Apostila do Minicurso Transformações Lineares através do Aplicativo SageMath®

Tiarles da R. Moralles Guterres[1], Vinicius S. Monego[2] e Alice de J. Kozakevicius[3]

25 de julho de 2016

[1] tiarlesmoralles@hotmail.com
[2] vinicius.monego@ecomp.ufsm.br
[3] alice.kozakevicius@gmail.com

Sumário

1 Introdução

Transformações Lineares é um dos conteúdos tratados na disciplina de Álgebra Linear e possui grande relevância para o estudo de vários outros tópicos (autovalor, autovetor, mudança de base, etc) dentro desta disciplina. Além disso, Transformações Lineares também são vistas em outras disciplinas como por exemplo métodos estatísticos, eletromagnetismo, equações diferenciais, modelagem para engenharia, etc.

Hoje em dia a disponibilidade de ferramentas computacionais nos permite modelar e simular problemas em diferentes áreas de forma iterativa e com visualização eficiente de resultados, aumentando a qualidade das análises decorrentes. Neste sentido, o minicurso proposto procura motivar o estudo da Álgebra Linear através da utilização do aplicativo livre e gratuito SageMath® , permitindo uma melhor visualização e interpretação de resultados geométricos referentes à disciplina. Para isso, abordaremos o capítulo de Transformações Lineares do plano no plano, conforme proposto no texto *Álgebra Linear com Aplicações* do Boldrini [1].

A vantagem da escolha deste aplicativo é devido à sua flexibilidade de utilização, pois pode ser feita completamente na nuvem, sem que o aplicativo tenha que ser instalado na máquina do usuário.

2 SageMath® : a ferramenta de desenvolvimento

SageMath® é um CAS (Computer Algebra System, ou Sistema de Álgebra Computacional) de código livre construído por muitos pacotes livres distintos unidos em uma interface usando a linguagem de programação Python. Sua missão é ser um substituto livre viável aos CAS proprietários tradicionais, como MATLAB® ou Mathematica® .

A seguir apresentam-se os passos principais para se poder trabalhar nesse ambiente computacional. Logo após, os principais conceitos e definições de transformações lineares serão explorados através do Sage. A forma mais fácil de se usar o SageMath é se cadastrando em `https://cloud.sagemath.com`. O SMC (SageMathCloud) é uma plataforma colaborativa baseada em computação em nuvem voltada para aplicações matemáticas.

Para realizar o cadastro:

1. Se for novo usuário: preencher os dados na página inicial e clicar em "Sign Up!", ao clicar em "Sign Up" você concorda com os Termos de Serviço.

2. Se já estiver cadastrado, informar o email e a senha do cadastro e clicar em "Sign in" .

O cadastro e acesso pode ser realizado também diretamente a partir de outras redes dm que o usuário já esteja inscrito como Facebook® , GitHub® , Google+® ou Twitter® .

3 Utilização do SageMath

3.1 Criar novo projeto

1. Clicar em "Projects" e depois na opção "Create New Project", dar nome ao projeto na linha da opção correspondente (será aberta uma janelinha para estas informações) e clicar em "Create Project without upgrades". Automaticamente o projeto criado aparecerá na sua lista de projetos existentes. Está opção garante que o projeto estará dentro de um pacote para usuários básicos do SageMath® , para equipes maiores ou projetos mais complexos existem os pacotes com upgrade para um melhor suporte aos desenvolvedores.

2. Abrir o projeto e clicar na aba "New" para criar um novo arquivo ou diretório.

3. Coloque um nome no arquivo e clique em "Sage Worksheet" para abrir uma planilha (espaço de trabalho).

É possível criar outro tipo de arquivo, como um arquivo LaTeX na opção "LaTeX Document" ou um arquivo de texto na opção "File". Para criar outro tipo de arquivo dentro de um mesmo projeto, é só clicar em "New" e escolher a opção desejada (Sage Worksheet, Jupyter Notebook, File, Folder, From Internet, LaTeX Document, Terminal, Task List).

