In [148…

```python
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
import sympy as sym
from sympy import init_printing

np.seterr(all='ignore')
```

Out[148…

```
{'divide': 'ignore', 'over': 'ignore', 'under': 'ignore', 'invalid': 'ignore'}
```

Introduction

We would like to approach the problem of pricing some asset through its present value by discounting its cashflows, an industry standard technique.

We would like to do this for some asset A, with periodic cash flows, say each year, which are growing at a rate G = (1+g) where 0 < g < 1 and discount it at some rate R = (1+r), 0 < r < 1.

Some Math

So, why do we discount? We discount a periodic payment by the rate of that period; following a very key aspect of finance: time value of money.

Cash can always be invested into something to earn a return, whether it is equities, fixed income, or a fund, meaning it is always better to have 1 dollar now, rather than 1 dollar in one years time, so obviously $1 in one years time, is worth less now, which can be found by discounting it at an appropriate rate.

$$p_0 = C_0 + C_1/R + \dots + C_T/R^T$$

And that actually our cash flow is growing, such that:

$$C_t = G^t C_0$$

So, we can compute a formula for present value.

$$p_0 = C_0 + GC_0/R + \dots + G^T C_0/R^T$$

$$p_0 = C_0(1 + GR^{-1} + \dots + G^T R^{-T})$$

Since finite geometric series can be represented by:

$$1 + a + a^2 + \dots + a^T = \frac{1 - a^{T+1}}{1 - a}$$

$$p_0 = C_0(1 + (GR^{-1}) + (GR^{-1})^2 + \dots + (GR^{-1})^T) = \frac{C_0(1 - G^{T+1}R^{-(T+1)})}{1 - GR^{-1}}$$

Now defining f1(x) as:

$$f_1(r) = (1 + r)^{-(T+1)}$$

We may now take the nth order taylor polynomial of f1(r) around r = 0, since r is probably very close to 0.

$$f_1(0) + f_1^{'}(0)(r) + \frac{f_1^{''}(0)}{2}(r^2) + \dots + \frac{f_1^{(n)}(0)}{n!}(r^n)$$

for some (k+1)th term, we have this very small term; assuming r < 1;

$$\frac{1}{k!}(r^k)$$

Which makes any term past the second extremely small; so can be disregarded. Thus, we have the following approximation for f1(r):

$$f_1(r) \approx f_1(0) + f_1^{'}(0)(r) = 1 - r(T + 1)$$

We do a similar process for

$$f_2(g) = (1 + g)^{(T+1)}$$

$$f_2(g) \approx 1 + g(T + 1)$$

So, we have the following approximation for the present value:

$$p_0 \approx C_0(T + 1)(1 + \frac{rg}{r - g})$$

since r < 1, g < 1, rg could be small enough to simplify it even more, with a rougher approximation:

$$p_0 \approx C_0(T + 1)$$

Coding and Testing our Approximations

Now we have 3 different ways to calculate the present value; lets see how well they are. Here, c defines the first cashflow payment, C_0.

