Übungsblatt: Pulsar-Timing-Arrays

DSA Roßleben – Kurs 5.1

Gravitationswellen, welche von Paaren von supermassiven Schwarzen Löchern oder von Phasenübergängen bei MeV-Temperaturen im primordialen Plasma erzeugt werden, haben extrem lange Wellenlängen von der Größenordnung von Lichtjahren, Schwingungsfrequenzen auf der nHz-Skala und Periodendauern von Jahren bis Jahrzehnten. Um Gravitationswellen von solchen *astronomischen* Ausmaßen zu detektieren braucht es auch Messgeräte entsprechender Größe. In diesem Arbeitsblatt beschäftigen wir uns mit Pulsar-Timing-Arrays, welche genau diese Aufgabe erfüllen. Dazu nutzt man die gesamte *Galaxie* als Detektor und misst die kleinen Längenänderungen, die durch die Gravitationswellen verursacht werden, mithilfe von *Pulsaren*.

Pulsare als kosmische Uhren

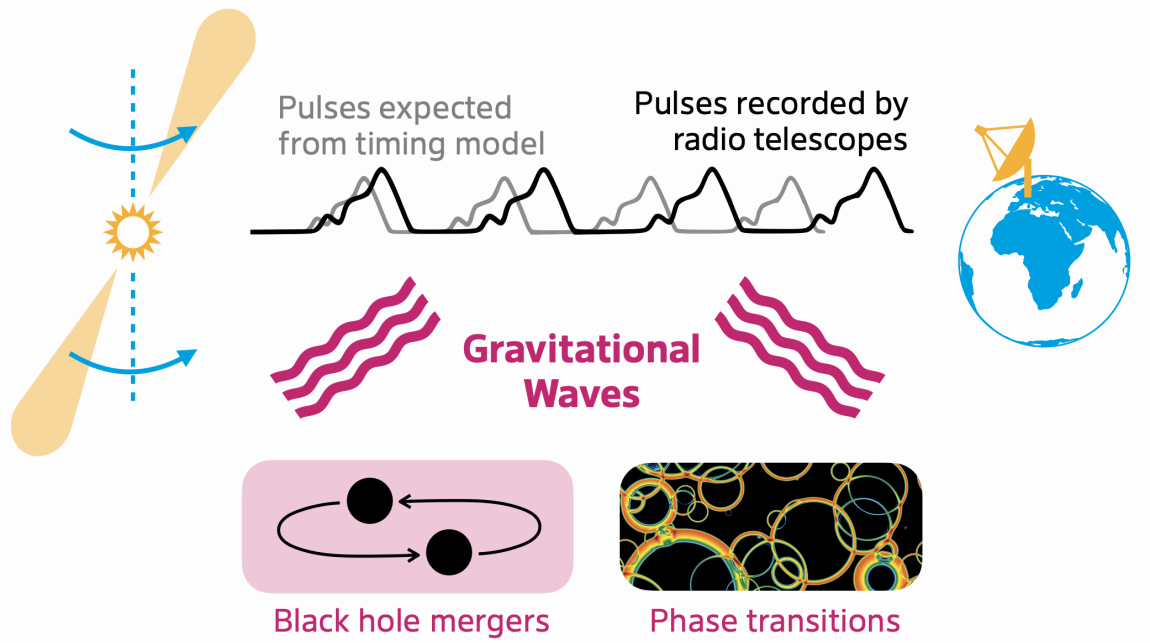

Abbildung 1: Schaubild aus einem Forschungshighlight der Universität Hamburg (Oktober 2024): „Der kosmische Brummen – woher kommt das Hintergrundrauschen aus Gravitationswellen?“. Das Signal, das NANOGrav und andere PTA-Kollaborationen messen, könnte von supermassiven Schwarzen-Loch-Binärsystemen oder von einem kosmologischen Phasenübergang im frühen Universum stammen. In Bringmann, Depta, Konstandin, Schmidt-Hoberg & Tasillo (*JCAP* 11 (2023) 053, arXiv:2306.09411) wurde untersucht, ob ein *Phasenübergang in einem dunklen Sektor* bei MeV-Temperaturen das Signal erklären kann und welche kosmologischen Nebenbedingungen dabei eingehalten werden müssen. Abbildung: C. Tasillo.

