Aufgabe 16 Runge-Kutta-Verfahren

Aufgabenteil a)

Implementierung des Runge-Kutta-Verfahrens

Diese Implementierung verwendet die selben Variablenbezeichnungen, wie in der Aufgabenstellung verwendet wurden.
Es werden auch vektorwertige Eingaben, also Differentialgleichungssysteme n-ter Ordnung unterstützt. Letzteres wird insbesondere für Aufgabenteil b) relevant.

```
RungeKutta[phi_, t_, y_, t0_, y0_, intervall_, steps_] :=
 (*Input vectors must be the same size*)
   Eingabe
 Module[{xList, yList, k1, k2, k3, k4, ti, yi, out},
 Modul
  yi = y0;

  out = {};
  For[i = 0, i ≤ (steps), i++,
  For–Schleife
   ti = t0 + intervall * i;
   k1 = phi /. Flatten[{t -> ti, Thread[y -> yi]}];
               ebne ein          fädle auf
   k2 =
    phi /. Flatten[{t -> ti + (intervall / 2), Thread[y -> yi + (intervall / 2) * k1]}];
           ebne ein                                fädle auf
   k3 = phi /. Flatten[{t -> ti + (intervall / 2), Thread[y -> yi + (intervall / 2) * k2]}];
               ebne ein                                fädle auf
   k4 = phi /. Flatten[{t -> ti + intervall, Thread[y -> yi + intervall * k3]}];
               ebne ein                          fädle auf
   yi = yi + (k1 / 6 + k2 / 3 + k3 / 3 + k4 / 6) * intervall;
   out = Append[out, yi];
         hänge an
  ];
  out
 ]
```

Aufgabenteil b)

Import und Setup

Für diesen Aufgabenteil werden die in der Aufgabenstellung gegebenen Daten verwendet. Lediglich das Gewicht von Venus wurde korrigiert.
Folgende Funktionen sind so implementiert, dass das System beliebig erweitert werden kann, sofern die Eingabewerte die selbe Struktur besitzen.

```
SetDirectory[NotebookDirectory[]];
```
lege Verzeichnis·· Notebook–Verzeichnis

```
DATA = Import["Sonnensystem.dat"];
```
importiere

```
Grid[Prepend[DATA, {"Himmelskörper", "Masse in kg",
```
Gitter stelle voran

```
   "x [km]", "y [km]", "z [km]", "Speed x", "Speed y", "Speed z"}],
 Frame → All, Background → {{LightGray, None}, {Gray, None}}]
```
alle Hintergrund hellgrau keine grau keine

Himmelskörper	Masse in kg	x [km]	y [km]	z [km]	Speed x	Speed y	Speed z
Sonne	1.9885×10^{30}	−42549.6	1.09119×10^{6}	−10360.8	−0.0132121	0.00421932	0.000330384
Merkur	3.302×10^{23}	3.46662×10^{7}	-5.31582×10^{7}	-7.62739×10^{6}	31.2959	28.649	−0.53125
Venus	4.8685×10^{24}	8.09436×10^{7}	7.27567×10^{7}	-3.70052×10^{6}	−23.3287	26.082	1.7036
Erde	5.97219×10^{24}	1.16571×10^{8}	9.30296×10^{7}	−14971.2	−18.933	23.2932	−0.00176365
Mond	7.349×10^{22}	1.16322×10^{8}	9.33037×10^{7}	−8261.3	−19.7126	22.5843	0.0893185
Mars	6.4171×10^{23}	2.07999×10^{8}	1.87549×10^{7}	-4.74511×10^{6}	−1.13703	26.2171	0.577176
Jupiter	1.89813×10^{27}	-3.8048×10^{8}	-7.05876×10^{8}	1.14384×10^{7}	11.3461	−5.57743	−0.23066
Saturn	5.6834×10^{26}	2.45756×10^{8}	-1.48395×10^{9}	1.60193×10^{7}	8.99658	1.54748	−0.384934
Uranus	8.6813×10^{25}	2.56406×10^{9}	1.50428×10^{9}	-2.76308×10^{7}	−3.49574	5.55627	0.0658456
Neptun	1.02413×10^{26}	4.32853×10^{9}	-1.14706×10^{9}	-7.61339×10^{7}	1.35653	5.28698	−0.139724

```
NAMES = DATA[[All, 1]];
```
alle

```
MASS = DATA[[All, 2]];
```
alle

```
Pos0 = DATA[[All, {3, 4, 5}]];
```
alle

```
Speed0 = DATA[[All, {6, 7, 8}]];
```
alle

