Fakulta elektrotechnická
České vysoké učení technické

Hillovy ekvipotenciály v systémech dvojhvězd

Jan Havlík
2.6.1999

Obsah:

1. Úvod

Tato práce vznikla jako semestrální projekt na Fakultě elektrotechnické Českého vysokého učení technického v Praze.

Jejím smyslem je ukázat možnosti numerické simulace gravitačních ekvipotenciál na systémech dvojhvězd a takovou simulaci naprogramovat v programovacím jazyce Pascal.

Práce má sloužit především k simulacím soustav dvojhvězd pro studijní účely a jako podklad pro zpracování problému soustav dvojhvězd ve vyšších programovacích jazycích, především v jazyku Java s možností následného spuštění v prostředí Internetu.

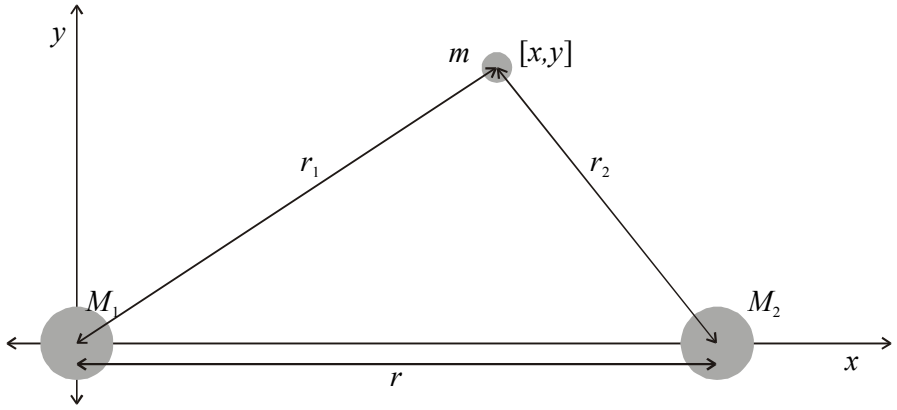
2. Teoretické zpracování

2.1. Soustavy dvojhvězd

Dvojhvězda je soustava dvou hvězd vzájemně gravitačně svázaných a rotujících kolem společného těžiště. Energie tělesa W_p v takovéto soustavě je dána součtem gravitačních energií od jednotlivých hvězd a rotační energie.

$$W_p = -G\frac{mM_1}{r_1} - G\frac{mM_2}{r_2} + \frac{1}{2}mr^2\omega^2 \qquad (1)$$

Význam použitých symbolů je zřejmý z následujícího obrázku. Všechny vztahy jsou odvozovány v soustavě rotující společně s oběma hvězdami kolem společného těžiště soustavy.



2.2. Odvození funkce potenciálu

Gravitační potenciál Φ odvodíme z podílu energie W_p (viz. (1)) a hmotnosti m.

$$\Phi = \frac{W_p}{m} = -G\frac{M_1}{r_1} - G\frac{M_2}{r_2} + \frac{r^2\omega^2}{2} \qquad (2)$$

Z třetího Keplerova zákona vyjádříme úhlovou rychlost ω.

$$\frac{r^3}{T^2} = \frac{G}{4\pi^2}(M_1 + M_2)$$
$$T^2 = \frac{4\pi^2 r^3}{G(M_1 + M_2)} \tag{3}$$
$$\omega^2 = \frac{4\pi^2}{T^2} = \frac{G(M_1 + M_2)}{r^3}$$

Z Pythagorovy věty odvodíme vztahy pro vzdálenosti r_1 a r_2.

$$r_1 = \sqrt{x^2 + y^2}$$
$$r_2 = \sqrt{(x - r)^2 + y^2} \tag{4}$$

Oboje dosadíme zpět do vztahu pro potenciál Φ.

$$\Phi(x, y) = -\frac{GM_1}{\sqrt{x^2 + y^2}} - \frac{GM_2}{\sqrt{(x - r)^2 + y^2}} + \frac{G(M_1 + M_2)}{2r} \tag{5}$$

2.3. Hillovy ekvipotenciály

Rocheovy ekvipotenciály, jež hledáme, jsou popsány rovnicí

$$\Phi(x, y) = konst. \tag{6}$$

Její řešení jsem ve své práci hledal dvěma způsoby, Newtonovou metodou a metodou Runge – Kutta, jejichž popis následuje.