Ein Pulsar ist ein extrem schnell rotierender Neutronenstern, der wie ein kosmischer Leuchtturm einen Strahl elektromagnetischer Strahlung aussendet. Zeigt dieser Strahl bei jeder Rotation zur Erde, registrieren wir ein regelmäßiges „Ticken“, bei manchen sogenannten *Millisekundenpulsaren* mit einer Regelmäßigkeit, die mit Atomuhren konkurriert. Diese Pulsare sind also extrem präzise „kosmische Uhren“.

Läuft nun eine Gravitationswelle zwischen Erde und Pulsar durch die Raumzeit, verändert sie die effektive Distanz zwischen Erde und Pulsar geringfügig. Dadurch kommt das Signal des Pulsars minimal früher oder später an als erwartet. Diese kleinen Abweichungen zwischen beobachteter und erwarteter Ankunftszeit nennt man *Timing-Residuen*.

Warum ein *Array*?

Das Signal eines Gravitationswellenhintergrunds in den Timing-Residuen eines einzelnen Pulsars ist winzig. Folglich ist es sehr schwierig, es von anderen Störeffekten (intrinsisches „Rauschen“ des Pulsars, Messfehler, Störungen durch das interstellare Medium, ...) zu unterscheiden. Der Trick: Ein Gravitationswellenhintergrund beeinflusst *alle* vermessenen Pulsare am Himmel *gleichzeitig* und mit einer ganz bestimmten, vorhersagbaren (sog. *Hellings-Downs-*) *Korrelation* zwischen je zwei Pulsaren, die nur vom Winkelabstand dieser beiden Pulsare am Himmel abhängt.

Indem man die Timing-Residuen von vielen ($\mathcal{O}(10\text{–}100)$) über die ganze Galaxie verteilten Millisekundenpulsaren gemeinsam analysiert, ein sog. *Pulsar-Timing-Array* (PTA), kann man nach genau diesem charakteristischen, korrelierten Muster suchen, das von zufälligem Rauschen klar unterscheidbar ist. Effektiv wird die Milchstraße selbst zu einem riesigen Gravitationswellendetektor, dessen „Armlängen“ die Distanzen zwischen Erde und den einzelnen Pulsaren sind (typischerweise $\mathcal{O}(10^3)$ Lichtjahre), passend zur enormen Wellenlänge der gesuchten nHz-Gravitationswellen.

Das „Free Spectrum“: Messung des Gravitationswellenhintergrunds

Kollaborationen wie NANOGrav (*North American Nanohertz Observatory for Gravitational Waves*) beobachten seit Jahrzehnten dutzende Pulsare und werten die Korrelationen ihrer Timing-Residuen aus. Das Ergebnis wird oft als sogenanntes *free spectrum* dargestellt: Für jede von 14 Frequenzen $f_k = k/T_{\mathrm{obs}}$ ($k = 1, \ldots, 14$, mit $T_{\mathrm{obs}} = 16$ Jahren Beobachtungsdauer) wird ein Messwert für $\log_{10}\big(h^2\Omega_{\mathrm{gw}}(f_k)\big)$ mit einer Unsicherheit angegeben, wobei $\Omega_{\mathrm{gw}}(f)$ die Energiedichte der Gravitationswellen pro logarithmischem Frequenzintervall ist, normiert auf die kritische Energiedichte des Universums. Die Multiplikation mit $h^2 = 0.674^2 = 0.454$ (dem quadrierten Hubble-Parameter in Einheiten von $100\,\mathrm{km/s/Mpc}$) macht die Messgröße unabhängig vom exakten Wert der Hubble-Konstante und folglich auch von der sog. *Hubble-Tension*. Die Beschreibung als free spectrum erlaubt also eine direkte Beschreibung der Stärke $h^2\Omega_{\mathrm{gw}}(f_k)$ des kosmischen Gravitationswellenhintergrunds bei einer bestimmer Frequenz f_k.

Aufgabe 1

NANOGrav berichtet sein Ergebnis als “free spectrum”: für jede von 14 Frequenzen $f_k = k/T_{\mathrm{obs}}$ ($k = 1, \ldots, 14$, $T_{\mathrm{obs}} = 16$ Jahre die Beobachtungsdauer) gibt es einen Messwert für $\log_{10}\left(h^2\Omega_{\mathrm{gw}}(f_k)\right)$ mit einer Unsicherheit. Damit ihr nicht auf einen Download von echten (sehr großen) Datensätzen warten müsst, geben wir euch eine vereinfachte Version dieser Messung als Zahlenwerte vor; sie ist aus den echten NANOGrav-15yr-Daten abgeleitet.
Das “free spectrum” modelliert jeden Frequenz-Bin k als unabhängige Gauß-Messung:

$$\log_{10}\big(h^2\Omega_{\mathrm{gw}}(f_k)\big) \;\sim\; \mathcal{N}(\mu_k, \sigma_k^2)\,. \tag{1}$$

