Assignment_12Dec

December 5, 2024

1 Mathematical Models - Assignment 1

[1]:
```
%display latex
```

[2]:
```
from sympy import Function, rsolve, sympify, Symbol, latex
from sympy.abc import n
from IPython.display import Math
import matplotlib.pyplot as plt
```

1. Find the solution for the following difference equations:

   (a) $x_{n+1} = \left(\frac{n+1}{n+2}\right)^2 \cdot x_n + \frac{1}{n+2}, \quad x_0 = 1;$

   (b) $x_{n+3} - 4 \cdot x_{n+2} + x_{n+1} + 6 \cdot x_n = 60 \cdot 4^n, \quad x_0 = 2, \; x_1 = 12, \; x_2 = 12;$

   (c) $x_{n+1} = \frac{2 \cdot x_n}{1 + 4 \cdot x_n}, \quad x_0 = 1$ (Hint: use substitution $x_n = \frac{1}{y_n}$)

[3]:
```
x = Function('x')
```

1.1 A

[4]:
```
f = x(n+1) - ((n+1)/(n+2))**2*x(n) - 1/(n+2);
sol = rsolve(f, x(n), {x(0):1})
print(f"symbollic solution:")
display(Math(latex(sol)))
```

```
symbollic solution:
```

$$\frac{n\,(n+3)}{2\,(n+1)^2}$$

1.2 B

[5]:
```
f = x(n+3) - 4*x(n+2) + x(n+1) + 6*x(n) - 60*4**n;
sol = rsolve(f, x(n), {x(0):2, x(1):12, x(2):12})
print(f"symbollic solution:")
display(Math(latex(sol)))
```

```
symbollic solution:
```

$-4\left(-1\right)^{n} + 6 \cdot 2^{2n} + 16 \cdot 2^{n} - 16 \cdot 3^{n}$

1.3 C

$$x_n = \frac{1}{y_n}$$

$$x_{n+1} = \frac{2 \cdot x_n}{1 + 4 \cdot x_n} \rightarrow \frac{1}{y_{n+1}} = \frac{2 \cdot \frac{1}{y_n}}{1 + 4 \cdot \frac{1}{y_n}} = \frac{2}{y_n + 4} \rightarrow y_{n+1} = \frac{y_n + 4}{2}$$

[6]:
```
f = x(n+1) - (x(n) + 4) / 2;
sol = rsolve(f, x(n), {x(0):1})
print(f"symbollic solution:")
display(Math(latex(sol)))
```

```
symbollic solution:
```

$4 - 3 \cdot 2^{-n}$

2. Let us consider the difference equation:

$$x_{n+1} = \frac{x_n^2 + 7}{2x_n}.$$

(a) Find the equilibrium points and study their stability.

(b) Make some numerical simulations.

$x^* = f(x^*, x^*, \ldots, x^*)$ x^* - equilibrium point $|f'(x^*)| < 1 \rightarrow x^*$ *asymptotically stable* $|f'(x^*)| > 1 \rightarrow x^*$ *unstable*

1.3.1 a)

[102]:
```
def eq_points_stability(f):
    eq = x == f(x)
    eqp = solve(eq, x)
    show(eqp)

    df = diff(f(x), x)
```

```
    for eq_point in eqp:
        stable = abs(df.subs(x=eq_point.rhs())) < 1
        if stable:
            show(eq_point.rhs())
            print(f"- eq_point is asymptotically stable")
        else:
            show(eq_point.rhs())
            print(f"- eq_point is unstable")
```

[103]:
```
x = var('x')
f = lambda x: (x^2 + 7) / (2*x)
eq_points_stability(f)
```

