Questão 1

Item A)

Recursos	Produto A	Produto B	Disponibilidade do recurso
Matéria Prima	20kg/kg	5kg/kg	9500kg/semana
Tempo	0,04hr/kg	0,12hr/kg	40hs/semana
Armazenamento	1kg	1kg	550kg/semana

|Lucro|45|20

Formulação total do problema de programação linear:

Função Objetivo: $Z = 45x_1 + 20x_2$

Matéria Prima: $20x_1 + 5x_2 \leq 9500$

Tempo: $0,040x_1 + 0,12x_2 \leq 40$

Armazenamento: $x_1 + x_2 \leq 550$

Restrições Lógicas: $x_1, x_2 \geq 0$

Onde x1 é quantiade de kilos do Produto A e x2 é quantiade de kilos do Produto B

Item B)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def funcao_obj(x1,z):
  x2 = (z-(45*x1))/20
  return (x1,x2)
```

In [3]:
```
def funcao_obj_2(x2,z):
    x1 =(z-(20*x2))/45
    return (x1,x2)
```

In [4]:
```
def rest_1(x1):
    return (9500-20*x1)/5
```

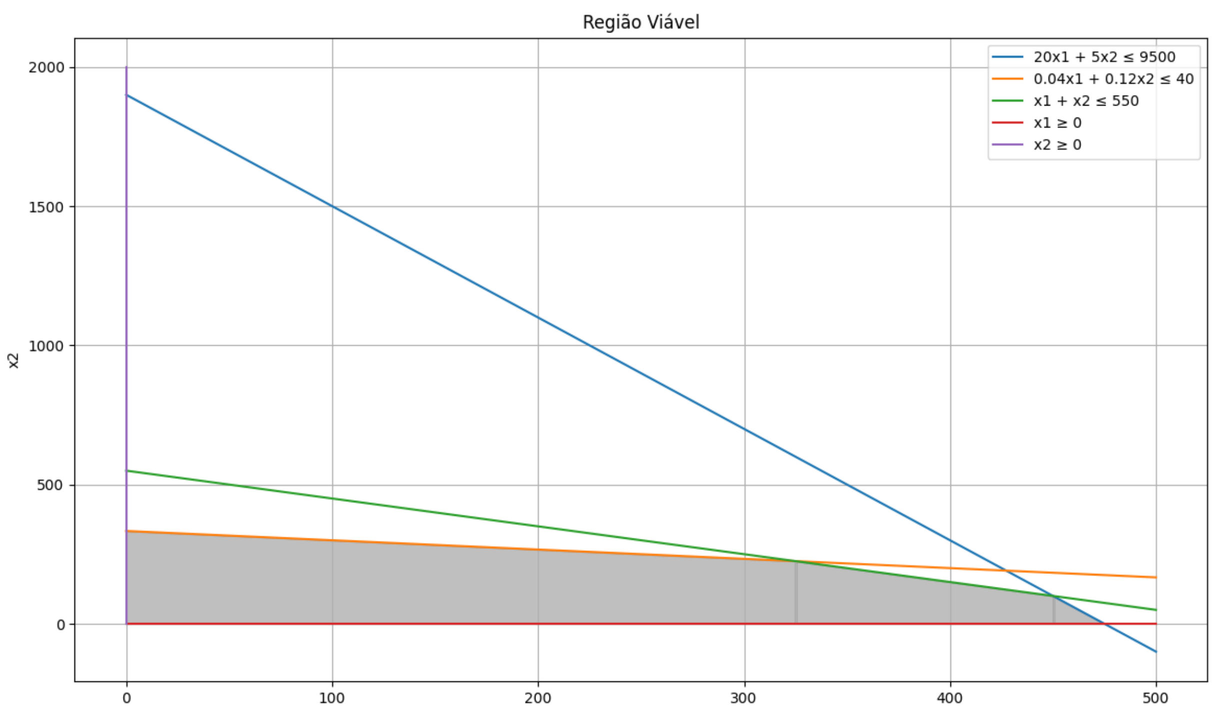
In [5]:
```
def rest_2(x1):
    return (40-0.04*x1)/0.12
```

In [6]:
```
def rest_3(x1):
    return 550-x1
```

In [7]:
```
def rest_4(x1):
    return 0*x1
```

In [8]:
```
x1 = np.linspace(0,500,1000)
x2 = np.linspace(0,2000,1000)
y1 = rest_1(x1)
y2 = rest_2(x1)
y3 = rest_3(x1)
y4 = rest_4(x1)
y5 = rest_4(x1)

plt.figure(figsize=(14, 8))  # Define o tamanho do gráfico
plt.grid()
plt.plot(x1, y1, label='20x1 + 5x2 ≤ 9500')
plt.plot(x1, y2, label='0.04x1 + 0.12x2 ≤ 40')
plt.plot(x1, y3, label='x1 + x2 ≤ 550')
plt.plot(x1,y4, label='x1 ≥ 0')
plt.plot(y5,x2, label='x2 ≥ 0')
plt.fill_between(x1, 0, y1, where=(y1 <= y3) & (y1 >= 0) & (x1 >= 0) , color
plt.fill_between(x1, 0, y2, where=(y2 >= 0) & (x1 >= 0) & (y2 <= y3), color=
plt.fill_between(x1, 0, y3, where=(y3 >= 0) & (x1 >= 0) & (y3 <= y1) & (y3 <

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Região Viável')

plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x7f2f469cf850>



Os pontos de insterção da região de convergência

Podemos perceber que para resolver as seguintes equações:

1. $\frac{9500-20x_1}{5} = 550 - x_1$

2. $\frac{40-0.04x_1}{0.12} = 550 - x_1$

3. $\frac{9550-20x_1}{5} = 0$

4. $\frac{40-0.12x_2}{0.04} = 0$

Resolvendo essas equações a gente obtem o seguinte os seguinte:

1. $x1 = 450$
2. $x1 = 325$
3. $x1 = 477.5$
4. $x2 \approx 333.3$

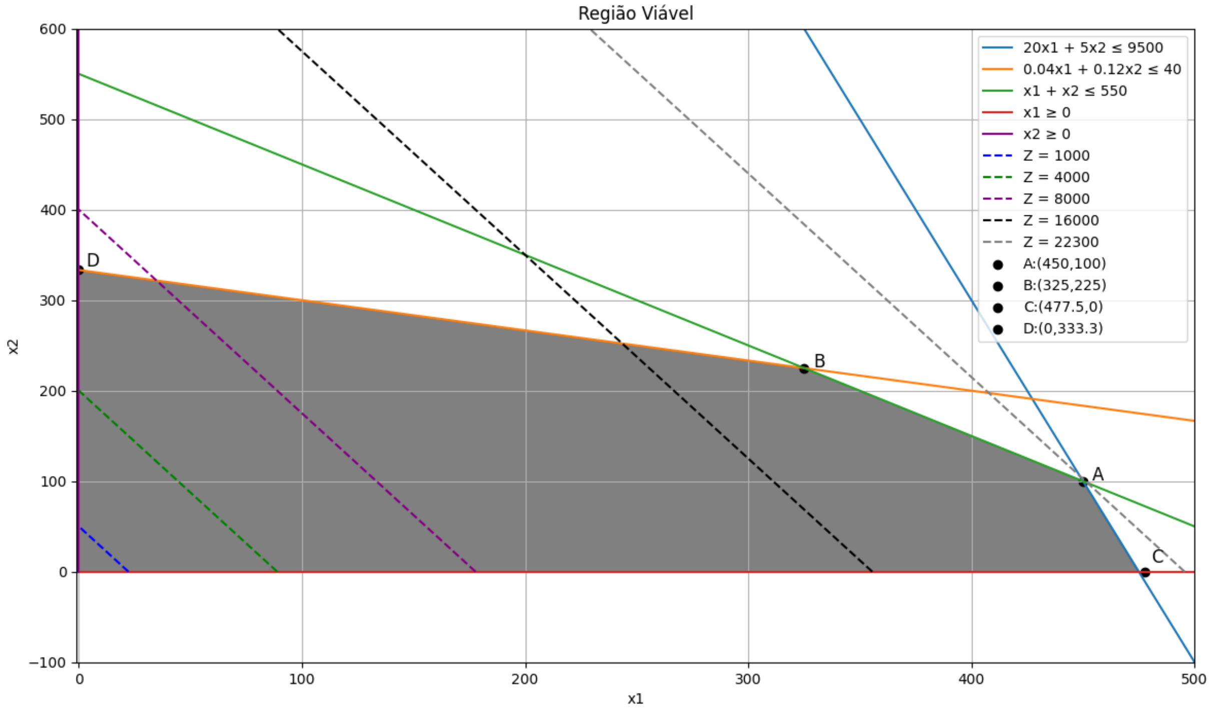
Aplicando os valres de Equação 1 e 2 em x1+x2 = 550 para encontrar o valor de x2:

1. $$x2 = 550 - 450\, x2 = 100$$
2. $$x2 = 550 - 325\, x2 = 225$$

Colocando os pares ordenados no gráfico para encontrar a solução ótima.

In [9]:
```
x1 = np.linspace(0,500,1000)
x2 = np.linspace(0,2000,1000)
x3 = np.linspace(0,100,1000)

y1 = rest_1(x1)
y2 = rest_2(x1)
y3 = rest_3(x1)
y4 = rest_4(x1)
y5 = rest_4(x1)

#Definindo as linhas da função objetivo
z1 = funcao_obj(0,1000) #z =1000
r1 = funcao_obj_2(0,1000)
z2 = funcao_obj(0,4000)#z =4000
r2 = funcao_obj_2(0,4000)
z3 = funcao_obj(0,8000)#z =8000
r3 = funcao_obj_2(0,8000)
z4 = funcao_obj(0,16000)#z =16000
r4 = funcao_obj_2(0,16000)
z5 = funcao_obj(0,22300)#z =22300
r5 = funcao_obj_2(0,22300)
```

```python
plt.figure(figsize=(14, 8))  # Define o tamanho do gráfico
plt.grid()

plt.plot(x1, y1, label='20x1 + 5x2 ≤ 9500')
plt.plot(x1, y2, label='0.04x1 + 0.12x2 ≤ 40')
plt.plot(x1, y3, label='x1 + x2 ≤ 550')
plt.plot(x1,y4, label='x1 ≥ 0')
plt.plot(y5,x2, label='x2 ≥ 0',color='purple')
plt.plot(r1, z1,color='blue',linestyle = 'dashed', label='Z = 1000')
plt.plot(r2, z2,color='green',linestyle = 'dashed',label='Z = 4000' )
plt.plot(r3, z3,color='purple',linestyle = 'dashed', label='Z = 8000')
plt.plot(r4, z4,color='black',linestyle = 'dashed', label='Z = 16000')
plt.plot(r5, z5,color='gray',linestyle = 'dashed', label='Z = 22300')

plt.fill_between(x1, 0, y1, where=(y1 <= y3) & (y1 >= 0) & (x1 >= 0) , color
plt.fill_between(x1, 0, y2, where=(y2 >= 0) & (x1 >= 0) & (y2 <= y3), color=
plt.fill_between(x1, 0, y3, where=(y3 >= 0) & (x1 >= 0) & (y3 <= y1) & (y3 <

plt.scatter(450, 100, color='black', label='A:(450,100)')
plt.scatter(325, 225, color='black', label='B:(325,225)')
plt.scatter(477.5, 0, color='black', label='C:(477.5,0)')
plt.scatter(0, 333.3, color='black', label='D:(0,333.3)')

plt.text(450+4, 100+1, 'A', fontsize=12, color='black')
plt.text(325+4, 225+1, 'B', fontsize=12, color='black')
plt.text(477.5+3, 0+10, 'C', fontsize=12, color='black')
plt.text(0+3, 333.3+4, 'D', fontsize=12, color='black')

plt.xlim(-1, 500)
plt.ylim(-100, 600)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Região Viável')

plt.legend()
graf = plt.show()
```



