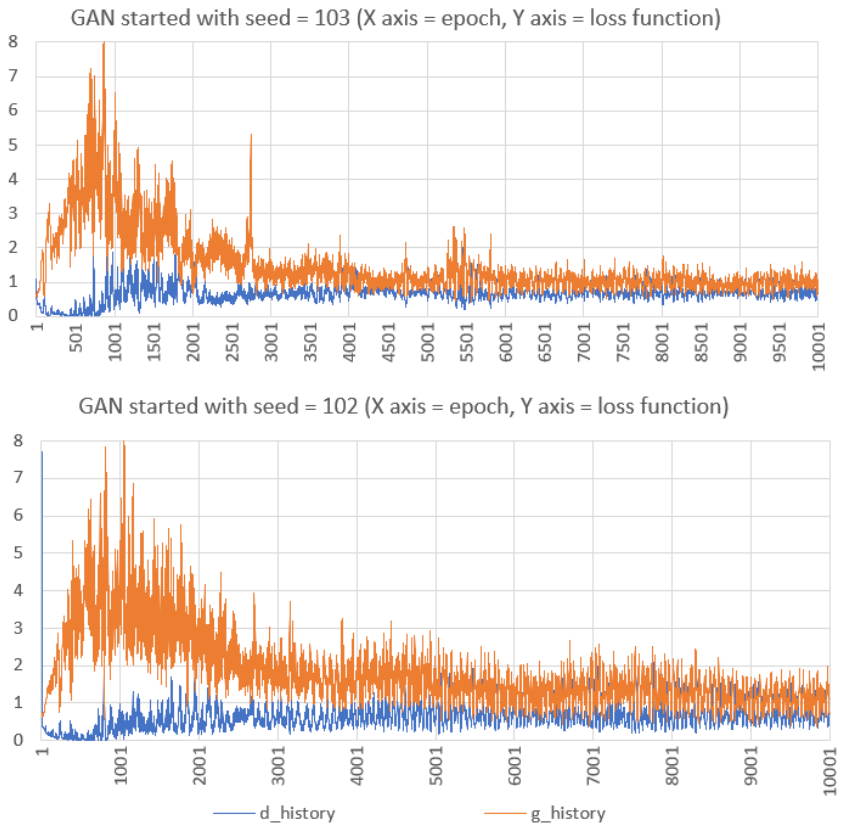
Practical AI & Machine Learning Projects and Datasets



Vincent Granville, Ph.D. | www.MLTechniques.com | Version 1.0, June 2023

Preface

This book is aimed at participants in the AI and machine learning certification program organized by my research lab MLtechniques.com. It features the projects offered to the participants, and covers generative AI, synthetic data, machine learning optimization, scientific computing with Python, data visualizations and animations, time series and spatial processes, NLP and large language models, as well as graph applications and more. These GPT-proof projects based on real life data, with solution and enterprise-grade Python code, are a great addition to your portfolio (GitHub) when completed. Hiring managers and instructors can use them as original, off-the-beaten-path homework to test candidates, as a complement or to differentiate themselves from competitors relying on overused, run-of-the-mill exercises.

To see how the program works and earn your certification(s), check out our FAQ posted here, or click on the "certification" tab on our website MLtechniques.com. Certifications can easily be displayed on your LinkedIn profile page in the credentials section, with just one click. Unlike many other programs, there is no exam or meaningless quizzes. Emphasis is on projects with real-life data, enterprise-grade code, efficient methods, and modern applications to build a strong portfolio and grow your career in little time. The guidance to succeed is provided by the founder of the company, one of the top and most well-known experts in machine learning, Dr. Vincent Granville. Jargon and unnecessary math are avoided, and simplicity is favored whenever possible. Nevertheless, the material is described as advanced by everyone who looked at it.

The related teaching and technical material (textbooks) can be purchased at MLtechniques.com/shop/. MLtechniques.com, the company offering the certifications, is a private, self-funded AI/ML research lab developing state-of-the-art open source technologies related to synthetic data, generative AI, cybersecurity, geospatial modeling, stochastic processes, chaos modeling, and AI-related statistical optimization.

About the author

Vincent Granville is a pioneering data scientist and machine learning expert, co-founder of Data Science Central (acquired by TechTarget), founder of MLTechniques.com, former VC-funded executive, author and patent owner.

Vincent's past corporate experience includes Visa, Wells Fargo, eBay, NBC, Microsoft, and CNET. Vincent is also a former post-doc at Cambridge University, and the National Institute of Statistical Sciences (NISS). He published in *Journal of Number Theory*, *Journal of the Royal Statistical Society* (Series B), and *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He is also the author of multiple books, available here. He lives in Washington state, and enjoys doing research on stochastic processes, dynamical systems, experimental math and probabilistic number theory.

Chapter 4

Scientific Computing

Many projects throughout this book feature scientific programming in Python. This section offers a selection that best illustrates what scientific computing is about. In many cases, special libraries are needed, or you have to process numbers with billions of digits. Applications range from heavy simulations to cybersecurity. In several instances, very efficient algorithms are required.

