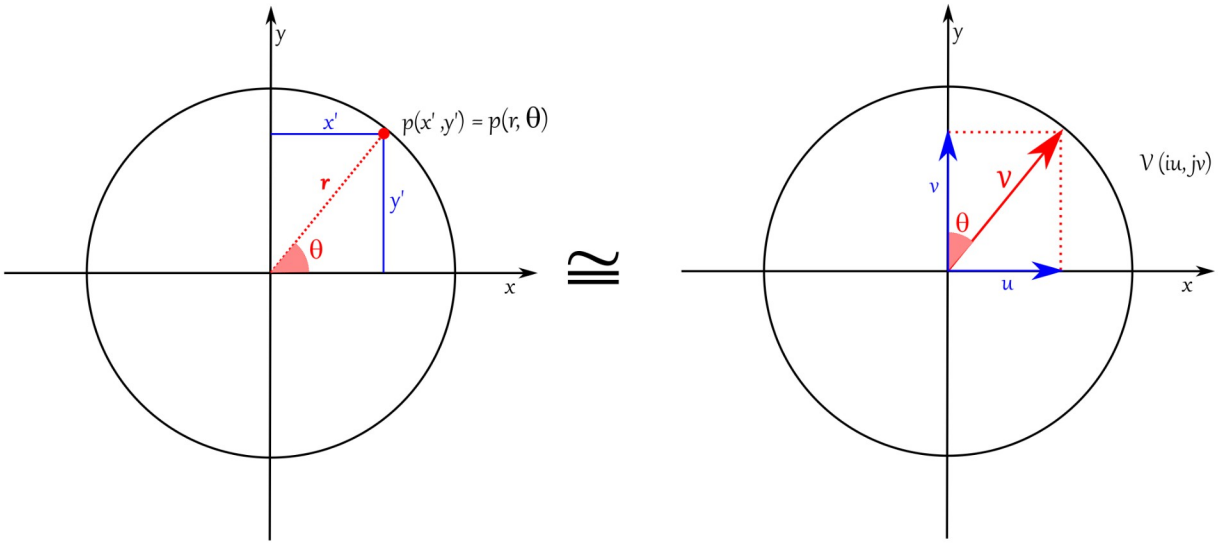
Analisando dados com Python

Processando correntes: uma digressão sobre vetores.

Parte 1

V1. 2021.10.25



Carlos A.F. Schettini

Laboratório de Oceanografia Costeira e Estuarina

LOCostE / IO / FURG

Objetivos:

1 Uma breve revisão sobre vetores

2 Conversão entre sistema polar e cartesiano

3 Rotação de vetores

4 Componentes principais (PCA)

Este tutorial é, inicialmente, uma breve revisão sobre vetores em nível bem elementar, e avançando para um tópico que deixa muita gente confusa que é rotação de vetores. Alunos de graduação de oceanografia muitas (!!) vezes questionam porque tem que fazer álgebra linear e geometria analítica (além de cálculo, mas aqui basta álgebra e geometria). Uma resposta direta é que em absolutamente qualquer área de atuação, cedo ou tarde, alguém vai se deparar com gráficos e regressão linear. Gráficos e sua interpretação já são ensinados antes da faculdade, mas é impossível pensar em ciência sem uma sólida compreensão dos elementos (Euclídes!) básicos da geometria e de álgebra que permitem construir gráficos. É o beabá científico! Então temos que ser devidamente introduzidos aos pontos, linhas e planos, e de quebra, eixos cartesianos. Eu recomendo fortemente que assistam a série de vídeos sobre álgebra linear do canal do Youtube 3Blues1Brown (https://www.youtube.com/watch?v=kjBOesZCoqc&list=PL0-GT3co4r2y2YErbmuJw2L5tW4Ew2O5B).

A parte física da oceanografia, ou oceanografia física, pode, genericamente, ser distinta entre oceanografia física descritiva e oceanografia física dinâmica. A parte descritiva, por sua

vez, pode ser subdividida na descrição de propriedades escalares (salinidade, temperatura, etc.) e vetoriais (corrente e vento [velocidade]). Um escalar é qualquer coisa que pode ser descrito com um único número que representa sua magnitude. Salinidade de 35 psu ou temperatura de 20 °C. Mas nada é tão simples… pois a pergunta é onde e quando a salinidade é 35 psu e a temperatura é 20 °C?

Qualquer observação precisa estar vinculada a um referencial temporal e espacial, caso contrário, não serve absolutamente para nada! Nosso referencial temporal é o calendário Gregoriano (anos, meses, dias). Mas, a marcação do tempo é totalmente relativa, e podemos ter o tempo 'universal' (UTC – coordinated universal time = GMT → Greenwich mean time), ou o tempo local (LT – local time) que depende do fuso horário em relação ao meridiano 0. Tem ainda o horário de verão, que adiciona ou remove 1 hora, às vezes… e tem também o calendário Juliano, e em computação o referencial de tempo pode ser ainda outro (ex: dias a partir de 1/Jan/1970 para o tempo numérico do Python/Matplotib.dates).

Quanto ao referencial espacial, separamos em horizontal e vertical. O horizontal pode ser o geográfico, que é um sistema de coordenadas esférico, onde a latitude é o ângulo entre o Equador e a posição, positivo para o norte e negativo para o sul (-90 a 90 graus) e a longitude é o ângulo em relação a um meridiano de referência (Greenwich, 0 a 360 graus). Mas tem também o sistema de UTM (Universal Transverse Mercator), onde as coordenadas são em distâncias lineares, e entra as 'zonas' para saber exatamente qual é a referência horizontal… Para referência vertical, geralmente usamos o nível médio do mar (msl = mean sea level), que, na verdade, não

existe. Ou, existe para um intervalo de tempo. Para alguns tipos de observações as vezes é mais conveniente usar o fundo como referência, e talvez você já tenha se deparado com um ‘mab’ (meters above the bottom). Tempo a parte, tudo é uma questão de espaço = geometria!

Descrever um vetor é um pouco mais complicado do que descrever um escalar, pois precisamos de pelo menos dois números descritores (sem considerar o tempo). Há mais de uma maneira de descrever vetores (Figura 1). Podemos usar o sistema polar (magnitude e ângulo, V(r, θ)) ou o sistema cartesiano (projeções sobre os eixos, V(u, v)). E aqui é um ponto muito importante para aqueles que querem se aprofundar (!) na parte física. É condição *sine qua non* entender e dominar a manipulação de vetores. Esse tutorial visa dar uma ajuda neste ‘sentido’! Para começar, responda: você sabe qual a diferença entre sentido e direção? Dizer que a corrente tem direção norte não tem sentido! :-). Aqui é o mesmo problema do peso e da massa! Corriqueiramente falamos que vamos nos pesar na farmácia, mas pesar é medir o peso → peso é força e em Newtons, e massa é em kg. Dizer ‘eu peso 75 kg’ é uma blasfêmia física!

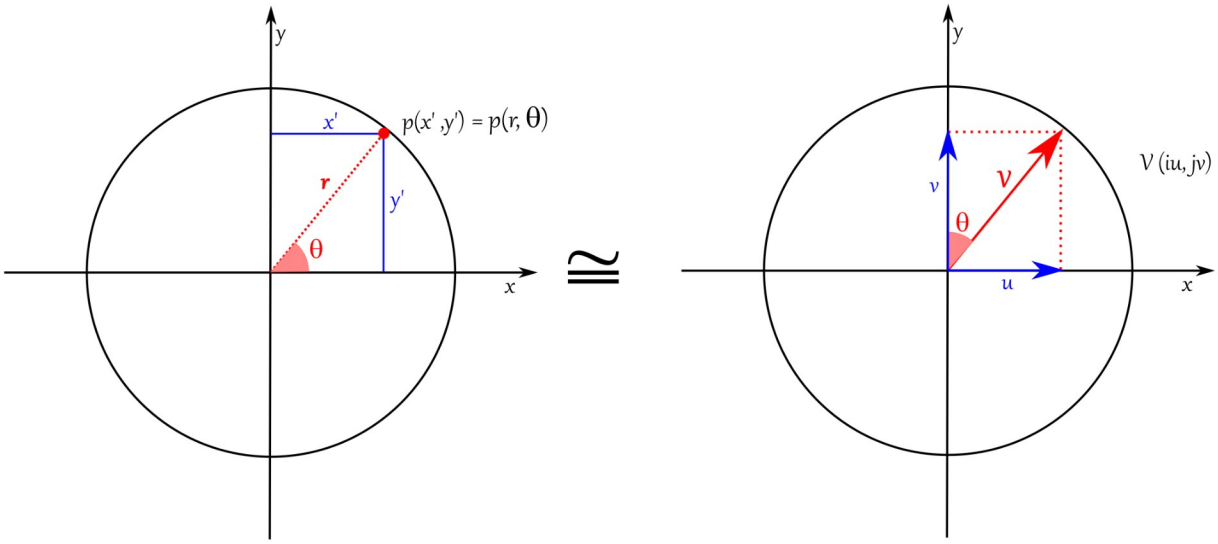


Figura 1: Equivalência das notações de vetores polar e cartesiana.