Traçando as funções objetivo podemos ver que elas vão se distanciando da origem

Na Situação limite, o ponto A é intercpetado pela reta da função objetivo. Então podemos presumir que as os valores de x1 e x2 no ponto A são os valores para se obter o valor ótimo da função objetivo

Item C)

Para aplicar o método simplex nesse problema precisamos primeiro colocar variavéis de folga nas inequações. Essa variavéis servem para

$$Z - 45x_1 - 20x_2 = 0$$

$$20x_1 + 5x_2 + x_3 = 9500$$

$$0,040x_1 + 0,12x_2 + x_4 = 40$$

$$x_1 + x_2 + x_5 = 550$$

$$x_1, x_2 \geq 0$$

Variáveis	Z	X1	X2	X3	X4	X5	Solução
Z	1	-45	-20	0	0	0	0
X3	0	20	5	1	0	0	9500
X4	0	0.04	0.12	0	1	0	40
X5	0	1	1	0	0	1	550

In [10]:
```python
def simplex(A):
    ms = np.array(A, dtype='float')
    nLinhas, nColunas = ms.shape
    var_folga = nColunas - nLinhas

    basicas = ["Z"] + [f"X{i+var_folga}" for i in range(1, nLinhas)]
    naoBasicas = ["Z"] + [f"X{i}" for i in range(1, nColunas-1)] + ["SOL"]
    copiaColunaSolucao = np.empty(nLinhas, dtype=float)

    def copiar_coluna_solucao():
        for i in range(0, nLinhas):
            copiaColunaSolucao[i] = ms[i][nColunas-1]

    def identificar_coluna_pivo():
        menor = 9999999999
        cMenor = -1

        for c in range(0, nColunas):
            if ms[0][c] < 0 and ms[0][c] < menor:
                menor = ms[0][c]
                cMenor = c

        return cMenor

    def identificar_linha_pivo(cPivo):
        menor = 99999999
        lMenor = -1

        for l in range(1, nLinhas):
            if ms[l][cPivo] != 0:
                aux = (copiaColunaSolucao[l] / ms[l][cPivo])
                if aux < menor and aux >= 0:
                    menor = aux
                    lMenor = l

        return lMenor

    def dividir_linha_pivo_pelo_elemento(lPivo, cPivo):
        pivo = ms[lPivo][cPivo]
        if pivo != 0:
            for c in range(nColunas):
                ms[lPivo][c] = (ms[lPivo][c] / pivo)

    def eliminar_coef_da_coluna_pivo(lPivo, cPivo):
        for l in range(nLinhas):
            if l != lPivo:
                multiplicador = -ms[l][cPivo]
                for c in range(nColunas):
                    ms[l][c] += (multiplicador * ms[lPivo][c])

    existeNegativo = 1
    while existeNegativo == 1:
        copiar_coluna_solucao()
        cPivo = identificar_coluna_pivo()
```

```
        if cPivo != -1:  # Critério de parada
            lPivo = identificar_linha_pivo(cPivo)
            dividir_linha_pivo_pelo_elemento(lPivo, cPivo)
            basicas[lPivo] = naoBasicas[cPivo]
            eliminar_coef_da_coluna_pivo(lPivo, cPivo)
        else:
            existeNegativo = 0

    return (ms,basicas)
```

In [11]:
```
coef = [[1,-45,-20,0,0,0,0],
           [0,20,5,1,0,0,9500],
           [0,0.040,0.12,0,1,0,40],
           [0,1,1,0,0,1,550]] #Coeficientes das variaveis

[tabela,var_basicas] = simplex(coef)
np.set_printoptions(precision=2, suppress=True)
print(tabela)
```

```
[[    1.        0.        0.        1.67      0.       11.67 22250.  ]
 [    0.        1.        0.        0.07      0.       -0.33   450.  ]
 [    0.        0.        0.        0.01      1.       -0.15    10.  ]
 [    0.        0.        1.       -0.07      0.        1.33   100.  ]]
```

In [12]:
```
nLinhas, nColunas = tabela.shape
for i in range(nLinhas):
  print(var_basicas[i],"=",tabela[i][nColunas-1])
```

