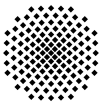

Universität Stuttgart

Institut für Angewandte Analysis und Numerische Simulation

Metrik für sEMG-Daten

metrik_waves.py

Lisa Dollmann, Maximilian Hack, Lilian Cathérine Lepère

Autor Lisa Dollmann
Maximilian Hack
Lilian Cathérine Lepère

Betreuer Aaron Krämer

Datum Stuttgart
27. Dezember 2021

Das ist die Dokumentation der Metrik *metrik_waves.py*. Die Metrik misst wie viele Wellen von welcher Geschwindigkeit in einem Datensatz von *data_creator_*00*x.py* vorkommen. Während im *data_creator_*001*.py* langsame Wellen seltener vorkommen, zeigt die Metrik, dass im *data_creator_*002*.py* die Anzahl an Wellen nur noch tendenziell steigt. Es kommt vor, dass es von höheren Geschwindigkeiten weniger Wellen gibt. In der Dokumentation werden Installation und Nutzung beschrieben, die enthaltenen Features und weitere Ideen aufgelistet, die Ergebnisse kurz interpretiert, relevante Programmausgaben dargestellt und erläutert.

Inhaltsverzeichnis

1 Installation und Nutzung

1.1 Module und Elemente

Das Programm benötigt die bereits an anderen Stellen des Projekts benötigten Module *pickle*, *numpy*, *matplotlib.pyplot* und *sys*.

1.2 Parameter

Es gibt die drei Eingabeparameter x des entsprechenden $data\_creator\_00x.py$, die n_speeds und $n\_waveLengths$. Deren Nutzung wird im folgenden Abschnitt näher erklärt.

1.3 Nutzung

Beispielbilder zur Veranschaulichung der Nutzung sind in Kapitel 3 eingebunden.
Um die Metrik *metrik_speeds.py* zu nutzen hat man zwei Optionen. Man muss entweder den Dateinamen eines Datensatzes übergeben oder dessen 'Kennzahlen'. Will man beispielsweise die Datei $DATA/creator\_001/RawData\_3\_velocities\_\_\_\_15\_waveLengths.pickle$ mit der Metrik analysieren, übergibt man dem Terminal entweder diesen Namen, also *python metrik_speeds.py* $DATA/creator\_001/RawData\_3\_velocities\_\_\_\_15\_waveLengths.pickle$ oder *python metrik_speeds.py* 1 3 15. Letzteres ist eine abkürzende Schreibweise,die im folgenden kurz erläutert wird:
Erzeugt man mit $data\_creator\_00x.py$ einen Datensatz, so ist der Dateiname immer nach dem selben Muster aufgebaut. Die Dateinamen sind vom Typ $DATA/creator\_001/RawData\_ + str(n\_speeds) + \_velocities\_\_\_\_ + str(n\_waveLengths) + \_waveLengths.pickle$ (bei $data\_creator\_002.py$ kommt im Dateinamen noch der $grow\_factor$ hinzu; die Metrik nutzt aber immer den Default Wert). Um das Aufrufen eines Datensatzes aus einer *pickle*-Datei zu Erleichtern kann man also auch die 'Kennzahlen' eingeben. Das sind die Zahl x des entsprechenden $data\_creator\_00x.py$, die n_speeds und $n\_waveLengths$. Ein Datensatz mit diesen 'Kennzahlen' muss bereits exisiteren!
Hat man das Programm ausgeführt zeigt es ein Histogramm, das die Anzahl an Wellen pro Geschwindigkeit darstellt.

