ÁLGEBRA LINEAR ALGORÍTMICA–PLE–LABORATÓRIO 1

1. Uma das vantagens de começar estudando álgebra linear no plano é que podemos desenhar tudo o que fizermos. Para isso precisamos carregar o pacote de desenhos do MAXIMA usando

```
load("draw");
```

A função que faz o desenho pode ser usada em duas formas diferentes: `draw2d` ou `wxdraw2d`. Se você usar a primeira forma, a figura aparece como parte da própria seção do MAXIMA; se você usar `draw2d`, a figura aparece em uma janela separada.

⚠ Dependendo da versão do MAXIMA e de onde foi instalado, a função `wxdraw2d` pode não funcionar. Além disso, ela não é adequada ao uso em animações, porque cada quadro vai aparecer em uma janela diferente.

Para definir um vetor de maneira que a função `draw2d` ou `wxdraw2d` consiga desenhá-lo usamos `vector(p1,p2)`, em que `p1` é o ponto em que o vetor começa e `p2` é o ponto onde termina. Como não estamos permitindo que os vetores sejam movidos para fora da origem, tomaremos sempre `p1` como sendo `[0,0]`. Prefiro atribuir `vector(p1,p2)` a uma variável porque assim fica mais fácil desenhar vários vetores ao mesmo tempo. Por exemplo, desenho os vetores $(2,3)$ e $(-2,1)$ fazendo:

```
v1:vector([0,0],[2,3]);
v2:vector([0,0],[-2,1]);
wxdraw2d([v1,v2]);
```

Se você tentou os passos acima, imagino que ficou totalmente decepcionado com o resultados. Podemos obter um resultado melhor ajustando o tamanho (`head_length`) e o ângulo (`head_angle`) da seta do vetor. Com as escolhas abaixo a figura fica bem melhor no meu computador, mas se você está usando o celular talvez precise escolher outros valores:

```
wxdraw2d(head_length=0.1,head_angle = 15, v1,v2);
```

2. Para nossa próxima atividade desenharemos duas bases ortonormais distintas do plano:

$$\varepsilon = \{(1,0),(0,1)\} \quad \text{e} \quad \beta = \{u_1, u_2\},$$

em que u_1 é o vetor obtido normalizando $(2,3)$ e u_2 é o vetor obtido normalizando $(-3,2)$. Lembre-se que *normalizar* um vetor significa dividi-lo por sua norma para obter um vetor unitário; isto é, de norma igual a um. O MAXIMA tem a função `uvect` que normaliza um vetor, mas para usá-la você tem que carregar a biblioteca `eigen`:

```
load("eigen");

e1:vector([0,0],[1,0]);

e2:vector([0,0],[0,1]);

u1:[2,3];

u1:uvect(u1);

u2:[-3,2];

u2:uvect(u2);

v1:vector([0,0],u1);

v2:vector([0,0],u2);

lista:[v1,v2,e1,e2];

wxdraw2d(head_length=0.05,head_angle = 15, lista);
```

Note que ajustei o tamanho da ponta da seta! Uma coisa frustrante é que o vetor $(1,0)$ ficou exatamente na borda da figura. Para evitar isto, podemos escolher o tamanho da janela gráfica. Por exemplo,

```
wxdraw2d(xrange=[-1.5,1.5],yrange=[-0.5,1.5], head_length=0.05,head_angle
= 15, lista);
```

3. Em uma de nossas próximas listas, precisaremos desenhar retângulos. A biblioteca `draw` faz isto usando a função `rectangle(p1,p2)`, em que `p1` e `p2` são *dois vértices opostos do retângulo*. Por exemplo, para desenhar o quadrado de lado um:

```
quadrado:rectangle([0,0],[1,1]);

wxdraw2d(xrange=[-1,2],yrange=[-1,2],fill_color=white,quadrado);
```

Note que ajustei a janela gráfica para fazer o quadrado aparecer claramente, caso contrário ele encheria toda a tela, porque o default é a menor janela gráfica em que a figura cabe. Além disso, optei por um quadrado que só tem a borda, se você quiser que o quadrado seja preenchido com outra cor é só trocar a cor no parâmetro `fill_color`.

4. Além da função para desenhar retângulos, o MAXIMA tem uma função que desenha polígonos mais gerais, chamada `polygon`. Diferentemente da função que desenha retângulos, `polygon` tem como entradas duas listas, a primeira contém as abscissas dos pontos do polígono, a segunda as respectivas ordenadas. Há duas coisas que você precisa ter em mente ao montar estas listas:

- as abscissas e ordenadas de cada ponto têm que aparecer na mesma posição nas duas listas;
- as abscissas e ordenadas de dois pontos ligados por um lado têm que aparecer em posições consecutivas nas listas.

Por exemplo, para desenhar o quadrado de vértices em

$$[0,0],[1,0],[0,1] \quad \text{e} \quad [1,1]$$

construímos os vetores de coordenadas:

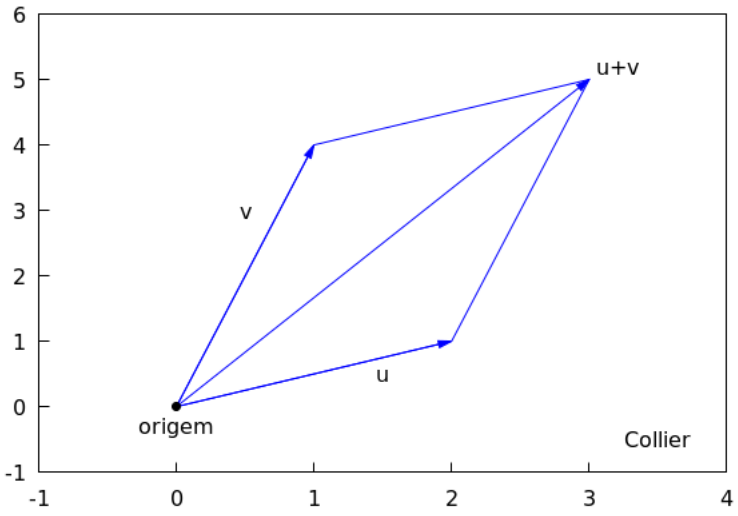
```
px:[0,1,1,0];

py:[0,0,1,1];
```

e em seguida desenhamos usando `wxdraw2d`:

```
wxdraw2d(xrange=[-1,2],yrange=[-1,2],fill_color=white,polygon(px,py));
```

Experimente desenhar o polígono com `px:[0,1,0,1];` e o mesmo `py` e veja o que acontece. Seguindo a sequência de pontos com este novo vetor `px` você rapidamente entenderá porque a figura ficou com essa cara.

5. Para encerrar, faça uma figura com os vetores $u = (2,1)$ e $v = (1,4)$ que ilustre a regra do paralelogramo. A figura deve conter, além dos dois vetores dados, sua soma e o paralelogramo do qual o vetor `u + v` é a diagonal maior. Minha versão da figura aparece abaixo.



Como você pode ver acrescentei, na minha figura, algumas coisas que não foram explicadas acima, como rótulos para os vetores e a origem. Para aprender essas e outras coisas que você pode fazer para tornar sua figura mais interessante, consulte o manual *Graphics with MAXIMA* de Wilhelm Haager que pode ser baixado de vários lugares na web, basta fazer uma busca. Quando sua figura estiver pronta faça upload para a pasta `lab1` no Drive de nossa turma no Google Classroom.