4.1 The music of the Riemann Hypothesis: sound generation

This first project is an introduction to sound generation in Python with wave files, as well as the MPmath Python library to work with special functions including in the complex plane. This library is particularly useful to physicists. No special knowledge of complex functions is required: we work with the real and imaginary part separately, as illustrated in Figure 4.1: in this example, the frequency attached to a musical note is the real part of the complex Dirichlet eta function η [Wiki] (re-scaled to be positive), the duration of a note is the imaginary part of η after re-scaling, and the volume – also called amplitude – is the the modulus [Wiki].

The project consists of the following steps:

Step 1: Read my article "The Sound that Data Makes", available here. It contains Python code to turn random data into music. Also download the technical paper "Math-free, Parameter-free Gradient Descent in Python" available from here. It contains Python code to compute the Dirichlet eta function (among others) using the MPmath library.

Step 2: Using the material gathered in step 1, generate a sound file (wav extension) for the Dirichlet eta function $\eta(\sigma + it)$, with $\sigma = \frac{1}{2}$ and t between 400,000 and 400,020. Create 300 musical notes, one for each value of t equally spaced in the interval in question. Each note has 3 components: the frequency, duration, and volume, corresponding respectively to the real part, imaginary part, and modulus of $\eta(\sigma + it)$ after proper re-scaling.

Step 3: Same as step 2, but this time with a different function of your choice. Or better, with actual data rather than sampled values from mathematical functions. For instance, try with the ocean tide data, or the planet inter-distance data investigated in project 3.1.

Step 4: Optional. Create a sound track for the video featured in Figure 4.2. You can watch the video on YouTube, here. The code to produce this video (with detailed explanations) is in section 4.3.2 in my book [5], and also on GitHub, here. To blend the sound track and the video together, see solution on Stack Exchange, here. An alternative is to convert the wav file to mp4 formart (see how to do it here) then use the Moviepy library to combine them both.

xxx link to RH xxx stereo / multiple instruments volume max when hitting a root

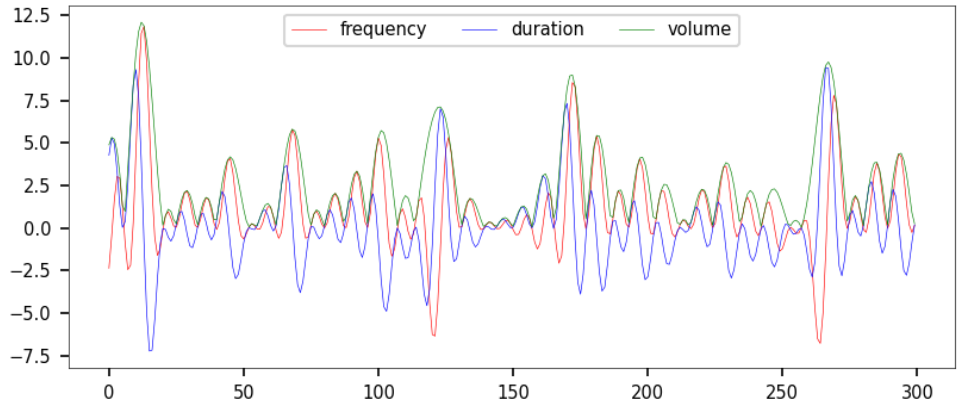


Figure 4.1: 300 musical notes, showing volume, duration and frequency

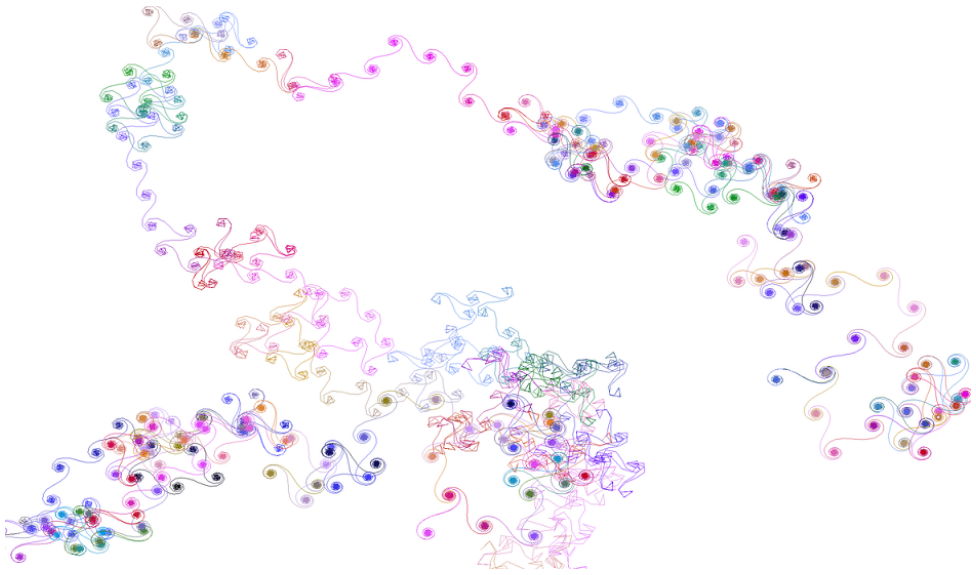


Figure 4.2: Last frame from the video featuring the convergence of the Dirichlet eta function

https://github.com/VincentGranville/Experimental-Math-Number-Theory/blob/main/Source-Code/soundRH.py
youtube video img sound

