Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Systemy mikroprocesorowe w sterowaniu

Sprawozdanie z projektu nr 1

Mateusz Dziwulski, Jakub Szczepański

Warszawa, 2017

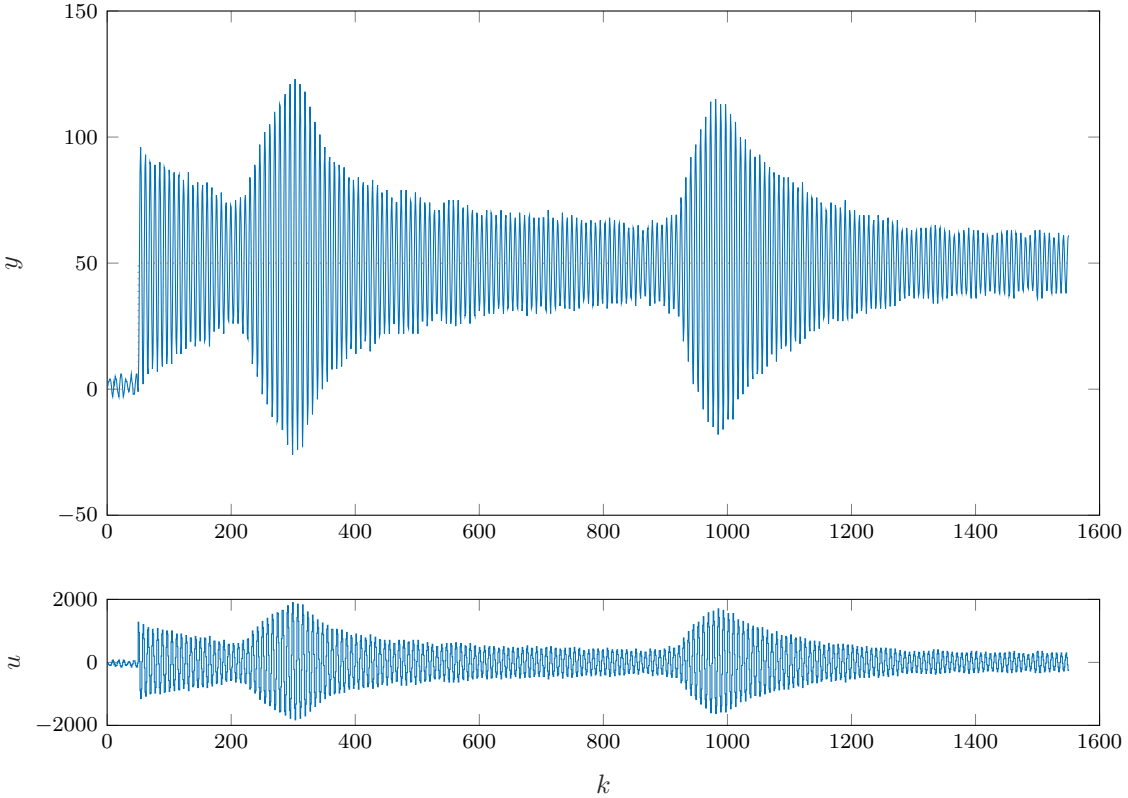
Spis treści

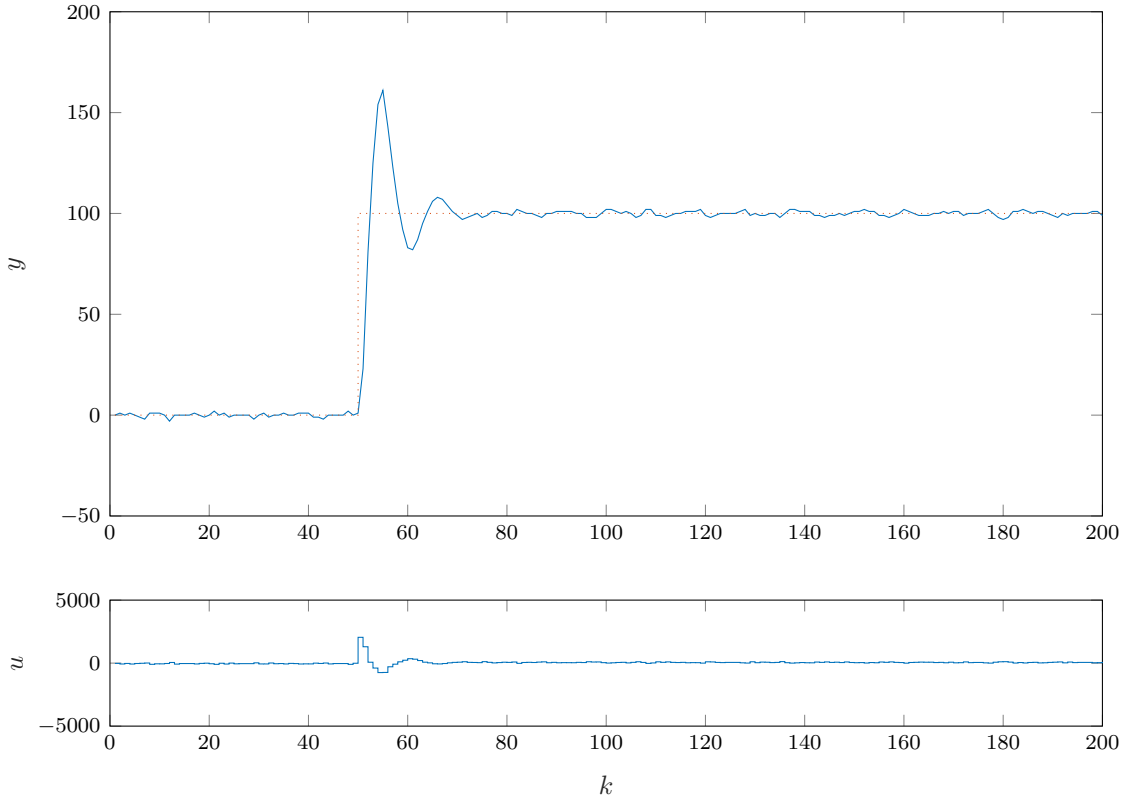
1. Wstęp

Dla prostoty i przejrzystości sprawozdania wykorzystane zostały oznaczenia pierwotnie wprowadzone w skrypcie z aktualnego laboratorium.

