Laboratorio di Fisica Computazionale

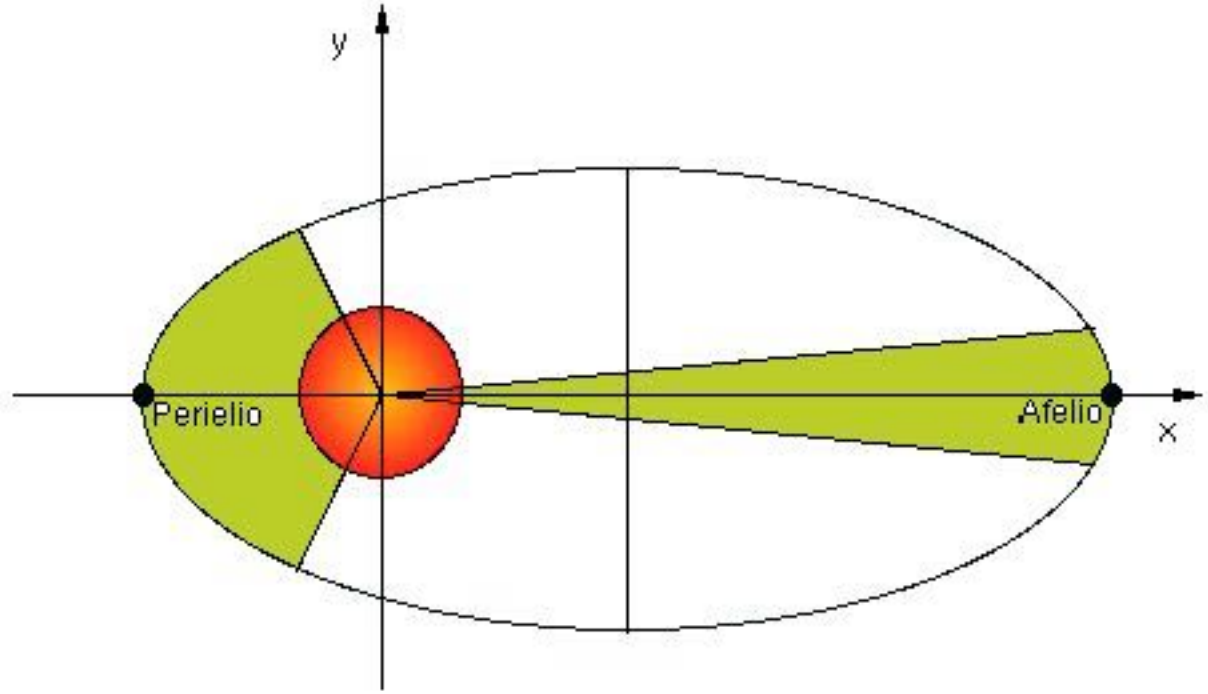
FI02004-4

Gravitazione e leggi di Keplero

Maria Peressi
Università degli Studi di Trieste - Dipartimento di Fisica
Sede di Miramare (Strada Costiera 11, Trieste)
e-mail: peressi@units.it

(molto materiale è frutto di una lunga collaborazione con il collega prof. G. Pastore che ringrazio!)

Leggi di Keplero



1) Ogni pianeta si muove su un piano su un'orbita ellittica con il sole su uno dei fuochi.

2) La velocita' di un pianeta cresce quando questo si avvicina al sole, in modo da coprire aree uguali in tempi uguali.

3) Se T e' il periodo e a il semiasse maggiore dell'ellisse, il rapporto T^2/a^3 e' lo stesso per tutti i pianeti che orbitano attorno al sole.

Domanda:
quanto bene è verificata la terza legge?

Pianeta	Mercurio	Venere	Marte	Giove	Saturno	Urano	Nettuno
Semiasse maggiore (10^6 km)	57,91	108,21	227,92	778,57	1433,53	2872,46	4495,06
Periodo orbitale (giorni)	87,969	224,701	686,980	4332,589	10759,22	30685,4	60189

Il nostro scopo

Dalla legge di Newton e la legge di gravitazione universale

$$\vec{F} = m_{pianeta}\vec{a} \qquad F = |\vec{F}| = G\frac{m_{pianeta}M_{sole}}{r^2}$$

arrivare alle leggi di Keplero, cioè alla traiettoria (f(x,y)) e eventualmente anche alla legge oraria (x(t), y(t))

Come fare a risolvere le equazioni differenziali? due strade:

1) analitica (esatta), quando possibile

2) numerica (approssimata), discretizzando le equazioni

Il nostro scopo

Dalla legge di Newton e la legge di gravitazione universale

$$\vec{F} = m_{pianeta}\vec{a} \qquad F = |\vec{F}| = G\frac{m_{pianeta}M_{sole}}{r^2}$$

arrivare alle leggi di Keplero, cioè alla traiettoria (f(x,y)) e eventualmente anche alla legge oraria (x(t), y(t))

Come fare a risolvere le equazioni differenziali? due strade:

1) analitica (esatta), quando possibile

2) numerica (approssimata), discretizzando le equazioni

Ricordiamo l'idea di base della discretizzazione:
se F = costante => moto uniformemente accelerato

L'algoritmo più semplice

Solita equazione del moto uniformemente accelerato, ma riferita all'intervallino di tempo $t \div t + \Delta t$, che va ripetutamente applicata da un intervallo a quello successivo **(iterazione)**.

algoritmo di EULERO

iterare

$$\boxed{\begin{aligned} x(t+\Delta t) &= x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 \\ v(t+\Delta t) &= v(t) + a(t)\Delta t \end{aligned}}$$

$$x(t) \Longrightarrow x(t+\Delta t) \Longrightarrow x(t+2\Delta t) \Longrightarrow x(t+3\Delta t) \Longrightarrow \ldots$$

$$v(t) \Longrightarrow v(t+\Delta t) \Longrightarrow v(t+2\Delta t) \Longrightarrow v(t+3\Delta t) \Longrightarrow \ldots$$

Un algoritmo migliore

La forza gravitazionale e di conseguenza l'accelerazione dipendono solo dalla posizione*; possibile utilizzare questo:

algoritmo di Velocity-VERLET

iterare

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$
$$v(t+\Delta t) = v(t) + \frac{1}{2}\big(a(t) + a(t+\Delta t)\big)\Delta t$$

(vedremo OK per conservazione energia)

* se ci sono forze che dipendono dalla velocità, NON si può usare questo algoritmo