3.2 Trabalhar em projeto já existente

Para abrir o projeto é preciso clicar em "Projects" e clicar no projeto que deseja abrir. Aparecerão todos os arquivos e diretórios criados dentro do projeto. É só clicar na opção que desejar abrir.

Uma planilha no Sage é composta por várias células nas quais se escrevem códigos. Pode-se pensar nessas células como sendo linhas com uma ou mais instruções. Para avaliar cada célula individualmente, é necessária a combinação de teclas $< shift > + < enter >$ no final de cada instrução. Isso fará com que o resultado da célula (instrução) apareça em uma janela. Este passo será considerado sempre que se desejar visualizar os resultados das operações em questão.

Para sair do SageMathCloud é só clicar onde é mostrado seu nome, no canto superior direito da tela, e depois clicar em "Sign out".

3.3 Começando a utilizar a planilha

Aqui descreveremos como executar as operações básicas das transformações lineares na planilha. Neste texto a parte indicada por `>>>` deve ser digitada em uma célula da planilha. A parte que não tem esse prefixo é o resultado esperado do código e não deve ser digitado.

A construção de um vetor é feita por um objeto do tipo vector, o qual recebe como parâmetros uma lista de números que representam as suas coordenadas. Uma lista é delimitada por colchetes. Para representar vetores no $\mathbb{R}^2$, a lista conterá dois elementos.

```
>>> v = vector([5, 6]) # <enter>
>>> print(v) # <shift> + <enter>
(5, 6) # Resultado que aparecerá na tela
```

Listing 1: Vetor

A construção de uma matriz é feita por um objeto do tipo matrix, o qual recebe como parâmetros uma lista de listas, cada uma indicando as linhas da matriz.

```
>>> M = matrix([[1, 2],
                [3, 4]])
>>> print(M)
[1 2]
[3 4]
```

Listing 2: Matriz

A operação de multiplicação de uma constante por um vetor, entre dois vetores (Produto Escalar ou Interno) ou entre um vetor e uma matriz é feita utilizando o operador *, como mostram os exemplos a seguir:

```
>>> k = 3                  # Constante
>>> v = vector([1, 0])     # vetor

>>> W1 = k*v               # Constante x Vetor
>>> print(W1)
(3, 0)
>>> p = vector([2, 2])
>>> W2 = v*p               # Vetor x Vetor
>>> print(W2)
2
>>> M = matrix([[1, 2],
                [3, 4]])
>>> c1 = M*v                # Matriz * Vetor (Extrai a coluna 1 de M)
>>> print(c1)
(1, 3)

>>> l1 = v*M                # Vetor * Matriz (Extrai a linha 1 de M)
>>> print(l1)
(1, 2)
```

Listing 3: Multiplicações

Soma de dois vetores:

```
>>> v = vector([5, 6])
>>> t = vector([3, 1])
>>> vsoma = v + t
>>> print(v, " + ", t, " = ", vsoma)
(5, 6)  +  (3, 1)  =  (8, 7)
>>> nulo = vector([0,0])
>>> arrow(nulo, v) + arrow(nulo, v+t, color='green') + arrow(v, v+t, color='red')
```

Listing 4: Soma de vetores

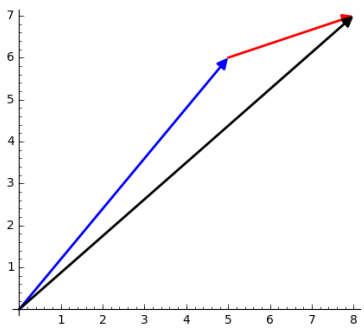

Figura 1: Exemplo da função arrow

4 Transformações Lineares - Transformações do plano no plano

Definição: Sejam $V = \mathbb{R}^2$ espaço domínio, $W = \mathbb{R}^2$ espaço contradomínio. Uma transformação $T : V \to W$, é uma aplicação que a cada $v \in V$ associa um vetor $w \in W$, $w = T(v)$, w sendo a imagem do vetor v pela aplicação T.

T é dita Transformação Linear se satisfaz as seguintes propriedades:

- $T(u + v) = T(u) + T(v)$, para quaisquer vetores u e $v \in V$.
- $T(av) = aT(v)$, para todo $a \in \mathbb{R}$ e para todo $v \in V$.

