**Trabalho Computacional 3**

**Entrega:** 08/10/2019

**PARTE 1**

**Exercício 1 -** *Escreva um programa, na linguagem de sua escolha, para resolver um sistema de n equações lineares e n incógnitas usando o* ***método de iterativo de Gauss-Jacobi****. Os dados de entrada devem ser: a) o número de equações n; b) a matriz aumentada [A b]. .A saída deve ser o vetor solução [x] ou uma mensagem informando que o método não convergiu. Explique a precisão usada.*

*Teste seu programa para resolver o seguinte sistema linear de equações Ax=b, onde*

*A = [1 -1 0 5; 3 -2 1 -1;1 1 9 4; 1 -7 2 3]*

*x = [x1 x2 x3 x4]^t*

*b = [18 8 47 32 ]^t*

**R:** A precisão utilizada de valor 10-6 é relacionado a quanto os valores de ***b***(solução) são relacionáveis à ***X*** (resposta do sistema linear), dado pelas distâncias dos valores (módulo). Além disso, há a consideração da linguagem programada (*Python*), que considera determinadas casas após a vírgula.

**Código implementado:**

| import numpy as np  A = np.array([[1, -1, 0, 5],[3, -2, 1, -1],[1, 1, 9, 4],[1, -7, 2, 3]])  b = np.array([18, 8, 47, 32])  X = np.array([0, 0, 0, 0])  x = np.array([0, 0, 0, 0])  print ('Matriz dada:')  print(A)  n = 4  ni = 0  while (ni < 1000):  for i in range(n):  soma = 0  for j in range(n):  if(j != i):  soma = soma + A[i,j]\*X[j]/A[i,i]  x[i] = (b[i]/A[i,i]) - soma    if (abs(np.linalg.norm(b) - np.linalg.norm(X)) < 10\*\*(-6)):  print('Resposta do sistema', X)  print('Número de iterações utilizadas: ', ni)  break  else:  X = x  ni = ni + 1;  if (x[i] > 100000):  print('O sistema diverge (Não convergente)')  break  **RESULTADO:**  *Matriz dada:*  *[[ 1 -1 0 5]*  *[ 3 -2 1 -1]*  *[ 1 1 9 4]*  *[ 1 -7 2 3]]*  *O sistema diverge (Não convergente)* |
| --- |

**Exercício 2 -** *Uma empresa fabrica três tipos de móveis: cadeiras, mesas e armários. Cada móvel requer uma quantidade de madeira, plástico e alumínio, conforme a tabela abaixo. A empresa tem em estoque 400 unidades de madeira, 600 unidades de plástico e 1500 unidades de alumínio. Por ser final de temporada, a empresa quer usar todo o seu estoque. Para fazer isso, quantas cadeiras, mesas e armários ela precisa fabricar? Use* ***o método de iterativo de Gauss-Jacobi.***

|  | ***Madeira*** | ***Plástico*** | ***Alumínio*** |
| --- | --- | --- | --- |
| ***Cadeira*** | *1 unidade* | *1 unidade* | *2 unidades* |
| ***Mesa*** | *1 unidade* | *1 unidade* | *3 unidades* |
| ***Armário*** | *1 unidade* | *2 unidades* | *5 unidades* |

**Código implementado:**

| import numpy as np  A = np.array([[1, 1, 1],[2, 3, 5],[1, 1, 2]])  b = np.array([400, 1500, 600])  X = np.array([0, 0, 0])  x = np.array([0, 0, 0])  print ('Matriz dada:')  print(A)  n = 3  m = 3  ni = 0  while (ni < 1000):  for i in range(n):  soma = 0  for j in range(m):  if(j != i):  soma = soma + A[i,j]\*X[j]/A[i,i]  x[i] = (b[i]/A[i,i]) - soma    if (abs(np.linalg.norm(b) - np.linalg.norm(X)) < 10\*\*(-6)):  break  else:  X = x  ni = ni + 1;  print('Resposta do sistema', X)  print('Número de iterações utilizadas: ', ni)  **RESULTADO:**  *Matriz dada:*  *[[1 1 1]*  *[2 3 5]*  *[1 1 2]]*  *Resposta do sistema [100 100 200]*  *Número de iterações utilizadas: 1000* |
| --- |

**R:** De acordo com o resultado obtido, temos que a quantidade de cadeiras é 100, quantidade de armários é 100 e a quantidade de mesas é de 200.

