Relatório do EP de MAC0209

João Pedro Feitosa, Marcelo Nascimento, Renan Ryu Kajihara

6 de março de 2024

Resumo

O presente relatório irá descrever as atividades realizadas no primeiro Exercício Programa da disciplina "Modelagem e Simulação (MAC0209)", que consiste em utilizar a linguagem de programação Python e suas bibliotecas para amostrar e plotar gráficos de diferentes funções, observando o impacto de diferentes taxas de amostragem no comportamento dessas funções na reta real.

Conteúdo

1 Introdução

A fim de familiarizar os alunos da disciplina com o uso de Jupyter Notebooks e a linguagem de programação Python, foi solicitada a condução de um experimento simples: os alunos deveriam, com o auxílio da ferramenta online Google Colaboratory, criar um arquivo do tipo Jupyter Notebook e, dentro deste, traçar gráficos para as funções seno, logarítmica e arco-tangente para diferentes taxas de amostragem. Os resultados desse experimento são mostrados a seguir neste relatório.

2 Objetivos

O objetivo que se buscou com este experimento foi verificar o impacto que diferentes taxas de amostragem têm na distribuição das funções estudadas, sendo elas as funções seno, logarítmica e arco-tangente, na reta real.

3 Cronograma

Amostragem e "plotagem"dos gráficos das funções e tabelamento dos dados dos experimentos : 03/03

Redação do relatório: 04/03 - 05/03

4 Dados e métodos

Utilizou-se, em primeiro lugar, a biblioteca NumPy do Python para a geração de intervalos de dados equiespaçados com limites inferior e superior fixos, mas com diferentes taxas de amostragem. Os limites escolhidos para os intervalos foram de 0.001 e 10, respectivamente. A escolha de 0.001 como intervalo inferior se tratou de um artifício para se aproximar do valor 0, para o qual a função logarítmica apresenta um valor indefinido.

Analisando-se múltiplas taxas de amostragem (t), as taxas escolhidas foram 3, 10 e 20, representativas do fenômeno observado conforme se aumenta o valor de t. Para cada uma dessas taxas, gerou-se os intervalos e, novamente com a utilização do NumPy, salvou-se em disco esses intervalos no formato de arquivo CSV e posteriormente leu-se esses arquivos de volta para o ambiente virtual.

Utilizou-se, por fim, a biblioteca Matplotlib do Python para plotar os gráficos das 3 funções estudadas: seno, logarítmica e arco-tangente, para cada um dos intervalos gerados. Os gráficos obtidos estão representados na próxima sessão deste relatório.

5 Resultados experimentais

Como o experimento é didático e baseado em funções conhecidas, neste caso, é possível visualizar o comportamento das funções na reta real, conforme a figura 1. Normalmente, isto não é possível, pois, o comum é obter um comportamento em função dos dados que já possui. Outro ponto importante é que uma imagem não é algo que valida se um modelo (no caso, as funções) está correto, mas é um instrumento importante para analisar se o modelo está indo para o caminho errado, visto que irá divergir muito do que se é esperado com base nos dados.

O estilo do gráfico foi escolhido para mostrar os pontos das amostras e a conexão dos pontos se torna necessária para mostrar a tendência da forma que estes pontos dendem a seguir.

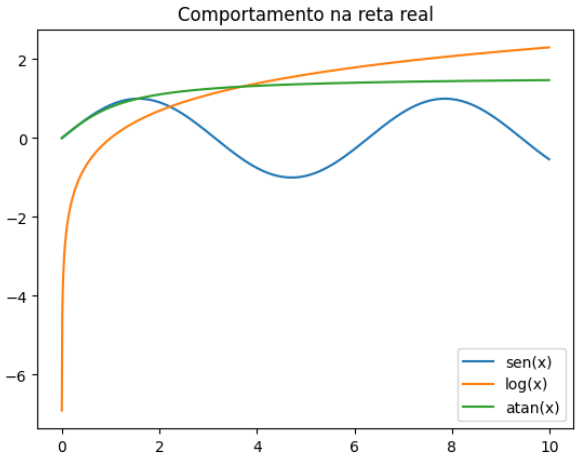


Figura 1: Comportamento das funções na reta real

Na figura 2, temos o resultado para um experimento com três amostras e fica nítido que a quantidade de pontos é muito pequena, fazendo com que o gráfico fique muito grosseiro e não demonstre uma forma aparente que possa ser estudada.