Não vamos entrar no mérito de como os vetores são registrados. Digo, as correntes e o vento, ou sendo mais específico, a **velocidade e direção** (**o <u>sentido</u> dependerá da notação!** veremos mais adiante) das correntes ou do vento. Há outros 'vetores', tais com a tensão de cisalhamento na superfície e no fundo, que aliás são 'tensores'. Mas vamos nos ater às correntes e ao vento. Porque plural e singular, não sei… mas me soa estranho escrever 'a corrente' ou 'os ventos', mas pode ser uma corrente qualquer (não elétrica!) e qualquer vento ou vários ventos!. Tanto as correntes quanto o vento representam a advecção do meio, e qualquer método que permita representar o deslocamento de uma parcela do meio (distância) em um intervalo de tempo fornecerá a velocidade (da água = correntes, do ar = vento!). Podemos ter abordagens ditas Euleriana (de Euler!) ou Lagrangeanas (de Lagrange!). Independente de qual usamos, obteremos um vetor velocidade em um sistema referencial. Ou seja, o ponto onde estava a

parcela no instante inicial, que será a ‘origem’ do sistema referencial, e o ponto para onde foi a parcela após um intervalo de tempo. Então, temos dois pontos. E, de Euclídes, dois pontos determinam uma linha (e só uma!) no espaço, e a linha não tem extremidades. É infinita. A diferença entre as posições dos pontos sobre esta linha fornece uma distância. No instante inicial tempo apenas um ponto e infinitas linhas passando por esse ponto, em todas as direções no espaço 3D. No instante final teremos dois pontos, e somente uma linha possível, que é a **direção**.

Colocando esta linha na horizontal e perpendicular na sua frente (estamos assim estabelecendo um sistema referencial), colocando o primeiro ponto exatamente à sua frente, podemos estabelecer que esta linha representa os números reais e o ponto a origem, sendo que apartir do ponto para esquerda os números serão < 0 e para direita > 0. Agora estamos estabelecendo a orientação, ou o **sentido**. Isso é arbitrário! Se a maioria de nós fosse canhota, seria mais lógico o contrário! A ordem que os pontos são registrados no tempo determinará para que lado da linha está ocorrendo o deslocamento, ou, para onde está ‘apontando’ o vetor. Podemos dividir a linha em qualquer unidade (metros ou polegada), bem como e o tempo (segundos ou dias), e tudo isso é arbitrário! O que importa é que os referenciais e escalas estejam solidamente estabelecidos.

Antes de pormos a mão na massa, lembro que uma boa fonte de confusão vetorial em oceanografia é a diferença da notação utilizada em meteorologia e oceanografia. A velocidade do vento é um vetor, e assim, a velocidade aponta para onde o vento vai! Mas em meteorologia o vento é referenciado da onde ele vem. Então, quando em meteorologia é dito que o vento é de

nordeste (45°), o sentido do vetor velocidade do vento é 45° + 180° = 225°. Tanto 45° quanto 225° definem a direção, mas o sentido é inequívoco para 225°.

Aproveito para lembrar também que a direção pode ser representada tanto por graus (0° até 360°) quanto por radianos. Um círculo tem 2π radianos ou 360°, e para aqueles que tem dificuldade, a conversão é uma regra de 3: dado um ângulo em graus θ, seu valor em radianos será $\theta \times \pi / 180$. E é bom lembrar aqui também que, via de regra, computadores e calculadoras preferem (!) radianos, a menos que seja explicitamente declarado que trabalha com graus! Se você tem dúvida, faça uma conta óbvia. Por exemplo, seno de 90° é 1 (se pensou 2x, dê uma revisada nas relações trigonométricas!). Se sua calculadora deu 0.89… ela está (!) considerando que a entrada é 90 radianos e não 90 graus! O mesmo vale para o Python!

E já que tocamos no assunto 'trigonometria', tá na mesmo pacote da geometria! Triângulos, senos, cosenos e tangentes… se até agora tu achava que isso era para passar no Enem, agora vai achar um bom uso para tudo isso! Entra aqui mais uma lembrança! Os ângulos trigonométricos tem sua origem no eixo horizontal à direita e aumentam no sentido anti-horário. Os ângulos da direção da velocidade (correntes ou vento) são geralmente referenciados ao norte (magnético ou geográfico!), e aumentam no sentido horário. Mas isso não importa muito pois é uma questão de referencial. O que importa é entender plenamente o que está fazendo com os vetores!

Nossa primeira meta é fazer conversão entre a notação polar e a notação cartesiana. Por que precisamos fazer isso? Equipamentos medem a direção com bússola, e assim, no sistema polar. Computadores só trabalham com sistema cartesiano. Para podermos analisar os dados de correntes medidos no sistema polar, temos que convertê-los para o sistema cartesiano, e temos que fazer isso com 100% de certeza que estamos fazendo direito.

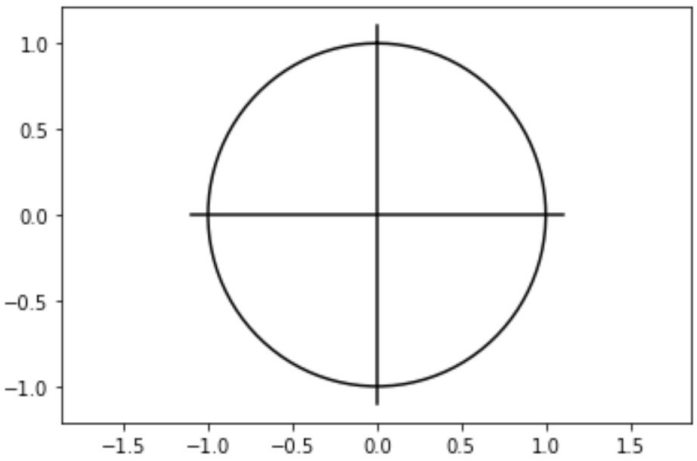
Seria mais didático começar transformando do sistema polar para o cartesiano. Digo, tenho uma corrente de 1,48 m/s com direção de 137º e quero obter as suas componentes cartesianas… mas como vamos usar o Python/Jupyter, temos que começar transformando o sistema cartesiano em sistema polar. Inicialmente vamos desenhar um círculo! Não é possível desenhar um círculo com o Python. O que podemos fazer é desenhar um polígono com muitos lados, tal que pareça um círculo. Sem maiores detalhes… (Código 1). Destrinchem o código… se fizerem os gráficos de ‘circ_x’ e ‘circ_y’ por ‘n’ terão as funções seno e coseno separadamente. Experimentem também diminuir número de segmentos do círculo, que é definido pela função ‘linspace()’, de 100 para 10, ou 4! Como nós vamos usar várias vezes este desenho de círculo e para que o código posterior fique mais limpo, vamos fazer uma função para isto (Código 2).

Rotação de Vetores

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
n = np.linspace(0, 2*np.pi, 100)
circ_x = np.sin(n)
circ_y = np.cos(n)

plt.plot(circ_x, circ_y, 'k')
plt.plot([0, 0], [-1.1, 1.1], 'k')
plt.plot([-1.1, 1.1],[0, 0], 'k')
plt.axis('equal')
plt.show()
```



Código 1: Desenhando um círculo.

```
def desenha_circulo():
    n = np.linspace(0, 2*np.pi, 100)
    circ_x = np.sin(n)
    circ_y = np.cos(n)

    fig, ax = plt.subplots()
    ax.plot(circ_x, circ_y, 'k')
    ax.plot([0, 0], [-1.1, 1.1], 'k')
    ax.plot([-1.1, 1.1],[0, 0], 'k')
    ax.axis('equal')
    return fig
```

Código 2: Função para desenhar um círculo.

Agora temos uma ferramenta para desenhar um círculo de raio 1. Para colocarmos um ponto qualquer neste desenho, basta definirmos os valores das coordenadas horizontal e vertical. Digamos P(0.6, -0.4)! Nesta notação, o primeiro valor representa a coordenada horizontal, e o segundo a vertical. Em Pythonês, fica (Código 3). A variável 'ponto' é um objeto lista com dois elementos. Podemos adicionar outros pontos, adicionando listas à 'ponto' , como

```
ponto = [[.6, -.4], [-.3, .8]]
```

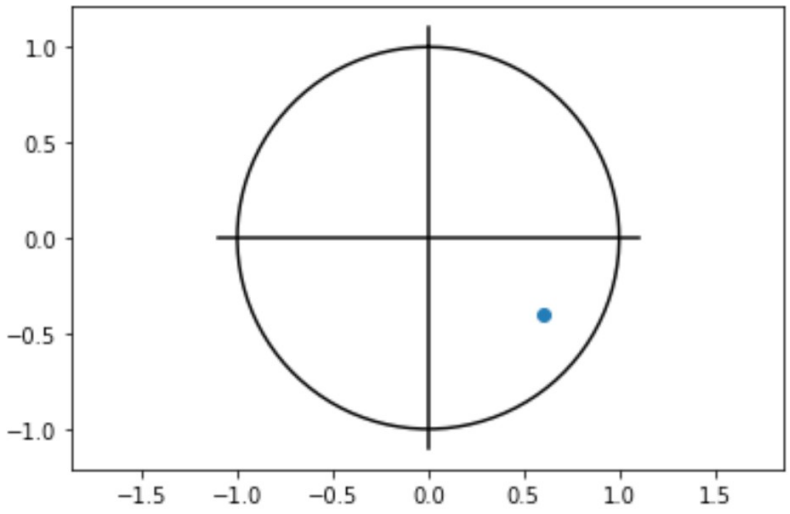
ou podemos fazer listas separadas para as coordendas, como

```
pontos_x = [.6, -.3]
pontos_y = [-.4, .8]
```

tendo a atenção de mudar as variáveis quando fizer o gráfico! Geralmente os dados vem desta forma! Uma tabela onde uma coluna estão os valore de 'x', e outra com os valores de 'y'.

