



Die Zukunft unseres Sonnensystems

Wo liegen die Grenzen der Vorhersagbarkeit?

Gliederung

Grundlagen



Chaos



Simulation



Vorhersagen



Weitere Zukunftsaussichten



Fazit

Grundlagen

Unser Sonnensystem

8 Planeten



300+ Monde

5 Zwergplaneten



ca. 1,4 Millionen Asteroiden
ca. 4000 Kometen

Grundlagen

n-Körper-Problem

- Beispiel: Sonne-Erde-Mond; Sonnensystem
- Muss für " $n_{\text{Körper}} > 2$ " numerisch gelöst werden
- Sehr hoher Rechenaufwand
- Allgemein nicht-periodisch und chaotisch

$$F = a * m = G \frac{m_1 m_2}{r^2}$$

≍

$$\overrightarrow{a_i(t)} = G \sum_{i \neq j} \frac{m_j}{\|\overrightarrow{r_{ji}(t)}\|^3} \overrightarrow{r_{ji}(t)}$$



Chaos

Grundlagen

Chaos

Simulation

Vorhersagen

Weitere
Zukunftsaussichten

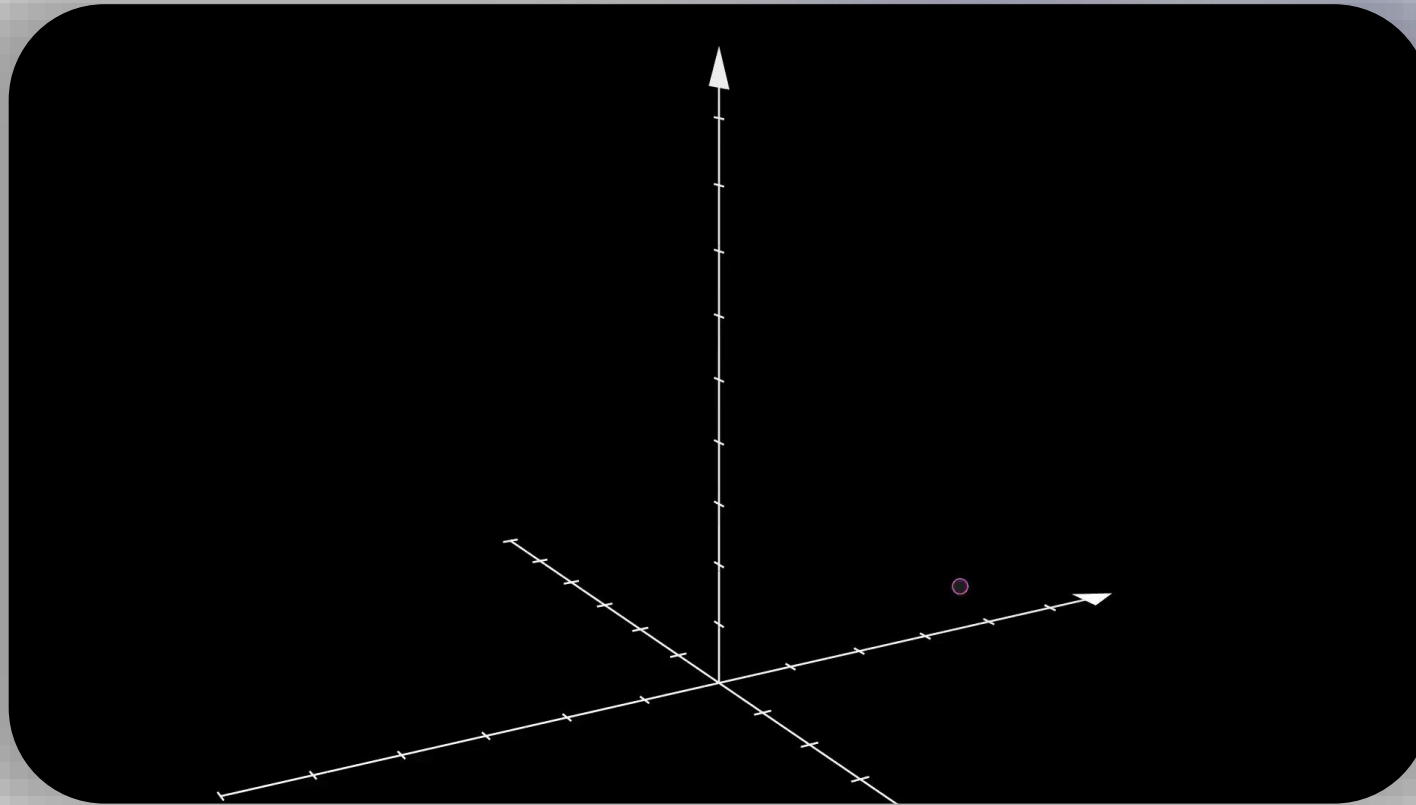
Fazit

Chaos

Sensitive Abhängigkeit von den Anfangsbedingungen „Schmetterlingseffekt“

- Kleine Fehler in den Anfangsbedingungen wachsen exponentiell
- Beschreibt Langzeitverhalten eines dynamischen Systems
- Im Allgemeinen nicht Periodisch
- Trotzdem noch Deterministisch
 - Gedankenexperiment: Laplacescher Dämon

Chaos



Beispiel chaotisches System:
Lorenz Attraktor

Chaos

In unserem Sonnensystem

- Ebenfalls n-Körper-System
 - Sonne, Planeten, Asteroiden und Kometen, Einflüsse außerhalb des Sonnensystems
 - Besonderheit: Ein Körper wesentlich größer als alle anderen » Sonne
- Lyapunov-Zeit von ≈ 5 Millionen Jahren
 - Nach wenigen Lyapunov-Zeiten verschwinden genaue Vorhersagbarkeiten
- Vorhersagen mithilfe von Simulationen