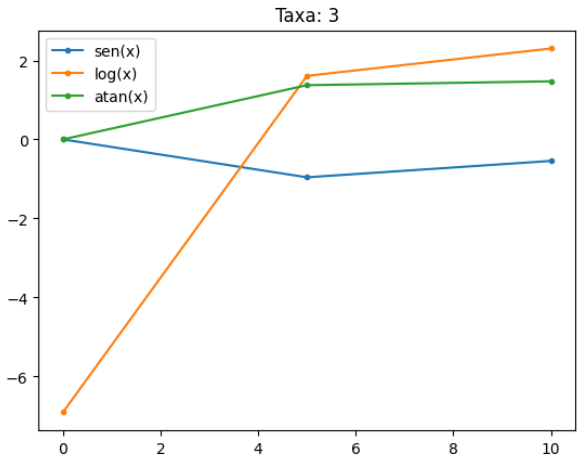


Figura 2: Comportamento com taxa de amostragem igual a 3

TAXA AMOSTRAGEM -> 3

	amostras	seno	log	arctg
0	0.0010	0.001000	-6.907755	0.001000
1	5.0005	-0.958782	1.609538	1.373420
2	10.0000	-0.544021	2.302585	1.471128

Figura 3: Dados do experimento com taxa igual a 3

Na figura 4, o gráfico começa a tomar forma das funções originais, mas ainda não se assemelha a uma função contínua.

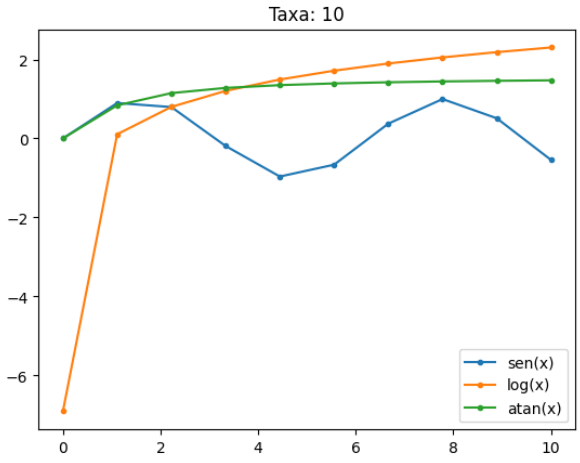


Figura 4: Comportamento com taxa de amostragem igual a 10

TAXA AMOSTRAGEM -> 10

	amostras	seno	log	arctg
0	0.001	0.001000	-6.907755	0.001000
1	1.112	0.896586	0.106160	0.838379
2	2.223	0.794748	0.798858	1.148073
3	3.334	-0.191222	1.204173	1.279395
4	4.445	-0.964464	1.491780	1.349509
5	5.556	-0.664770	1.714878	1.392717
6	6.667	0.374460	1.897170	1.421914
7	7.778	0.997115	2.051299	1.442930
8	8.889	0.510510	2.184815	1.458769
9	10.000	-0.544021	2.302585	1.471128

Figura 5: Dados do experimento com taxa igual a 10

Finalmente, na figura 6, o gráfico já se assemelha as funções originais e após este número, visualmente, não há mais grandes diferenças entre as taxas de amostragem para o domínio escolhido.