**PARTE 2**

**Exercício 1 -** *Análogo ao Ex1 - Parte 1, mas agora utilizando o* ***Método de iterativo de Gauss-Seidel.***

**Código implementado:**

| import numpy as np  n = 4  A = np.array([[1., -1, 0, 5],[3, -2, 1, -1],[1, 1, 9, 4],[1, -7, 2, 3]])  b = np.array([18., 8, 47, 32])  x0 = np.array([0.0, 0, 0, 0])  x1 = np.copy(x0)  for cs in range(1000):  for i in range(n):  x1[i] = b[i]  for j in range(n):  if i != j:  x1[i] -= A[i,j]\*x1[j]  x1[i] /= A[i,i]  if (np.linalg.norm(x1-x0,np.inf)) > 1000:  print("Método diverge")  break  if (np.linalg.norm(x1-x0,np.inf) < 10\*\*(-6)):  print(x1)  print(cs+1)  break  else:  x0 = np.copy(x1)  **RESULTADO:**  Método diverge |
| --- |

**R:** De acordo com o resultado obtido, temos o método de Gauss-Seidel, que diverge nesse caso de sistema linear.

**Exercício 2 -** *Análogo ao Ex2 - Parte 1, mas agora utilizando o* ***Método de iterativo de Gauss-Seidel.***

**Código implementado:**

| import numpy as np  n = 3  A = np.array([[1.0, 1.0, 1.0],[2.0, 3.0, 5.0],[1.0, 1.0, 2.0]])  b = np.array([400.0, 1500.0, 600.0])  x0 = np.array([0.0,0,0])  x1 = np.copy(x0)  for cs in range(1000):  for i in range(n):  x1[i] = b[i]  for j in range(n):  if i != j:  x1[i] -= A[i,j]\*x1[j]  x1[i] /= A[i,i]  if (np.linalg.norm(x1-x0,np.inf)) > 1000:  print("Método diverge")  break  if (np.linalg.norm(x1-x0,np.inf) < 10\*\*(-6)):  print(x1)  print(cs+1)  break  else:  x0 = np.copy(x1)  **RESULTADO:**  [100.00000353 99.99999667 199.9999999 ]  207 |
| --- |

**R:** O método de Gauss-Seidel converge para esse sistema linear, com resultados análogos aos métodos já aplicados nos exercícios anteriores que utilizaram esse mesmo sistema.

**Exercício 3 -** *Gere uma matriz esparsa 1000 x 1000 tal que aii é diferente de zero para todo i. Considere que 70 a 85% das entradas são nulas. Gere um vetor aleatório b em R^(1000). Resolva o sistema linear Ax=b usando os algoritmos que implementou: Gauss, Gauss-Jacobi e Gauss-Seidel. Compare os resultados e o desempenho de cada método.*

**Código implementado:**

| import numpy as np  from random import randint  def jacobi (A, b, x0, n):  x1 = np.zeros((n,), dtype = float)  for cs in range(1000):  for i in range(n):  x1[i] = b[i]  for j in range(n):  if i != j:  x1[i] -= A[i,j]\*x0[j]    x1[i] /= A[i,i]  if (np.linalg.norm(x1-x0,np.inf)) > 1000:  print("Método de Jacobi diverge")  break  if (np.linalg.norm(x1-x0,np.inf) < 10\*\*(-6)):  print("Jacobi", x1)  print(cs+1)  break  else:  x0 = np.copy(x1)  def seidel (A, b, x0, n):  x1 = np.copy(x0)  for cs in range(1000):  for i in range(n):  x1[i] = b[i]  for j in range(n):  if i != j:  x1[i] -= (A[i,j]\*x1[j])  x1[i] /= A[i,i]  if (np.linalg.norm(x1-x0,np.inf)) > 1000:  print("Método de Seidel diverge")  break  if (np.linalg.norm(x1-x0,np.inf) < 10\*\*(-6)):  print("Seidel", x1)  print(cs+1)  break  else:  x0 = np.copy(x1)  n = 1000  b = np.random.rand(n,1) \* 10  passou = 0  cont = 0  A = np.zeros((n,n), dtype = float)  while passou != (n-1):  soma = 0  inseridos = 0  insere = np.random.rand(1,1) \* 100  A[passou,passou] = insere  while inseridos < 150:  insere = np.random.rand(1,1) \* 100  k = randint(0, 999)  if((soma + insere) < A[passou,passou]) and (k != passou):  A[passou,k] = insere  inseridos += 1  passou += 1  x0 = np.random.rand(n,1) \* 100  jacobi (A, b, x0, n)  seidel (A, b, x0, n)  x0 = np.random.rand(n,1) \* 100  jacobi (A, b, x0, n)  seidel (A, b, x0, n)  x0 = np.random.rand(n,1) \* 100  jacobi (A, b, x0, n)  seidel (A, b, x0, n) |
| --- |