```
ponto = [.6, -.4]

desenha_circulo();
plt.plot(ponto[0], ponto[1], 'o')
plt.show()
```



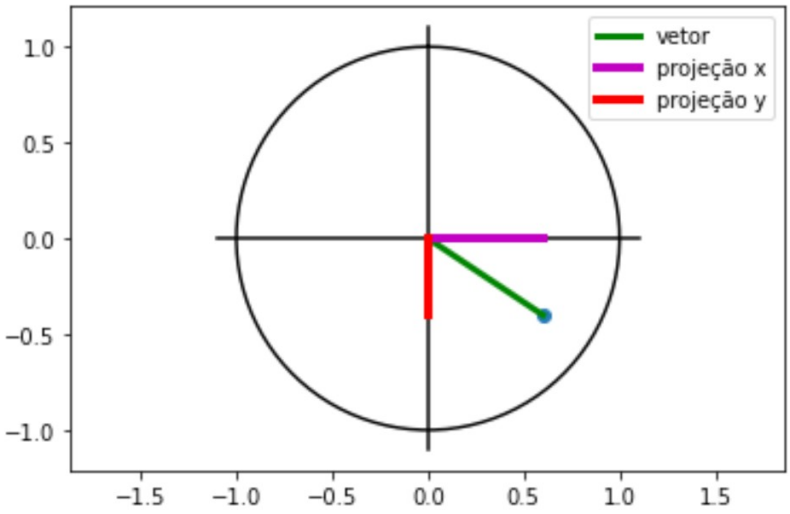
Código 3: Colocando um ponto no círculo.

Para ‘enxergarmos’ este ponto como um vetor, ele tem que primeiro virar uma ‘linha’, e para termos uma linha temos que ter um segundo ponto, que será a origem. E, ligando a origem ao primeiro valor do ponto teremos a projeção no eixo x, e ligando a origem ao segundo valor do ponto teremos a projeção no eixo y (Código 4).

```
valor_x = .6
valor_y = -.4

linha_x = [0, valor_x]
linha_y = [0, valor_y]
origem = [0, 0]

desenha_circulo();
plt.plot(valor_x, valor_y, 'o')
plt.plot(linha_x, linha_y, 'g', linewidth=3, label='vetor')
plt.plot(linha_x, origem, 'm', linewidth=4, label='projeção x')
plt.plot(origem, linha_y, 'r', linewidth=4, label='projeção y')
plt.legend()
plt.show()
```



Código 4: Desenhando o vetor e suas projeções.

Retornando ao raciocínio, queremos converter o vetor (ponto) p(0.6, -0.4) em velocidade e direção p(r, θ). Como vamos trabalhar com dados de correntes, vamos usar o referencial geográfico. Incluam estas linhas da função que desenha o círculo!

```
ax.text(0, 1.15, 'N', ha='center')
ax.text(0, -1.15, 'S', ha='center', va='top')
ax.text(1.15, 0 , 'E', va='center')
ax.text(-1.15, 0, 'O', ha='right', va='center')
ax.set_ylim(-1.3, 1.3)
```

e teremos a Figura 2.

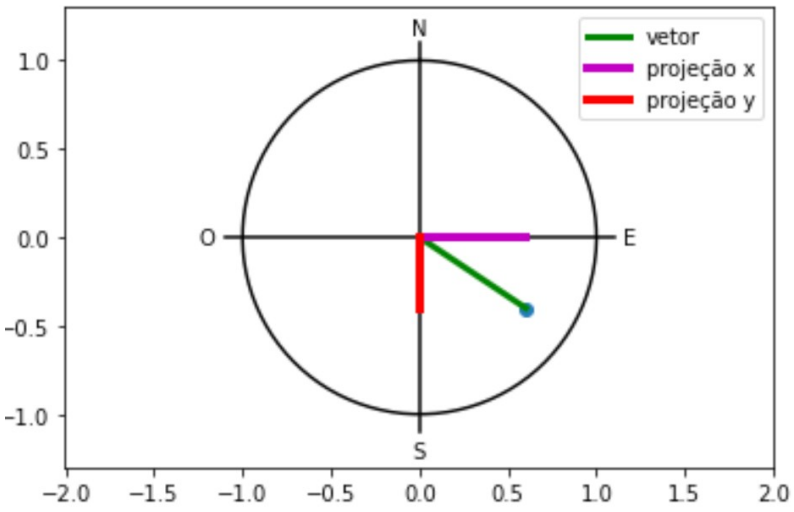


Figura 2: Círculo com referência geográfica.

A magnitude do vetor (r) é a hipotenusa. As suas projeções sobre os eixos são os catetos (Figura). Então, aplicação direta de Pitágoras obtemos r (ou, a velocidade). Em inglês, a componente escalar da velocidade é ‘speed’, e as componentes vetoriais (cartesianas) são ‘velocity’. Nós temos que distinguir usando o adjetivo ‘escalar’ ou ‘vetorial’. Pitágoras no Python (Código 5).

```
velocidade = (valor_x**2 + valor_y**2)**.5
print(np.round(velocidade,2))
```

```
0.72
```

Código 5: Aplicando Pitágoras.

Para achar a direção, é um pouco mais complicado. A direção é o ângulo formado entre o norte e o ventor (linha verde), no sentido horário, e sabemos que é algo entre 90º e 180º. Na caixa de utilidades trigonométricas, se temos os catetos, podemos obter um ângulo usando a tangente,

$$\tan\theta = \frac{cateto\ oposto}{cateto\ adjacente}$$

e temos duas opções para θ: o ângulo formado entre o vetor (verde) e a projeção x (magenta) ou o vetor (verde) e a projeção y (vermelho). Como são ângulos complementares, conhecendo um, sabe-se o outro. E no final das contas, tanto faz, pois depois teremos que dar um jeito de acertar o ângulo pois os limites da função tangente é entre -90° e 90°. Como a razão entre os catetos fornecerá a tangente de θ, a função arco-tangente dará o ângulo, obviamente (!) em radianos. Em Pythonês (Código 6).

```
theta = np.arctan(valor_y/valor_x) *180/np.pi
print(np.round(theta,1))
```

```
-33.7
```

Código 6: Achando o ângulo no quadrante, em graus.

O ângulo é -33.7° porque isso é trigonometria, o zero é no eixo x (E) e aumenta no sentido anti-horário. Para ajudar a entender e visualizar, vamos incluir no código

```
plt.text(-1.9, 1.15, 'Velocidade = '+ str(np.round(velocidade,2)), fontsize=12)
plt.text(-1.9, 0.98, 'Direção = '+ str(np.round(theta,1)) +'$^o$', fontsize=12 )
```

antes do 'plt.show()' para apresentar os resultados. Agora temos uma ferramenta que nos ajuda a explorar o processo de conversão. Mudando os valores de entrada (valor_x e valor_y), vemos o resultado gráfico do vetor no círculo e a magnitude e direção da velocidade. Atente-se para dar valores menores do que 1, pois estamos nosso círculo tem raio = 1. Colocar algum valor zero dará problema na divisão do arcotangente, mas 0.00001 é praticamente zero e funcionará. Colocando valores iguais, o ângulo resultante será sempre 45° (+ ou -). Colocando valores de x pequenos comparados com o de y (0.1 e 0.5)(+ ou -) o vetor ficará próximo do eixo Y, e assim por diante.

Agora que você já brincou com os vetores, e viu como o ângulo varia dando-se várias opções para as componentes cartesianas, você deve tentar estabelecer a regra para converter o ângulo dado no ângulo 'geográfico'. Dependendo do quadrante do vetor, podemos transformar o ângulo para o valor desejado! Dados os valores das componentes, como podemos saber em que quadrante está o vetor? (tente resolver antes de continuar …)

A resposta é que podemos fazer um teste! Se ambas as componentes são positivas, estamos do 1.o quadrante e o ângulo dever variar entre 0 e 90° no sentido horário… e assim por diante. Então, como transformar o ângulo para que ele tenha o valor correto na referência geográfica? (tente resolver antes de continuar…) Espero que tenha chego na solução. É apenas

um quebra-cabeça lógico, e se resolveu corretamente, se colocar valores como 0.1 e 0.5, o ângulo será 11.3º, o que é coerente pois está muito perto do eixo Y. E se colocar -0.1 e -0.5, dará 191.3°. Teste várias opções para ter certeza que funciona.

Uma vez 'dominado' o assunto da conversão do sistema cartesiano para polar, podemos converter essa transformação em uma função que chamaremos 'cart2polar'… 2 = to (Inglês)! E agora, o caminho contrário… transformar velocidade e direção nas componentes cartesianas! Isso é um problema relativamente simples. (sério… tente fazer sozinho antes de continuar!). E, assim, criamos a função 'polar2cart'. Essas duas funções (Código 7) são ferramentas muito utilizadas em qualquer análise de dados que envolvam vetores, e é interessante torná-las funções em um pacote que possa ser carregado nos notebooks. Ou, até que aprenda fazer isso, dá para copiar e colar de um notebook para outro.

