La deuxième loi de Kepler – avec XINT et PSTricks

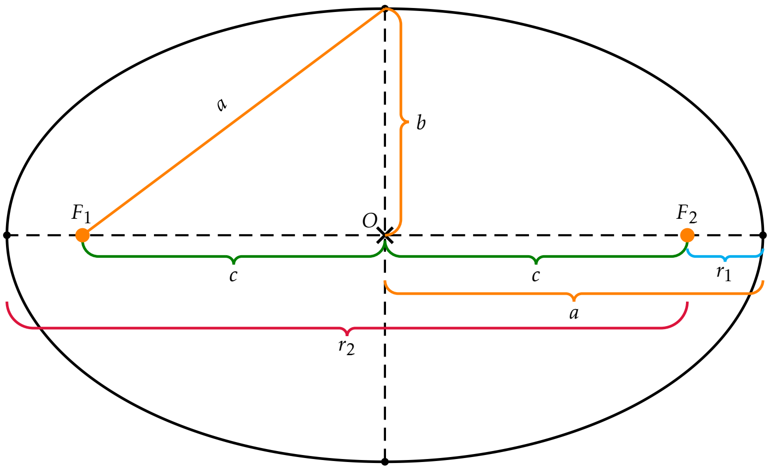
Thomas Söll et Jürgen Gilg

13 décembre 2019

1 Étude mathématique des ellipses

Dans le dessin suivant on présente la syntaxe des paramètres d'une ellipse.

O	centre de l'ellipse
a	demi-grand axe
b	demi-petit axe
c	la distance séparant le centre de l'ellipse de l'un des foyers
e	l'excentricité de l'ellipse : $e = \frac{c}{a}$

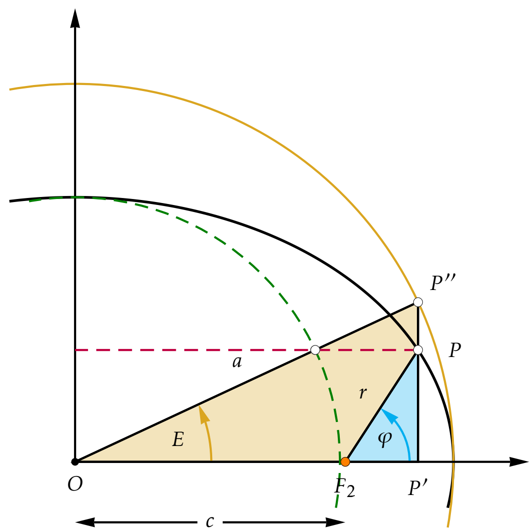


Avec le théorème de Pythagore :

$$c^2 = a^2 - b^2$$

2 Deuxième loi de Kepler – la théorie

La transformation de l'angle E (angle au centre de l'ellipse) avec l'angle φ (l'angle de sommet le foyer de l'ellipse – l'angle polaire).



Pour l'angle polaire φ :

$$\cos(\varphi) = \frac{a \cdot \cos(E) - c}{r} = \frac{a \cdot \cos(E) - c}{a - c \cdot \cos(E)} \tag{1}$$

$$\sin(\varphi) = \frac{b \cdot \sin(E)}{r} = \frac{\sqrt{a^2 - c^2} \cdot \sin(E)}{a - c \cdot \cos(E)} \tag{2}$$

La formule de transformation entre les angles E (origine) et φ (foyer F_2) :

$$\tan\left(\frac{\varphi}{2}\right) = \sqrt{\frac{a+c}{a-c}} \cdot \tan\left(\frac{E}{2}\right)$$

En prenant la formule (2) et en dérivant par rapport à t :

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(\sin(\varphi)) &= \cos(\varphi) \cdot \dot{\varphi} \\
&= \frac{b\cos(E) \cdot \dot{E}(a - c\cos(E)) - c\sin(E) \cdot \dot{E} b \sin(E)}{(a - c\cos(E))^2} \\
&= \dots \\
&= b\dot{E} \cdot \frac{a\cos(E) - c}{(a - c\cos(E))^2}
\end{aligned}$$

On divise par $\cos(\varphi)$, alors

$$\dot{\varphi} = \frac{\mathrm{d}\varphi}{\mathrm{d}t} = \frac{b\dot{E}}{a - c\cos(E)} = \frac{b\dot{E}}{r} \tag{3}$$

La deuxième loi de Kepler (*Loi des aires*) dit :

Le rayon-vecteur reliant une planète au Soleil balaie des aires égales en des temps égaux.