$$\left[x = -\sqrt{7}, x = \sqrt{7}\right]$$

$$-\sqrt{7}$$

```
- eq_point is asymptotically stable
```

$$\sqrt{7}$$

```
- eq_point is asymptotically stable
```

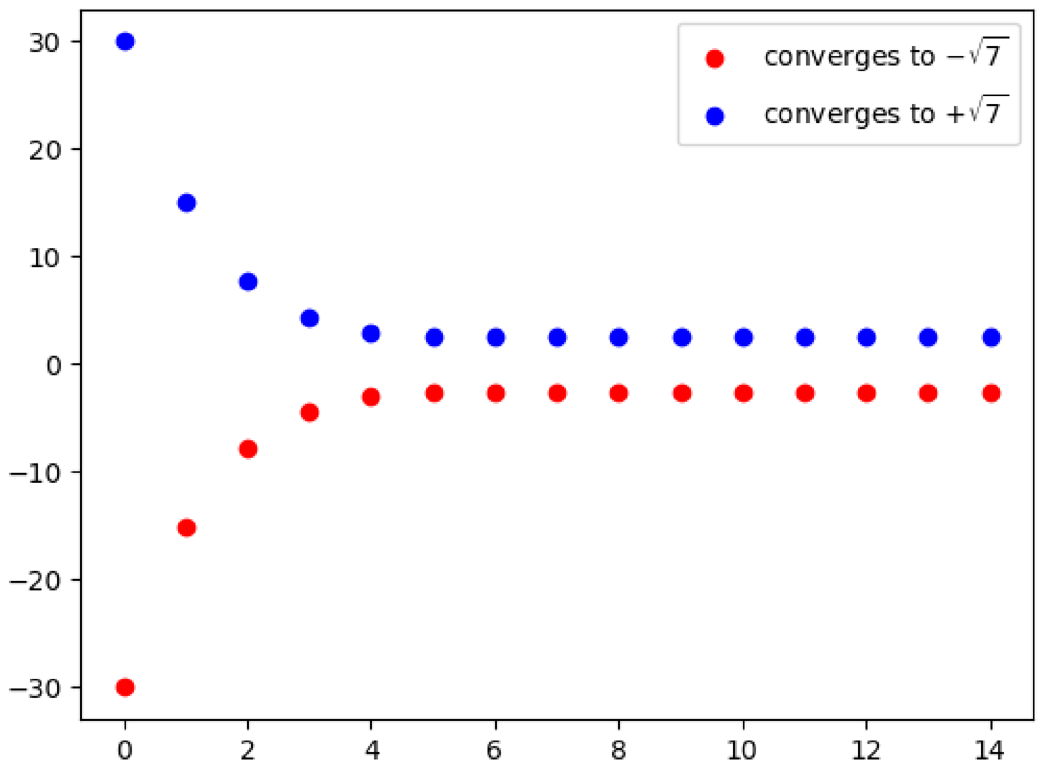
1.3.2 b)

[9]:
```
x = var('x')
show(f(x))
N = 30
x = [0] * N
x[0] = -30
for i in range(1, N / 2):
    x[i] = f(x[i-1])

x[N / 2] = 30
for i in range(N / 2 + 1, N):
    x[i] = f(x[i-1])

plt.plot(range(N/2), x[:N/2], 'ro', label=r"converges to $-\sqrt{7}$")
plt.plot(range(N/2), x[N/2:N], 'bo', label=r"converges to $+\sqrt{7}$")
plt.legend()
plt.show()
```

$$\frac{x^2 + 7}{2\,x}$$



3. Let us consider the difference equation:

$$x_{n+1} = x_n^2 - 3.$$

   (a) Find the 2-periodic cycle and study its stability.
   (b) Make numerical simulation.

1.3.3 a) 2-Periodic Cycle and Stability Study

[114]:
```
def stability(df, check_points):
    for point in check_points:
        stable = abs(df.subs(x=point)) < 1
        if stable:
            show(f"point: {point} is asymptotically stable")
        else:
            show(f"point: {point} is unstable")
```

2-periodic cycle: $[1, -2]$

$$f(1) = -2, \; f(-2) = 1$$

[111]:
```
x = var('x')
f = lambda x: x**2 - 3
f(x)
```

[111]: $x^2 - 3$

[112]:
```
# check cycle:
val = -2
print(f"start value: {val}")
for step in range(15):
    val = f(val)
    print(f"step {step + 1}, val = {val}")
```

```
start value: -2
step 1, val = 1
step 2, val = -2
step 3, val = 1
step 4, val = -2
step 5, val = 1
step 6, val = -2
step 7, val = 1
step 8, val = -2
step 9, val = 1
step 10, val = -2
step 11, val = 1
step 12, val = -2
step 13, val = 1
step 14, val = -2
step 15, val = 1
```

stability check

[115]:
```
check_points = [1, -2]
df = diff(f(x), x)
stability(df, check_points)
```

```
point: 1 is unstable
```

```
point: -2 is unstable
```