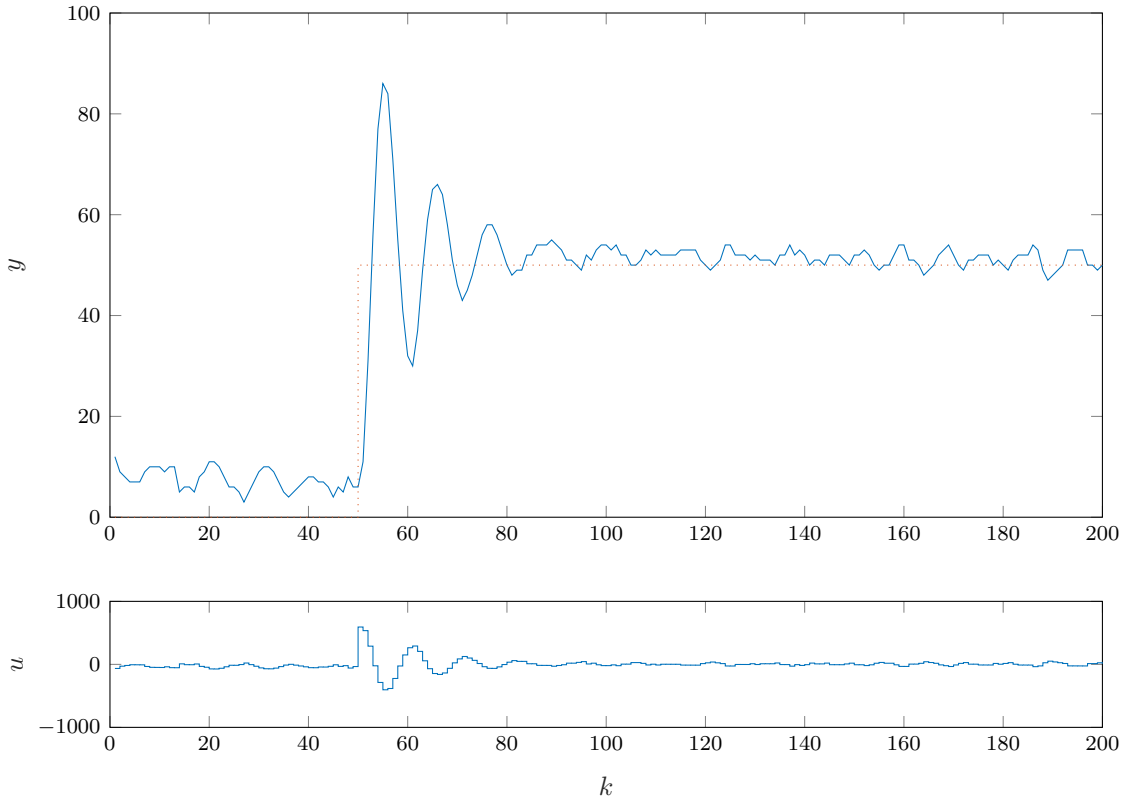
2. Algorytm PID

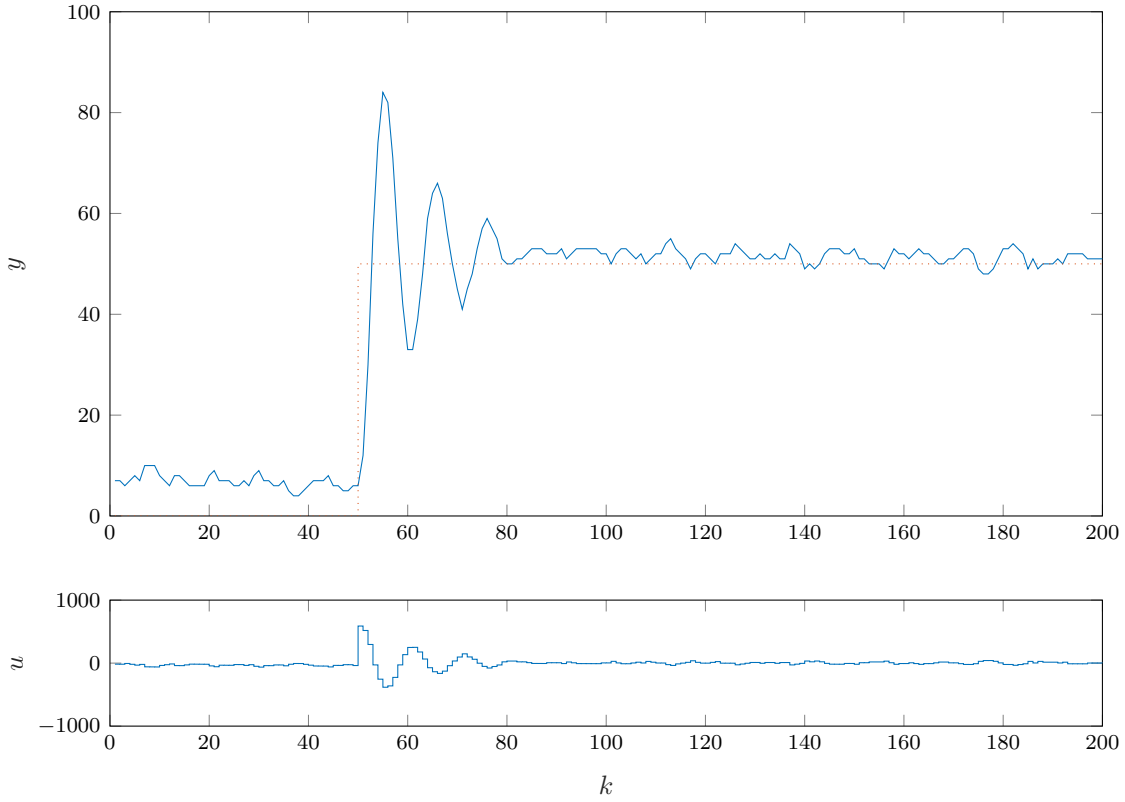
Napisać coś o różnicy w taktowaniu.

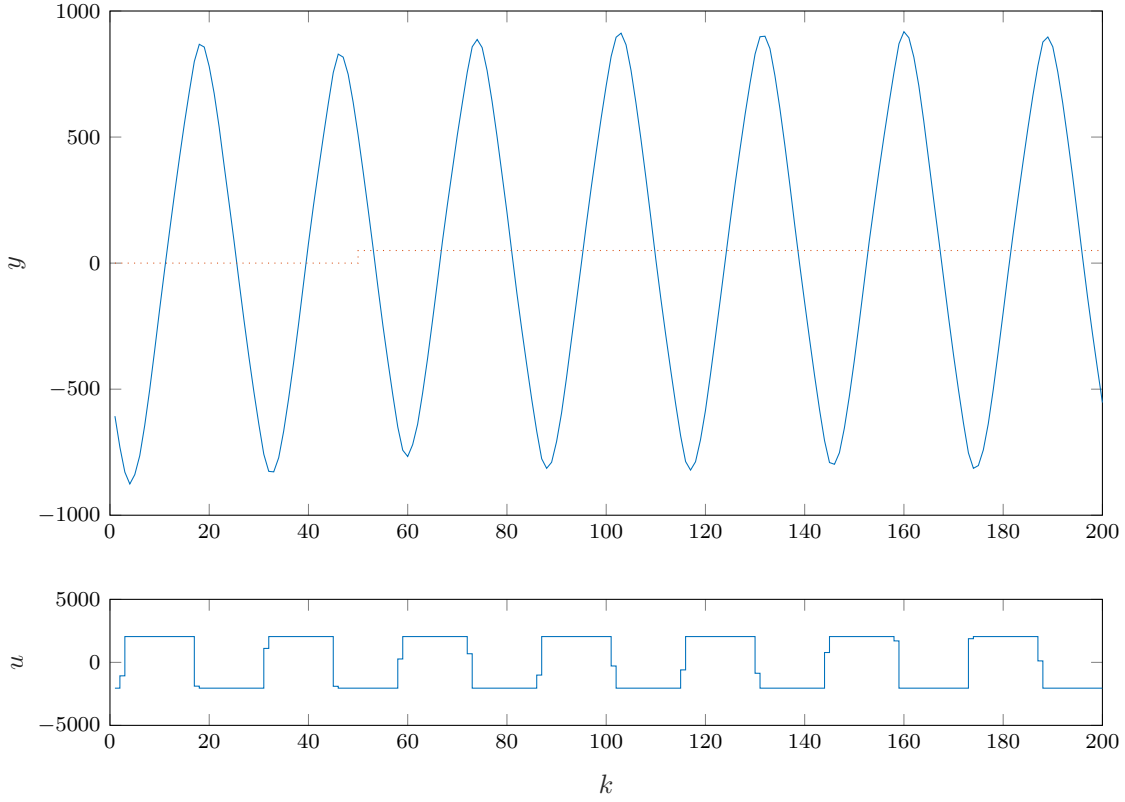


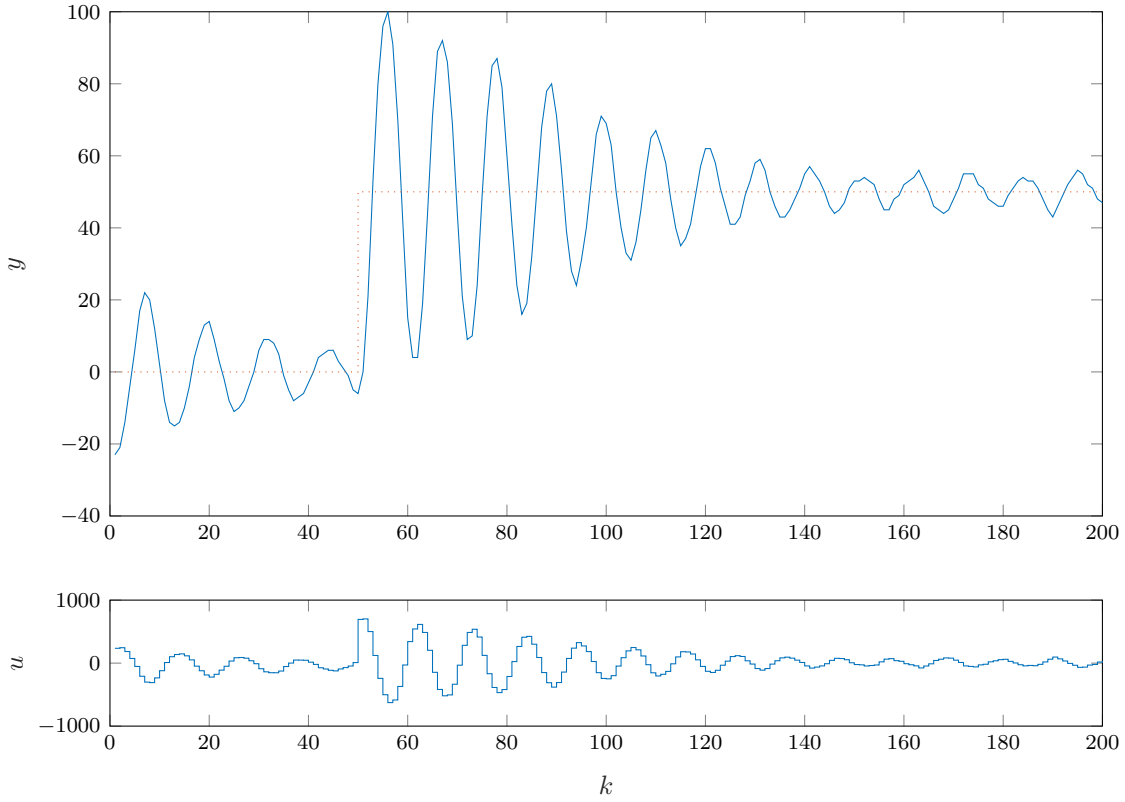


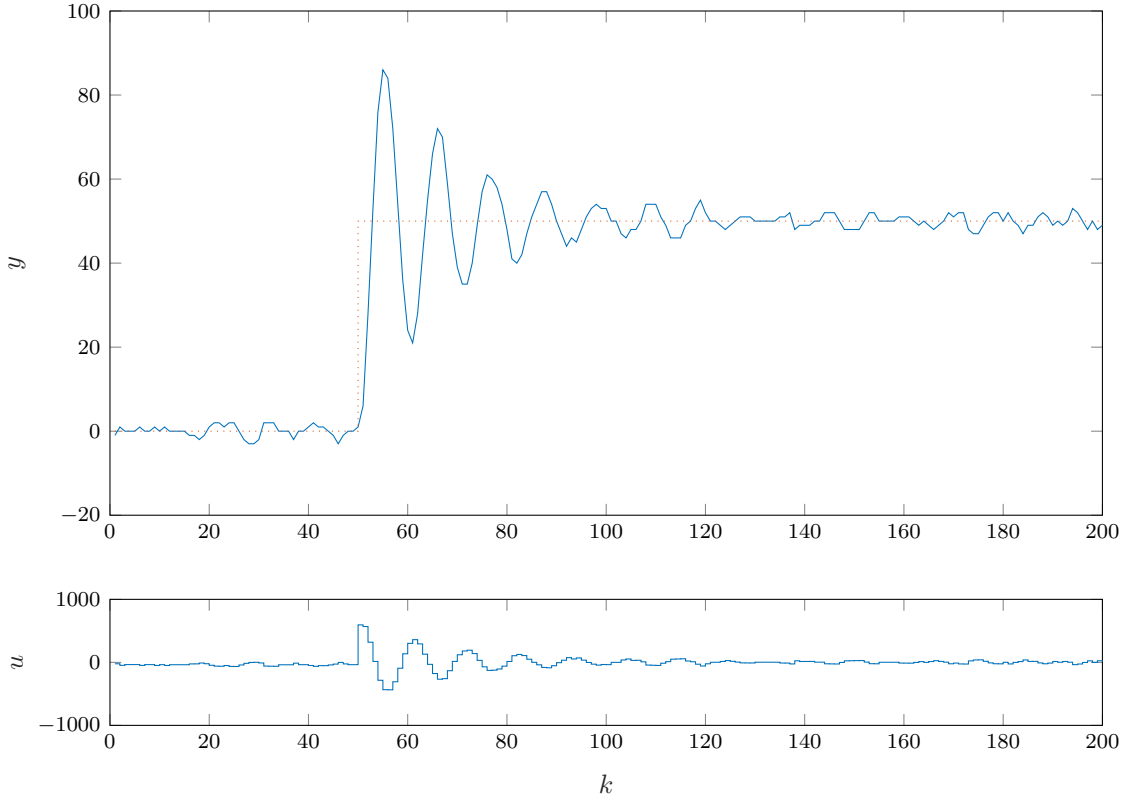
Tu wspomnieć, że P to osc kryt.

