Hochschule Merseburg

Fachbereich

Ingenieur- und Naturwissenschaften

Bachelorarbeit

Thema: Berechnung einer Bahnbewegung bei Drehimpulserhaltung

Name: Sheng Caiwei

Studiengang: Mechatronik

Matrikel: 21008

Betreuer: Prof. Dr. Hartmut Kröner

Dr. Dariush Ehsani

Inhaltsverzeichnis

1.Einleitung

1.1 Numerische Lösungen einer Differenzialgleichung

Numerische Lösungen einer Differenzialgleichung benötigt man für die Analyse von Algorithmen für viele mathematische, physikalische, biologische, chemische oder wirtschaftswissenschaftliche Fragestellungen. In der Physik wird beispielsweise die Bahnbewegung von Planeten durch Differenzialgleichungen beschrieben.

Der Zweck von numerischen Lösungen einer Differenzialgleichung ist nicht das genaue Ergebnis, sondern eine Näherungslösung in einem "vernünftigen" Bereich von Fehlern.

Dabei benutzen wir für die näherungsweise Berechnung von Lösungen der Bahnbewegung das Software-Paket.

1.2 Implizite und explizite Differenzialgleichung

Ein Differenzialgleichung ist implizit, wenn sie in der Form

$F(t; x; x'; ...; x^{(n)}) = 0$ notiert ist.

Zum Beispiel: $x'(t) + x(t) \cdot x''(t) = 7 \cdot x(t)$

Ein Differenzialgleichung ist explizit, wenn sie nach ihrer höchsten Ableitung aufgelöst ist, also die Form $x^{(n)}(t) = F(t; x; x'; ...; x^{(n-1)})$ hat.

Zum Beispiel: $x'(t) = -2y(t) + x(t)$

Nicht alle impliziten Differenzialgleichungen können explizit dargestellt werden.

Aber jede explizite Differenzialgleichung lässt sich als implizite Differenzialgleichung schreiben.

1.3 Mathematica-Anweisung NDSolve zur numerischen Lösung von Differenzialgleichungen und Differenzialgleichungssystemen

1.3.1 Die Lösung einer Differenzialgleichung

1) Unterschied von Dsolve und NDSolve

Die „Syntax" von DSolve (noch keine numerische Berechnung!) benötigt eine Differenzialgleichung, die Bezeichnung der Funktion mit dem Argument, schlieβlich das Argument selbst.

Bsp.: DSolve[$y'[x] == y[x]$,$y[x]$,x].

Man kann die Anweisung NDSolve als Erweiterung der Anweisung DSolve betrachten, insofern die Syntax im Grunde übernommen wird. Hier ist die Anfangsbedingung unbedingt erforderlich, sonst gibt es „Gemecker" vom System.

Dazu sind a) die Anfangsbedingung samt Differenzialgleichung in eine Liste zu schreiben, b) anstelle des Arguments eine Liste zu verwenden, in welcher Argument, Anfangs- und Endwert des Arguments stehen.

Bsp.: NDSolve[$\{y'[x] == y[x], y[0] == 1\}$,$y[x]$,$\{x,0,10\}$]ShE.

2) Der Zugriff auf die numerische Lösung

Um auf die ermittelte Lösung zugreifen zu können, kann man so verfahren, dass man eine Funktion mit selbst gewähltem Namen über diese Lösung definiert (zuweist).

Bsp.: yexp[x_]=y[x]/.%[[1]]

1.3.2 Lösung von Differenzialgleichungssystemen

1) Die Anweisung DSolve bei einem System von Differenzialgleichungen.

Die Differenzialgleichungen werden jetzt in eine Liste eingetragen, ebenso die Funktionen.

Bsp.: Dsolve$[\{y'[t] == x[t], x'[t] == -y[t]\}, \{y[t], x[t]\}, t]$.

2) Die Anweisung NDSolve

Zusätzlich muss jetzt das Argument eine Spezifikation erfahren, d.h. anstelle des Arguments steht jetzt eine Liste, welche Argument, Anfangs- und Endwert des Arguments enthählt.

Bsp.: NDSolve$[\{y'[t] == x[t], x'[t] == -y[t], x[0] == 1, y[0] == 0\}, \{x[t], y[t]\}, \{t, 0, 6.3\}]$

Wie der Endwert zu wählen ist, hängt von der Anwendung zusammen. Es ist hierbei zu beachten, dass im allgemeinen die Ergebnisse um so ungenauer werden, je weiter die Werte vom Anfangswert entfernt liegen.

3) Der Zugriff auf die Lösung(en)

Der Zugriff auf die Lösungen erfolgt prinzipiell so wie der Zugriff auf die numerische Lösung.Man kann diesen Zugriff in einereinzigen Anweisung zusammenfassen, indem man wieder Listen verwendet.