2.3.1. Newtonova metoda

Z funkčního předpisu funkce Φ (viz. (5)) vyjádříme normálový vektor $\vec{n}$

$$\vec{n} = \left(\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y}\right) \tag{7}$$

a dále tečný vektor $\vec{\tau}$.

$$\vec{\tau} = \left(-\frac{\partial \Phi}{\partial y}, \frac{\partial \Phi}{\partial x}\right) \tag{8}$$

Z tečného vektoru pak můžeme vyjádřit vztahy pro přírůstek ekvipotenciály v obou souřadnicích x i y.

$$\frac{dx}{dt} = -\frac{\partial \Phi}{\partial y}$$
$$\frac{dy}{dt} = \frac{\partial \Phi}{\partial x} \tag{9}$$

Při znalosti souřadnic bodu $[x_n,y_n]$ pak můžeme souřadnice následujícího bodu $[x_{n+1},y_{n+1}]$ vypočítat ze vztahů.

$$x_{n+1} = x_n + \left.\frac{dx}{dt}\right|_n \cdot \Delta t$$
$$y_{n+1} = y_n + \left.\frac{dy}{dt}\right|_n \cdot \Delta t \tag{10}$$

Tedy platí, že

$$x_{n+1} = x_n - \left.\frac{\partial \Phi}{\partial y}\right|_n \cdot \Delta t$$
$$y_{n+1} = y_n + \left.\frac{\partial \Phi}{\partial x}\right|_n \cdot \Delta t \tag{11}$$

2.3.2. Metoda Runge – Kutta

Označíme

$$f(x,y) = -\frac{\partial \Phi}{\partial y}$$
$$g(x,y) = \frac{\partial \Phi}{\partial x} \tag{12}$$

V každém bodě $[x_n,y_n]$ potom určíme sadu konstant K a L:

$$\begin{aligned}
K_1 &= f(x_n, y_n) \\
L_1 &= g(x_n, y_n) \\
K_2 &= f(x_n + \frac{K_1}{2}\Delta t, y_n + \frac{L_1}{2}\Delta t) \\
L_2 &= g(x_n + \frac{K_1}{2}\Delta t, y_n + \frac{L_1}{2}\Delta t) \\
K_3 &= f(x_n + \frac{K_2}{2}\Delta t, y_n + \frac{L_2}{2}\Delta t) \\
L_3 &= g(x_n + \frac{K_2}{2}\Delta t, y_n + \frac{L_2}{2}\Delta t) \\
K_4 &= f(x_n + K_3\Delta t, y_n + L_3\Delta t) \\
L_4 &= g(x_n + K_3\Delta t, y_n + L_3\Delta t)
\end{aligned} \tag{13}$$

Z nich pak vypočteme souřadnice bodu $[x_{n+1},y_{n+1}]$ pomocí vztahů

$$\begin{aligned}
x_{n+1} &= x_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)\Delta t \\
y_{n+1} &= y_n + \frac{1}{6}(L_1 + 2L_2 + 2L_3 + L_4)\Delta t
\end{aligned} \tag{14}$$