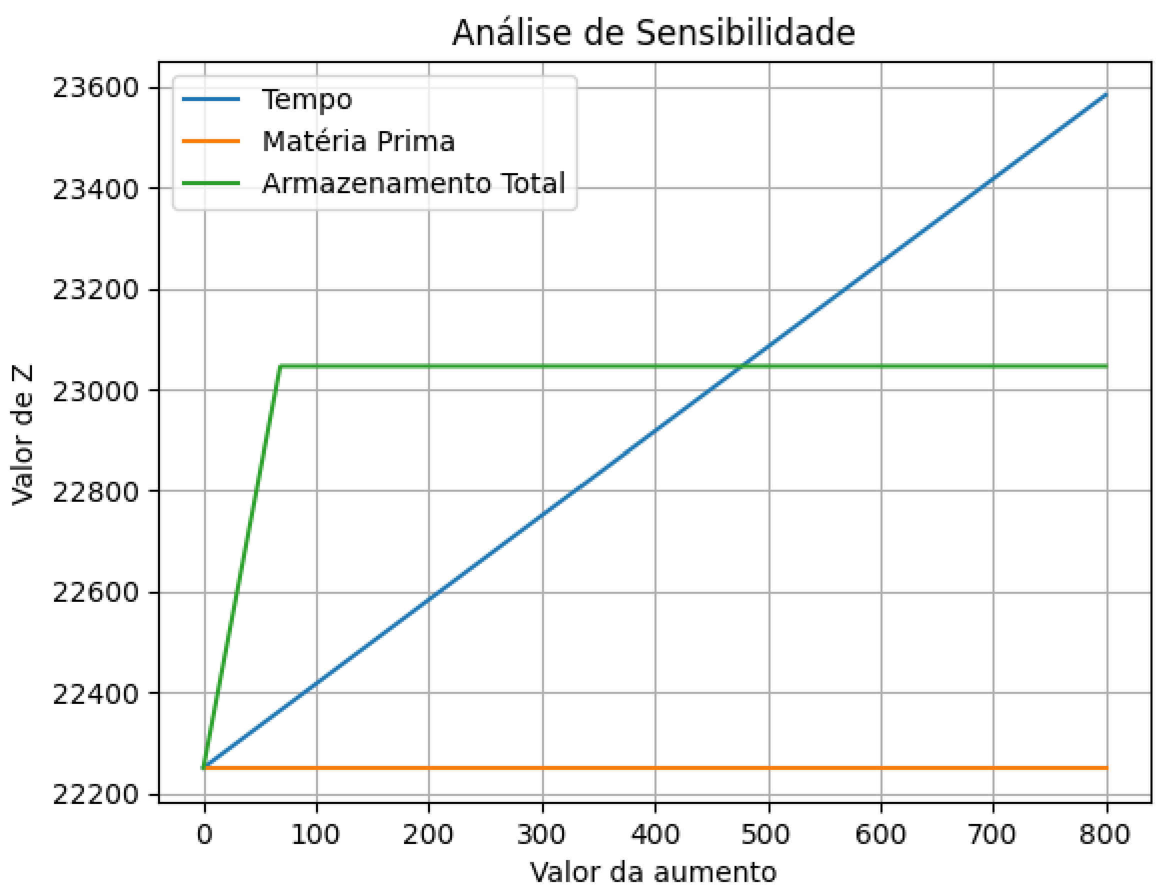
```
Z = 22250.0
X1 = 450.0
X5 = 10.0
X2 = 100.0
```

Variáveis	Z	X1	X2	X3	X4	X5	Solução
Z	1	0	0	1.67	0	11.67	22250
X1	0	1	0	0.07	0	-0.33	450
X4	0	0	0	0.01	1	-0.15	10
X2	0	0	1	-0.07	0	1.33	100

Item D)

In [13]:
```
valores_variados = np.linspace(0,800,1000)

resultados_1 = []
resultados_2 = []
resultados_3 = []

for valor in valores_variados:
    aux = np.copy(coef)
    aux[1][-1] += valor #Tempo
    [tabela, var_basicas] = simplex(aux)
    resultados_1.append(tabela[0][-1])
```

```
    aux = np.copy(coef)
    aux[2][-1] += valor #Gás bruto
    [tabela, var_basicas] = simplex(aux)
    resultados_2.append(tabela[0][-1])

    aux = np.copy(coef)
    aux[3][-1] += valor #Gás bruto
    [tabela, var_basicas] = simplex(aux)
    resultados_3.append(tabela[0][-1])

plt.plot(valores_variados, resultados_1,label='Tempo')
plt.plot(valores_variados, resultados_2,label='Matéria Prima')
plt.plot(valores_variados, resultados_3,label='Armazenamento Total')
plt.xlabel('Valor da aumento')
plt.ylabel('Valor de Z')
plt.title('Análise de Sensibilidade')
plt.legend()
plt.grid(True)
```



Pelo gráfico de lucro obtido aumentado cada restrição que a empresa tenha, podemos ver que logo de cara para um aumento muito pouco dos recursos, aumentar o Armazenamento total leva a um aumento maior. Contudo, entre aumentos maiores que 500 podemos ver que aumentar o tempo de produção é que leva a um lucro maior, visto que o lucro com aumento do Armazenamento tende a estabilizar, enquanto o lucro aumentando o tempo só continua a crescer.

Porém, um aumento de 500 horas a mais de produção na semana é práticamente invíavel, devido ao fato de que não existe tempo o sucificiente para isso, logo para se obter um lucro maior é recomendável aumentar o armazenamento total dos produtos.

Também podemos ver que aumentar a matéria prima não gera nem aumento significativo de lucro se comparado com as outras condições.

