Tarefa 2

Equipe:

- Flávia Érika Almeida Giló Azevedo | RA: 162641
- Elian Raquel Laura Riveros | RA: 265685
- Yuliana Guadalupe Apaza Yllachura | RA: 234986

In [1]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from matplotlib import cm

def grafico(valores_x_fx):
  fig = plt.figure(figsize = (12, 8))
  fig.suptitle('Valores de f(x) nos passos da descida do gradiente', fontsize = 
16)
  ax = plt.axes(projection = "3d")
  f = lambda x1, x2: (1 - x1)**2 + 100*(x2 - x1**2)**2
  x1 = np.arange(-1.25, 1.25, 0.01)
  x2 = np.arange(-1.25, 1.25, 0.01)
  x1, x2 = np.meshgrid(x1, x2)
  y = f(x1,x2)
  minima = np.array([1,1])
  minima_ = minima.reshape(-1, 1)
  ax.plot_surface(x1, x2, y, cmap=cm.jet, linewidth=0, antialiased=False, alpha=
0.2)
  ax.plot(*minima_, f(*minima_), 'r*', markersize=10)
  ax.scatter3D(valores_x_fx[:,:1], valores_x_fx[:,1:2], valores_x_fx[:,2:3], col
or='r', s = 0.1 )
  ax.set_xlabel('x1', fontsize = 16)
  ax.set_ylabel('x2', fontsize = 16)
  ax.set_zlabel('f (x1,x2)', fontsize = 16)
  plt.tick_params(labelsize = 10)
  plt.show()
```

Definição da função de Rosenbrock 2D e sua derivada

In [2]:

```
# Função Rosenbrock
def funcao_rosenbrock(x1, x2, a = 1, b = 100):
  return (a - x1)**2 + b * (x2 - x1**2)**2

# Derivada da Função Rosenbrock
def derivada_rosenbrock(x1, x2):
  return np.array([2 * (200 * x1**3 - 200 * x1 * x2 + x1 - 1), 200 * (x2 - x1**2
)])
```

Questão 1: Implementação de descida do gradiente com gradiente explícito

In [3]:

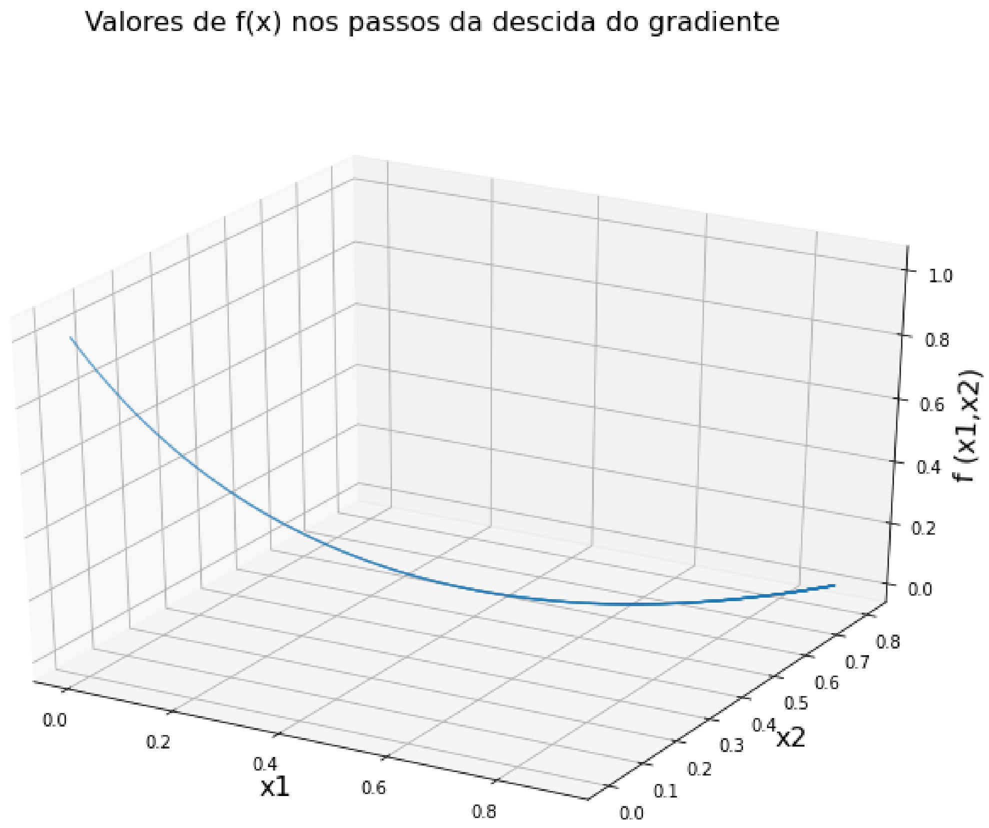
```
def gradiente_descendente(derivada_funcao, x_inicial, learning_rate, tolerancia_
epsilon, max_iter, taxa_decaimento_lr = 0):
  x_novo = x_inicial

  # Guarda os valores de [x1, x2, f(x1,x2)] no processo de descida do gradiente
  valores_x_fx = np.array([0, 0, 0])

  for i in range(1, max_iter+1):
    x_atual = x_novo
    x_fx = np.array([x_atual[0], x_atual[1], funcao_rosenbrock(x_atual[0], x_atu
al[1])])
    valores_x_fx = np.append(valores_x_fx, x_fx)
    x_novo = x_atual - learning_rate * derivada_funcao(x_atual[0], x_atual[1])