2.3.3. Časování

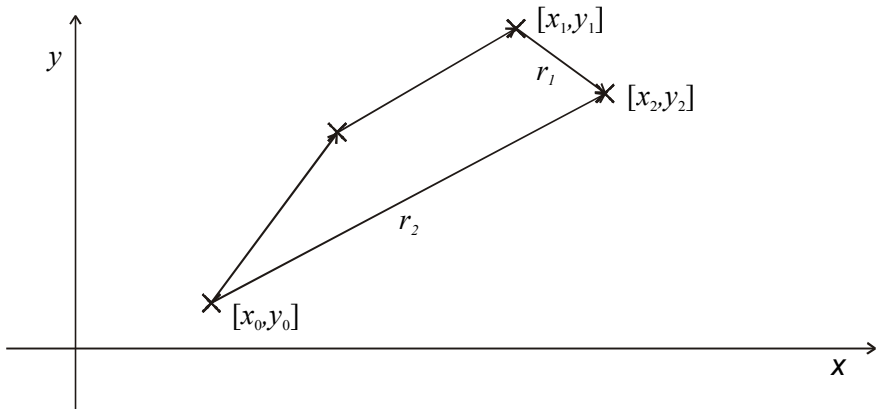
V obou výše uvedených metodách vystupuje jako jediný volitelný parametr časový krok Δt. Právě jeho volbou tedy můžeme ovlivnit charakter získaných výsledků i vlastnosti celého algoritmu. Příliš malá hodnota Δt mám poskytne velmi přesné výsledky, ale celý výpočet může zpomalit pod únosnou mez, naopak velká hodnota kroku Δt nám zajistí svižnost celého výpočtu, avšak za cenu neúnosně velké numerické chyby.

Z výše uvedeného je zřejmé, že správná volba hodnoty Δt může zcela zásadním způsobem ovlivnit rychlost i přesnost celého algoritmu. Protože křivost Hiloĺových křivek se bod od bodu mění, ukázalo se, že pokud by měla být použita taková fixní hodnota parametru Δt, aby výpočet běžel s dostatečnou přesností, bude tak malá, že výpočet poběží velmi pomalu. Proto jsem se

rozhodl použít metody dynamického kroku, která je sice výpočetně náročnější, ale v konečném hodnocení je při zachování požadované přesnosti rychlejší.

2.3.4. Metoda dynamického kroku

Princip metody spočívá v tom, že algoritmus sám rozhodne, jak velký je potřeba zvolit krok Δt pro zachování dané přesnosti. Pokud je počítaná křivka velmi křivá, algoritmus sám zjemní krok tak, aby splnil požadovanou relativní přesnost a naopak pokud se výpočet pohybuje v místech s malou křivostí, je krok volen větší a výpočet tedy běží rychleji.



Z daného výchozího bodu o souřadnicích $[x_0,y_0]$ vypočteme následujícím způsobem dva body $[x_1,y_1]$ a $[x_2,y_2]$. Bod $[x_2,y_2]$ jako nejbližší další bod se zvoleným krokem Δt a bod $[x_1,y_1]$ jako druhý nejbližší bod s krokem $\frac{\Delta t}{2}$. To znamená, že bychom měli získat dva téměř shodné body, protože zatímco v prvním případě jsme se posunuli o jeden krok s délkou Δt, v druhém případě jsme se posunuli o dva kroky poloviční délky.

Tedy platí-li

$$\begin{aligned} x_{n+1} &= f(x_n, y_n, \Delta t) \\ y_{n+1} &= g(x_n, y_n, \Delta t) \end{aligned} \tag{15}$$

pak platí

$$
\begin{aligned}
x_2 &= f(x_0, y_0, \Delta t) \\
y_2 &= g(x_0, y_0, \Delta t) \\
x_1 &= f(f(x_0, y_0, \frac{\Delta t}{2}), g(x_0, y_0, \frac{\Delta t}{2}), \frac{\Delta t}{2}) \\
y_1 &= g(f(x_0, y_0, \frac{\Delta t}{2}), g(x_0, y_0, \frac{\Delta t}{2}), \frac{\Delta t}{2})
\end{aligned}
\tag{16}
$$

Mezi takto vypočítanými body $[x_1,y_1]$ a $[x_2,y_2]$ a bodem $[x_0,y_0]$ definujeme vzdálenosti r_1 a r_2, z nichž potom vypočteme relativní chybu δ.

$$
\begin{aligned}
r_1 &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
r_2 &= \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2} \\
\delta &= \frac{r_1}{r_2}
\end{aligned}
\tag{17}
$$

Takto vypočtenou relativní chybu pak porovnáme se zvoleným tolerančním schématem $[\delta_{min}, \delta_{max}]$.

