Temat:			
Rozwiązywanie UARL metodami bezpośrednimi (1)			
Wykonał:	Wydział:	Kierunek	Grupa:
Marcin Fabrykowski	FiIS	Inf. Stos.	grupa 3

1. Metoda eliminacji Gaussa-Jordana
   Metoda ta pozwala na redukcje zadanej macierzy do macierzy jednostkowej.
   Mając układ równań np:

$$\begin{aligned} 2*x_0+4*x_1+6*x_2&=46 \\ 7*x_0+1*x_1+2*x_2&=34 \\ 4*x_0+4*x_1+4*x_2&=40 \end{aligned} \tag{1}$$

   Zamieniamy miejscami wiersz 1 i 2:

$$\begin{aligned} 7*x_0+1*x_1+2*x_2&=34 \\ 2*x_0+4*x_1+6*x_2&=46 \\ 4*x_0+4*x_1+4*x_2&=40 \end{aligned} \tag{2}$$

   Dzielimy wiersz pierwszy przez $a_{11}=7$

$$\begin{aligned} 1*x_0+0.143*x_1+0.286*x_2&=4.857 \\ 2*x_0+4*x_1+6*x_2&=46 \\ 4*x_0+4*x_1+4*x_2&=40 \end{aligned} \tag{3}$$

   Następnie odejmujemy od pozostałych wierszy wiersz pierwszy pomnożony przez odpowiedni współczynnik a_{i1},
   Dla wiersza 2: $a_{21}=2$

$$\begin{aligned} 1*x_0+0.143*x_1+0.286*x_2&=4.857 \\ 0*x_0+3.714*x_1+5.429*x_2&=36.286 \\ 0*x_0+3.429x_1+2.857*x_2&=20.571 \end{aligned} \tag{4}$$

   Następnie dzielimy wiersz drugi przez współczynnik $a_{22}=3.714$

$$\begin{aligned} 1*x_0+0.143*x_1+0.286*x_2&=4.857 \\ 0*x_0+1*x_1+1.462*x_2&=9.769 \\ 0*x_0+3.429x_1+2.857*x_2&=20.571 \end{aligned} \tag{5}$$

I podobnie jak wcześniej, odejmujemy go od pozostałych wierszy, mnożąc przez a_{i2}

$$\begin{aligned}1 * x_0 + 0 * x_1 + 0.077 * x_2 &= 3.460 \\ 0 * x_0 + 1 * x_1 + 1.462 * x_2 &= 9.769 \\ 0 * x_0 + 0 * x_1 - 2.154 * x_2 &= -12.923\end{aligned} \tag{6}$$

Analogicznie postępujemy z wierszem 3:

$$\begin{aligned}1 * x_0 + 0 * x_1 + 0.077 * x_2 &= 3.460 \\ 0 * x_0 + 1 * x_1 + 0 * x_2 &= 0.997 \\ 0 * x_0 + 0 * x_1 + 1 * x_2 &= 6\end{aligned} \tag{7}$$

Odejmujemy:

$$\begin{aligned}1 * x_0 + 0 * x_1 + 0 * x_2 &= 2.998 \\ 0 * x_0 + 1 * x_1 + 0 * x_2 &= 0.997 \\ 0 * x_0 + 0 * x_1 + 1 * x_2 &= 6\end{aligned} \tag{8}$$

Z powyższego układu równań wynika wprost, że:

$$\begin{aligned}x_0 &= 2.998 \\ x_1 &= 0.997 \\ x_2 &= 6\end{aligned} \tag{9}$$

2. Wykonanie ćwiczenia
   Celem ćwiczenia jest rozwiązanie zadanej macierzy:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & (\omega^2h^2-2) & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & (\omega^2h^2-2) & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & (\omega^2h^2-2) & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & (\omega^2h^2-2) & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & (\omega^2h^2-2) & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} A \\ v_0h \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

Do tego celu uzyjemy własnoręcznie napisanego programu, wykorzustującego biblioteki numeryczne *nrutil*