4.1.1 Solution

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.io import wavfile
import mpmath

#-- Create the list of musical notes

scale=[]
for k in range(35, 65):
   note=440*2**((k-49)/12)
   if k%12 != 0 and k%12 != 2 and k%12 != 5 and k%12 != 7 and k%12 != 10:
      scale.append(note) # add musical note (skip half tones)
n_notes = len(scale) # number of musical notes


#-- Generate the data

n = 300
```

```
sigma = 0.5
min_t = 400000
max_t = 400020

def create_data(f, nobs, min_t, max_t, sigma):
   z_real = []
   z_imag = []
   z_modulus = []
   incr_t = (max_t - min_t) / nobs
   for t in np.arange(min_t, max_t, incr_t):
      if f == 'Zeta':
         z = mpmath.zeta(complex(sigma, t))
      elif f == 'Eta':
         z = mpmath.altzeta(complex(sigma, t))
      z_real.append(float(z.real))
      z_imag.append(float(z.imag))
      modulus = np.sqrt(z.real*z.real + z.imag*z.imag)
      z_modulus.append(float(modulus))
   return(z_real, z_imag, z_modulus)

(z_real, z_imag, z_modulus) = create_data('Eta', n, min_t, max_t, sigma)

size = len(z_real) # should be identical to nobs
x = np.arange(size)

# frequency of each note
y = z_real
min = np.min(y)
max = np.max(y)
yf = 0.999*n_notes*(y-min)/(max-min)

# duration of each note
z = z_imag
min = np.min(z)
max = np.max(z)
zf = 0.1 + 0.4*(z-min)/(max-min)

# volume of each note
v = z_modulus
min = np.min(v)
max = np.max(v)
vf = 500 + 2000*(1 - (v-min)/(max-min))

#-- plot data

mpl.rcParams['axes.linewidth'] = 0.3
fig, ax = plt.subplots()
ax.tick_params(axis='x', labelsize=7)
ax.tick_params(axis='y', labelsize=7)
plt.rcParams['axes.linewidth'] = 0.1
plt.plot(x, y, color='red', linewidth = 0.3)
plt.plot(x, z, color='blue', linewidth = 0.3)
plt.plot(x, v, color='green', linewidth = 0.3)
plt.legend(['frequency','duration','volume'], fontsize="7",
   loc ="upper center", ncol=3)
plt.show()

#-- Turn the data into music

def get_sine_wave(frequency, duration, sample_rate=44100, amplitude=4096):
   t = np.linspace(0, duration, int(sample_rate*duration))
   wave = amplitude*np.sin(2*np.pi*frequency*t)
   return wave

wave=[]
for t in x: # loop over dataset observations, create one note per observation
```

```
    note = int(yf[t])
    duration = zf[t]
    frequency = scale[note]
    volume = vf[t] ## 2048
    new_wave = get_sine_wave(frequency, duration = zf[t], amplitude = vf[t])
    wave = np.concatenate((wave,new_wave))
wavfile.write('sound.wav', rate=44100, data=wave.astype(np.int16))
```

---

grade math multiplication, correl 1/(pq) math-free gradient descent mpmath euler product L4 gmpy2 long runs in quadratic irrationals: runs6b.py and maxrun.py

link to scientific computing, collision graphs nice graph in LaTeX, nice one with star collisions agglomerative processes

xxx add more textcolor/index xxxx graph generation / sound generation: move to scientific data xxx agent base modeling : geospatial section xxx make this interview questions

Bibliography

[1] Vincent Granville. Feature clustering: A simple solution to many machine learning problems. *Preprint*, pages 1–6, 2023. MLTechniques.com [Link]. 32

[2] Vincent Granville. Generative AI: Synthetic data vendor comparison and benchmarking best practices. *Preprint*, pages 1–13, 2023. MLTechniques.com [Link]. 25

[3] Vincent Granville. Massively speed-up your learning algorithm, with stochastic thinning. *Preprint*, pages 1–13, 2023. MLTechniques.com [Link]. 32

[4] Vincent Granville. Smart grid search for faster hyperparameter tuning. *Preprint*, pages 1–8, 2023. MLTechniques.com [Link]. 30, 32

[5] Vincent Granville. *Synthetic Data and Generative AI*. Elsevier, 2024. [Link]. 14, 20, 24, 25, 29, 30, 32, 33, 35

[6] Elisabeth Griesbauer. *Vine Copula Based Synthetic Data Generation for Classification*. 2022. Master Thesis, Technical University of Munich [Link]. 31

[7] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *Preprint*, pages 1–23, 2022. Submitted to IEEE PAMI [Link]. 30

Index