**R:** Ambos os métodos não convergiram nos três testes

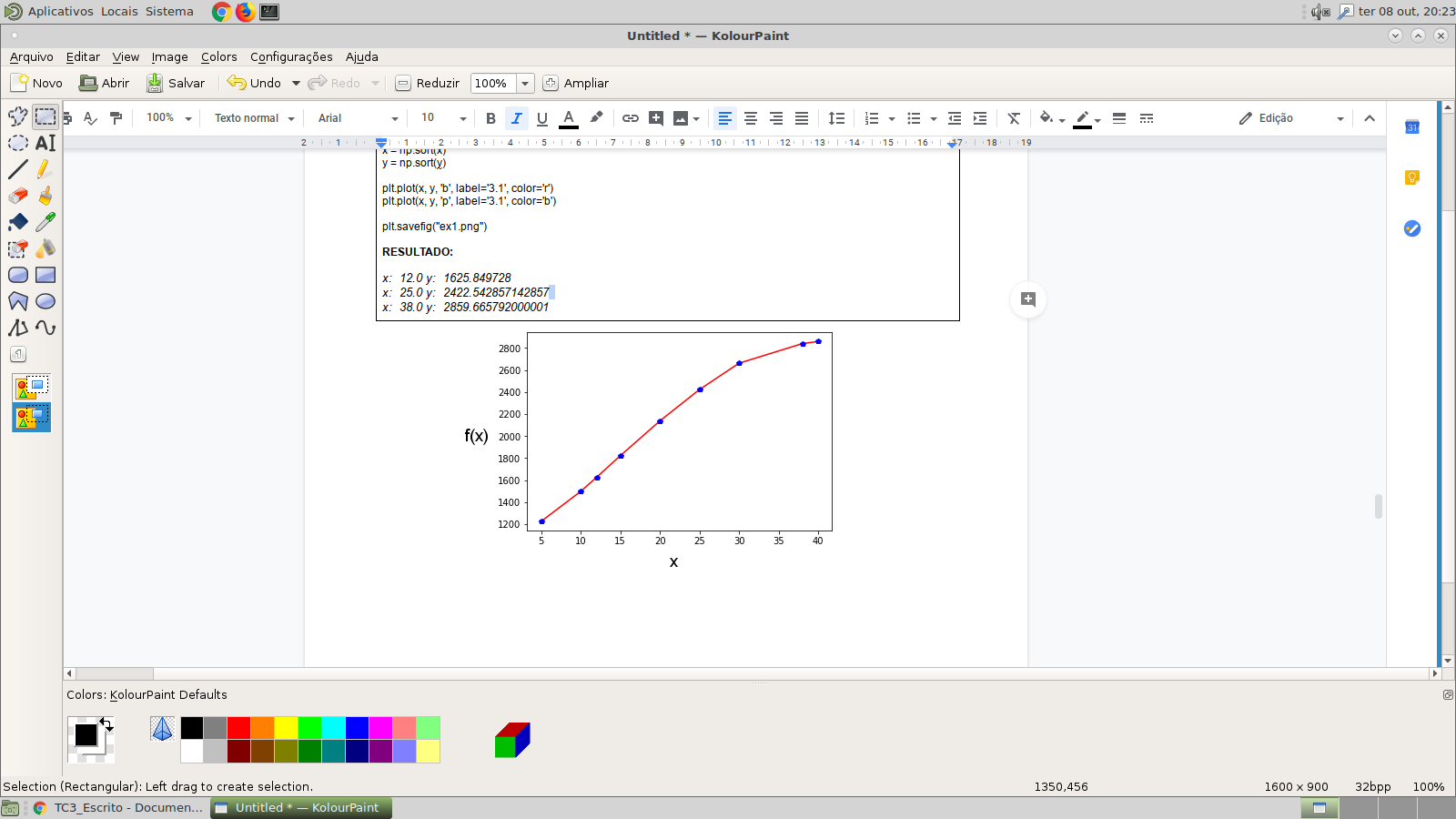
**PARTE 3**

**Exercício 1 -** *A tabela abaixo dá a viscosidade V do etanol como uma função do percentual w de álcool anidro. Determine a viscosidade para w=12, w=25 e w=38. Use interpolação polinomial para obter um polinômio de grau menor ou igual a 5. Faça o gráfico do polinômio encontrado e destaque os pontos dados.*

| **w** | 5 | 10 | 15 | 20 | 30 | 40 |
| --- | --- | --- | --- | --- | --- | --- |
| **V** | 1226 | 1498 | 1822 | 2138 | 2662 | 2840 |