```
:
def cart2polar(x, y):
    velocidade = (x**2 + y**2)**.5
    theta = np.arctan(y/x) *180/np.pi
    if x > 0:
        theta = 90 - theta
    else:
        theta = 270 - theta
    return velocidade, theta

def polar2cart(vel, direcao):
    u = vel * np.sin(direcao*np.pi/180)
    v = vel * np.cos(direcao*np.pi/180)
    return u, v
```

Código 7: funções cart2polar a polar2cat.

Agora podemos passar ao assunto de rotação de vetores. Consideremos o problema da correção da declinação magnética. Instrumentos (e.g., ADCP) registram a direção em relação norte magnético, e, se necessário for, os dados precisam ser corrigidos para que a direção seja o norte geográfico. Para saber a declinação magnética para onde um equipamento registrou dados, consulte https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml. Ou, verifique na carta náutica e faça as contas! Digamos que o valor da declinação seja -16.8°. Isso significa que o norte magnético está à esquerda no norte geográfico. Então, se um instrumento forneceu a direção de 77°, este ângulo é em relação norte magnético, e basta diminuir o valor da declinação para obter o ângulo em relação ao norte geográfico = 60.2°. Simples assim. O cuidado que se deve ter no caso de uma série temporal com muitos valores (está processando no Python, certo?), pode ter valores de direção entre 0 e 16,8° que após a correção ficarão negativos. Isto não é de fato um problema, mas se for fazer um gráfico com a direção, fica estranho ter valores negativos. Então, basta achar estes valores e somar 360° (for & if).

No caso de que os dados estão em componentes cartesianas (vamos usar a denominação de u e v para as componentes cartesianas da velocidade nos eixos x e y, respectivamente), uma solução é convertê-los para o sistema polar, e então fazer a correção da declinação magnética. Mas há uma maneira mais elegante de fazer isso. Assista ao vídeo https://www.youtube.com/watch?v=OYuoPTRVzxY. Ou seja, considerando que podemos considerar as componentes cartesianas dos vetores em uma notação matricial, a rotação passa a ser uma questão de rotação matricial. Embora o nome pareça pomposo, é aritmética de matrizes

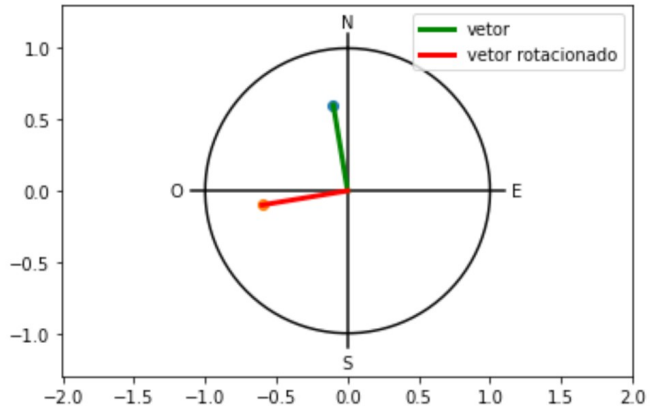
e trigonometria, como o vídeo demonstra muito claramente. No nosso caso é importante lembrar que a rotação deve ser no sentido horário, então basta considerar um ângulo de rotação negativo quando usar a equação da rotação. Será que funciona? Podemos reformar nossa ferramenta de visualização de vetores para confirmar (Código 8).

```
def rotaciona_vetor(u, v, theta):
    theta_rad = -theta*np.pi/180
    ur = u * np.cos(theta_rad) - v * np.sin(theta_rad)
    vr = u * np.sin(theta_rad) + v * np.cos(theta_rad)
    return ur, vr
```

```
valor_x = -.1
valor_y = .6
valor_xr, valor_yr = rotaciona_vetor(valor_x, valor_y, 270)

linha_x = [0, valor_x]
linha_y = [0, valor_y]
linha_x2 = [0, valor_xr]
linha_y2 = [0, valor_yr]

desenha_circulo();
plt.plot(valor_x, valor_y, 'o')
plt.plot(linha_x, linha_y, 'g', linewidth=3, label='vetor')
plt.plot(valor_xr, valor_yr, 'o')
plt.plot(linha_x2, linha_y2, 'r', linewidth=3, label='vetor rotacionado')
plt.legend()
plt.show()
```



Código 8: Rotação matricial com vetores, e visualização.

Algumas vezes precisamos rotacionar os dados de correntes ou vento para maximizar a variância dos dados sobre um dos eixos cartesianos. Por exemplo, a variância (ou energia!) das correntes em um estuário com geometria canalizada (muito longo e estreito) será principalmente no seu eixo longitudinal. O 'eixo' não é uma linha reta, mas algo que acompanha a curvatura do sistema, que ocasionalmente pode até ficar coincidente com os eixos N-S ou L-O (Figura 3). As correntes alinhadas ao eixo são ditas correntes longitudinais, e as correntes perpendiculares ao eixo serão as correntes transversais (ou circulação secundária).

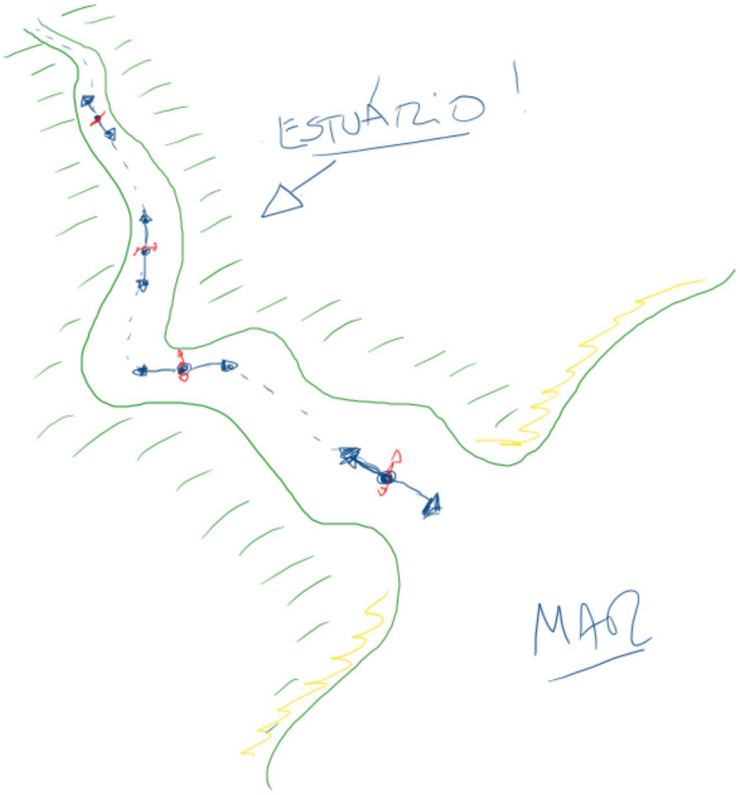


Figura 3: Esquema de um estuário e as correntes longitudinais.

Neste caso, a notação polar é de muito pouco ajuda e precisamos trabalhar com os dados na sua forma cartesiana u e v, pois, rotacionando os dados para sua componente longitudinal (principal) para o local onde os dados foram obtidos, os valores de correntes assumirão valores positivos e negativos, que, por convenção, poderão representar enchente ou vazante, ou vice-versa caso quem esteja analisando os dados assim o desejar! Para isso vamos precisar de dados observados para poder demonstrar isto mais claramente.

O arquivo 'Tutorial_vetores_dados_S4.pkl' é um arquivo binário gerado em outro notebook e salvo com o pacote 'pickle' (`import pickle`)(Código 9). Este arquivo contém uma lista com quatro elementos, sendo que cada elemento é uma lista que contém os registros de (1) tempo, já no formato 'datetime', (2) nível da água, (3) componente u e (4) componente v da velocidade. Aqui o nível é referenciado pelo nível médio, pela redução da média dos dados, e as componentes da velocidade são converticas para m/s. Então, sempre, a primeira coisa é dar uma olhada nos dados (Figura 4).

Estes dados foram coletados durante um experimento de campo na Baia de Vitória, ES (Figura 5). Foi utilizado um correntógrafo eletro-magnético da marca InterOcean modelo S4. O instrumento foi instalado suspenso a 3 m abaixo da superfície, o que é aproximadamente a meia água (Figura 6). O instrumento registrou dados em intervalos de 5 minutos, e cada registro é uma média da medição durante 120 s a uma taxa amostral de 2 Hz (= 240 amostras, 'burst'), para um período amostral de 25 horas. Os períodos de integração (120 s) são muito importantes no registro de dados de correntes para remover oscilações de alta frequência gerada por ondas.

Como é notável na Figura 4, o experimento capturou dois ciclos completos de maré semi-diurna (12:25 horas), e as componentes da corrente variam de acordo, embora com um padrão mais ruidoso.

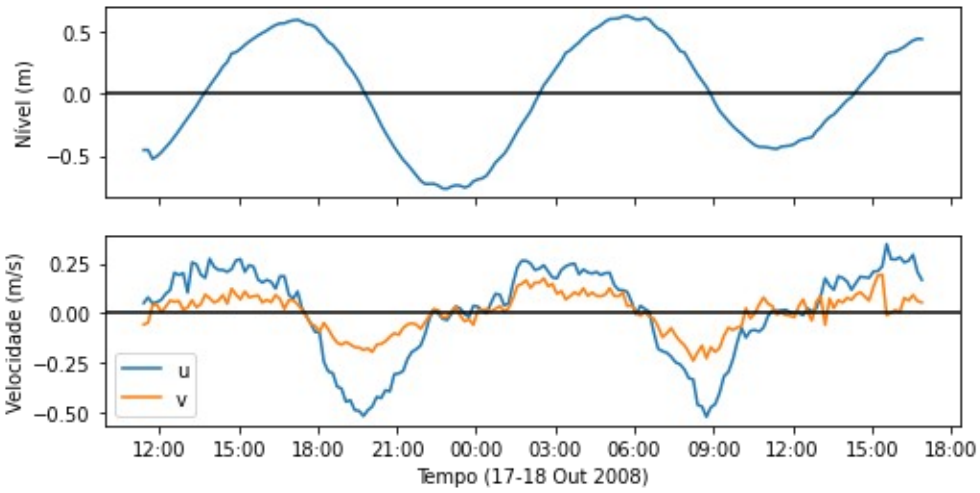


Figura 4: Inspecionando os dados carregados.