2 Enthaltene Features und deren Ausgaben

2.1 Dateiname aus 'Kennzahlen'

Wie in 1.3 bereits angesprochen, werden Dateien entweder über deren Name oder über sogenannte Kennzahlen aufgerufen. Das funktioniert mithilfe des Moduls *sys*. Das Meiste zu diesem Vorgehen ist in 1.3 bereits erklärt. An dieser Stelle soll noch auf zwei Punkte eingegangen werden. Das Programm soll anhand der eingegebenen *sys*-Argumente herausfinden, ob ein Dateiname oder nur die Kennzahlen der Datei übergeben wurden. Dazu wird mittels einer if-Abfrage überprüft wie lange der *string* des ersten *sys*-Arguments

ist. Da im Moment mehr als 1000 *data_creators* sehr nicht absehbar sind, sollte ein *string*, der mehr als 4 Zeichen beinhaltet ein Dateiname sein. Der zweite Punkt bezieht sich auf die Methode, die aus den Kennzahlen den Dateinamen zusammenfügt. Der Dateiname für Daten aus dem *data_creator_*002*.py* enthält den Zusatz $grow\_factor\_\_4$. Dies muss also im Fall x=2 geprüft und angepasst werden. Das wurde mithilfe einer if-Abfrage umgesetzt.
Daraus ergeben sich zwei @ToDos:

2.1.1 @ToDo: $str(sys.argv[4])$

Man kann ein viertes Argument $str(sys.argv[4])$ übergeben um den $grow\_factor$ ebenfalls variabel zu halten. Eventuell kann man diesem *sys*-Argument schlicht den Default-Wert 4 zuordnen, damit lässt man dem Nutzer die Entscheidung.

2.1.2 @ToDo: if-Abfrage

Eventuell kann man die erste if-Abfrage, ob ein Dateiname oder nur dessen Kennzahlen übergeben wurden, kompakter fassen.
Man könnte beispielsweise nur den Dateinamen als Ganzes übergeben und die Option mit den Kennzahlen streichen. Wenn es mehr als die bisherigen beiden *data_creator_*00x*.py* gibt, können die Kennzahlen aufgrund möglicher Zusätze im Dateinamen, wie das beim $grow\_factor$ in *data_creator_*002*.py* der Fall ist, kompliziert werden. Man müsste entweder bei all diesen Zusätzen auf einen Default-Wert hoffen, oder 2.1.1 für all diese Zusätze umsetzen. Das ist entweder sehr steif oder ab spätestens 7 *sys*-Argumenten sehr verwirrend und Tippfehler-anfällig.
Bisher erleichtert die Kennzahlen-Schreibweise aber die Eingabe und bleibt vorerst erhalten.

2.2 Pickle

Nachdem der Dateiname aus den *sys*-Argumenten erstellt wurde, wird die Datei mithilfe des *pickle*-Moduls entpickelt. Das läuft standardmäßig ab.

2.3 Dimension der Daten

Je nachdem welcher *data_creator_*00x*.py* zum Erstellen eines Datensatzes genutzt wurde, sind die Daten aus der geladenen *.pickle*-Datei einfach (für x=1) oder mehrfach (für x=2) verschachtelt. Da die vorliegende Metrik nur die absolute Häufigkeit einer Geschwindigkeit in einem Datensatz misst, ist der Grad der Verschachtelung irrelevant. Man kann in dieser Metrik also alle Daten in das selbe Format umwandeln, um die Daten aus jedem *data_creator_*00x*.py* verarbeiten zu können.
Dafür werden die Daten zunächst in $numpy - arrays$ umgewandelt. Eine if-Abfrage prüft dann über den Befehl *.ndim*, ob die Daten eindimensional sind. Falls nicht, werden sie über den Befehl $.flatten()$ in ein eindimensionales $array$ umgewandelt.

2.4 Liste mit Anzahl Wellen

Um die Anzahl an Wellen später in einem Histogramm darzustellen, werden sie zunächst in einer Liste namens *waves* gesammelt. Dafür werden sie in einer for-Schleife aus den Metadaten extrahiert. Dabei werden für jedes Szenario zunächst die *keys*, also die Geschwindigkeiten, mittels $d.meta\_.iData\_[:][j].keys()$ extrahiert. Danach werden die zugehörigen *values*, also wie oft die jeweilige Geschwindigkeit auftritt, via $d.meta\_.iData\_[:][j].get(i)$ bestimmt. Entsprechend oft wir dann die Geschwindigkeit der Liste *waves* hinzugefügt. Diese Methode wurde gewählt um damit python ein Histogramm erstellen kann.

