In [78]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [79]:
```
# há duas variáveis: tempo e distância, onde distância, é decomposta em rang

### propriedades

massa = 17.60 #kg
g = -9.81 # m/s²
v0 = 18 # m/s

medicoes = {
    "t1":[0.3, 4.96],
    "t2":[0.6, 9.03],
    "t3":[0.7, 10.2],
    "t4":[0.9, 12.23],
    "t5":[1.4, 15.59],
    "t6":[2.2, 15.86],
    "t7":[2.7, 12.84],
    "t8":[3.0, 9.85],
    "t9":[3.2, 7.37],
    "t10":[3.5, 2.91]
}
```

Tabela experimental exigida no tópico 5

In [80]:
```
df = pd.DataFrame(medicoes)
df = df.T
df.rename(columns={0:"tempo (s)", 1:"distância (m)"}, inplace=True)
df
```

Out[80]:

	tempo (s)	distância (m)
t1	0.3	4.96
t2	0.6	9.03
t3	0.7	10.20
t4	0.9	12.23
t5	1.4	15.59
t6	2.2	15.86
t7	2.7	12.84
t8	3.0	9.85
t9	3.2	7.37
t10	3.5	2.91

Condições iniciais (tóipico 6):

V0 = 18 # m/s j g = -9,81 # m/s² j (aceleração da gravidade)

origem = (0,0)

MRUV

função horária da posição (tópico 7)

y = v0 *t + 0.5* g *t ** 2 y = 18* t - 4.905 * t ** 2 # (m)

função horária da velocidade (tópico 8)

v = np.sqrt(v0 * *2 + 2* g * y) # (m/s)

função horária da aceleração (tópico 9)

a = -9.81 # (m/s²)

Tabela experimental exigida no tópico 10

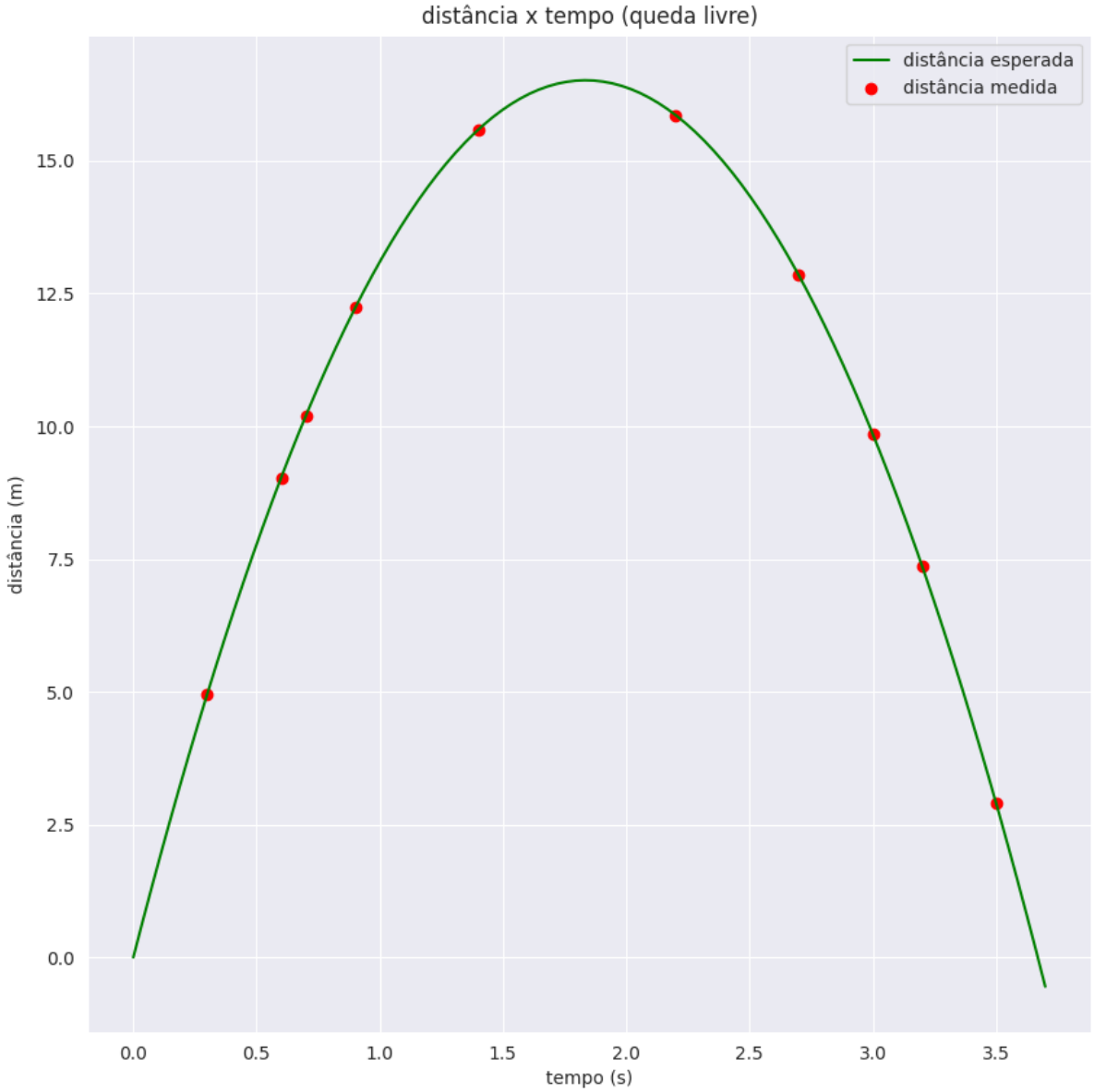
In [81]:
```
medicoes_velocidade = list(map(lambda d: np.sqrt(v0 ** 2 + 2 * g * d[1]), me
df['velocidade (m/s)'] = medicoes_velocidade
df['aceleração (m/s²)'] = -9.81
df
```