```
s4_file = 'Tutorial_vetores_dados_S4.pkl'

with open(s4_file, 'rb') as src:
    s4 = pickle.load(src)

tempo = s4[0]
nivel = s4[1] - np.mean(s4[1])
u = s4[2]/100
v = s4[3]/100
```

Código 9: Carregando dados pré-processados com 'pickle' e transformando.

O S4 tem um sensor de pressão, e obviamente neste caso ele vai fornecer um valor da distância entre o instrumento e a superfície, que é constante (~), uma vez que estava pendurado em uma bóia! Os dados de nível da água foram registrados por um marégrafo que foi instalado na margem próxima do fundeio, registrando dados de maneira similar (intervalo amostral de 10 min, burts de 120 s). Este experimento foi para avaliar a hidrodinâmica da baía durante condições de sizígia. Havia outros instrumentos em outros locais também. Essa contextualização é relevante para dizer que estes dados foram coletados criteriosamente!

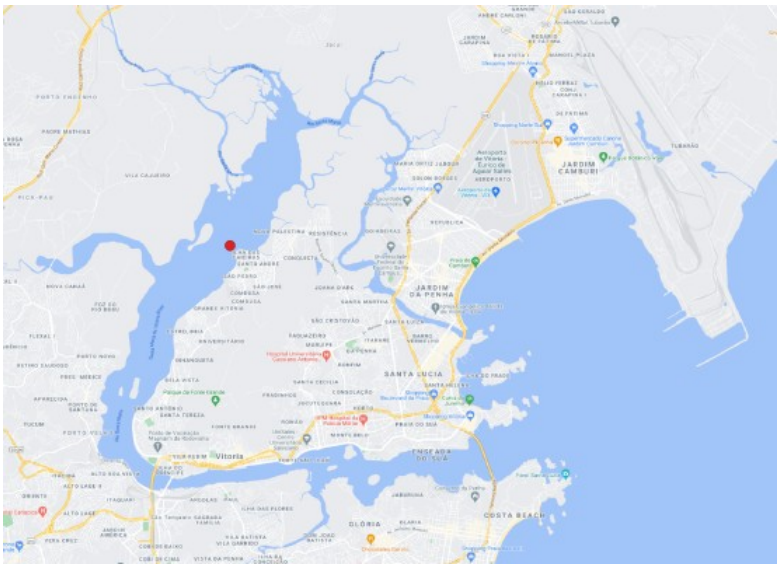


Figura 5: Localização de onde foram gerados os dados de correntes e nível da água.

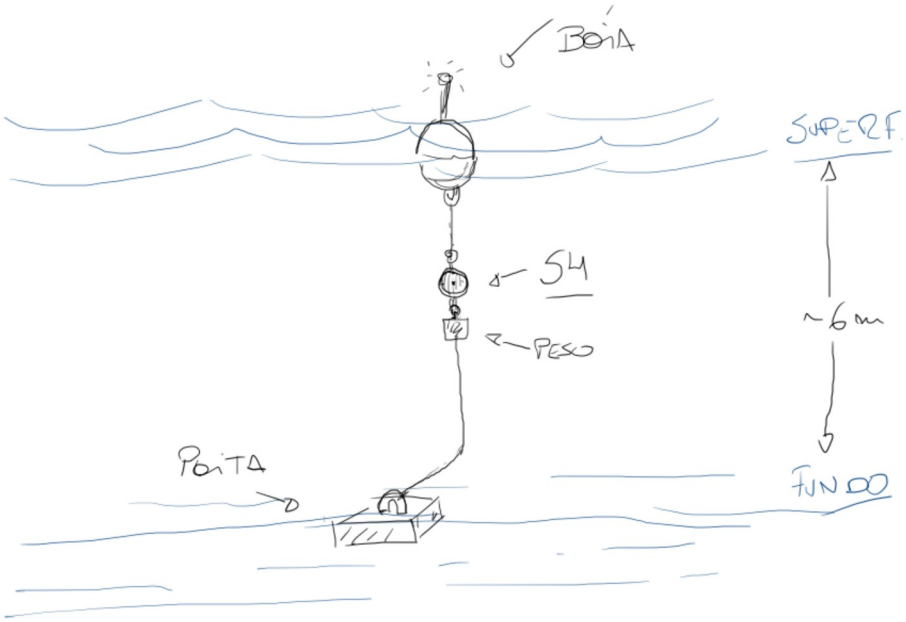


Figura 6: Esquema do fundeio do S4.

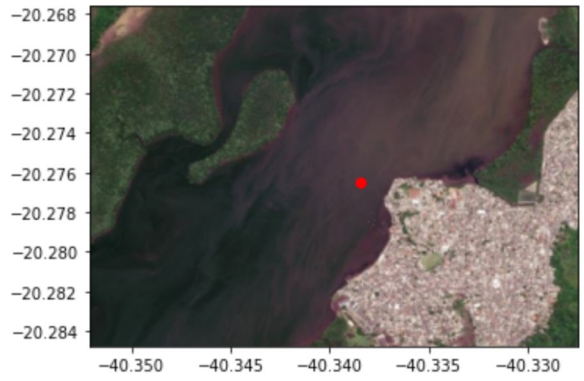
Além dos dados de corrente, nesse tutorial acompanha também uma imagem no formato '.tiff' georeferenciada, que é um 'geotiff'. Essa imagem foi gerada usando o QGIS, e pega a região do entorno do fundeio. Uma maneira de carregar e visualizar a imagem no notebook é usando o pacote 'rasterio' (`import rasterio` `from rasterio.plot import show`). Caso tenha problemas na instalação do pacote, ignore a parte de carregar e visualizar a imagem, mas considere isso como um desafio e uma coisa muito útil! Se não com esse pacote, com outro. Então… Código 10.