$$\|\delta(t)\| \approx e^{\lambda t} \|\delta_0\|$$

$\delta \hat{=}$ Differenz zweier Systeme

$\lambda \hat{=}$ Lyapunov-Exponent



Simulation

Grundlagen

Chaos

Simulation

Vorhersagen

Weitere
Zukunftsaussichten

Fazit

```

1  G = 6.6743e-11 # Gravitationskonstante in m^3 kg^-1 s^-2
2  AU_to_m = 1.496e11 # Astronomische Einheit in Meter
3  AUday_to_ms = AU_to_m / 86400 # AU/Tag in m/s
4
5  with open("solar_system.json", "r") as f:
6      data = json.load(f)
7  labels = [
8      "Sun",
9      "Mercury",
10     "Venus",
11     "Earth",
12     "Mars",
13     "Jupiter",
14     "Saturn",
15     "Uranus",
16     "Neptune",
17     "Pluto",
18 ]
19 rows = []
20 for name in labels:
21     if name in data:
22         body = data[name]
23         mass = body["mass"]
24         pos = body["position"]
25         vel = body["velocity"]
26         row = [mass, pos["x"], pos["y"], pos["z"], vel["vx"], vel["vy"], vel["vz"]]
27         rows.append(row)
28 # [mass, x, y, z, vx, vy, vz]
29 bodies = np.array(rows, dtype=np.float64)
30 bodies[:, 1:4] *= AU_to_m
31 bodies[:, 4:7] *= AUday_to_ms
32
33 dt = 86400 / 100 # Zeitschritt (in Sekunden) - 1/100 Tag
34 tolerance = 1e1 # Toleranz für die Simulation
35 axis_limit = 1e12 # Achsenlimit für die Plots

```

```

1 def update(frame):
2     global bodies, sim_steps, display_index
3     bodies_new, corrected_dt = rkdp45(bodies, dt)
4     bodies[:] = bodies_new
5     sim_steps += 1
6     sim_time[sim_steps] = sim_time.get(sim_steps - 1, 0) + corrected_dt
7
8     for i in range(bodies.shape[0]):
9         history[i].append(bodies[i, 1:4].copy())
10
11     # Update Slider-Bereich dynamisch
12     slider.valmax = sim_steps
13     slider.ax.set_xlim(slider.valmin, slider.valmax)
14
15     # Automatische Slider-Bewegung nur wenn nicht pausiert
16     if not paused and not slider_active:
17         display_index = sim_steps
18         slider.set_val(display_index)
19
20     update_display(display_index)
21     return planets + trails + [time_text]
22
23
24 ani = FuncAnimation(fig, update, interval=0, cache_frame_data=False)
25 plt.show()

```

```

1 def update(frame):
2     global bodies, sim_steps, display_index
3     bodies_new, corrected_dt = rkdp45(bodies, dt)
4     bodies[:] = bodies_new
5     sim_steps += 1
6     sim_time[sim_steps] = sim_time.get(sim_steps - 1, 0) + corrected_dt
7
8     for i in range(bodies.shape[0]):
9         history[i].append(bodies[i, 1:4].copy())
10
11     # Update Slider-Bereich dynamisch
12     slider.valmax = sim_steps
13     slider.ax.set_xlim(slider.valmin, slider.valmax)
14
15     # Automatische Slider-Bewegung nur wenn nicht pausiert
16     if not paused and not slider_active:
17         display_index = sim_steps
18         slider.set_val(display_index)
19
20     update_display(display_index)
21     return planets + trails + [time_text]
22
23
24 ani = FuncAnimation(fig, update, interval=0, cache_frame_data=False)
25 plt.show()

```

```

1  @nb.njit(fastmath=True, parallel=True)
2  def acceleration(bodies):
3      n = bodies.shape[0]
4      acc = np.zeros((n, 7))
5      for i in range(n):
6          acc[i, 0] = 0.0 # Massen-Ableitung = 0
7          acc[i, 1:4] = bodies[i, 4:7] # dx/dt = vx, dy/dt = vy, dz/dt = vz
8          for j in range(n):
9              if i == j:
10                 continue
11                 dx = bodies[j, 1] - bodies[i, 1]
12                 dy = bodies[j, 2] - bodies[i, 2]
13                 dz = bodies[j, 3] - bodies[i, 3]
14                 dist = np.sqrt(dx * dx + dy * dy + dz * dz)
15                 acc[i, 4] += G * bodies[j, 0] * dx / (dist**3)
16                 acc[i, 5] += G * bodies[j, 0] * dy / (dist**3)
17                 acc[i, 6] += G * bodies[j, 0] * dz / (dist**3)
18  return acc

```

$$\overrightarrow{a_i(t)} = G \sum_{i \neq j} m_i \frac{r_{ji}(t)}{\|r_{ji}(t)\|^3}$$

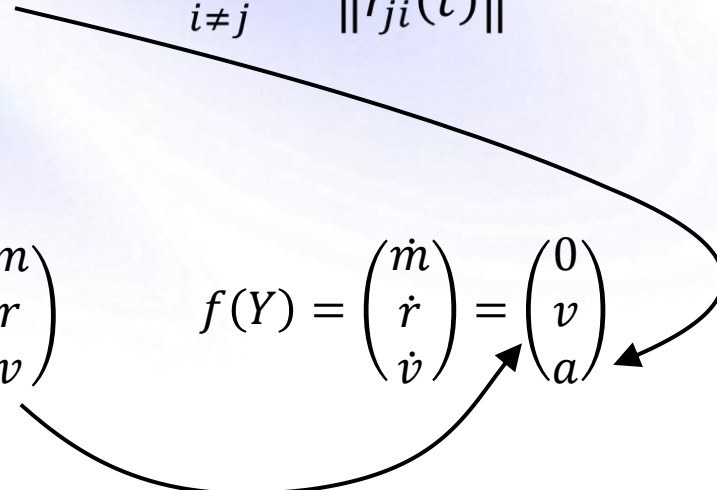
$$Y_o = \begin{pmatrix} m \\ r \\ v \end{pmatrix} \quad f(Y) = \begin{pmatrix} \dot{m} \\ \dot{r} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 \\ v \\ a \end{pmatrix}$$

```

1 @nb.njit(fastmath=True, parallel=True)
2 def euler(bodies, dt):
3     acc = acceleration(bodies)
4     bodies[:, 1:4] += dt * bodies[:, 4:7]
5     bodies[:, 4:7] += dt * acc[:, 4:7]
6     return bodies, dt