2.5 Das Histogramm

Das Histogramm bildet dann die Anzahl an Wellen im Datensatz jeder vorkommenden Geschwindigkeit ab. Da die genaue Anzahl auf dem Bild schwer abzulesen ist, wird die Anzahl an Wellen der jeweiligen Geschwindigkeit zusätzlich in Zahlenform in der Konsole ausgegeben.
Das Histogramm und die dazugehörige Ausgabe im Terminal lassen Verbesserungsmöglichkeiten offen.

2.5.1 @ToDo: Anzahl *bins*

Erzeugt man ein Histogramm, muss man die Anzahl und die Schrittweite zwischen den *bins* im Voraus festlegen. *Bins* sind die Balken im Histogramm. Legt man beispielsweise fest, dass es 5 *bins* mit der Schrittweite 2.5 geben soll, so erhält man die *bins* 0, 2.5, 5, 7.5 und 10. Tritt dann die Geschwindigkeit 3 auf, wird diese schlichtweg der 2.5 zugeordnet. Um ein genaues Bild von der Häufigkeit der Geschwindigkeiten zu bekommen, müsste man also die Anzahl und die Schrittweite an *bins* immer passend zum Datensatz wählen. Dafür nutzt man die Tatsache, dass die Geschwindigkeit maximal 10 wird. Außerdem legt man in jedem Datensatz *n_speeds*, also die Anzahl an verschiedenen Geschwindigkeiten in diesem Datensatz, fest. Zusammen ergibt sich dann $\frac{1}{n\_speeds-1}$ als optimale Schrittweite und $(n\_speeds - 1) \cdot 10 + 2$ als optimale Anzahl an *bins*. Dafür könnte man *n_speeds* aus dem Dateinamen des Datensatzes auslesen. *n_speeds* steht an der Stelle 25 und, falls *n_speeds* echt größer ist als 9, an den Stellen 25 und 26 im Dateiname.

3 Interpretation

3.1 *data_creator_*001

Für Daten aus dem *data_creator_*001 veranschaulicht die Metrik, dass schnellere Wellen häufiger vorkommen. Für fast alle Wellen gilt: je schneller die Wellen sind umso mehr gibt es davon.

3.2 *data_creator_*002

Im *data_creator_*002 gilt die Tendenz aus 3.1 zwar noch, die Ausnahmen häufen sich aber deutlich. Die Erklärung könnte die *metrik_speeds* liefern: hohe Geschwindigkeiten treten in deutlich weniger Szenarien auf, als niedrige.

4 Relevante Programmausgaben

In diesem Abschnitt werden im Text referenzierte Programmausgaben veranschaulicht.

4.1 *data_creator_*001

Die folgenden Bilder zeigen das Histogramm und die Ausgabe im Terminal bei Daten, die von *data_creator_*001 erzeugt wurden.