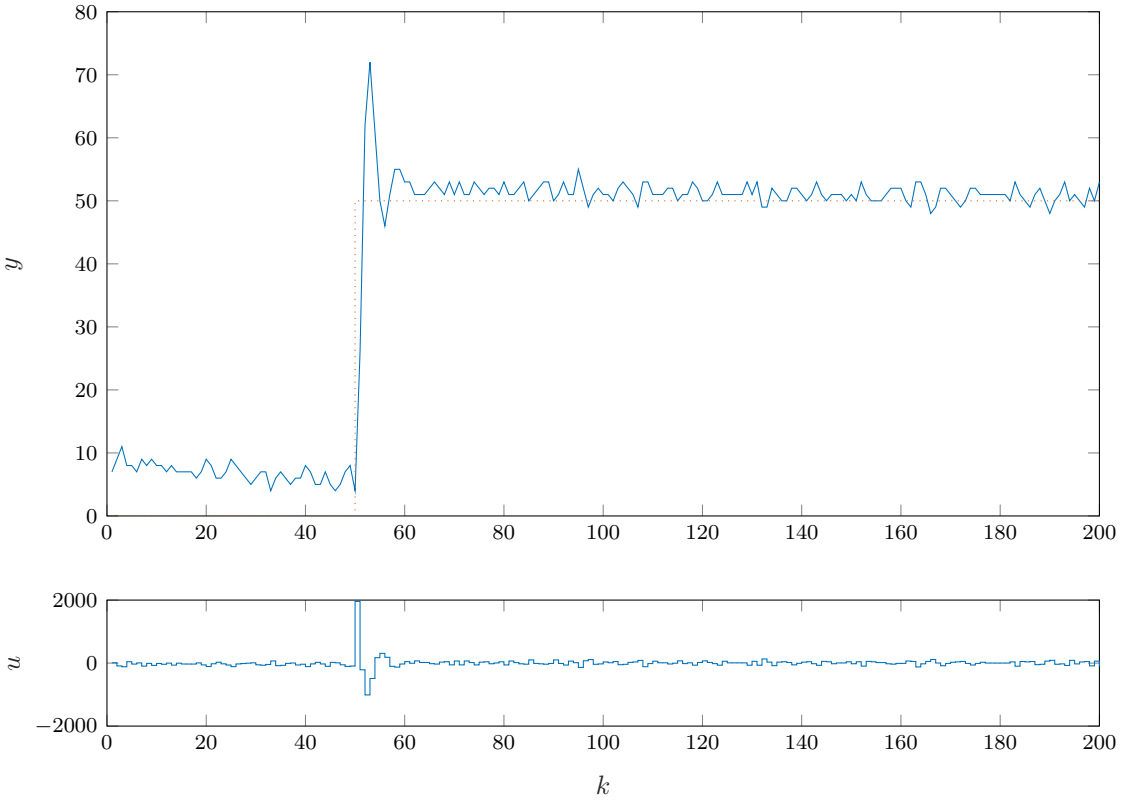


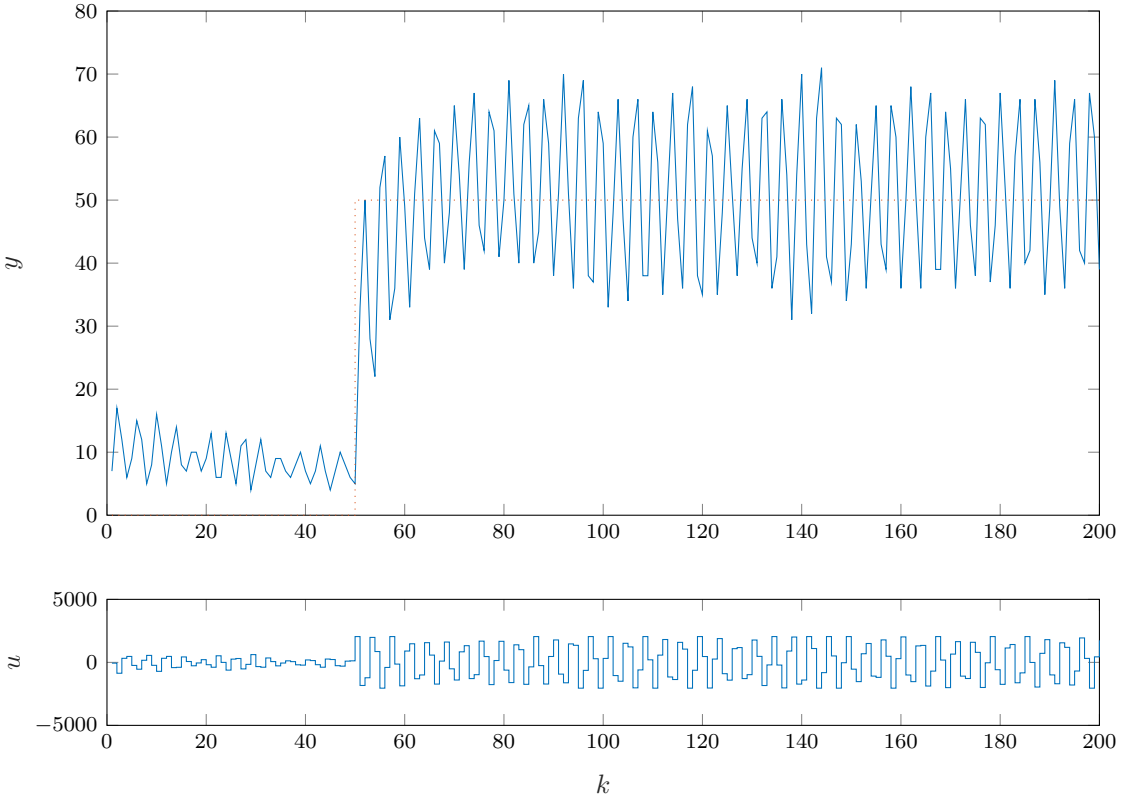

y
u
k

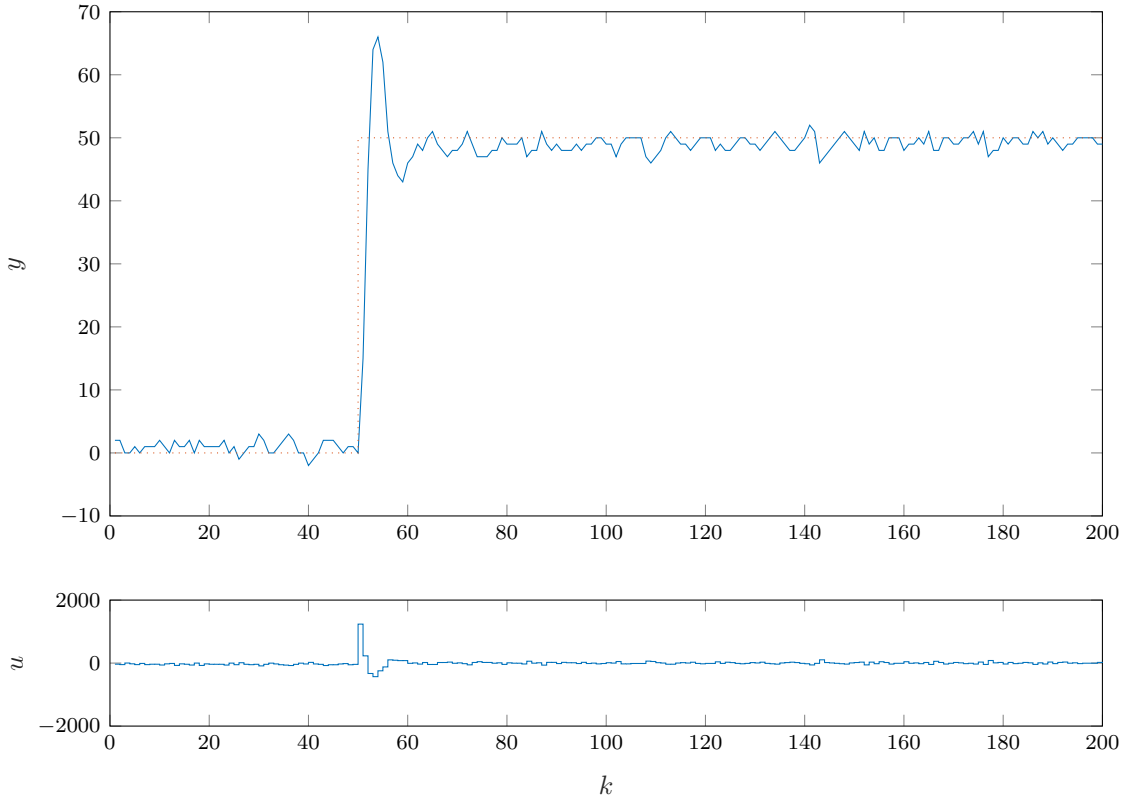

y
u
k

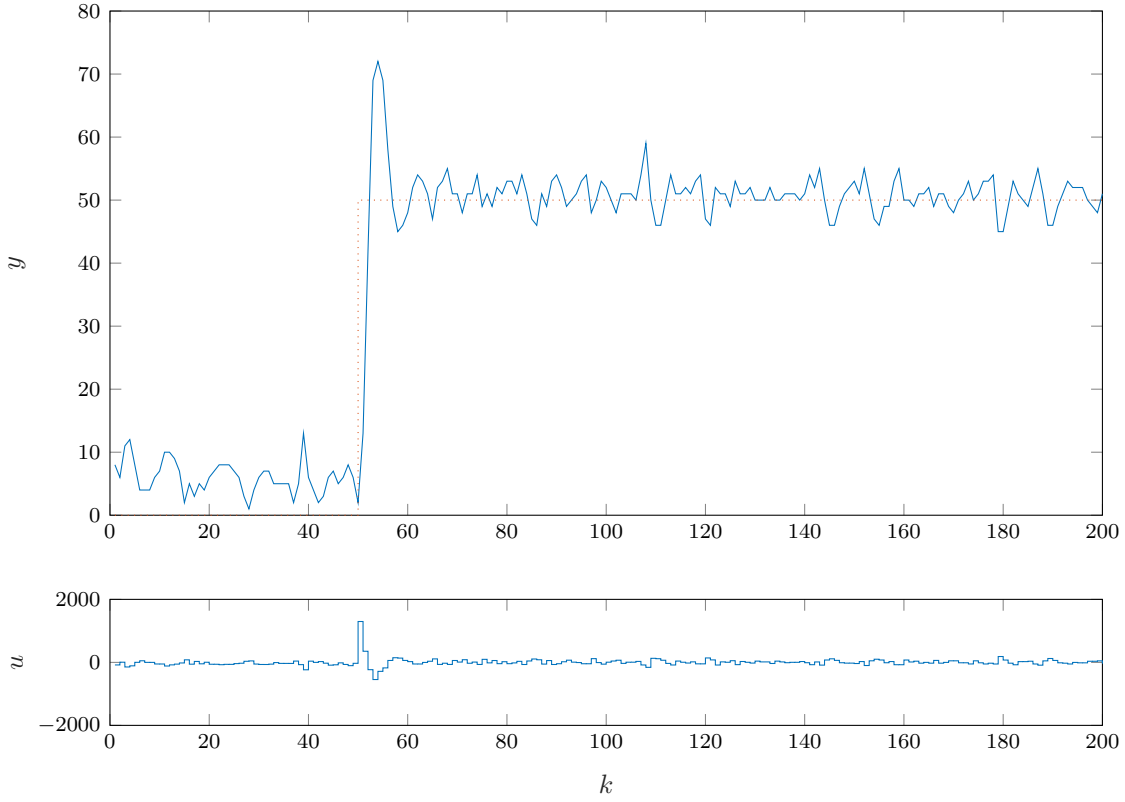

y
u
k

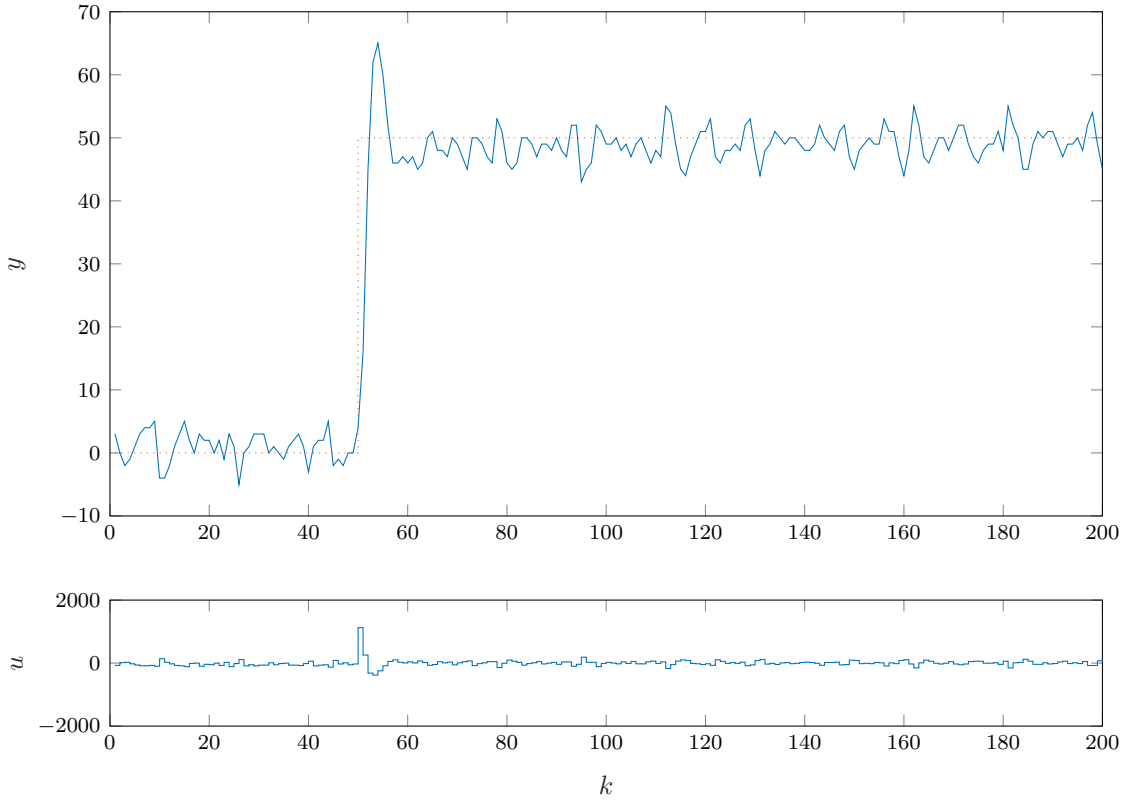

y
u
k


y
u
k


y
u
k


y
70
60
50
40
30