```
img_file = 'Fundeio_BaiaVitoria_geo.tiff'
img = rasterio.open(img_file)

# posição fundeio
tx = -40.338476
ty = -20.276495

fig, ax = plt.subplots()
ax.plot(tx, ty, 'or')
show(img, ax=ax)
```

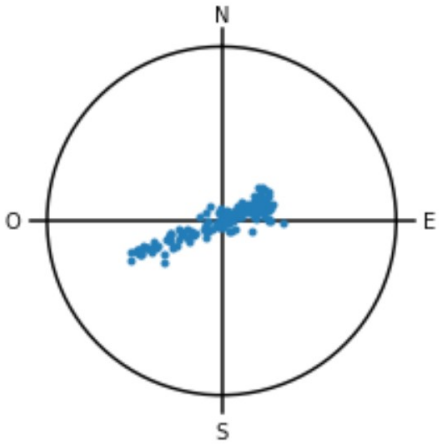
```
<AxesSubplot:>
```



Código 10: Carregando a mostrando a imagem com o ponto de fundeio do S4.

A imagem da área georeferenciada será importante para passar a noção exata do que significa a rotação de vetores. Podemos *plotar* os dados de correntes (u, v) sobre a imagem para ver como se distribuem! A imagem está referenciada em coordenadas geográficas (longitude e latitude) em graus e décimos de graus, e as componentes da velocidade estão em m/s, mas, no fim, são apenas números e para o Python pouco importa! Aqui vem a importância de entender bem a questão de pontos, linhas e vetores, minunciosamente explicados acima. As componentes da velocidade tem a origem no zero dos eixos. Podemos então deslocar a origem e colocá-la na mesma posição do fundeio. E, aplicando uma correção linear, ou fator de escala, podemos aumentar ou diminuir o tamanho dos vetores V(u, v) para que fiquem visualmente proporcionais na imagem. Da mesma forma que podemos fazer uma visualização dos vetores/pontos sobre o nosso círculo de referência geográfica (Código 11), podemos fazer o mesmo sobre a imagem (Código 12).

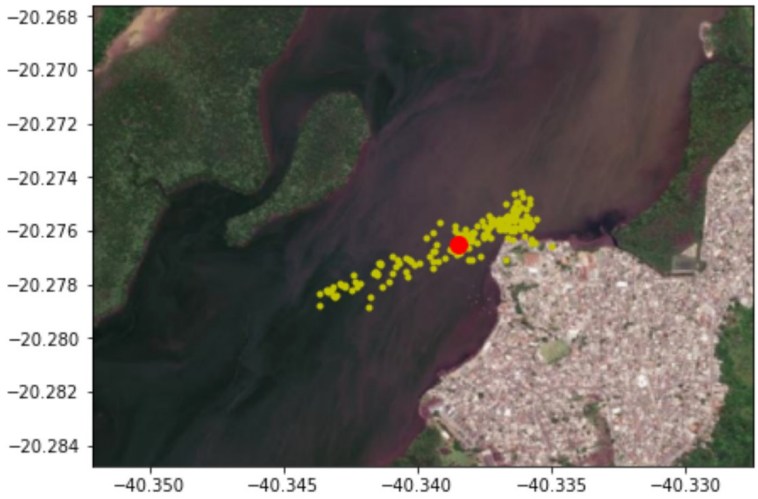
```
desenha_circulo();
plt.plot(u, v, '.')
plt.axis('off')
plt.show()
```



Código 11: Visualizando os dados no círculo de referência geográfica.

```
escala = .01
u2 = u * escala + tx
v2 = v * escala + ty

fig, ax = plt.subplots(figsize=(7,7))
show(img, ax=ax)
ax.plot(u2, v2, 'y.')
ax.plot(tx, ty, 'ro', markersize=10)
plt.show()
```



Código 12: Visualizando os vetores na imagem. O ponto vermelho é o local do fundeio.

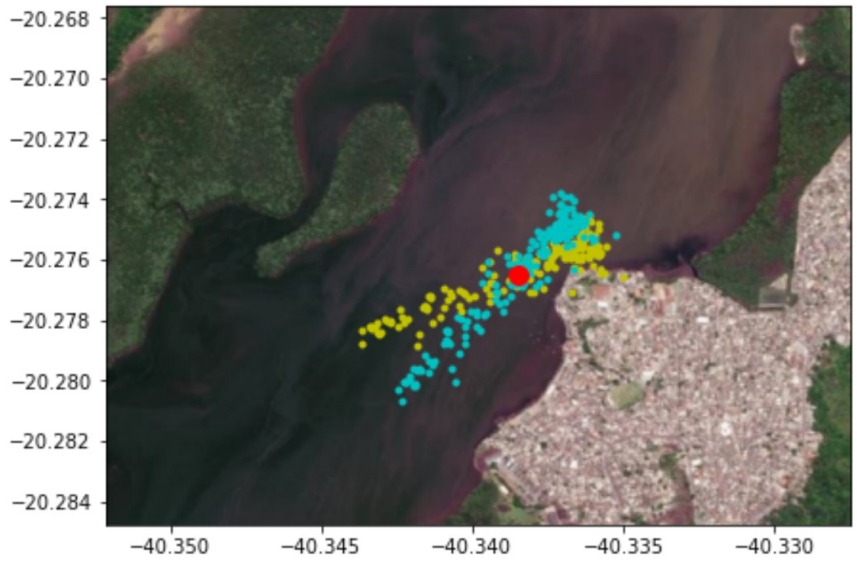
Analisando a figura do Código 12, para olhos experientes esse padrão parece anômalo no sentido que o alinhamento das correntes não está paralelo (grosseiramente falando) com as margens (pode explorar mais isto colocando o ponto no Google Earth para verificar!). Uma das primeiras perguntas que fazemos quando pegamos dados de correntes é se os dados foram corrigidos para a declinação magnética. Nesses caso eles não foram. A declinação magnética para essa posição e para esse ano é aproximadamente -24°. Podemos usar a função que criamos

para rotacionar vetores e ver o que acontece (Código 13). Agora o alinhamento dos vetores está muito mais coerente com a geometria local.

```
ur = []
vr = []
for i in range(len(u2)):
    cu, cv = rotaciona_vetor(u[i], v[i], -24)
    ur.append(cu)
    vr.append(cv)

escala = .01
u2 = np.array(u) * escala + tx
v2 = np.array(v) * escala + ty
ur2 = np.array(ur) * escala + tx
vr2 = np.array(vr) * escala + ty