Questão 2

Item A)

|Recursos|Regular|Premium|Supreme|Disponibilidade do recurso| |------------|------------|------------|------------|------------| |Gás Bruto|7 m³/ton|11 m³/ton|15 m³/ton|154 m³/semana| |Tempo de produção|10 hr/ton|8 hr/ton|12 hr/ton|80 hr/semana| |Armazenamento|9 ton|6 ton|5 ton| |Lucro|150 reais/ton|175 reais/ton|250 reais/ton

Formulação total do problema de programação linear:

Função Objetivo: $Z = 150x_1 + 175x_2 + 250x_3$

Tempo: $10x_1 + 8x_2 + 12x_3 \leq 80$

Gás Bruto: $7x_1 + 11x_2 + 15x_3 \leq 154$

$x_1 + \leq 9$

$x_2 + \leq 6$

$x_3 + \leq 5$

$x_1, x_2, x_3 \geq 0$

Item B)

Para aplicar o método simplex nesse problema precisamos primeiro colocar variavéis de folga nas inequações assim como na questão 1.

Função Objetivo: $Z - 150x_1 - 175x_2 - 250X_3 = 0$

Tempo: $10x_1 + 8x_2 + 12x_3 + x_4 = 80$

Gás Bruto: $7x_1 + 11x_2 + 15x_3 + x5 = 154$

$x_1 + x_6 = 9$

$x_2 + x_7 = 6$

$x_3 + x_8 = 5$

$x_1, x_2, x_3 \geq 0$

Agora que temos as equações com as variavéis de folga, podemos colocar os coeficientes numa tabela. Apartir dessa tabela a gente vai implementar o método simplex

Tabela para o problema da Questão 2:

Variáveis	Z	X1	X2	X3	X4	X5	X6	X7	X8	Solução
Z	1	-150	-175	-250	0	0	0	0	0	0
X4	0	10	8	12	1	0	0	0	0	80
X5	0	7	11	15	0	1	0	0	0	154
X6	0	1	0	0	0	0	1	0	0	9
X7	0	0	1	0	0	0	0	1	0	6
X8	0	0	0	1	0	0	0	0	1	5

In [14]:
```
coef1 = [[1,-150,-175,-250,0,0,0,0,0,0], [0,10,8,12,1,0,0,0,0,80], [0,7,11,1

[tabela,var_basicas] = simplex(coef1)
np.set_printoptions(precision=2, suppress=True)
print(tabela)
```

```
[[   1.      58.33     0.       0.      20.83     0.       0.       8.33     0.
  1716.67]
 [   0.       0.       1.       0.       0.       0.       0.       1.       0.
     6.  ]
 [   0.      -5.5      0.       0.      -1.25     1.       0.      -1.       0.
    48.  ]
 [   0.       1.       0.       0.       0.       0.       1.       0.       0.
     9.  ]
 [   0.      -0.83     0.       0.      -0.08     0.       0.       0.67     1.
     2.33]
 [   0.       0.83     0.       1.       0.08     0.       0.      -0.67     0.
     2.67]]
```

In [15]:
```
nLinhas, nColunas = tabela.shape
for i in range(nLinhas):
  print(var_basicas[i],"=",tabela[i][nColunas-1])
```

```
Z = 1716.6666666666667
X2 = 6.0
X6 = 48.0
X7 = 9.0
X8 = 2.3333333333333335
X3 = 2.6666666666666665
```

Variáveis	Z	X1	X2	X3	X4	X5	X6	X7	X8	Solução
Z	1	58.33	0	0	20.83	0	0	8.33	0	1716.67

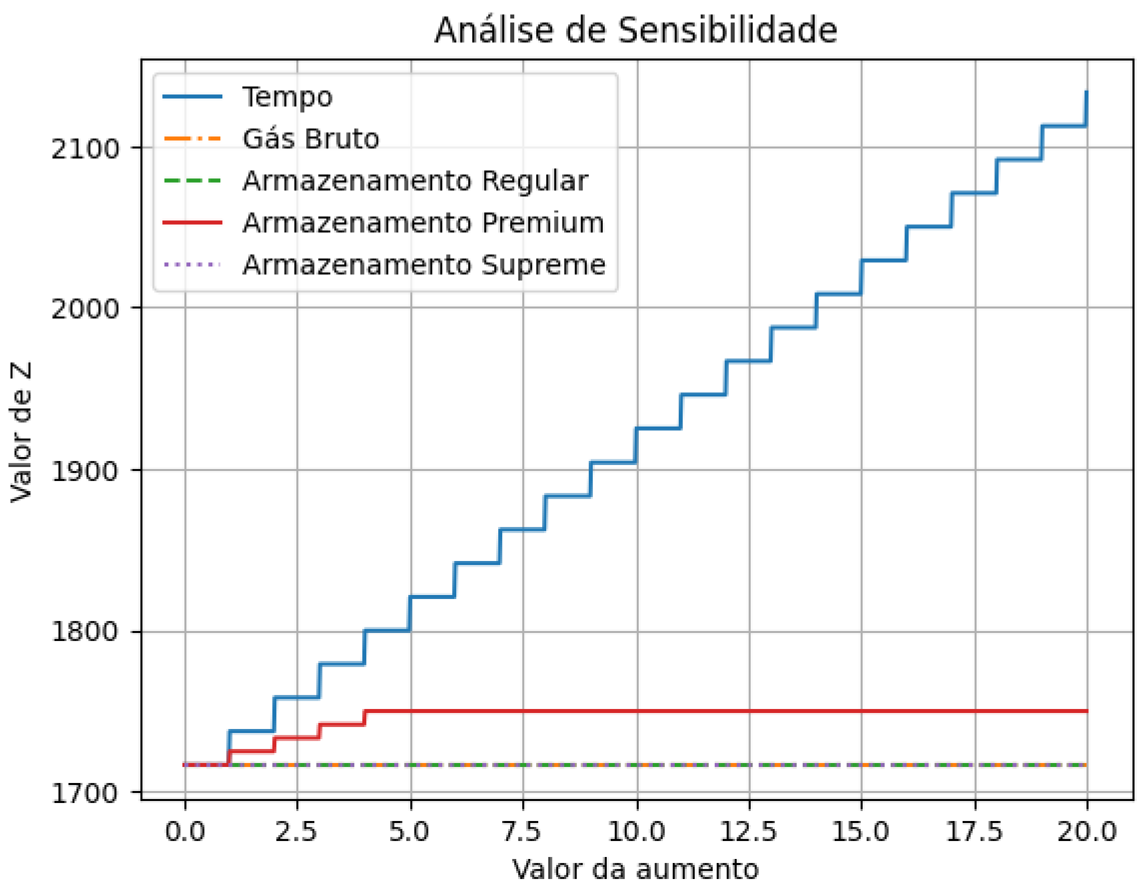
Variáveis	Z	X1	X2	X3	X4	X5	X6	X7	X8	Solução
X2	0	0	1	0	0	0	0	1	0	6
X6	0	-5.5	0	0	-1.25	1	0	-1	0	48
X7	0	1	0	0	0	0	1	0	0	9
X8	0	-0.83	0	0	-0.08	0	0	0.67	1	2.33
X3	0	0.83	0	1	0.08	0	0	-0.67	0	2.67