Il nostro scopo

Dalla legge di Newton e la legge di gravitazione universale

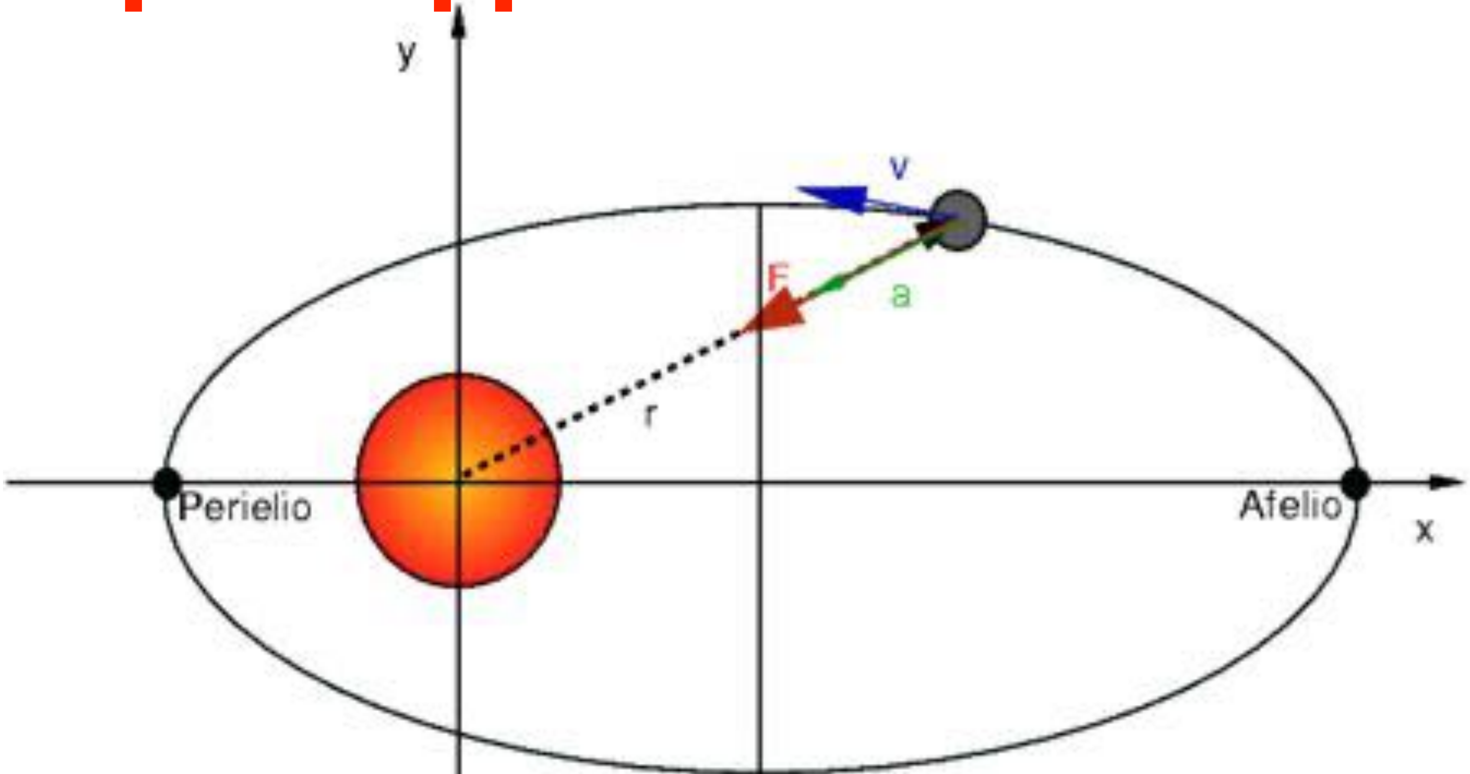
$$\vec{F} = m_{pianeta}\vec{a} \qquad F = |\vec{F}| = G\frac{m_{pianeta}M_{sole}}{r^2}$$

1) arrivare numericamente alle leggi di Keplero.
(cammino inverso rispetto alla storia!)
[per noi -non per studenti- con vari dettagli:
controllo numerico ellisse, conservazione energia...]

2) sperimentare "cosa succede se" la legge di forza fosse diversa

3) estendere l'uso del codice a un sistema a piu' corpi

Un sistema di riferimento per l’approccio numerico

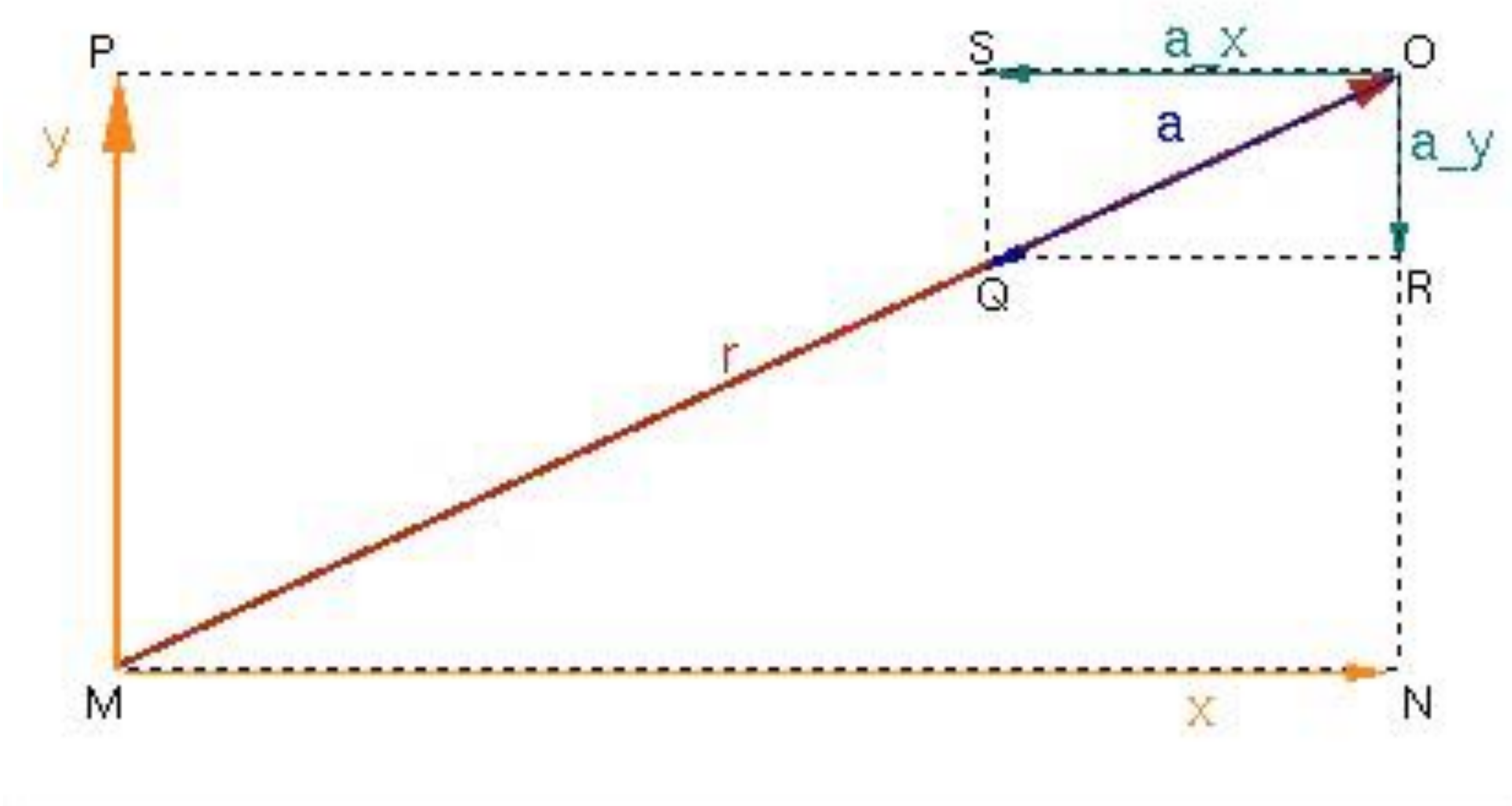


Sole **fisso** all’origine

Grandezze importanti: i **vettori** $\vec{r}$, $\vec{v}$, $\vec{a}$:

$\vec{r}$ **posizione**, congiungente Sole-pianeta;
$\vec{v}$ **velocità**, sempre tangente all’orbita;
$\vec{a}$ **accelerazione**, diretta come $\vec{r}$, ma dal pianeta al Sole;
indica la **variazione** della velocità in modulo (=valore) e/o verso.

Scomposizione del moto in componenti cartesiane



$\vec{r}$ e $\vec{a}$ sono sulla stessa **direzione**. Dalla similitudine dei triangoli MOP e MON rispetto ai triangoli QOR e QOS:

$$a_x = -a\frac{x}{r}, \qquad a_y = -a\frac{y}{r}$$

NB: "$-$" perchè $\vec{a}$ e $\vec{r}$ hanno versi opposti.

Eguagliamo l'espressione di $F = |\vec{F}|$ (modulo) nella legge di Newton

$$F = m_{pianeta} a$$

e in quella di gravitazione universale:

$$F = G \frac{m_{pianeta} M_{sole}}{r^2}$$

ottenendo:

$$\boxed{a = G \frac{M_{sole}}{r^2}}$$

(NB la massa del pianeta non entra nell'espressione dell'accelerazione $\Longrightarrow$ il moto non dipende dalla massa del pianeta, ma solo da quella del sole.)