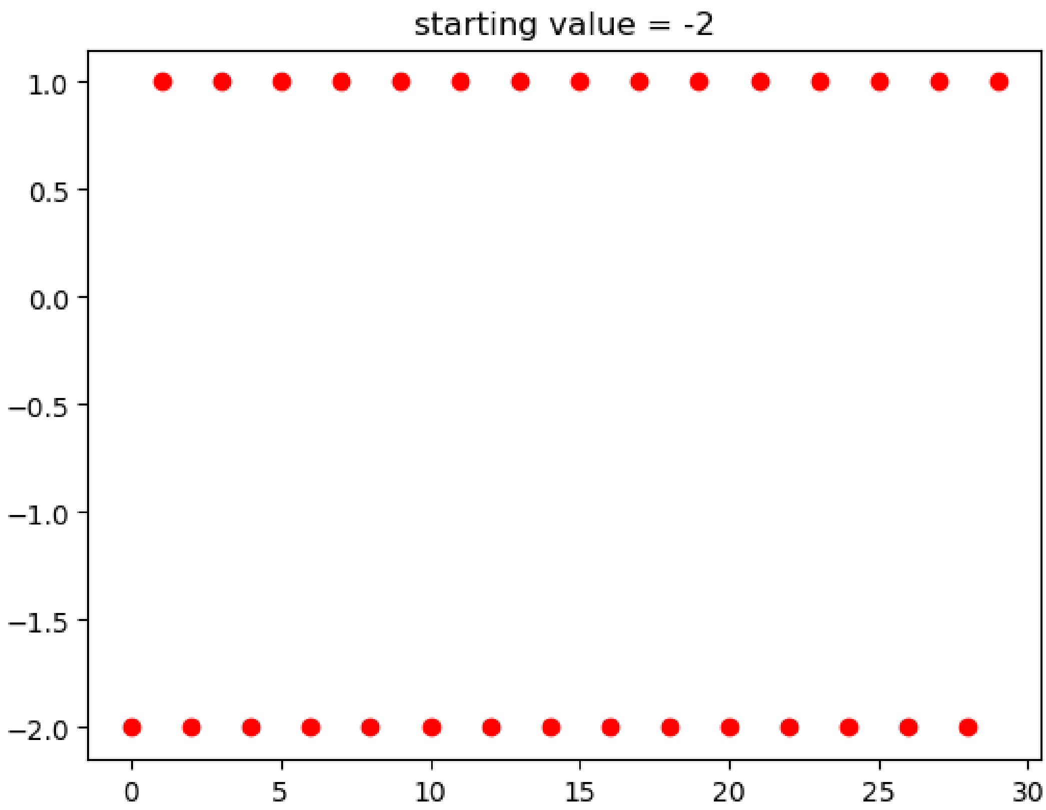
1.3.4 b) Numerical Simulations

[77]:
```
x = var('x')
show(f(x))
N = 30
x = [0] * N
x[0] = -2

for i in range(1, N):
    x[i] = f(x[i-1])
```

```
plt.plot(range(N), x, 'ro')
plt.title(f"starting value = {x[0]}")
plt.show()
```

