Universidade Católica de Brasília

Pró-Reitoria Acadêmica
Curso de Ciência da Computação
Trabalho N1

IMPLEMENTAÇÃO DE UM PROBLEMA NA COMPUTAÇÃO

Autores: Arthur Lemos Bendini
Beatriz Nascimento Costa Medina
João Felipe Schwaab Pinto

Orientador: Professor Caio Costa

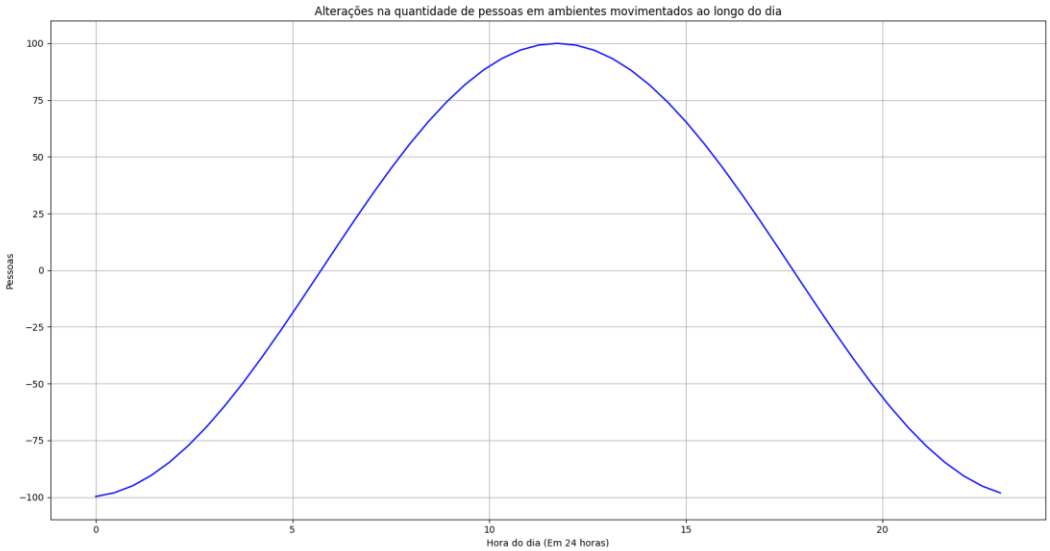
Brasília - DF
2024

MODELO MATEMÁTICO – OSCILAÇÃO DA PRESENÇA DE PESSOAS EM UM AMBIENTE MOVIMENTADO CONFORME O HORÁRIO DO DIA

Locais mais movimentados, como feiras, costumam ter uma oscilação na presença de pessoas no ambiente a depender do horário. É perceptível que os horários comerciais de certos ambientes são grandes influenciadores nessas mudanças. Este modelo matemático possibilita uma análise simples dessa oscilação de pessoas em um certo ambiente comercial ao decorrer das 24 horas do dia.

Sendo assim, utilizando a linguagem de programação *Python* e uma biblioteca denominada *matplotlib* com o *Visual Studio Code*, é possível criar uma representação em gráfico desse fenômeno oscilatório com uma equação, e em seguida plotando, neste gráfico, uma função ondulatória que se repete diariamente, preferencialmente durante dias úteis para maior precisão com fatos observados na realidade.

Eis a representação, em gráfico, deste fenômeno:



Observe o eixo X, representando as horas diárias, e o eixo Y, representando a oscilação de pessoas no local em um período de tempo. Nota-se um crescimento

gradual na parte da manhã, até chegar no pico ao meio-dia, que procede a cair conforme a tarde passa.

Esta representação gráfica foi possível com a ajuda do código a seguir. Também está, junto deste documento, anexado o arquivo do código escrito para essa atividade.

modelo_matematico_A1_AT1.py

```python
#Meu modelo matemático envolve a oscilação de pessoas em um ambiente movimentado de acordo com um horário do dia.
#Ex: Em horários comerciais, a movimentação costuma ser bem maior em comparação a outros horários
#Ao longo do dia, o número de pessoas presentes em um local irá ser alterado constantemente, isso pode ser visualizado aqui
import numpy as np
import matplotlib.pyplot as plt

#função da onda de oscilação
def oscila_pessoa(t, amp, freq, fase): #t = tempo; amp = amplitude de onda; freq = frequência de onda; fase = fase física
    return amp * np.sin(2 * np.pi * freq * t + fase) #isto constrói o formato da onda, np.sin faz o formato ondulatório

#Valores dos parâmetros
t = np.linspace(0, 23) #tempo em horas
amp = 100
freq = 1/24 #frequência por hora no dia
fase = -1.5 #Momento em que as pessoas estão mais presentes no local, período entre 9h e 18h, com pico às 12h

pessoas = oscila_pessoa(t,amp,freq,fase) #variável que chama a função


#Construindo o gráfico
plt.figure(figsize=(10,6)) #tamanho do gráfico
plt.plot(t, pessoas, color='blue')

plt.title('Alterações na quantidade de pessoas em ambientes movimentados ao longo do dia') #Título
plt.xlabel('Hora do dia (Em 24 horas)') #Nome do Eixo X
plt.ylabel('Pessoas') #Nome do Eixo Y
plt.grid(True) #Apresenta linhas de grade no gráfico

plt.show() #Mostra o gráfico na tela

```

Primeiro, é preciso importar as bibliotecas do *NumPy* – Para trabalhar com operações matemáticas mais complexas – E o *Matplotlib* – Para possibilitar a criação visual de um gráfico com base no código. (Importados como **np** e **plt** respectivamente, para facilitar a leitura geral do código e reduzir o tempo de digitação)

```python
import numpy as np
import matplotlib.pyplot as plt
```

Em seguida, foi criada uma função que define a aparência e o formato da oscilação da onda no gráfico para representar corretamente os dados. A função define os parâmetros necessários para a formação da onda, são eles:

- t = Tempo;
- amp = Amplitude da onda (Ou Amplitude de pessoas presentes);
- freq = Frequência da onda;
- fase = Fase física da onda.

