Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Mariusz Kamola

Sterowanie z powtarzaną optymalizacją

Optymalizacja z nierównomierną dyskretyzacją sterowania

Ocena:

Podpis:

Data:

praca dyplomowa

Warszawa, maj 1997 r.

Spis treści

1. Przedstawienie problemu

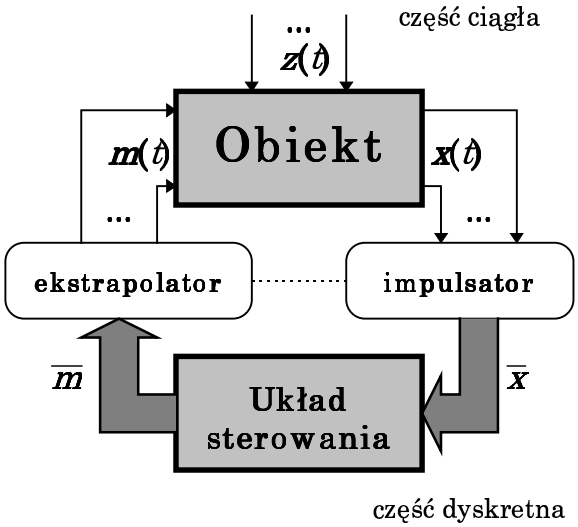
Gwałtowny wzrost możliwości komputerów w ostatnich latach zmienił podejście do niektórych, opracowanych już wcześniej, metod sterowania obiektami dynamicznymi. Towarzyszy mu spadek cen pamięci operacyjnych i dyskowych, co pozwala zapomnieć o problemie przechowywania wielkiej liczby danych, uniemożliwiającym implementację niektórych algorytmów jeszcze 15 lat temu. Wzrosła szybkość współpracy z urządzeniami peryferyjnymi i upowszechniło się stosowanie pamięci wirtualnej w większości systemów operacyjnych — w następstwie czego procedury wymagające wielkich zasobów mogą już być uruchamiane na komputerach niższej klasy. Powiększyło się pole zastosowań wyrafinowanych metod sterowania.

Równie istotnym (obok pamięci) zagadnieniem jest szybkość wykonywania programów. Jej zwiększenie zawdzięczamy dwóm czynnikom. Pierwszy to może nie tak spektakularny, jak w przypadku pamięci, wzrost szybkości samego procesora, która corocznie bije rekord o kolejne 10 MHz. Drugi to zastosowanie przetwarzania równoległego, stanowiące jednocześnie wyzwanie do ulepszania starych i projektowania nowych algorytmów w pełni wykorzystujących wieloprocesorową strukturę komputera.

W tym kontekście można postrzegać prace wykonywane ostatnio w Instytucie Automatyki i Informatyki Stosowanej. Badania algorytmów programowania dynamicznego [5] czy też algorytmów sterowania z powtarzaną optymalizacją [6] dowodzą, że gra jest warta świeczki; że przy obecnym poziomie techniki komputerowej zaawansowane systemy sterowania zyskały rację bytu.

Przedmiotem mojej pracy jest sterowanie z powtarzaną optymalizacją obiektem dynamicznym. Stanowi ona uzupełnienie i rozwinięcie pracy [6] według wskazówek prof. Malinowskiego, któremu składam podziękowania za wydatną pomoc w przygotowaniach prowadzących do jej powstania. Rozdział bieżący zawiera sformułowanie problemu, którym się zajmowałem. W następnym staram się umotywować wybór właśnie tego algorytmu, przedstawić jego pozycję wśród innych popularnych metod sterowania. Rozdział trzeci obejmuje szczegółowy opis teoretyczny sterowania z powtarzaną optymalizacją ze szczególnym uwzględnieniem przypadku z horyzontem skończonym i nierównomiernym rozmieszczeniem chwil dyskretyzacji sterowania. Opis jest opatrzony komentarzami dotyczącymi niektórych zagadnień implementacyjnych. Rozdział czwarty opisuje badanie składowych algorytmu (całkowania, optymalizacji etc.) połączone z poszukiwaniem najlepszych wartości parametrów. Prowadzi ono do procedury projektowej, przedstawionej wraz z przykładem na końcu rozdziału. W rozdziale piątym znajdzie Czytelnik analizę innych własności algorytmu: wpływu opóźnień decyzyjnych, sterowania z ustalonym horyzontem lokalnym oraz porównanie wyników sterowania z powtarzaną optymalizacją z wynikami sterowania regulatorem PID.

Sterowanie obiektem dynamicznym odbywa się w układzie jak na rysunku obok (1).



Rys. 1 Struktura układu sterowania

Sterowany *obiekt* ma naturę ciągłą a *układ sterowania* — dyskretną. Dyskretyzacja stanu dokonuje się w *impulsatorze*: *stan* obiektu $\boldsymbol{x}(t)$ jest mierzony i podawany dalej ($\overline{\boldsymbol{x}}$) tylko w chwilach t_i zwanych *punktami pomiaru stanu*. Tworzą one zbiór T_s. Uciąglenie *sterowania* $\overline{\boldsymbol{m}}$ dokonuje się w *ekstrapolatorze*. Jest on zerowego rzędu (ZOH) co oznacza, że sterowanie wystawione przez układ sterowania w chwili t_i jest podtrzymywane aż do wystawienia następnego. Układ sterowania jest wielowymiarowy: $\boldsymbol{x} \in \Re^n$ jest wektorem *zmiennych stanu*, $\boldsymbol{m} \in \Re^r$ jest wektorem sterowań a $\boldsymbol{z} \in \Re^p$ — wektorem zakłóceń (czyli wejść niesterowalnych).

Obiekt jest opisany układem równań różniczkowych:

$$\dot{\boldsymbol{x}}(t) = \mathbf{f}\left(\boldsymbol{x}(t), \boldsymbol{m}(t), \boldsymbol{z}(t), t\right). \tag{1}$$

Niech proces sterowania rozpoczyna się w chwili t_0, $t_0 \in T_s$, nazywanej *początkiem sesji sterowania*; kończy zaś w chwili t_f nazywanej *horyzontem globalnym sterowania*. Dopuszcza się $t_f = \infty$.

Szczegóły działania układu sterowania nie są jeszcze istotne. Jego rola polega na wystawieniu sterowania natychmiast po otrzymaniu uaktualnionej informacji o stanie obiektu. Oznacza to, że regulator jest wyidealizowany — nie wprowadza żadnych opóźnień decyzyjnych. *Punkty interwencji* pokrywają się z punktami pomiaru stanu. Takie uproszczone podejście pomoże w zrozumieniu funkcjonowania algorytmu repetycyjnego i w doborze jego podstawowych parametrów. W rozdziale ostatnim opisywany model zostanie uzupełniony o opóźnienia decyzyjne, których wpływ zostanie przebadany.

Aby powyższy schemat sterowania przystawał jeszcze bardziej do rzeczywistości, zostały wprowadzone *ograniczenia na sterowanie*. Mogą one odzwierciedlać *bariery rzeczywiste*, jak np. wydajność źródła energii. Mogą też, umiejętnie zastosowane, w sposób pośredni ograniczać *wartości zmiennych stanu* obiektu (ograniczenia implikowane). Jedynym bowiem sposobem na spełnienie ograniczeń na zmienne stanu jest stosowanie takiego sterowania, by mieć pewność, że nie zostaną one naruszone nawet

w przypadku skrajnie niekorzystnego działania natury reprezentowanej przez zakłócenia. Można podać w ogólności, że dla każdego $t_i \in T_s$ sterowanie $\boldsymbol{m}(t_i) \in \mathbf{M}(\boldsymbol{x}(t_i))$. Dalsze rozważania odnoszą się tylko do pierwszego aspektu ograniczeń, tj. ograniczeń rzeczywistych.