$x^2 - 3$

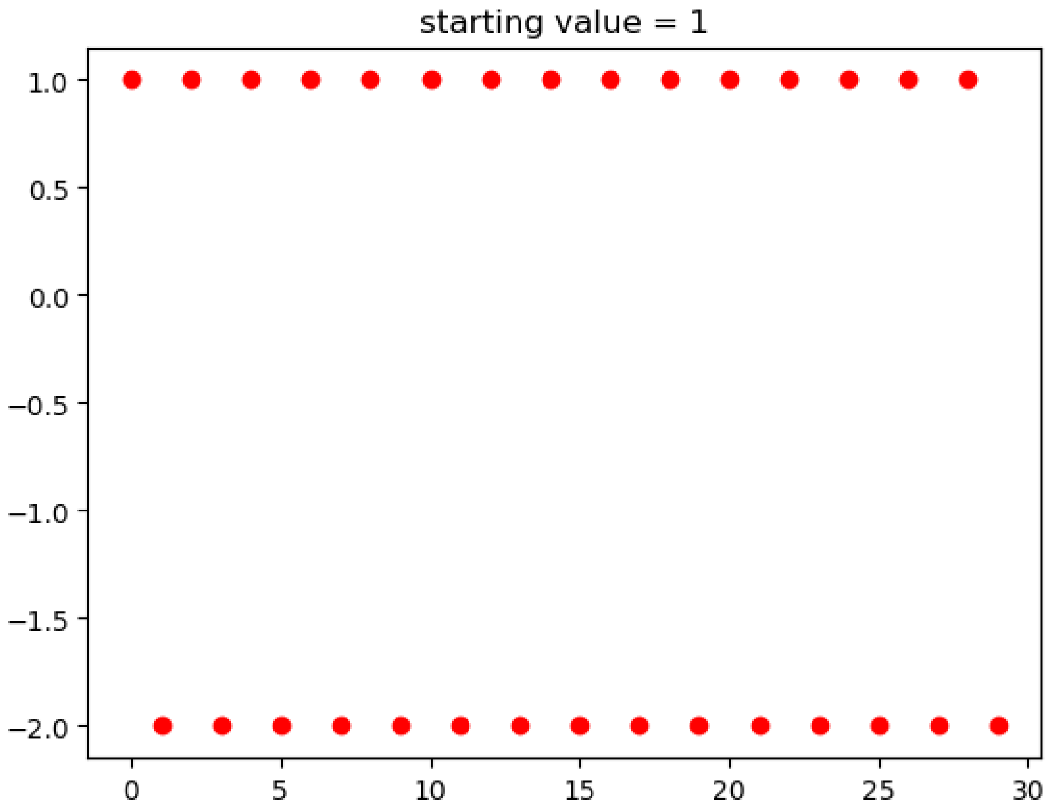


[95]:
```
x = var('x')
show(f(x))
N = 30
x = [0] * N
x[0] = 1

for i in range(1, N):
    x[i] = f(x[i-1])

plt.plot(range(N), x, 'ro')
plt.title(f"starting value = {x[0]}")
plt.show()
```

$x^2 - 3$

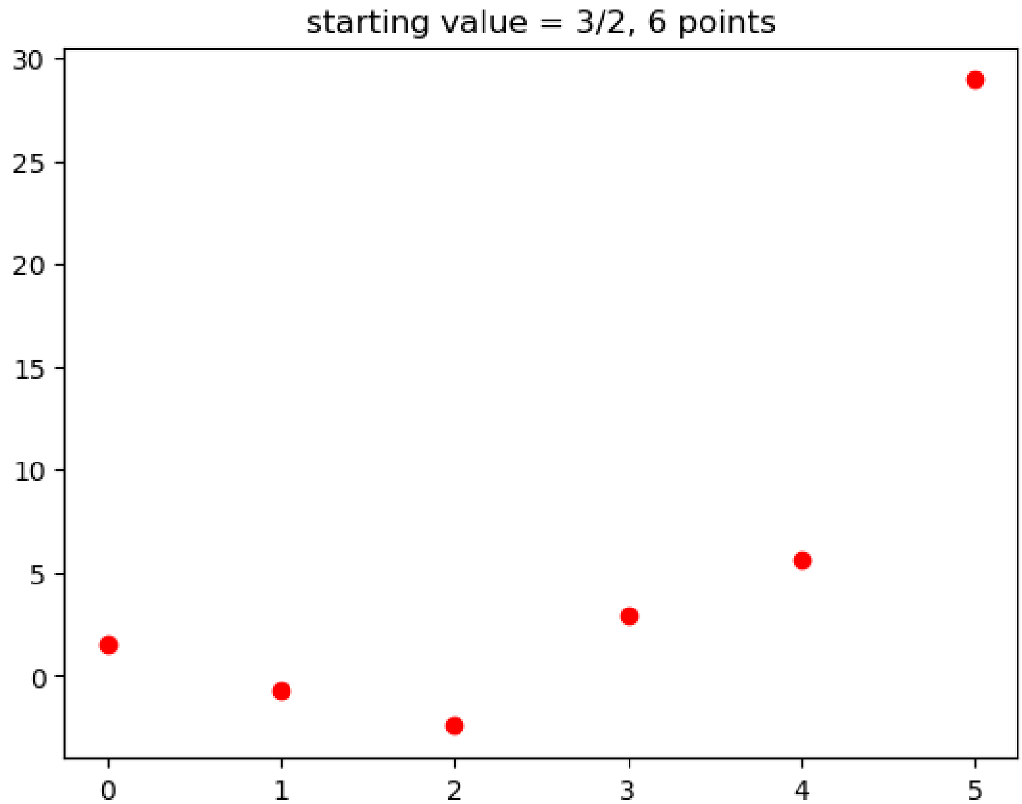


[83]:
```
x = var('x')
show(f(x))
N = 6
x = [0] * N
x[0] = 3/2

for i in range(1, N):
    x[i] = f(x[i-1])

plt.plot(range(N), x, 'ro')
plt.title(f"starting value = {x[0]}, {N} points")
plt.show()
```