    # A política de redução do learning rate será "ativada" apenas se taxa_decai
mento_lr != 0
    # Ou seja, quando taxa_decaimento_lr for passado explicitamente na chamada d
a função com um valor diferente de zero
    learning_rate = learning_rate - (learning_rate * taxa_decaimento_lr)

    # Define outra condição de parada além do número máximo de iterações
    tolerancia = np.abs(funcao_rosenbrock(x_novo[0], x_novo[1]) - funcao_rosenbr
ock(x_atual[0], x_atual[1]))
    if tolerancia < tolerancia_epsilon:
      return x_atual, i, valores_x_fx[3:]
  return x_novo, max_iter, valores_x_fx[3:]
```

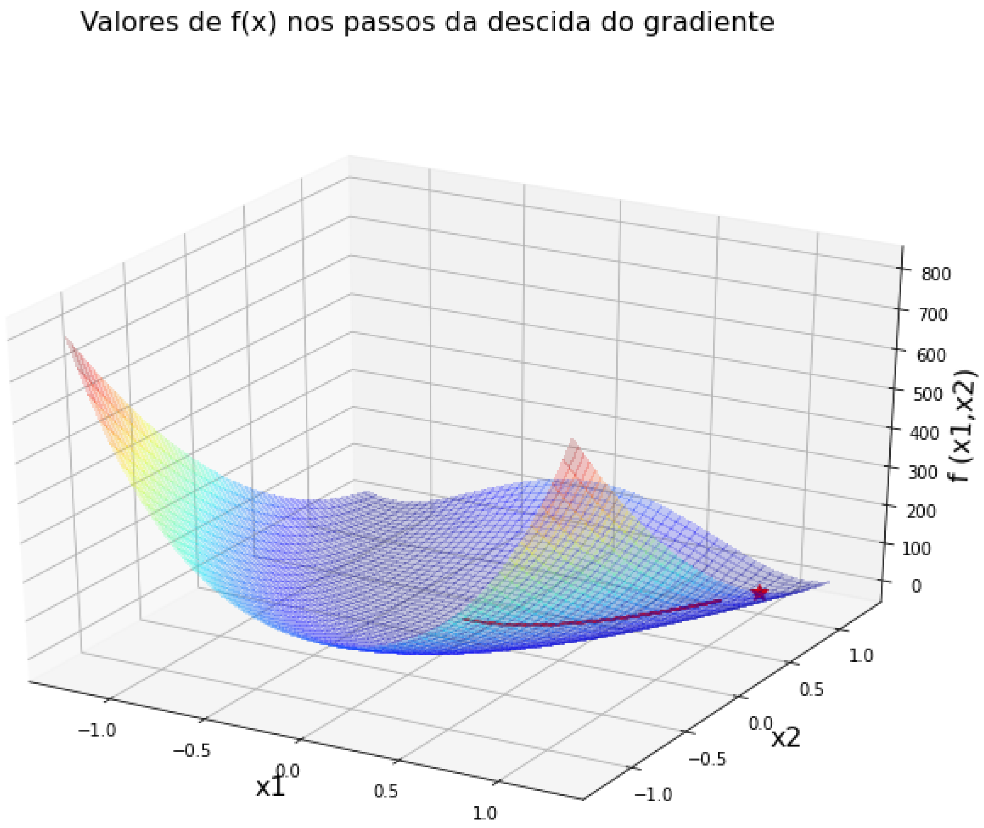
Questão 1.1: Learning rate = 1.e-3

In [4]:

```
learning_rate = 1e-3
tolerancia_epsilon = 1e-5
max_iter = 50000
x_inicial = np.array([0, 0])

x_min, num_iter, valores_x_fx = gradiente_descendente(derivada_rosenbrock, x_ini
cial, learning_rate, tolerancia_epsilon, max_iter)

print('Learning rate:', learning_rate)
print('Ponto mínimo =', x_min)
print('Função Rosenbrock no ponto mínimo =', funcao_rosenbrock(x_min[0], x_min[1
]))
print('Derivada da função Rosenbrock no ponto mínimo =', derivada_rosenbrock(x_m
in[0], x_min[1]))
print('Número de iterações =', num_iter)

# Valores de [x1, x2] e f(x1,x2) nos passos da descida do gradiente
valores_x_fx = valores_x_fx.reshape(-1, 3)
print("\nMatriz com todos os pontos [x1, x2] e seus respectivos valores da funçã
o rosenbrock f(x1,x2) na descida do gradiente:")
print(valores_x_fx)
print('\n')

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
fig = plt.figure(figsize = (12, 8))
fig.suptitle('Valores de f(x) nos passos da descida do gradiente', fontsize = 16
)
ax = plt.axes(projection = "3d")
ax.scatter3D(valores_x_fx[:,:1], valores_x_fx[:,1:2], valores_x_fx[:,2:3], 'gra
y', s = 0.1 )
ax.set_xlabel('x1', fontsize = 16)
ax.set_ylabel('x2', fontsize = 16)
ax.set_zlabel('f (x1,x2)', fontsize = 16)
plt.tick_params(labelsize = 10)
plt.show()