In [149…

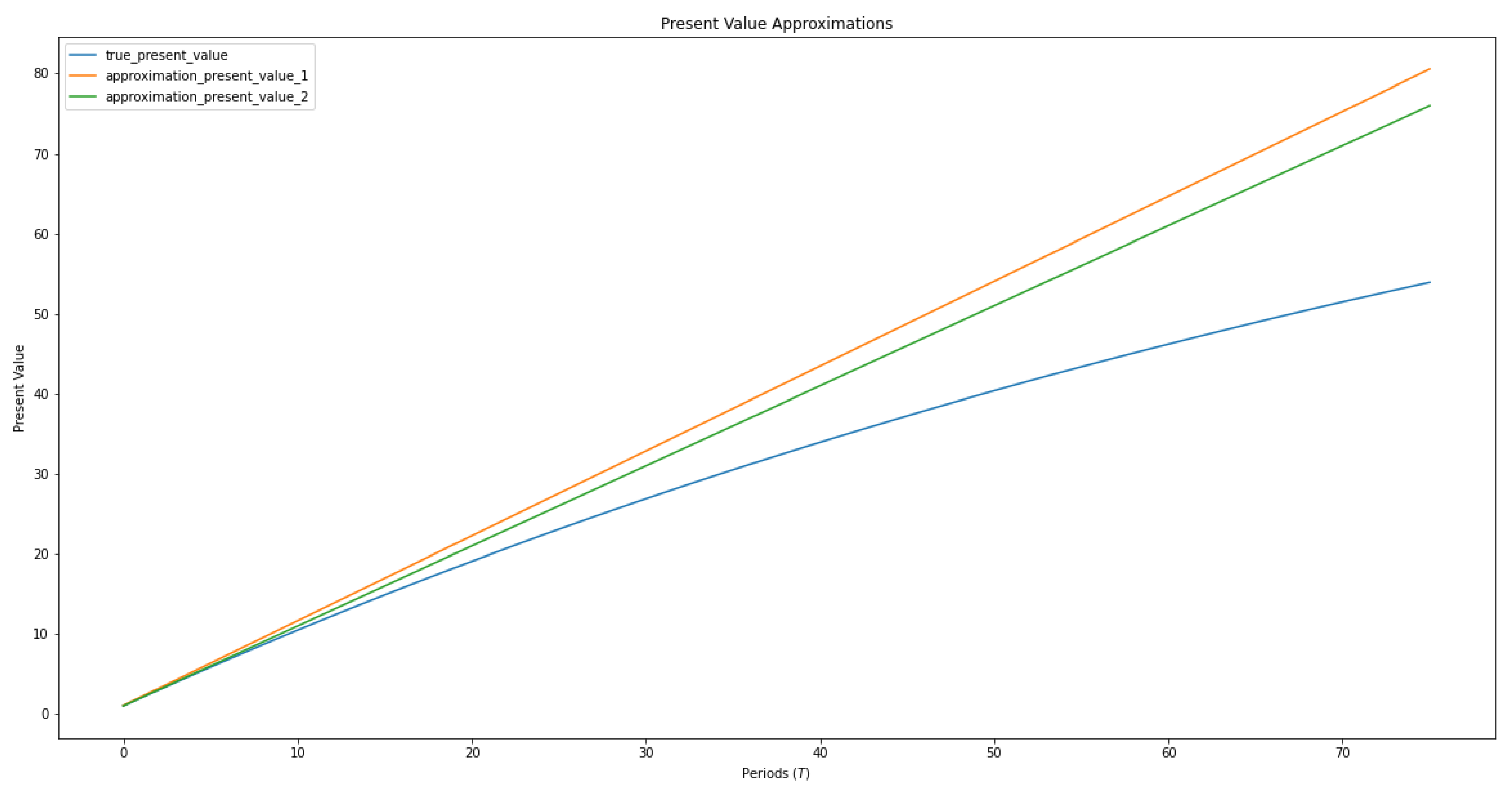
```python
def true_present_value(r, g, c, T):
    R = 1 + r
    G = 1 + g
    return (c*(1 - G**(T + 1) * R**((-1) * T - 1)))/(1 - G * R**(-1))

def approximation_present_value_1(r, g, c, T):
    return c * (T + 1) * (1 + (r*g)/(r - g))

def approximation_present_value_2(r, g, c, T):
    return c * (T + 1)

functions = [true_present_value, approximation_present_value_1, approximation_present_value_2]
```

In [150…

```python
T = [i for i in range(0, 76)]
g = 0.02
r = 0.03
c = 1

fig, ax = plt.subplots()
fig.set_size_inches(20, 10)
ax.set_title("Present Value Approximations")
ax.set_xlabel("Periods ($T$)")
ax.set_ylabel("Present Value")

for f in functions:
    plt.plot(T, [f(*(r, g, c, i)) for i in T], label=f.__name__)

ax.legend()
```