Resultado importante:
Dada uma base $B = \{e_1, e_2\}$ do espaço domínio V, e uma base D do contradomínio W, temos:

1. As coordenadas de um vetor v na base B são as constantes da combinação linear deste vetor em função dos elementos da base e a notação é dada por: $[v]_B = (a, b)_B := ae_1 + be_2$;

2. A imagem de v pela aplicação linear T leva em consideração as coordenadas de v na base B, uma vez que $T(v) = T(a, b) = T(ae_1 + be_2) = T(ae_1) + T(be_2) = aT(e_1) + bT(e_2)$;

3. Como o contradomínio W ainda pode ter uma base D diferente da base do domínio, $T(e_1)$ e $T(e_2)$ devem ser escritos na base D de W. Com isso a transformação T pode ser escrita na forma matricial que leva em consideração estes fatos: $[T(e_1)T(e_2)]^B_D[v]_B = T(v)$ e a matriz $[T(e_1)T(e_2)]^B_D$ é dita a matriz da transformação em relação às bases B e D.

Assim, toda transformação linear possui uma forma matricial que depende da escolha das bases para o domínio e contradominio. Na verdade, dada uma matriz, também é possível se determinar uma única transformação linear associada à matriz dada, ver Boldrini [1]. Dito isso, $[T]v = w$ recebe esta notação simplificada, quando as bases B e D forem as canônicas de cada espaço. Sendo portanto $[T]$ a matriz da transformação linear da canônica na canônica, e w a imagem de v pela transformação T, ambos vetores escritos nas respectivas bases canônicas.

Dependendo da aplicação e do software, pode ser interessante também considerar a transposta desta relação matricial: assim podemos tratar $w^t = v^t[T]^t$.

Para representarmos uma Transformação Linear no SageMath® , são necessários dois tipos de dados: a matriz e o vetor, a operação se dá pela multiplicação da matriz pelo vetor, na forma:

$$\begin{bmatrix} a_{00} & a_{10} \\ a_{01} & a_{11} \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{1}$$

Como nome já diz a transformação é linear, onde:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0a_{00} + y_0a_{10} \\ x_0a_{01} + y_0a_{11} \end{bmatrix}$$

Para introduzirmos melhor o conceito de transformações temos alguns exemplos básicos de transformações do plano no plano (ou no $\mathbb{R}^2$) conforme o que é visto em [1]:

4.1 Expansão e contração uniforme

O vetor transformado mantém direção e sentido, porém o seu módulo varia:

```
>>> v = vector([5, 6])    # vetor
>>> M = matrix([[3, 0],
                [0, 3]]) # Matriz de Transformação

>>> W = M*v
>>> print(W)

>>> v_plot = arrow((0,0), v, color="black")
>>> W_plot = arrow((0,0), W, color="red")

>>> show(W_plot + v_plot, aspect_ratio=1)
(15, 18)
```

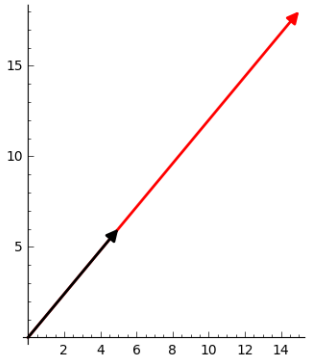

Figura 2: Expansão uniforme

4.2 Reflexão em torno do eixo

Para refletir um vetor em torno de um dos eixos basta modificar o sentido de uma de suas componentes:

```
>>> v = vector([5, 6])     # Vetor
>>> M = matrix([[1, 0],
               [0, -1]]) # Matriz de Transformação

>>> nulo = vector([0,0])

>>> W = M*v
>>> print(W)

>>> v_plot = arrow(nulo, v, color="black")
>>> W_plot = arrow(nulo, W, color="red")

>>> show(v_plot + W_plot, aspect_ratio=1)
(5, -6)
```