Listing 1: main.cpp

```
#include <iostream>
#include <nrutil.h>
#include <nrutil.c>
#include <gaussj.c>
```

```
using namespace std;
int main()
{
        float h=0.1,v=0,A=1.0,w=1,tm;
        tm=2*3.14*3.0;
        int n=(int)tm/h;
        float **a,**b;
        a=matrix(1,n,1,n);
        b=matrix(1,n,1,1);
        int i,j;
        cout<<"n="<<n<<endl;
        for(i=1;i<=n;i++)
        {
                b[i][1]=0.0;
                for(j=1;j<=n;j++)
                {
                        a[i][j]=0.0;
                        if(i==j) a[i][j]=1.0;
                        if((i-1)==j) a[i][j]=w*w*h*h-2.0;
                        if((i-2)==j) a[i][j]=1.0;
                }
        }

        a[2][1]=-1.0;
        b[1][1]=A;
        b[2][1]=v*h;

        gaussj(a,n,b,1);
        FILE *file=fopen("dane.dat","w");
        for(i=1;i<n;i++) fprintf(file,"%f %f\n",h*i,b[i][1]);
        fclose(file);
        return 0;
};
```

W efekcie czego otrzymamy wynik:

Listing 2: dane.dat

```
0.100000 1.000000
0.200000 1.000000
0.300000 0.990000
0.400000 0.970100
0.500000 0.940499
0.600000 0.901493
0.700000 0.853472
0.800000 0.796916
0.900000 0.732391
1.000000 0.660542
1.100000 0.582088
1.200000 0.497813
```

1.300000	0.408559
1.400000	0.315220
1.500000	0.218729
1.600000	0.120051
1.700000	0.020172
1.800000	−0.079909
1.900000	−0.179191
2.000000	−0.276680
2.100000	−0.371403
2.200000	−0.462412
2.300000	−0.548797
2.400000	−0.629693
2.500000	−0.704293
2.600000	−0.771850
2.700000	−0.831688
2.800000	−0.883210
2.900000	−0.925899
3.000000	−0.959329
3.100000	−0.983166
3.200000	−0.997171
3.300000	−1.001205
3.400000	−0.995226
3.500000	−0.979295
3.600000	−0.953571
3.700000	−0.918312
3.800000	−0.873869
3.900000	−0.820688
4.000000	−0.759299
4.100000	−0.690318
4.200000	−0.614433
4.300000	−0.532404
4.400000	−0.445051
4.500000	−0.353248
4.600000	−0.257912
4.700000	−0.159997
4.800000	−0.060482
4.900000	0.039638
5.000000	0.139362
5.100000	0.237692
5.200000	0.333645
5.300000	0.426261
5.400000	0.514615
5.500000	0.597823
5.600000	0.675052
5.700000	0.745531
5.800000	0.808555
5.900000	0.863493
6.000000	0.909796
6.100000	0.947001

6.200000	0.974736
6.300000	0.992724
6.400000	1.000784
6.500000	0.998837
6.600000	0.986901
6.700000	0.965096
6.800000	0.933640
6.900000	0.892848
7.000000	0.843127
7.100000	0.784975
7.200000	0.718973
7.300000	0.645781
7.400000	0.566132
7.500000	0.480821
7.600000	0.390702
7.700000	0.296676
7.800000	0.199683
7.900000	0.100693
8.000000	0.000697
8.100000	−0.099307
8.200000	−0.198318
8.300000	−0.295345
8.400000	−0.389419
8.500000	−0.479598
8.600000	−0.564982
8.700000	−0.644716
8.800000	−0.718002
8.900000	−0.784109
9.000000	−0.842375
9.100000	−0.892216
9.200000	−0.933136
9.300000	−0.964724
9.400000	−0.986665
9.500000	−0.998739
9.600000	−1.000826
9.700000	−0.992904
9.800000	−0.975054
9.900000	−0.947453
10.000000	−0.910377
10.100000	−0.864197
10.200000	−0.809376
10.300000	−0.746461
10.400000	−0.676081
10.500000	−0.598940
10.600000	−0.515810
10.700000	−0.427521
10.800000	−0.334958
10.900000	−0.239045
11.000000	−0.140741