fig, ax = plt.subplots(figsize=(7,7))
show(img, ax=ax)
ax.plot(u2, v2, 'y.')
ax.plot(ur2, vr2, 'c.')
ax.plot(tx, ty, 'ro', markersize=10)
plt.show()
```



Código 13: Agora com a correção da declinação magnética.

A figura resultante do Código 13 demonstra que os vetores ou pontos são absolutamente os mesmos, sendo que a única coisa que mudou foi o sistema referencial à que estão atrelados. Inicialmente, o norte magnético, depois, ao norte geográfico. Agora, ao último tópico deste tutorial, que é a decomposição para a componente principal. Estes dados foram coletados em um estuário, que usualmente pode ser simplificado como um canal: duas bordas & comprimento muito maior que a largura. A água não pode atravessar as bordas, então, quando a maré sobe na plataforma, surge um gradiente de pressão que força a água estuário adentro, e a orientação do deslocamento da água será primeiramente determinado pela orientação do eixo longitudinal do canal. E é exatamente isto que reflete a figura! Ou a água vai para um lado, ou vai para outro. Porém, as observações não caem exatamente em uma linha, ocorrendo uma dispersão de pontos transversal, o que é denominado de efeito da circulação secundária.

Embora a figura do código Código 13 seja interessante e permita visualizar que as correntes para justante (para a boca) são muito mais intensas do que as correntes para montante (para cabeceira), não nos serve para muito mais. Para podermos computar o transporte de qualquer coisa (entre outras coisas interessantes que dá para fazer), precisamos ter a velocidade longitudinal. Para isso, precisamos rotacionar os dados para este novo sistema referencial! Diga-se que nem precisaríamos ter feito a correção da declinação magnética! A partir dos dados brutos, podemos ir direto para este ponto. A declinação é mais usada em sistemas onde o fluxo não é fortemente canalizado, como lagoas ou plataforma continental.

Uma maneira de fazer isso é simplesmente identificar a direção modal, e então fazer a rotação para um dos eixos. Não importa qual, contanto que saiba o que está fazendo! Antes, vamos dar uma olhada na série temporal de u e v após a rotação (Figura 7). Percebam a diferença

com os dados antes de rotacionar. Agora as componentes estão praticamente sobrepostas, com valores praticamente iguais! Vocês porderia sugerir qual será o valor de rotação? Qual o ângulo entre a hipotenusa (magnitude da velocidade) e um cateto (componente da velocidade) quando os dois catetos são iguais?

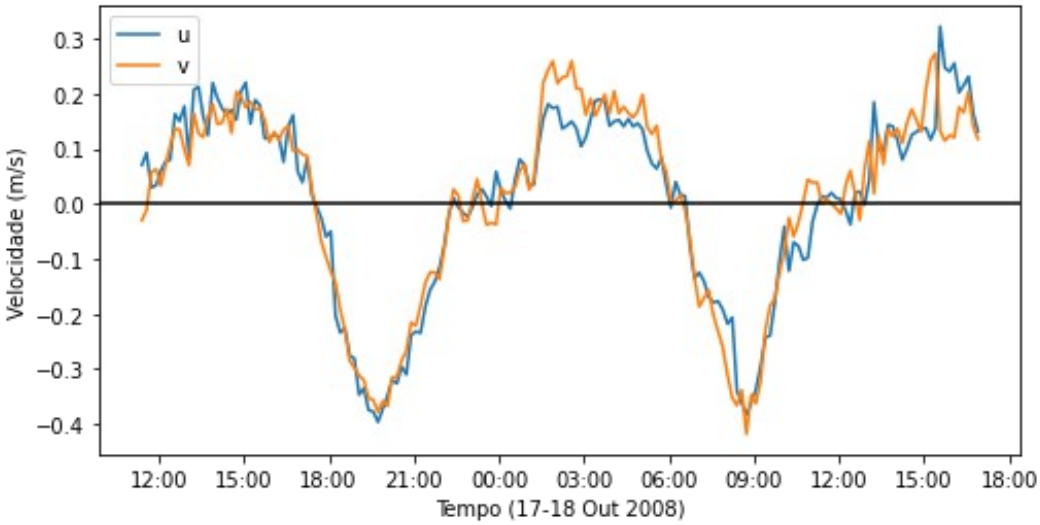


Figura 7: Séries temporais de u e v corrigidos para declinação magnética.

Para achar a moda da direção, temos que ter direção! Para termos a direção, usamos a função que criamos 'cart2polar'… e fazemos um histograma. Poderia ser um gráfico de barras, o que pareceria de fato um histograma, mas aqui é a mesma coisa porém com linha (Figura 8). Como seria esperado, há duas modas. Uma que reflete as correntes de enchente e outra de vazante. E, a maior moda, que será aproximadamente 180° da outra, é, tal como previmos, em torno de 45°.

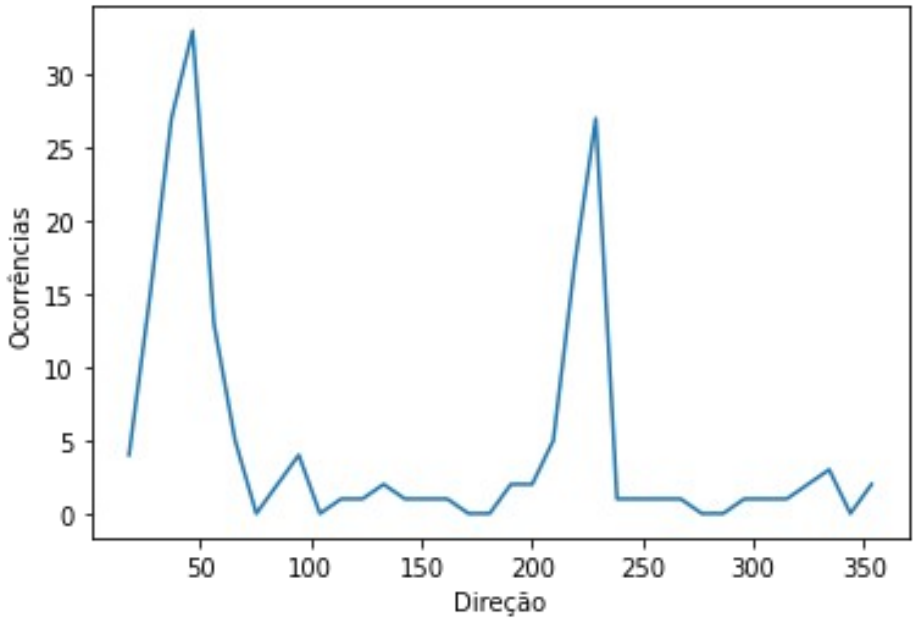


Figura 8: Distribuição de ocorrências de direção da corrente por classe de direção.

Se rotacionarmos os dados em -47°, alinharemos os dados ao eixo y (norte-sul), e se somarmos 43° alinharemos os dados ao eixo x (leste-oeste). Para fazer isso podemos fazer direto sobre o arranjo de dados de direção e depois usar a função 'polar2cart' ou fazer diretamente usando a funçào 'rotaciona_vetor', o resultado será o mesmo. O resultado da rotação no cículo fica como na Figura 9, e as séries temporais como na Figura 10. Agora a variância está concentrada na linha laranja, que então atribuímos à componentes longitudinal das correntes. Neste caso, valores positivos (para o norte no sistema referencial rotacionado) indicam correntes entrando no estuário (enchente), enquanto que valores negativos indicam correntes saindo do

estuário. Mas essa notação é arbitrária, e se o contrário for preferível, basta multiplicar tudo por -1.

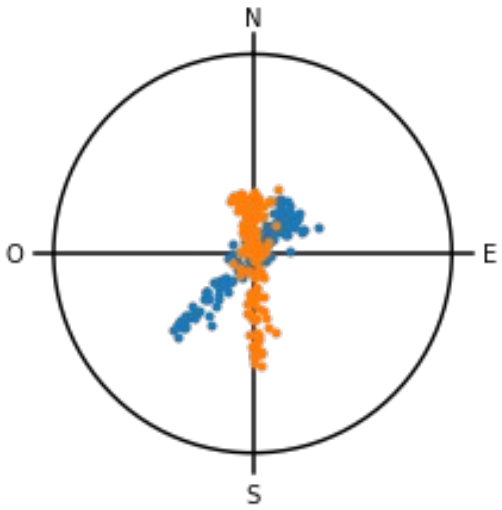


Figura 9: Rotação dos dados para o eixo y.

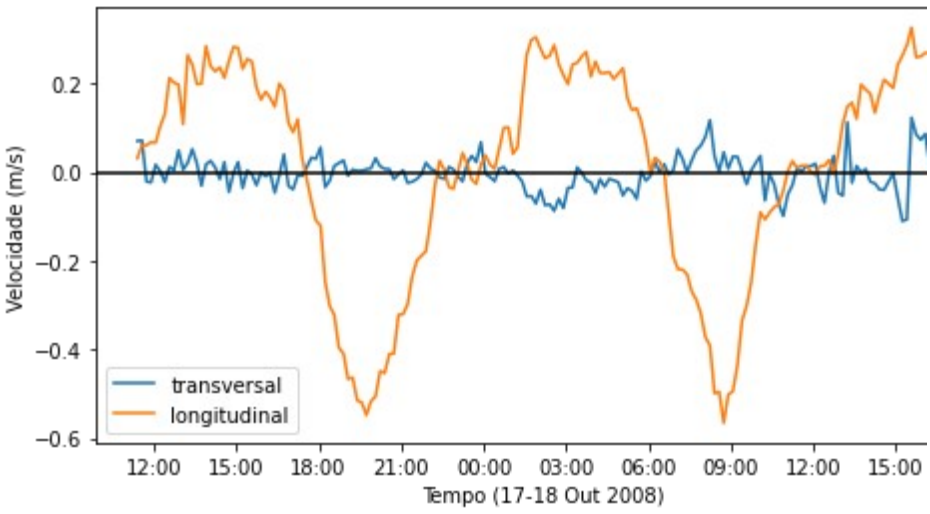


Figura 10: Séries temporais dos dados rotacionados para o eixo y.

E quase como sempre, há mais de um jeito da fazermos isso. Uma solução mais elegante é através da análise de componentes principais, usualmente chamada de PCA do acrônimo em Inglês. PCA é uma técnica que enquadramos como multivariada, que de certa forma é a mesma coisa que EOF (empirical orthogonal functions) ou SVD (single value decomposition). Estas análises, antes de tudo, são análises para ajudar a interpretar a variância (e co-variância) dos dados, uma vez que estejam arranjados em matrizes. Como vimos acima, vetores podem ser considerados matrizes 1 x 2, e uma série temporal de vetores pode ser arranjado como uma matriz N x 2, sendo N o número de observações.

Embora os termos matrizes, co-variância e multivariada possam soar complexos (assistam aos vídeos do 3Blue1Brow, entre outros!!), não são. Não que sejam simples, de fato, como aplicação de Pitágoras, mas não são magia negra onde se coloca seus dados de um lado e magicamente as componentes principais saem do outro! O princípio básico é determinar os autovalores e autovetores da matriz de covariância dos dados. Os autovetores fornecerão os eixos perpendiculares sobre os quais os dados se ajustam, enquanto que os autovalores fornecerão a proporção da variança entre os eixos. No presente caso trata-se apenas de rotação dos dados que pode ser obtida pelo produto entre matricial entre os dados e os autovetores.

Fazer a PCA requer apenas 6 linhas de código (Código 14). Montamos a matriz de dados onde as linhas são as variáveis (u e v) e as colunas são as amostras temporais. Geramos a matriz de covariancia, os autovalores e autovetores e calculamos o percentual da variância explicada por cada componentes principal. Pronto! Agora podemos gerar a transformação linear dos

vetores/pontos usando o produto matricial entre cada dados com os autovetores (Código 15). A Figura 11 mostra o resultado comparando o método ‘tosco’ do método ‘elegante’. Um está ao contrário do outro, tanto pra quem é a componente longitudinal e transversal (cores) quanto pela variação (positivo/negativo). Mas isso não importa muito pois sabendo o que foi feito, basta pegar a componente de maior variancia pois está é a que representa as correntes longitudinais. E positivo e negativo é arbitrário.

Análise de Componentes Principais