20
10
0
−10
0
20
40
60
80
100
120
140
160
180
200
u
2000
0
−2000
k

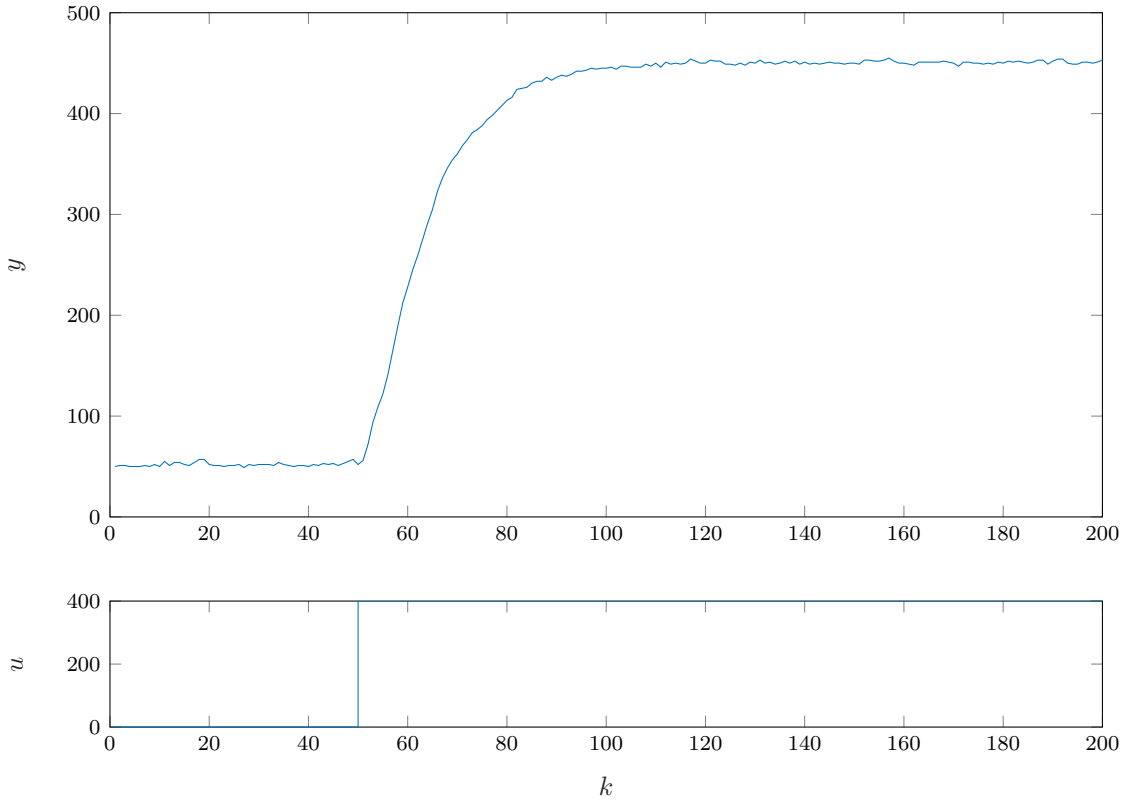

y
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200
u
2000
0
−2000
k


70
60
50
40
30
20
10
0
−10
y
0
20
40
60
80
100
120
140
160
180
200
2000
0
−2000
u
k