Per ogni componente del moto:

$$\begin{aligned} a_x &= -G \frac{M_{sole}}{r^2} \cdot \frac{x}{r} = -G \frac{M_{sole} x}{r^3} \\ a_y &= -G \frac{M_{sole}}{r^2} \cdot \frac{y}{r} = -G \frac{M_{sole} y}{r^3} \end{aligned}$$

Dati per Sole-Terra

1.99e30	massa Sole in Kg
3.1558e7	periodo orbita Terra in secondi
1.521e11	pos.x afelio, in m
2.929e4	vel.y, in m/s

Programmi e altro materiale didattico

1) Keplero.zip : Codice **in Java**, da decomprimere, compilare e eseguire:
una possibilita' e' con Bluej (lanciare BlueJ da Applicazioni
e dal menu Project => Open Project => selezionare la cartella "keplero";
Tasto destro del mouse sull'icona "Keplero", seleziona "void(main[Strings..", ok.
(lancia Keplero.class)

2) altri programmi **(in Fortran**: keplero.f90, ellisse.f90), da compilare e poi lanciare

$ gfortran keplero.f90 [⏎] *(o altro compilatore fortran)*

e poi

$ ./a.out [⏎] *(lancia l'eseguibile)*

macro per visualizzazione delle energie dal prog. java: plot-keplero-java ; fare:
$gnuplot> load 'plot-keplero-java' [⏎]

Overview del progetto Keplero in Java

Calcola e visualizza il moto di un pianeta (in generale di un oggetto orbitante) attorno ad una stella considerata fissa.

La parte di calcolo è concentrata nella classe **MotoPianeta.**

L'interfaccia grafica utente è descritta nella classe **Keplero** (la quale implementa anche il metodo “main” necessario per avere un programma eseguibile)

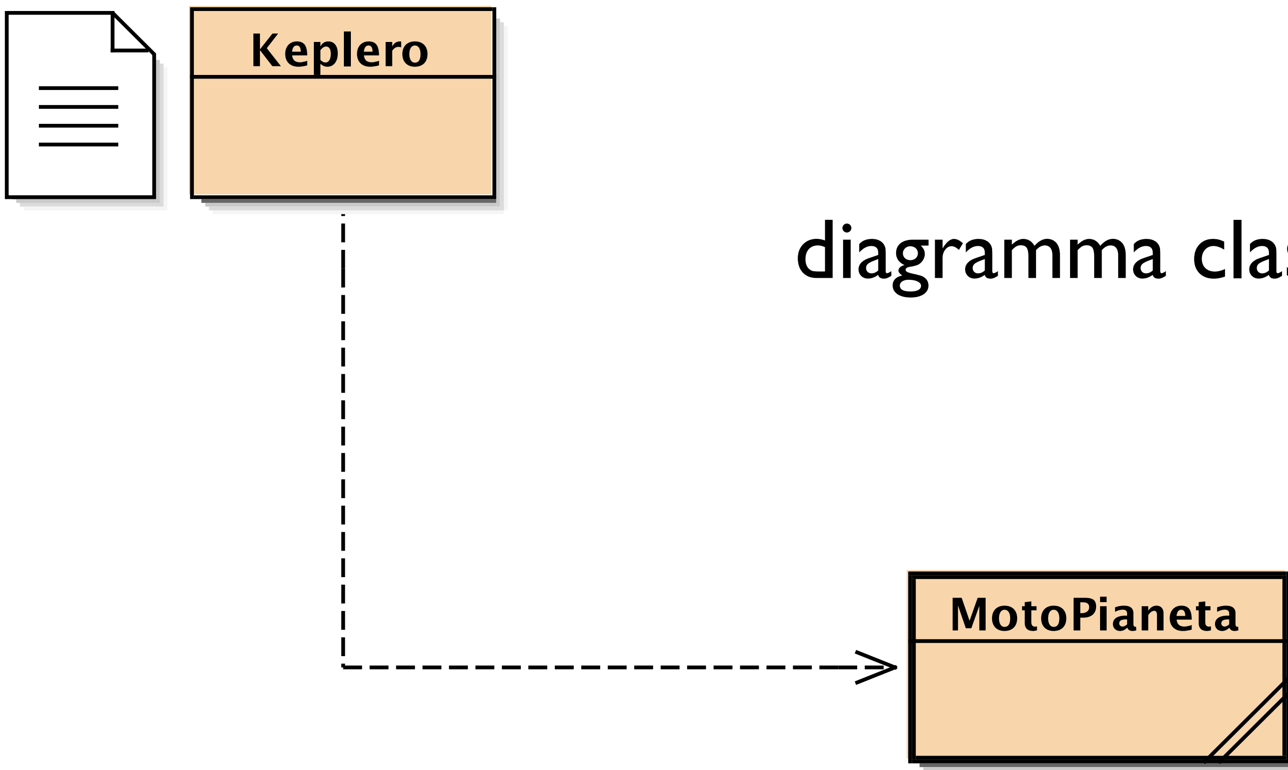
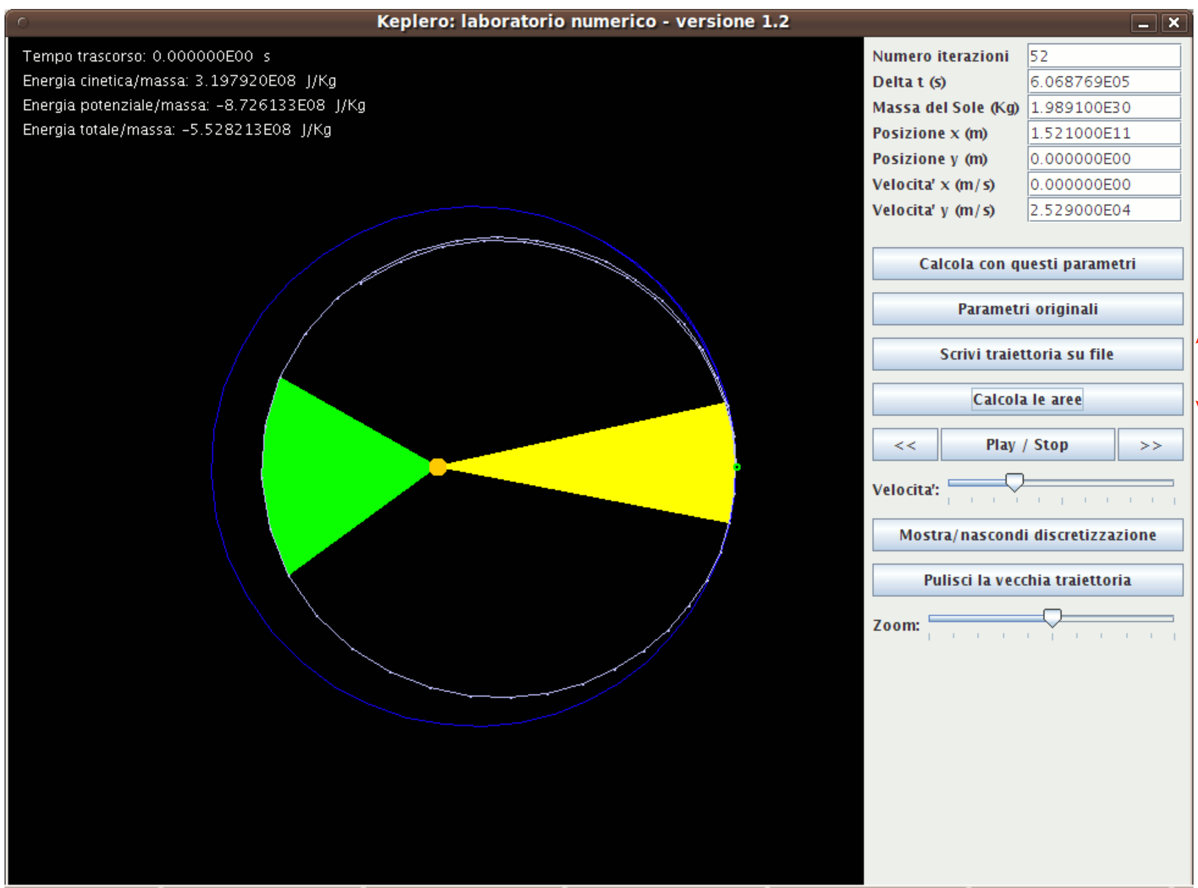


diagramma classi

Overview del progetto Keplero in Java

I vari campi dell’interfaccia grafica consentono di **modificare i parametri** del calcolo, fisici e numerici per la discretizzazione (predefiniti: Terra intorno al Sole, con discretizzazione ad una settimana).
I pulsanti permettono di effettuare una **animazione** dell'orbita, di **cambiare i parametri** di visualizzazione, di **scrivere su file**, di **ripristinare i parametri predefiniti**.
Il programma <u>consente di mostrare le ultime due traiettorie calcolate</u>, così da confrontare direttamente il risultato di valori fisici diversi (nell'esempio qui riportato, in blu scuro la traiettoria della Terra e in chiaro con una velocità iniziale ridotta).