Hilfsfunktionen

Gravity() berechnet einen Beschleunigungsvektor für jeden Planeten im System.

```
Gravity[posVectors_, massVector_] :=
 (*Input vectors must be the same size*)
   Eingabe
 Module[{length, i, j, pos = posVectors, mass = massVector, gravityCONST, out},
 Modul
  out = {};
  gravityCONST = 6.674 * 10^(-20);
  length = Length[pos];
           Länge
  For[i = 1, i ≤ length, i++,
  For–Schleife
   out = Append[out, Evaluate[Sum[gravityCONST * mass[[i]] * mass[[j]] *
         hänge an      werte aus   summiere
        (pos[[j]] - pos[[i]]) / (Piecewise[{{1, Norm[(pos[[j]] - pos[[i]])]^3 == 0}},
                                  stückweise          Norm
           Norm[(pos[[j]] - pos[[i]])]^3]), {j, 1, length}]]]
             Norm


  ];

  out
 ]
```

Sonnensystem führt für das Sonnensystem das Runge-Kutta-Vefahren durch. Anschließend werden die Positionsvektoren aus der Ergebnismenge extrahiert und die Impulsvektoren verworfen.

```
SonnenSystem[pos0Vectors_, Impulse0_, mass_, precision_, steps_] :=
  (*Input vectors must be the same size*)
   Eingabe
  Module[{pp, rr, p, r, phi, px, py, pz, rx, ry, rz, data, t, length, , i, k, tmp, out},
  Modul
   length = Length[pos0Vectors];
            Länge
   pp[i_] = {px_i[t], py_i[t], pz_i[t]};
   rr[i_] = {rx_i[t], ry_i[t], rz_i[t]};
   p[t_] = Array[pp, length];
           Array
   r[t_] = Array[rr, length];
           Array
   phi[t_] := Flatten[{(1 / MASS) * p[t], Flatten[Gravity[r[t], MASS]]}];
              ebne ein                    ebne ein
   data := RungeKutta[phi[t], t, Flatten[{r[t], p[t]}],
                                    ebne ein
     0, Flatten[{Pos0, Impulse0}], precision, steps];
      ebne ein
   data = data[[All, 1 ;; (length * 3)]];
                alle
   (*removing impulse vectors from the set*)
   out = {};
   For[i = 1, i ≤ steps + 1, i++,
   For–Schleife
    (*(steps+1) because RungeKutta includes set of y0"*)
    tmp = {};
    For[k = 0, k < length, k++,
    For–Schleife
     tmp = Append[tmp, data[[i, (3 * k + 1) ;; (3 * k + 3)]]];
           hänge an


    ];
    out = Append[out, tmp];
          hänge an
   ];
   out
  ];
```