Pokud platí $\delta_{min} < \delta < \delta_{max}$, pokračujeme výpočtem z bodu $[x_2,y_2]$ při zachování velikosti kroku Δt.

Pokud platí $\delta \leq \delta_{min}$, zvětšíme velikost kroku Δt na $2 \cdot \Delta t$ a výpočet opakujeme v bodě $[x_0,y_0]$.

Pokud platí $\delta_{max} \leq \delta$, zmenšíme velikost kroku Δt na $\frac{\Delta t}{2}$ a výpočet opakujeme v bodě $[x_0,y_0]$.

Tímto postupem je zajištěno, že výpočet poběží maximální možnou rychlostí při zachování daného tolerančního schématu.

3. Implementace algoritmů a programové zpracování v jazyce PASCAL

3.1. Volba metody a porovnání jednotlivých algoritmů

Při psaní programu jsem byl postaven před výběr metody pro numerické jádro programu. Počáteční volba byla velice jednoduchá. Z možných metod je nejjednodušší a nejsnáze použitelná Newtonova metoda (viz. 2.3.1), neboť její princip je naprosto zřejmý a její správný běh včetně mezivýsledků je tedy velice snadno kontrolovatelný.

Po naprogramování se však ukázalo, že pokud má metoda poskytovat dostatečně přesné výsledky, je potřeba použít velmi malý krok Δt řádu 10^{-5} až 10^{-6}. Takto malý krok ale velmi zpomaluje běh programu a v praxi je jen obtížně použitelný.

Proto jsem upustil od realizace Newtonovou metodou a zvolil metodu Runge – Kutha (viz. kap. 2.3.2), která sice není tak jednoduchá a přehledná, ale poskytuje mnohem přesnější výsledky i pro řádově větší krok Δt. Protože křivost jednotlivých vykreslovaných křivek se výrazně mění nejen pro jednotlivé ekvipotenciály, ale i pro různé body jedné křivky, ukázalo se jako nutnost doplnit tuto metodu ještě dynamickým časováním (viz. kap. 2.3.4). Takto naprogramované jádro se ukázalo jako velice rychlé i při zachování požadované přesnosti. Jediným volitelným parametrem takovéhoto algoritmu pak zůstává toleranční schéma pro povolenou relativní chybu, které jsem nakonec volil zkusmo a jehož hodnoty jsem ponechal na $\delta_{min} = 10^{-4}$ a $\delta_{max} = 10^{-2}$.

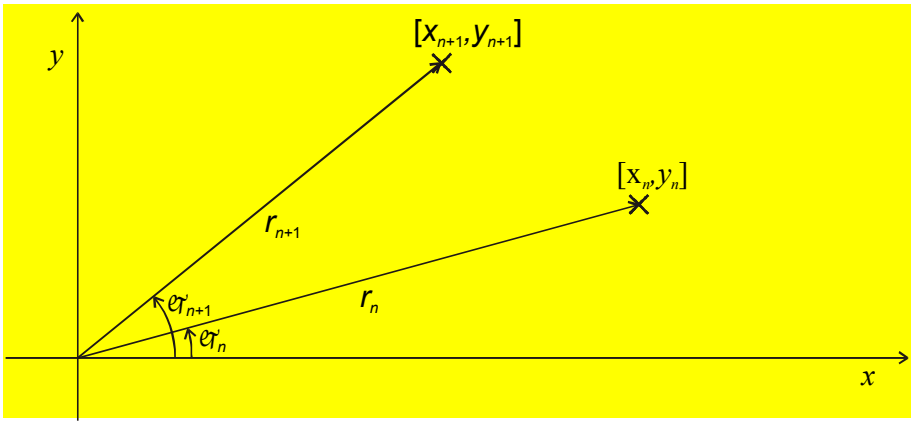
3.2. Konstrukce ekvipotenciály

Každou ekvipotenciálu můžeme konstruovat z libovolného bodu v prostoru, který označíme souřadnicemi $[x_0,y_0]$. Pomocí výše uvedeného postupu napočítáme další bod křivky a posuneme se do něho. Tento postup

opakujeme tak dlouho, dokud nevykreslíme celou křivku. Protože však není možné uvažovat konec vykreslování jako návrat zpět do bodu $[x_0,y_0]$, je třeba konec vykreslování detekovat nějakým jiným vhodným způsobem.