L'aire doit être constante :

$$r^2 \cdot \dot{\varphi} = \text{cste.} = C$$

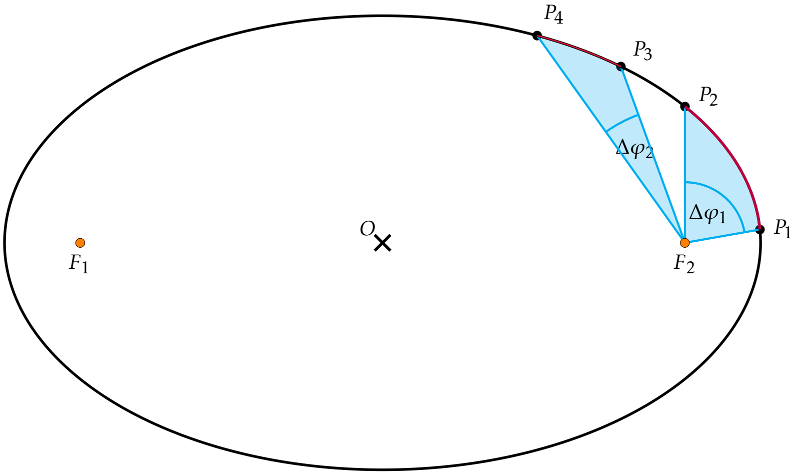
L'équation (3) résolue pour $\dot{E}$:

$$\dot{E} = \frac{\mathrm{d}E}{\mathrm{d}t} = \frac{r^2 \dot{\varphi}}{br} = \frac{C}{br} = \frac{C}{b(a - c\cos(E))}$$

Par integration

$$\int \mathrm{d}t = \int \frac{b}{C}(a - c\cos(E))\,\mathrm{d}E$$

$$t = t(E) : t = \frac{b}{C}(aE - c\sin(E))$$



3 Commandes

On définit les paramètres de l'ellipse :

```
\xintdefvar Ox, Oy := 7, 2;                    % center of the ellipse
\xintdefvar a_el, b_el := 5, 3;                % demi axes
\xintdefvar phi_1, phi_2, phi_3 := 10, 90, 110;  % polar angles

\xintdefvar e_el := sqrt(a_el^2-b_el^2);       % distance focus center
\xintdefvar F_1x, F_1y, F_2x, F_2y := Ox-e_el, Oy, Ox+e_el, Oy; % coordinates of the foci
\xintdefvar eps_el := e_el/a_el;               % eccentricity
```

On définit la fonction de l'ellipse :

```
\xintdeffloatefunc trans_phi_M(x) := ifsgn([Ellipse(x)][0]-Ox,
                                        atand(([Ellipse(x)][1]-Oy)/([Ellipse(x)][0]-Ox))+180,
                                        0,
                                        atand(([Ellipse(x)][1]-Oy)/([Ellipse(x)][0]-Ox))
                                        );
\xintdeffloatefunc trans_M_phi(x) := ifsgn([Ellipse_M(x)][0]-F_2x,
                                        atand(([Ellipse_M(x)][1]-F_2y)/([Ellipse_M(x)][0]-F_2x))+180,
                                        0,
                                        atand(([Ellipse_M(x)][1]-F_2y)/([Ellipse_M(x)][0]-F_2x))
                                        );
\xintdeffloatefunc phi_Pol_to_E(x) :=if(x=180,
                                     180,
                                     if(x>180,
                                        2*atand(sqrt((a_el-e_el)/(a_el+e_el))*tand(x/2))+360,
                                        2*atand(sqrt((a_el-e_el)/(a_el+e_el))*tand(x/2))
                                        )
                                     );
\xintdeffloatefunc E_to_phi_Pol(x) := if(x=180,
                                      180,
                                      if(x>180,
                                         2*atand(sqrt((a_el+e_el)/(a_el-e_el))*tand(x/2))+360,
                                         2*atand(sqrt((a_el+e_el)/(a_el-e_el))*tand(x/2))
                                         )
                                      );
```