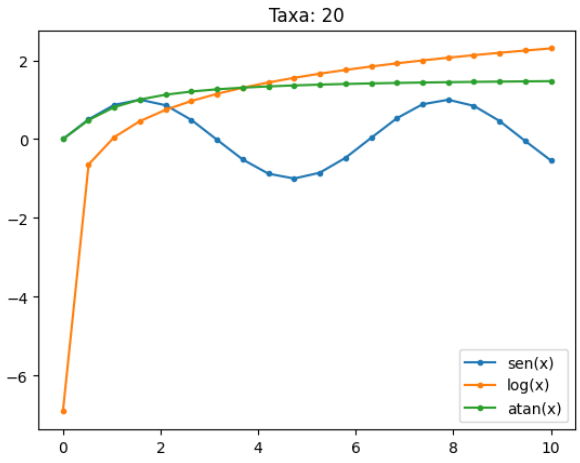


Figura 6: Comportamento com taxa de amostragem igual a 20

	amostras	seno	log	arctg
0	0.001000	0.001000	-6.907755	0.001000
1	0.527263	0.503170	-0.640056	0.485220
2	1.053526	0.869172	0.052143	0.811458
3	1.579789	0.999960	0.457292	1.006468
4	2.106053	0.860138	0.744815	1.127493
5	2.632316	0.487546	0.967864	1.207742
6	3.158579	-0.016985	1.150122	1.264182
7	3.684842	-0.516920	1.304228	1.305797
8	4.211105	-0.876966	1.437725	1.337647
9	4.737368	-0.999688	1.555482	1.362763
10	5.263632	-0.851874	1.660821	1.383051
11	5.789895	-0.473527	1.756114	1.399769
12	6.316158	0.032967	1.843111	1.413776
13	6.842421	0.530539	1.923142	1.425677
14	7.368684	0.884536	1.997239	1.435911
15	7.894947	0.999161	2.066223	1.444804
16	8.421211	0.843393	2.130754	1.452602
17	8.947474	0.459386	2.191371	1.459495
18	9.473737	-0.048939	2.248523	1.465631
19	10.000000	-0.544021	2.302585	1.471128

Figura 7: Dados do experimento com taxa igual a 20

6 Conclusão

Este exercício demonstrou como a quantidade de amostras é importante em um experimento. Com poucas amostras não é possível generalizar os dados para um comportamento maior, porém, a partir de um determinado número de amostras, a coleta se torna pouco necessária, uma vez que o comportamento já pode ser previsto, com bastante precisão, por meio dos dados obtidos. Entretanto é nítido que, quanto maior o número de dados coletados, maior a pecisão dos resultados obtidos.

Além disso, é perceptível que ferramentas como Numpy, Pandas e Matplotlib são ótimas para processamento e visualização de dados. Com poucos comandos é possível invocar algorítmos e rotinas que demorariam muito mais tempo se fosse necessário implementa-las do zero.

7 Notebook

As saídas dos comandos apt-get e pip foram reduzidas, pois, na versão orignal, o conteúdo total preencheu 10 páginas de logs de instalação.

EP1_MAC0209

March 5, 2024

Exercício de MAC0209 - Modelagem e Simulação

João Pedro Feitosa - 10741569 (IME-USP)

Marcelo Nascimento - 11222012 (IME-USP)

Renan Ryu Kajihara - 14605762 (IME-USP)