Bsp.: $\{xc[t\_], ys[t\_]\} = \{x[t], y[t]\} /. \%[[1]]$

Die Funktion ys stellt eine numerisch ermittelte sin-Funktion im Intervall [0,6.3] dar, die Funktionen xc eine mumerisch ermittelte cos-Funktion. Die erhaltenen Funktion können zur Berechnung von Funktionswerten, zum Plotten usw. verwendt werden.[1]

1.4 Bahnbewegung bei Drehimpulserhaltung

1.4.1 Keplersches Gesetz

1. Keplersches Gesetz

   Die Planeten bewegen sich auf elliptischen Bahnen, in deren gemeinsamem Brennpunkt die Sonne steht.

2. Keplersches Gesetz

   Ein von der Sonne zum Planeten gezogener Fahrstrahl überstreicht in gleichen Zeiten gleich große Flächen.

3. Keplersches Gesetz

   Die Quadrate der Umlaufzeiten zweier Planeten verhalten sich wie die Kuben der großen Bahnhalbachsen.[2]

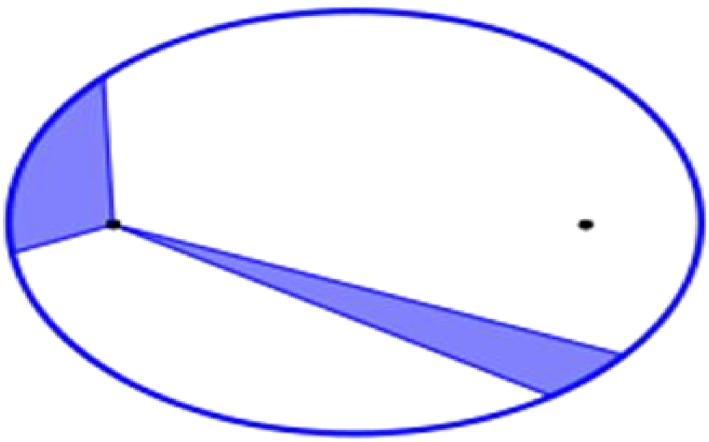


Abbildung 1: 2. Keplersches Gesetz

1.4.2 Erdbahn

Die Erdbahn wird in guter Näherung durch eine Ellipse mit der Sonne in einem der beiden Brennpunkte beschrieben, wie es vom ersten Keplerschen Gesetz verlangt wird.[3]