Celem sterowania jest minimalizacja *wskaźnika jakości* (łącznego kosztu sterowania obiektem), zdefiniowanego następująco:

$$J = \int_{t_0}^{t_f} \mathbf{f}_0(\boldsymbol{x}(t), \boldsymbol{m}(t), \boldsymbol{z}(t), t)\,\mathbf{d}\,t. \tag{2}$$

Należy zauważyć że problem był dotychczas zdefiniowany w sposób ogólny. Nie czyni się np. założenia o liniowości obiektu; nie określono też charakteru zakłóceń. A sposób ich modelowania ma olbrzymi wpływ na wybór metody sterowania. Odzwierciedlają one niepewność i dlatego dobrze byłoby zdefiniować je w postaci procesu Markowa bądź też układu dynamicznego pobudzanego szumem białym. Wówczas J stanie się zmienną losową. Najczęściej jednak takie opisanie zakłóceń jest bardzo kosztowne (lub czasochłonne), o ile w ogóle możliwe. Zdarza się, że posiadane informacje o zakłóceniach są bardzo skąpe i wówczas niepewność jest modelowana przez pojedynczą prognozę $\boldsymbol{z}^*(t)$ bądź ich pęk. Ale i to nie zawsze jest możliwe.

Wielce istotna jest także funkcja podcałkowa $\mathbf{f}_0$: w szczególnych przypadkach, w połączeniu z uproszczonym modelem obiektu ($\mathbf{f}$ — liniowa, bez ograniczeń), daje tzw. zadanie LQG. Znakomicie ułatwia to procedurę projektowania układu sterowania.

Rolą projektanta jest wybranie układu sterowania tak, by w pełni wykorzystać wszystkie dostępne a istotne informacje o obiekcie i zakłóceniach. Rozdział następny przedstawia wybrane podejścia do problemu.

2. Metody sterowania

Zadaniem układu sterowania jest znalezienie i zrealizowanie sterowania $\overline{\boldsymbol{m}} = \{\boldsymbol{m}(t_0), \boldsymbol{m}(t_1), \ldots\}$. Sposób (algorytm) wyznaczania sterowania w chwili t_i nazwijmy *regułą decyzyjną* μ_i. Argumentami (danym dla reguły decyzyjnej) mogą być wszystkie mierzalne wielkości występujące w systemie w chwili obecnej oraz w przeszłości. Mogą być nimi np. stan obiektu $\boldsymbol{x}$, *wyjścia* $\boldsymbol{y} = \mathbf{g}(\boldsymbol{x})$, zakłócenia (rozkłady bądź prognozy) — w chwili obecnej i w okresie od t_0 do t_{i-1}.

Reguły decyzyjne można podzielić na *stacjonarne* i *niestacjonarne*. Reguła (tj. odwzorowanie informacji o systemie w sterowanie) stacjonarna jest niezmienna dla wszystkich chwil t_i. Reguła niestacjonarna zmienia się w czasie. Niestacjonarność może wynikać z przesunięcia części obliczeń prowadzących do wyznaczenia sterowania z realizacji reguły (w trakcie sterowania) w samą jej definicję (określaną przed sesją sterowania). Pojawia się dylemat: czy lepiej przeznaczyć duże nakłady (czasu, obliczeń) na wypracowanie reguły decyzyjnej, której realizacja będzie łatwa (czyli szybka), czy też, rezygnując z długiego strojenia, zastosować regułę, w której większość obliczeń wykonuje się na bieżąco?

Wybór zależy od konkretnego problemu.

2.1 Regulator PID

W dyskretnym regulatorze PID reguła decyzyjna jest określona wzorem

$$m(t_i) = \mu\left(\boldsymbol{I}(t_i), \alpha\right), \quad \text{gdzie} \quad \boldsymbol{I}(t_i) = \left\{m(t_{i-1}), y(t_i), y(t_{i-1}), y(t_{i-2})\right\} \tag{3}$$

jest fragmentem historii wyjścia obiektu $\boldsymbol{y} = \mathbf{g}(\boldsymbol{x})$ i sterowania. Łatwo zauważyć, że reguła ta jest stacjonarna i sparametryzowana przez $\alpha = \left[k_r, T_p, T_w, T_z, \hat{y}\right]$. Reguła μ jest określona wzorem:

$$\mu\left(m(t_{i-1}), y(t_i), y(t_{i-1}), y(t_{i-2}), \begin{bmatrix} k_r \\ T_p \\ T_w \\ T_z \\ \hat{y} \end{bmatrix}\right) = m(t_{i-1}) + k_r\left(\varepsilon(t_i) - \varepsilon(t_{i-1})\right) + \tag{4}$$

$$+ k_r \frac{T_p}{T_z}\varepsilon(t_i) + k_r \frac{T_w}{T_p}\left(\varepsilon(t_i) - 2\varepsilon(t_{i-1}) + \varepsilon(t_{i-2})\right),$$

gdzie ε oznacza uchyb, $\varepsilon(t_i) = \hat{y} - y(t_i)$; $\hat{y}$ jest wartością zadaną wyjścia, T_p to odległość między kolejnymi punktami interwencji $T_p = t_i - t_{i-1}$ a k_r, T_w i T_z to odpowiednio współczynnik wzmocnienia, czas zdwojenia i czas wyprzedzenia z nastaw Zieglera-Nicholsa regulatora PID dla obiektu ciągłego.

Reguła decyzyjna dyskretnego regulatora PID jest więc prosta a wyznaczenia sterowania dokonuje się błyskawicznie. Prostota jest, niestety, okupiona koniecznością znalezienia parametrów k_r, T_w i T_z; więc cały wysiłek skonstruowania reguły musi być podjęty jeszcze przed rozpoczęciem sesji sterowania. Wadą PID jest to, że w trakcie pracy nie uwzględnia informacji o zakłóceniach (4). Nie oznacza to, bynajmniej, że nic o nich nie wie. Wszak sama procedura doboru nastaw Zieglera-Nicholsa dokonuje się w obecności zakłóceń i trójka parametrów k_r, T_w, T_z w pewnym stopniu je odzwierciedla. Kłopot w tym, że nie zawsze sygnał zakłócający to proces Wienera: są przypadki, w których zakłócenia mogą przez długi czas przyjmować wartości skrajne — wówczas prostota PID może zaszkodzić.

Regulator ten ma jeszcze inne wady. Kolejną jest założenie o liniowym modelu obiektu, co predestynuje PID do obiektów liniowych lub zlinearyzowanych. Możliwe jest podłączenie go do obiektu nieliniowego, ale raczej w układzie kompensacji zakłóceń, po linearyzacji. W zasadzie nie nadaje się on do przestawiania. Ograniczenie to omija się stosując regulatory obszarowe z przejściem między obszarami realizowanym przez logikę rozmytą. Jeszcze innym problemem jest to, że PID ma tylko jedną wartość wejściową: uchyb. Trudności pojawiają się także przy stosowaniu PID w regulacji wielowymiarowej. Wypracowano [4] rozwiązanie dla niektórych układów o dwóch wielkościach regulowanych: polega ono na takim przetworzeniu uchybu (odprzęgnięciu), by każdy odzwierciedlał tylko jedno zakłócenie — następnie stosuje się regulator PID diagonalny.

Podsumowując: regulatory PID są bardzo proste w działaniu, szybkie a ich nastrojenie — nieskomplikowane. W wersji podstawowej ich działanie jest dobrze określone tylko dla obiektów liniowych i zlinearyzowanych. Rozszerzenie dziedziny zastosowań uzyskuje się przez połączenie PID z innymi technikami regulacji.

W tej klasie regulatorów istnieją też bardziej złożone rozwiązania. Regulator PID jest sparametryzowany tylko przez trzy wielkości; regulatory w postaci sieci neuronowych mogą mieć ich setki i, niewątpliwie, działać o wiele lepiej. Niestety, uzyskanie optymalnych wartości tych parametrów musi być poprzedzone długotrwałym strojeniem, co wymaga czasu i znacznej wiedzy o zakłóceniach. A ta nie zawsze jest dostępna.