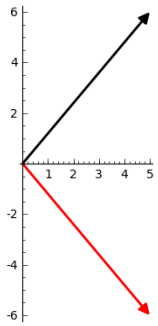

Figura 3: Rotação no eixo

4.3 Reflexão na origem

Para refletir um vetor na origem muda-se o sentido de de todo o vetor:

```
>>> v = vector([5, 6])     # Vetor
>>> M = matrix([[-1, 0],
               [0, -1]]) # Matriz de Transformação

>>> nulo = vector([0, 0])

>>> W = M*v
>>> print(W)

>>> v_plot = arrow(nulo, v, color="black")
>>> W_plot = arrow(nulo, W, color="red")

>>> show(v_plot + W_plot, aspect_ratio=1)
(-5, -6)
```

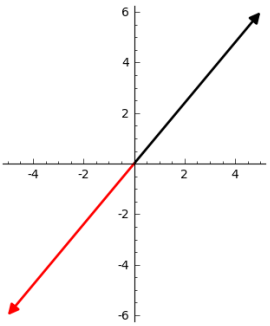

Figura 4: Rotação na origem

4.4 Rotação em ângulo

A rotação de um vetor em ângulo θ depende da matriz de rotação. Neste caso, apra cada escolha de θ é possível variar o ângulo de orientação do vetor, mantendo seu módulo.

```
>>> v = vector([1, 0]) # vetor

>>> theta = pi/3

>>> M = matrix([[cos(theta), -sin(theta)], # Matriz de rotação
               [sin(theta), cos(theta)]])
```

```
>>> nulo = vector([0,0])

>>> W = M*v

>>> print(W)

>>> v_plot = arrow(nulo, v, color="black")
>>> W_plot = arrow(nulo, W, color="red")

>>> show(v_plot + W_plot, aspect_ratio=1)
(1/2, 1/2*sqrt(3))
```

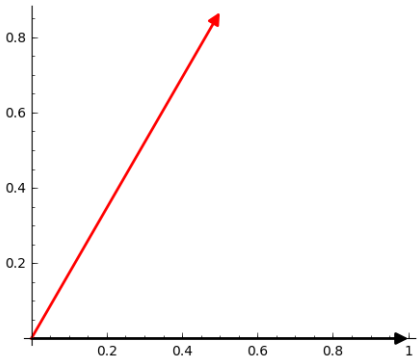

Figura 5: Rotação em ângulo

4.5 Cisalhamento

No cisalhamento uma das componentes permanece a mesma e a outra sofre transformação em função da outra componente:

```
>>> v = vector([5, 6])     # Vetor
>>> M = matrix([[1, 0],
               [2, 1]])   # Matriz de Transformação

>>> nulo = vector([0, 0])

>>> W = M*v
>>> print(W)
(5, 16)
>>> v_plot = arrow(nulo, v, color="black")
>>> W_plot = arrow(nulo, W, color="red")

>>> show(v_plot + W_plot, aspect_ratio=1)
```

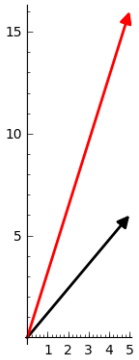

Figura 6: Cisalhamento Horizontal

5 Tópicos Complementares

5.1 Translação

A translação de um vetor é a mudança de posição deste vetor através da soma com um valor constante. Esta transformação não é linear, no entanto pode ser tratada através de sua representação via SageMath. O resultado é a mudança do vetor para uma posição paralela, preservando todos os demais elementos: direção, sentido e módulo do vetor. Lembrando que um vetor é obtido através da variação entre duas posições inicial A e final B, (v = Ponto B - Ponto A), um vetor somado a um ponto é então um novo ponto (v + Ponto A = Ponto B). Essa é a essência da translçação a ser representada agora.

```
>>> v = vector([4, 5])      # Vetor 1
>>> t = vector([3, 1])      # Vetor 2
>>> m = identity_matrix(2) # Matriz Identidade
>>> print(m)
```

```
[1 0]
[0 1]

>>> W = v * m              # = Vetor 1
>>> W = W + t

>>> print(W)

>>> nulo = vector([0,0])

>>> v_plot =  arrow(nulo, v   , color = "blue")
>>> W_plot = arrow(t, W , color = "black")

>>> show(v_plot + W_plot, aspect_ratio=1)
(7, 6)
```