Após definir os parâmetros, a função deve retornar uma equação matemática que constrói o formato observado no gráfico. Abaixo está a equação utilizada:

```
amp * np.sin(2 * np.pi * freq * t + fase)
```

A equação consiste em multiplicar a amplitude da onda pelo seno do resultado do cálculo em parênteses (2 • π • freq • t + fase). O **seno** é utilizado devido sua propriedade de fornecer suavidade ao formato da onda, além de facilitar o controle da amplitude e frequência da mesma. Já o **π** é utilizado como uma ferramenta de escala da onda. A fase de onda define a posição relativa dela em seu ciclo no gráfico. Indica o ponto em que ela oscila.

Agora, é necessário atribuir valores aos parâmetros da equação para que a onda seja calculada de forma correta.

```
#Valores dos parâmetros
t = np.linspace(0, 23) #tempo em horas
amp = 100
freq = 1/24 #frequência por hora no dia
fase = -1.5 #Momento em que as pessoas estão mais presentes no local, período entre 9h e 18h, com pico às 12h

pessoas = oscila_pessoa(t,amp,freq,fase) #variável que chama a função
```

- **t** – O valor do **Tempo** foi definido utilizando a função *linspace*, que retorna números simetricamente espaçados dentro de um intervalo específico entre dois números. Neste caso, intervalos de 5 em 5 horas foram criados e colocados no eixo X do gráfico.
- **amp** – O valor da **Amplitude** foi definido para representar uma quantidade não especificada de pessoas, variando de -100 a 100 no gráfico ao longo do dia.
- **freq** – O valor da **Frequência** de onda foi definido como uma fração de 1/24, cada parte representando uma hora do dia.

- **fase** – O valor da **Fase** foi definido por tentativa e erro para obter a posição desejada da oscilação no gráfico, para melhor precisão da representação dos dados.

Atribuídos os valores aos seus respectivos parâmetros, é criada uma variável utilizada para chamar a função anteriormente definida. Esta variável será usada para permitir que o gráfico seja plotado com os valores necessários.

Com isso, apenas resta a construção do gráfico, onde entram mais termos do *Matplotlib*.

Primeiro, é feito o uso da função **figure(figsize=(x,y))** para definir o tamanho do gráfico. Depois. A função **plot()** é colocada com dois parâmetros dentro dos parênteses que juntos constroem o formato da imagem do gráfico (Y e X, respectivamente). Por fins estéticos, colocamos a cor da linha como azul.

Feito isso, são definidos o título e os nomes dos eixos X e Y do gráfico para indicar o que cada valor representa. Também é possível ativar as grades quadriculadas do gráfico para melhor visualização com **plt.grid(True)**.

```
#Construindo o gráfico
plt.figure(figsize=(10,6)) #tamanho do gráfico
plt.plot(t, pessoas, color='blue')

plt.title('Alterações na quantidade de pessoas em ambientes movimentados ao longo do dia') #Título
plt.xlabel('Hora do dia (Em 24 horas)') #Nome do Eixo X
plt.ylabel('Pessoas') #Nome do Eixo Y
plt.grid(True) #Apresenta linhas de grade no gráfico

plt.show() #Mostra o gráfico na tela
```

Finalizando, é chamada a função **plt.show()**, fazendo com que o gráfico seja mostrado na tela após o código ser executado. Vale lembrar que o código só poderá ser executado caso a máquina tenha tanto Python quanto o Matplotlib instalados.

CONCLUSÃO

A partir desse levantamento, é relevante citar o significativo uso dessa modelagem matemática, cujo propósito se estende em diversos setores. Pode ser utilizada, por exemplo, como recurso para aumentar a frota de ônibus nos horários de pico, a fim de diminuir a superlotação. Além disso, pode até mesmo ser utilizada para a formação de insights nas áreas de marketing e finanças, como forma de pesquisa para investimentos, ou seja, ocorreria um aproveitamento a mais em horários que estivessem propícios a trafegar maior quantidade de clientes, com o objetivo de obter maiores lucros. Ademais, também é possível utilizá-la para a segurança pública e bem-estar da população, uma vez que há um tráfego intenso de indivíduos em vários períodos, assim, inúmeras pessoas transitam e, consequentemente, as chances de ocorrerem acidentes aumentam. Ao aplicar tal modelo, é possível evitar maus cenários e diversas situações podem melhorar em grandes proporções.

REFERÊNCIAS BIBLIOGRÁFICAS

WIKIPEDIA. Business Hours. Disponível em: https://en.wikipedia.org/wiki/Business_hours Acesso em: 04/04/2024

MATPLOTLIB, PyPlot – Documentação oficial. Pyplot Tutorial. Disponível em: https://matplotlib.org/stable/tutorials/pyplot.html#sphx-glr-tutorials-pyplot-py Acesso em: 04/04/2024

PYTHON. Python Documentation. Python Software Foundation. Disponível em: https://docs.python.org/3/ Acesso em: 04/04/2024