2.2 Programowanie dynamiczne — DP

Reguła decyzyjna wynikająca z programowania dynamicznego jest niestacjonarna. Sterowanie na pojedynczym etapie wyraża się wzorem:

$$m(t_i) = \mu\big(\boldsymbol{I}(t_i)\big), \quad \text{gdzie} \quad \boldsymbol{I}(t_i) = \boldsymbol{x}(t_i). \tag{5}$$

Zastosowane sterowanie zależy wyłącznie od stanu w chwili t_i, przy czym reguła decyzyjna może być odmienna dla każdego $t_i \in T_s$. Sterowanie według wyznaczonych wcześniej reguł jest wyznaczane bardzo szybko. Reguła decyzyjna przybiera w szczególnych przypadkach (zadanie LQG) formę macierzy $n \times r$ — w ogólności jest to r funkcji n zmiennych. Cały ciężar obliczeń przenosi się, podobnie jak poprzednio, w stronę wyznaczenia owej reguły dla każdego punktu interwencji. Dokonuje się tego przed początkiem sesji sterowania.

Programowanie dynamiczne wynika [3] z zasady optymalności Bellmana, w myśl której sterowanie optymalne układu (1) przy addytywnym wskaźniku jakości (określony wg (2) jest nim) w przedziale $< t_i, t_f)$ nie zależy od historii układu (czyli od $t < t_i$) i jest określone wyłącznie przez stan $\boldsymbol{x}(t_i)$ i wskaźnik jakości

$$J(t_i) = \int_{t_i}^{t_f} \mathbf{f}_0\big(\boldsymbol{x}(t), \boldsymbol{m}(t), \boldsymbol{z}(t), t\big)\,\mathbf{d}\,t. \tag{6}$$

Z zasady Bellmana wynika poniższa procedura wyznaczania sterowania optymalnego. Najpierw należy zdyskretyzować równanie stanu i wskaźnik jakości sterowania, co jest zgodne z założeniami poczynionymi w rozdz. 1. Daje to równanie stanu $\boldsymbol{x}_{k+1} = \mathbf{f}_k\big(\boldsymbol{x}_k, \boldsymbol{m}_k, \boldsymbol{z}_k\big)$ i wskaźnik jakości (koszt na etap) $J_k\big(\boldsymbol{x}_k, \boldsymbol{m}_k, \boldsymbol{z}_k\big)$. Przyjmijmy ponadto, że $\boldsymbol{z}(t)$ zmienia się tylko w chwilach T_s. Taka dyskretyzacja daje, dla t_f skończonego N etapów sterowania. Jeśli przez J_n^* oznaczymy koszt za etapy od n do $N-1$

$$J_n^* = \sum_{k=n}^{N-1} J_k\big(\boldsymbol{x}_k, \boldsymbol{m}_k, \boldsymbol{z}_k\big), \tag{7}$$

to przy znanych zakłóceniach $\{\boldsymbol{z}_n, \boldsymbol{z}_{n+1}, \ldots, \boldsymbol{z}_{N-1}\}$ można tak dobrać sterowanie $\{\boldsymbol{m}_n, \boldsymbol{m}_{n+1}, \ldots, \boldsymbol{m}_{N-1}\}$, by J_n^* był jak najmniejszy:

$$S_n^* = \min_{\{\boldsymbol{m}_n, \boldsymbol{m}_{n+1}, \ldots, \boldsymbol{m}_{N-1}\}} J_n^* \quad | \quad \{\boldsymbol{z}_n, \boldsymbol{z}_{n+1}, \ldots, \boldsymbol{z}_{N-1}\}. \tag{8}$$

Jeżeli $\boldsymbol{z}_k$ jest zmienną losową, wówczas J_k staje się również zmienną losową a J_n^* oznacza wartość oczekiwaną

$$J_n^* = \mathbf{E}\left(\sum_{k=n}^{N-1} J_k\left(\boldsymbol{x}_k, \boldsymbol{m}_k, \boldsymbol{z}_k \right) \right). \tag{9}$$

Zgodnie z zasadą Bellmana S_n^* zależy jedynie od $\boldsymbol{x}_n$. Można wyrazić S_n^* również tak:

$$S_n^* = \min_{\boldsymbol{m}_n} \left(J_n + S_{n+1}^* \right). \tag{10}$$

Z rekurencyjnej postaci wzoru (10) wynika następujący algorytm wyznaczania reguły decyzyjnej: na początku znajduje się S_{n-1}^* a następnie, znając S_{n-1}^* można wyznaczyć S_{n-2}^*, S_{n-3}^*, itd. — aż do S_0^*. Wówczas, mając regułę decyzyjną oraz stan początkowy obiektu, przystępuje się do sterowania.

Przewaga tego algorytmu nad regulacją PID polega na tym, że potrafi on w pełni skonsumować informacje o rozkładzie zakłóceń. Łatwo też zrealizować ograniczanie sterowania oraz niejawne ograniczenia stanu. To ostatnie wprowadza się przez zawężenie przedziału dopuszczalnych sterowań tak, by do chwili następnej interwencji stan obiektu nie naruszył ograniczeń, nawet dla najbardziej niesprzyjających zakłóceń.

Programowanie dynamiczne na też wady. Jedną z nich jest konieczność zdyskretyzowania stanu obiektu i rozkładu zakłóceń. Nie jest to istotne, jeżeli obiekt jest dyskretny z natury, ale dla obiektów nieliniowych ciągłych trafna dyskretyzacja stanu może przysporzyć problemów. Inną przykrą cechą DP jest tzw. *przekleństwo wymiarowości*, czyli wykładnicza zależność czasu obliczeń od liczby zmiennych stanu i sterowań. I to ono przesądza o przydatności algorytmu (który *notabene* świetnie nadaje się do zrównoleglenia). Ponadto DP w przedstawionej tutaj postaci nie ma zastosowania gdy $t_f = \infty$ — chyba, że przyjmie się dodatkowe założenia i wprowadzi *dyskonto* dla kolejnych etapów sterowania.

2.3 Sterowanie repetycyjne — BPC

Reguła decyzyjna w sterowaniu repetycyjnym jest stacjonarna, podobnie, jak dla PID. Różnica polega na tym że, gdy w regulatorze PID cały wysiłek uzyskania korzystnego wskaźnika jakości przejawia się w strojeniu regulatora *przed* sesją sterowania, to w sterowaniu repetycyjnym procedura wiodąca ku sterowaniu optymalnemu jest powtarzana *na bieżąco*, podczas sesji. Reguła decyzyjna jest opisana tak:

$$m(t_i) = \mu\left(\boldsymbol{I}(t_i) \right), \quad \text{gdzie} \quad \boldsymbol{I}(t_i) = \left[\boldsymbol{x}(t_i), \boldsymbol{z}_i^* \right] \tag{11}$$

a $\boldsymbol{z}_i^*$ jest prognozą przebiegu zakłóceń od t_i do $t_i + T_i^*$.

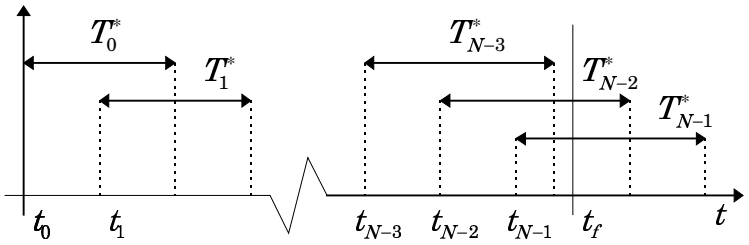
Idea sterowania repetycyjnego [1] polega na tym, by w chwili t_i, mając informacje o stanie obiektu i prognozę zakłóceń, wyznaczyć trajektorię sterowania $\boldsymbol{m}_i^*(t)$, które minimalizuje wskaźnik jakości

$$J_i^* = \int_{t_i}^{t_i+T_i^*} \mathbf{f}_0\left(\boldsymbol{x}_i^*(t), \boldsymbol{m}_i^*(t), \boldsymbol{z}_i^*\right) \mathbf{d}\, t + \mathbf{Q}\left(\boldsymbol{x}_i^*\left(t_i + T_i^*\right)\right). \tag{12}$$

Przy czym $\boldsymbol{m}_i^*(t)$ jest przebiegiem wyznaczonym dla lokalnego przedziału sterowania $< t_i, t_i + T_i^* >$; $\mathbf{f}_0$ i $\mathbf{Q}$ to wskaźniki jakości na lokalnym przedziale sterowania.