4.1.1 Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen

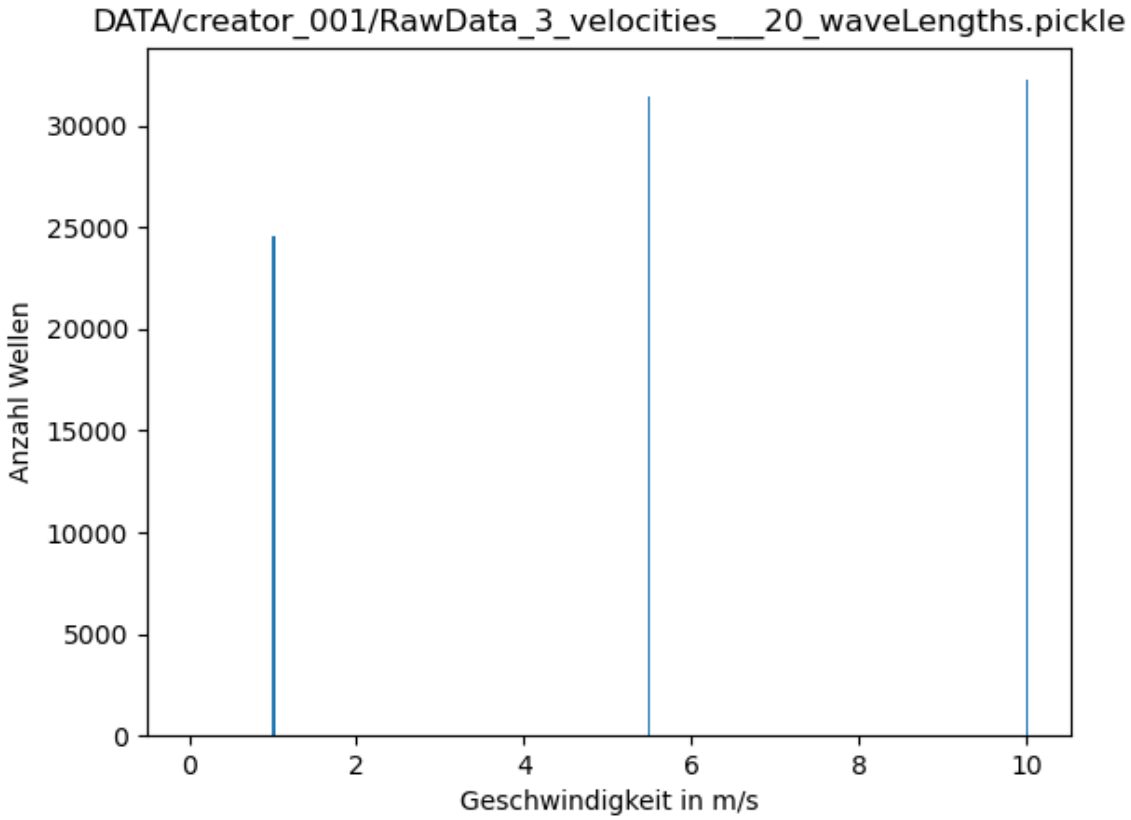

Abbildung 1: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_*001 erzeugt wurde. Der Datensatz enthält drei verschiedene Geschwindigkeiten zu jeweils 20 Wellenlängen. Man erkennt, dass es deutlich weniger Wellen der Geschwindigkeit 1 gibt. Zwischen 5 und 10 ist der Anwuchs nicht mehr so deutlich.