Item C)

In [16]:

```
valores_variados = np.linspace(0,20,1000)

resultados_1 = []
resultados_2 = []
resultados_3 = []
resultados_4 = []
resultados_5 = []

for valor in valores_variados:
    aux = np.copy(coef1)
    aux[1][-1] += valor #Tempo
    [tabela, var_basicas] = simplex(aux)
    resultados_1.append(tabela[0][-1])

    aux = np.copy(coef1)
    aux[2][-1] += valor #Gás bruto
    [tabela, var_basicas] = simplex(aux)
    resultados_2.append(tabela[0][-1])

    aux = np.copy(coef1)
    aux[3][-1] += valor #Regular
    [tabela, var_basicas] = simplex(aux)
    resultados_3.append(tabela[0][-1])

    aux = np.copy(coef1)
    aux[4][-1] += valor #Premium
    [tabela, var_basicas] = simplex(aux)
    resultados_4.append(tabela[0][-1])

    aux = np.copy(coef1)
    aux[5][-1] += valor #Supreme
    [tabela, var_basicas] = simplex(aux)
    resultados_5.append(tabela[0][-1])

plt.plot(valores_variados, resultados_1,label='Tempo')
plt.plot(valores_variados, resultados_2,linestyle='dashdot',label='Gás Bruto
plt.plot(valores_variados, resultados_3,linestyle='--',label='Armazenamento
plt.plot(valores_variados, resultados_4,label='Armazenamento Premium')
plt.plot(valores_variados, resultados_5,linestyle='dotted',label='Armazename
plt.xlabel('Valor da aumento')
plt.ylabel('Valor de Z')
```

```
plt.title('Análise de Sensibilidade')
plt.legend()
plt.grid(True)
```



Pelo gráfico podemos ver que quando aumentamos o tempo de produção do dos gáses obtemos um lucro maior.

Também podemos ver que aumentar as restrições não gera nem aumento significativo de lucro.

Questão 3

In [17]:
```
def objetivo(x1,z):
  x2 = (z-(1.75*x1))/1.25
  return (x1,x2)
```

In [18]:
```
def objetivo_2(x2,z):
  x1 = (z-(1.25*x2))/1.75
  return (x1,x2)
```

In [19]:
```
def func_1(x):
    return (14 - 1.2*x)/2.25
```

In [20]:
```python
def func_2(x):
    return (8-x)/1.1
```

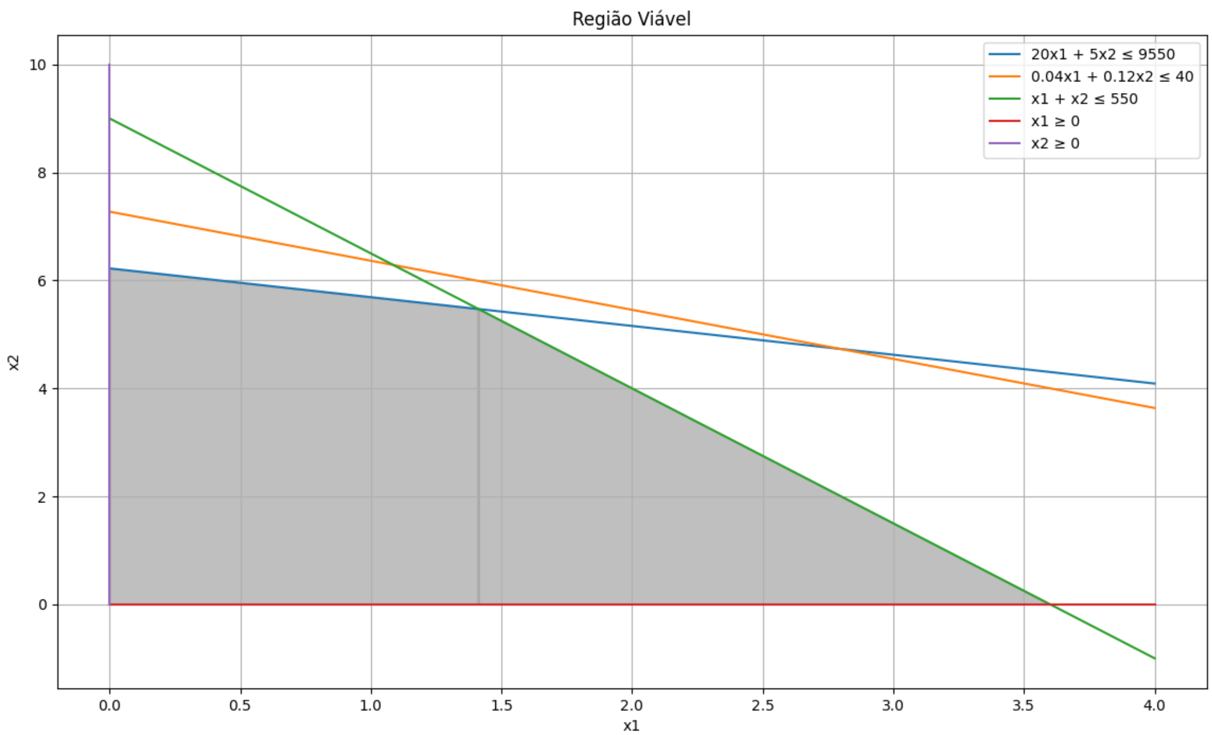
In [21]:
```python
def func_3(x):
    return (9 - 2.5*x)
```