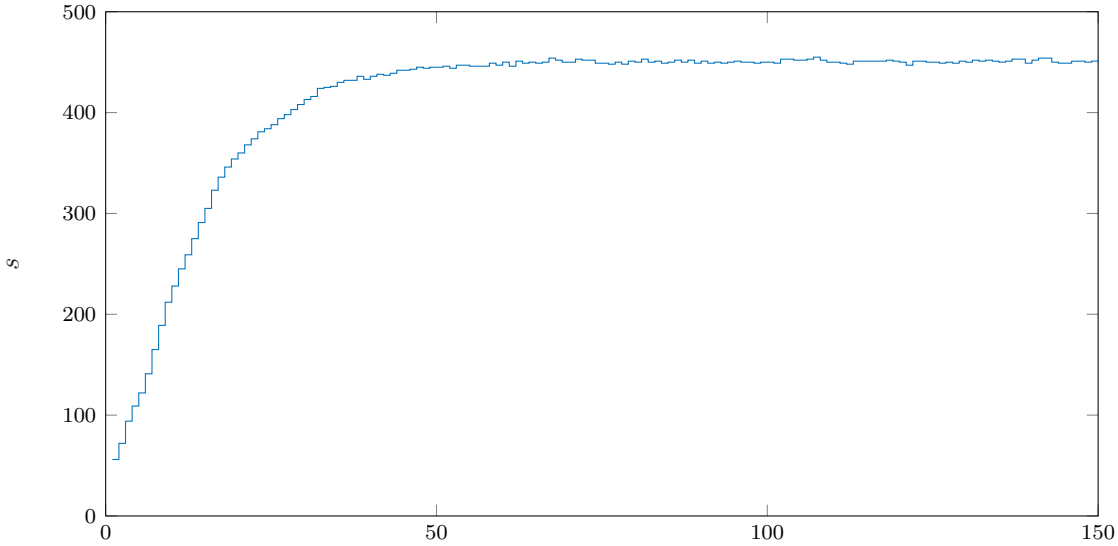
3. Algorytm DMC

Do implementacji cyfrowego algorytmu DMC na mikronoktrolerze oprócz plików źródłowych zastosowano skrypt z MATLABa wykonujący niezbędne przekształcenia uzyskanej odpowiedzi skokowej i zadanych parametrów na użyteczne wartości.

Odpowiedź skokową uzyskano poprzez zmierzenie odpowiedzi układu na skok sygnału wejściowego o 400, jak pokazano na rysunku 3.1. Obcięta odpowiedź, stosowana w samych obliczeniach (przed normalizacją do skoku jednostkowego z punktu pracy), została przedstawiona na rysunku 3.2.



Rys. 3.1: Odpowiedź skokowa układu



Rys. 3.2: Obcięta odpowiedź skokowa układu

Skrypt, którego użyto do wyliczenia macierzy DMC zawarto w pliku `params.m`:

```
Ypp = 50;
dU = 400;
D = ?; %parametry - zmieniane w kolejnych probach
N = ?;
Nu = ?;
lambda = ?;

load step.mat %zaladowanie pliku z odpowiedzia skokowa
s = (s - Ypp)/dU;


M=zeros(N,Nu);
for i=1:N
  for j=1:Nu
    if (i>=j)
      M(i,j)=s(i-j+1);
    end;
  end;
end;

Mp=zeros(N,D-1);
for i=1:N
  for j=1:D-1
    if i+j<=D
      Mp(i,j)=s(i+j)-s(j);
    else
      Mp(i,j)=s(D)-s(j);
    end;
  end;
end;

K=((M'*M+lambda*eye(Nu))^-1)*M';
```

```
Ku=K(1,:)*Mp;
Ke=sum(K(1,:));


%wypisanie odpowiednio sformatowanego przypisania w C
fprintf(strcat('  static float Ke = ',sprintf('%.4f',Ke),';\n',...
'  static float Ku[] = {',sprintf('%.4f,',Ku),'\b};\n'));
```

Kod realizujący główną pętlę regulatora DMC zawarty w pliku `main.c` przedstawia się następująco:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  if(htim->Instance == TIM2){
    static float y = 0.0f;
    static float u = 0.0f, u_ = 0.0f; //u_ = u(k-1), u = u(k)

    static float yzad = 0.0f;
    static int iter = 0;
    if (++iter > 150) {
      //yzad = rand() % 4096 - 2048;
      yzad  = 200 - yzad; //naprzemienne skoki miedzy 0 i zadana
      iter = 0;           //wartoscia
    }

    y = (input-2048.0f); // przejscie z 0 - 4095 do -2048 - 2047

        //inicjalizacja
    static const int D = 50;
    static float e = 0, dup[D-1], du = 0;
    static int it = 0;

    //Parametry uzyskane z matlaba
    static float Ke = ??;
    static float Ku[] = ??;


    e = yzad - y;

    du = Ke * e;

    //reczne odjecie iloczynu wektorow Ku i dUp
    for(it = 0; it < D-1; it++)
      du -= Ku[it] * dup[it];

        //przesuniecie wektora dUp
    for(it = D-2; it >= 1; it--)
      dup[it] = dup[it - 1];
    dup[0] = du;

        //wyznaczenie nowego sterowania
    u = u_ + du;
```

```
    u_ = u;

    if(u >  2047.0f) u =  2047.0f; %limity
    if(u < -2048.0f) u = -2048.0f;


    output = u+2048.0f; // przejscie z -2048 - 2047 do 0 - 4095

    updateControlSignalValue(output);