EP Módulo de Introdução ***

1 Setup

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

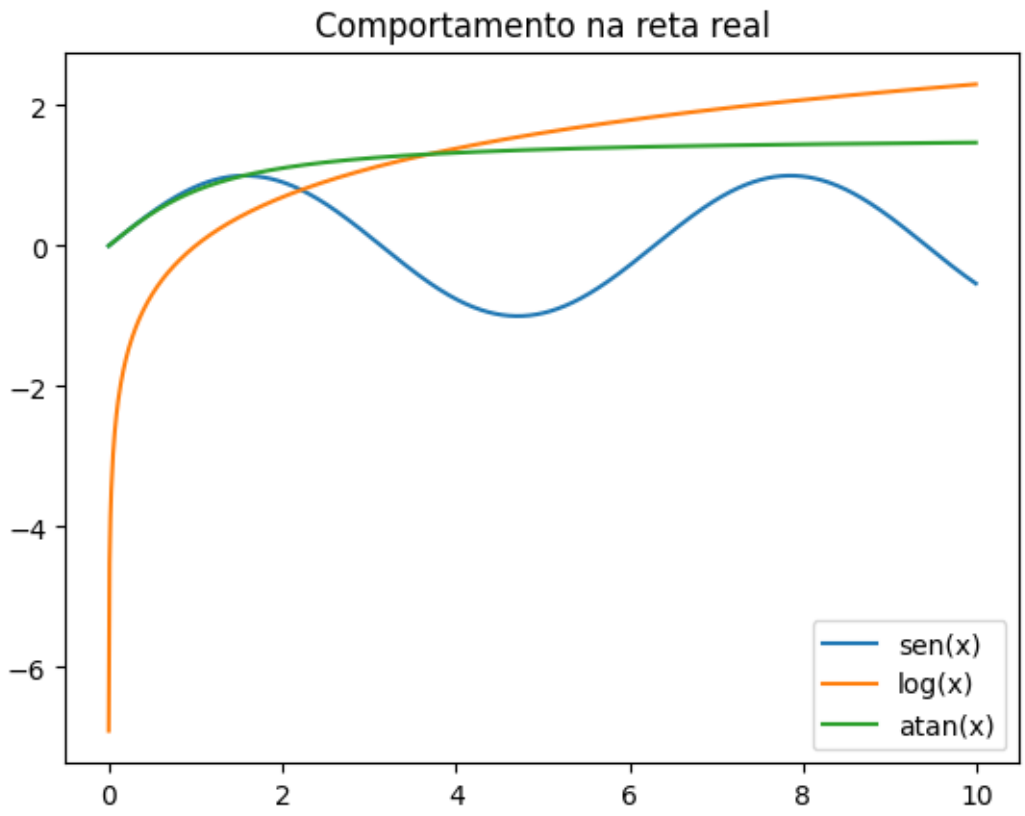
2 Lib

[ ]:
```
def sen(t):
  return np.sin(t)

def log(t):
  return np.log(t)

def atat(t):
  return np.arctan(t)

def amostragem(n):
  return np.linspace(0.001, 10, n)
```