Dabei sind μ_k und σ_k der Erwartungswert und die Standardabweichung von $\log_{10}(h^2\Omega_{\mathrm{gw}})$ – also *nicht* $\log_{10}\mu_k$ oder $\log_{10}\sigma_k$! Im Code unten heißen diese Größen entsprechend `mu_log10` (“Mittelwert im log10-Raum”) und `sigma_log10` (“Streuung im log10-Raum”).

a) Lege eine Datei `mock_likelihood.py` an und kopiere folgenden Code hinein:

```
import numpy as np

T_obs = 16.0 * 365.25 * 24 * 3600.0     # 16 Jahre, in Sekunden
f0 = 1.0 / T_obs                        # niedrigste Fourier-Frequenz (Hz)
```

```
n_bins = 14
freqs = f0 * np.arange(1, n_bins + 1)  # 14 Frequenz-Bins (Hz)

nHz = 1e-9                             # 1 nHz in Hz, zur Umrechnung

# Mittelwert und Unsicherheit von log10(h^2 Omega_gw) in den 14
↪ Frequenz-Bins
mu_log10 = np.array([
    -10.1074, -8.9820, -8.9061, -8.9676, -10.0959, -10.3628, -10.6969,
     -8.1848, -10.1441, -9.1966, -9.2726, -9.6692, -9.6163, -9.3616,
])

sigma_log10 = np.array([
    0.3978, 0.1909, 0.2624, 0.5017, 1.7334, 1.9396, 1.3870, 1.2318,
    1.4141, 1.6662, 1.6218, 1.3132, 1.2425, 1.2703,
])
```

b) Plotte diese „Mock-Daten“: `freqs` auf der x-Achse (logarithmisch, `plt.xscale("log")`), `mu_log10` auf der y-Achse mit Fehlerbalken `yerr=sigma_log10` (benutze `plt.errorbar`).

Hinweis: `freqs` ist in Hz gegeben, das sind sehr kleine, unhandliche Zahlen. Überlegt euch, in welcher Einheit man Frequenzen im PTA-Band sinnvoll angibt (Größenordnung der Frequenzen? Es gibt schon eine passende Konstante in eurem Code…) und rechnet `freqs` entsprechend um, bevor ihr sie plottet.

c) *Diskussion:* Bei welchen Frequenzen ist das Frequenzspektrum am genauesten bestimmt?

Musterlösung 1

a) Die Datei `mock_likelihood.py` mit den Daten oben anlegen; diese Datei wird uns ab jetzt durch den Rest dieses Arbeitsblatts begleiten.
b) Da `freqs` Werte um $10^{-9}\,\mathrm{Hz} = 1\,\mathrm{nHz}$ hat, bietet es sich an, in Einheiten von `nHz` zu plotten:

```
import matplotlib.pyplot as plt

plt.errorbar(freqs / nHz, mu_log10, yerr=sigma_log10, fmt="o")
plt.xscale("log")
plt.xlabel(r"$f$ [nHz]")
plt.ylabel(r"$\log_{10}(h^2 \Omega_{\rm gw})$")
plt.show()
```

c) Die kleinsten Unsicherheiten ($\sigma_{\log 10} \approx 0.19$–$0.26$) findet man bei den Frequenz-Bins 2 und 3 (f_2, f_3); hier ist die Messung also am genauesten. Bin 8 sticht heraus: hier ist $\mu \approx -8.18$ deutlich größer (also der Hintergrund stärker) als in den Nachbar-Bins, allerdings mit relativ großer Unsicherheit ($\sigma \approx 1.23$). Insgesamt erkennt man bereits in den Rohdaten (Abbildung 2) eine „Beule“ im Spektrum bei niedrigen Frequenzen – genau diese Beule ließe sich mit einem physikalischen Modell (z. B. einem Phasenübergang im frühen Universum) erklären.