$x^2 - 3$

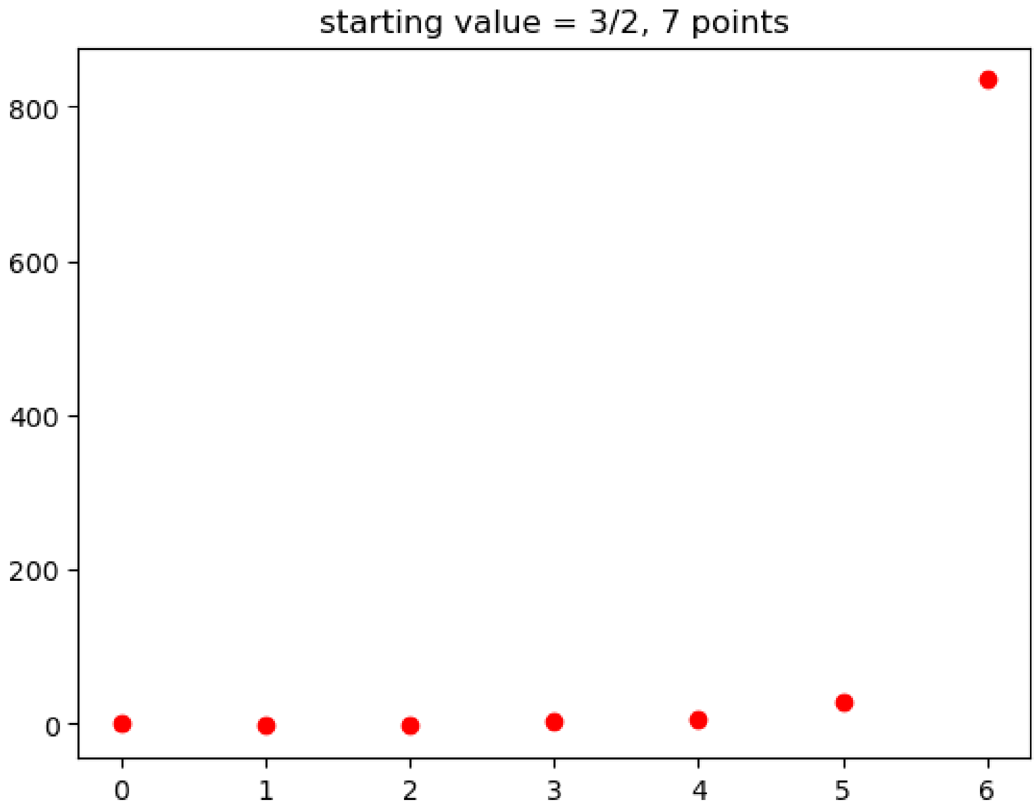


[86]:
```
x = var('x')
show(f(x))
N = 7
x = [0] * N
x[0] = 3/2

for i in range(1, N):
    x[i] = f(x[i-1])

plt.plot(range(N), x, 'ro')
plt.title(f"starting value = {x[0]}, {N} points")
plt.show()
```