Le quantità fisiche più rilevanti (energie) sono visualizzate all'interno dell'area del grafico.



E’ possibile esportare su file la traiettoria, incluso la velocità, l'accelerazione e l'energia cinetica e potenziale.

Un pulsante consente di effettuare il calcolo delle aree, mostrando anche graficamente l'uguaglianza dell'area spazzata in tempi uguali.

Cosi' e' implementato il calcolo dell'accelerazione in `MotoPianeta.java`, sia per i valori iniziali:

```
// Imposta le condizioni iniziali
pos_x[0] = _pos0x;
pos_y[0] = _pos0y;

double r =
Math.sqrt(pos_x[0]*pos_x[0]+pos_y[0]*pos_y[0]);

acc_x[0] = -G*massaSole*pos_x[0]/Math.pow(r,3);
acc_y[0] = -G*massaSole*pos_y[0]/Math.pow(r,3);
```

$$
\begin{aligned}
a_x &= -G\frac{M_{sole}}{r^2}\cdot\frac{x}{r} = -G\frac{M_{sole}x}{r^3} \\
a_y &= -G\frac{M_{sole}}{r^2}\cdot\frac{y}{r} = -G\frac{M_{sole}y}{r^3}
\end{aligned}
$$

che per quelli nel generico istante di tempo "`i`" o "`i+1`":

```
acc_x[i+1] = -G*massaSole*pos_x[i+1]/Math.pow(r,3);
acc_y[i+1] = -G*massaSole*pos_y[i+1]/Math.pow(r,3);
```

Cosi' e' implementato l'algoritmo di Verlet per il calcolo di posizioni e velocita' in `MotoPianeta.java`:

```
// Integra numericamente l'equazione del moto (Verlet)

for (int i=0;i<niter-1;i++) {
```

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$

```
    pos_x[i+1] = pos_x[i] + vel_x[i]*dt + 0.5*acc_x[i]*dt*dt;
    pos_y[i+1] = pos_y[i] + vel_y[i]*dt + 0.5*acc_y[i]*dt*dt;

    r = Math.sqrt(pos_x[i+1]*pos_x[i+1]+pos_y[i+1]*pos_y[i+1]);
```

Avendo calcolato $x(t+\Delta t)$ ora posso calcolare anche $a(t+\Delta t)$:

```
    acc_x[i+1] = -G*massaSole*pos_x[i+1]/Math.pow(r,3);
    acc_y[i+1] = -G*massaSole*pos_y[i+1]/Math.pow(r,3);
```

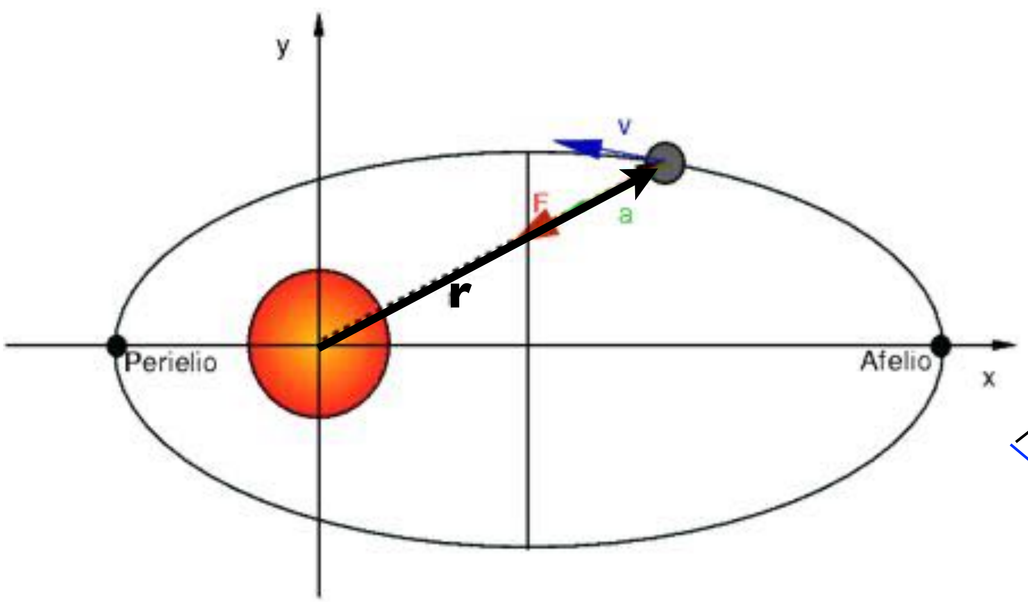
$$v(t+\Delta t) = v(t) + \frac{1}{2}\big(a(t) + a(t+\Delta t)\big)\Delta t$$

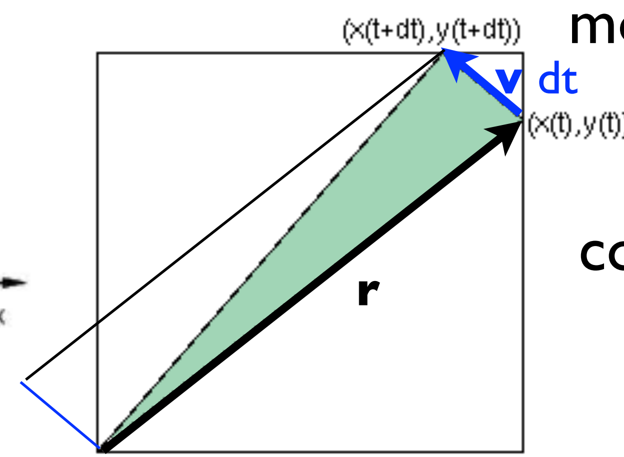
```
    vel_x[i+1] = vel_x[i] + 0.5*(acc_x[i]+acc_x[i+1])*dt;
    vel_y[i+1] = vel_y[i] + 0.5*(acc_y[i]+acc_y[i+1])*dt;
}
```

Cosi' e' implementato il calcolo delle energie e del momento angolare in `MotoPianeta.java`:

```
enCin[i] = 0.5* ( vel_x[i]*vel_x[i] + vel_y[i]*vel_y[i] );
enPot[i] = -G*massaSole/ Math.sqrt( pos_x[i]*pos_x[i] +
                        pos_y[i]*pos_y[i]  );
```

```
dArea[i] = 0.5*dt*Math.abs( pos_x[i]*vel_y[i] -
                  pos_y[i]*vel_x[i]  );
```





area = $|\mathbf{r} \wedge \mathbf{v}|$ dt/2
e' legata al
momento angolare:
$\mathbf{L} = \mathbf{r} \wedge \mathbf{p}$
che dev'essere
costante in quanto
la forza ha
momento nullo

Suggerimento grafico per ricavare la formula dell'area

Overview del codice Keplero in Fortran90

Calcola e NON visualizza il moto. Necessario fare il post-processing dei dati che vengono scritti su file (**pos.out, area.out, en.out**)

Qui si usano variabili con definizione di “tipo” (anziche’ i soliti vettori):

```
type::vett2d                ! def tipo derivato in fortran. Equivalente
                            ! ad un tipo RECORD in Pascal
                            ! puo' anche essere sostituito da un ARRAY
     real(kind=realkind)::x,y
end type

type(vett2d)                :: pos,vel,a,pos_old
```

quindi le componenti sono: pos%x, pos%y etc etc.
e nei cicli iterativi (DO LOOP) vengono aggiornate (sovrascritte) anziche’ considerarne i vari valori etichettati da un indice “i” di iterazione come nel codice java; esempio:

```
pos%x = pos%x + vel%x * dt + 0.5* a%x * dt**2
```

Questo sta per pos_x[i+1] e questo per pos_x[i]