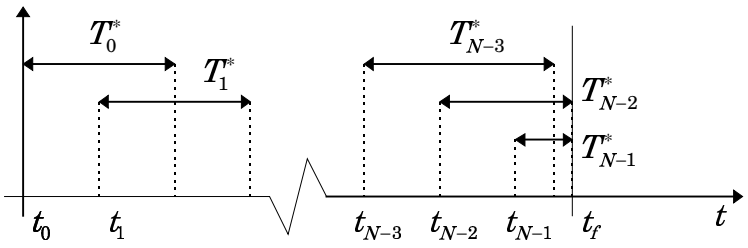
Po znalezieniu $\boldsymbol{m}_i^*(t)$ steruje się nim obiekt do chwili t_{i+1} i wówczas cała procedura znajdowania sterowania optymalnego $\boldsymbol{m}_{i+1}^*(t)$ dla lokalnego przedziału sterowania powtarza się. Należy zauważyć, że nie zostały poczynione dotychczas żadne założenia co do dyskretyzacji sterowania na lokalnym przedziale sterowania; odrzucanie części $\boldsymbol{m}_i^*$ leżącej poza t_{i+1} sugeruje, że można by stosować na tym odcinku zgrubną dyskretyzację.

Można też rozpatrzyć różne taktyki doboru długości lokalnego przedziału sterowania T_i^*. W wersji prostszej T_i^* nie zależy od i — więc od pewnej chwili horyzont lokalny znajduje się poza horyzontem globalnym (2).



Rys. 2 Horyzont lokalny przesuwany swobodnie

Takie postępowanie nie jest niczym uzasadnione, można więc rozważyć sytuację, gdy w miarę zbliżania się do horyzontu globalnego T_i^* ulega skracaniu tak, by horyzont lokalny nigdy nie wybiegał poza t_f (3).



Rys. 3 Horyzont lokalny ograniczany

Obie taktyki mają swoje zastosowania. Ta z przesuwanym horyzontem lokalnym nadaje się do zadań na horyzoncie nieskończonym. Druga zmniejsza nakład obliczeń i poprawia jakość sterowania przy zbliżaniu się do końca sesji.

Osobny problem stanowi wyznaczenie optymalnej sekwencji sterującej $\boldsymbol{m}_i^*$ na lokalnym przedziale sterowania. Zadanie jest w pełni deterministyczne, można więc potraktować wskaźnik jakości jako funkcję $N_i^* \cdot r$ zmiennych, gdzie N_i^* — liczba punktów dyskretyzacji sterowania na i-tym lokalnym przedziale sterowania. Do jej zminimalizowania można wykorzystać jedną z metod bezgradientowych, np. Powella, ale lepiej skorzystać z procedury wyznaczania gradientu zredukowanego do przestrzeni sterowań. Umożliwia ona znalezienie gradientu J_i^* po $\boldsymbol{m}_i^*$ i pozwala zastosować gradientowe algorytmy optymalizacji. O tym szerzej w następnym rozdziale.

Powyżej została przedstawiona idea sterowania repetycyjnego w swej pierwotnej wersji. Jest to punkt wyjścia do modyfikacji pozwalających, na przykład, na korzystanie z wielowariantowej prognozy zakłóceń. Wówczas algorytm znajdowania sterowania dla przedziału lokalnego nie działa już całkowicie w otwartej pętli sprzężenia zwrotnego: przy obliczaniu sterowania jest brane pod uwagę to, że informacja o stanie obiektu będzie dostępna w przyszłości. Doniosłość modyfikacji ujawnia się, gdy ewentualne trajektorie zakłóceń są dobrze określone, ale to, która się spełni, jest wynikiem losowania. Tylko wówczas dołącza problem znany z algorytmu DP: jak uzyskać prawdopodobieństwa owych trajektorii..

2.4 Wnioski

Każdy z powyższych algorytmów sterowania ma swoją specyfikę; wady i zalety. Można stwierdzić, że prostota reguł decyzyjnych jest zawsze okupiona rezygnacją z dodatkowych informacji (historii stanu, rozkładu lub prognozy zakłóceń) lub uproszczeniem modelu obiektu. Z drugiej strony, algorytmy zaawansowane wymagają wielu obliczeń (przed lub w trakcie sterowania) i dodatkowych danych. Jeśli tylko są dostępne, to opłaca się je wykorzystać.

Przypomnijmy sobie, jak został postawiony problem. Zakładamy, że sterowany obiekt może być nieliniowy: określenie to bardzo ogólne; nie można więc odrzucić przypadków *silnej* nieliniowości — a ta wyklucza zastosowanie regulatora PID. Jeśli dopowiemy jeszcze, że pewna wiedza o zakłóceniach jest dostępna, to pozostaje wybór między DP a BPC.

Nie jest możliwe stworzenie rankingu tych dwóch metod w oderwaniu od konkretnego problemu. Owszem, można powiedzieć, że DP jest bardziej ogólny, gdyż dla zakłóceń o rozkładzie punktowym da takie same wyniki, jak BPC na pojedynczym etapie. Ale rozkład może się zmienić już podczas realizacji sterowania. A tego DP nie uwzględnia. Z kolei BPC na bieżąco reaguje na zmianę prognozy zakłóceń, ale wyznacza sterowanie na horyzoncie lokalnym przy pełnym determinizmie, co przy błędnej prognozie może przynieść fatalne skutki. Rozpatrzmy przykład polegający na optymalnym ogrzewaniu osiedla mieszkaniowego od 15 do 20 stycznia. DP dostaje 15 I o północy rozkład temperatury od 15 do 20 stycznia i po 3–godzinnych obliczeniach produkuje regułę decyzyjną. Natomiast BPC

pobiera co 6 godzin coraz trafniejsze prognozy temperatury i wyznacza na bieżąco sterowanie optymalne. Jeśli prognozy te będą trafne to, przypuszczalnie, BPC da lepsze wyniki niż PD. Jeśli nie — przyniesie więcej szkód niż korzyści.

Idealnym rozwiązaniem byłoby cykliczne uruchamianie DP w momencie, gdy jest dostępny zaktualizowany rozkład temperatury a następnie weryfikacja reguły decyzyjnej. Często jednak uzyskanie rozkładu jest na tyle kosztowne a dynamika obiektu tak szybka, że taka idea staje się nierealna.

Od tej pory zakładam, że informacja o zakłóceniach w chwili t_i jest dostępna w postaci pojedynczej prognozy na okres $\langle t_i, t_i + T_i^* \rangle$. Będę więc sterował obiektem przy pomocy algorytmu BPC. O tym, jak wykorzystać BPC w praktyce — w następnym rozdziale.