    while(HAL_UART_GetState(&huart) == HAL_UART_STATE_BUSY_TX);
    sprintf(text,"U=%+8.2f;Y=%+8.2f;Yzad=%+8.2f;",u,y,yzad); // 36 znakow
    if(HAL_UART_Transmit_IT(&huart, (uint8_t*)text, 36)!= HAL_OK){
      Error_Handler();
    }
  }
}
```

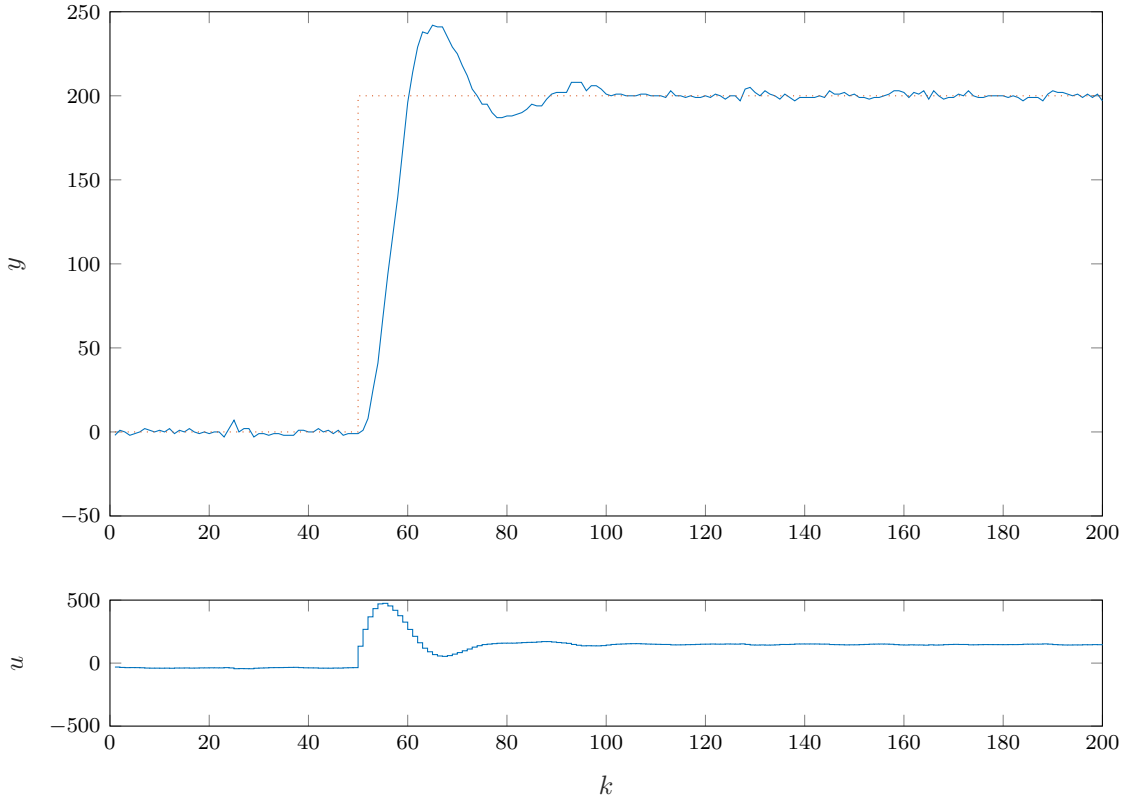
Testy konkretnych parametrów regulatora przeprowadzano z większym skokiem wartości zadanej niż w przypadku PID - pozwoliło to na lepsze zobaczenie i ocenienie różnic między konkretnymi iteracjami. Na wykresach od 3.3 do 3.8 można zobaczyć kolejne przebiegi.

Zgodnie z kolejnością podpunktów w skrypcie do laboratorium, najpierw dobierano jeden parametr, a następnie dla najlepszej jego wartości wybierano następny. Ich kolejność λ, następnie N_u, a na końcu N. Parametr D dobrano na podstawie stabilizacji odpowiedzi skokowej.

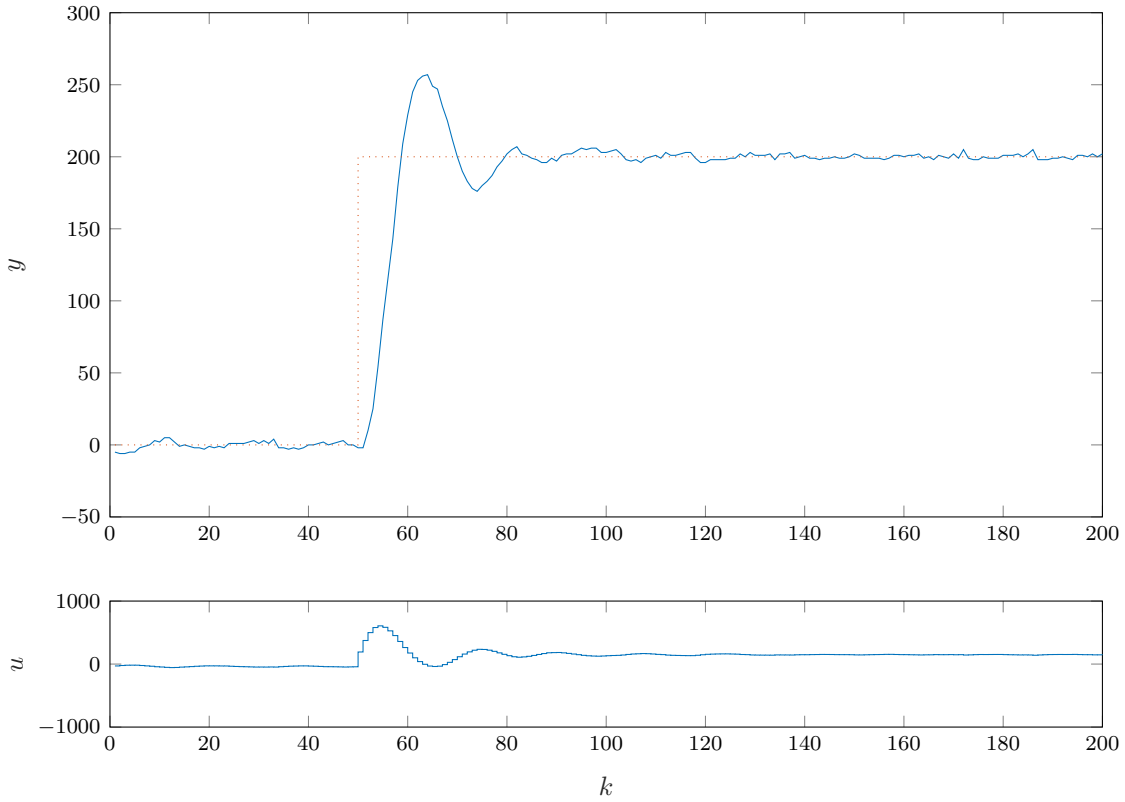