4.1.2 Histogramm mit 30 Geschwindigkeiten und 30 Wellenlängen

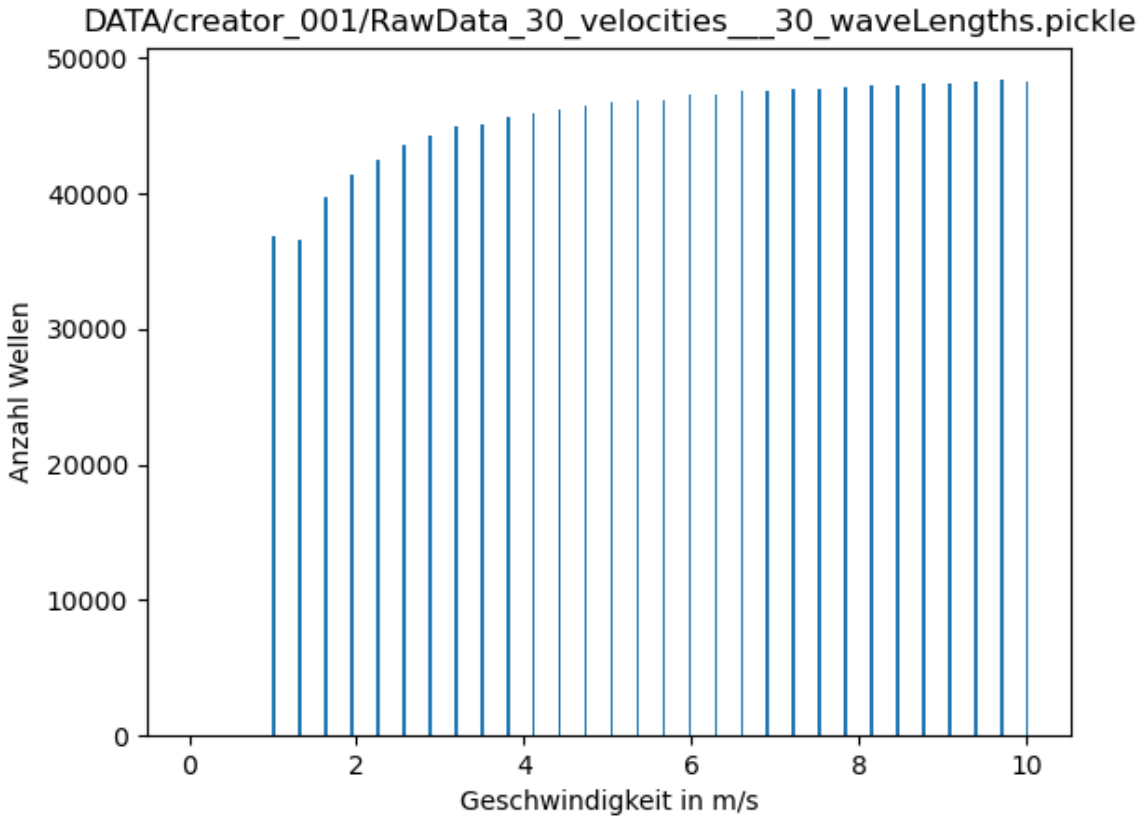

Abbildung 2: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_*001 erzeugt wurde. Der Datensatz enthält 30 verschiedene Geschwindigkeiten zu jeweils 30 Wellenlängen. Auch hier sieht man den Zuwachs. Man erkennt aber auch deutlich die Ausnahme bei ca 1,31 m/s.

4.2 *data_creator_*002

Dieser Teil zeigt von der *metrik_speeds.py* erzeugt Bilder aus Daten, die mit dem *data_creator_*002 erzeugt wurden.