```

$$\overrightarrow{a_i(t)} = G \sum_{i \neq j} m_i \frac{r_{ji}(t)}{\|r_{ji}(t)\|^3}$$

$$Y_o = \begin{pmatrix} m \\ r \\ v \end{pmatrix} \quad f(Y) = \begin{pmatrix} \dot{m} \\ \dot{r} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 \\ v \\ a \end{pmatrix}$$


$$Y_{t+\Delta t} = \begin{pmatrix} m \\ r \\ v \end{pmatrix} = \int_t^{t+\Delta t} f(Y) dY$$


```

1 @nb.njit(fastmath=True, parallel=True)
2 def rkdp45(bodies, dt):
3     # c-Werte (Zeitanteile)
4     c2, c3, c4, c5, c6, c7 = 1 / 5, 3 / 10, 4 / 5, 8 / 9, 1.0, 1.0
5
6     # Butcher-Tabellen-Koeffizienten:
7     a21 = 1 / 5
8     a31, a32 = 3 / 40, 9 / 40
9     a41, a42, a43 = 44 / 45, -56 / 15, 32 / 9
10    a51, a52, a53, a54 = 19372 / 6561, -25360 / 2187, 64448 / 6561, -212 / 729
11    a61, a62, a63, a64, a65 = 9017 / 3168, -355 / 33, 46732 / 5247, 49 / 176, -5103 / 18656
12    a71, a72, a73, a74, a75, a76 = 35 / 384, 0.0, 500 / 1113, 125 / 192, -2187 / 6784, 11 / 84
13
14    # Koeffizienten für 5. Ordnung (5th-Order Lösung):
15    b1, b2, b3, b4, b5, b6 = 35 / 384, 0.0, 500 / 1113, 125 / 192, -2187 / 6784, 11 / 84
16    # b7 = 0.0 implizit
17
18    # Koeffizienten für die 4. Ordnung (eingebettete Lösung):
19    b1s, b2s, b3s, b4s, b5s, b6s, b7s = 5179 / 57600, 0.0, 7571 / 16695, 393 / 640, -92097 / 339200, 187 / 2100, 1 / 40
20
21    k1 = acceleration(bodies)
22    k2 = acceleration(bodies + dt * (a21 * k1))
23    k3 = acceleration(bodies + dt * (a31 * k1 + a32 * k2))
24    k4 = acceleration(bodies + dt * (a41 * k1 + a42 * k2 + a43 * k3))
25    k5 = acceleration(bodies + dt * (a51 * k1 + a52 * k2 + a53 * k3 + a54 * k4))
26    k6 = acceleration(bodies + dt * (a61 * k1 + a62 * k2 + a63 * k3 + a64 * k4 + a65 * k5))
27    k7 = acceleration(bodies + dt * (a71 * k1 + a72 * k2 + a73 * k3 + a74 * k4 + a75 * k5 + a76 * k6))
28
29    y5 = bodies + dt * (b1 * k1 + b2 * k2 + b3 * k3 + b4 * k4 + b5 * k5 + b6 * k6)
30    y4 = bodies + dt * (b1s * k1 + b2s * k2 + b3s * k3 + b4s * k4 + b5s * k5 + b6s * k6 + b7s * k7)
31    error = np.linalg.norm(y5 - y4)
32
33    print("dt:", dt, "error:", error)
34    if error > tolerance:
35        dt = 0.99 * dt * (tolerance / error) ** 0.01
36        return rkdp45(bodies, dt)
37    return y5, dt

```

```
python n-body-system-3d.py + v
D:\5. PK\planets sim git:(clean-no-lfs)±1 (24.695s)
python n-body-system-3d.py
dt: 8640.0 error: 0.00021088739823063943
dt: 8640.0 error: 0.00021582672974188822
dt: 8640.0 error: 0.0002149012258524981
dt: 8640.0 error: 0.00021395368476679653
dt: 8640.0 error: 0.00021304605149891624
dt: 8640.0 error: 0.00021215430611568132
dt: 8640.0 error: 0.00021126311111111111
dt: 8640.0 error: 0.00021037202222222222
dt: 8640.0 error: 0.00020948093333333333
dt: 8640.0 error: 0.00020859004444444444
dt: 8640.0 error: 0.00020769915555555556
dt: 8640.0 error: 0.00020680826666666667
dt: 8640.0 error: 0.00020591737777777778
dt: 8640.0 error: 0.00020502648888888889
dt: 8640.0 error: 0.00020413559999999999
dt: 8640.0 error: 0.00020324471111111111
dt: 8640.0 error: 0.00020235382222222222
dt: 8640.0 error: 0.00020146293333333333
dt: 8640.0 error: 0.00020057204444444444
dt: 8640.0 error: 0.000200005795615007672
dt: 8640.0 error: 0.0001993946447180948
dt: 8640.0 error: 0.00019863629803153275
dt: 8640.0 error: 0.00019805360271915115
dt: 8640.0 error: 0.00019731729650149715
dt: 8640.0 error: 0.00019668186418540234
dt: 8640.0 error: 0.00019600834545404372
dt: 8640.0 error: 0.00019533897687898383
dt: 8640.0 error: 0.00019477951921814936
dt: 8640.0 error: 0.00019416646535119674
dt: 8640.0 error: 0.00019361224691091474
dt: 8640.0 error: 0.00019296196633185957
dt: 8640.0 error: 0.00019236963297626378
dt: 8640.0 error: 0.00019179091540419134
dt: 8640.0 error: 0.00019130884549546404
dt: 8640.0 error: 0.00019073037403959954
dt: 8640.0 error: 0.00019021538075133105
dt: 8640.0 error: 0.0001896824004034355
dt: 8640.0 error: 0.00018921788969058546
dt: 8640.0 error: 0.00018866060988932326
dt: 8640.0 error: 0.00018821595137130784
dt: 8640.0 error: 0.00018767499272887112
dt: 8640.0 error: 0.00018730650923568876
dt: 8640.0 error: 0.0001868524549497479
dt: 8640.0 error: 0.00018634546341326485
dt: 8640.0 error: 0.0001859643508643321
dt: 8640.0 error: 0.00018552345460412512
dt: 8640.0 error: 0.00018504767041882194
dt: 8640.0 error: 0.00018466165790078016
dt: 8640.0 error: 0.00018426017245093195
dt: 8640.0 error: 0.0001839054790011843

D:\5. PK\planets sim git:(clean-no-lfs)±1
python n-body-system-3d.py
D:\5. PK\planets sim\n-body-system-3d.py:248: UserWarning: Attempting to set identical low and high xlims makes transformation singular; automatically expanding.
  slider = Slider(ax_slider, "History", 0, 0, valinit=0, valstep=1)
```

Vorhersagen

- Laskar 2008
 - Statistische Wahrscheinlichkeiten für Instabilitäten
 - Einfluss von relativistischen Effekten
 - 1001 Durchführen
- Laskar 2012
 - Erklärt historischen Kontext
 - Zusammenfassung früherer Studien und deren Aussagen