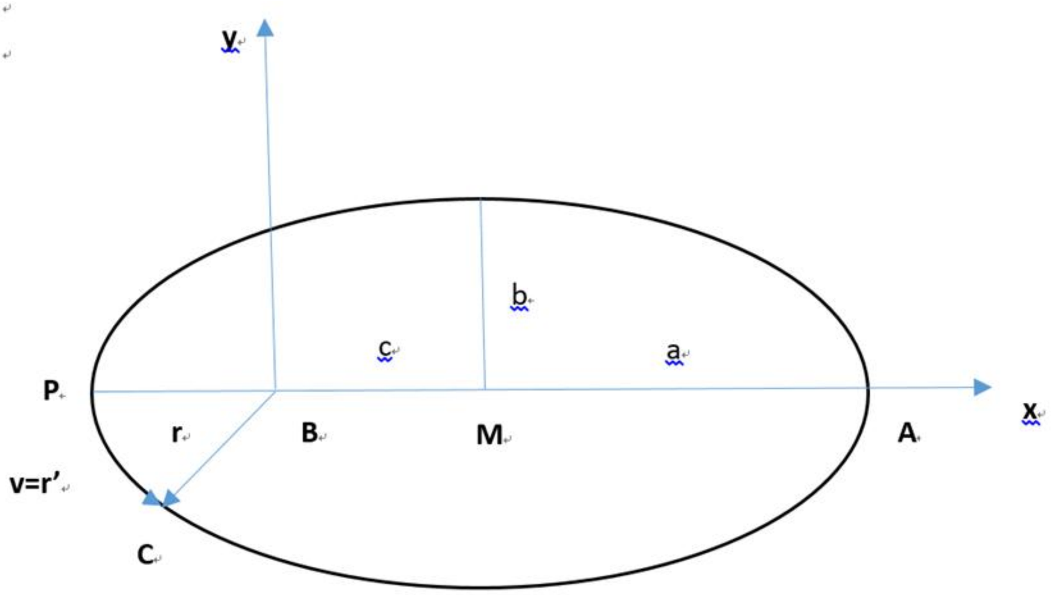


Abbildung 2: Erdbahn

C : Erde

P : Periapsis

A : Apoapsis

B : Sonne

M : Ellipsenzentrum

a : Länge der groβen Halbachse

e : numerische Exzentrizität

1.4.3 Drehimpuls und Drehimpulserhaltungssatz

Der Drehimpuls ist eine physikalische Erhaltungsgröße, die den mechanischen Bewegungszustand eines physikalischen Objekts charakterisiert. Der Drehimpuls nimmt damit eine ähnliche Stellung wie die Energie oder der Impuls ein. Der Drehimpuls eines Körpers ist umso größer, je schneller er sich dreht und je größer sein Massenträgheitsmoment ist. Um den Drehimpuls eines Körpers zu ändern, muss ein Drehmoment auf den Körper einwirken.[4]

$\vec{L} = mr^2\vec{\omega} = \mathrm{m} \cdot \vec{r} \times \vec{r}'$

Der Drehimpuls eines isolierten physikalischen Systems bleibt nach Betrag und Richtung unverändert, egal welche Kräfte und Wechselwirkungen zwischen den Bestandteilen des Systems wirken. Dies wird als Drehimpulserhaltung bezeichnet.
Fast perfekt isolierte Systeme sind die Atomkerne, die Moleküle in verdünnten Gasen und astronomische Objekte im Weltall. Ein Planet bewegt sich auf seiner Bahn umso schneller, je näher er der Sonne ist. Dies folgt aus der Drehimpulserhaltung und wird durch das zweite Keplersche Gesetz ausgedrückt.[5]

1.4.4 Differenzialgleichung von Bahnbewegung

Der Drehimpuls ist

$$\vec{L} = \mathrm{m} \cdot \vec{r} \times \vec{r}' = \mathrm{m} \begin{vmatrix} i & j & k \\ x & y & z \\ x' & y' & z' \end{vmatrix}$$

Bei Einheitsvektoren

$$\mathrm{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathrm{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathrm{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

$$\vec{r}' = \frac{dr}{\mathrm{dt}} = \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix}$$

In Fall Bahnebene : z=0

Dann gilt:

$$L = \mathrm{m}\begin{vmatrix} i & j & k \\ x & y & 0 \\ x' & y' & 0 \end{vmatrix} = \mathrm{m}\begin{pmatrix} 0 \\ 0 \\ xy' - yx' \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix}$$

In Fall c=1,

$$\text{(i)} \qquad xy' - yx' = c = 1$$

Die Umlaufbahn hat idealisert die Form einer Ellipse.So gilt

$$\text{(ii')} \qquad \left(\frac{x(t)-e}{a}\right)^2 + \left(\frac{y(t)}{b}\right)^2 = 1$$

$$e^2 = a^2 - b^2$$

$$a = 1$$

$$\text{(ii)} \qquad (x-e)^2 + \left(\frac{y}{b}\right)^2 = 1$$

Die Gleichungen (i) und (ii) sind ein System nichtliniearer Differentialgleichungen 1.Ordnung. Das ist implizit.

2. Die einfache numerische Lösung des expliziten und impliziten Problems

Aufgabe: (i) $x^2(t)+y^2(\mathrm{t})=1$, (ii) $\mathrm{xy'-yx'}=1$ $(x'=dx/dt, y'=dy/dt)$ und Anfangsbedingungen $x(0)=1$, $y(0)=0$ mit NDSolve implizit lösen und das Ergebnis für "groβe" t mit der exakten sowie der numerischen Lösung des expliziten Problems vergleichen.

Frage: Wie groβ muss t gewählt werden, dass die Unterschiede deutlich werden?

Die Gleichungen (1) $x^2(t)+y^2(\mathrm{t})=1$ und (2) $\mathrm{xy'-yx'}=1$ sind ein System nichtliniearer Differentialgleichungen 1.Ordnung. Das ist implizit.

Und wir können das System auch explizit behandeln.

$$x^2+y^2=1$$

Für $x=x(t)$, $y=y(t)$ differenzieren wir nach t :

$$2x\cdot x'+2y\cdot y'=0 \tag{3}$$

DGL (3) multiplizieren wir mit und x benutzen $xy'=1+yx'$:

$$x^2\cdot x'+y\cdot(1+y\cdot x')=0$$

Jetzt vereinfachen wir:

$$(x^2+y^2)x'+y=0$$

Aufgrund $x^2+y^2=1$, so gibt

$$x'+y=0$$

$$x'=-y \tag{4}$$

DGL (3) multiplizieren wir mit y:

$$xy \cdot x' + y^2 \cdot y' = 0$$

Aufgrund $xy' - yx' = 1$, also $y \cdot x' = x \cdot y' - 1$ ergibt sich

$$x(xy' - 1) + y^2 \cdot y' = 0$$

Jetzt vereinfachen wir:

$$(x^2 + y^2)y' - x = 0$$

Bei $x^2 + y^2 = 1$, so gibt

$$y' - x = 0$$

$$y' = x \qquad (5)$$

Die Gleichungen (4) $x' = -y$ und (5) $y' = x$ sind explizit.