$x^2 - 3$

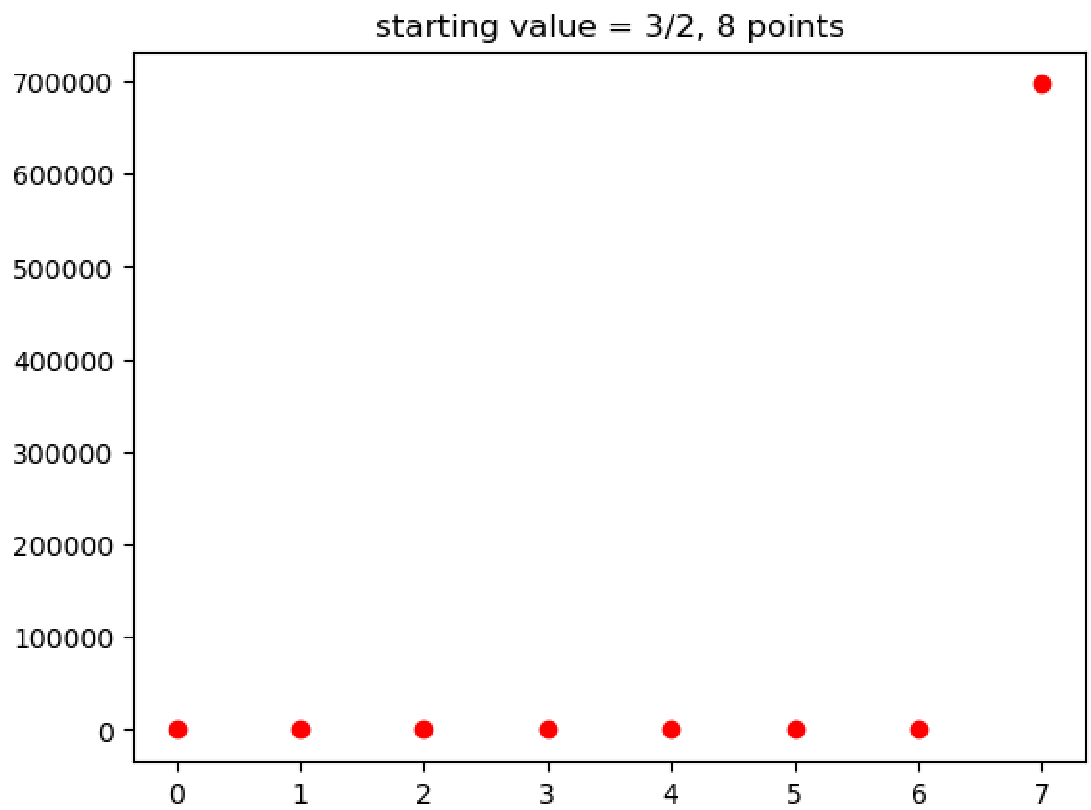


[96]:
```
x = var('x')
show(f(x))
N = 8
x = [0] * N
x[0] = 3/2

for i in range(1, N):
    x[i] = f(x[i-1])

plt.plot(range(N), x, 'ro')
plt.title(f"starting value = {x[0]}, {N} points")
plt.show()
```

$x^2 - 3$



4. Consider the simple interest formula $S_n = (1 + np)S_0$ and the compound interest formula $S_n = (1 + p/r)^n S_0$. There are two options to earn interest. Company A offers simple interest at a rate of 4%. Company B offers compound interest at a 3% rate with a conversion period of one month.

   (a) Calculate for the both cases the amount on deposit after 5, 10, 15, and 20 years for principal $S_0 = 1000$.

   (b) Which interest offer maximizes the amount on deposit after 5, 10, 15 and 20 years?

1.3.5 a)

[116]:
```python
def simple_interest(S0, n):
    return S0 * (1 + 0.04 * n)

# r = 12, monthly
def compound_monthly(S0, n):
    return S0 * (1 + 0.03/12)**(12*n)

S0 = 1000
```

```python
years = [5, 10, 15, 20]

print("Comparison of interest rates for S0 = $1000:")
print("\nYear | Company A (Simple) | Company B (Monthly)")
print("-" * 45)

results_A = []
results_B = []

for n in years:
    A = simple_interest(S0, n)
    B = compound_monthly(S0, n)

    results_A.append(A)
    results_B.append(B)

    print(f"{n:2d} | ${A:14.2f} | ${B:14.2f}")
```

```
Comparison of interest rates for S0 = $1000:

Year | Company A (Simple) | Company B (Monthly)
---------------------------------------------
 5 | $       1200.00 | $       1161.62
10 | $       1400.00 | $       1349.35
15 | $       1600.00 | $       1567.43
20 | $       1800.00 | $       1820.75
```

1.3.6 b)

[99]:
```python
print("\nBest option for each period:")
for i, n in enumerate(years):
    best = max(results_A[i], results_B[i])
    if best == results_A[i]:
        company = "Company A (Simple)"
    elif best == results_B[i]:
        company = "Company B (Monthly)"
    print(f"{n} years: {company}")
```

```

Best option for each period:
5 years: Company A (Simple)
10 years: Company A (Simple)
15 years: Company A (Simple)
20 years: Company B (Monthly)
```