4.2.1 Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen

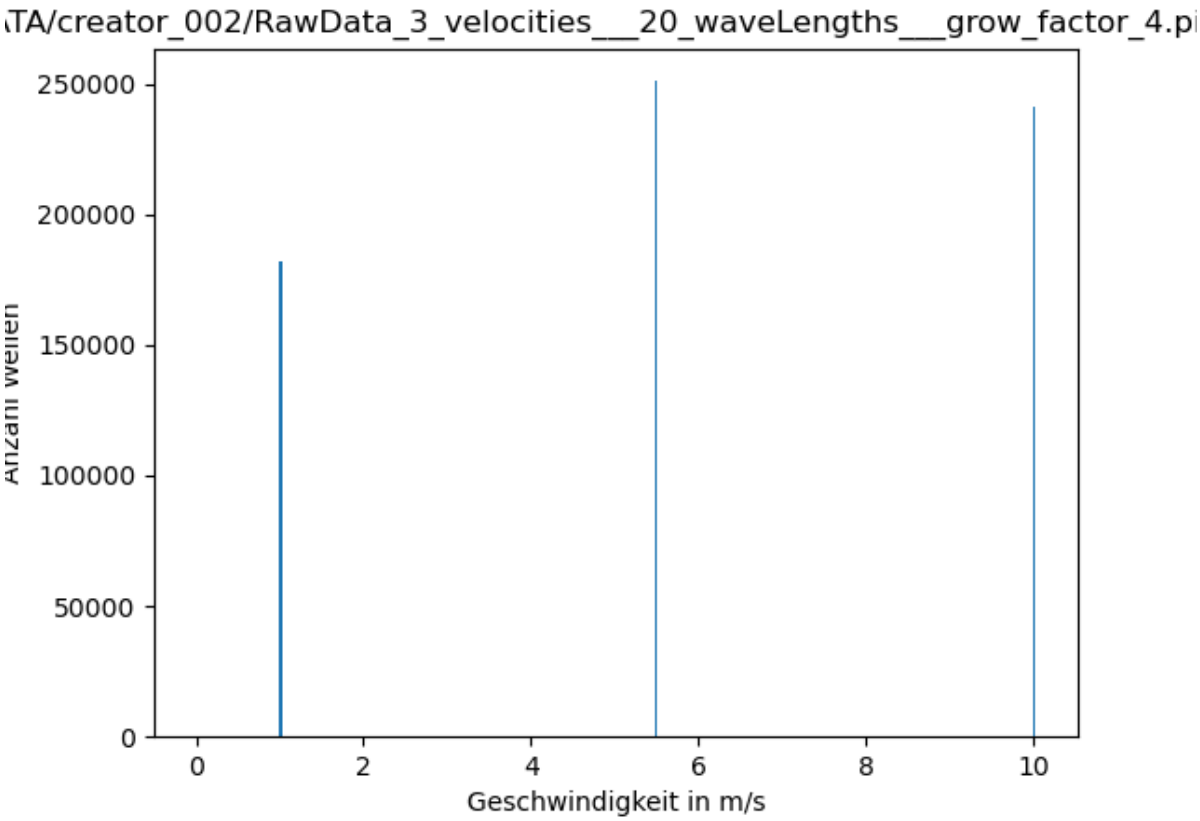

Abbildung 3: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_*002 erzeugt wurde. Auch dieser Datensatz enthält drei verschiedene Geschwindigkeiten. Bei 5.5 m/s erkennt man, dass die Anzahl an Wellen für hohe Geschwindigkeiten zwischendurch höher sein kann (vgl. 3.2).