3.2.1. Problém detekce „konce“

Jako nejvhodnější postup pro rozpoznání konce vykreslování se ukázala postupná detekce vykresleného úhlu.



Kartézské souřadnice každého napočítaného bodu $[x_{n+1},y_{n+1}]$ jsou převedeny do polárních souřadnic $[r_{n+1},\varphi_{n+1}]$. Z takto získaných souřadnic a znalosti souřadnic minulého bodu se vypočte změna v úhlu

$$\Delta\varphi_{n+1} = \varphi_{n+1} - \varphi_n \tag{18}$$

Takto získané hodnoty $\Delta\varphi_{n+1}$ se postupně sčítají a vykreslování se zastaví ve chvíli, kdy

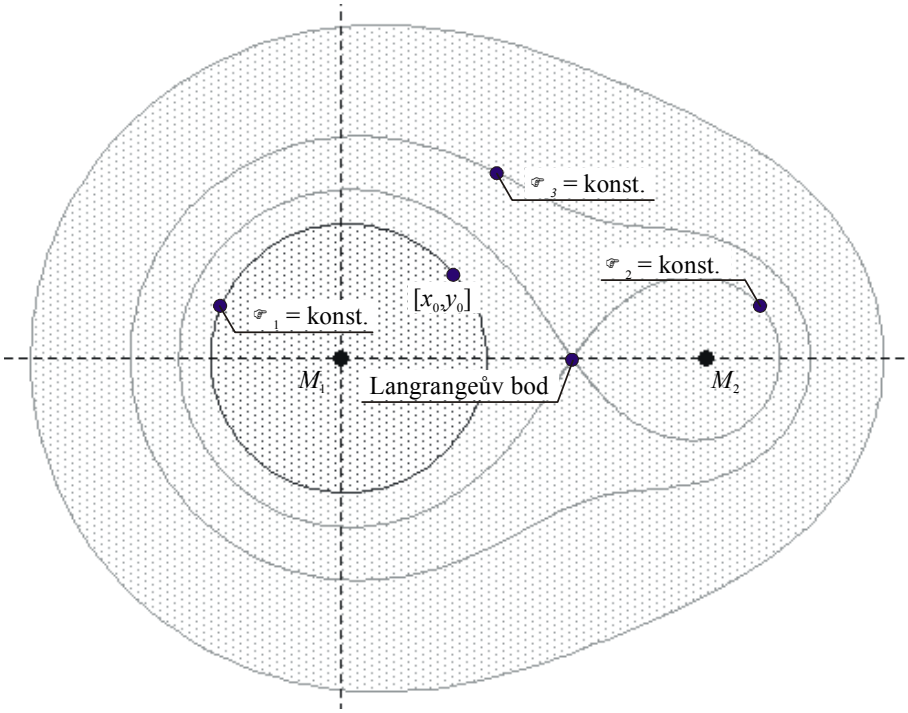
$$\sum_n \Delta\varphi_{n+1} \geq 2\pi \tag{19}$$

Z uvedeného je zřejmé, že hodnoty souřadnice r_n nejsou pro výpočet nutné a v praxi proto můžeme jejich výpočet vynechat.

Správná funkce výše popsaného postupu však ke svojí činnosti vyžaduje vhodně definovaný úhel φ_n, resp. vhodně zvolený počátek souřadnic pro jeho výpočet. Pro ten musí platit, že vedeme-li z něho polopřímku libovolným směrem, kreslenou ekvipotenciálu protneme právě jednou. Rozborem tohoto požadavku snadno zjistíme, že při výpočtu úhlu musíme dodržet následující pravidla:

1) Pokud vykreslujeme některou ekvipotenciálu ležící od jednoho z bodů M_1, M_2 blíže než Rocheova ekvipotenciála (prochází vnitřním Lagrangeovým bodem soustavy), volíme počátek soustavy souřadnic do toho z bodů M_1, M_2, jemuž je kreslená ekvipotenciála blíže než Rocheova křivka.