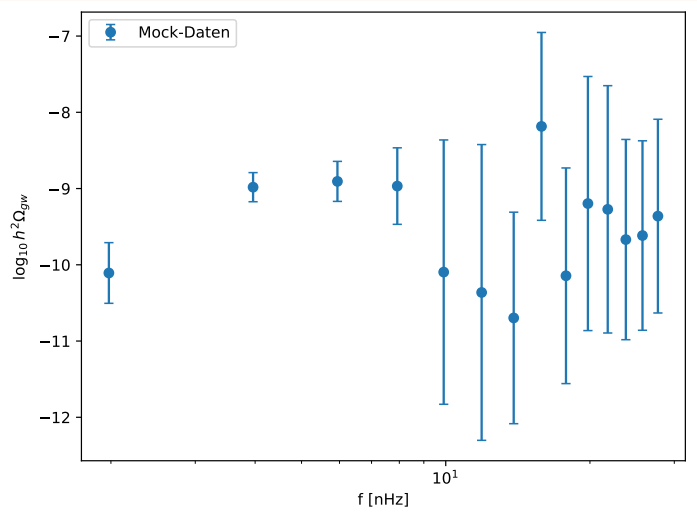

Abbildung 2: Die Mock-Daten: $\log_{10}(h^2\Omega_{\mathrm{gw}})$ als Funktion der Frequenz f (in nHz) mit Fehlerbalken $\sigma_{\log 10}$. Bei niedrigen Frequenzen erkennt man eine „Beule“ im Spektrum.

Wir haben nun die Daten einer echten Gravitationswellenmessung der NANOGrav-Kollaboration (2023) in der Hand. Die spannende Frage ist: *Welches physikalische Modell passt am besten zu diesen 14 Datenpunkten, und mit welcher Unsicherheit lassen sich seine Parameter bestimmen?* Um diese Frage zu beantworten, benutzen wir die gleichen Methoden, die wir bereits am Schuhpendel-Beispiel kennengelernt haben: Wir definieren ein Modell, eine Likelihood, einen Prior und sampeln dann die Posterior mit dem Metropolis-Hastings-Algorithmus. Dann können wir uns Gedanken darüber machen, ob diese Modellparameter mit anderen kosmologischen Beobachtungen konsistent sind, und ob das Modell überhaupt physikalisch plausibel ist.

Alles zusammen: ein erstes Spektrum fitten

Jetzt bringen wir alles zusammen: den Metropolis-Hastings-Sampler aus dem Kapitel „Monte-Carlo-Methoden“ (Datei `mh_sampler.py`) und die Likelihood basierend auf den Daten aus Aufgabe 1. Als Modell benutzen wir zunächst ein einfaches „Toy model“ mit der *Form* eines Peaks:

$$h^2\Omega_{\mathrm{gw}}(f) = A \cdot \frac{x}{1+x^2}\,, \qquad x = \frac{f}{f_{\mathrm{peak}}}\,. \tag{2}$$

Man kann leicht zeigen, dass $x/(1+x^2)$ sein Maximum bei $x=1$, also bei $f = f_{\mathrm{peak}}$, hat; f_{peak} ist also die Peak-Frequenz, und A legt die Höhe des Spektrums am Peak fest. Logarithmiert man Gleichung (2), wird aus dem Produkt eine Summe:

$$\log_{10}\!\left(h^2\Omega_{\mathrm{gw}}(f)\right) = \log_{10} A + \log_{10}\!\left(\frac{x}{1+x^2}\right). \tag{3}$$

Genau wie f_{peak} über viele Größenordnungen variieren kann, sampeln wir also $\log_{10} A$ und $\log_{10} f_{\mathrm{peak}}$ statt A und f_{peak} selbst.

Aufgabe 2

a) Schreibe eine Funktion

```
def broken_powerlaw_log10(f_hz, log10A, f_pivot_hz):
```

```
    x = f_hz / f_pivot_hz
    return log10A + np.log10(x / (1.0 + x**2))
```

b) Definiere einen Gleichverteilungs-Prior für $\theta = (\log_{10} A, \log_{10} f_{\text{peak}})$:

```
LOG_A_MIN, LOG_A_MAX = -13.0, -4.0
LOG_F_PEAK_MIN, LOG_F_PEAK_MAX = -10.0, -7.0   # f_peak in [0.1, 100] nHz

def log_prior(theta):
    log10A, log_f_peak = theta
    if (LOG_A_MIN <= log10A <= LOG_A_MAX) and (LOG_F_PEAK_MIN <=
    ↪  log_f_peak <= LOG_F_PEAK_MAX):
        return 0.0
    return -np.inf
```

und schreibe, analog zum Aufbau von `log_posterior` aus dem Kapitel „Monte-Carlo-Methoden“, eine Funktion `log_posterior(theta)`, die (i) `log_prior(theta)` auswertet und sofort `-np.inf` zurückgibt, falls dieser nicht endlich ist, (ii) sonst mit `broken_powerlaw_log10` das Modell an den 14 Frequenzen `freqs` berechnet, und (iii) `log_prior + log_likelihood(model)` zurückgibt.