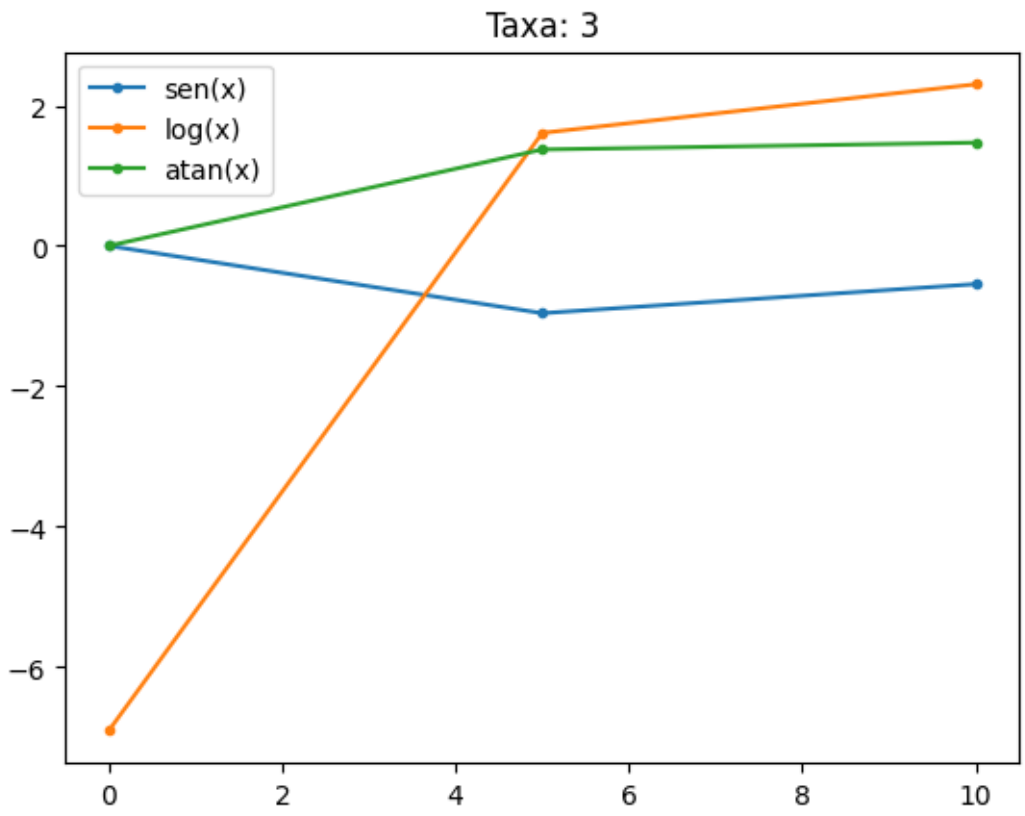
3 Main

[ ]:
```
def main():

  # Comportamento na reta real
  amostra = np.arange (0.001, 10, 0.01)
```

```
  plt.plot(amostra, sen(amostra), label='sen(x)')
  plt.plot(amostra, log(amostra), label='log(x)')
  plt.plot(amostra, atat(amostra), label='atan(x)')
  plt.legend()
  plt.title('Comportamento na reta real')
  plt.show()

  # Funções por taxa de amostragem
  for taxa_amostragem in [3, 10, 20]:

    print()
    print(f"TAXA AMOSTRAGEM -> {taxa_amostragem}")

    # Gera e salva as amostras
    amostra_gerada = amostragem(taxa_amostragem)
    np.savetxt(str(taxa_amostragem) + '_sen.csv',    [amostra_gerada,␣
↪sen(amostra_gerada)],  delimiter=',')
    np.savetxt(str(taxa_amostragem) + '_log.csv',    [amostra_gerada,␣
↪log(amostra_gerada)],  delimiter=',')
    np.savetxt(str(taxa_amostragem) + '_atan.csv',   [amostra_gerada,␣
↪atat(amostra_gerada)], delimiter=',')

    # Le as amostras
    amostra_sen, sen_amostra = np.loadtxt(str(taxa_amostragem) + '_sen.csv',␣
↪delimiter=',')
    amostra_log, log_amostra = np.loadtxt(str(taxa_amostragem) + '_log.csv',␣
↪delimiter=',')
    amostra_atan, atan_amostra = np.loadtxt(str(taxa_amostragem) + '_atan.csv',␣
↪delimiter=',')

    # Imprime os dados em tabela
    dados = pd.DataFrame(np.transpose([amostra_sen, sen_amostra, log_amostra,␣
↪atan_amostra]), columns= ['amostras', 'seno', 'log','arctg'])
    display(dados)

    print()

    # Gera os graficos
    plt.plot(amostra_sen, sen_amostra, '.-', label='sen(x)')
    plt.plot(amostra_log, log_amostra, '.-', label='log(x)')
    plt.plot(amostra_atan, atan_amostra, '.-', label='atan(x)')
    plt.legend()
    plt.title('Taxa: ' + str(taxa_amostragem))
    plt.show()

main()
```