Vorhersagen

- Brown & Rein 2020
 - Öffentlicher Datensatz zu 96 Durchführungen
 - Volle N-Körper-Integration mit Open-Source-Tools
- Brown & Rein 2022
 - 2880 Durchführungen mit Open-Source Code

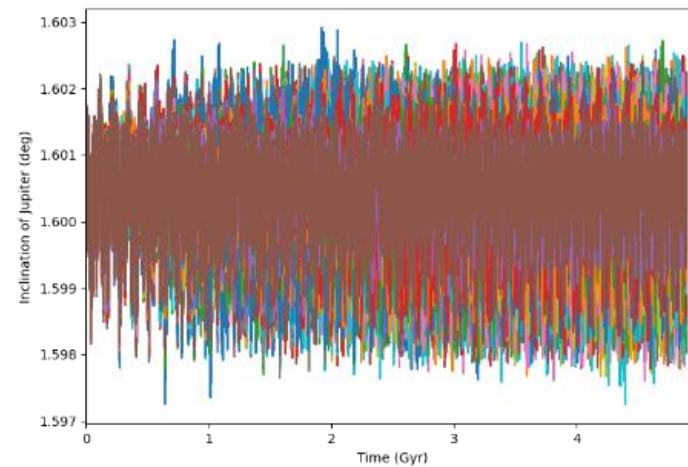
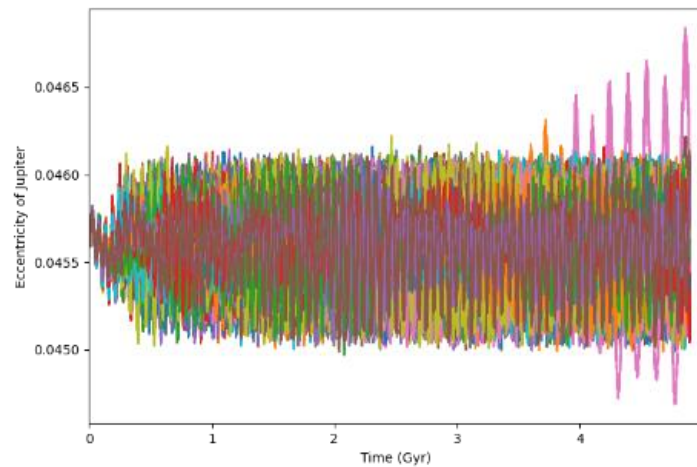
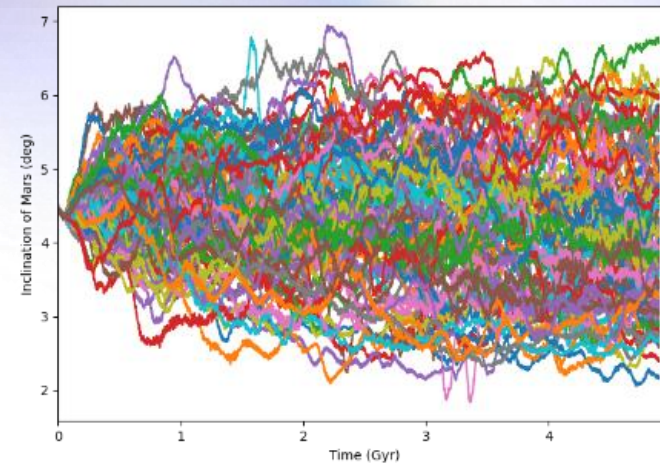
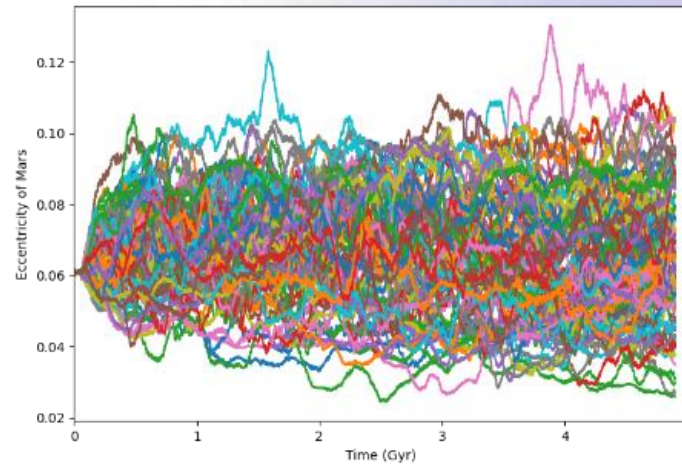
Vorhersagen

Innere & Äußere Planeten

Gyr interval. As it was mentioned before, (Laskar, 1990, 1994), there is practically no diffusion for the outer planet system that behaves nearly as a quasiperiodic and regular system. On the opposite, there is a significant diffusion of the eccentricities and inclinations of the inner planets. The statistics on the maximum values reached by the ec-