3. Opis algorytmu

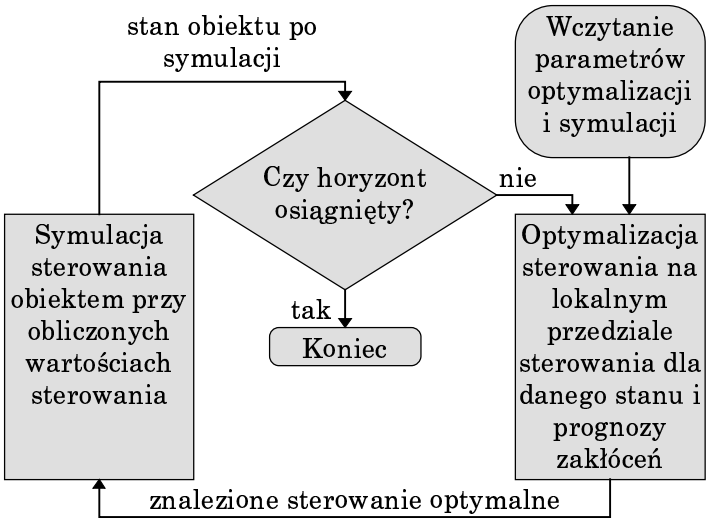
3.1 Organizacja symulacji sterowania

Teoria

Przyjmijmy dla wygody, że pomiar stanu obiektu i sterowanie następują w tych samych chwilach i ich rozmieszczenie na osi czasu jest równomierne. Takie podejście jest powszechnie stosowane w praktyce. Niech punkt t_f także spełnia to założenie, wobec czego możemy opisać zbiór T_s wzorem

$$T_s = \{t_0, t_0 + T, t_0 + 2T, \ldots, t_0 + NT\} \quad \text{przy czym} \quad t_f = t_0 + NT. \tag{13}$$

Nazwijmy T *długością etapu sterowania* a N *— liczbą etapów sterowania*. T_s ma $N+1$ elementów, ale t_f nie jest punktem interwencji. Z pojedynczym *etapem sterowania* wiązać się będzie zarówno proces decyzyjny (optymalizacja na przedziale lokalnym) prowadzący do wyznaczenia sterowania, jak i wystawienie go obiektowi w na czas od t_i do t_{i+1}. Pod-



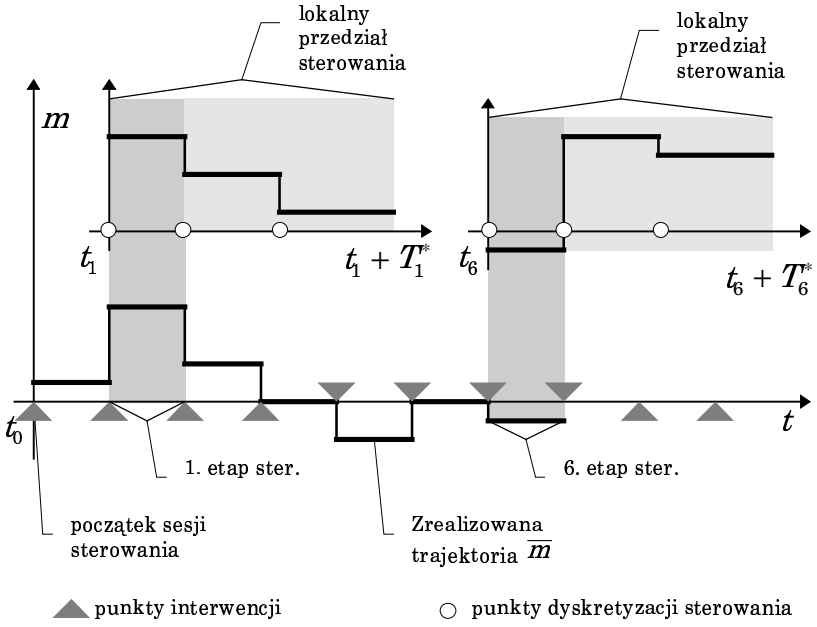
Rys. 4 Podstawowy algorytm symulacji

stawowy algorytm symulacji wygląda więc tak, jak na rysunku (4).

Realizacja

Ostatecznie zrezygnowałem ze stworzenia interfejsu graficznego dla użytkownika. Wszystkie parametry optymalizacji umieszcza się w plikach

konfiguracyjnych, które są następnie wczytywane i interpretowane. W programie t_f jest określone pośrednio poprzez liczbę etapów sterowania. Ponadto założyłem, że długość etapu sterowania jest określona odległością między dwoma pierwszymi punktami dyskretyzacji na lokalnym prze-



Rys. 5 Kolejne etapy sterowania repetycyjnego

dziale sterowania (5).

Nie ma powodu, by optymalizator sterowania generował trajektorię drobno zdyskretyzowaną, skoro i tak tylko jej wartość w chwili t_i zostanie przyłożona do obiektu i podtrzymana do t_{i+1}. Symulator dostaje więc od optymalizatora czas, do którego ma być wykonana symulacja ze znalezionym sterowaniem. Pojęcie *czas* nie ma na razie nic wspólnego z upływem czasu rzeczywistego i istnieje wyłącznie w świecie modeli matematycznych.

3.2 Organizacja na poziomie etapu sterowania

Teoria

Weźmy pod uwagę pojedynczy etap sterowania $< t_i, t_{i+1})$. Przy całkowym wskaźniku jakości dla globalnego przedziału sterowania, wskaźnik jakości za etap $< t_i, t_{i+1})$ wyraża się wzorem

$$J_i = \int_{t_i}^{t_{i+1}} \mathbf{f}_0\big(\boldsymbol{x}(t), \boldsymbol{m}(t), \boldsymbol{z}(t), t\big)\,\mathbf{d}\,t, \tag{14}$$

lecz biorąc pod uwagę ostatnie założenia o stałości sterowania na etapie i niezmiennym T, mamy

$$J_i = \int_{t_i}^{t_{i+T}} \mathrm{f}_0\left(\boldsymbol{x}(t), \boldsymbol{m}_i^*(t), \boldsymbol{z}(t), t\right) \mathrm{d}\, t, \tag{15}$$

gdzie $\boldsymbol{m}_i^*(t)$ oznacza część sterowania wyznaczonego na i-tym lokalnym przedziale.

Jak, w takim razie, znajduje to sterowanie optymalizator? Jak już wspomniałem, zna on równania stanu obiektu, jego stan w chwili t_i, równanie wskaźnika jakości i prognozę zakłócenia. (Wszystkie one mogą różnić się od *faktycznych*, zdefiniowanych dla modelu do symulacji, ale na razie przyjmijmy, że optymalizator ma rzetelne informacje o obiekcie.) Jako dodatkowy parametr dostaje on z procedury nadrzędnej (zawiadującej symulacją) długość lokalnego przedziału T_i^*, dla którego ma wyznaczyć sterowanie minimalizujące wskaźnik jakości (12).

Zazwyczaj nie można sobie pozwolić na tak duże T_i^*, by $t_i + T_i^*$ było równe t_f.Dlatego wprowadza się funkcję $\mathbf{Q}$ *kary za stan* w horyzoncie lokalnym. Symbolizuje ona wszystkie kłopoty piętrzące się dlatego, że optymalizacja nie został wykonana do t_f, lecz na krótszym dystansie. Dla $t_f = \infty$ wykonanie takiej optymalizacji jest przecież niemożliwe. Sterowanie $\boldsymbol{m}_i^*$, na które powoływałem się w (12) jest pewnym przebiegiem hipotetycznym. Hipotetycznym dlatego, że tylko jego część dotycząca $< t_i, t_{i+1})$ będzie zrealizowana na pewno; pozostała — prawie na pewno — nie. Osobne zagadnienie stanowi dobór punktów dyskretyzacji $\boldsymbol{m}_i^*$. Można wyobrazić sobie ich równomierne rozmieszczenie na T_i^*, w szczególności takie, by pokrywały się z punktami interwencji. Nie jest to jednak konieczne, bo i tak część $\boldsymbol{m}_i^*$ leżąca poza t_{i+1} nie zostanie zrealizowana.

Sformalizowane zadanie optymalizacji będzie brzmieć: dla i-tego etapu znaleźć

$$\mathbf{arg} \min_{\left(\boldsymbol{m}_{i,0}^*, \boldsymbol{m}_{i,1}^*, \ldots, \boldsymbol{m}_{i,N_i^*-1}^*\right)} J_i^*\left(\boldsymbol{x}_i^*(t), \left(\boldsymbol{m}_{i,0}^*, \boldsymbol{m}_{i,1}^*, \ldots, \boldsymbol{m}_{i,N_i^*-1}^*\right), \boldsymbol{z}_i^*, t\right). \tag{16}$$

(Przebieg $\boldsymbol{m}_i^*(t)$ wynika z uciąglenia sekwencji $\left(\boldsymbol{m}_{i,0}^*, \boldsymbol{m}_{i,1}^*, \ldots, \boldsymbol{m}_{i,N_i^*-1}^*\right)$ wyznaczonej w punktach dyskretyzacji $\left(t_{i,0}, t_{i,1}, \ldots, t_{i,N_i^*-1}\right)$ na i-tym przedziale lokalnym.) Jest to zadanie optymalizacji statycznej funkcji $r \cdot N_i^*$ zmiennych (wymiarowość wektora sterowań *razy* ilość punktów dyskretyzacji).