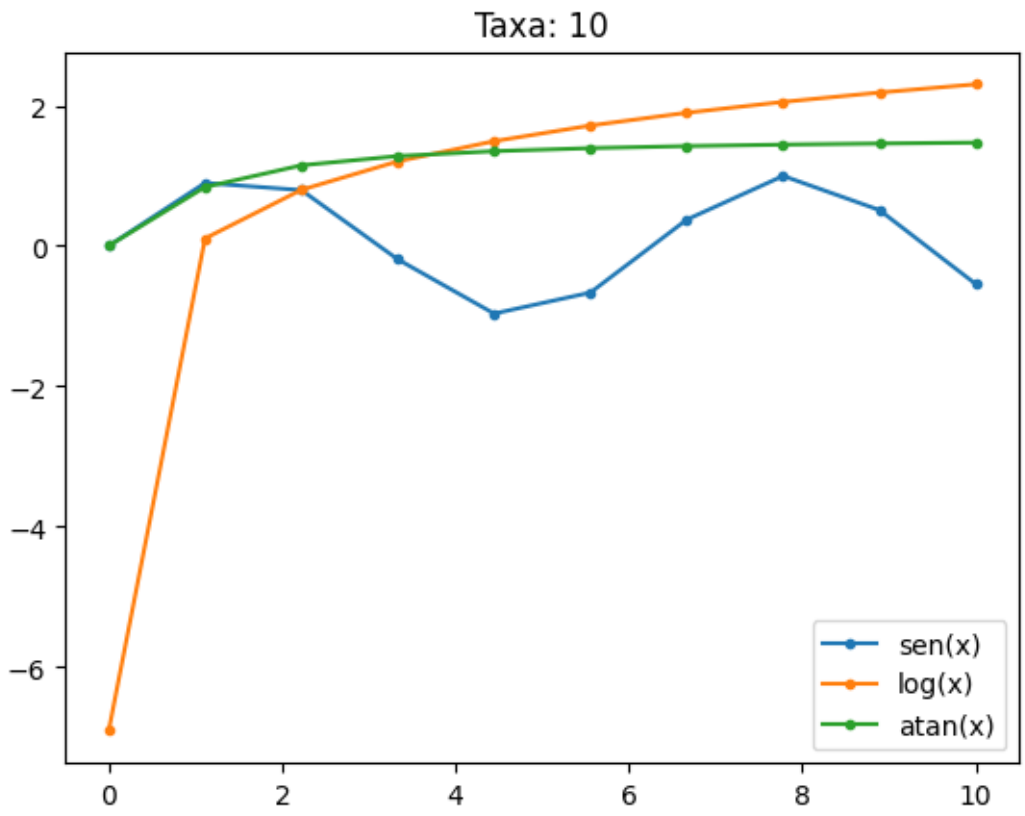



```
TAXA AMOSTRAGEM -> 3

   amostras      seno       log     arctg
0    0.0010  0.001000 -6.907755  0.001000
1    5.0005 -0.958782  1.609538  1.373420
2   10.0000 -0.544021  2.302585  1.471128
```