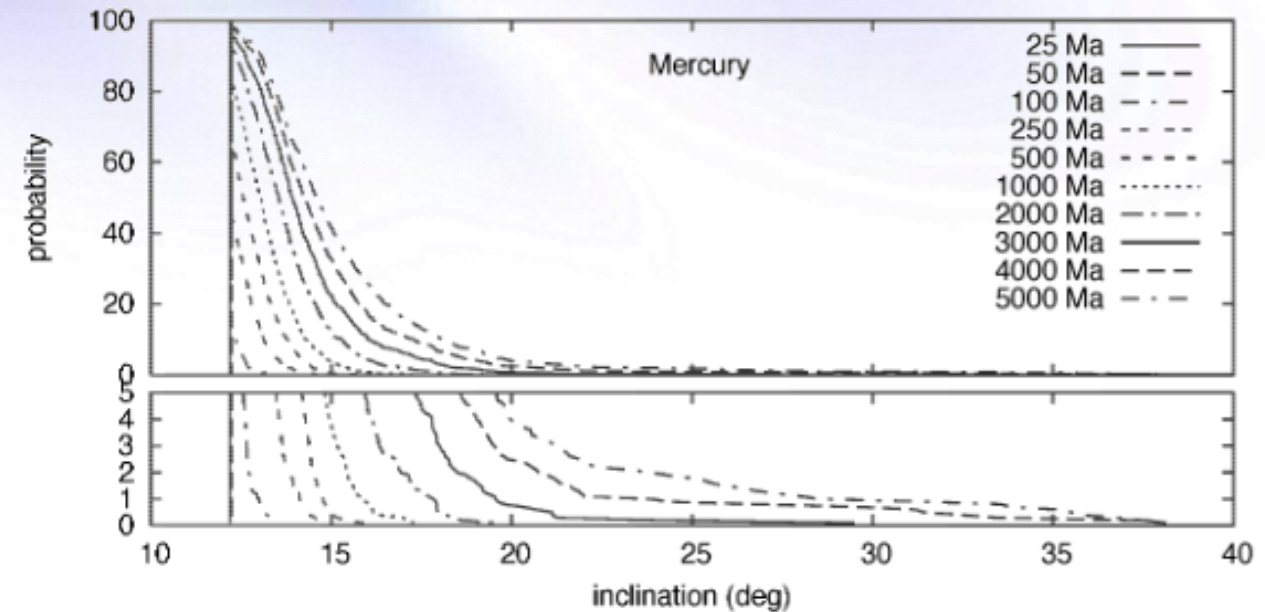
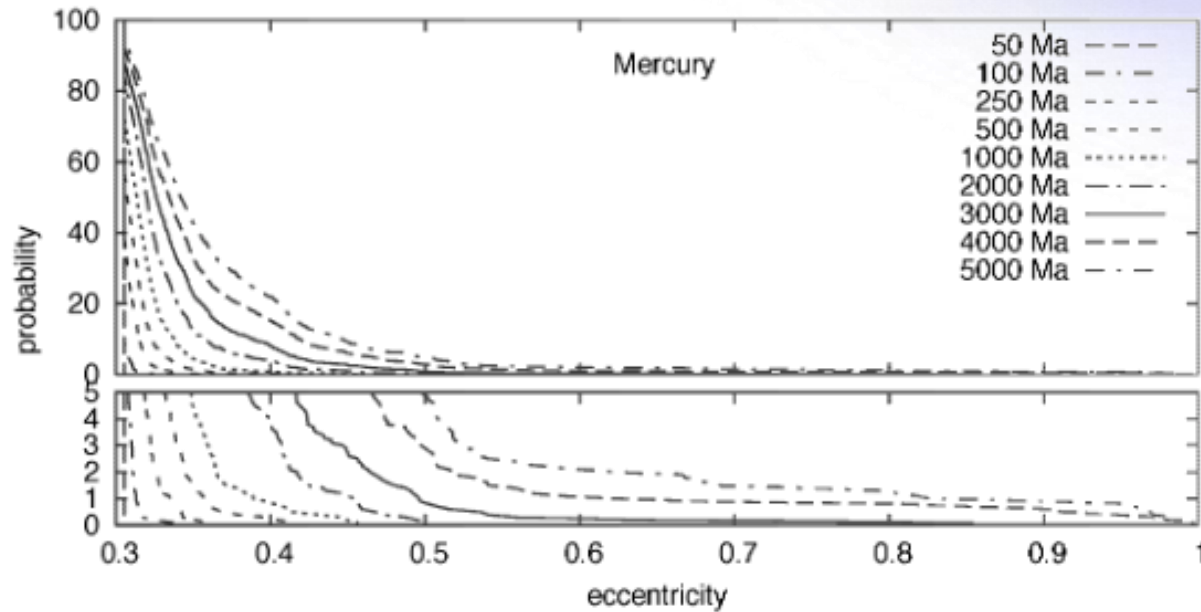
Vorhersagen