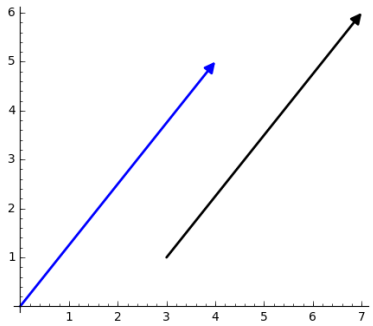

Figura 7: Translação

5.2 Transformações com matrizes de Markov

As matrizes de Markov são matrizes que possuem a soma das suas colunas igual a um (1).

- Elas sempre terão um autovalor igual a 1;
- Depois de transformado, o vetor imagem terá a soma de suas componentes preservada: $x_0 + y_0 = x_1 + y_1$.

Por exemplo:

```
>>> v = matrix([[5],
                [7]]) # -> Vetor na vertical

>>> soma = v[0]+v[1]   # Soma das componentes
>>> print(soma)
(12)

>>> Q = matrix([[.2, .7],
                [.8, .3]]) # -> Matriz de Markov (Sum.  Colunas = 1)

>>> W = Q*v                # Formato da Transformação
>>> soma = W[0]+W[1]
>>> print(soma)
(12.0000000000000)
```

5.3 Formas básicas no $\mathbb{R}^3$

O SageMath® dá suporte para trabalhar com representações gráficas (plots) em 2D e 3D, veja em [3]. Para exemplificar, apresentaremos formas básicas no $\mathbb{R}^3$:

- **Plano e seu Vetor Normal:**

```
>>> plane(x, y, z) = x + z        # Definindo uma função dependente de x, y e z
>>> v_normal = vector([1,0,1])
>>> lim = 1                       # Delimitações do espaço para plotagem
```

```
>>> plane_plot = implicit_plot3d(plane, (x, -lim, lim),
    (y, -lim, lim), (z, -lim, lim), mesh=True, color="lime")
    # Função utilizada para plotagens implícitas em 3D

>>> v_normal_plot = arrow((0,0,0), v_normal, color="black")

>>> show(plane_plot + v_normal_plot)
```

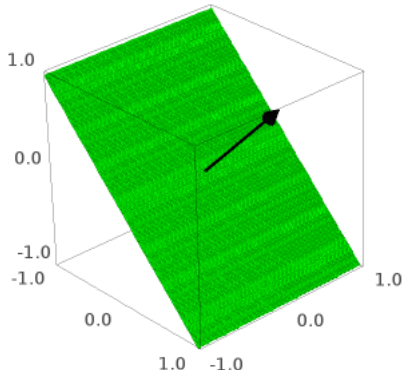

Figura 8: Plano e Vetor normal

- **Circunferência:**

```
>>> cir(x,y,z) = x^2 + y^2 - 4
>>> lim = 2

>>> cir_plot = implicit_plot3d(cir,
    (x, -lim, lim), (y, -lim, lim), (z, -lim, lim),
    mesh=True, color="red")

>>> show(cir_plot)
```

Esta circuferência no $\mathbb{R}^2$ é limitada nos eixos x e y, ela existe para todo z quando representada no $\mathbb{R}^3$.

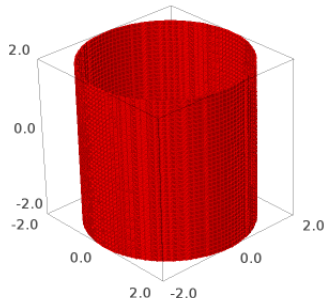

Figura 9: Circuferência

- **Esfera:**

```
>>> esf(x,y,z) = x^2 + y^2 + z^2 - 4
>>> lim = 2

>>> esf_plot = implicit_plot3d(esf,
    (x, -lim, lim),
    (y, -lim, lim),
    (z, -lim, lim),
    mesh=True)

>>> show(esf_plot)
```