Out[81]:

	tempo (s)	distância (m)	velocidade (m/s)	aceleração (m/s²)
t1	0.3	4.96	15.056055	-9.81
t2	0.6	9.03	12.117401	-9.81
t3	0.7	10.20	11.129960	-9.81
t4	0.9	12.23	9.167737	-9.81
t5	1.4	15.59	4.257253	-9.81
t6	2.2	15.86	3.581452	-9.81
t7	2.7	12.84	8.489947	-9.81
t8	3.0	9.85	11.434291	-9.81
t9	3.2	7.37	13.394051	-9.81
t10	3.5	2.91	16.337252	-9.81

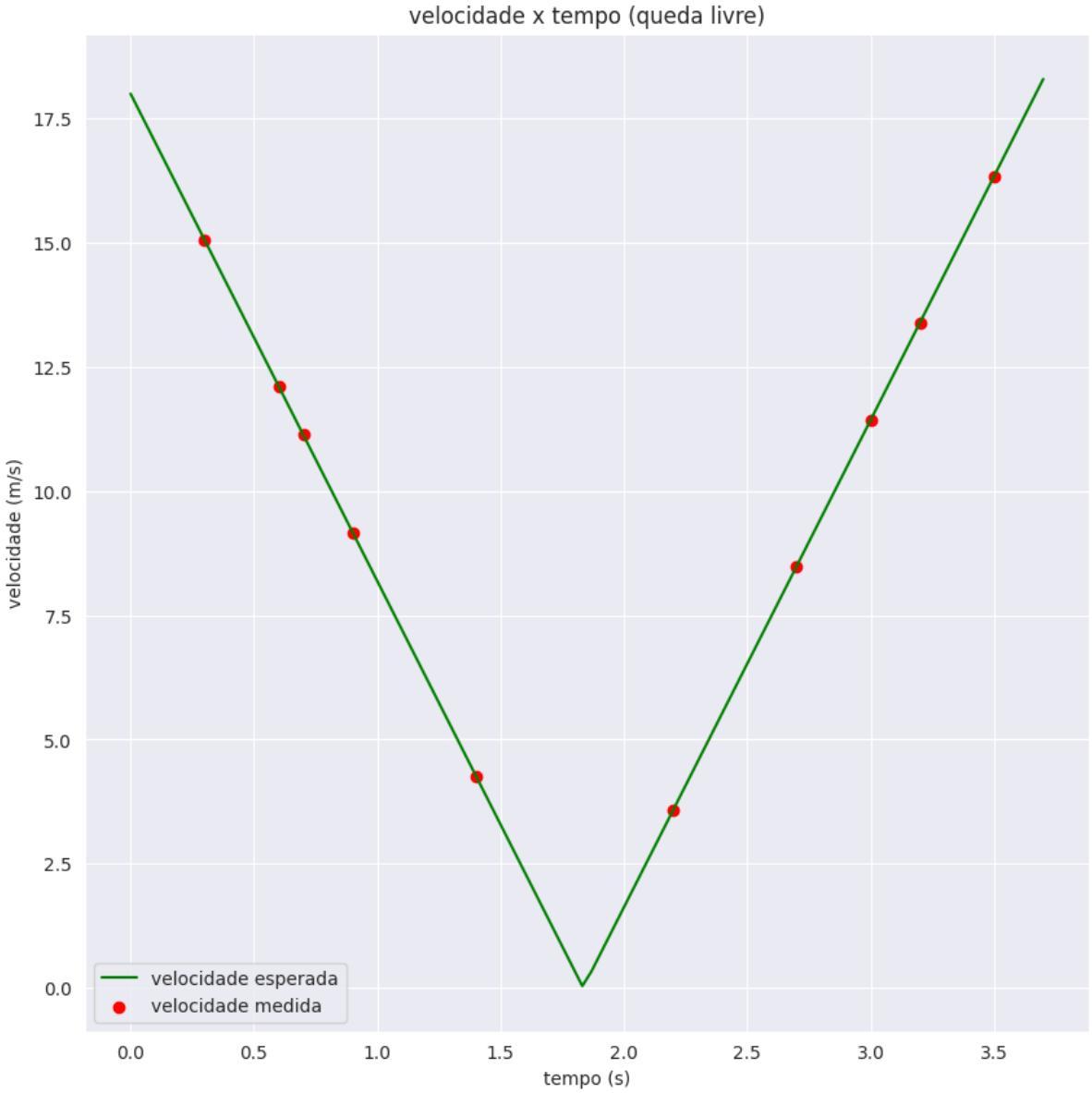
Gráficos: poisção x tempo (tópico 11)

In [82]:
```
d_waited = list(map(lambda t: v0* t + 0.5 * g * t ** 2, np.linspace(0, 3.7,
ax, fig = plt.subplots(figsize=(10, 10))
plt.plot(np.linspace(0, 3.7, 100), d_waited, label="distância esperada", col
plt.scatter(df['tempo (s)'], df['distância (m)'], label="distância medida",
plt.xlabel("tempo (s)")
plt.ylabel("distância (m)")
plt.title("distância x tempo (queda livre)")
plt.legend()
plt.show()
```