2) Pokud vykreslujeme některou ekvipotenciálu ležící od obou bodů M_1, M_2 dále než Rocheova ekvipotenciála, volíme počátek souřednic do vnitřního Lagrangeova bodu soustavy M_1,M_2.



Problém detekce „konce“ se tak redukuje na úkol rozpoznání, kterou ekvipotenciálu budeme vykreslovat, resp. která ekvipotenciála prochází zvoleným počátečním bodem $[x_0,y_0]$.

Protože víme, že platí (viz. obr.)

$$\Phi_1 < \Phi_2 < \Phi_3 \tag{20}$$

stačí před započetím vykreslování vypočítat hodnotu potenciálu v počátečním bodě $[x_0,y_0]$ a porovnat ji s hodnotou gravitačního potenciálu Φ_L ve vnitřním Lagrangeově bodě soustavy $[x_L,0]$. Souřadnici x_L vypočteme snadno

z požadavku rovnosti potenciálu od obou těles soustavy M_1, M_2

$$-G\frac{M_1}{x_L^2} = -G\frac{M_2}{(r-x_L)^2} \tag{21}$$

Po úpravách dostaneme

$$\begin{aligned} x_L &= \frac{1}{2}r & M_1 &= M_2 \\ x_L &= \frac{M_1 - \sqrt{M_1 M_2}}{M_1 - M_2}r & M_1 &\neq M_2 \end{aligned} \tag{22}$$

3.3. Popis některých důležitých procedur a funkcí programu

3.3.1. Seznam použitých proměnných a konstant

Proměnné deklarované v části `Const` jsou takzvané proměnné s počáteční hodnotou, to znamená, že se jedná o klasickou proměnnou jejíž hodnota je před spuštěním programu nastavena na hodnotu uvedenou za znakem =.

```
const m1:real = 1;          {hmotnost 1.hvezdy}
      m2:real = 1;          {hmotnost 1.hvezdy}
      mi:real = 0.2;        {cetnost pruseciku s x}
var maxx,maxy:integer;      {promenne pro grafiku}
    xasp,yasp:word;         {promenne pro grafiku}
    i,j,k:integer;          {promenne pro cykly}
    a,b:integer;            {souradnice bodu na obrazovce}
    x,y,x0,y0,oldx,oldy:real;   {souradnice aktualniho
       bodu, souradnice prvniho bodu a souradnice minuleho
       bodu}
    Fi,Fi0,Fi_old:real;     {uhel k aktualnimu, prvnimu
       a minulemu bodu}
    psi0:real;              {potencial ve vnitrnim L. bode}
    r1:real;                {x souradnice vnitriho L. bodu}
    t:real;                 {krok}
    q1,q2:real;             {normovane hmotnosti}
    tl:integer;             {stav tlacitka mysi}
    text:string;
    ch:char;
```

3.3.2. Procedura *Vstupy*

Procedura *Vstupy* umožňuje zadat hmotnosti obou hvězd soustavy M_1, M_2 a dále parametr určující hustotu vykreslení základních ekvipotenciál.

Jednotlivé hodnoty se zadávají po kliknutí myší na příslušné okénko z klávesnice, vstup je potřeba potvrdit stiskem klávesy Enter. Další běh programu, tedy výpočet ekvipotenciál, se pak spustí kliknutím na ikonu OK.

Hodnoty hmotností M_1 a M_2 jsou pak ještě před spuštěním výpočtů normovány (viz. kapitola 3.3.3. Procedura *Normovani*).

Program neumožňuje zadat jako parametr pro výpočet vzdálenost hvězd *r*, ale implicitně uvažuje její hodnotu $r = 1$, neboť ta nijak kvalitativně neovlivňuje tvar gravitačních ekvipotenciál, určuje pouze měřítko zobrazení a pro simulaci tedy není důležitá.