Zwiększenie λ skutkowało wygładzeniem przejścia między wartościami przed i po skokiem. Zmniejszenie jej powodowało przyspieszenie tego przejścia - aż do pewnej wartości minimalnej, po której przy której pojawiały się oscylacje. Końcowa wartość to $0,5$.

Zmiana wartości horyzontu sterowania skutkowała bardzo niewielkimi zmianami. Zdecydowano się na wartość niższą niż początkowa, jako skutkująca nieznacznie mniejszym przeregulowaniem.

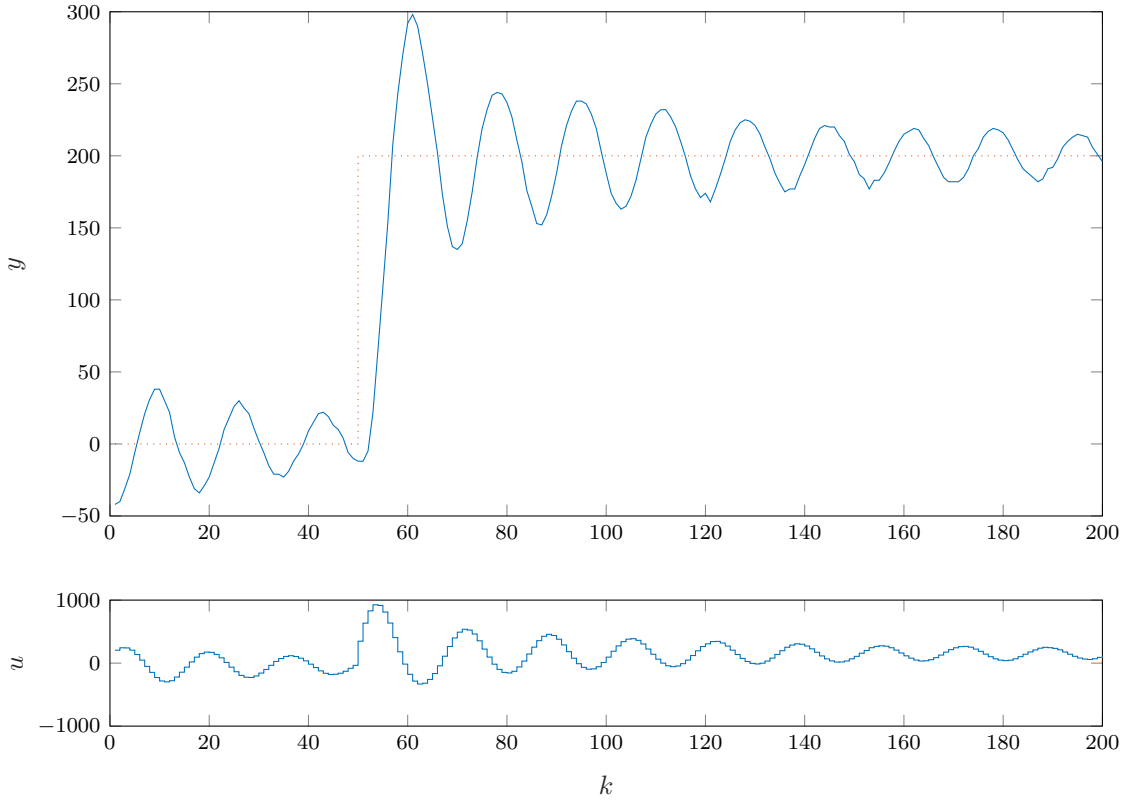
Zmiana wartości horyzontu predykcji również skutkowała prawie niewidocznymi zmianami - zmniejszenie jego wartości skutkowało względnym wygładzeniem przebiegu, ale też nieznacznie zwiększała przeregulowanie. Do testów porównawczych z PID wybrano regulator z rysunku 3.6, o wartości $N = D = 50$.



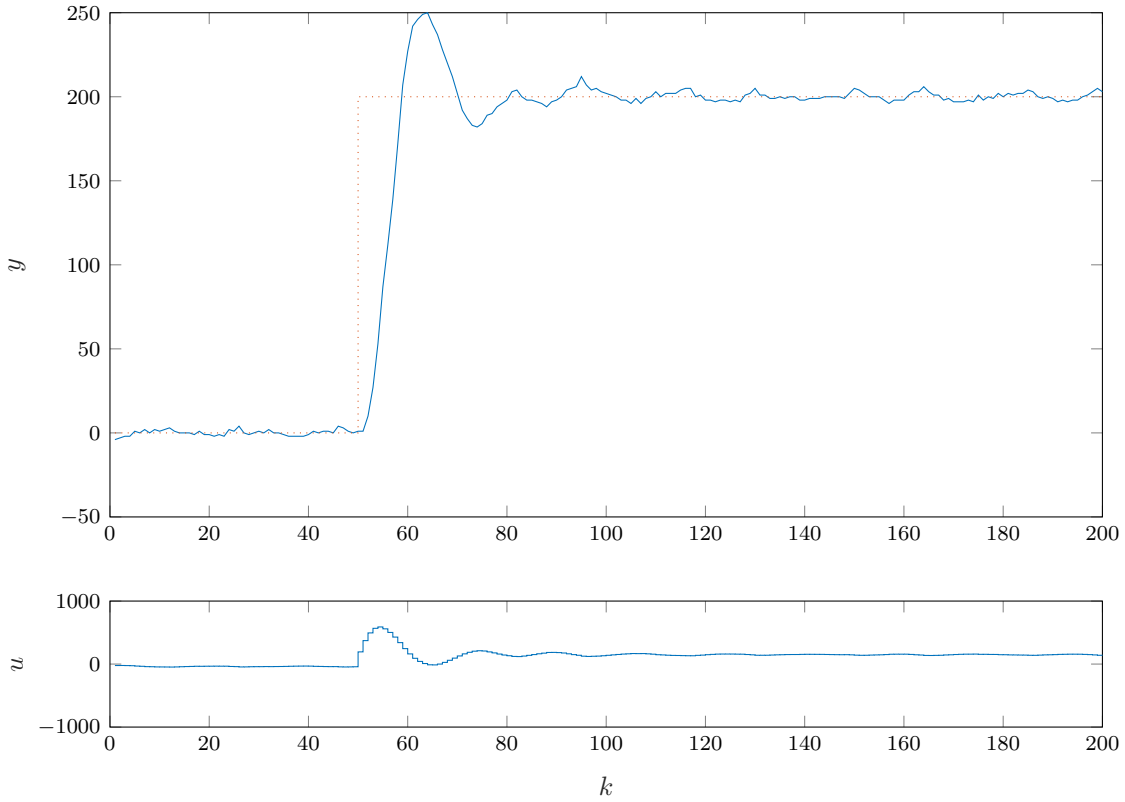
Rys. 3.3: $D = 50$, $N = 50$, $Nu = 50$, $\lambda = 1$