Aby rozwiązać (16), zdecydowałem się użyć metody gradientowej BFGS. Zanim ją omówię, w następnym punkcie przedstawię metodę obliczania wartości funkcji i jej gradientu.

Realizacja

Zadawanie dyskretyzacji na horyzoncie lokalnym odbywa się w sposób uproszczony. Użytkownik definiuje zbiór punktów, w których miałaby nastąpić interwencja przez podanie kolejno ich odległości od t_i. Rzecz jasna, pierwszą wartością będzie zawsze 0, gdyż najistotniejsze jest sterowanie dla t_i. Następna określa (p. uproszczenie w *Realizacji* podrozdziału 3.1) odległość T między (globalnymi) punktami interwencji (czyli długość etapu). Te dwie odległości oraz kilka następnych definiują *szablon* dyskretyzacji sterowania, jednakowej dla wszystkich przedziałów lokalnych. Aplikowany do każdego t_i daje aktualny *zbiór punktów dyskretyzacji* sterowania $\{t_{i,j}\}$. W przypadku ze skracanym T_i^* z $\{t_{i,j}\}$ eliminowane są elementy, które wypadłyby poza t_f.

Każdy z obu modułów (tj. symulatora i optymalizatora) posiada swój własny zestaw parametrów. Pozwala to w łatwy sposób zasymulować sterowanie przy np. niepełnej wiedzy o obiekcie (różnice w równaniach stanu) czy zakłóceniach (różnice między $\boldsymbol{z}(t)$ a kolejnymi $\boldsymbol{z}_i^*(t)$).

3.3 Uzyskiwanie wskaźnika jakości i jego gradientu

Teoria

Jako że dalsze rozważania będą dotyczyły wyłącznie optymalizacji sterowania na horyzoncie lokalnym, można dla czytelności opuścić indeks i i gwiazdkę $*$ przy oznaczeniach $\boldsymbol{\mu}$, $\boldsymbol{x}$ i $\boldsymbol{z}$. Sterowanie $\boldsymbol{\mu}(t)$ będzie traktowane jako przebieg ciągły; $\boldsymbol{\mu}$ bez zależności od t — jako ciąg $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{N^*-1})$. Procedury obliczające wskaźnik jakości i jego gradient są wywoływane na użytek modułu optymalizującego, dlatego w rozważaniach w tym punkcie $\boldsymbol{\mu}$ jest traktowane jako parametr wywołania, słowem — dana.

Obliczenie wskaźnika jakości polega, po pierwsze, na scałkowaniu równania stanu obiektu, by znaleźć trajektorie stanu $\boldsymbol{x}(t)$ na $\langle t_0, t_0 + T^* \rangle$. Następnie, mając te trajektorie, można wyznaczyć wskaźnik jakości J^*: całkując $\mathbf{f}_0$ i uzupełniając wynik o karę za stan końcowy $\mathbf{Q}$ (wg (12)). Oczywiście, pozostaje problem, jakiej procedury użyć do całkowań: o tym później.

Wyznaczenie gradientu $\partial J^* / \partial \boldsymbol{\mu}$ nie jest tak proste. Aby tego dokonać, należy tymczasowo założyć, że sterowanie jest ciągłe (tzn. niekoniecznie przedziałami stałe) i sięgnąć do procedury wyznaczania gradientu wskaźnika jakości zredukowanego do przestrzeni sterowań.

Wygodnie jest na początku sprowadzić definicję problemu, tj. równania stanu i wskaźnika jakości do postaci standardowej. Można bez uszczerbku dla poprawności wprowadzić równanie stanu

$$\dot{\boldsymbol{X}} = \mathbf{F}(\boldsymbol{X}, \boldsymbol{U}) \quad \text{gdzie} \quad \mathbf{F}(\boldsymbol{X}, \boldsymbol{U}) = \begin{bmatrix} \mathrm{f}_0(\boldsymbol{X}, \boldsymbol{U}) \\ \mathbf{f}(\boldsymbol{X}, \boldsymbol{U}) \end{bmatrix} \quad \text{oraz} \tag{17}$$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{m}(t) \\ \boldsymbol{z} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_0 \\ \boldsymbol{x} \end{bmatrix},$$

przy warunkach początkowych $\boldsymbol{X}(t_0) = \begin{bmatrix} 0 \\ \boldsymbol{x}(t_0) \end{bmatrix}$.

Składowa całkowa wskaźnika jakości została potraktowana jako jeszcze jedna zmienna stanu x_0 z warunkiem początkowym 0 a zakłócenia i sterowania stanowią wektor $\boldsymbol{U}$ wartości wejściowych obiektu. Taki zapis pozwala zdefiniować J^* na $\langle t_0, t_0 + T^* \rangle$ jako prostą funkcję nowego rozszerzonego stanu $\boldsymbol{X}$ w chwili $t_0 + T^*$:

$$J^*(\boldsymbol{U}) = \varphi(\boldsymbol{X}(t)) = x_0(t_0 + T^*) + \mathbf{Q}(\boldsymbol{x}(t_0 + T^*)). \tag{18}$$

Pozbyliśmy się w ten sposób kłopotliwej całki występującej poprzednio w sposób jawny. Następnym etapem jest zdefiniowanie funkcji zwanej hamiltonianem:

$$\mathbf{H}(\boldsymbol{X}, \Lambda, \boldsymbol{U}, t) \overset{\mathbf{df}}{=} \left(\langle \Lambda(t), \mathbf{F}(\boldsymbol{X}, \boldsymbol{U}, t) \rangle\right) = \sum_{k=0}^{p} \lambda_k(t) \mathbf{f}_k(\boldsymbol{x}, \boldsymbol{U}, t) = \tag{19}$$

$$= \lambda_0 \mathbf{f}_0(\boldsymbol{x}, \boldsymbol{U}, t) + \sum_{k=1}^{p} \lambda_k(t) \mathrm{f}_k(\boldsymbol{x}, \boldsymbol{U}, t)$$

gdzie $\Lambda(t)$ jest pewnym przebiegiem, rozwiązaniem równania różniczkowego sprzężonego. Jeśli sterowanie $\boldsymbol{U}$ jest optymalne (tj. minimalizuje wskaźnik jakości), to hamiltonian osiąga maksimum dla każdego t. Zatem jeżeli uda się nam zmaksymalizować $\mathbf{H}$ to tym samym znaleźliśmy sterowanie optymalne. Trzeba jeszcze sprecyzować Λ. Otóż

$$\dot{\Lambda} = -\frac{\partial \mathbf{H}}{\partial \boldsymbol{X}^{\mathbf{T}}}. \tag{20}$$

Korzystając z (17) mamy ($\Lambda = \begin{bmatrix} \lambda_0 \\ \lambda \end{bmatrix}^{\mathbf{T}}$):

$$\dot{\lambda}_0 = -\frac{\partial \mathbf{H}}{\partial x_0} = 0, \tag{21}$$

$$\dot{\lambda} = -\frac{\partial \mathbf{H}}{\partial \boldsymbol{x}} = -\lambda_0 \frac{\partial \mathbf{f}_0}{\partial \boldsymbol{x}} - \lambda \frac{\partial \mathbf{f}}{\partial \boldsymbol{x}},$$

przy warunkach brzegowych

$$\lambda_0(t_0 + T^*) = -\frac{\partial \varphi\left(\boldsymbol{X}(t_0 + T^*)\right)}{\partial x_0} = -1, \tag{22}$$

$$\lambda\,(t_0 + T^*) = -\frac{\partial \varphi\left(\boldsymbol{X}(t_0 + T^*)\right)}{\partial \boldsymbol{x}} = -\frac{\partial \mathbf{Q}}{\partial \boldsymbol{x}}(t_0 + T^*).$$