```
matriz = np.vstack((ur, vr))

cov_matriz = np.cov(matriz)

auto_valores, auto_vetores = np.linalg.eig(cov_matriz)

variancia_explicada = []
for i in auto_valores:
    variancia_explicada.append((i/sum(auto_valores))*100)

print('formato da matriz de dados =', matriz.shape)
print('variância explicada (%) =', np.round(variancia_explicada, 1))
print('autovalores = ', np.round(auto_valores, 3))
print('autovetores = ', np.round(auto_vetores, 3))
print('matriz de covariancia = ', np.round(cov_matriz, 3))
```

```
formato da matriz de dados = (2, 177)
variância explicada (%) = [ 2.4 97.6]
autovalores =  [0.001 0.058]
autovetores =  [[-0.714 -0.7  ]
 [ 0.7   -0.714]]
matriz de covariancia =  [[0.029 0.028]
 [0.028 0.03 ]]
```

Código 14: Fazendo a PCA com os dados de correntes.

Redução dimensional

https://www.youtube.com/watch?v=ZqXnPcyIAL8

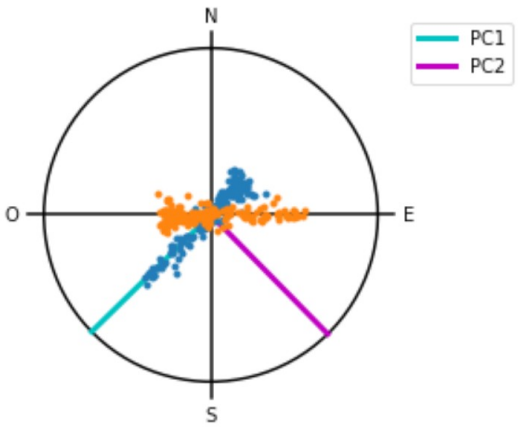
```
a = auto_vetores[0,0]
b = auto_vetores[0,1]

c = auto_vetores[1,0]
d = auto_vetores[1,1]

ur3 = a*ur + b*vr
vr3 = c*ur + d*vr

desenha_circulo();
plt.plot([0, auto_vetores[0,0]], [0, auto_vetores[0,1]], 'c', linewidth=3, label='PC1')
plt.plot([0, auto_vetores[1,0]], [0, auto_vetores[1,1]], 'm', linewidth=3, label='PC2')

plt.plot(ur, vr, '.')
plt.plot(ur3, vr3, '.')
plt.axis('equal')
plt.axis('off')
plt.legend()
plt.show()
```



Código 15: Redução dimensional para o eixo X.

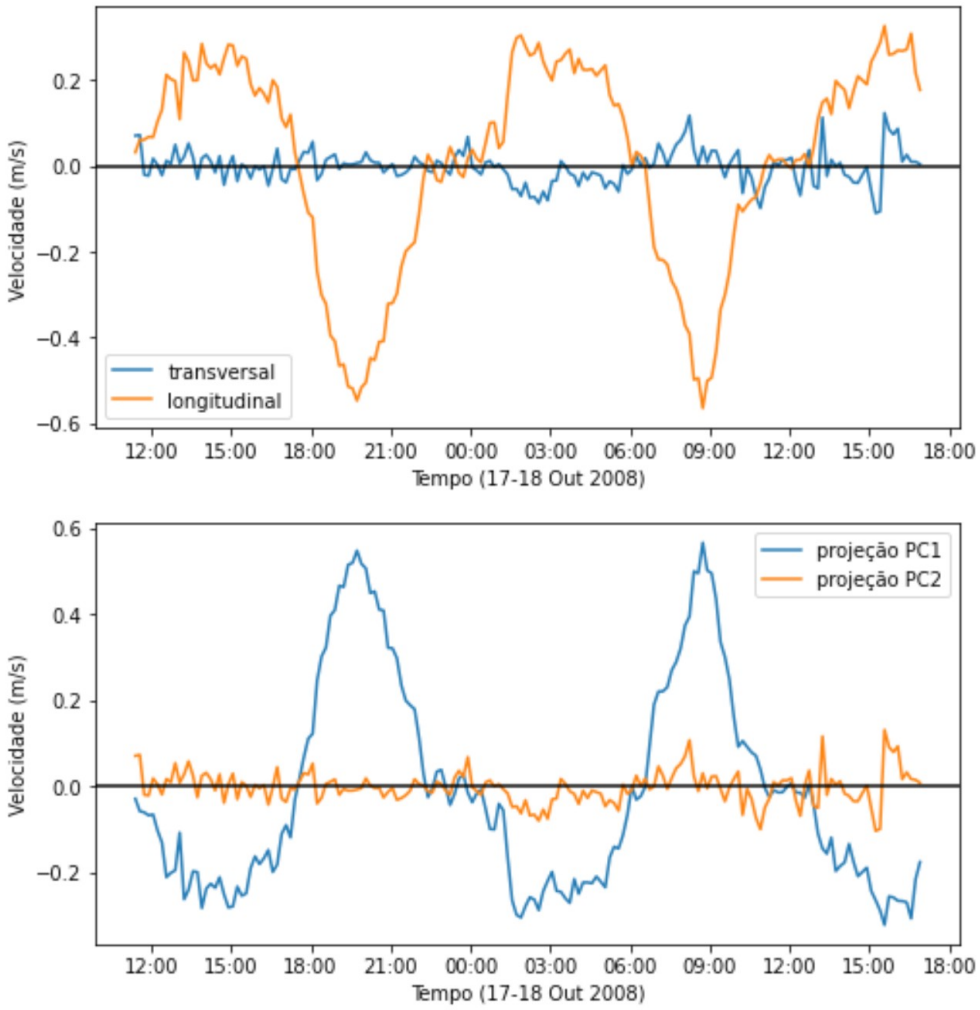


Figura 11: Comparando os resultados das duas maneiras de fazer a decoposição para o eixo longitudinal.

Bônus!

Uma vez que temos a corrente longitudinal, definimos que enchente é positivo, e podemos fazer um diagrama de estágio de maré. Esse diagrama permite avaliar o comportamento da onda de maré onde foram coletados os dados. A relação nível/velocidade para uma onda estacionária é um círculo, pois a velocidade tem um atraso de fase de 90° em relação ao nível (como seno e coseno). A relação para uma onda totalmente progressiva nível e velocidade estão em fase, e o diagrama é uma linha diagonal. Uma onda mixta resultará em uma elípse inclinada. O que temos aqui (Figura 12) é uma coisa meio torta mas que lembra mais um círculo do que uma elipse, e então podemos dizer que a onda é estacionária. O que faz sentido já que estamos quase na cabeceira da baía. A deformação em relação ao círculo se deve principalmente à morfologia e hipsometria do sistema, o que produz correntes de vazante muito mais intensas do que as correntes de enchente (é um sistema dominado pela vazante).

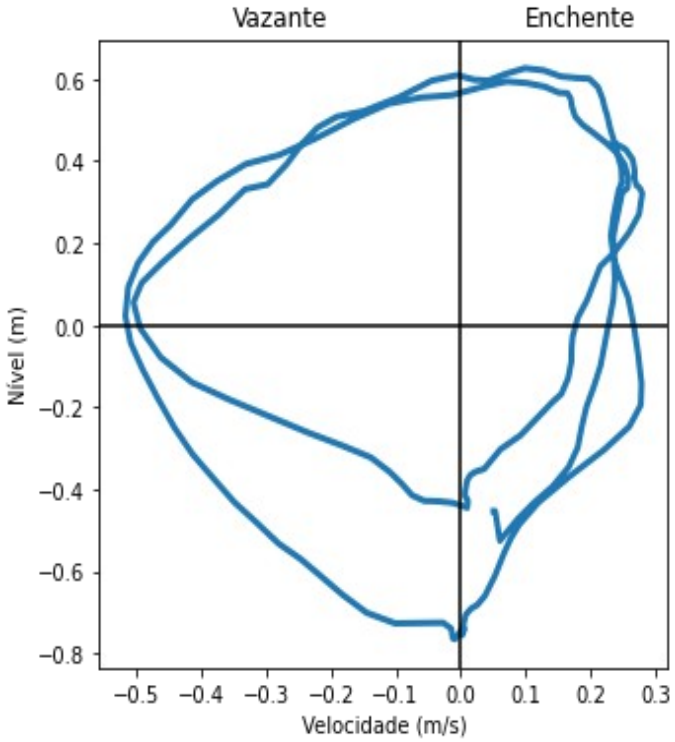


Figura 12: Diagrama de estágio de maré.