Overview del codice Keplero in Fortran90

L'algoritmo di Verlet per la velocita' richiede l'accelerazione ad uno step e a quello successivo, quindi (siccome sovrascriviamo i valori delle variabili tra un istante di tempo e l'altro) si implementa in due blocchi:

```
pos%x = pos%x + vel%x * dt + 0.5* a%x * dt**2
pos%y = pos%y + vel%y * dt + 0.5* a%y * dt**2

vel%x = vel%x + 0.5 * dt * a%x
vel%y = vel%y + 0.5 * dt * a%y
```

$$v(t+\Delta t) = v(t) + \frac{1}{2}(a(t) + a(t+\Delta t))\Delta t$$

```
! qui le posizioni sono gia' quelle allo step t+dt mentre per le
! velocita' manca ancora il termine con l' accelerazione a t+dt

r = (pos%x**2 + pos%y**2)**0.5
....
vel%x = vel%x + 0.5 * dt * a%x
vel%y = vel%y + 0.5 * dt * a%y
```

$$v(t+\Delta t) = v(t) + \frac{1}{2}(a(t) + a(t+\Delta t))\Delta t$$

Overview di un altro codice Keplero in Fortran90

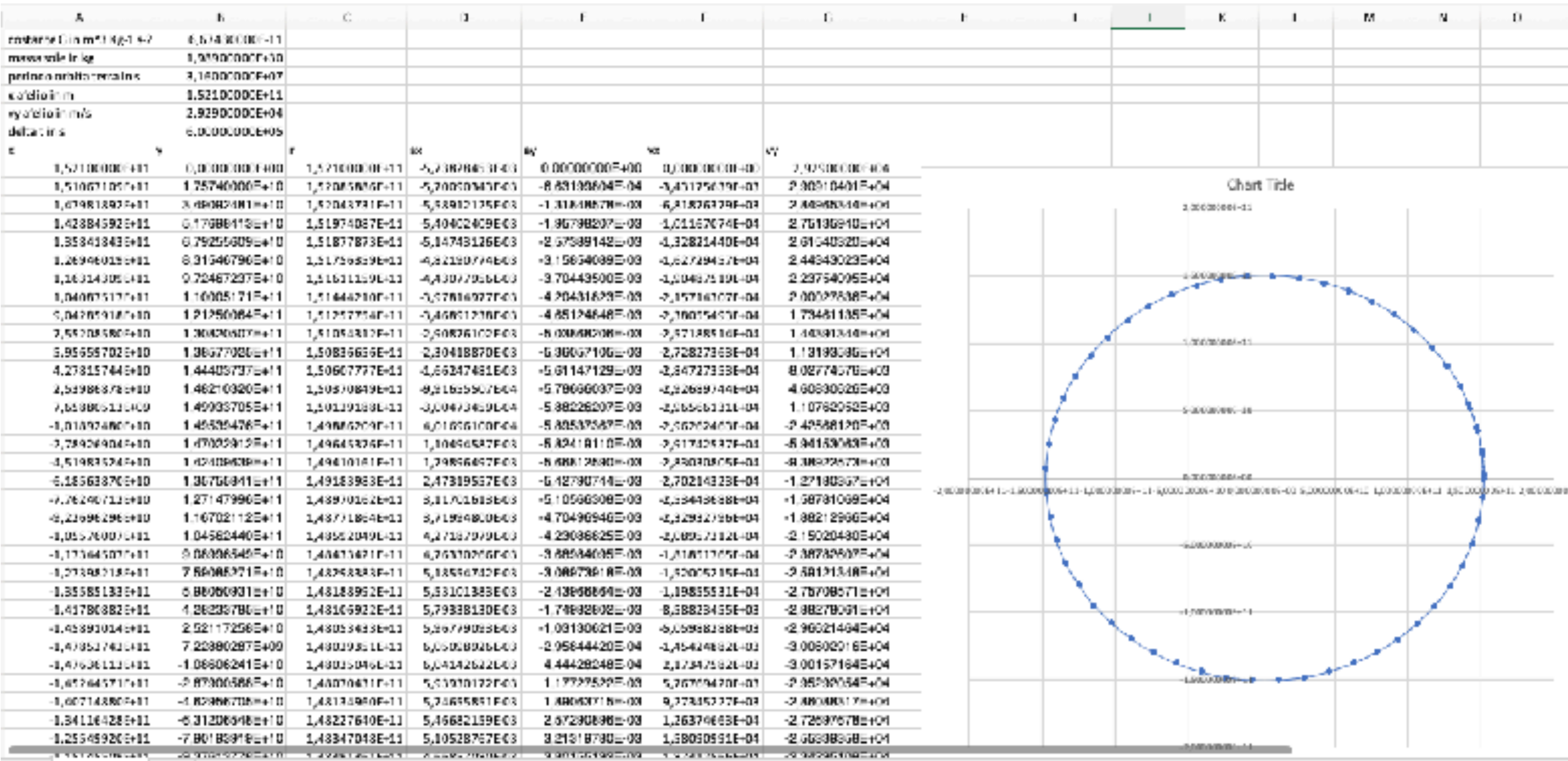
senza utilizzo del tipo “dati” ma con i vettori

utilizza interfaccia

già impostato per 3 corpi

```
real(kind=kr)     :: dt,ekin,epot,dist,ang,ang_prec
real(kind=kr),dimension(3):: L,vcm, posTS
real(kind=kr),dimension(3,nbody):: pos,vel,f
```

... infine:
su un foglio excel!



Il sistema solare - parametri fisici

MERCURY	
Mass (10^24 kg)	0.3302
Semimajor axis (10^6 km)	57.91
Sidereal orbit period (days)	87.969
Perihelion (10^6 km)	46.00
Aphelion (10^6 km)	69.82
Mean orbital velocity (km/s)	47.87
Max. orbital velocity (km/s)	58.98
Min. orbital velocity (km/s)	38.86
Orbit eccentricity	0.2056

VENUS	
Mass (10^24 kg)	4.8685
Semimajor axis (10^6 km)	108.21
Sidereal orbit period (days)	224.701
Perihelion (10^6 km)	107.48
Aphelion (10^6 km)	108.94
Mean orbital velocity (km/s)	35.02
Max. orbital velocity (km/s)	35.26
Min. orbital velocity (km/s)	34.79
Orbit eccentricity	0.0067

EARTH	
Mass (10^24 kg)	5.9736
Semimajor axis (10^6 km)	149.60
Sidereal orbit period (days)	365.256
Perihelion (10^6 km)	147.09
Aphelion (10^6 km)	152.10
Mean orbital velocity (km/s)	29.78
Max. orbital velocity (km/s)	30.29
Min. orbital velocity (km/s)	29.29
Orbit eccentricity	0.0167

MARS	
Mass (10^24 kg)	0.64185
Semimajor axis (10^6 km)	227.92
Sidereal orbit period (days)	686.980
Perihelion (10^6 km)	206.62
Aphelion (10^6 km)	249.23
Mean orbital velocity (km/s)	24.13
Max. orbital velocity (km/s)	26.50
Min. orbital velocity (km/s)	21.97
Orbit eccentricity	0.0935

JUPITER	
Mass (10^24 kg)	1,898.6
Semimajor axis (10^6 km)	778.57
Sidereal orbit period (days)	4,332.589
Perihelion (10^6 km)	740.52
Aphelion (10^6 km)	816.62
Mean orbital velocity (km/s)	13.07
Max. orbital velocity (km/s)	13.72
Min. orbital velocity (km/s)	12.44
Orbit eccentricity	0.0489

SATURN	
Mass (10^24 kg)	568.46
Semimajor axis (10^6 km)	1,433.53
Sidereal orbit period (days)	10,759.22
Perihelion (10^6 km)	1,352.55
Aphelion (10^6 km)	1,514.50
Mean orbital velocity (km/s)	9.69
Max. orbital velocity (km/s)	10.18
Min. orbital velocity (km/s)	9.09
Orbit eccentricity	0.0565

URANUS	
Mass (10^24 kg)	86.832
Semimajor axis (10^6 km)	2,872.46
Sidereal orbit period (days)	30,685.4
Perihelion (10^6 km)	2,741.30
Aphelion (10^6 km)	3,003.62
Mean orbital velocity (km/s)	6.81
Max. orbital velocity (km/s)	7.11
Min. orbital velocity (km/s)	6.49
Orbit eccentricity	0.0457

NEPTUNE	
Mass (10^24 kg)	102.43
Semimajor axis (10^6 km)	4,495.06
Sidereal orbit period (days)	60,189.
Perihelion (10^6 km)	4,444.45
Aphelion (10^6 km)	4,545.67
Mean orbital velocity (km/s)	5.43
Max. orbital velocity (km/s)	5.50
Min. orbital velocity (km/s)	5.37
Orbit eccentricity	0.0113