Anwedung von Mathematica:

In[1]:DSolve[{x'[t] == − y[t], y'[t] == x[t], x[0] == 1, y[0] == 0}, {x[t], y[t]}, t]

Out[1]:={{x[t] -> Cos[t], y[t] -> Sin[t]}}

In[2]:=NDSolve[{x'[t] == -y[t], y'[t] == x[t], x[0] == 1, y[0] == 0}, {x[t], y[t]},
{t, 0, 10000}, MaxSteps -> 10000]

NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 1324.0067628254349`. >>

Out[2]:={{x[t] -> InterpolatingFunction[{{0., 1324.01}}, <>][t],
y[t] -> InterpolatingFunction[{{0., 1324.01}}, <>][t]}}

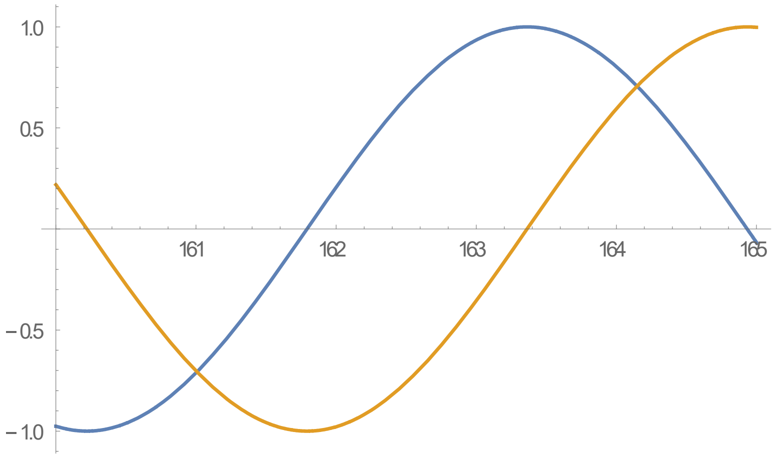
In[3]:={x3[t_], y3[t_]} = {x[t], y[t]} /. %[[1]]

Out[3]:={InterpolatingFunction[{{0., 1324.01}}, <>][t],
InterpolatingFunction[{{0., 1324.01}}, <>][t]}

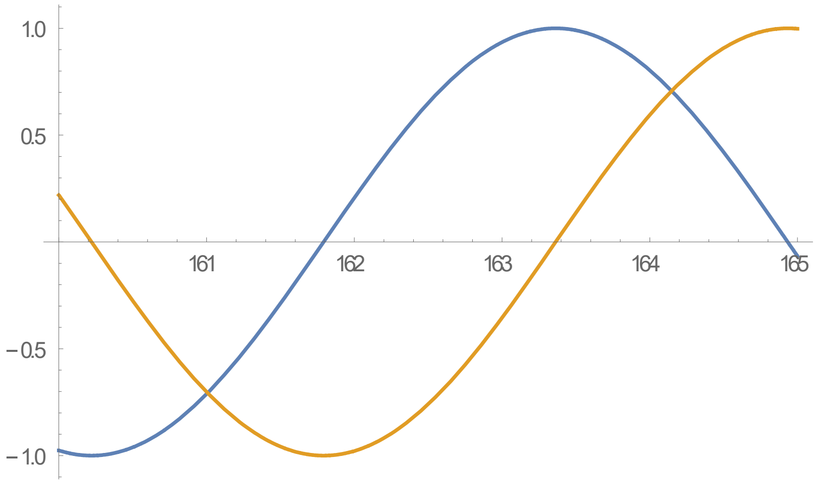
In[4]:=NDSolve[{x[t]^2 + y[t]^2 == 1, x[t]*y'[t] - y[t]*x'[t] == 1,
x[0] == 1, y[0] == 0}, {x[t], y[t]}, {t, 0, 10000}, MaxSteps -> 10000]

NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 164.9420111323749`. >>

```
Out[4]:={{x[t] -> InterpolatingFunction[{{0., 164.942}}, <>][t],
         y[t] -> InterpolatingFunction[{{0., 164.942}}, <>][t]}}
In[5]:={x4[t_], y4[t_]} = {x[t], y[t]} /. %[[1]]
Out[5]:={InterpolatingFunction[{{0., 164.942}}, <>][t],
        InterpolatingFunction[{{0., 164.942}}, <>][t]}
In[6]:=Plot[{x3[t], y3[t]}, {t, 160.0, 165.0}]
Out[6]=
```




```
In[7]:=Plot[{x4[t], y4[t]}, {t, 160.0, 165.0}]
Out[7]=
```