Innere & Äußere Planeten



Vorhersagen

Merkur

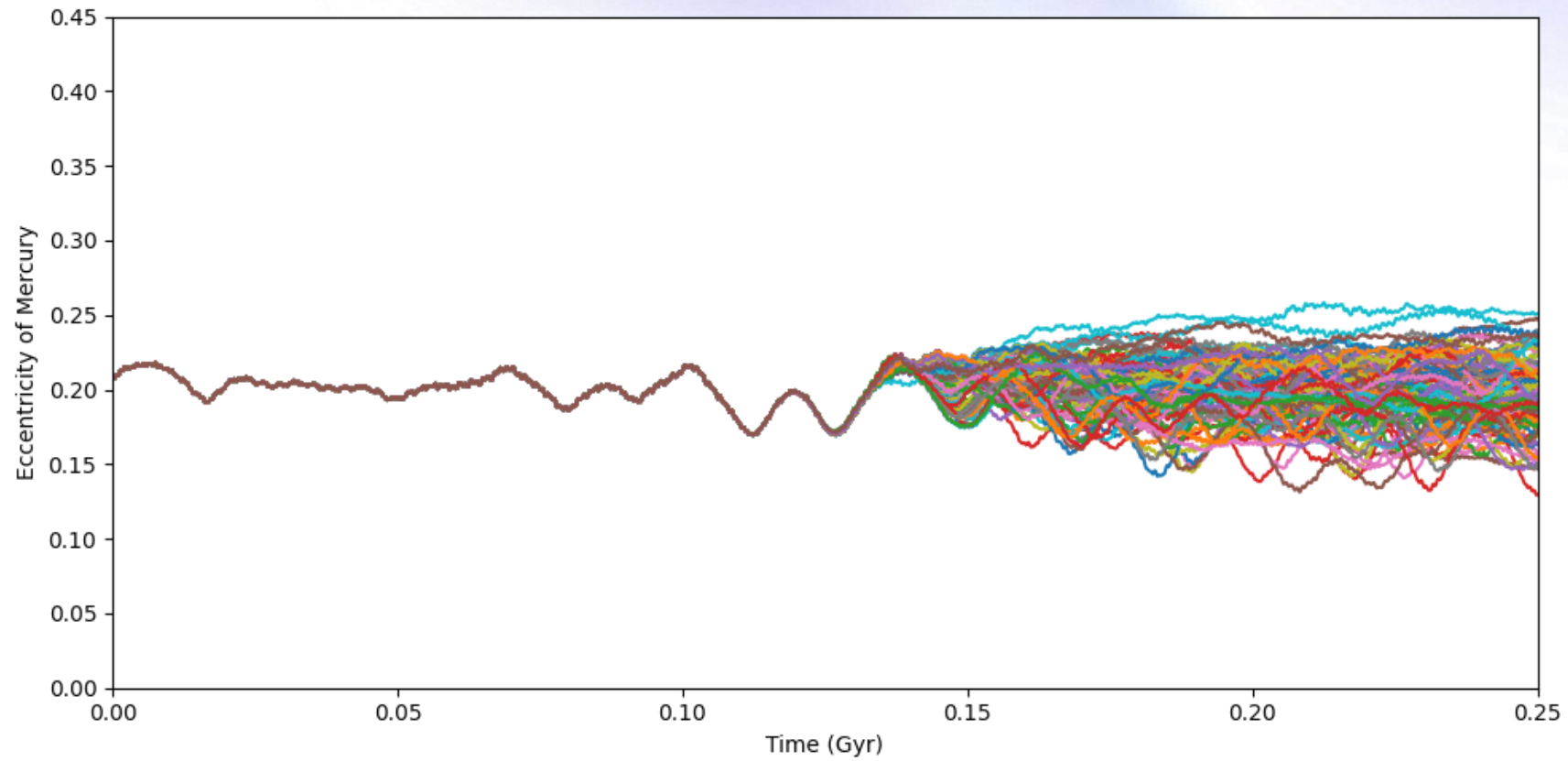


$e \gtrsim 0.8 \rightarrow$ Gefahr Kollision mit Venus
 $e \gtrsim 0.95 \rightarrow$ Gefahr Kollision mit Sonne
 $e > 1 \rightarrow$ Rauswurf

Abb. 3: Laskar Diagramme der inneren Planeten

Vorhersagen

Merkur



Vorhersagen

Newton vs. Relativität

Newtonsche Mechanik

e_{m0}	500	1000	1500	2000	3000	4000	5000
0.35	130	341	478	558	692	763	812
0.40	75	249	373	449	589	684	747
0.50	24	118	226	306	442	552	640
0.60	16	76	169	238	364	476	564
0.70	14	67	150	218	343	454	541
0.80	12	63	141	209	331	442	531
0.90	12	61	138	202	325	441	530

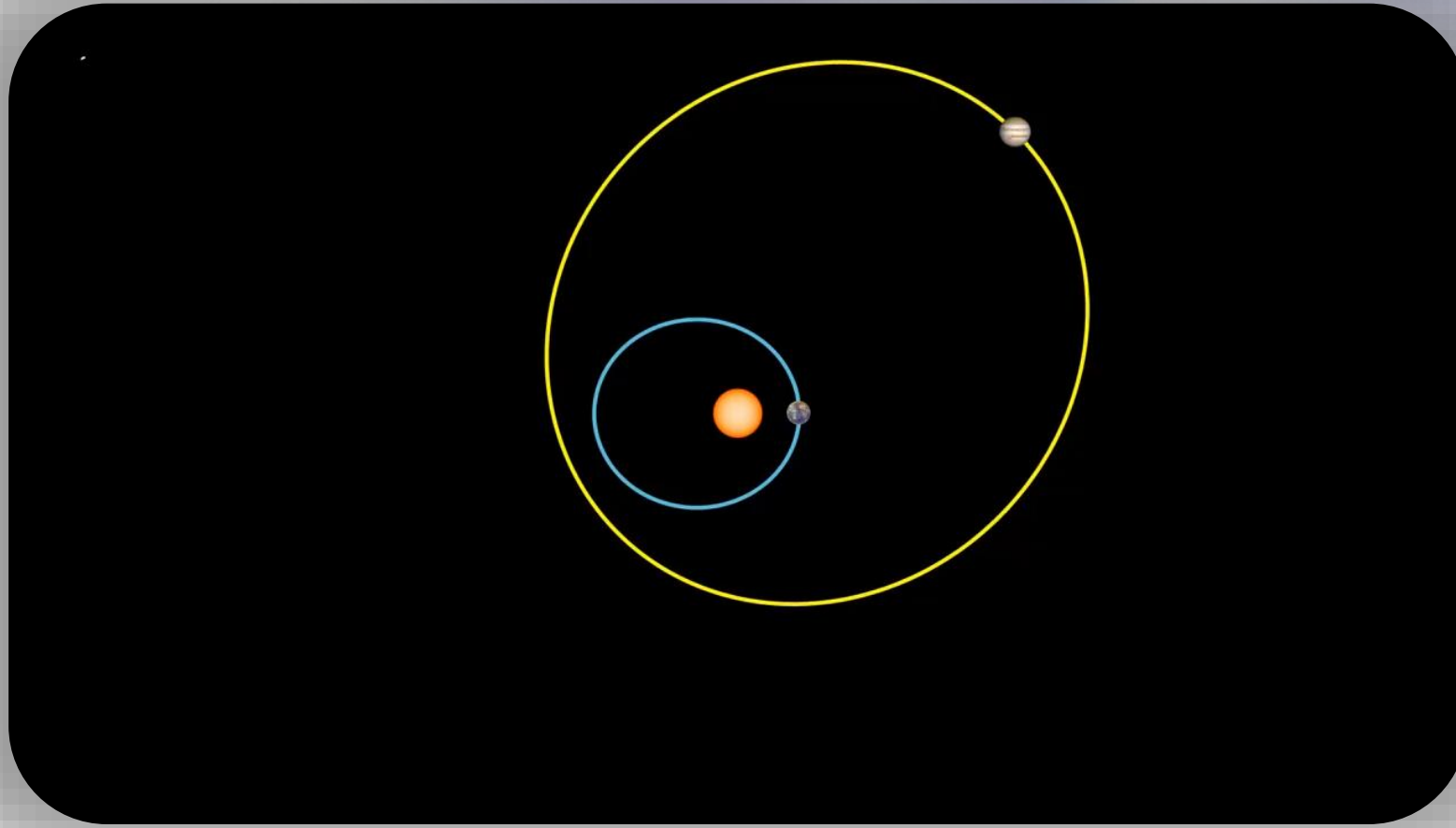
Allgemeine Relativitätstheorie

e_{m0}	500	1000	1500	2000	3000	4000	5000
0.35	25	75	128	165	280	366	427
0.40	4	21	38	52	113	180	243
0.50	0	0	0	0	6	19	33
0.60	0	0	0	0	0	6	10
0.70	0	0	0	0	0	6	10
0.80	0	0	0	0	0	2	8
0.90	0	0	0	0	0	0	2