c) Rufe den Sampler auf:

```
chain, logp, acc = metropolis_hastings(
log_posterior, x0=[-8.0, -9.0], step_sizes=[0.45, 0.22],
n_steps=50000, seed=42)
samples = chain[5000:]   # burn-in
```

Gib die Akzeptanzrate aus, erstelle einen corner plot von `samples` und bestimme den best-fit-Punkt sowie Median und 16%/84%-Perzentile für A und $\log_{10} f_{\text{pivot}}$. Diese befinden sich im Corner-Plot oberhalb der Panels auf der Diagonalen.

d) Plotte die Mock-Daten (wie in Aufgabe 1) zusammen mit der Modellkurve für den besten Parametersatz:

```
f_plot = np.logspace(np.log10(freqs[0]) - 1, np.log10(freqs[-1]) + 1, 500)
```

Welche Peak-Frequenz f_{peak} bevorzugen die Daten (in nHz)? Wie gut ist sie eingegrenzt?

Musterlösung 2

a)/b)

```
def broken_powerlaw_log10(f_hz, log10A, f_pivot_hz):
    x = f_hz / f_pivot_hz
    return log10A + np.log10(x / (1.0 + x**2))

LOG_A_MIN, LOG_A_MAX = -13.0, -4.0
LOG_F_PEAK_MIN, LOG_F_PEAK_MAX = -10.0, -7.0

def log_prior(theta):
    log10A, log_f_peak = theta
    if (LOG_A_MIN <= log10A <= LOG_A_MAX) and (LOG_F_PEAK_MIN <= log_f_peak <=
    ↪  LOG_F_PEAK_MAX):
        return 0.0
```

```
    return -np.inf

def log_posterior(theta):
    lp = log_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    log10A, log_f_peak = theta
    model = broken_powerlaw_log10(freqs, log10A, 10**log_f_peak)
    return lp + log_likelihood(model)
```

c) Der Aufruf von `metropolis_hastings` mit den angegebenen Einstellungen liefert eine Akzeptanzrate von acc ≈ 0.3, wieder im gewünschten Bereich 0.2–0.5. Der corner plot (Abbildung 3) zeigt für $\log_{10} A$ und $\log_{10} f_{\text{peak}}$ jeweils eine annähernd glockenförmige marginalisierte Verteilung; im 2D-Panel erkennt man eine leichte Korrelation zwischen $\log_{10} A$ und $\log_{10} f_{\text{peak}}$ (eine höhere Amplitude A „passt“ zu einer leicht anderen Peak-Frequenz, ohne die Daten schlechter zu erklären – eine *Entartung*, wie sie bei Fits mit mehreren Parametern häufig auftritt). Best-fit-Punkt und die 16%/84%-Perzentile lassen sich direkt aus dem Corner-Plot ablesen.