PLUTO	
Mass (10^24 kg)	0.0125
Semimajor axis (10^6 km)	5906.38
Sidereal orbit period (days)	90,465
Perihelion (10^6 km)	4436.82
Aphelion (10^6 km)	7375.93
Mean orbital velocity (km/s)	4.72
Max. orbital velocity (km/s)	6.10
Min. orbital velocity (km/s)	3.71
Orbit eccentricity	0.2488

Il sistema solare - parametri utili

Pianeta	Periodo rivoluzione	Raggio medio dell'orbita (m)	Afelio (m)	Massa (unità $massa_{terra}$)	Eccentricità $\sqrt{1-(b/a)^2}$
Mercurio	87.97g	5.79×10^{10}	6.97×10^{10}	0.055	0.206
Venere	224.70g	1.08×10^{11}	1.09×10^{11}	0.815	0.007
Terra	365.25g	1.49×10^{11}	1.521×10^{11}	1.0	0.017
Marte	686.98g	2.28×10^{11}	2.491×10^{11}	0.107	0.093
Giove	11.86a	7.78×10^{11}	8.157×10^{11}	317.94	0.048
Saturno	29.46a	1.43×10^{12}	1.507×10^{12}	95.18	0.056
Urano	84.02a	2.87×10^{12}	3.004×10^{12}	14.53	0.046
Nettuno	164.79a	4.50×10^{12}	4.537×10^{12}	17.13	0.010
Plutone	247.70a	5.90×10^{12}	7.375×10^{12}	0.0022	0.248

parametri numerici "ragionevoli" per l'uso del codice

TERRA:

scelta del Δt: una settimana=1/52 del periodo di rivoluzione:

Δt = 6*10^5 sec

(questi infatti i parametri di default :

PREDEF_NITER = 52

PREDEF_DELTAT = 365.25 x 24 x 3600 / PREDEF_NITER)

- questa scelta di Δt è ok per ricostruire l'orbita per il primo periodo, ma non è sufficiente per mantenere una buona accuratezza su simulazioni lunghe

- comunque da sottolineare l'importanza di **FAR SPERIMENTARE agli studenti la dipendenza dei risultati dal parametro Δt**

parametri numerici "ragionevoli" per l'uso del codice

PLUTONE:

scelta del Δt: (sappiamo che) il periodo di rivoluzione e' circa $9 * 10^4$ giorni cioe' $7.8 * 10^9$ sec;

- prendiamo Δt circa 1/50 del periodo di rivoluzione (in analogia con la Terra), $\Delta t = 1.5 * 10^8$ sec => risultati "brutti", anche solo per 10 orbite
- perche'? inadeguato (a causa dell'**eccentricita**'
-discretizzazione male vicino al perielio, meglio all'afelio)

- prendiamo allora Δt circa 1/1000 del periodo di rivoluzione: $\boldsymbol{\Delta t = 10^7}$ **sec** => meglio

Spunti per la sperimentazione numerica

- osservare (orbita chiusa...)
- Δ t
- what-if (se cambiamo i parametri fisici, es. la vel.)?
  niente di qualitativamente nuovo ...
- inserire componenti (x,y), (v_x,v_y): provare a inserire i dati per afelio o perielio e verificare che l'orbita e' la stessa
- legge aree (agli studenti in un secondo momento)
- what-if (se cambiamo la legge di forza)?
  qualitativamente qualcosa di nuovo !

Analisi traiettoria

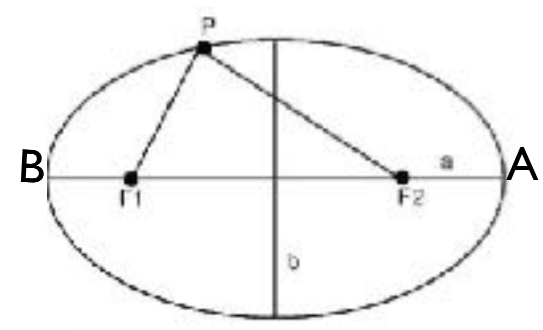
Si puo' verificare numericamente che l'orbita e' un'ellisse?

SI : vedere dati traiettoria su file (traiettoria.txt, 3ª e 4ª colonna)

Via breve: sappiamo (per costruzione) che un fuoco e' $F_2(0,0)$; si e' partiti dal perielio con coordinate $(x_{perielio},0)$, troviamo dal file di posizioni l'afelio $(x_{afelio},0)$; allora il secondo fuoco e' $F_1(x_{afelio}+x_{perielio},0)$ e $2a= x_{perielio}-x_{afelio}$.
Calcolare per tutti i punti della traiettoria $(PF_1+PF_2 -2a)/(2a)$ (variazione relativa)

Vedi: ellisse.f90 (risultati in deviazioni-ellisse.txt)

ellisse: F_1, F_2 punti focali (fissi); $PF_1+PF_2 =$ costante $=2a$
eccentricità: $e = \sqrt{1-(b/a)^2}$; a e b: semiassi maggiore e minore; misura lo "schiacciamento" dell'ellisse (cerchio: $e = 0 \Leftrightarrow a = b$);
perielio: punto di minima distanza dal sole;
afelio: punto di massima distanza dal sole.



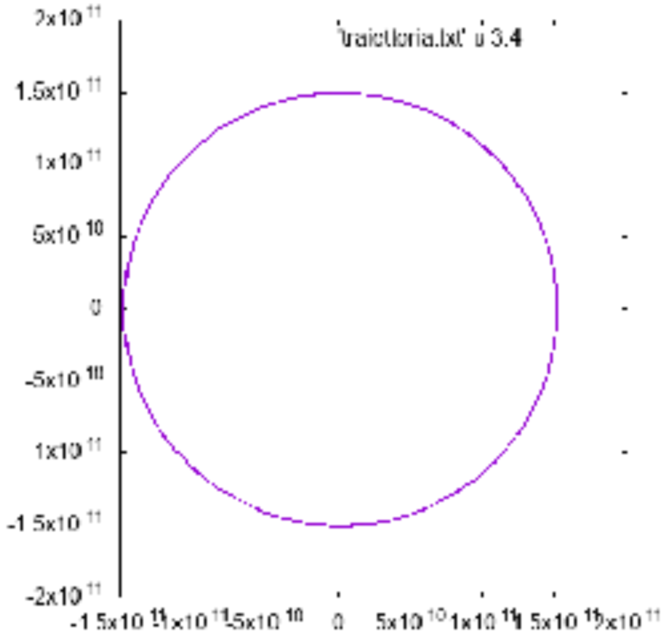
Visualizzazione con gnuplot

```
gnuplot> set size ratio -1
```

(predispone il plot con stessa scale in x e y)

```
gnuplot> p 'traiettoria.txt' u 3:4 w l
```

(plotta i dati dal file indicato usando colonne 3,4 come x,y e unendo i punti con linea spezzata)