3.3.3. Procedura *Normovani*

Procedura *Normovani* slouží k znormování zadaných hmotností M_1, M_2 před spuštěním výpočtu gravitačních ekvipotenciál. Hmotnosti jsou normovány tak, aby byl zachován jejich vzájemný poměr a hodnota větší z nich byla rovna jedné. Normované hodnoty hmotností jsou uloženy do proměnných q_1 a q_2.

Vztahy pro normování lze snadno odvodit z výše uvedených pravidel:

$$\begin{aligned} q_1 = 1 \quad q_2 = \frac{M_2}{M_1} \qquad & M_1 \geq M_2 \\ q_1 = \frac{M_1}{M_2} \quad q_2 = 1 \qquad & M_1 < M_2 \end{aligned} \tag{23}$$

```
     procedure normovani;
     {normuje hmotnosti; vetsi hmotnost znormuje k 1, zachova
pomer}
     begin;
           if m1>=m2 then
            begin
             q2:=m2/m1;
             q1:=1;
            end
            else
            begin
             q1:=m1/m2;
```

```
        q2:=1;
       end;
end;
```

3.3.4. Funkce *Potencial*

Funkce *Potencial* vrací hodnotu gravitačního potenciálu v bodě o souřadnicích [*x,y*].

```
function potencial(x,y:real):real;
{vraci hodnotu potencialu v parametru}
var denom1,denom2:real;
begin;
  denom1:=sqrt(sqr(x)+sqr(y));
  denom2:=sqrt(sqr(x-1)+sqr(y));
  potencial:=q1/denom1+q2/denom2;
end;
```

3.3.5. Funkce *Uhel*

Funkce vrací úhel jako polární souřadnici bodu zadaného kartézskými souřadnicemi [*x,y*]. Počátek polárních souřadnic je nastaven do bodu podle diskuse v kap. 3.2.1. Problém detekce „konce“.

```
function uhel(x,y:real):real;
{vraci uhel osa x - bod[x,y]}
 begin;
 if potencial(x,y)<psi0 then x:=x-r1
                        else
                             if x>r1 then x:=x-1;
 if x>0 then
    if y>=0 then uhel:= arctan(y/x)
            else uhel:= 2*Pi + arctan(y/x);
 if x<0 then
    if y>=0  then uhel:= Pi + arctan(y/x)
             else uhel:= Pi + arctan(y/x);
 if x=0 then
    if y>0 then uhel:=Pi/2
           else uhel:=3*Pi/2;
end;
```

3.3.6. Funkce *DeltaFi*

Funkce vrací rozdíl hodnoty úhlové souřadnice bodů $[x_n,y_n]$ a $[x_{n+1},y_{n+1}]$. Podmínce If .. Then je ošetřena správná funkce pro případ, že bod $[x_{n+1},y_{n+1}]$ leží nad osou *x*, ale předchozí bod leží ještě pod osou *x*.

```
function DeltaFi(Fi,Fi_old:real):real;
{napocita rozdil uhlu v parametru}
begin;
  if (Fi<(Pi/2)) and (Fi_old>(3*Pi/2))
   then
     DeltaFi:=((2*Pi)-Fi_old)+Fi
   else
     DeltaFi:=Fi-Fi_old;
end;
```

3.3.7. Funkce *Lbod*

Vypočte hodnotu souřadnice x_L vnitřního Lagrangeova bodu soustavy M_1, M_2. Odvození viz. kap. 3.2.1. Problém detekce „konce“.

```
function LBod(q1,q2:real):real;
{vraci x souradnici vnitrniho Lagrangeova bodu soustavy}
begin;
 if q1=q2 then LBod:=1/2
          else LBod:=(q1-sqrt(q1*q2))/(q1-q2);
end;
```