Z (22) wynika, że $\lambda_0(t) = \mathbf{const} = -1$, zatem hamiltonian upraszcza się do:

$$\mathbf{H} = -\mathrm{f}_0(\boldsymbol{x}, \boldsymbol{U}) + \langle \lambda \mathbf{f}(\boldsymbol{x}, \boldsymbol{U}) \rangle \tag{23}$$

a część równania sprzężonego do:

$$\dot{\lambda} = \frac{\partial \mathbf{f}_0}{\partial \boldsymbol{U}} - \lambda \frac{\partial \mathbf{f}}{\partial \boldsymbol{U}} \quad \text{przy czym} \quad \lambda(t_0 + T^*) = -\frac{\partial \mathbf{Q}}{\partial \boldsymbol{x}}(t_0 + T^*). \tag{24}$$

Ponadto:

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{U}} = -\frac{\partial \mathbf{f}_0}{\partial \boldsymbol{U}} + \lambda \frac{\partial \mathbf{f}}{\partial \boldsymbol{U}}. \tag{25}$$

Z wyłożonej teorii wyłania się następujący algorytm znajdowania gradientu: przy pewnym początkowym $\boldsymbol{U}_0$ rozwiązujemy (całkujemy)

równanie (17), by otrzymać przebieg $\boldsymbol{X}(t)$, w szczególności wartość $\boldsymbol{X}(t_0 + T^*)$. Mając $\boldsymbol{X}(t_0 + T^*)$ wyznaczamy $\Lambda(t_0 + T^*)$ i całkujemy równanie (20) w kierunku malejącego czasu. Dysponując oboma przebiegami, możemy określić $\partial \mathbf{H} / \partial U$ w każdej chwili czasu i korygując sterowanie proporcjonalnie do $-\partial \mathbf{H} / \partial \boldsymbol{m}(t)$ ulepszać wskaźnik jakości, aż $\forall t$ $\partial \mathbf{H} / \partial \boldsymbol{U} = 0$.

W praktyce stosuje się wszakże sterowanie zdyskretyzowane. Dlatego wzór (25) w wersji dyskretnej przyjmie postać:

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{U}_j} = \int_{t_j}^{t_{j+1}} -\frac{\partial \mathbf{f}_0}{\partial \boldsymbol{U}} + \lambda \frac{\partial \mathbf{f}}{\partial \boldsymbol{U}} \mathbf{d}\, t, \tag{26}$$

ale skoro optymalizator może oddziaływać na J^* tylko przez $\boldsymbol{m}$, interesujące staje się tylko

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{m}_j} = \int_{t_j}^{t_{j+1}} -\frac{\partial \mathbf{f}_0}{\partial \boldsymbol{m}} + \lambda \frac{\partial \mathbf{f}}{\partial \boldsymbol{m}} \mathbf{d}\, t. \tag{27}$$

Wyznaczenie gradientu J^* w pojedynczym kroku optymalizacji będzie więc przebiegać tak:

- przy danym $\boldsymbol{m}$ następuje scałkowanie równań stanu do $t_0 + J^*$ i zapamiętanie rozwiązania;
- na podstawie $\boldsymbol{X}(t_0 + T^*)$ dokonuje się obliczenia $\Lambda(t_0 + T^*)$;
- mając $\Lambda(t_0 + T^*)$ całkuje się równanie sprzężone w kierunku malejącego czasu
- dysponując oboma rozwiązaniami, tj. $\boldsymbol{X}(t)$ oraz $\Lambda(t)$, oblicza się pochodne hamiltonianu względem zdyskretyzowanych sterowań przykładanych w punktach t_j, dokonując całkowania zgodnie z (27).

Realizacja

Całkowanie tak równania stanu, równania sprzężonego, jak i (27) jest wykonywane w tym samym module. Krok całkowania jest dobierany automatycznie tak, by zapewnić zadaną dokładność.

Można spodziewać się, że wyznaczenie macierzy jakobianowych funkcji $\mathbf{f}$ po $\boldsymbol{x}$ bądź $\boldsymbol{m}$ byłoby dla użytkownika żmudne i kłopotliwe. Dlatego odbywa się ono automatycznie. Co prawda, zaimplementowany algorytm różniczkowania symbolicznego nie sprosta wyrażeniom typu $x^{\sin x}$, ale większość wyrażeń jest przetwarzana i upraszczana poprawnie.

3.4 Algorytm optymalizacji

Teoria

Algorytm optymalizacji ma za cel znalezienie argumentu minimalizującego wskaźnik jakości na lokalnym przedziale sterowania (16). Dokonując wyboru z całej grupy dostępnych obecnie algorytmów, trzeba kierować się następującymi wytycznymi:

- algorytm powinien być skuteczny i niezawodny dla różnych rodzajów minimalizowanych funkcji;
- algorytm powinien być wystarczająco szybki — inaczej sterowanie obiektem w czasie rzeczywistym będzie niemożliwe;
- algorytm powinien być dostosowany do specyfiki zadania, tzn. uwzględniać ograniczenia.

Ponadto:

- algorytm powinien wykorzystywać jak najpełniej wszelkie dodatkowe, a dostępne, informacje o problemie; zwłaszcza te o gradiencie funkcji celu.

Warto zauważyć, że trzy pierwsze kryteria są priorytetowe. To znaczy, że przede wszystkim one powinny decydować o wyborze algorytmu. Jeżeli, na przykład, znajdziemy algorytm który, co prawda, nie wykorzystuje informacji o gradiencie, ale liczy stosownie szybko i dokładnie, to powinniśmy go zastosować. Nieracjonalnym byłoby implementowanie kilku algorytmów-kandydatów i sprawdzanie ich efektywności w praktyce. Zdecydowałem się wykorzystać algorytm *BFGS*. W formie uzasadnienia wyboru przedstawiam poniżej jego porównanie z dwoma innymi, równie popularnymi: *poszukiwań sympleksowych* i *Powella*.

Pierwsza, działając w przestrzeni L-wymiarowej, polega na początkowym wyznaczeniu *sympleksu*, czyli zbioru $L+1$ niewspółliniowych punktów a następnie przekształcaniu go tak, by cały znalazł się dostatecznie blisko rozwiązania zadania. Możliwe transformacje to odbicie, ekspansja i kontrakcje. Niestety, metoda ta, choć prosta i niezawodna, nie nadaje się do naszego problemu. Nie wymagając gradientu, dokonuje bardzo wielu oszacowań funkcji celu. Zważywszy, że każde z nich oznacza dwa całkowania, wydaje się, że poszukiwanie sympleksowe (szybkie dla zadań prostych) tutaj działałoby zbyt wolno.

Druga z nich działa minimalizując funkcję celu wzdłuż tzw. kierunków sprzężonych Powella. Zakłada się istnienie zbioru *kierunków sprzężonych*, tj. takich, że minimalizacja wzdłuż jednego z nich nie „zepsuje" wyniku poprzedniej minimalizacji wzdłuż innego. W ten sposób, dokonując L minimalizacji po wszystkich kierunkach ze zbioru, osiąga się minimum. Dla funkcji w postaci formy kwadratowej zostanie ono znalezione po $L\cdot(L+1)$ poszukiwaniach w kierunku. Co L poszukiwań jeden z kierunków w zbiorze jest modyfikowany. Schemat poszukiwań wg Powella może stanowić przyczynek do dyskusji, która z metod: Powella czy BFGS jest tutaj stosowniejsza.