In[8]:={{Cos[164.], x3[164], x4[164]}, {Sin[164.], y3[164], y4[164]}}

Out[8]:={{0.803775,0.803777,0.803774},{0.594933,0.594933,0.594935}}

In[9]:={{Cos[165.], x3[165], x4[165]}, {Sin[165.], y3[165], y4[165]}}

InterpolatingFunction::dmval: Input value {165} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

InterpolatingFunction::dmval: Input value {165} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

Out[9]:={{-0.0663369, -0.0663361, -0.0663395}, {0.997797, 0.997799, 0.997797}}

Davon können wir sehen, hat die explizite Differenzialgleichung in gleichen Fall ($t \in (0,10000), MaxSteps \to 10000$) gröβere Werte für die Variable "t".

Für explizite Differenzialgleichung ist die Grenze der Variable "t" 1324.006.Aber für implizite Differenzialgleichung ist die Grenze der Variable "t" 164.942. Durch Vergleich die Grenze der Variable "t" können wir finden, dass die explizite Differenzialgleichung besser ist.

3. Die numerische Lösung des expliziten und impliziten Problems bei der Kepler-Ellipse

Aufgabe: (6) $(x(t)-e)^2 + y^2(t)/b^2 = 1$, (7) $xy' - yx' = 1$ $(x' = dx/dt, y' = dy/dt)$ und Anfangsbedingungen $x(0) = 1+e$, $y(0) = 0$ $(b = e = 1/\sqrt{2})$ mit NDSolve implizit lösen und das Ergebnis für "groβe" t mit der exakten sowie der numerischen Lösung des expliziten Problems vergleichen. (Die groβe Halbachse wird a=1 gesetzt im Sinne einer Normierung.)

Frage: Wie groβ muss t gewählt werden, dass die Unterschiede deutlich werden?

Die Gleichungen (6) $(x(t)-e)^2 + y^2(t)/b^2 = 1$ und (7) $xy' - yx' = 1$ sind ein System nichtliniearer Differentialgleichungen 1.Ordnung. Das ist implizit.

Und wir können die System auch explizit behandeln.

$$(x-e)^2+(\frac{y}{b})^2 = 1$$

Für $x = x(t), \quad y = y(t)$ differenzieren wir nach t :

$$2(x-e)\cdot x' + 2\frac{y}{b^2}\cdot y' = 0 \qquad (8)$$

DGL (8) multiplizieren wir mit x:

$$(x-e)(x-e+e)x'+\frac{y}{b^2}(1+y\cdot x') = 0$$

Jetzt vereinfachen wir:

$$\left[(x-e)^2 + e(x-e) + \left(\frac{y}{b}\right)^2\right]\cdot x' + \frac{y}{b^2} = 0$$

Aufgrund $(x-e)^2+(\frac{y}{b})^2 = 1$, also

$$[1 + e(x-e)]\cdot x' + \frac{y}{b^2} = 0$$

$$x' = -\frac{\frac{y}{b^2}}{1+e(x-e)} \qquad e<<1 \qquad (9)$$

DGL (8) multiplizieren wir mit y:

$$(x-e)x'y + \left(\frac{y}{b}\right)^2\cdot y' = 0$$

Aufgrund $(x-e)^2+(\frac{y}{b})^2 = 1$, also $\left(\frac{y}{b}\right)^2 = 1-(x-e)^2$

$$(x-e)(xy'-1) + \left(1-(x-e)^2\right)\cdot y' = 0$$

Jetzt vereinfachen wir:

$$(1+e(x-e))\cdot y' - (x-e) = 0$$

$$y' = \frac{x-e}{1+e(x-e)} \qquad e<<1 \qquad (10)$$

Die Gleichungen (9) $x' = -\frac{\frac{y}{b^2}}{1+e(x-e)}$ und (10) $y' = \frac{x-e}{1+e(x-e)}$ sind explizit.