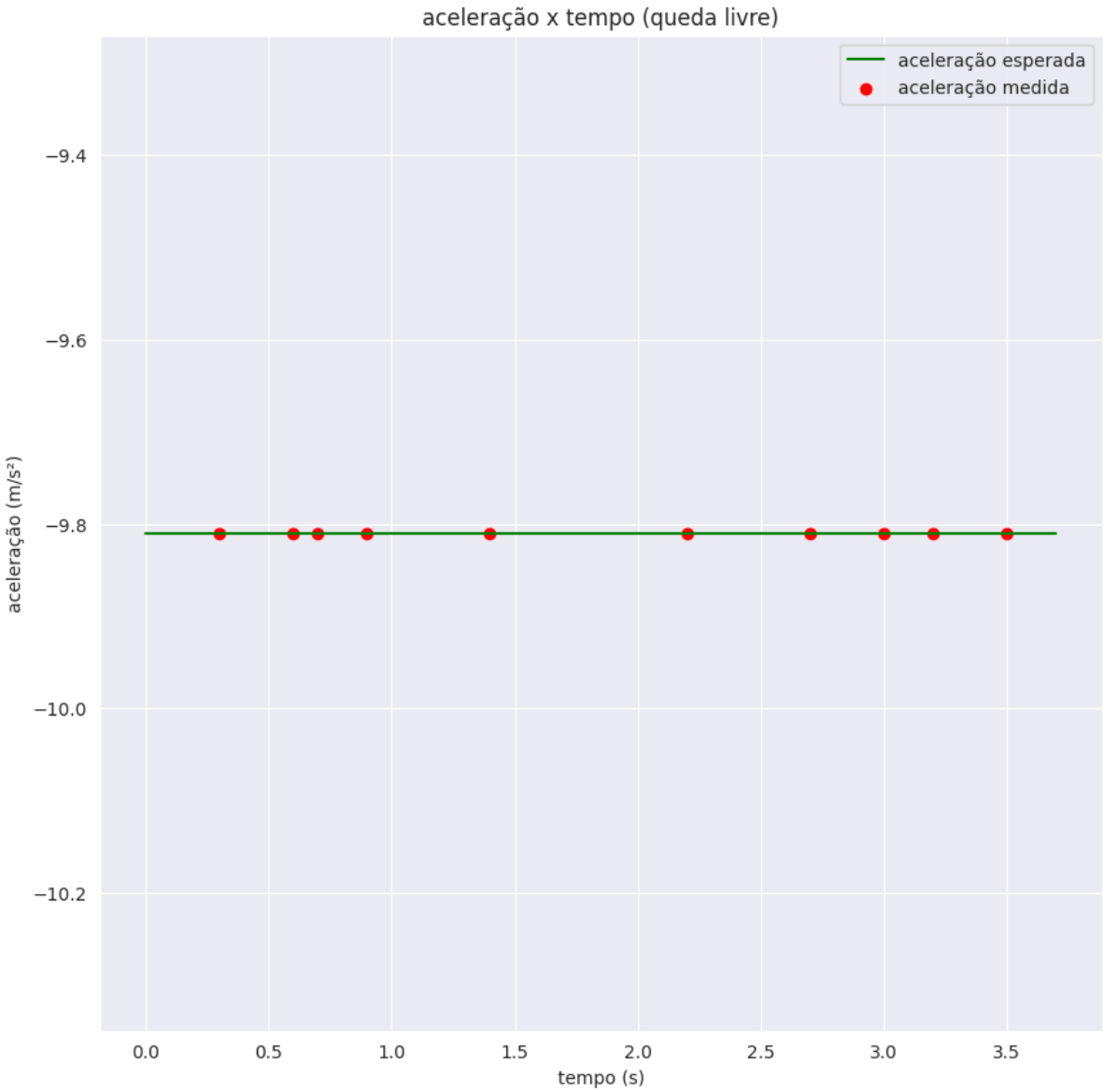
Gráficos: velocidade x tempo (tópico 12)

In [83]:
```
v_waited = list(map(lambda d: np.sqrt(v0 ** 2 + 2 * g * d), d_waited))

ax2, fig2 = plt.subplots(figsize=(10, 10))
plt.plot(np.linspace(0, 3.7, 100), v_waited, label="velocidade esperada", co
plt.scatter(df['tempo (s)'], df['velocidade (m/s)'], label="velocidade medic
plt.xlabel("tempo (s)")
plt.ylabel("velocidade (m/s)")
plt.title("velocidade x tempo (queda livre)")
plt.legend()
plt.show()
```



Gráficos: aceleração x tempo (tópico 13)

In [84]:
```
ax3, fig3 = plt.subplots(figsize=(10, 10))

plt.plot(np.linspace(0, 3.7, 100), np.full(100, -9.81), label="aceleração es
plt.scatter(df['tempo (s)'], df['aceleração (m/s²)'], label="aceleração medi
plt.xlabel("tempo (s)")
plt.ylabel("aceleração (m/s²)")
plt.title("aceleração x tempo (queda livre)")
plt.legend()
plt.show()
```


aceleração x tempo (queda livre)
aceleração esperada
aceleração medida
−9.4
−9.6
−9.8
−10.0
−10.2
aceleração (m/s²)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
tempo (s)

Perguntas

1. O referencial adotado foi a base do canhão, assim considerando a origem como o ponto de lançamento do projétil.

2. O sistema de coordenadas escolhido foi o cartesiano, com escala em metros e segundos, orientado para o eixo x para a direita e para o eixo y para cima.

3. Os gráficos gerados estão no tópico 11, 12 e 13.

4. No caso em que o gráfico está transladado para além do alcance máximo do projétil, os valores de posição são negativos, pois o projétil está abaixo da origem. As curvas dos gráficos permaneceriam as mesmas, apenas o gráfico de posição x tempo seria transladada para baixo.