**Código implementado:**

| import numpy as np  import matplotlib.pyplot as plt  def metodoLagrange(A, ordem, X):    P = 0  for i in range(ordem+1):  pDif1 = 1  pDif2 = 1  for j in range(ordem+1):  if(i != j):  pDif1 \*= (X-A[0,j])  pDif2 \*= (A[0,i]-A[0,j])  L = pDif1 / pDif2  P += A[1,i]\*L    print('x: ', X, 'y: ', P)  return P    A = np.array([[5, 10, 15, 20, 30, 40], [1226, 1498, 1822, 2138, 2662, 2840]])  x = np.copy(A[0])  y = np.copy(A[1])  inc = 12.0  x = np.append(x, inc)  y = np.append(y, metodoLagrange(A, 5, inc))  inc = 25.0  x = np.append(x, inc)  y = np.append(y, metodoLagrange(A, 5, inc))  inc = 38.0  x = np.append(x, inc)  y = np.append(y, metodoLagrange(A, 5, inc))  x = np.sort(x)  y = np.sort(y)  plt.plot(x, y, 'b', label='3.1', color='r')  plt.plot(x, y, 'p', label='3.1', color='b')  plt.savefig("ex1.png")  **RESULTADO:**  *x: 12.0 y: 1625.849728*  *x: 25.0 y: 2422.542857142857*  *x: 38.0 y: 2859.665792000001* |
| --- |



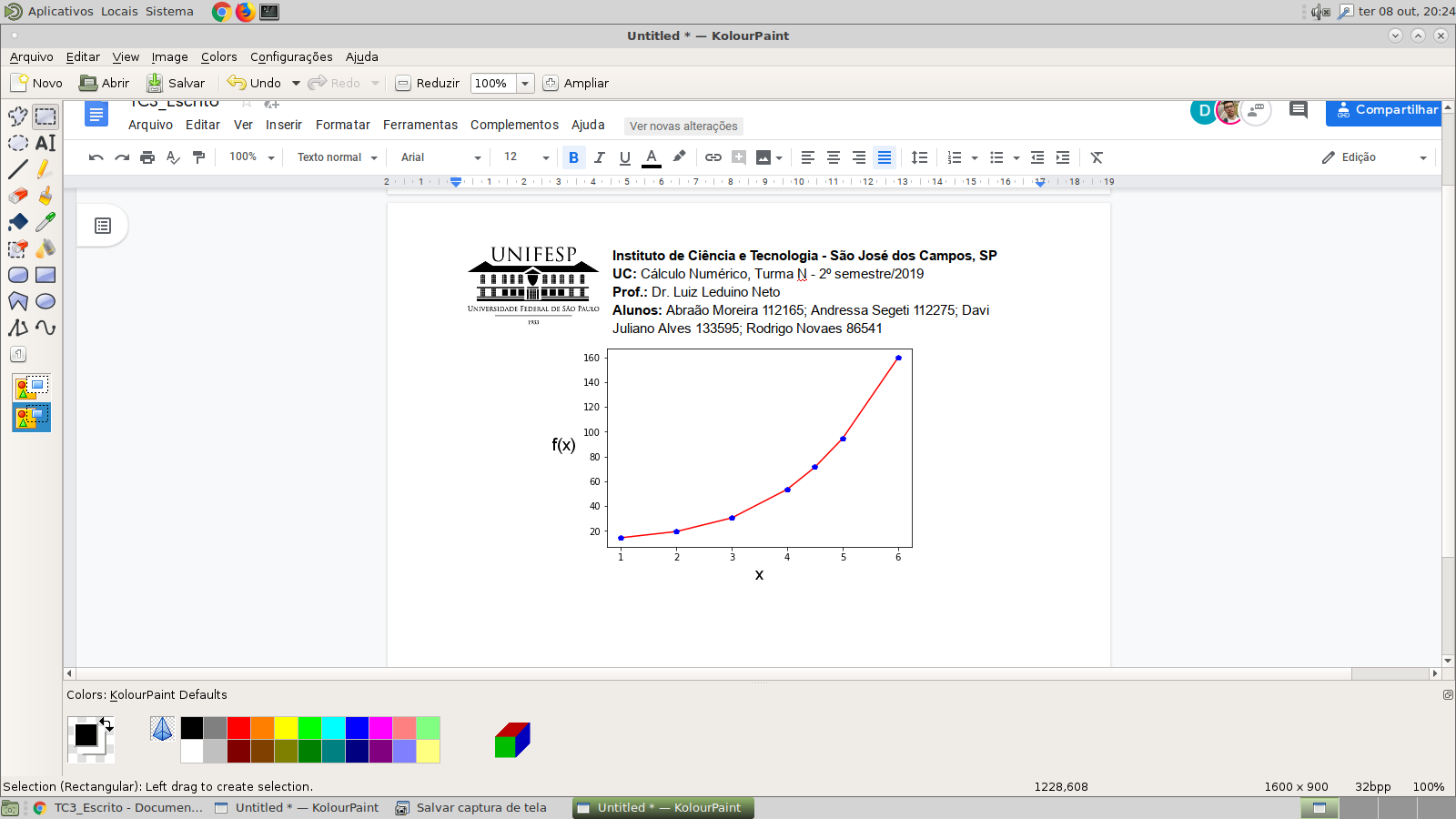
**Exercício 2 -** *Determine um polinômio de grau <=5, usando a forma de lagrange, que interpola a tabela a seguir:*