Abb. 4: Laskar Diagramme – Vergleich Newton vs. GR

Vorhersagen

Säkulare Resonanzen



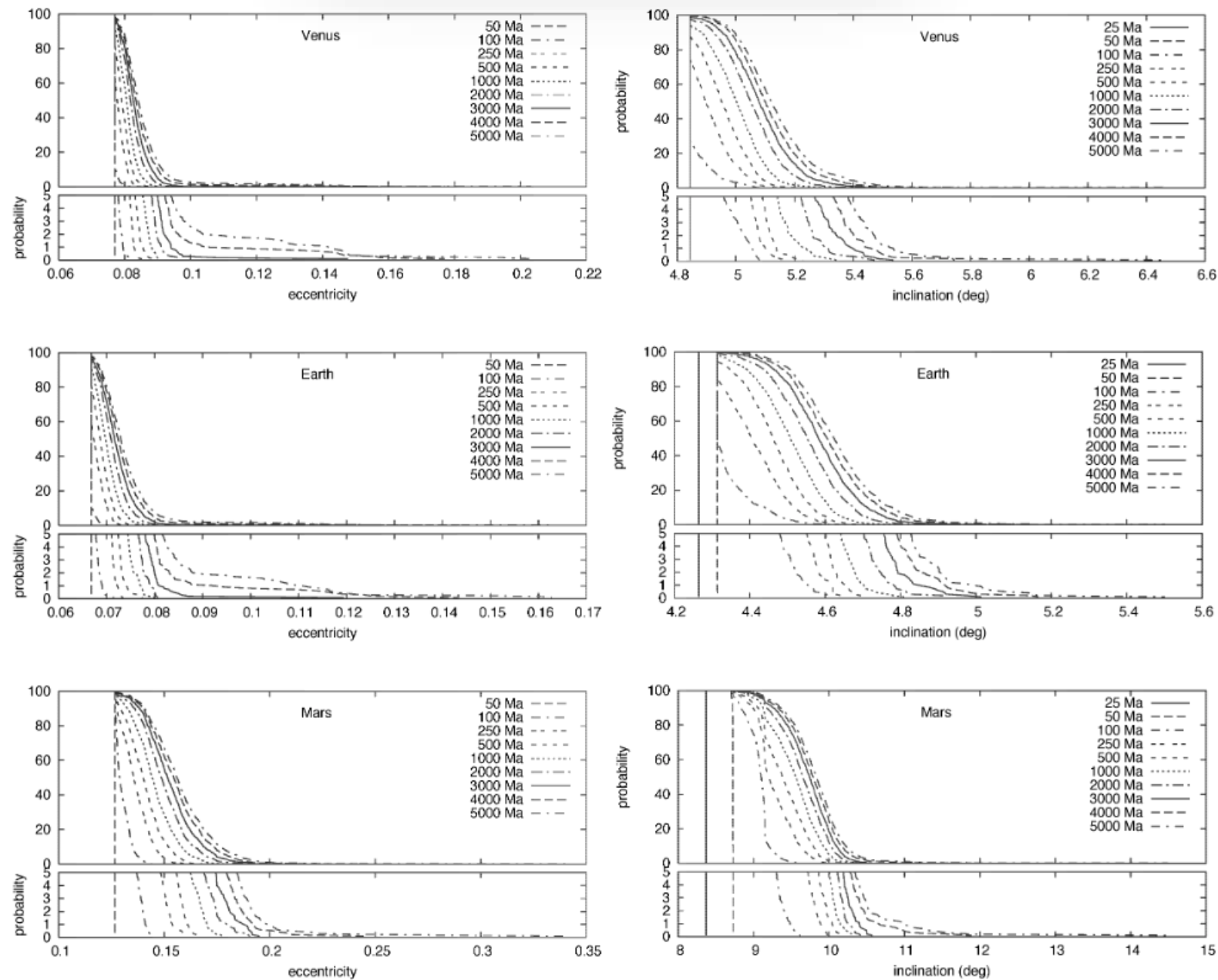


Abb. 3:
Laskar Diagramme
der inneren Planeten

Vorhersagen

Auf einen Blick

- Chaotische Effekte setzen nach ≈ 100 Myr ein
- Kollision Merkurs $\approx 1\%$ nach 5 Gyr
 - Ohne relativistische Effekte $\approx 60\%$
 - Hauptsächlich durch Merkur-Jupiter-Resonanz
- Kollision Mars-Erde sehr unwahrscheinlich
 - $< 0,2\%$

Weitere Zukunftsaussichten

Tod der Sonne – Sonne als Roter Riese

- Wasserstoffbrennende Phase endet in $\approx 5 (\pm 0,5)$ Gyr
 - Vollständige Verbrauch von Wasserstoffatomen
 - Maximum von vielen Simulationen
- Anstieg des Radius auf $\approx 0,75$ AE
 - Entspricht der Umlaufbahn von Venus
- Verliert über Sonnenwinde ca. 28% ihrer Masse

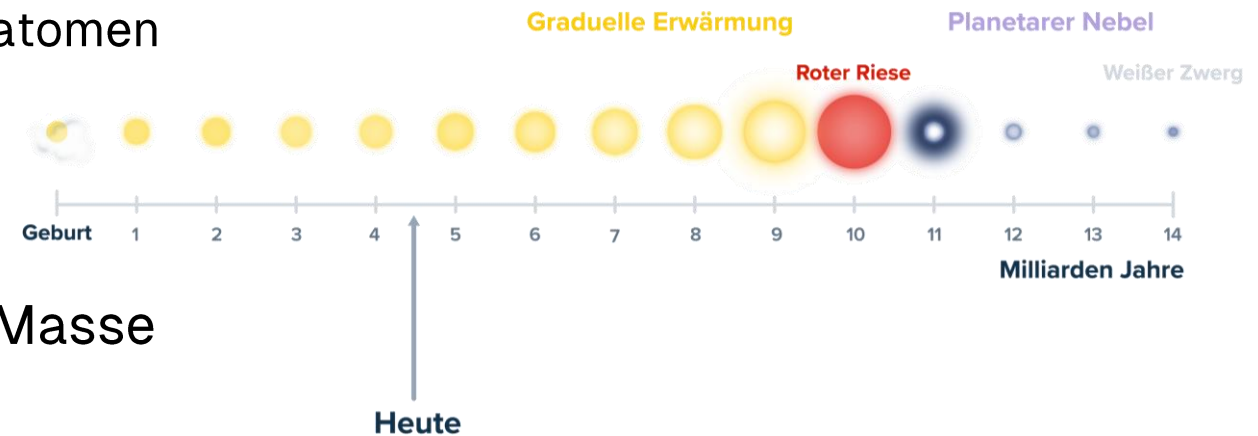


Abb. 5: Lebenszyklus der Sonne

Weitere Zukunftsaussichten

Asteroiden und Kometen

- Lyapunov-Zeiten von 100 – 100.000 Jahren
- Für die Erde gefährliche Objekte früh genug erkannt
- Kollisionen mit Planeten können Simulationen verfälschen
- Einschlaghäufigkeiten
 - " $d_{\text{Objekt}} \geq 1\text{km}$ " ca. alle 0,5 Myr
 - " $d_{\text{Objekt}} \approx 5\text{km}$ " ca. alle 20 Myr
 - " $d_{\text{Objekt}} \approx 10\text{km}$ " ca. alle 100 Myr