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
grafico(valores_x_fx)
```

```
Learning rate: 0.001
Ponto mínimo = [0.89731737 0.8047416 ]
Função Rosenbrock no ponto mínimo = 0.010562807758210266
Derivada da função Rosenbrock no ponto mínimo = [-0.04856236 -0.0873
7315]
Número de iterações = 3096

Matriz com todos os pontos [x1, x2] e seus respectivos valores da fu
nção rosenbrock f(x1,x2) na descida do gradiente:
[[0.00000000e+00 0.00000000e+00 1.00000000e+00]
 [2.00000000e-03 0.00000000e+00 9.96004002e-01]
 [3.99599680e-03 8.00000000e-07 9.92023997e-01]
 ...
 [8.97220164e-01 8.04566718e-01 1.05828182e-02]
 [8.97268780e-01 8.04654179e-01 1.05728078e-02]
 [8.97317370e-01 8.04741596e-01 1.05628078e-02]]
```





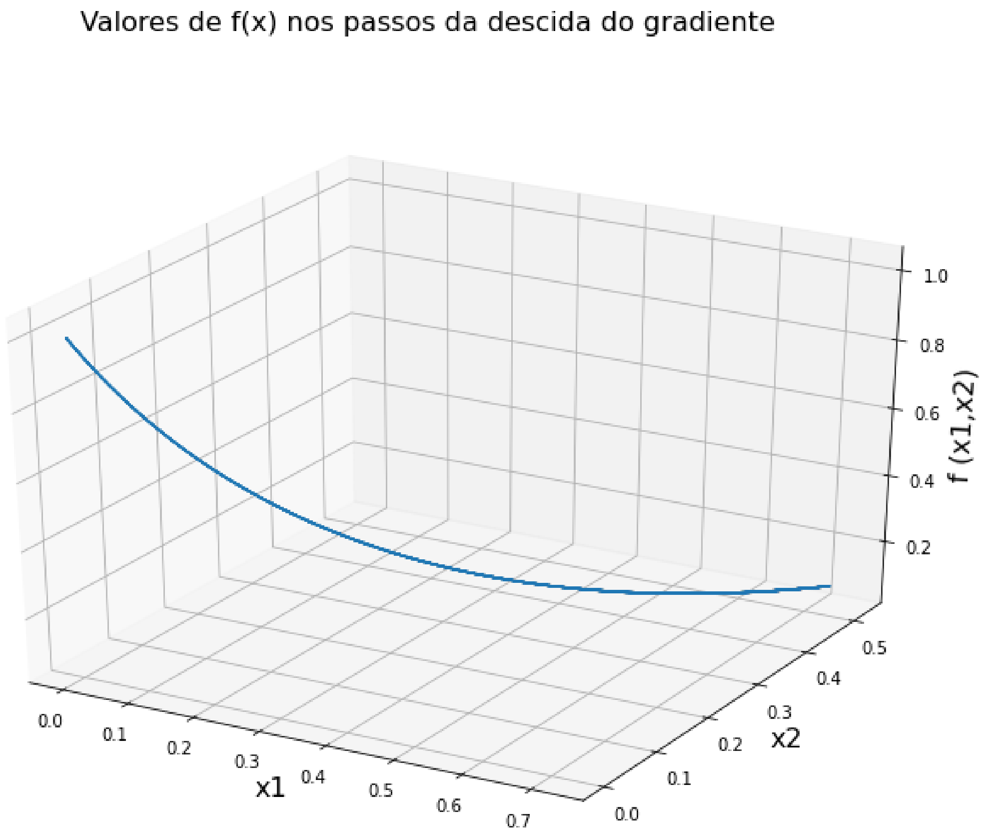
- Com o learning rate de 1.e-3, após 3096 iterações, o algoritmo de gradiente descendente implementado converge para o ponto de mínimo local [0.89731737, 0.8047416], onde o valor da função Rosenbrock é 0.010562807758210266 e sua derivada é [-0.04856236, -0.08737315].
- Nota-se que o mínimo alcançado está relativamente próximo do ponto [1, 1], que se sabe ser o mínimo global da função Rosenbrock 2D. O valor só não se aproximou mais de [1, 1] devido à tolerância de parada estabelecida em 1e-5.

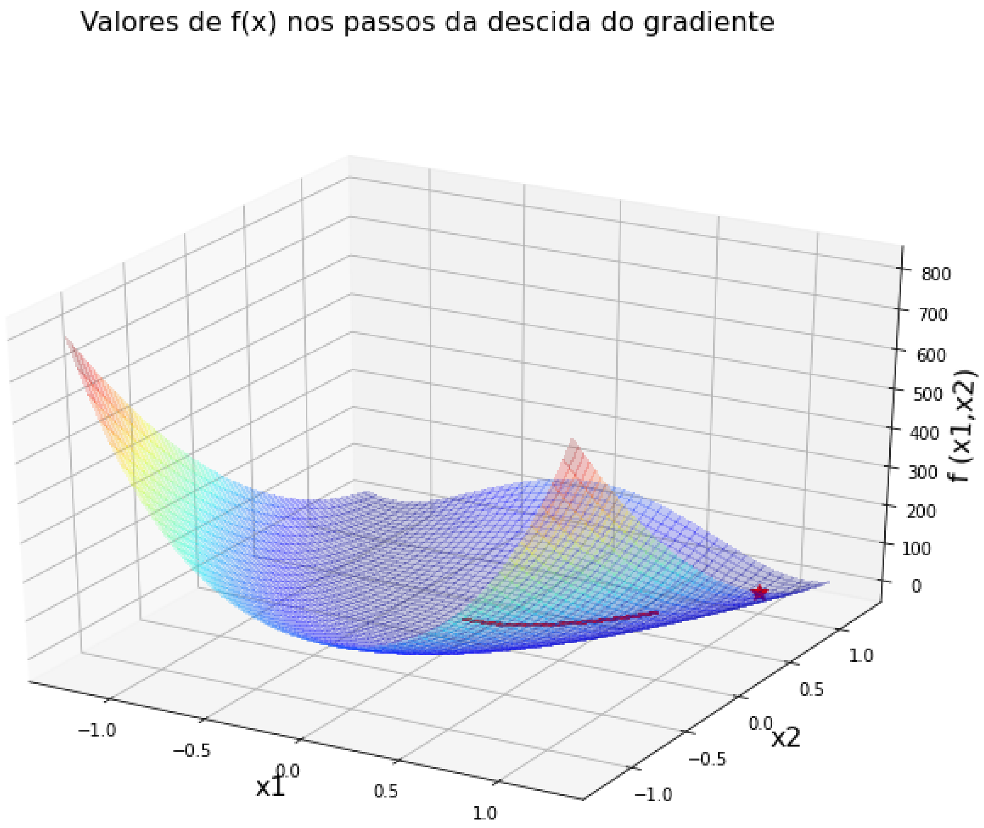
Questão 1.2: Learning rate = 1.e-4

In [5]:

```
learning_rate = 1e-4
tolerancia_epsilon = 1e-5
max_iter = 50000
x_inicial = np.array([0, 0])

x_min, num_iter, valores_x_fx = gradiente_descendente(derivada_rosenbrock, x_ini
cial, learning_rate, tolerancia_epsilon, max_iter)

print('Learning rate:', learning_rate)
print('Ponto mínimo =', x_min)
print('Função Rosenbrock no ponto mínimo =', funcao_rosenbrock(x_min[0], x_min[1
]))
print('Derivada da função Rosenbrock no ponto mínimo =', derivada_rosenbrock(x_m
in[0], x_min[1]))
print('Número de iterações =', num_iter)

# Valores de [x1, x2] e f(x1,x2) nos passos da descida do gradiente
valores_x_fx = valores_x_fx.reshape(-1, 3)
print("\nMatriz com todos os pontos [x1, x2] e seus respectivos valores da funçã
o rosenbrock f(x1,x2) na descida do gradiente:")
print(valores_x_fx)
print('\n')

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
fig = plt.figure(figsize = (12, 8))
fig.suptitle('Valores de f(x) nos passos da descida do gradiente', fontsize = 16
)
ax = plt.axes(projection = "3d")
ax.scatter3D(valores_x_fx[:,:1], valores_x_fx[:,1:2], valores_x_fx[:,2:3], 'gra
y', s = 0.1 )
ax.set_xlabel('x1', fontsize = 16)
ax.set_ylabel('x2', fontsize = 16)
ax.set_zlabel('f (x1,x2)', fontsize = 16)
plt.tick_params(labelsize = 10)
plt.show()

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
grafico(valores_x_fx)
```

```
Learning rate: 0.0001
Ponto mínimo = [0.72223396 0.5203204 ]
Função Rosenbrock no ponto mínimo = 0.07732336151538695
Derivada da função Rosenbrock no ponto mínimo = [-0.17953942 -0.2602
9838]
Número de iterações = 12384

Matriz com todos os pontos [x1, x2] e seus respectivos valores da fu
nção rosenbrock f(x1,x2) na descida do gradiente:
[[0.00000000e+00 0.00000000e+00 1.00000000e+00]
 [2.00000000e-04 0.00000000e+00 9.99600040e-01]
 [3.99960000e-04 8.00000000e-10 9.99200240e-01]
 ...
 [7.22198046e-01 5.20268335e-01 7.73433635e-02]
 [7.22216004e-01 5.20294368e-01 7.73333617e-02]
 [7.22233959e-01 5.20320400e-01 7.73233615e-02]]
```


Valores de f(x) nos passos da descida do gradiente
x1
x2
f (x1,x2)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.1
0.2
0.3
0.4
0.5
0.2
0.4
0.6
0.8
1.0



- Com o learning rate de 1.e-4, após 12384 iterações, o algoritmo de gradiente descendente implementado converge para o ponto de mínimo local [0.72223396, 0.5203204], onde o valor da função Rosenbrock é 0.07732336151538695 e sua derivada é [-0.17953942, -0.26029838].
- Nota-se, portanto, que o mínimo [0.72223396, 0.5203204] alcançado com o learning rate 1.e-4 é pior do que o mínimo [0.89731737, 0.8047416] alcançado com o learning rate 1.e-3, uma vez que sabemos ser o ponto [1, 1] o mínimo global da função Rosenbrock 2D.

Questão 1.3: Usand Learning rate grande (1.e-2)

In [6]:

```
learning_rate = 1e-2
tolerancia_epsilon = 1e-5
max_iter = 50000
x_inicial = np.array([0, 0])

x_min, num_iter, valores_x_fx = gradiente_descendente(derivada_rosenbrock, x_ini
cial, learning_rate, tolerancia_epsilon, max_iter)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Runt
imeWarning: overflow encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doing
imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: Runt
imeWarning: overflow encountered in double_scalars
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Runt
imeWarning: invalid value encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doing
imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: Runt
imeWarning: invalid value encountered in double_scalars
  import sys
```

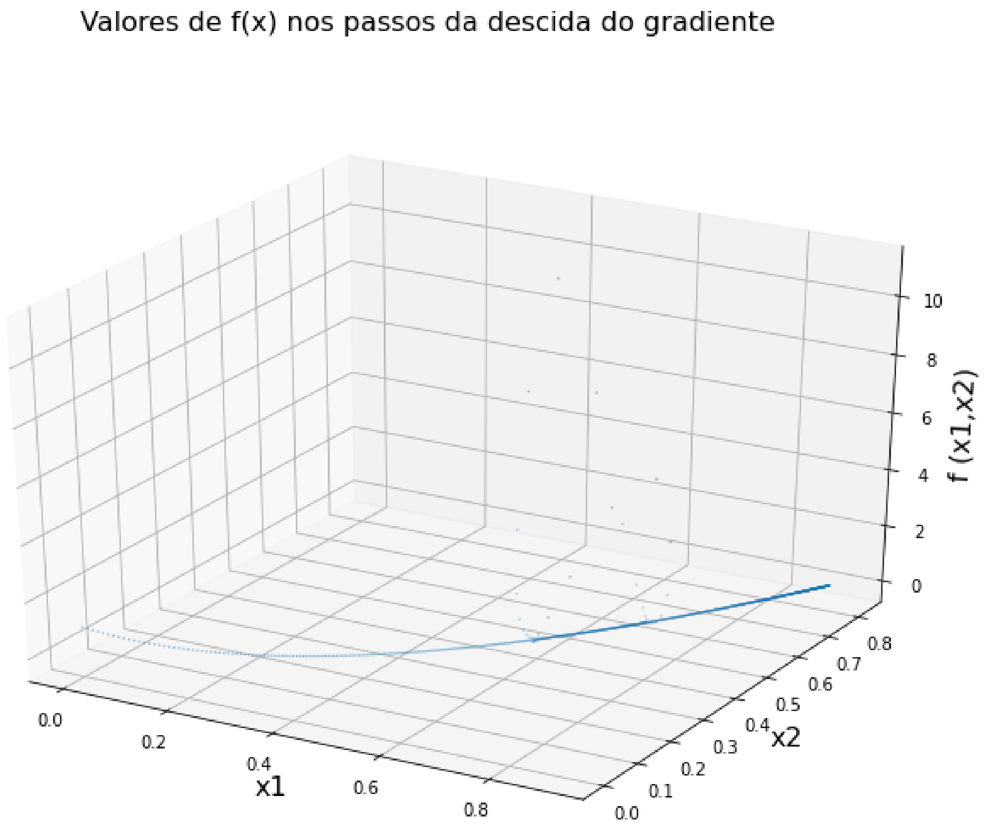
- Com o learning rate de 1.e-2, não alcançamos convergência na descida do gradiente e recebemos um erro na execução.

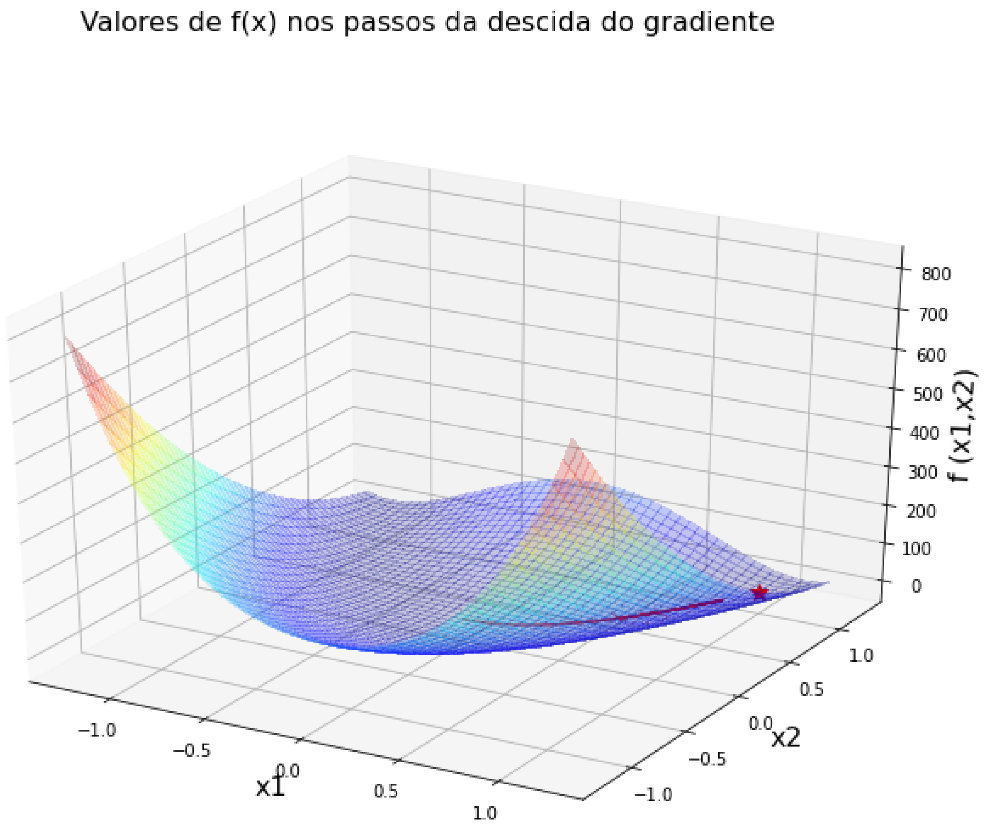
Questão 1.4: Política de redução do Learning rate

In [7]:

```
learning_rate_inicial = 5e-3
taxa_decaimento_lr = 0.001
tolerancia_epsilon = 1e-5
max_iter = 50000
x_inicial = np.array([0, 0])

# A política de redução do learning rate foi implementada na função gradiente_de
scendente() definida anteriormente
# Por padrão, na função gradiente_descendente(), taxa_decaimento_lr = 0
# A política será "ativada" apenas se taxa_decaimento_lr != 0 for passado explic
itamente na chamada da função
# Assim, chamaremos agora gradiente_descendente() com taxa_decaimento_lr = 0.001

x_min, num_iter, valores_x_fx = gradiente_descendente(derivada_rosenbrock, x_ini
cial, learning_rate_inicial, tolerancia_epsilon, max_iter, taxa_decaimento_lr)

print('Learning rate inicial:', learning_rate_inicial)
print('Ponto mínimo =', x_min)
print('Função Rosenbrock no ponto mínimo =', funcao_rosenbrock(x_min[0], x_min[1
]))
print('Derivada da função Rosenbrock no ponto mínimo =', derivada_rosenbrock(x_m
in[0], x_min[1]))
print('Número de iterações =', num_iter)

# Valores de [x1, x2] e f(x1,x2) nos passos da descida do gradiente
valores_x_fx = valores_x_fx.reshape(-1, 3)
print("\nMatriz com todos os pontos [x1, x2] e seus respectivos valores da funçã
o rosenbrock f(x1,x2) na descida do gradiente:")
print(valores_x_fx)
print('\n')

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
fig = plt.figure(figsize = (12, 8))
fig.suptitle('Valores de f(x) nos passos da descida do gradiente', fontsize = 16
)
ax = plt.axes(projection = "3d")
ax.scatter3D(valores_x_fx[:,:1], valores_x_fx[:,1:2], valores_x_fx[:,2:3], 'gra
y', s = 0.1 )
ax.set_xlabel('x1', fontsize = 16)
ax.set_ylabel('x2', fontsize = 16)
ax.set_zlabel('f (x1,x2)', fontsize = 16)
plt.tick_params(labelsize = 10)
plt.show()

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
grafico(valores_x_fx)
```

```
Learning rate inicial: 0.005
Ponto mínimo = [0.90380234 0.81645096]
Função Rosenbrock no ponto mínimo = 0.009270612669528767
Derivada da função Rosenbrock no ponto mínimo = [-0.04499818 -0.0815
428 ]
Número de iterações = 1468

Matriz com todos os pontos [x1, x2] e seus respectivos valores da fu
nção rosenbrock f(x1,x2) na descida do gradiente:
[[0.00000000e+00 0.00000000e+00 1.00000000e+00]
 [1.00000000e-02 0.00000000e+00 9.80101000e-01]
 [1.98881020e-02 9.99000000e-05 9.60628073e-01]
 ...
 [9.03698390e-01 8.16262601e-01 9.29066108e-03]
 [9.03750408e-01 8.16356853e-01 9.28062600e-03]
 [9.03802341e-01 8.16450958e-01 9.27061267e-03]]
```


Valores de f(x) nos passos da descida do gradiente
x1
x2
f (x1,x2)
0.0
0.2
0.4
0.6
0.8
0.1
0.3
0.5
0.7
0
2
4
6
8
10



- Implementando a política de redução do learning rate com valor inicial de 5.e-3, após 1468 iterações, obtemos ponto mínimo [0.90380234, 0.81645096]. Nesse ponto, a função Rosenbrock tem valor 0.009270612669528767 e sua derivada é [-0.04499818, -0.0815428].
- Esse resultado é bem compatível com o reportado para o learning rate de 1.e-3 sem política de redução, onde o algoritmo converge para o ponto de mínimo local [0.89731737, 0.8047416], exceto pela convergência mais lenta, de 3096 iterações.

Questão 2: Usando o tensorflow para calcular o gradiente

In [8]:

```
# Função Rosenbrock
def funcao_rosenbrock(x1, x2, a = 1, b = 100):
  return (a - x1)**2 + b * (x2 - x1**2)**2
```

In [9]:

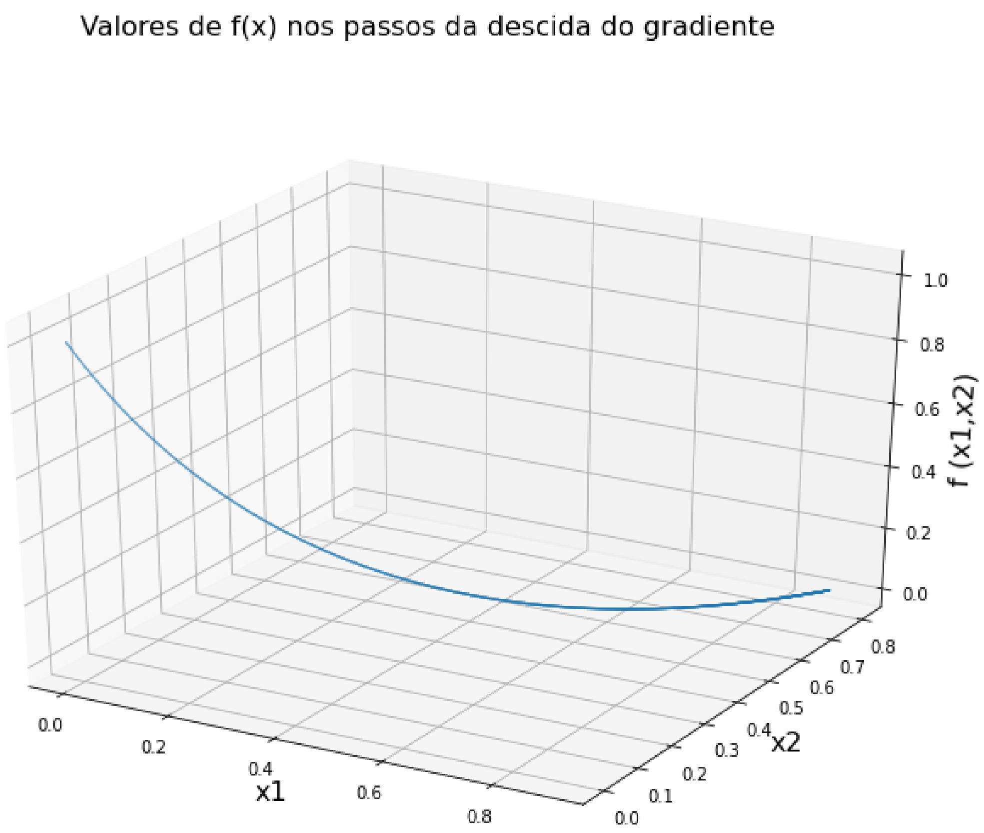
```
def gradiente_descendente_tensorflow(x1, x2):
  with tf.GradientTape(persistent = True) as tape:
    tape.watch(x1)
    tape.watch(x2)
    # Calcula a funcão Rosenbrock no ponto [x1,x2]
    y = funcao_rosenbrock(x1, x2)

  # Calcula o gradiente da funcão Rosenbrock no ponto [x1,x2]
  gradiente_x1 = tape.gradient(y, x1).numpy()
  gradiente_x2 = tape.gradient(y, x2).numpy()

  return gradiente_x1, gradiente_x2
```

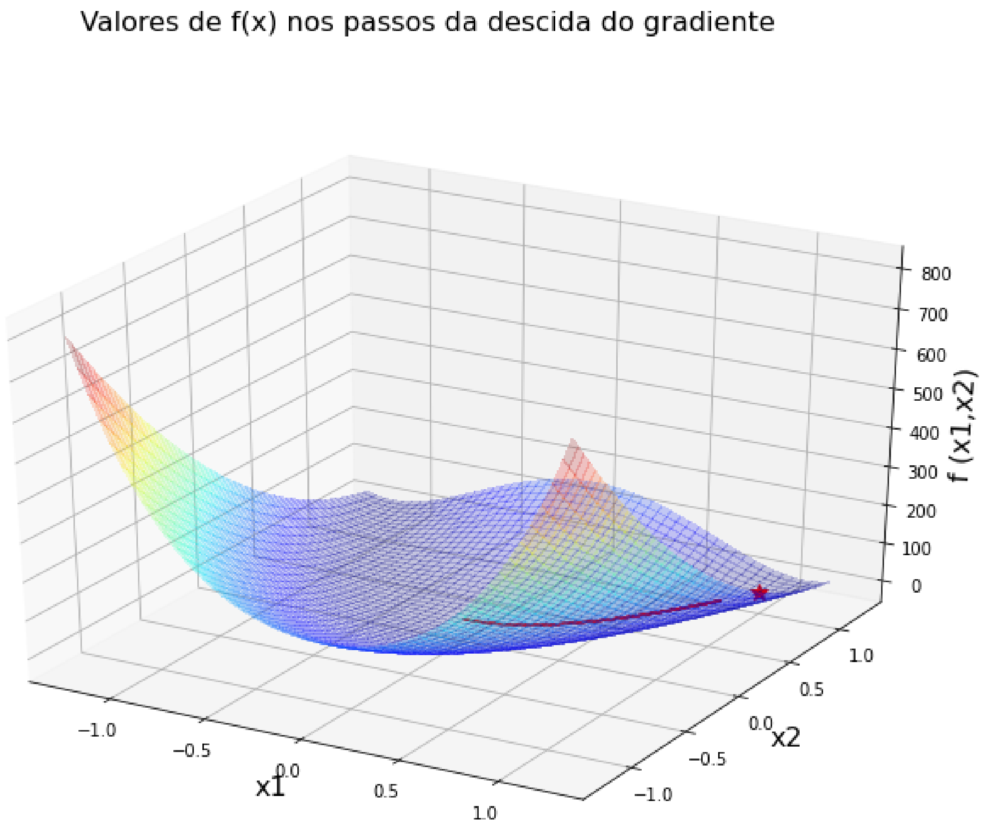
In [10]:

```
learning_rate = 1e-3
tolerancia_epsilon = 1e-5
max_iter = 50000

# Ponto inicial [x1,x2]
x1 = tf.Variable(0.0, trainable = True)
x2 = tf.Variable(0.0, trainable = True)

# Guarda os valores de [x1, x2, f(x1,x2)] no processo de descida do gradiente
valores_x_fx = []
valores_x_fx.append(np.array([x1.numpy(), x2.numpy(), funcao_rosenbrock(x1, x2
)]))

# Calcula o gradiente da funcão Rosenbrock no ponto inicial [x1,x2]
gradiente_x1, gradiente_x2 = gradiente_descendente_tensorflow(x1, x2)

# Atualiza o ponto inicial [x1,x2]
x1.assign_sub(gradiente_x1 * learning_rate)
x2.assign_sub(gradiente_x2 * learning_rate)

iteracao = 0

while iteracao < max_iter:
  x1_anterior = x1.numpy()
  x2_anterior = x2.numpy()

  # Guarda os valores de [x1, x2, f(x1,x2)] no processo de descida do gradiente
  valores_x_fx.append(np.array([x1.numpy(), x2.numpy(), funcao_rosenbrock(x1, x2
)]))

  # Calcula o gradiente da funcão Rosenbrock no ponto [x1,x2]
  gradiente_x1, gradiente_x2 = gradiente_descendente_tensorflow(x1, x2)

  # Atualiza o ponto [x1,x2]
  x1.assign_sub(gradiente_x1 * learning_rate)
  x2.assign_sub(gradiente_x2 * learning_rate)

  # Define outra condição de parada além do número máximo de iterações
  tolerancia = np.abs(funcao_rosenbrock(x1, x2) - funcao_rosenbrock(x1_anterior,
x2_anterior))

  iteracao = iteracao + 1

  if tolerancia < tolerancia_epsilon:
    break

print('Learning rate:', learning_rate)
print('Ponto mínimo =', valores_x_fx[len(valores_x_fx)-1][:2])
print('Função Rosenbrock no ponto mínimo =', valores_x_fx[len(valores_x_fx)-1][2
:])
print('Derivada da função Rosenbrock no ponto mínimo =', derivada_rosenbrock(val
ores_x_fx[len(valores_x_fx)-1][0], valores_x_fx[len(valores_x_fx)-1][1]))
print('Número de iterações =', len(valores_x_fx))

# Valores de [x1, x2] e f(x1,x2) nos passos da descida do gradiente
valores_x_fx = np.array(valores_x_fx)
valores_x_fx = valores_x_fx.reshape(-1, 3)
print("\nMatriz com todos os pontos [x1, x2] e seus respectivos valores da funçã
o rosenbrock f(x1,x2) na descida do gradiente:")
```

```
print(valores_x_fx)
print('\n')

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
fig = plt.figure(figsize = (12, 8))
fig.suptitle('Valores de f(x) nos passos da descida do gradiente', fontsize = 16
)
ax = plt.axes(projection = "3d")
ax.scatter3D(valores_x_fx[:,:1], valores_x_fx[:,1:2], valores_x_fx[:,2:3], 'gra
y', s = 0.1 )
ax.set_xlabel('x1', fontsize = 16)
ax.set_ylabel('x2', fontsize = 16)
ax.set_zlabel('f (x1,x2)', fontsize = 16)
plt.tick_params(labelsize = 10)
plt.show()

# Plot dos valores de f(x1,x2) nos passos da descida do gradiente
grafico(valores_x_fx)
```

```
Learning rate: 0.001
Ponto mínimo = [0.89731705 0.80474097]
Função Rosenbrock no ponto mínimo = [0.01056288]
Derivada da função Rosenbrock no ponto mínimo = [-0.04854135 -0.0873
8525]
Número de iterações = 3096

Matriz com todos os pontos [x1, x2] e seus respectivos valores da fu
nção rosenbrock f(x1,x2) na descida do gradiente:
[[0.0000000e+00 0.0000000e+00 1.0000000e+00]
 [2.0000001e-03 0.0000000e+00 9.9600405e-01]
 [3.9959969e-03 8.0000012e-07 9.9202394e-01]
 ...
 [8.9721984e-01 8.0456609e-01 1.0582892e-02]
 [8.9726841e-01 8.0465358e-01 1.0572878e-02]
 [8.9731705e-01 8.0474097e-01 1.0562876e-02]]
```





- Utilizando o Tensorflow para computar automaticamente o gradiente, com learning rate 1.e-3, após 3096 iterações, o algoritmo converge para o ponto de mínimo local [0.89731705, 0.80474097], onde o valor da função Rosenbrock é 0.01056288 e sua derivada é [-0.04854135, -0.08738525].
- Ressalta-se que esse resultado é praticamente idêntico ao reportado para o learning rate de 1.e-3 da Questão 1.1, onde computamos o gradiente com a função implementada na mão. Nos dois caso, o algoritmo levou 3096 iterações para convergir, tendo diferido apenas nas últimas casas decimais. Na Questão 1.1, obtivemos ponto mínimo [0.89731737, 0.8047416], onde o valor da função Rosenbrock é 0.010562807758210266 e sua derivada é [-0.04856236, -0.08737315]