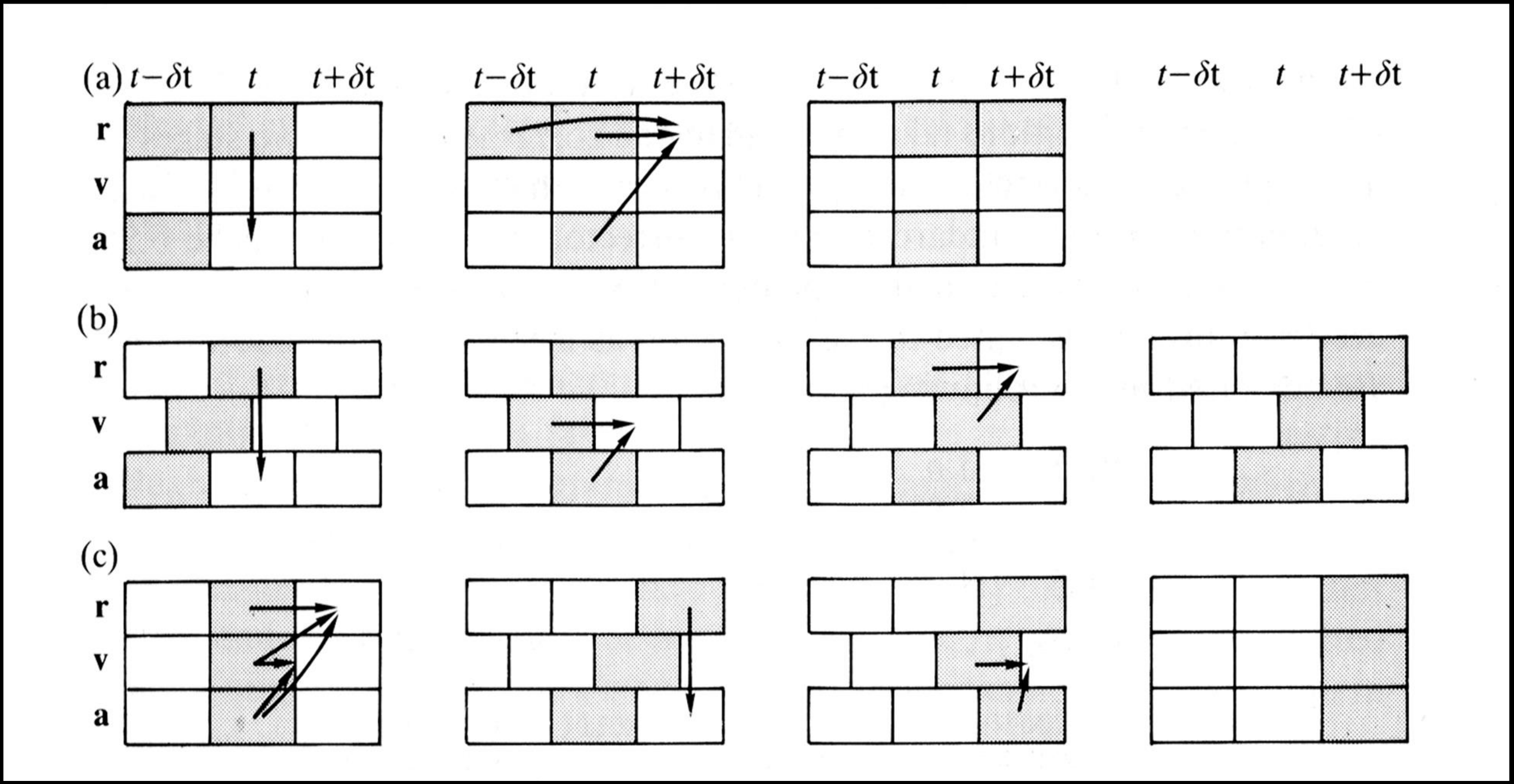
Zalety prędkościowego Verleta

- Położenia i prędkości znane jednocześnie (w tych samych chwilach)
- Początek symulacji – naturalny

Wady

- Bardziej kosztowny numerycznie niż inne wersje Verleta (trzy operacje na cykl)

Verlet: podsumowanie



(a): Verlet (b): krokowy (żabka) (c): prędkościowy