| **x** | 1 | 2 | 3 | 4 | 5 | 6 |
| --- | --- | --- | --- | --- | --- | --- |
| **f(x)** | 14.5 | 19.5 | 30.5 | 53.5 | 94.5 | 159.5 |

*Use esse polinômio para encontrar o valor de f(4.5). Compare com o valor real f(4.5)=71.375. Faça o gráfico do polinômio encontrado e destaque os pontos dados.*

**Código implementado:**

| import numpy as np  import matplotlib.pyplot as plt  def metodoLagrange(A, ordem, X):    P = 0  for i in range(ordem+1):  pDif1 = 1  pDif2 = 1  for j in range(ordem+1):  if(i != j):  pDif1 \*= (X-A[0,j])  pDif2 \*= (A[0,i]-A[0,j])  L = pDif1 / pDif2  P += A[1,i]\*L    print('x: ', X, 'y: ', P)  return P    A = np.array([[1.0, 2, 3, 4, 5, 6], [14.5, 19.5, 30.5, 53.5, 94.5, 159.5]])  x = np.copy(A[0])  y = np.copy(A[1])  inc = 4.5  x = np.append(x, inc)  y = np.append(y, metodoLagrange(A, 5, inc))  x = np.sort(x)  y = np.sort(y)  plt.plot(x, y, 'b', label='3.1', color='r')  plt.plot(x, y, 'p', label='3.1', color='b')  plt.savefig("ex2.png")  **RESULTADO:**  *x: 4.5 y: 71.375* |
| --- |