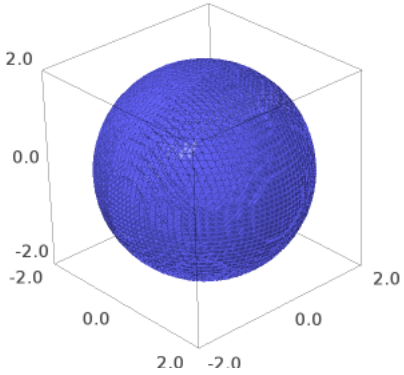

Figura 10: Esfera

5.4 Outras formas de abordagem de transformações lineares com o SageMath®

No material [5] é apresentado algumas formas bem interessantes e intuitivas de abordar transformações lineares no SageMath® .

No exemplo:

```
>>> x1, x2, x3 = var('x1, x2, x3')
>>> outputs = [ -x1 +        2*x3,
                  x1 + 3*x2 + 7*x3,
```

```
                  x1 +   x2 +   x3,
                2*x1 + 3*x2 + 5*x3]
>>> T_symbolic(x1, x2, x3) = outputs          # Matriz com variáveis
```

São criadas variáveis (x1, x2, x3) e depois postas em uma lista de equações.

```
>>> T = linear_transformation(QQ^3, QQ^4, T_symbolic)

Vector space morphism represented by the matrix:
[-1  1  1  2]
[ 0  3  1  3]
[ 2  7  1  5]
Domain: Vector space of dimension 3 over Rational Field
Codomain: Vector space of dimension 4 over Rational Field
```

Com esta função cria-se uma função para transformação linear com domínio de 3 dimensões e contradomínio de 4 dimensões (QQ^3 e QQ^4, respectivamente).

```
>>> u = vector(QQ, [3, -1, 2])
>>> w = T(u)

>>> print(w)
(1, 14, 4, 13)
```

Por fim, um vetor $\vec{u}$ é transformado pela matriz $[T]$, na forma mais utilizada em Álgebra Linear [1].

5.5 Aplicação: Processamento de imagens

As operações vistas até aqui podem ser aplicadas para realizar transformações em imagens. Como exemplo, suponha uma imagem de resolução 500x500 (altura x largura) e o eixo de rotação no centro da imagem, no ponto (250,250), onde a origem será (0,0). Queremos aplicar uma transformação de rotação na imagem fazendo a imagem girar 30º. Qual será a nova posição do pixel do ponto $[-13, 30]$ já em relação a essa origem?

Para isso aplica-se a matriz de rotação neste ponto.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(30) & -sin(30) \\ sin(30) & cos(30) \end{bmatrix} \begin{bmatrix} -13 \\ 30 \end{bmatrix}$$

Multiplicando a matriz de rotação pelo vetor, obtemos o resultado (arredondado) $[-26, 19]$. Essa é a nova coordenada do ponto $[-13, 30]$ após uma rotação de 30º.

Programas de manipulação de imagem como o GIMP utilizam este mesmo procedimento na rotação de imagens. A rotação consiste em aplicar a transformada em cada pixel da imagem representado pelas suas coordenadas $[x, y]$. Este processo é iterativo e leva mais tempo de execução quanto maior for a resolução da imagem (mais pontos precisam ser calculados). A Figura 5.5 mostra a interface gráfica de usuário aplicando uma transformação de rotação.