Anwendung von Mathematica :

In[1]:=$b = 1/\sqrt{2}; e = 1/\sqrt{2}$;

In[2]:=NDSolve[{x'[t] == -y[t]/(b^2 (1 + e (x[t] - e))),

y'[t] == (x[t] - e)/(1 + e*(x[t] - e)), x[0] == 1 + e, y[0] == 0},

{x[t], y[t]}, {t, 0, 1000}, MaxSteps -> 10000]

NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 384.6462622758641`. >>

Out[2]:={{x[t] -> InterpolatingFunction[{{0., 384.646}}, <>][t],

y[t] -> InterpolatingFunction[{{0., 384.646}}, <>][t]}}

In[3]:={x1[t_], y1[t_]} = {x[t], y[t]} /. %[[1]]

Out[3]:={InterpolatingFunction[{{0., 384.646}}, <>][t],

InterpolatingFunction[{{0., 384.646}}, <>][t]}

In[4]:=NDSolve[{(x[t] - e)^2 + y[t]^2 == 1, x[t]*y'[t] - y[t]*x'[t] == 1,

x[0] == 1 + e, y[0] == 0}, {x[t], y[t]}, {t, 0, 1000}, MaxSteps -> 10000]

NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 104.61182819139727`. >>

Out[4]:={{x[t] -> InterpolatingFunction[{{0., 104.612}}, <>][t],

y[t] -> InterpolatingFunction[{{0., 104.612}}, <>][t]}}

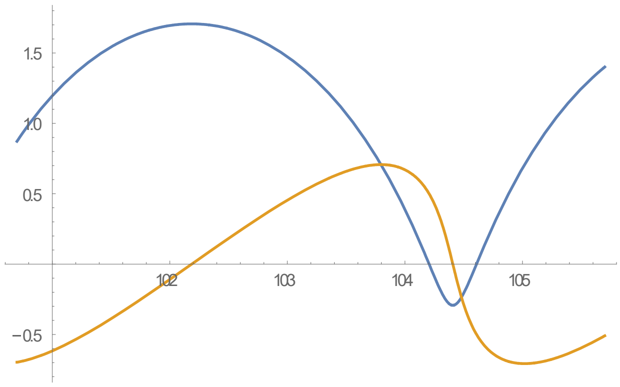
In[5]:={x2[t_], y2[t_]} = {x[t], y[t]} /. %[[1]]

Out[5]:={InterpolatingFunction[{{0., 104.612}}, <>][t],

InterpolatingFunction[{{0., 104.612}}, <>][t]}

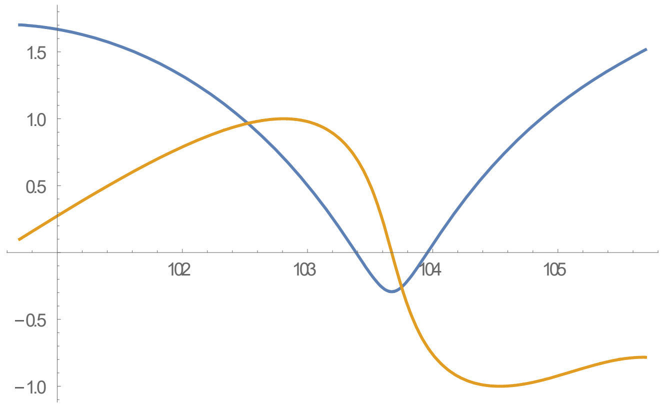
In[6]:=Plot[{x1[t], y1[t]}, {t, 100.7, 105.7}]

Out[6]=



In[7]:=Plot[{x2[t],y2[t]},{t,100.7,105.7}]

Out[7]=



In[8]:={{Cos[105.],x1[105],x2[105]},{Sin[105.],y1[105],y2[105]}}

InterpolatingFunction::dmval: Input value {105} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

InterpolatingFunction::dmval: Input value {105} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

Out[8]:={{-0.240959,0.68073,1.10274},{-0.970535,-0.706856,-0.922373}}

Davon können wir sehen, hat die explizite Differenzialgleichung in gleichen Fall ($t \in (0,1000), MaxSteps \to 10000$) größere Werte für die Variable "t".

Für explizite Differenzialgleichung ist die Grenze der Variable "t" 384.646. Aber für implizite Differenzialgleichung ist die Grenze der Variable "t" 104.611. Durch Vergleich die Grenze der Variable "t" können wir finden, die explizite Differenzialgleichung auch besser ist.

4. Zusammenfassung

Wir testen Berechnungsverfahren für elliptische Bahnen (Bewegung von Himmelskörpern) für explizite und implizite Differenzialgleichungssysteme, und vergleichen die zwei Differenzialgleichungssysteme, um die bessere zu wählen. Durch Vergleich können wir finden, dass die zwei Aufgaben zu gleichem Ergebnis führen. Die Grenze der Variable “t” im expliziten Differenzialgleichungssystem ist gröβer. So sind die Lösungen der expliziten Differenzialgleichungssysteme genauer als die der impliziten Differenzialgleichungssysteme.

5. Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werten (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Sheng Caiwei

6. Quelle

[1] http://reference.wolfram.com/language/ref/NDSolve.html

[2] https://de.wikipedia.org/wiki/Keplersche_Gesetze

[3]http://baike.baidu.com/link?url=_WQFJPmyNdYqaAOAKlyQ7fPSUEUVZPGwrF_BcrfaET1l8jVBJgbLQpU6JV0-H_dW6Rbmll8qopBPu0IRrQyb20pQliOowBgJ6cPP6l6XhDFRGUVwmLhyycAxeMQUvhWA

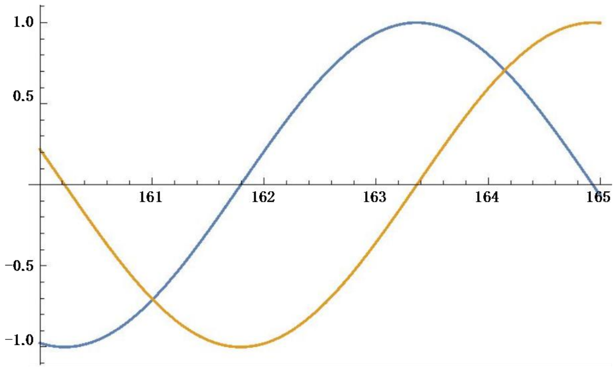
[4] https://www.lernhelfer.de/schuelerlexikon/physik-abitur/artikel/drehimpuls

[5] https://de.wikipedia.org/wiki/Drehimpuls

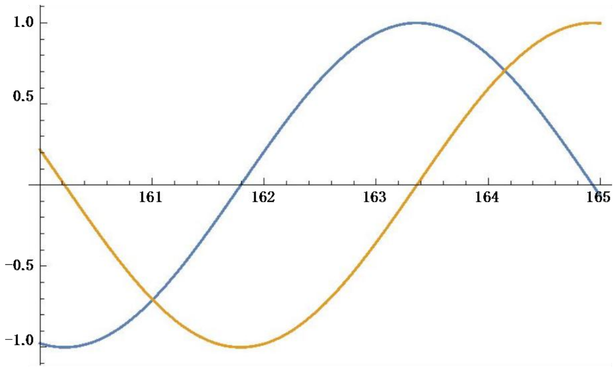
Anhang

```
求解微分方程
{{x[t] → Cos[t], y[t] → Sin[t]}}

NDSolve[{x'[t] == -y[t], y'[t] == x[t], x[0] == 1, y[0] == 0},
数值求解微分方程组
 {x[t], y[t]}, {t, 0, 1000}, MaxSteps → 10 000]
                              最多步数

 x[t] → InterpolatingFunction[ Domain: 0., 1.00×10^3  Output: scalar ][t],

 y[t] → InterpolatingFunction[ Domain: 0., 1.00×10^3  Output: scalar ][t]

{x3[t_], y3[t_]} = {x[t], y[t]} /. %[[1]]

 InterpolatingFunction[ Domain: 0., 1.00×10^3  Output: scalar ][t],

 InterpolatingFunction[ Domain: 0., 1.00×10^3  Output: scalar ][t]

NDSolve[{x[t]^2 + y[t]^2 == 1, x[t] * y'[t] - y[t] * x'[t] == 1, x[0] == 1, y[0] == 0},
数值求解微分方程组
 {x[t], y[t]}, {t, 0, 1000}, MaxSteps → 10 000]
                              最多步数

NDSolve::mxst : Maximum number of 10000 steps reached at the point t == 164.9420111323749`.

 x[t] → InterpolatingFunction[ Domain: {{0., 165.}}  Output: scalar ][t],

 y[t] → InterpolatingFunction[ Domain: {{0., 165.}}  Output: scalar ][t]

{x4[t_], y4[t_]} = {x[t], y[t]} /. %[[1]]

 InterpolatingFunction[ Domain: {{0., 165.}}  Output: scalar ][t],

 InterpolatingFunction[ Domain: {{0., 165.}}  Output: scalar ][t]
```

```
Plot[{x3[t], y3[t]}, {t, 160.0, 165.0}]
```
绘图




```
Plot[{x4[t], y4[t]}, {t, 160.0, 165.0}]
```
绘图




```
{{Cos[164.], x3[164], x4[164]}, {Sin[164.], y3[164], y4[164]}}
```
余弦 正弦

```
{{0.803775, 0.803777, 0.803774}, {0.594933, 0.594933, 0.594935}}
```

```
{{Cos[165.], x3[165], x4[165]}, {Sin[165.], y3[165], y4[165]}}
```
余弦 正弦