(di default visualizza su schermo)

```
gnuplot> set term png
```

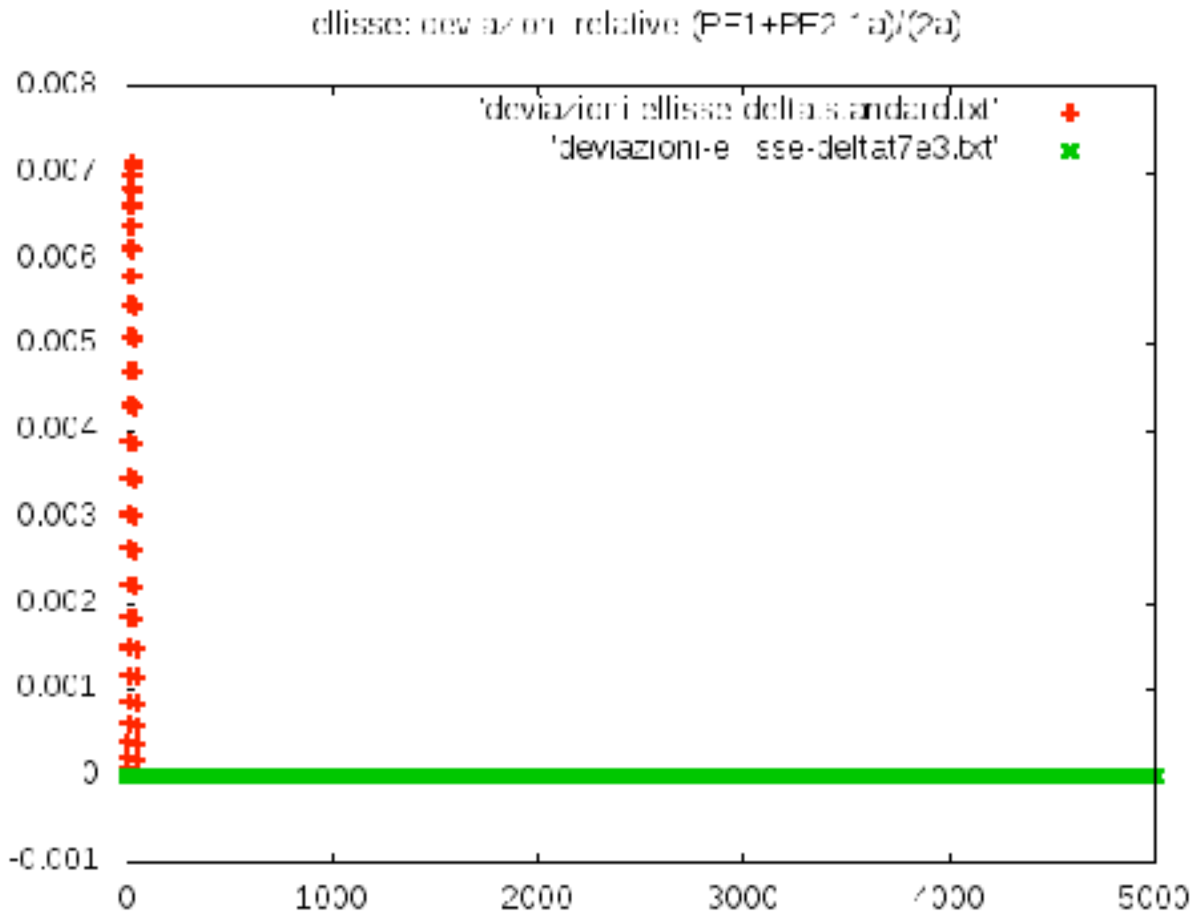
(cosi' non visualizza su schermo ma apre un file di tipo png a altro, se si vuole, ad es. jpeg)

```
Terminal type is now 'png'
Fontconfig warning: ignoring UTF-8: not a valid region tag
Options are 'nocrop enhanced size 640,480 font "arial,12.0" '
gnuplot> set output 'traiettoria.png'
```

(qui viene dato il nome specifico al file dove si salva il grafico)

```
gnuplot> p 'traiettoria.txt' u 3:4 w l
```

Analisi traiettoria



Spunto alternativo per un possibile approfondimento "matematico": verificare che si tratta di un ellisse usando la formula in coordinate polari: $r(1+e\cos\theta)=l$

Un altro approfondimento sulla discretizzazione

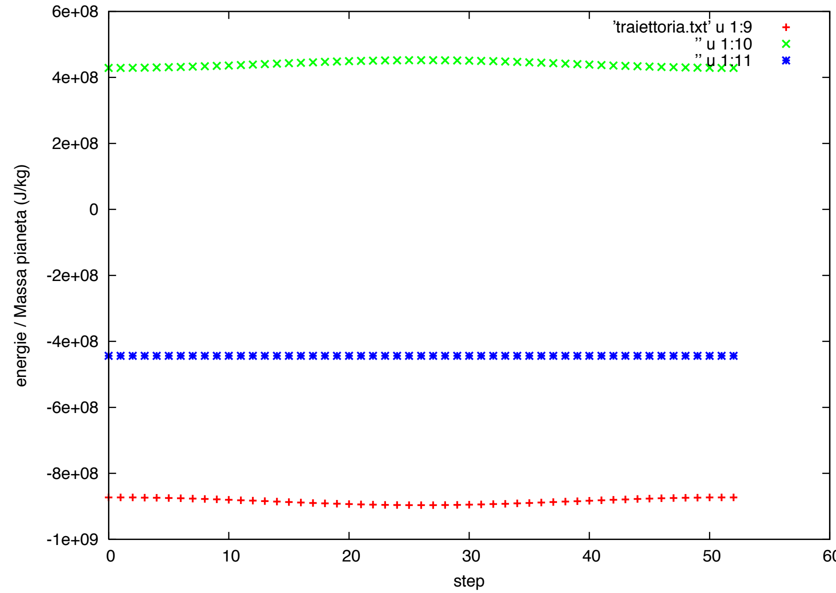
Finora abbiamo considerato FISSA la griglia di discretizzazione, ma puo' essere presa **variabile**, utilizzando un opportuno algoritmo di variazione del passo di integrazione:

esempio: per orbite molto eccentriche (o iperboliche) opportuno considerare un passo variabile (inversamente proporzionale ?) con la distanza dal fuoco

Analisi delle costanti del moto

Si puo' verificare numericamente, in modo quantitativo, che l'energia si conserva?

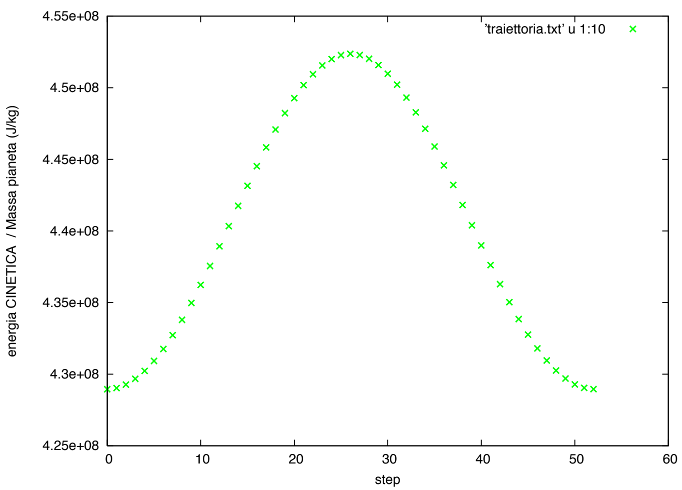
SI : v. traiettoria.txt (ultime colonne) e plot-keplero (macro per gnuplot)

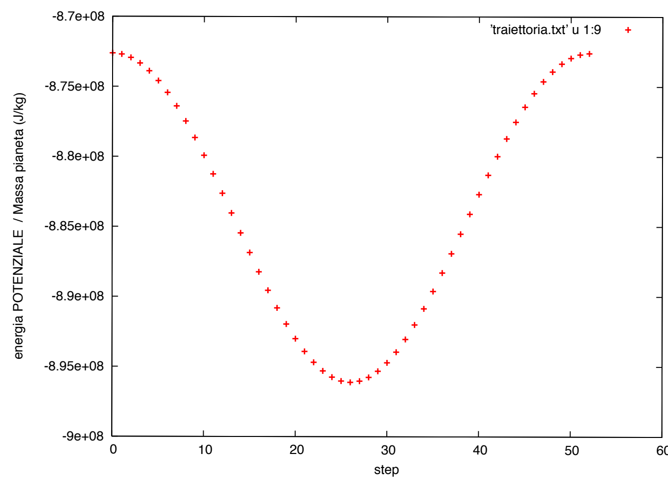


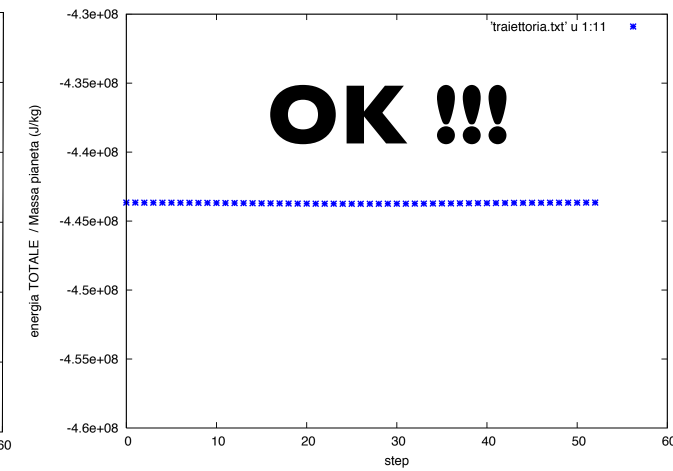
cinetica

potenziale

totale







La legge di gravitazione POCO MODIFICATA

$$\mathbf{F} = \mathbf{G'}\frac{\mathbf{m_{pianeta}M_{sole}}}{\mathbf{r^{(2+\delta)}}}$$

$$a_x = \left(G' \ \frac{M_{sole}}{r^{(2+\delta)}}\right)_x = \frac{G' M_{sole} x}{r^{(3+\delta)}}$$

|δ|<<1

La legge di gravitazione MOLTO MODIFICATA

$$\mathbf{F} = \mathbf{G''}\frac{\mathbf{m_{pianeta}M_{sole}}}{\mathbf{r^{3}}}$$

|δ|=1

$$a_x = \left(G'' \ \frac{M_{sole}}{r^{3}}\right)_x = \frac{G'' M_{sole} x}{r^{4}} \quad (idem\ per\ a_y)$$

Attenzione nel codice alle righe:

```
acc_x[...] = -G*massaSole*pos_x[...]/Math.pow(r,3...);
acc_y[...] = -G*massaSole*pos_y[...]/Math.pow(r,3...);
```

Cerchiamo di trovare (“sperimentalmente”!) quelle condizioni di velocita’ per cui ritroviamo un orbita chiusa....
Quale spiegazione....?