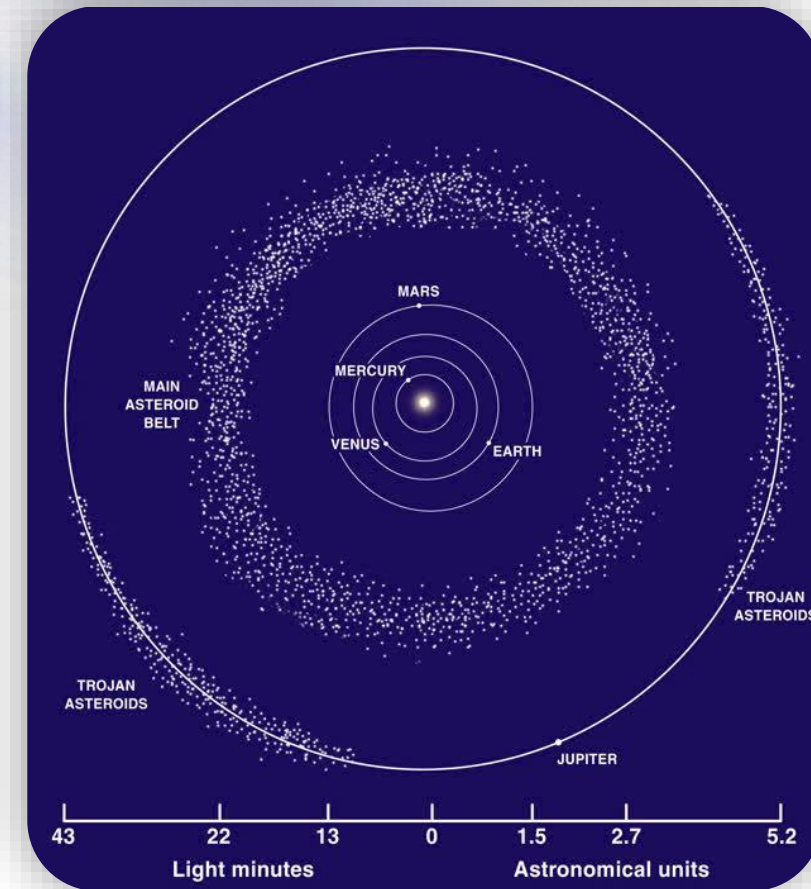


Abb. 2: Asteroidengürtel & Positionen

Fazit

Wo liegen die Grenzen der Vorhersagbarkeit?

Fazit

Nächste \approx 100 Millionen Jahre

- Quasiperiodische Planetenbahnen
 - Keine Kollisionen & Auswürfe
- Kleine Unsicherheiten von Asteroiden & Kometen
 - Allerdings $> 96\%$ aller NEAs bereits bekannt
 - Ablenkbar und Berechenbar
- Hohe Vorhersagbarkeit in naher Zukunft

100 Millionen – 5 Milliarden Jahre

- Chaotische Effekte
 - Nur noch statistische Aussagen möglich
 - 0,38mm Änderung
 - > völlig andere Bahnen nach ≈ 200 Myr
- Unvorhersagbare Sterneneinflüsse
- Lebenszyklus der Sonne sicher bekannt
- Vorhersagbarkeit nimmt mit der Zeit stark ab

Quellen

- <https://doi.org/10.48550/arXiv.2012.05177>
 - <https://doi.org/10.48550/arXiv.0802.3371>
 - <https://doi.org/10.48550/arXiv.1209.5996>
 - <https://doi.org/10.48550/arXiv.1705.00527>
 - <https://doi.org/10.48550/arXiv.1506.01084>
 - <https://zenodo.org/records/4299102>
 - <https://doi.org/10.1051/0004-6361/202140989>
-
- <https://orbital-mechanics.space/>
 - <https://rebound.readthedocs.io/en/latest/integrators/#whfast>
 - <https://eyes.nasa.gov/apps/solar-system/>
 - <https://science.nasa.gov/solar-system/>
 - <https://ssd.jpl.nasa.gov/horizons/app.html>

Quellen

- <https://en.wikipedia.org/wiki/Ergodicity>
- https://en.wikipedia.org/wiki/Two-body_problem
- https://en.wikipedia.org/wiki/Three-body_problem
- https://en.wikipedia.org/wiki/Liouville%E2%80%93Arnold_theorem
- https://en.wikipedia.org/wiki/Orbital_resonance
- https://en.wikipedia.org/wiki/Stability_of_the_Solar_System
- https://en.wikipedia.org/wiki/Lagrange_point
- https://en.wikipedia.org/wiki/Newton%27s_law_of_universal_gravitation
- https://en.wikipedia.org/wiki/Orbital_period
- https://en.wikipedia.org/wiki/Lyapunov_time
- https://en.wikipedia.org/wiki/Dormand%E2%80%93Prince_method
- https://en.wikipedia.org/wiki/Bessel_function
- https://en.wikipedia.org/wiki/Kepler%27s_equation
- <https://en.wikipedia.org/wiki/Ellipse>
- https://en.wikipedia.org/wiki/Orbit_equation
- https://en.wikipedia.org/wiki/Newton%27s_method
- https://en.wikipedia.org/wiki/Hamiltonian_mechanics
- https://en.wikipedia.org/wiki/Verlet_integration
- https://en.wikipedia.org/wiki/Numerical_integration
- https://en.wikipedia.org/wiki/Periodic_function

Quellen

- Abb. 1: https://commons.wikimedia.org/wiki/File:Kepler%27s_equation_scheme_German.svg
- Abb. 2: <https://www.spektrum.de/news/asteroidenguertel-um-sonne-gesteinsbrocken-zwischen-jupiter-und-mars/982787>
- Abb. 3 & 4: <https://doi.org/10.48550/arXiv.0802.3371>
- Abb. 5: <https://www.studysmarter.de/schule/physik/astronomie/sonne/>