ÁLGEBRA LINEAR ALGORÍTMICA–2020.2–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: `wxdraw2d` ou `draw2d`. 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 `p1+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. Para encerrar vamos aprender a animar uma figura no Maxima usando a função `with_slider_draw`. Por exemplo, a função abaixo desenha uma onda que vai passando e se dissipando. Para vê-la em ação, basta copiar o código em uma célula de input na janela do Maxima e clicar `enter`. A função vai ser processada e você verá uma janela

gráfico sendo aberta abaixo do código. Clique na janela para que a animação seja executada.

```
F(t,x) := exp(-(x-t)^2/0.1)/(t+1);
with_slider_draw(
    t, /* nome da variável utilizada pelo slider */
    makelist(i,i,0,1.5,0.1), /* lista de valores que t deve tomar */
    xrange=[0,1], /* dimensão horizontal da janela do desenho */
    yrange=[0,1], /* dimensão horizontal da janela do desenho */
    explicit(F(t,x), x, 0, 2) /* desenha a função */
)$
```

Note o uso de `makelist`, cuja sintaxe é

$$\texttt{makelist(f(i),i,i}_0\texttt{,i}_1\texttt{,r)}$$

em que `f(i)` é uma função da variável `i`, que tem valor inicial $\texttt{i}_0$, valor final $\texttt{i}_1$ e incremento igual a `r`. Em outras palavras, a lista que é gerada será

$$[\texttt{f(i}_0\texttt{)},\texttt{f(i}_0+\texttt{r)},\texttt{f(i}_0+2\texttt{r)},\ldots,\texttt{f(i}_1\texttt{)}]$$

⚠ Cada linha do código da função `with_slider_draw` tem que acabar com uma vírgula (e não um ponto-e-vírgula!) exceto a que precede o último parêntesis. Isto também vai ocorrer quando definirmos nossas próprias funções no Maxima.

4. A tarefa desta semana é usar a função `with_slider_draw` para criar a animação de um vetor que vai girando, em sentido anti-horário, em torno da origem, até dar uma volta completa. Para isso, abra uma sessão do Maxima e crie uma célula de texto, onde você deve escrever seu nome e DRE. Em seguida, use uma célula de input para escrever o código da função `with_slider_draw` seguindo o modelo acima. Contudo, para esta tarefa, não há necessidade de definir uma função como a $F(t,x)$ do exemplo acima, você pode escrever o vetor a ser desenhado diretamente na última linha de código. Dicas:

1. as opções de ajuste da seta do vetor devem ser colocadas logo acima da última linha, na qual o vetor será definido;
2. use senos e cossenos para criar o vetor de norma 1.

Não se esqueça de comentar o código. Uma vez que o código esteja funcionando, ele deve ser gravado usando o formato:

`nome_DRE_lab01`

que fará o Maxima utilizar a extensão `wxmx`. Seu código deverá então ser entregue usando a atividade correspondente ao Laboratório 1 no Google Classroom.

5. Um desafio maior, valendo um bônus extra na nota, é fazer um relógio com dois ponteiros, um mais longo e um mais curto, sendo que, quando o grande dá uma volta, o pequeno anda apenas o arco equivalente a uma hora e a seguinte. Dessa vez os ponteiros terão que rodar no sentido horário, em vez de anti-horário como na tarefa descrita no item 4. Para posicionar as horas no seu relógio, você pode usar `label`(["`texto`",x_0,y_0]), em que x_0 e y_0 são a abscissa e ordenada do ponto onde a string `texto` deve ser inserida. No caso do relógio as strings são simplesmente os números das horas. Você também pode usar `label` para assinar sua obra no canto direito inferior.