DynamicSonnensystemAnimation() erstellt eine Animation. Für die Berechnung der Daten wird die Hilfsfunktion Sonnensystem() verwendet.

```
(*km/s,km,years*)

DynamicSonnensystemAnimation[planetNames_, pos0Vectors_,
  speed0Vectors_, mass_, years_, evaluationsteps_, plotRange_] :=
 (*Input vectors must be the same size*)
   Eingabe
 Module[{r, impulse0Vectors, intervall},
 Modul


  impulse0Vectors = mass * speed0Vectors;
```

```
intervall = (years * 365 * 24 * 60 * 60) / evaluationsteps;
Clear[dataset, INTERVALL, colors];
dataset = {};
INTERVALL = intervall;

dataset =
 SonnenSystem[pos0Vectors, impulse0Vectors, mass, intervall, evaluationsteps];

plotrange = plotRange;
Manipulate[



 Graphics3D[Dynamic@
    (*produces errors as long as calculation
     of dataset is running. Does not affect end result*)

    Table[{ColorData["DarkBands"][c / Length[planetNames]],
       PointSize[.005], Point[dataset[[t, c]]]}, {c, 1, Length[planetNames]}],

   Axes → True,
   AxesStyle → Directive[FontSize → 17, FontFamily → "Helvetica"],
   AxesLabel → {"x", "y", "z"}, BoxRatios → {1, 1, 1},
   ImageSize → 1000, PlotRange → plotrange, SphericalRegion → True],
  Row[{Control[{t, 1, evaluationsteps, 1, Animator, ImageSize → Small,
      AnimationRunning → False, AnimationRate → 10, AnimationRepetitions → 1}],
    Spacer[10],
    Dynamic["Day no. " <> ToString[ Round[(((t - 1) * INTERVALL) / (60 * 60 * 24))]]],
    "              " Grid[{Flatten[Table[{ColorData["DarkBands"][
            c / Length[planetNames]]}, {c, 1, Length[planetNames]}]], NAMES}]}],
   FrameMargins → 0

 ]
]
```

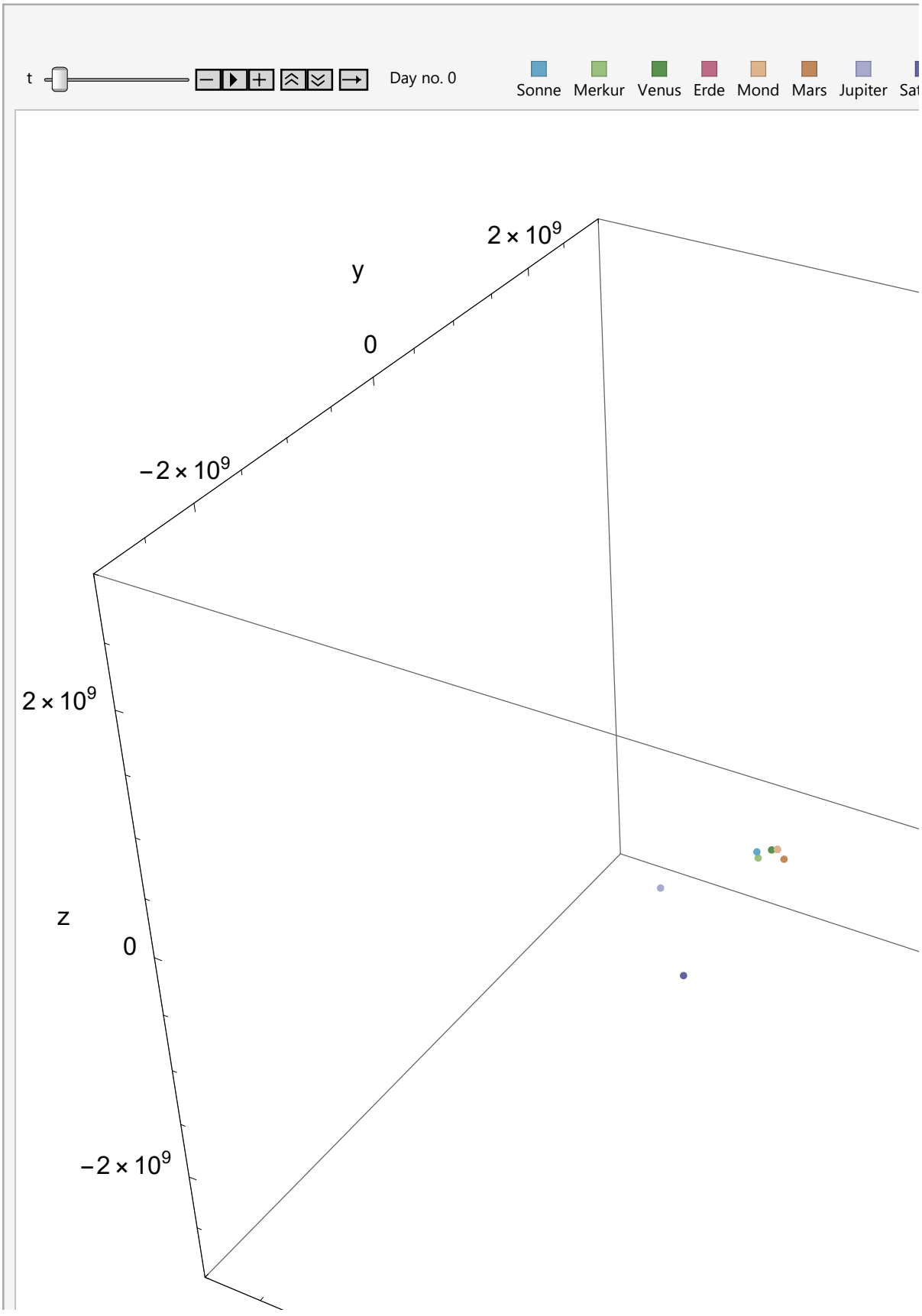
Demo mit allen Anfangswerten:

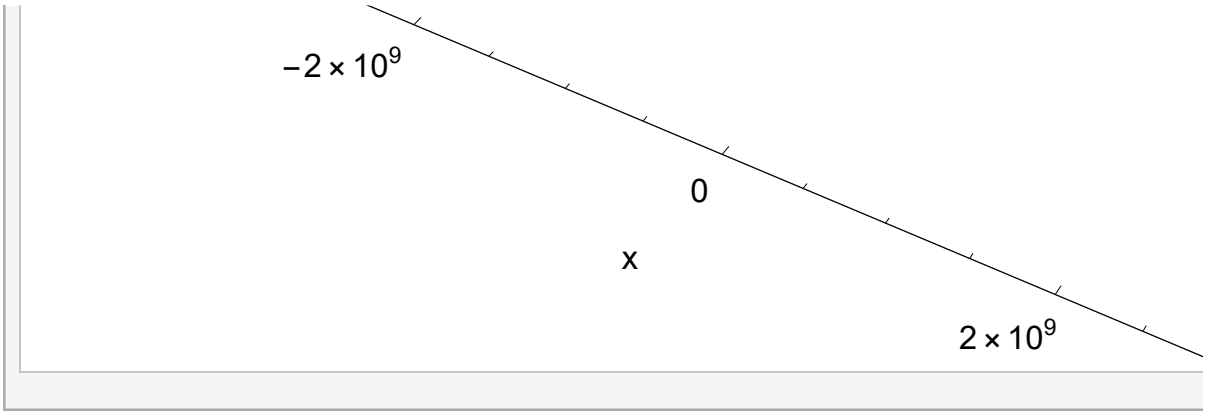
```
(*IMPORTANT: Enable Dynamic Updating*)
```

dynamisch

In[56]:=
```
DynamicSonnensystemAnimation[NAMES, Pos0, Speed0, MASS, 1, 300, 3 * 10^9]
```

Out[56]=





Export der Animation als Video in das Verzeichnis dieses Notebooks

```
Export["Sonnensystem.avi",
 DynamicSonnensystemAnimation[NAMES, Pos0, Speed0, MASS, 1, 100, 3 * 10^9]]
```

exportiere

Sonnensystem.avi