TAXA AMOSTRAGEM -> 10

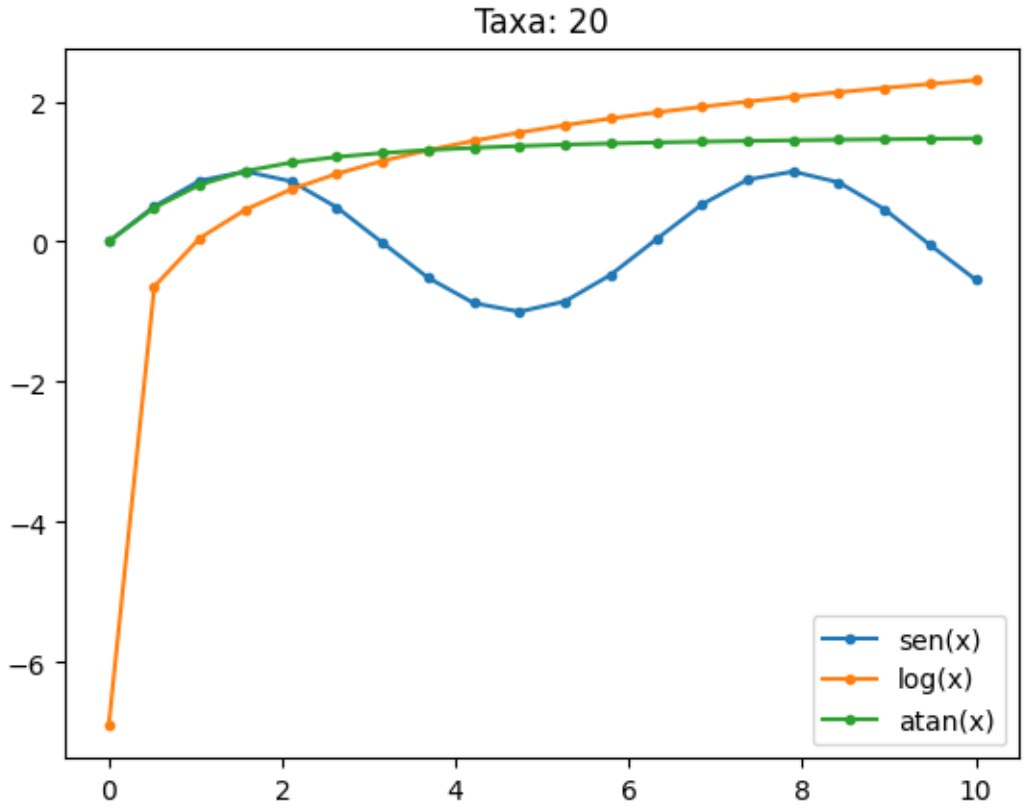
	amostras	seno	log	arctg
0	0.001	0.001000	-6.907755	0.001000
1	1.112	0.896586	0.106160	0.838379
2	2.223	0.794748	0.798858	1.148073
3	3.334	-0.191222	1.204173	1.279395
4	4.445	-0.964464	1.491780	1.349509
5	5.556	-0.664770	1.714878	1.392717
6	6.667	0.374460	1.897170	1.421914
7	7.778	0.997115	2.051299	1.442930
8	8.889	0.510510	2.184815	1.458769
9	10.000	-0.544021	2.302585	1.471128



TAXA AMOSTRAGEM -> 20

	amostras	seno	log	arctg
0	0.001000	0.001000	-6.907755	0.001000
1	0.527263	0.503170	-0.640056	0.485220
2	1.053526	0.869172	0.052143	0.811458
3	1.579789	0.999960	0.457292	1.006468
4	2.106053	0.860138	0.744815	1.127493
5	2.632316	0.487546	0.967864	1.207742
6	3.158579	-0.016985	1.150122	1.264182
7	3.684842	-0.516920	1.304228	1.305797
8	4.211105	-0.876966	1.437725	1.337647
9	4.737368	-0.999688	1.555482	1.362763
10	5.263632	-0.851874	1.660821	1.383051
11	5.789895	-0.473527	1.756114	1.399769
12	6.316158	0.032967	1.843111	1.413776
13	6.842421	0.530539	1.923142	1.425677
14	7.368684	0.884536	1.997239	1.435911
15	7.894947	0.999161	2.066223	1.444804
16	8.421211	0.843393	2.130754	1.452602
17	8.947474	0.459386	2.191371	1.459495

```
18    9.473737 -0.048939   2.248523   1.465631
19   10.000000 -0.544021   2.302585   1.471128
```



[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
pandoc set to manually installed.
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcommons-logging-java libcommons-parent-
java
  libfontbox-java libfontenc1 libgs9 libgs9-common libidn12 libijs-0.35
libjbig2dec0 libkpathsea6
  libpdfbox-java libptexenc1 libruby3.0 libsynctex2 libteckit0 libtexlua53
libtexluajit2 libwoff1
```

[ ]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[1]:
```
!cp "drive/My Drive/Colab Notebooks/EP1_MAC0209.ipynb" ./
#!jupyter nbconvert--to PDF "EP1_MAC0209.ipynb"
#!cp "EP1_MAC0209.pdf" "drive/My Drive/Colab Notebooks/"
```

```
cp: cannot stat 'drive/My Drive/Colab Notebooks/EP1_MAC0209.ipynb': No such file
or directory
```