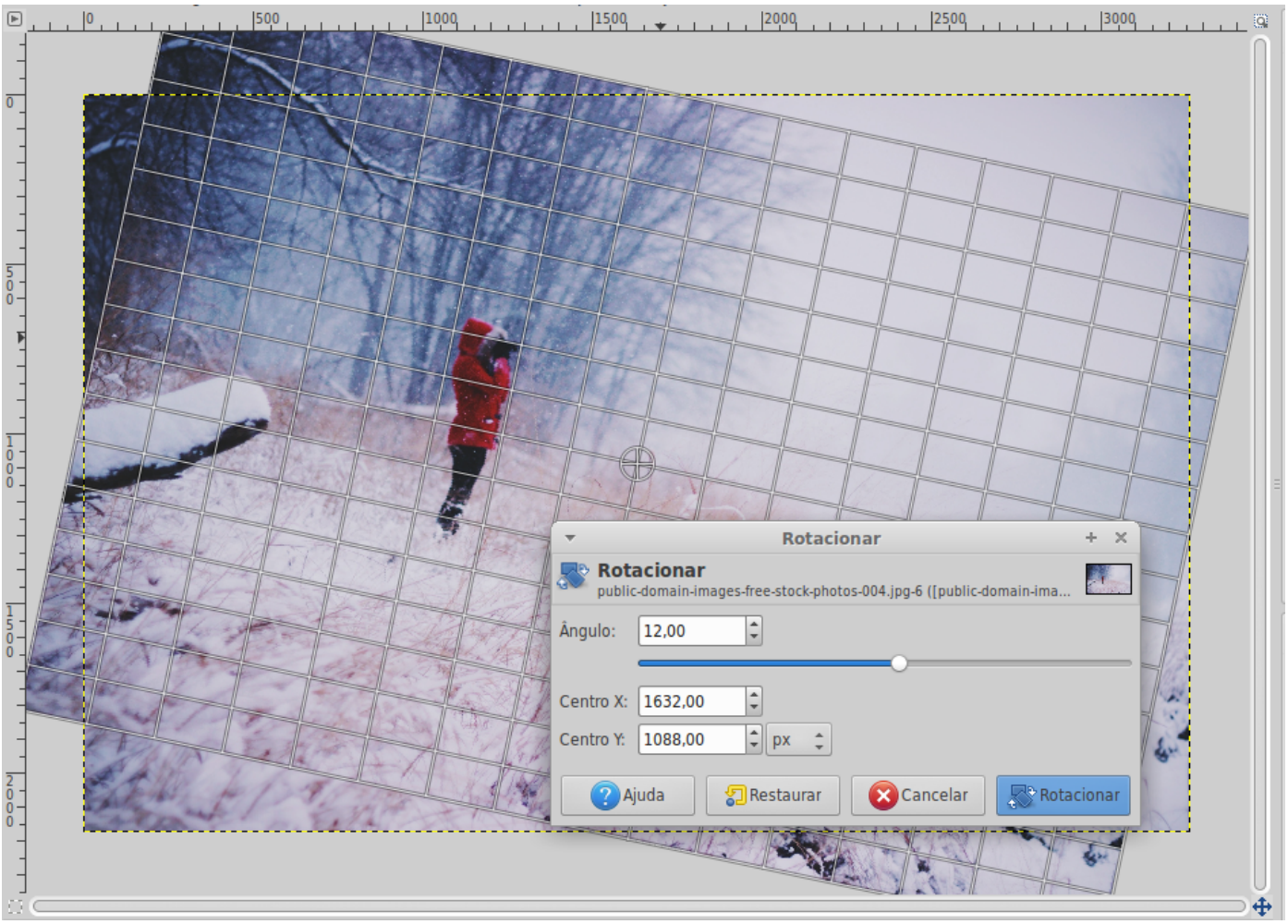

Figura 11: Imagem original e sua rotação de 12º em torno do centro, realizada no GIMP.

Referências

[1] J. L. Boldrini, S. I. R. Costa, V. R. Ribeiro, and H. G. Wetzler. *Álgebra Linear e Aplicações. Sec. 5.2 Transformações do plano no plano.* Harper-Row, São Paulo, 1987.

[2] Página do professor Robert A. Beezer da University of Puget Sound, Tacoma, WA. Disponível em `http://buzzard.ups.edu`

[3] Capítulo da Documentação do Sagemath® que aborda os vários tipos de plotagens suportados pela ferramenta. Disponível em `http://doc.sagemath.org/html/en/reference/plotting/sage/plot/plot.html`

[4] Parte complementar do livro "A First Course in Linear Algebra", de Robert A. Beezer. Disponível em `http://linear.ups.edu/`

[5] GIMP (GNU Image Manipulation Program). Disponível em `http://www.gimp.org/`