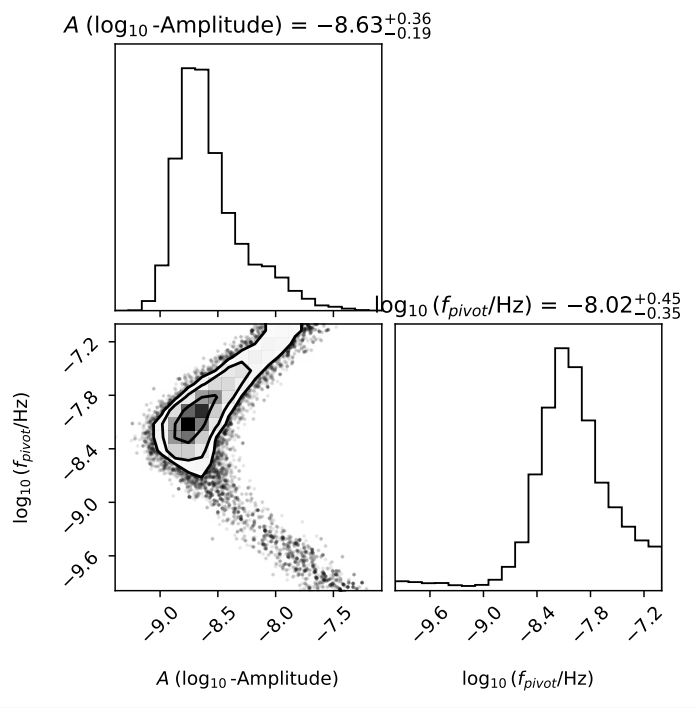

Abbildung 3: Corner plot der Samples für $\log_{10} A$ und $\log_{10} f_{\text{peak}}$. Beide Parameter sind gut bestimmt (annähernd glockenförmige marginalisierte Verteilungen); im 2D-Panel ist eine leichte Korrelation (Entartung) zwischen den beiden Parametern zu erkennen.

d) Die beste Modellkurve folgt der „Beule“ in den Mock-Daten (vgl. Aufgabe 1c)) und hat ihren Peak bei einer Frequenz f_{peak} von einigen nHz – also genau im Bereich, in dem die Mock-Daten die größten Werte von $\log_{10} h^2 \Omega_{\text{gw}}$ zeigen. Das 16%/84%-Intervall für $\log_{10} f_{\text{peak}}$ erstreckt sich dabei über etwa eine Größenordnung – die Daten legen die Peak-Frequenz also schon recht gut fest, aber nicht beliebig genau.

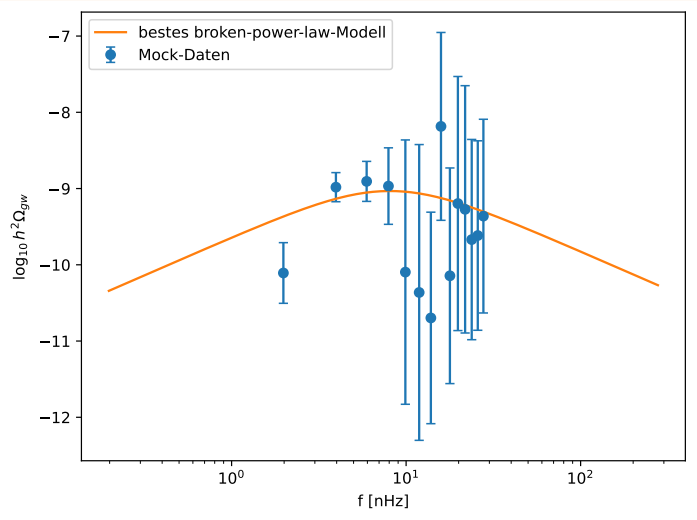

Abbildung 4: Die PTA-Daten (Punkte mit Fehlerbalken) zusammen mit der besten Modellkurve $h^2\Omega_{\mathrm{gw}}(f)$ für den best-fit-Parametersatz. Die Modellkurve folgt der „Beule“ bei niedrigen Frequenzen.

Damit haben wir zum ersten Mal erfolgreich ein Modell an einen (simulierten) Gravitationswellenhintergrund gefittet (Abbildung 4)! Ein naheliegender nächster Schritt wäre, dieses noch unmotivierte „Toy model“ schrittweise durch *physikalische* Modelle zu ersetzen – etwa das Spektrum eines Phasenübergangs im frühen Universum mit den Parametern $\alpha, \beta/H, T_*$. Der entscheidende Punkt dabei: Der Sampler `metropolis_hastings` selbst bliebe dabei *komplett unverändert*; nur das Modell in `log_posterior` wird ausgetauscht. Das ist die ganze Stärke eines allgemeinen MCMC-Samplers: Einmal geschrieben, lässt er sich auf praktisch *jedes* Modell anwenden, das sich als `log_posterior(theta)` schreiben lässt.

Extra-Aufgabe: Mehr Parameter, derselbe Sampler

Das Toy model aus Gleichung (2) aus Aufgabe 2 hat eine feste Form: links von f_{peak} steigt $h^2\Omega_{\mathrm{gw}}(f) \propto x$, rechts davon fällt es $\propto x^{-1}$ (mit $x = f/f_{\mathrm{peak}}$).