****

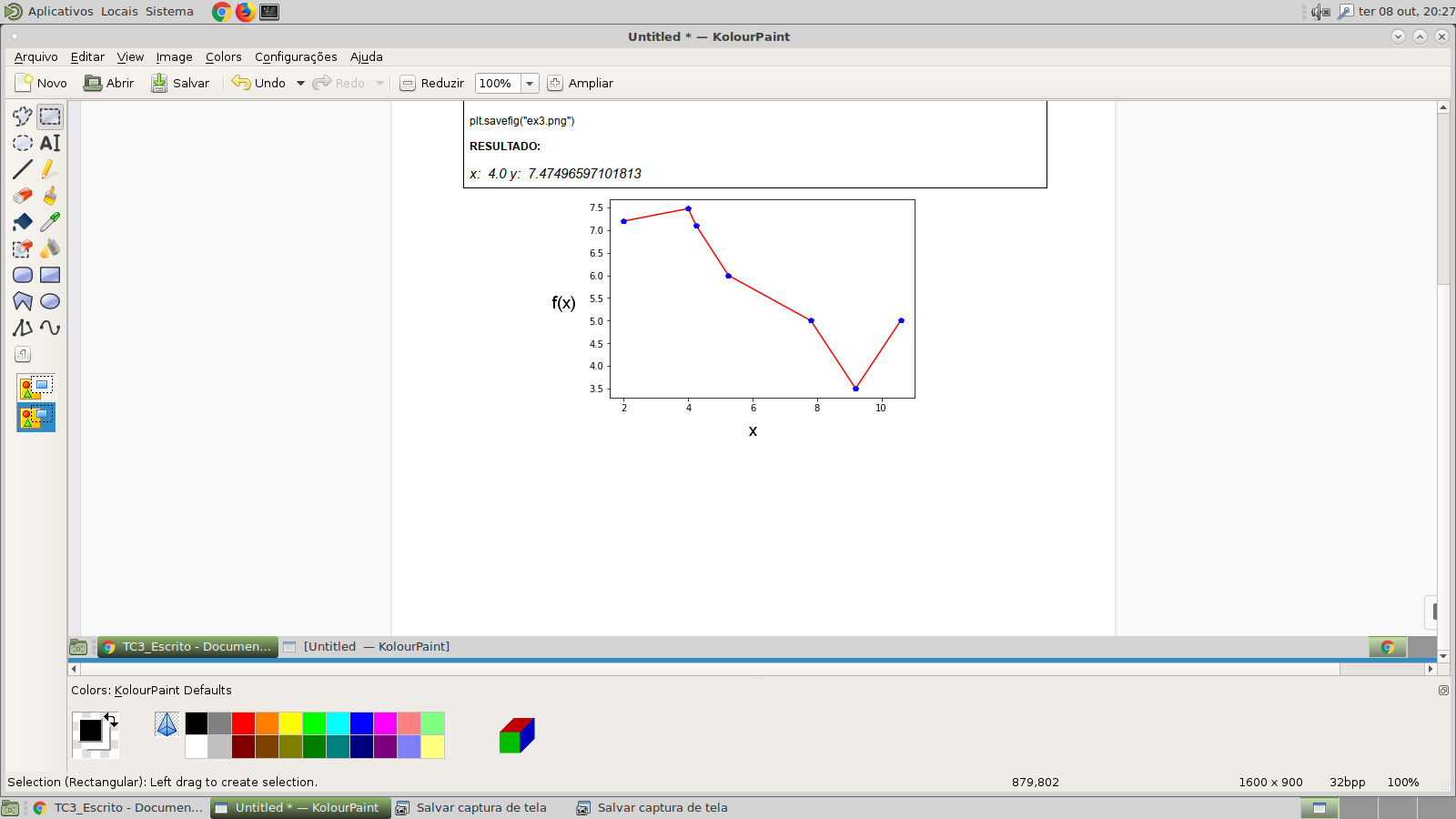
**Exercício 3 -** *Um braço robótico com um scanner a laser está fazendo uma verificação rápida da qualidade dos furos feitos em uma placa retangular. Os centros dos furos na placa descrevem o caminho que o braço precisa tomar. Os centros dos furos estão localizados em um sistema de coordenadas cartesianas (com a origem no canto inferior esquerdo da placa) dado pelas especificações na tabela a seguir..*

| **x (cm)** | **y(cm)** |
| --- | --- |
| 2.00 | 7.2 |
| 4.25 | 7.1 |
| 5.25 | 6.0 |
| 7.81 | 5.0 |
| 9.20 | 3.5 |
| 10.60 | 5.0 |

*Se o laser estiver percorrendo um caminho linear de x=2 a x=4.25, qual é o valor de y quando x=4?* ***Dica:*** *Use um polinômio de lagrange 1a ordem.*

**Código implementado:**

| import numpy as np  import matplotlib.pyplot as plt  def metodoLagrange(A, ordem, X):    P = 0  for i in range(ordem+1):  pDif1 = 1  pDif2 = 1  for j in range(ordem+1):  if(i != j):  pDif1 \*= (X-A[0,j])  pDif2 \*= (A[0,i]-A[0,j])  L = pDif1 / pDif2  P += A[1,i]\*L    if X == 4:  print('x: ', X, 'y: ', P)  return P    A = np.array([[2.0, 4.25, 5.25, 7.81, 9.20, 10.60], [7.2, 7.1, 6.0, 5.0, 3.5, 5.0]])  x = np.copy(A[0])  y = []  inc = 4  x = np.append(x, inc)  x = np.sort(x)  for i in x:  y = np.append(y, metodoLagrange(A, 5, i))  plt.plot(x, y, 'b', label='3.1', color='r')  plt.plot(x, y, 'p', label='3.1', color='b')  plt.savefig("ex3.png")    **RESULTADO:**  *x: 4.0 y: 7.47496597101813* |
| --- |

**