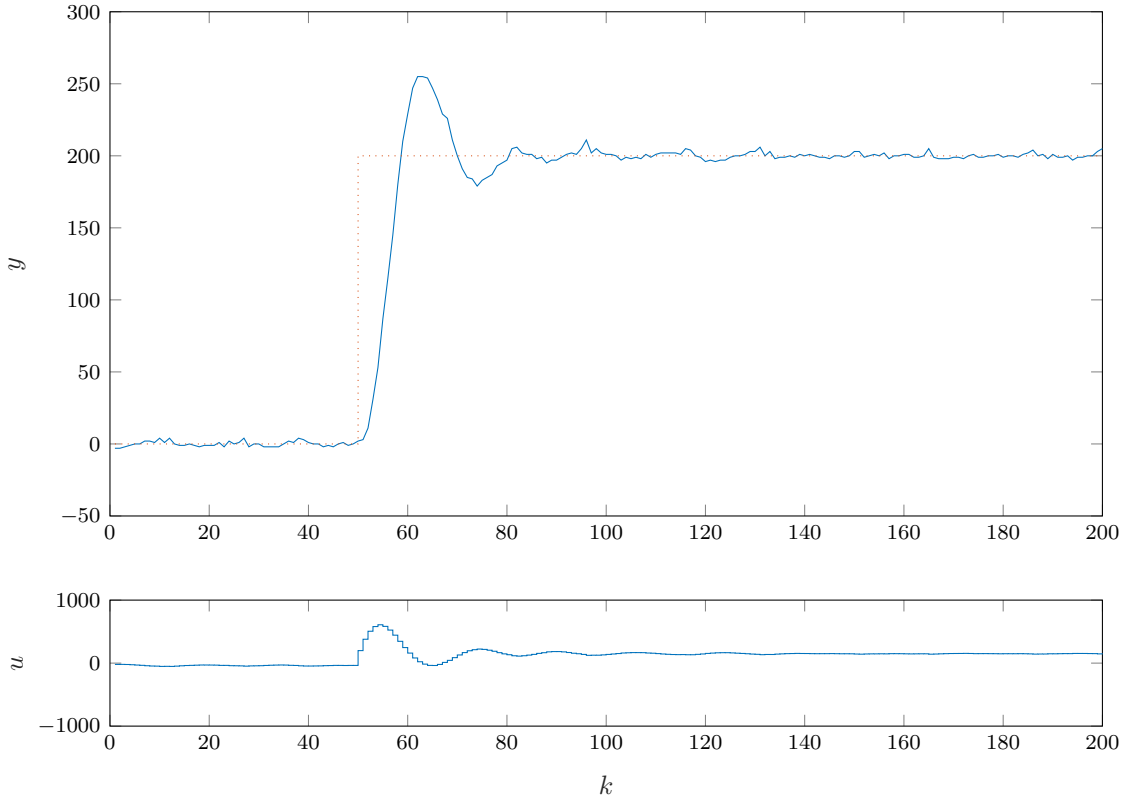
Rys. 3.4: $D = 50$, $N = 50$, $Nu = 50$, $\lambda = 0,5$



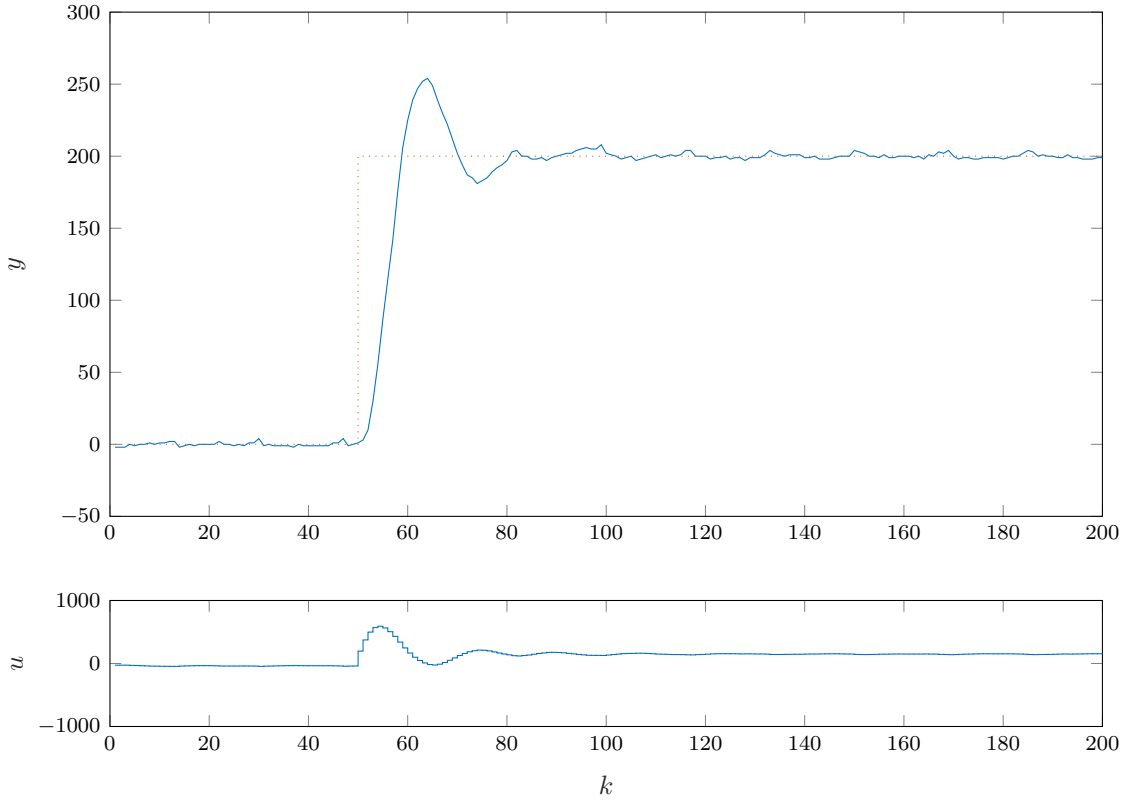
Rys. 3.5: $D = 50$, $N = 50$, $Nu = 50$, $\lambda = 0,2$ - oscylacje



Rys. 3.6: $D = 50$, $N = 50$, $Nu = 10$, $\lambda = 0,5$ - najlepszy z uzyskanych regulatorów

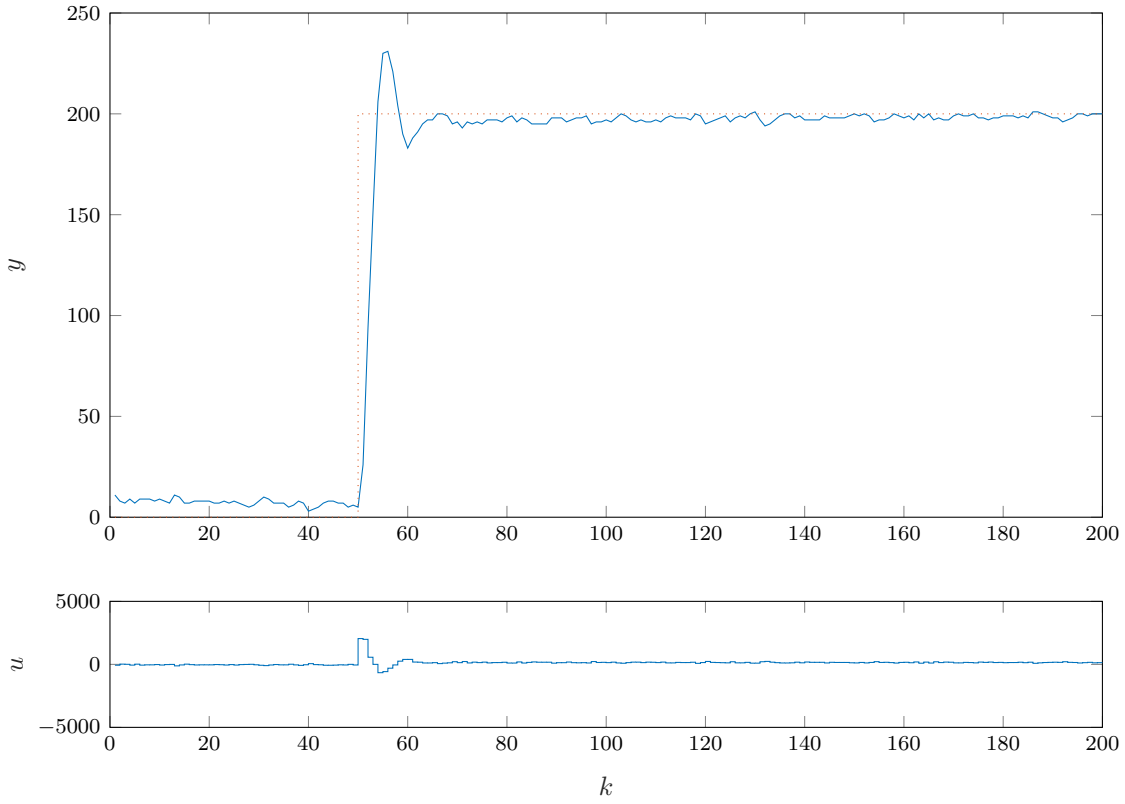


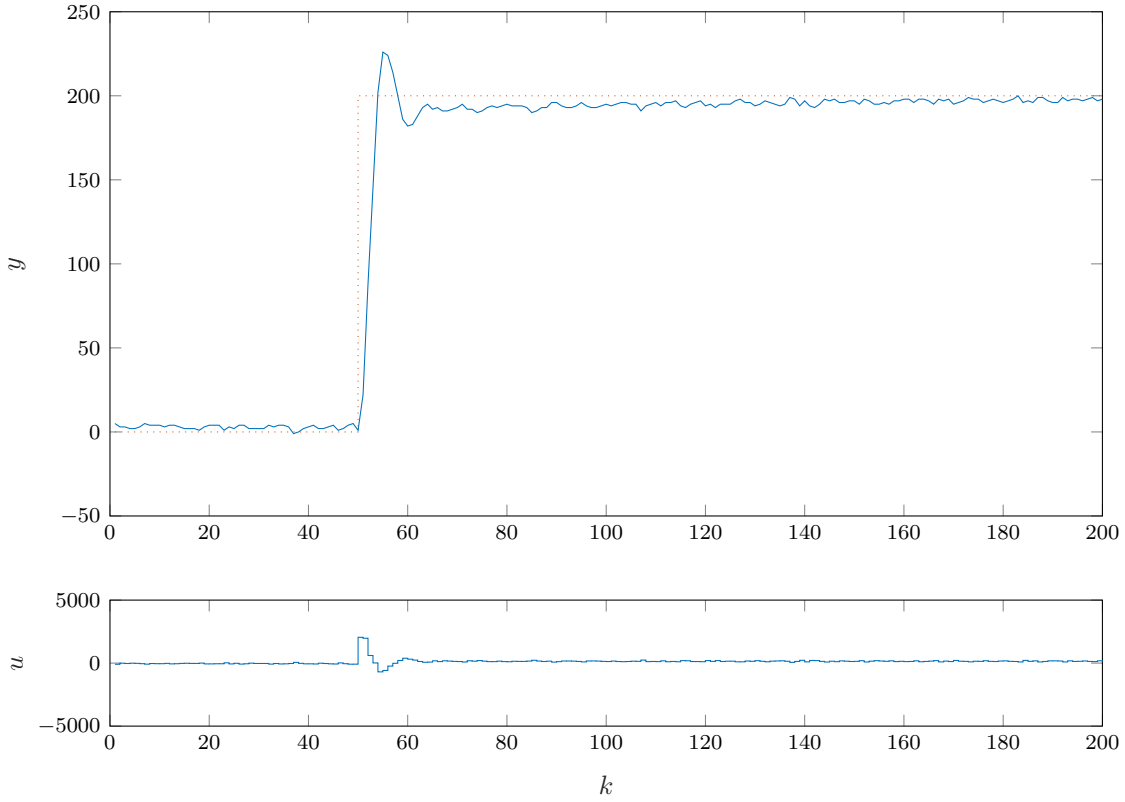
Rys. 3.7: $D = 50$, $N = 20$, $Nu = 10$, $\lambda = 0,5$

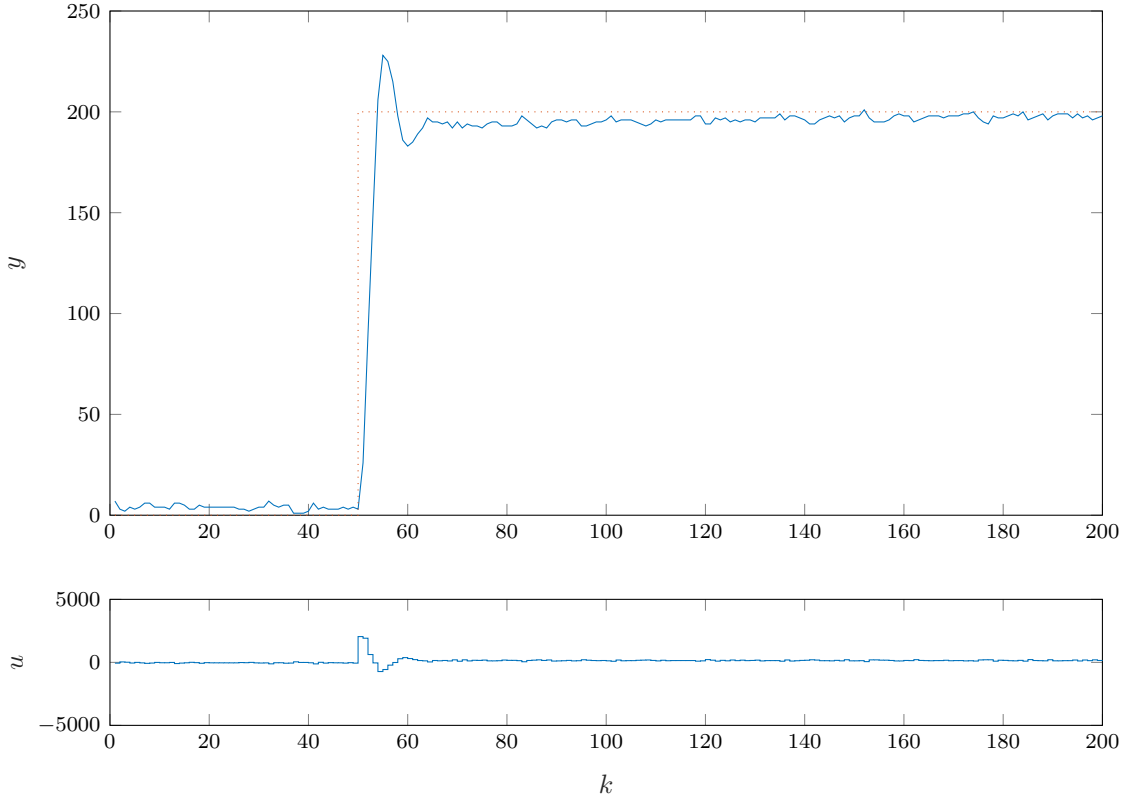


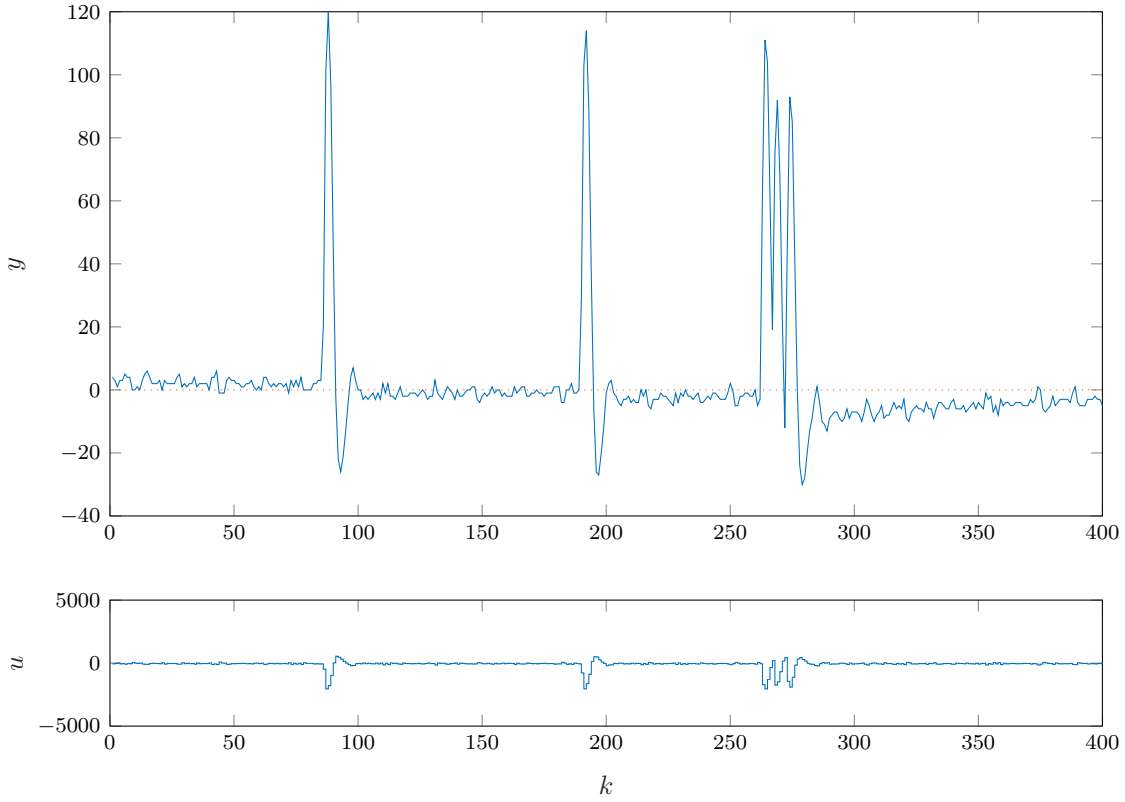
Rys. 3.8: $D = 50$, $N = 30$, $Nu = 10$, $\lambda = 0,5$

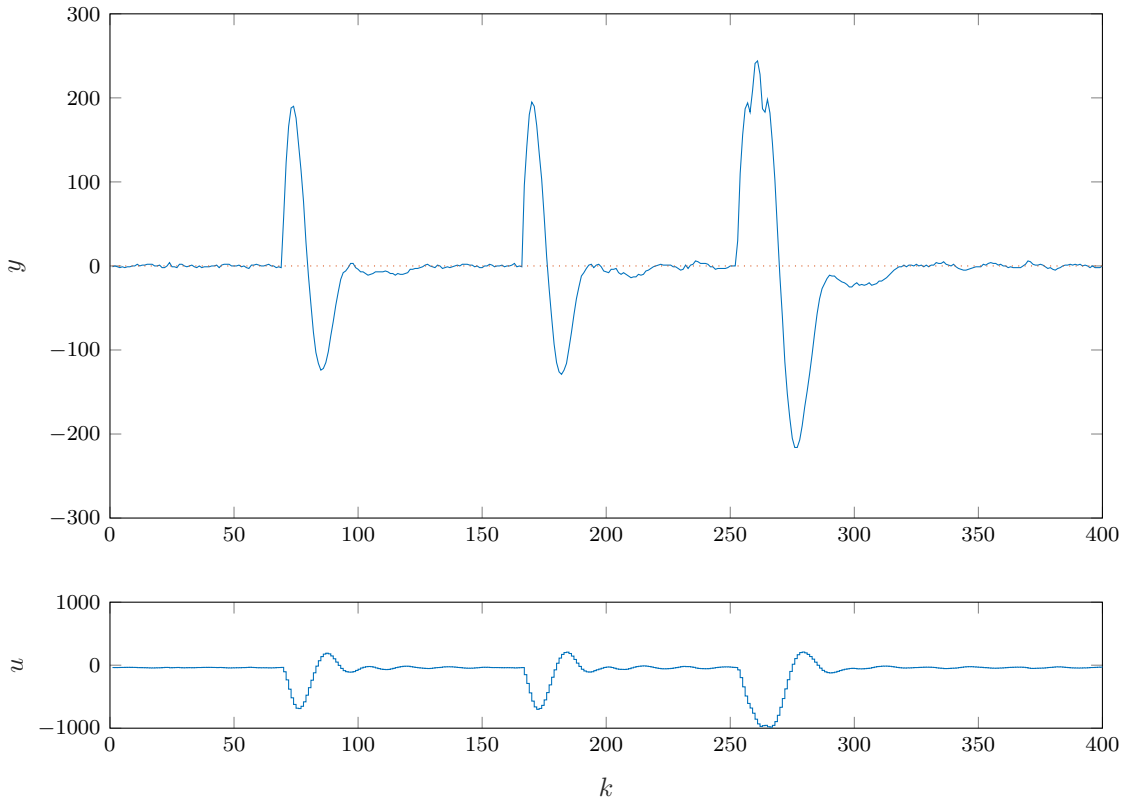
4. Porównanie najlepszych realizacji




250
200
150
100
50
0
−50
y
0
20
40
60
80
100
120
140
160
180
200
5000
0
−5000
u
k


250
200
150
100
50
0
y
0
20
40
60
80
100
120
140
160
180
200
5000
0
−5000
u
k


y
u
k


y
u
k