In [22]:
```python
def func_4(x):
    return 0*x
```

Item A)

In [23]:
```python
x1 = np.linspace(0,4,1000)
x2 = np.linspace(0,10,1000)
y1 = func_1(x1)
y2 = func_2(x1)
y3 = func_3(x1)
y4 = func_4(x1)
y5 = func_4(x1)

plt.figure(figsize=(14, 8))  # Define o tamanho do gráfico
plt.grid()
plt.plot(x1, y1, label='20x1 + 5x2 ≤ 9550')
plt.plot(x1, y2, label='0.04x1 + 0.12x2 ≤ 40')
plt.plot(x1, y3, label='x1 + x2 ≤ 550')
plt.plot(x1,y4, label='x1 ≥ 0')
plt.plot(y5,x2, label='x2 ≥ 0')

plt.fill_between(x1, 0, y1, where=(y1 <= y3) & (y1 >= 0) & (x1 >= 0) , color
plt.fill_between(x1, 0, y3, where=(y3 >= 0) & (x1 >= 0) & (y3 <= y1) & (y3 <

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Região Viável')

plt.legend()
```

Out[23]: `<matplotlib.legend.Legend at 0x7f2f46707c90>`



Os pontos de insterção da região de convergência

Podemos perceber que para resolver as seguintes equações:

$$\frac{14-1.2x_1}{2.25} = 9 - 2.5x_1$$

$$9 - 2.5x_1 = 0$$

$$\frac{14-2.25x_2}{1.2} = 0$$

Resolvendo essas equações a gente obtem o seguinte os seguinte:

1. $x_1 \approx 1.4125$
2. $x_1 = 3.6$
3. $x_2 \approx 6.222$

Aplicando os valres da Equação 1 em 2.5x_1+x_2=9 para encontrar o valor de x_2:

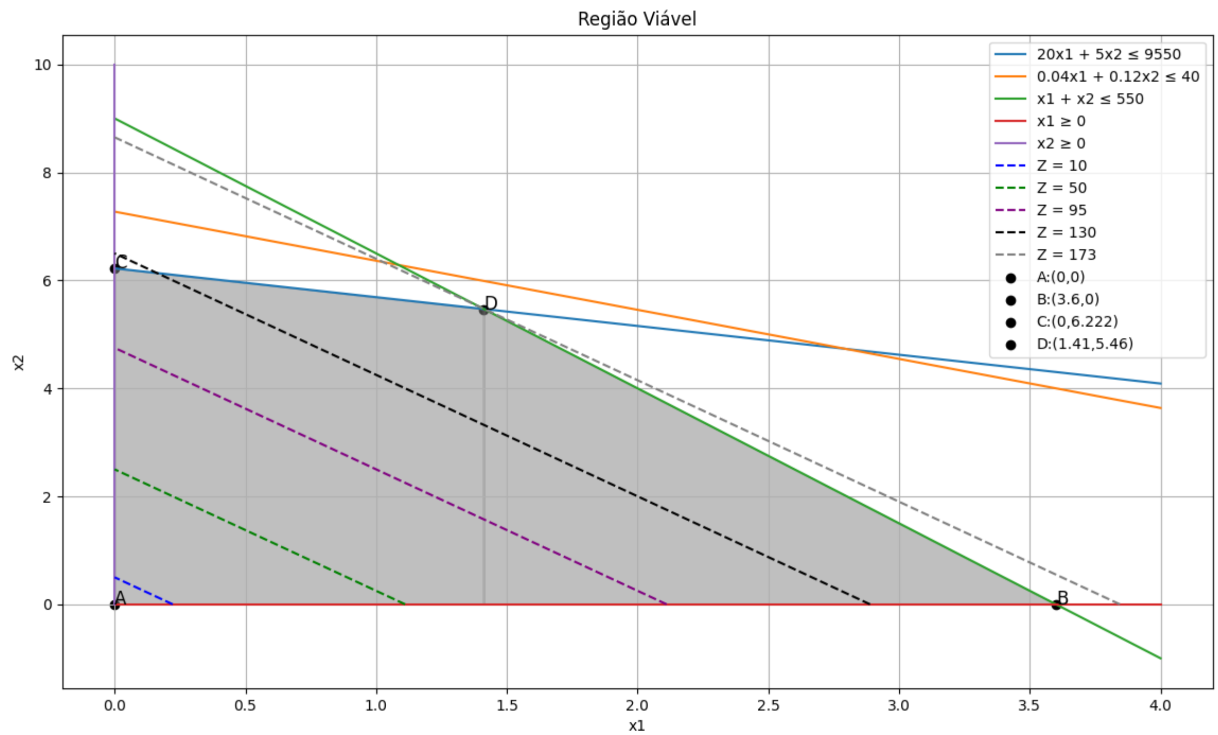
$$x_2 = 9 - 3.53125 \, x_2 \approx 5.468$$

Colocando os pares ordenados no gráfico para encontrar a solução ótima.