Aufgabe 3

Verallgemeinere das Toy model zu

$$\log_{10}\big(h^2\Omega_{\mathrm{gw}}(f)\big) = \log_{10} A + \log_{10}\left(\frac{x^a}{1+x^c}\right), \qquad x = \frac{f}{f_{\mathrm{peak}}}, \tag{4}$$

mit zwei zusätzlichen freien Parametern a und c (für $a = 1, c = 2$ ist das genau das Modell aus Gleichung (2) aus Aufgabe 2).

a) Was bedeuten a und $c - a$ für die Steigung des Spektrums links bzw. rechts von f_{peak} (jeweils auf einer doppelt-logarithmischen Achse)?

b) Erweitere `log_prior` und `log_posterior` so, dass $\theta = (\log_{10} A, \log_{10} f_{\mathrm{peak}}, a, c)$ jetzt *vier* Einträge hat. Wähle sinnvolle (gleichverteilte) Prior-Bereiche für a und c, z. B. $a \in [0.5, 5]$, $c \in [1, 8]$.

c) Muss sich an `metropolis_hastings` selbst irgendetwas ändern, damit es mit 4 statt 2 Parametern funktioniert? Probiere es aus; du brauchst nur `x0`

und `step_sizes` auf vier Einträge zu erweitern, z. B. `x0=[-8.0,-9.0,1.0,2.0]`, `step_sizes=[0.2,0.2,0.2,0.35]`, `n_steps=100000`. Was sagt dir das über die Allgemeinheit eures MCMC-Algorithmus?

d) Erstelle wieder einen corner plot und einen Daten-Plot. Welche Werte für a und c bevorzugen die Daten? Vergleiche mit $a = 1, c = 2$ aus Aufgabe 2; ist das einfachere Toy model eine gute Näherung?

Musterlösung 3

a) Für $x \ll 1$ (also $f \ll f_{\text{peak}}$) gilt $x^c \ll 1$, sodass $\frac{x^a}{1+x^c} \approx x^a$ und damit $\log_{10}(h^2\Omega_{\text{gw}}) \approx \log_{10} A + a \log_{10} x$; a ist also die Steigung des Spektrums *links* von f_{peak} in einer doppeltlogarithmischen Darstellung. Für $x \gg 1$ gilt $\frac{x^a}{1+x^c} \approx x^{a-c}$, also $\log_{10}(h^2\Omega_{\text{gw}}) \approx \log_{10} A + (a - c) \log_{10} x$ – die Steigung *rechts* von f_{peak} ist $a - c$, also um $c - a$ *flacher* (steiler abfallend) als links. Für $a = 1, c = 2$ ergibt das genau die Steigungen $+1$ und -1 des ursprünglichen Toy models (2).
b) Wir erweitern Prior, Modellfunktion und Posterior um die zwei zusätzlichen Parameter:

```
A_MIN, A_MAX = -13.0, -4.0
LOG_F_PEAK_MIN, LOG_F_PEAK_MAX = -10.0, -7.0
A_EXP_MIN, A_EXP_MAX = 0.5, 5.0
C_EXP_MIN, C_EXP_MAX = 1.0, 8.0

def log_prior(theta):
    A, log_f_peak, a, c = theta
    if not (A_MIN <= A <= A_MAX):
        return -np.inf
    if not (LOG_F_PEAK_MIN <= log_f_peak <= LOG_F_PEAK_MAX):
        return -np.inf
    if not (A_EXP_MIN <= a <= A_EXP_MAX):
        return -np.inf
    if not (C_EXP_MIN <= c <= C_EXP_MAX):
        return -np.inf
    return 0.0

def broken_powerlaw_general_log10(f_hz, A, f_peak_hz, a, c):
    x = f_hz / f_peak_hz
    return A + np.log10(x**a / (1.0 + x**c))

def log_posterior(theta):
    lp = log_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    A, log_f_peak, a, c = theta
    model = broken_powerlaw_general_log10(freqs, A, 10**log_f_peak, a, c)
    return lp + log_likelihood(model)
```

c) An `metropolis_hastings` selbst muss *nichts* geändert werden; `ndim` wird automatisch aus der Länge von `x0` bestimmt, und `rng.normal(0.0, step_sizes, size=ndim)` funktioniert für beliebig viele Dimensionen. Der Algorithmus kennt `log_posterior` nur als „Black Box“, die für ein θ eine einzelne Zahl zurückgibt; wie viele Einträge θ hat, spielt für den Sampler keine Rolle.
d) Der corner plot (Abbildung 5) zeigt, dass a und c von den Daten tatsächlich eingegrenzt werden, allerdings nicht sonderlich präzise: Für nur 14 Datenpunkte sind 4 freie Parameter

schon eine ganze Menge. Die bevorzugten Werte liegen typischerweise nahe bei $a \approx 3$ und $c \approx 4.5$, also weit entfernt vom einfachen Toy model aus Aufgabe 2: Die zusätzliche Freiheit in a und c wird von den Daten nicht zwingend benötigt, das einfachere Modell ist also bereits eine gute Näherung (Abbildung 6).