Out[150…

```
<matplotlib.legend.Legend at 0x130220b2848>
```



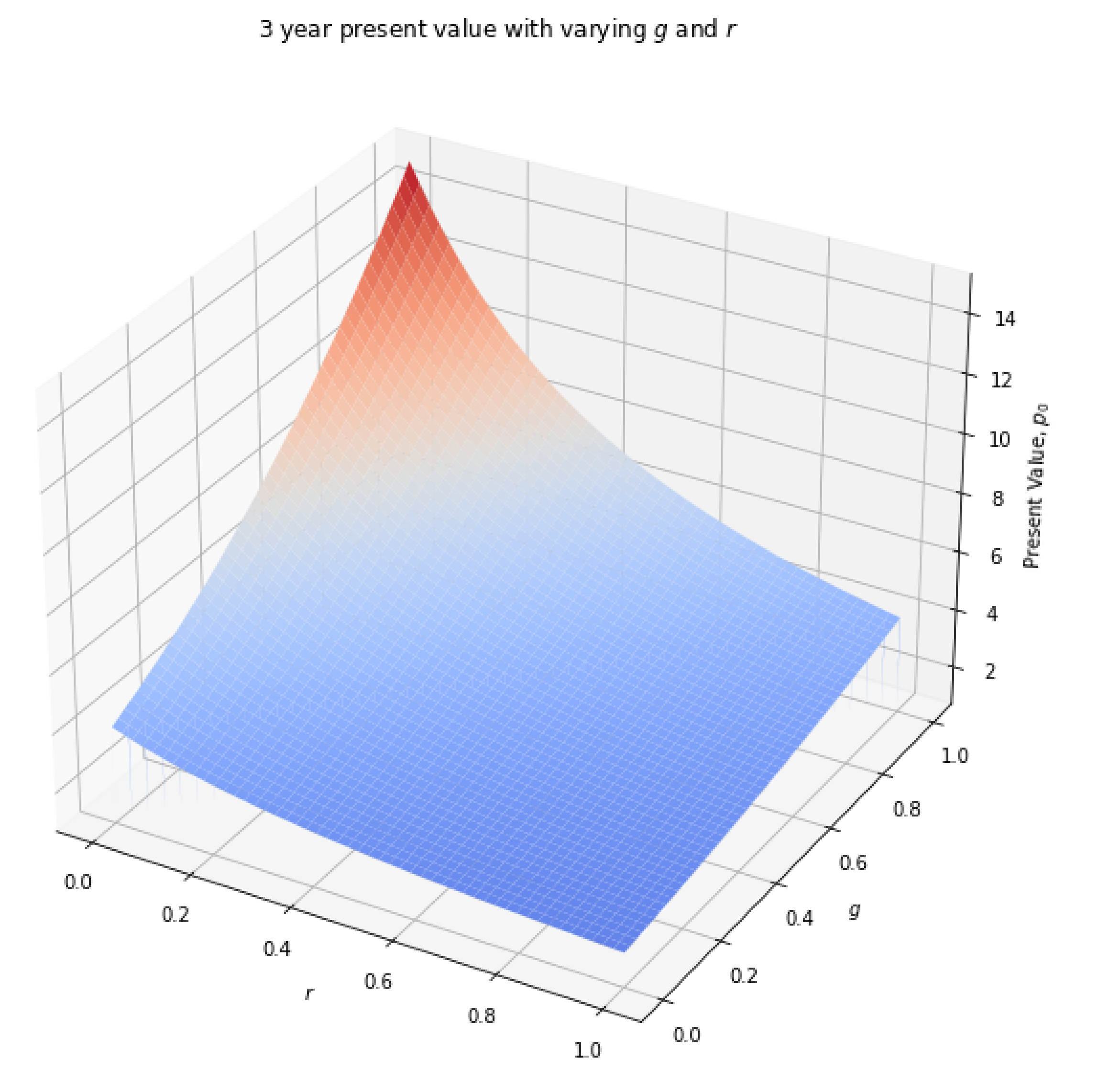
So, actually out approximations are pretty good up to around 20 time periods. Since these time periods are usually in years, 20 sounds about right. We can also check how our approximations vary with a changing r and g, with some 3d plots.

In [151…

```python
fig = plt.figure()
fig.set_size_inches(20, 10)
ax = plt.axes(projection='3d')

r = np.linspace(0, 1, 2501)
g = np.linspace(0, 1, 2501)

X, Y = np.meshgrid(r, g)
Z = true_present_value(X, Y, 1, 3)

Z[(X == Y)] = 1

surface = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
    antialiased=True, clim=(0, 15))

ax.set_xlabel('$r$')
ax.set_ylabel('$g$')
ax.set_zlabel('Present Value, $p_0$')
ax.set_title('3 year present value with varying $g$ and $r$')
```

Out[151…

```
Text(0.5, 0.92, '3 year present value with varying $g$ and $r$')
```



In [152…

```python
g, r, c = sym.symbols('g, r, c0')
G = (1 + g)
R = (1 + r)
p0 =  c / (1 - G * R**(-1))
init_printing(use_latex='mathjax')
print('Present Value')
p0
```

```
Present Value
```

Out[152…

$$\frac{c_0}{-\frac{g+1}{r+1} + 1}$$

In [153…

```python
print('Partial Derivative of p0 with Respect to g')
dp_dg = sym.diff(p0, g)
dp_dg
```

```
Partial Derivative of p0 with Respect to g
```

Out[153…

$$\frac{c_0}{(r + 1)\left(-\frac{g+1}{r+1} + 1\right)^2}$$

So, our partial derivative of p0 with respect to g, is always positive, meaning, assuming r constant, increasing our g value will increase our present value; good to see the maths is consistent with the intuition.

In [154…

```
print('Partial Derivative of p0 with Respect to r')
dp_dr = sym.diff(p0, r)
dp_dr
```

```
Partial Derivative of p0 with Respect to r
```

Out[154…

$$- \frac{c_0(g + 1)}{(r + 1)^2 \left(- \frac{g+1}{r+1} + 1 \right)^2}$$

So, our partial derivative of p0 with respect to r, is always negative, meaning, assuming g constant, increasing our r value will decrease our present value; good to see the maths is consistent with the intuition.