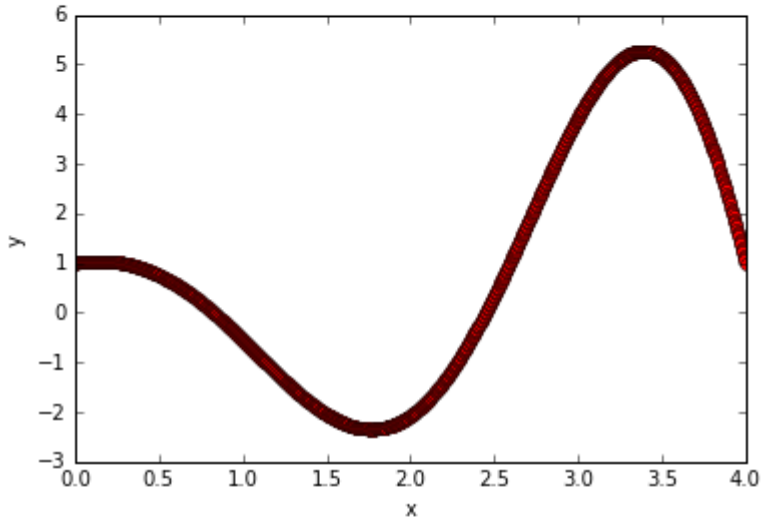
Autor: Paweł Salwa

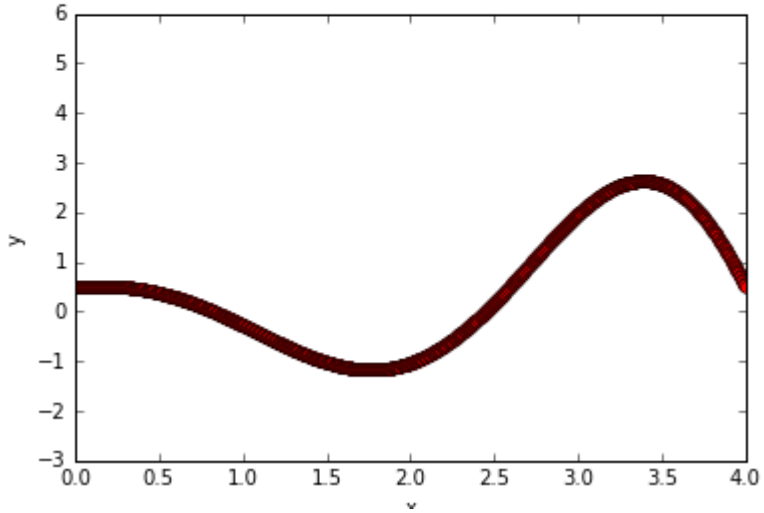
Program ten ma złożoność liniową O(n)- wykorzystałem algorytm thomasa i shermana-morrisona, więc tak jak w poprzednim zadaniu program ma kilka pętli tylko pojedynczych.

Output:

```
start
2
3
4
5
6
7
po algorytmie thomasa:
```




```
po algorytmie shermana morrisona:
```



Kod w pythonie:

```
#=============================================================
def shermanMorrison(Z, Q ,UV):
	global N

	mian = 0.
	liczn = 0.

	for i in range(0,N):
		liczn += Z[i]*UV[i]
		mian += Q[i]*UV[i]

	mian += 1

	for i in range(0,N):
		Q[i] *= (liczn/mian)

	for i in range(0,N):
		Z[i] = Z[i]-Q[i]


#=============================================================
def thomas(A,B,C,D):
	global N

	N -= 1

	C[0] /= B[0]
	D[0] /= B[0]

	for i in range(1,N):
		C[i] /= B[i] - A[i]*C[i-1]
		D[i] = (D[i] - A[i]*D[i-1]) / (B[i] - A[i]*C[i-1])

	D[N] = (D[N] - A[N]*D[N-1]) / (B[N] - A[N]*C[N-1])


	i = N - 1
	while i >= 0:
		D[i] -= C[i] * D[i+1]
		i -= 1

#=============================================================
print 'start'

N = 1001

A = []
B = []
C = []
```

```
#=====tworzenie macierzy ze wzorow====
h0 = 4.0 / (N-1)
d1 = 2. * h0
d2 = h0 ** 2.

A.append(0.)
B.append(1.)
C.append(0.)

for i in range(1,N-1):
        A.append(1./d1 + 1./d2)
        B.append(-2./d2 + 4.)
        C.append(-1./d1 + 1./d2)

A.append(0.)
B.append(1.)
C.append(0.)
print '2'

X = [] #tworze liste X z N elementami
X.append(1.)
for i in range(1,N-1):
        X.append(0)
X.append(1.)
print '3'

thomas(A,B,C,X)
print '4'
N = 1001 # bo thomas mi moze zmienic n

wynik1x = []#pierwsza maciez- potrzebne do wykresu (thomas)
wynik2x = []
for i in range(0,N):
        #wynik1 += (repr(i*h0) + ' ' + repr(X[i]) + '\n')
        wynik1x.append(i*h0)
        wynik2x.append(X[i])

print '5'
#==============inicjalizacja wektorow============================
del X
X = [] #jeszcze raz
X.append(1.)
for i in range(1,N-1):
        X.append(0)
X.append(1.)
Q = [] #tworze liste Q z N elementami
Q.append(1.)
for i in range(1,N-1):
        Q.append(0)
Q.append(1.)
V = [] #tworze liste V z N elementami
V.append(1.)
```

```
for i in range(1,N-1):
        V.append(0)
V.append(1.)
V[0]=-3 # ze wskazwki
V[1]= 4
V[2]=-1
#=======obliczanie ukladu rownan=======================================

thomas(A,B,C,X)
thomas(A,B,C,Q)
shermanMorrison(X,Q,V)
print '6'

wynik1y = []#druga maciez- potrzebne do wykresu(sherman morr)
wynik2y = []
for i in range(0,N):
        #wynik2 += (repr(i*h0) + ' ' + repr(X[i]) + '\n')
        wynik1y.append(i*h0)
        wynik2y.append(X[i])
print '7'
#===========lecimy z wykresem:===============
import matplotlib.pyplot as plt
fig = plt.figure()

plt.plot(wynik1x, wynik2x, 'ro')
plt.axis([0, 4, -3, 6])
plt.ylabel('y')
plt.xlabel('x')
print "po algorytmie thomasa:"
plt.show()
fig.savefig('zad6_plot1.pdf')

del fig

fig2 = plt.figure()

plt.plot(wynik1y, wynik2y, 'ro')
plt.axis([0, 4, -3, 6])
plt.ylabel('y')
plt.xlabel('x')
print "po algorytmie shermana morrisona:"
plt.show()
fig2.savefig('zad6_plot2.pdf')
del fig2
```