InterpolatingFunction::dmval: Input value 165 lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction::dmval: Input value 165 lies outside the range of data in the interpolating function. Extrapolation will be used.

```
{{-0.0663369, -0.0663361, -0.0663395}, {0.997797, 0.997799, 0.997797}}
```

```
b = 1  √2 ; e = 1  √2 ;

NDSolve[{x'[t] == -y[t] / (b^2 (1 + e (x[t] - e))),
   y'[t] == (x[t] - e) / (1 + e * (x[t] - e)), x[0] == 1 + e, y[0] == 0},
  {x[t], y[t]}, {t, 0, 1000}, MaxSteps → 10 000]
```

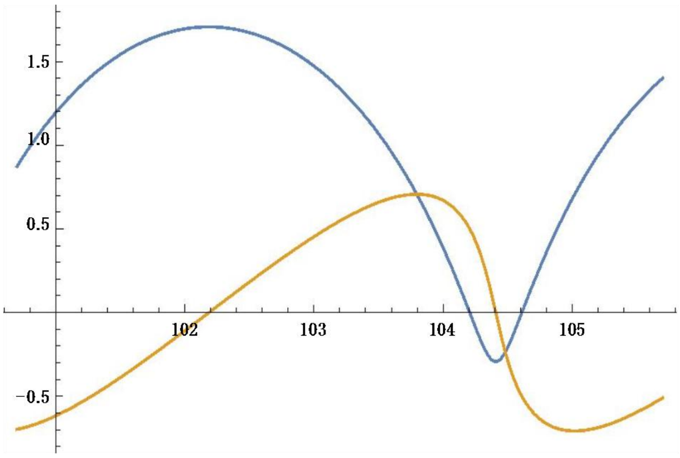
NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 384.6462622758641`.

```
 x[t] → InterpolatingFunction[ Domain: {{0., 385.}} Output: scalar ][t],

 y[t] → InterpolatingFunction[ Domain: {{0., 385.}} Output: scalar ][t]

{x1[t_], y1[t_]} = {x[t], y[t]} /. %[[1]]

 InterpolatingFunction[ Domain: {{0., 385.}} Output: scalar ][t],

 InterpolatingFunction[ Domain: {{0., 385.}} Output: scalar ][t]


NDSolve[{(x[t] - e)^2 + y[t]^2 == 1, x[t] * y'[t] - y[t] * x'[t] == 1,
   x[0] == 1 + e, y[0] == 0}, {x[t], y[t]}, {t, 0, 1000}, MaxSteps → 10 000]
```

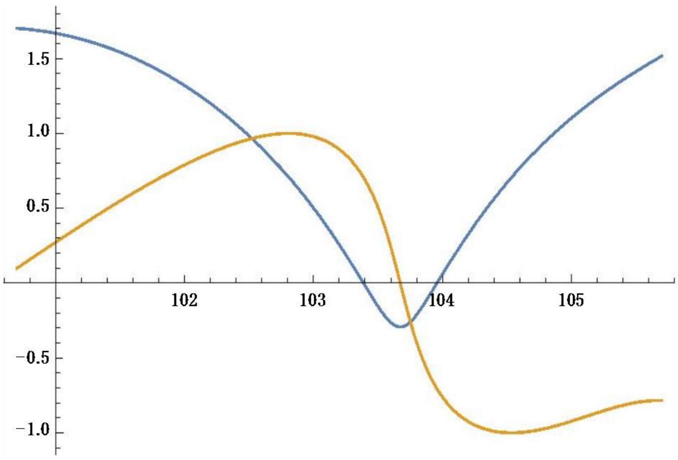
NDSolve::mxst: Maximum number of 10000 steps reached at the point t == 104.61182819139727`.

```
 x[t] → InterpolatingFunction[ Domain: {{0., 105.}} Output: scalar ][t],

 y[t] → InterpolatingFunction[ Domain: {{0., 105.}} Output: scalar ][t]

{x2[t_], y2[t_]} = {x[t], y[t]} /. %[[1]]

 InterpolatingFunction[ Domain: {{0., 105.}} Output: scalar ][t],

 InterpolatingFunction[ Domain: {{0., 105.}} Output: scalar ][t]
```

```
Plot[{x1[t], y1[t]}, {t, 100.7, 105.7}]
```




```
Plot[{x2[t], y2[t]}, {t, 100.7, 105.7}]
```




```
{{Cos[105.], x1[105], x2[105]}, {Sin[105.], y1[105], y2[105]}}
```

InterpolatingFunction::dmval:

Input value {105} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction::dmval:

Input value {105} lies outside the range of data in the interpolating function. Extrapolation will be used.

```
{{-0.240959, 0.68073, 1.10274}, {-0.970535, -0.706856, -0.922373}}
```