4.2.2 Histogramm mit 30 Geschwindigkeiten und 30 Wellenlängen

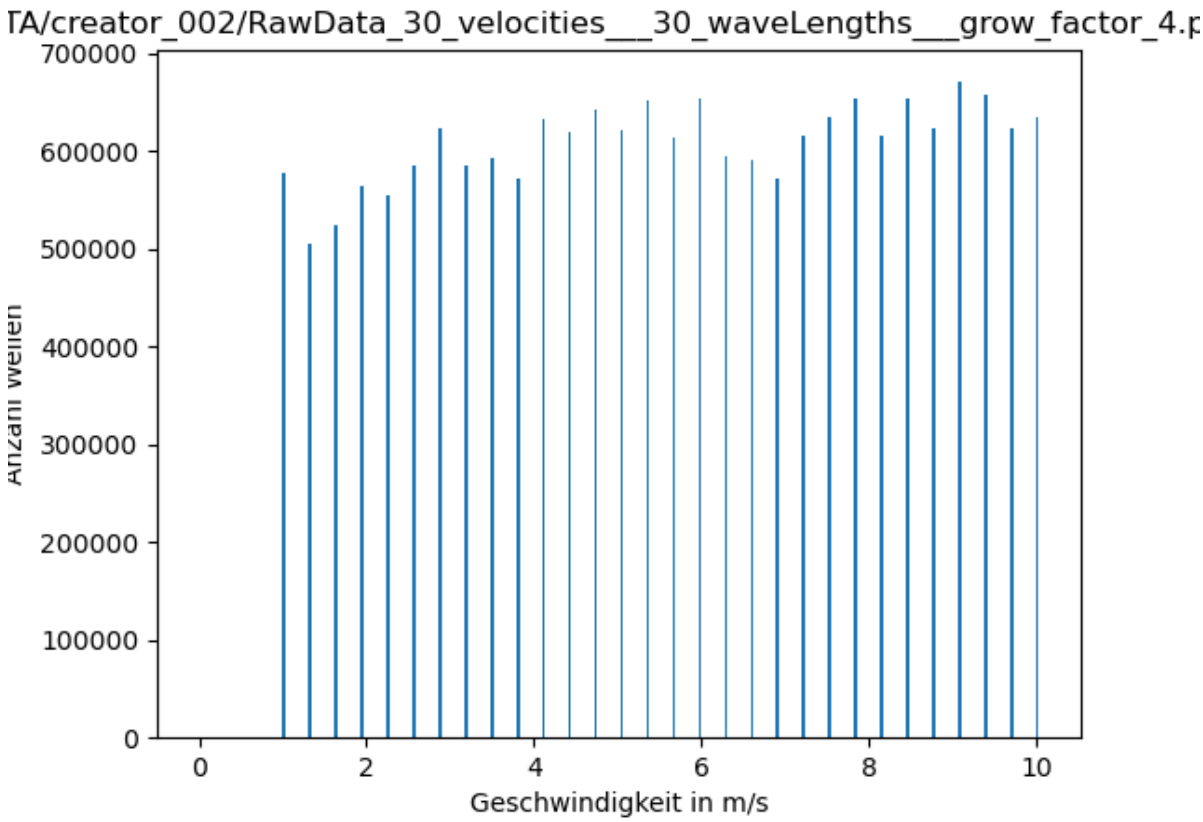

Abbildung 4: Das Histogramm eines Datensatzes mit dreißig Geschwindigkeiten, der mithilfe des *data_creator*_002 erzeugt wurde.

4.2.3 Terminalausgabe bei 30 Geschwindigkeiten und 30 Wellenlängen

```
(base) PS D:\sandbox\github\aarons> python .\metrik_waves.py 1 30 30
[Geschwindigkeit, Anzahl Wellen]
[[1.0, 36853.0], [1.3103448275862069, 36571.0], [1.6206896551724137, 39807.0], [1.9310344827586208, 41361.0], [2.2413793
103448274, 42483.0], [2.5517241379310347, 43587.0], [2.8620689655172415, 44331.0], [3.1724137931034484, 44924.0], [3.482
7586206896552, 45068.0], [3.793103448275862, 45656.0], [4.103448275862069, 45969.0], [4.413793103448276, 46218.0], [4.72
4137931034483, 46424.0], [5.0344827586206895, 46701.0], [5.344827586206897, 46885.0], [5.655172413793103, 46941.0], [5.9
655172413793105, 47297.0], [6.275862068965517, 47322.0], [6.586206896551724, 47578.0], [6.896551724137931, 47603.0], [7.
206896551724138, 47726.0], [7.517241379310345, 47730.0], [7.827586206896552, 47848.0], [8.137931034482758, 47987.0], [8.
448275862068966, 48020.0], [8.758620689655173, 48152.0], [9.068965517241379, 48106.0], [9.379310344827585, 48268.0], [9.
689655172413794, 48376.0], [10.0, 48268.0]]
(base) PS D:\sandbox\github\aarons> python .\metrik_waves.py 2 30 30
[Geschwindigkeit, Anzahl Wellen]
[[1.0, 578086.0], [1.3103448275862069, 506016.0], [1.6206896551724137, 524356.0], [1.9310344827586208, 563491.0], [2.241
3793103448274, 554157.0], [2.5517241379310347, 585986.0], [2.8620689655172415, 622711.0], [3.1724137931034484, 584258.0]
, [3.4827586206896552, 591894.0], [3.793103448275862, 571288.0], [4.103448275862069, 633343.0], [4.413793103448276, 6200
73.0], [4.724137931034483, 641398.0], [5.0344827586206895, 622079.0], [5.344827586206897, 651813.0], [5.655172413793103,
 614481.0], [5.9655172413793105, 653609.0], [6.275862068965517, 594518.0], [6.586206896551724, 590959.0], [6.896551724137
7931, 572367.0], [7.206896551724138, 615173.0], [7.517241379310345, 634489.0], [7.827586206896552, 654523.0], [8.1379310
34482758, 615487.0], [8.448275862068966, 654386.0], [8.758620689655173, 623791.0], [9.068965517241379, 670520.0], [9.379
310344827585, 658193.0], [9.689655172413794, 623131.0], [10.0, 634314.0]]
```

Abbildung 5: Die Ausgabe im Terminal zu den beiden Histogrammen aus 2 und 4.