11.100000	-0.041030
11.200000	0.059091
11.300000	0.158621
11.400000	0.256565
11.500000	0.351944
11.600000	0.443803
11.700000	0.531224
11.800000	0.613333
11.900000	0.689308
12.000000	0.758390
12.100000	0.819889
12.200000	0.873188
12.300000	0.917756
12.400000	0.953146
12.500000	0.979004
12.600000	0.995072
12.700000	1.001190
12.800000	0.997296
12.900000	0.983429
13.000000	0.959727
13.100000	0.926428
13.200000	0.883865
13.300000	0.832463
13.400000	0.772736
13.500000	0.705283
13.600000	0.630776
13.700000	0.549961
13.800000	0.463647
13.900000	0.372696
14.000000	0.278019
14.100000	0.180561
14.200000	0.081298
14.300000	-0.018779
14.400000	-0.118667
14.500000	-0.217369
14.600000	-0.313898
14.700000	-0.407287
14.800000	-0.496603
14.900000	-0.580954
15.000000	-0.659494
15.100000	-0.731440
15.200000	-0.796072
15.300000	-0.852742
15.400000	-0.900886
15.500000	-0.940020
15.600000	-0.969754
15.700000	-0.989790
15.800000	-0.999929
15.900000	-1.000068

```
16.000000 -0.990207
16.100000 -0.970443
16.200000 -0.940975
16.300000 -0.902098
16.400000 -0.854199
16.500000 -0.797758
16.600000 -0.733339
16.700000 -0.661588
16.800000 -0.583220
16.900000 -0.499020
17.000000 -0.409830
17.100000 -0.316541
17.200000 -0.220088
17.300000 -0.121433
17.400000 -0.021564
17.500000 0.078521
17.600000 0.177821
17.700000 0.275342
17.800000 0.370110
```

Co po wygenerowaniu wykresu przy użyciu programu Gnuplot skryptem:

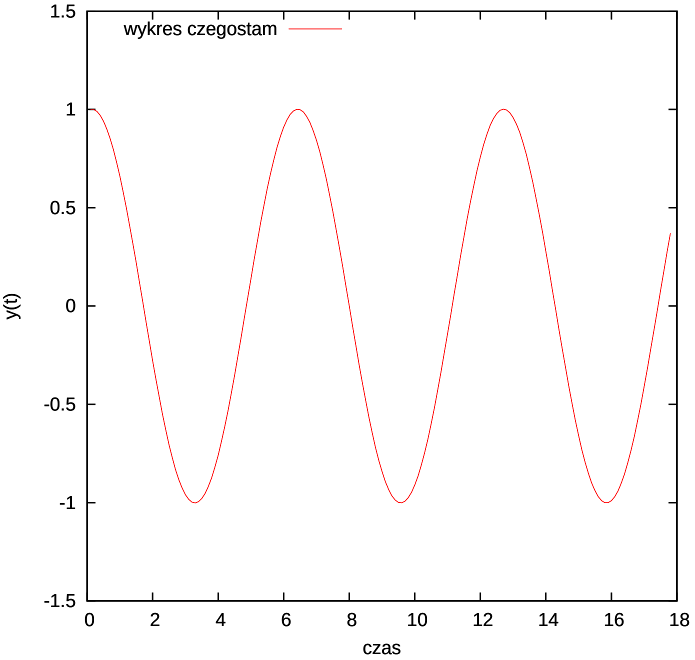
Listing 3: plot.sh

```
#!/usr/bin/gnuplot
set term postscript  color enhanced eps
set out "plot.eps"
set key left top
set size square
set xlabel "czas"
set ylabel "y(t)"
plot "dane.dat" using 1:2 title "wykres czegostam" w l
replot
```

da nam wynik jak na Rys. 1

3. Wniski
   Jak widać na wykresie, wynik jest zgodny z przewidywanym, co może świadczyć o poprawności metody. Trzeba jednak zwrócić uwagę na ewentualne niedokładności wynikające z dokładności zapisu liczb zmiennoprzecinkowych.
   Ponadto, metoda Gaussa-Jordana sprawdza się przy małych macierzach. w przypadku większych macierzy algorytm jest nieefektywny:
   dla h=0.1:

Rysunek 1: Wynik




```
i9fabryk@fatcat:~/numerki/1$ time ./main
n=179

real    0m0.094s
user    0m0.080s
sys     0m0.000s
```

natomiast dla h=0.01:

```
i9fabryk@fatcat:~/numerki/1$ time ./main
n=1800

real    1m22.110s
user    1m21.770s
sys     0m0.130s

i9fabryk@fatcat:~/numerki/1$ time ./main
n=17999
^C
```

```
real    3m40.085s
user    3m38.290s
sys     0m1.640s
```

Program nie zakończł działania w rozsądnym czasie. Przypuszczalny oczekiwany czas to 24h.

h	czas [s]
0.1	0.094
0.01	88.110
0.001	???