Les formules pour la transformation des angles E en φ et vice versa :

```
\xintdeffunc trans_phi_M(x) := ifsgn([Ellipse(x)][0]-Ox,
                                  atand(([Ellipse(x)][1]-Oy)/([Ellipse(x)][0]-Ox))+180,
                                  0,
                                  atand(([Ellipse(x)][1]-Oy)/([Ellipse(x)][0]-Ox))
                                  );
\xintdeffunc trans_M_phi(x) := ifsgn([Ellipse_M(x)][0]-F_2x,
                                  atand(([Ellipse_M(x)][1]-F_2y)/([Ellipse_M(x)][0]-F_2x))+180,
                                  0,
                                  atand(([Ellipse_M(x)][1]-F_2y)/([Ellipse_M(x)][0]-F_2x))
                                  );
% phi --> E
\xintdeffunc phi_Pol_to_E(x) := if(x=180,
                                180,
                                if(x>180,
                                   2*atand(sqrt((a_el-e_el)/(a_el+e_el))*tand(x/2))+360,
                                   2*atand(sqrt((a_el-e_el)/(a_el+e_el))*tand(x/2))
                                   )
                                );
% E --> phi
\xintdeffunc E_to_phi_Pol(x) := if(x=180,
                                180,
                                if(x>180,
                                   2*atand(sqrt((a_el+e_el)/(a_el-e_el))*tand(x/2))+360,
                                   2*atand(sqrt((a_el+e_el)/(a_el-e_el))*tand(x/2))
                                   )
                                );
```

Integration et la méthode de Newton pour calculer φ_4 :

```
%% INTEGRATION -- ATTENTION -- calculate in RAD not DEGREES
% ---- ATTENTION --- phi_1, phi_2, phi_3 ARE NOW HARD CODED IN THE FUNCTIONS HERE ----
% NEWTON: gets root by x <-- x - delta, delta = f(x)/f'(x)
% so let's simplify f in order for computations to have been done already
% *oneDegree converts from degrees to radians
\xintdeffloatvar phiE_1, phiE_2, phiE_3 := seq(phi_Pol_to_E(x), x = phi_1, phi_2, phi_3);
\xintdeffloatefunc f(x, a, e):= a*x-e*sin(x)+e*(sind(phiE_2)+sind(phiE_3)-sind(phiE_1))-a*(phiE_2+phiE_3-phiE_1)*oneDegree;
\xintdeffloatefunc fD(x, a, e):= a-e*cos(x); % derivative of f(x)
```

```
% multiplication by oneRadian converts from Radians to Degrees
\xintdeffloatvar phiE := iter(oneDegree * phiE_3 {;}% WE NEED TO HIDE THIS ; FROM deffloatvar <<< TO BE IMPROVED IN XINT
                                        % initial value (in RADIANS) is E
                                        % conversion of phi_3
                                        subs((abs(D) < 1e-6)?
                                        % this test of absolute error is not very good in floating point context
                                             {break(@-D)}% stop now
                                             {@ - D} % @ is previous value, replace it with @ -D
                                             , D = f(@, a_el, e_el)/fD(@, a_el, e_el)
                                            )% end of substitution of D value
                        , i = 1++)% iterate as many times as needed
                       * oneRadian;% end of definition of phiE (final value already converted to DEGREES)

% Now get the calculated angle phi_4 back from phiE
% We have defined E_to_phi_Pol as using floating point, hence the \xintfloatexpr wrapper
\xintdefvar phi_4 := \xintfloatexpr E_to_phi_Pol(phiE)\relax;
```