In [24]:
```
x1 = np.linspace(0,4,1000)
x2 = np.linspace(0,10,1000)
y1 = func_1(x1)
y2 = func_2(x1)
y3 = func_3(x1)
y4 = func_4(x1)
```

```python
y5 = func_4(x1)

#Definindo as linhas da função objetivo
z1 = funcao_obj(0,10) #z =1000
r1 = funcao_obj_2(0,10)
z2 = funcao_obj(0,50)#z =4000
r2 = funcao_obj_2(0,50)
z3 = funcao_obj(0,95)#z =8000
r3 = funcao_obj_2(0,95)
z4 = funcao_obj(0,130)#z =16000
r4 = funcao_obj_2(0,130)
z5 = funcao_obj(0,173)#z =22300
r5 = funcao_obj_2(0,173)

plt.figure(figsize=(14, 8))  # Define o tamanho do gráfico
plt.grid()
plt.plot(x1, y1, label='20x1 + 5x2 ≤ 9550')
plt.plot(x1, y2, label='0.04x1 + 0.12x2 ≤ 40')
plt.plot(x1, y3, label='x1 + x2 ≤ 550')
plt.plot(x1,y4, label='x1 ≥ 0')
plt.plot(y5,x2, label='x2 ≥ 0')
plt.plot(r1, z1,color='blue',linestyle = 'dashed', label='Z = 10')
plt.plot(r2, z2,color='green',linestyle = 'dashed',label='Z = 50' )
plt.plot(r3, z3,color='purple',linestyle = 'dashed', label='Z = 95')
plt.plot(r4, z4,color='black',linestyle = 'dashed', label='Z = 130')
plt.plot(r5, z5,color='gray',linestyle = 'dashed', label='Z = 173')

plt.scatter(0,0, color='black', label='A:(0,0)')
plt.scatter(3.6,0, color='black', label='B:(3.6,0)')
plt.scatter(0, 6.222, color='black', label='C:(0,6.222)')
plt.scatter(1.41, 5.46, color='black', label='D:(1.41,5.46)')

plt.text(0, 0, 'A', fontsize=12, color='black')
plt.text(3.6, 0, 'B', fontsize=12, color='black')
plt.text(0, 6.222, 'C', fontsize=12, color='black')
plt.text(1.41, 5.46, 'D', fontsize=12, color='black')

plt.fill_between(x1, 0, y1, where=(y1 <= y3) & (y1 >= 0) & (x1 >= 0) , color
plt.fill_between(x1, 0, y3, where=(y3 >= 0) & (x1 >= 0) & (y3 <= y1) & (y3 <

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Região Viável')

plt.legend()
```

Out[24]: <matplotlib.legend.Legend at 0x7f2f46874a50>



Traçando as funções objetivo podemos ver que elas vão se distanciando da origem

Na Situação limite, o ponto D é intercpetado pela reta da função objetivo. Então podemos presumir que as os valores de x1 e x2 no ponto D são os valores para se obter o valor ótimo da função objetivo

Item B)

Para aplicar o método simplex nesse problema precisamos primeiro colocar variavéis de folga nas inequações. Essa variavéis servem para

$Z - 1.75x_1 - 1.25x_2 = 0$

$1.2x_1 + 2.25x_2 + x_3 = 14$

$x_1 + 1.1x_2 + x_4 = 8$

$2.5x_1 + x_2 + x_5 = 9$

$x_1, x_2 \geq 0$

Variáveis	Z	X1	X2	X3	X4	X5	Solução
Z	1	-1.75	-1.25	0	0	0	0
X3	0	1.2	2.25	1	0	0	14
X4	0	1	1.1	0	1	0	8
X5	0	2.5	1	0	0	1	9

In [25]:
```
coef3 = [[1,-1.75,-1.25,0,0,0,0],
           [0,1.2,2.25,1,0,0,14],
           [0,1,1.1,0,1,0,8],
           [0,2.5,1,0,0,1,9]] #Coeficientes das variaveis

[tabela,var_basicas] = simplex(coef3)
np.set_printoptions(precision=2, suppress=True)
print(tabela)
```

```
[[ 1.    0.    0.    0.31  0.    0.55  9.31]
 [ 0.    0.    1.    0.56  0.   -0.27  5.47]
 [ 0.    0.    0.   -0.4   1.   -0.21  0.57]
 [ 0.    1.    0.   -0.23  0.    0.51  1.41]]
```

In [26]:
```
nLinhas, nColunas = tabela.shape
for i in range(nLinhas):
  print(var_basicas[i],"=",tabela[i][nColunas-1])
```

```
Z = 9.307909604519773
X2 = 5.468926553672316
X5 = 0.571751412429379
X1 = 1.4124293785310735
```

Variáveis	Z	X1	X2	X3	X4	X5	Solução
Z	1	0	0	0.31	0	0.55	9.31
X2	0	0	1	0.56	0	-0.27	5.47
X5	0	0	0	-0.4	1	-0.21	0.57
X1	0	1	0	-0.23	0	0.51	1.41