Il pianeta rimane su un’orbita chiusa, per la precisione circolare, se la forza da’ luogo ad un’accelerazione puramente centripeta :

$$G'' \frac{mM}{r^3} = \frac{mv^2}{r} \Longrightarrow v = v_{eq} = \sqrt{\frac{G'' M}{r^2}}$$

Altra possibile modifica della legge di gravitazione

(forza su satellite in orbita equatoriale;
la modifica è dovuta al rigonfiamento equatoriale terrestre)

$$\frac{GM}{r^2}\left[1+\frac{3}{2}J_2\left(\frac{R}{r}\right)^2\right]$$

R = raggio terrestre
J_2 = 1.0826 10^{-3}

Generalizzazione
al problema di 3 o piu' corpi

Metodi Runge-Kutta

Runge-Kutta 4 - integrazione di equazioni differenziali di primo grado.

Come quello di Verlet, e' un metodo "**ad un passo**", cioe' calcola una soluzione partendo da un punto iniziale e la integra procedendo un passo alla volta, e calcolando il valore della soluzione utilizzando solamente i punti che vanno dal nodo t_n al nodo t_{n+1} ; e' **esplicito** (puo' essere implementato attraverso un meccanismo di tipo ricorsivo).

Utile per risolvere $y'(t)= f(t,y(t))$ con la condizione iniziale $y(t_0) = y_0$.
L'eq. F=ma e' del secondo grado per le coordinate, ma ne otteniamo una di primo grado se la scriviamo per la velocita' (e poi integreremo quella)

Utile nel caso di forze dipendenti dalla velocita'

Metodi Runge-Kutta

Esempio: lancio di un corpo in campo gravitazionale in presenza di attrito dell'aria, no rotazione

In qs. caso e' possibile anche la soluzione analitica (utile se vogliamo usare il problema per valutare la bonta' dell'algoritmo numerico)

$$m\frac{dv_x}{dt} = -cv_x$$
$$m\frac{dv_y}{dt} = -mg - cv_y$$

$$v_x(t) = v_{0x} e^{-\frac{c}{m}t}$$
$$v_y(t) = (v_{0y} + \frac{gm}{c})e^{-\frac{c}{m}t} - \frac{gm}{c}$$
$$x(t) = \frac{m}{c} v_{0x}(1 - e^{-\frac{c}{m}t})$$
$$y(t) = \frac{m}{c}(\frac{mg}{c} + v_{0y})(1 - e^{-\frac{c}{m}t}) - \frac{mgt}{c}$$

$$\vec{v}_{n+1} = \vec{v}_n + \frac{h}{6}(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4)$$

con :

$$\vec{K}_1 = \begin{pmatrix} -\frac{c}{m} & 0 \\ 0 & -\frac{c}{m} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \begin{pmatrix} 0 \\ -g \end{pmatrix}$$
$$\vec{K}_2 = \begin{pmatrix} -\frac{c}{m} & 0 \\ 0 & -\frac{c}{m} \end{pmatrix} \left[\begin{pmatrix} v_x \\ v_y \end{pmatrix} + \frac{1}{2}h\vec{K}_1 \right] + \begin{pmatrix} 0 \\ -g \end{pmatrix}$$
$$\vec{K}_3 = \begin{pmatrix} -\frac{c}{m} & 0 \\ 0 & -\frac{c}{m} \end{pmatrix} \left[\begin{pmatrix} v_x \\ v_y \end{pmatrix} + \frac{1}{2}h\vec{K}_2 \right] + \begin{pmatrix} 0 \\ -g \end{pmatrix}$$
$$\vec{K}_4 = \begin{pmatrix} -\frac{c}{m} & 0 \\ 0 & -\frac{c}{m} \end{pmatrix} \left[\begin{pmatrix} v_x \\ v_y \end{pmatrix} + h\vec{K}_3 \right] + \begin{pmatrix} 0 \\ -g \end{pmatrix}$$

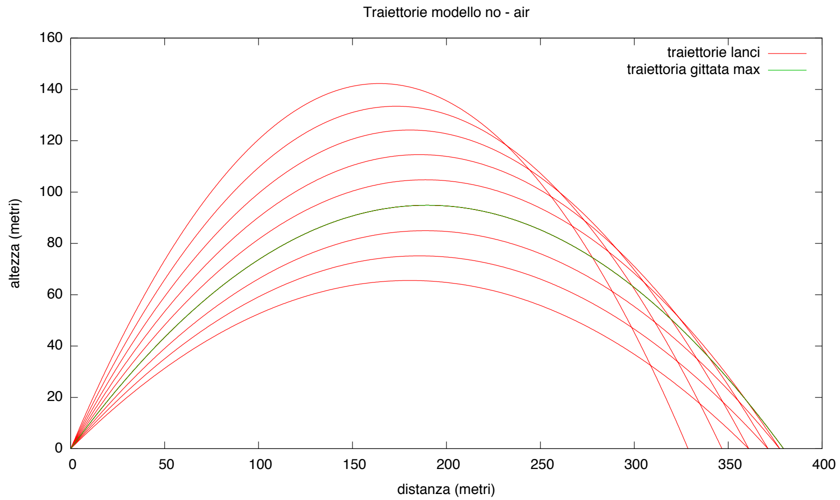


FIGURA 13. **Traiettorie per il modello no - air** Le traiettorie riportate sono per un modello senza aria: si evidenzia la traiettoria di gittata massima che si ottiene per un angolo di 45 gradi, con un valore di ~ 380 metri.

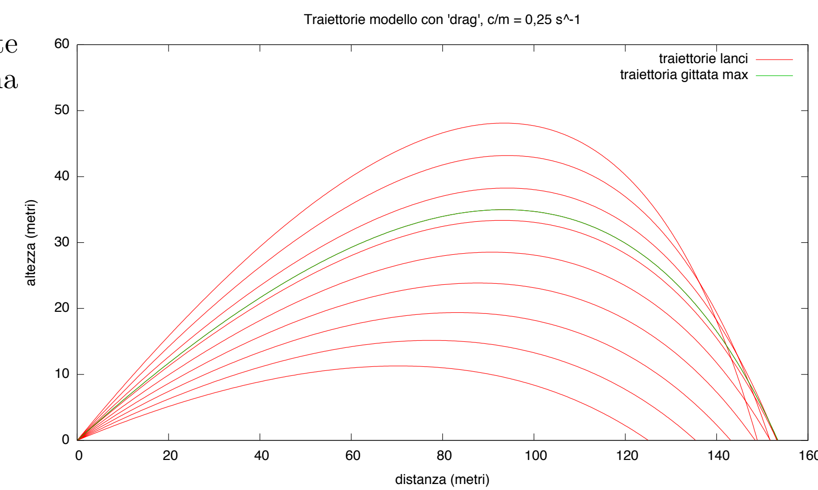


FIGURA 14. **Traiettorie per il modello con sola *drag*** In questo caso le traiettorie risultano notevolmente ridimensionate sia in altezza che in lunghezza con una gittata massima pari a ~ 153 metri. L'angolo di gittata massima si ottiene ora per un angolo pari a 32 gradi