Faire le dessin :

```
\begin{pspicture}[showgrid=false,saveNodeCoors,NodeCoorPrefix=n](0,-1)(14,5)
\psset{dotscale=0.8}
\psset{PointSymbol=*,plotpoints=600}
\footnotesize
% Set nodes to center and foci
\pstGeonode[PosAngle=135,PointName={O},PointSymbol=+,dotscale=2,dotangle=45](\XcalcR{Ox},\XcalcR{Oy}){O}
\pnodes(\XcalcR{F_1x},\XcalcR{F_1y}){F_1}(\XcalcR{F_2x},\XcalcR{F_2y}){F_2}
\pstEllipseFocusNode[PosAngle=-90](O)(\XcalcR{a_el},\XcalcR{b_el}){F_1}{F_2}
% Set dots on the ellipse at the points with the polar angles phi_1, phi_2, ...
\xintForpair #1#2 in { (P_1,phi_1),(P_2,phi_2),(P_3,phi_3),(P_4,phi_4)}\do
      {\pnode(\XcalcR{[Ellipse(#2)][0]},\XcalcR{[Ellipse(#2)][1]}){#1}%
       \psdot[dotstyle=*,dotscale=1](#1)
       \uput[\XcalcR{trans_phi_M(#2)}](#1){$#1$}
      }
% Fill area between phi_1 and phi_2
\pscustom[fillstyle=solid,fillcolor=cyan,opacity=0.25,linestyle=none,polarplot,algebraic]{%
   \translate(\XcalcR{F_2x},\XcalcR{F_2y})
   \psplot{\XcalcR{phi_1*Pi/180}}{\XcalcR{phi_2*Pi/180}}{\XcalcR{b_el^2/a_el}/(1+\XcalcR{sqrt(a_el^2-b_el^2)/a_el}*cos(x))}
   \lineto(F_2)
   \closepath
}
% Set the angle label
\pstMarkAngle[linecolor=cyan,MarkAngleRadius=0.8,LabelSep=0.5]{P_1}{F_2}{P_2}{$\Delta\varphi_1$}
\psline[linecolor=cyan](F_2)(P_1)
\psline[linecolor=cyan](F_2)(P_2)
% Fill area between phi_1 and phi_2
\pscustom[fillstyle=solid,fillcolor=cyan,opacity=0.25,linestyle=none,polarplot,algebraic]{%
   \translate(\XcalcR{F_2x},\XcalcR{F_2y})
   \psplot{\XcalcR{phi_3*Pi/180}}{\XcalcR{phi_4*Pi/180}}{\XcalcR{b_el^2/a_el}/(1+\XcalcR{sqrt(a_el^2-b_el^2)/a_el}*cos(x))}
   \lineto(F_2)
   \closepath
}
% Set the angle label
\pstMarkAngle[linecolor=cyan,MarkAngleRadius=1.8,LabelSep=1.4]{P_3}{F_2}{P_4}{$\Delta\varphi_2$}
\psline[linecolor=cyan](F_2)(P_3)
\psline[linecolor=cyan](F_2)(P_4)
% orange dots at the foci
\psdots[dotstyle=o,dotscale=1,fillcolor=orange](F_1)(F_2)
% Draw ellipse and color parts of the ellipse
\pstEllipse[linecolor=black,linewidth=1pt](O)(\XcalcR{a_el},\XcalcR{b_el})
\rput(F_2){%
          \psplot[polarplot,plotstyle=curve,linewidth=0.5pt,linecolor=Crimson]%
          {\XcalcR{phi_3}}{\XcalcR{phi_4}}{\XcalcR{b_el^2/a_el} 1 \XcalcR{sqrt(a_el^2-b_el^2)/a_el} x cos mul add div}
          }
\pstGeneralEllipse[linecolor=purple,linewidth=1pt](O)(\XcalcR{a_el},\XcalcR{b_el})%
                 [0][\XcalcR{phi_Pol_to_E(phi_1)}][\XcalcR{phi_Pol_to_E(phi_2)}]
\end{pspicture}
```