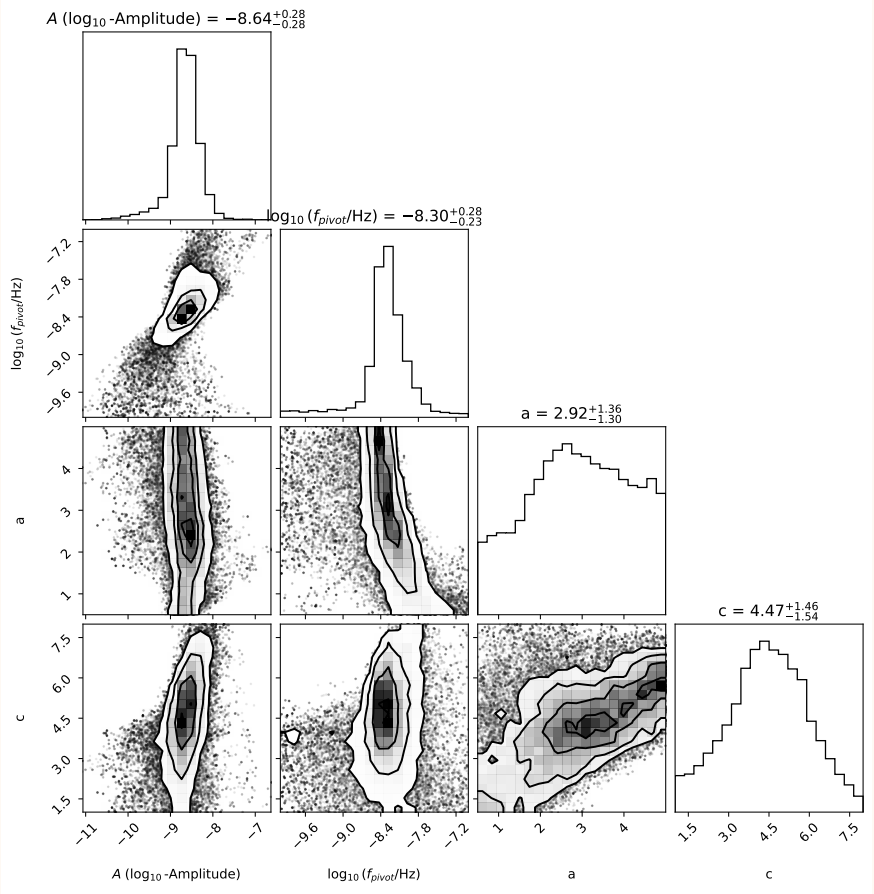

Abbildung 5: Corner plot für das verallgemeinerte Modell aus Gleichung (4) mit den vier Parametern $\log_{10} A$, $\log_{10} f_{\mathrm{peak}}$, a und c.

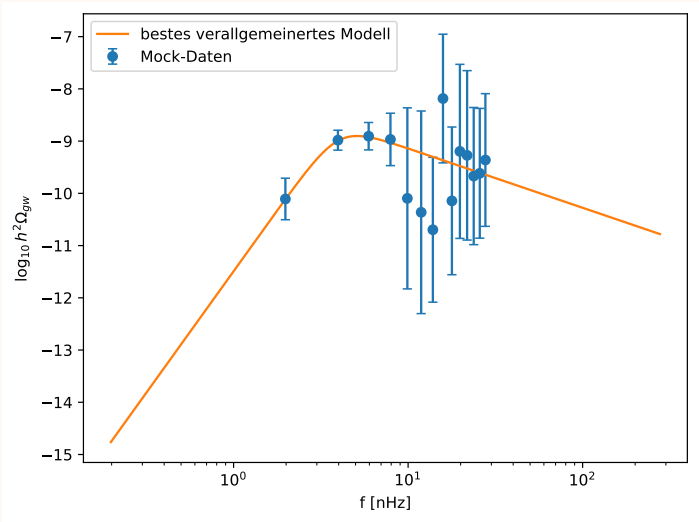

Abbildung 6: Mock-Daten und bestes verallgemeinertes Modell aus Gleichung (4).