3.3.8. Procedura *Next_XY*

Procedura slouží k výpočtu souřadnic dalšího bodu vykreslované ekvipotenciály ze zadaných souřadnic $[x_n,y_n]$. K výpočtu souřadnic dalšího bodu využívá metody Runge-Kutha 4. řádu (implementována ve funkcích *Next_X* a *Next_Y*) s dynamickým časem (implementován zde). Toleranční schéma pro relativní chybu je nastaveno na rozsah $10^{-2} > \delta > 10^{-4}$. Popis metod viz kap. 2.3.2 a 2.3.4.

```
procedure next_xy;
var x1,y1,x2,y2,r,r0,chyba:real;
begin;
 repeat
  x2:=next_x(x,y,t);
  y2:=next_y(x,y,t);
  x1:=next_x(x,y,t/2);
  y1:=next_y(x,y,t/2);
  x1:=next_x(x1,y1,t/2);
  y1:=next_y(x1,y1,t/2);
  r0:=sqrt(sqr(x2-x)+sqr(y2-y));
  r:=sqrt(sqr(x2-x1)+sqr(y2-y1));
  chyba:=r/r0;
  if chyba>1e-2 then t:=t/2;
  if chyba<1e-4 then t:=2*t;
 until (chyba>=1e-4) and (chyba<=1e-2);
```

```
 x:=x2;
 y:=y2;
end;
```

3.3.9. Procedura *One_Line*

Procedura vykreslí celou gravitační ekvipotenciálu procházející zadaným bodem [*x,y*]. Vykreslování je třeba provádět příkazem `LineTo`, nikoli `PutPixel`. Jen tak je možno zajistit, že vznikne uzavřená křivka i při větších vzdálenostech jednotlivých bodů. Stejně tak je třeba zajistit spojení prvního a posledního bodu křivky. Z výše uvedeného vyplývá, že při vykreslování ekvipotenciály je třeba, aby program dodržel následující postup:

1) Uloží souřadnice výchozího bodu [*x,y*] do proměnných *x*0 a *y*0.
2) Nastaví grafický kurzor na souřadnice [*x*0, *y*0]
3) Vypočte a postupně pospojuje všechny další bodu ekvipotenciály.
4) Spojí poslední bod ekvipotenciály s výchozím bodem [*x*0, *y*0].

```
procedure One_Line;
{vykresli ekvipotencialu prochazejici bodem [x,y]}
begin;
   t:=0.5;
   x0:=x;  y0:=y;
   Fi0:=0;
   Fi:=uhel(x,y);
   converse;
   moveto(a,b);
   repeat
     converse;
     lineto(a,b);
     Fi_old:=Fi;
     next_xy;
     Fi:=uhel(x,y);
     Fi0:=Fi0+DeltaFi(Fi,Fi_old);
   until Fi0>2*Pi;
   x:=x0;
   y:=y0;
   converse;
   lineto(a,b);
end;
```

3.3.10. Procedura *Basic_Lines*

Procedura slouží k vykreslení základních ekvipotenciálních křivek v soustavě M_1, M_2. Jejich hustotu lze volit změnou zadáveného parametru. Po vykreslení předdefinovaných ekvipotenciál program poskytuje uživateli možnost vybrat kliknutím myší libovolný bod, v němž vykreslí další ekvipotenciálu.

4. Závěr

Na předcházejících stránkách jsem se pokusil ukázat možnosti počítačové simulace gravitačních ekvipotenciál v systémech dvojhvězd, shrnout jednotlivé možnosti postupu a diskutovat jejich výhody a nevýhody.

V průběhu práce se vyskytlo několik překážek, které se podařilo poměrně dobře odstranit a na které je v práci upozorňováno, takže by při dalším zpracování tématu a případném aplikování použitých postupů ve vyšších programovacích jazycích již neměly nastat žádné závažnější potíže.

Program Hill.exe, který vznikl v rámci této práce, umožňuje po zadání patřičných parametrů vykreslení několika základních gravitačních ekvipotenciál tak, aby poskytly co možná nejnázornější náhled na rozložení intenzity gravitačního pole v dané soustavě a dále umožňuje po volbě kliknutím myší vést ekvipotenciálu libovolným zvoleným bodem dané soustavy.