Algorytm BFGS (Broyden-Fletcher-Goldfarb-Shanno) wykorzystuje informacje o gradiencie $\mathbf{g}(\boldsymbol{x})$ minimalizowanej funkcji $\mathbf{f}(\boldsymbol{x})$. Działa w ten sposób, że w kroku i dokonuje się minimalizacji $\mathbf{f}$ w kierunku $\boldsymbol{p}_i$. Kierunek poszukiwań powstaje z $\mathbf{g}$ i aproksymacji odwrotności hesjanu $\mathbf{f}$: macierzy $\boldsymbol{H}_i$. Obliczenia przebiegają następująco:

Krok 0
- inicjacja algorytmu: $\boldsymbol{H}_0 = \boldsymbol{E}$ (macierz jednostkowa), $\boldsymbol{p}_0 = -\mathbf{g}(\boldsymbol{x}_0)$.

Krok i
- minimalizacja w kierunku $\boldsymbol{p}_i$; jej wynikiem jest $\boldsymbol{x}_{i+1}$;
- testy stopu ze względu na $\mathbf{g}(\boldsymbol{x}_{i+1})$ oraz $\boldsymbol{x}_{i+1} - \boldsymbol{x}_i$;
- uaktualnienie macierzy $\boldsymbol{H}$:

$$\boldsymbol{H}_{i+1} = \boldsymbol{H}_i + \frac{\delta^T \boldsymbol{x} \cdot \delta \boldsymbol{x}}{\delta^T \boldsymbol{x} \cdot \delta \boldsymbol{g}} - \frac{(\boldsymbol{H}_i \cdot \delta \boldsymbol{g})^T \cdot (\boldsymbol{H}_i \cdot \delta \boldsymbol{g})}{\delta \boldsymbol{g} \cdot \boldsymbol{H}_i \cdot \delta \boldsymbol{g}} + (\delta \boldsymbol{g} \cdot \boldsymbol{H}_i \cdot \delta \boldsymbol{g}) \cdot (\boldsymbol{u}^T \cdot \boldsymbol{u}), \qquad (28)$$

$$\delta \boldsymbol{x} = \boldsymbol{x}_{i+1} - \boldsymbol{x}_{i+1} \quad \text{oraz} \quad \boldsymbol{u} = \frac{\delta \boldsymbol{x}}{\delta \boldsymbol{x} \cdot \delta g} - \frac{\boldsymbol{H}_i \cdot \delta g}{\delta g \cdot \boldsymbol{H}_i \cdot \delta g}.$$

$$\delta \boldsymbol{g} = \mathbf{g}(\boldsymbol{x}_{i+1}) - \mathbf{g}(\boldsymbol{x}_{i+1})$$

- wygenerowanie nowego kierunku $\boldsymbol{p}_{i+1} = -\boldsymbol{H}_{i+1} \boldsymbol{g}_{i+1}$.

Algorytm minimalizacji w kierunku korzysta z wartości funkcji i jej gradientu obliczonych dla $\boldsymbol{x}_i$. Z początku podejmowany jest pełen krok newtonowski, tj. $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{p}_i$, ale może on nie dać żądanego rezultatu, gdyż w $(\boldsymbol{x}_i + \boldsymbol{p}_i)$ kwadratowa aproksymacja $\mathbf{f}$ może już być nieważna. Dlatego w kolejnych próbach minimalizacji w kierunku $\mathbf{f}$ badana jest w punktach $(\boldsymbol{x}_i + \alpha_j \boldsymbol{p}_i)$, gdzie $\alpha_j \in (0,1)$. Sprawdzanie kolejnych $(\boldsymbol{x}_i + \alpha_j \boldsymbol{p}_i)$ polega na aproksymacji $\mathbf{f}$ w kierunku wielomianem trzeciego stopnia, połączonego z redukcją α.

Oba ostatnie algorytmy cechuje zbieżność kwadratowa. Jeśli założymy, że liczba kroków potrzebnych do znalezienia minimum $\mathbf{f}$ jest taka sama i równa D, to

- dla algorytmu Powella zostanie dokonanych około D^2 minimalizacji w kierunku;
- dla algorytmu BFGS zostanie dokonanych D minimalizacji w kierunku i D oszacowań gradientu.

Można zastanawiać się, która z metod pochłonie więcej czasu. Na pewno policzenie gradientu nie jest proste: składają się nań dwie symulacje ($\boldsymbol{X}$ i Λ) oraz N^* całkowań według (27); $L = N^* \cdot r$. Z kolei obliczenie $\mathbf{f}$ to tylko jedna symulacja i jedno całkowanie. Sądzę, że przy gęstej dyskretyzacji sterowania (N^* — duże) nakład obliczeń na wyznaczenie gradientu będzie mniejszy niż L-krotna minimalizacja w kierunku (Powell). Poza tym, minimalizacja w kierunku z wykorzystaniem gradientu (BFGS) z miejsca ma przewagę nad bezgradientową: Brenta czy złotego podziału (Powell).

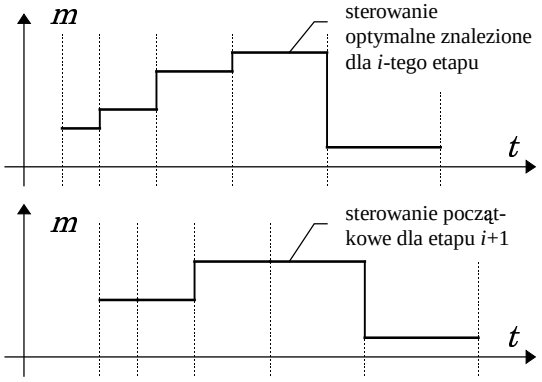
Do optymalizacji sterowania użyłem więc algorytmu BFGS w wersji z [2] z małymi modyfikacjami.

Realizacja

Najważniejszym uzupełnieniem, jakie trzeba było wprowadzić do standardowego algorytmu, było respektowanie ograniczeń kostkowych na sterowanie oznaczane w tym podrozdziale przez $\boldsymbol{x}$. Są one zadawane w postaci dwóch wektorów: ograniczeń dolnych i górnych. To oznacza, że przed uruchomieniem algorytmu położenie punktu początkowego $\boldsymbol{x}_0$ jest sprawdzane i, ewentualnie, korygowane.

Wyegzekwowanie ograniczeń na $\boldsymbol{x}_{i+1}$ następuje po minimalizacji w kierunku. Jeśli nowo znaleziony $\boldsymbol{x}_{i+1}$ jest poza ograniczeniami, to zostaje przesunięty wzdłuż $\boldsymbol{p}_i$ w stronę $\boldsymbol{x}_i$ tak, by spełniał wszystkie ograniczenia. Tymczasowo oznaczmy skorygowane $\boldsymbol{x}_{i+1}$ przez $\boldsymbol{x}^+_{i+1}$. Składowe $\boldsymbol{g}\left(\boldsymbol{x}^+_{i+1}\right)$, które „wypychają" $\boldsymbol{x}^+_{i+1}$ poza aktywne ograniczenia są zerowane — zatem wymiar zadania optymalizacji *de facto* się zmniejsza. Podstawienie $\boldsymbol{x}_{i+1} := \boldsymbol{x}^+_{i+1}$ kończy modyfikacje.

Dodatkowo, każdorazowe powiększanie się zbioru aktywnych ograniczeń powoduje *reset*, tj. $\boldsymbol{H}_{i+1} := \boldsymbol{E}$ oraz $\boldsymbol{p}_{i=1} := -\boldsymbol{g}\left(\boldsymbol{x}^+_{i+1}\right)$ (po wyzerowaniu zbędnych składowych w $\boldsymbol{g}(\boldsymbol{x}_i)$). Koniecznym okazało się również dodatkowe wykonywanie testu stopu przed minimalizacją w kierunku. Nie działała ona poprawnie, gdy $\boldsymbol{g}\left(\boldsymbol{x}_i\right)$ był zbyt mały.



Rys. 6 Dobór punktu startowego optymalizacji

Kwestia doboru punktu startowego optymalizacji (6) została rozstrzygnięta następująco: znaleziony na etapie i-tym optymalny przebieg sterujący (wykres górny) zostaje zapamiętany. W kroku i+1 sterowania początkowe (przed optymalizacją) zostają zainicjowane wartościami z przebiegu zapamiętanego, jakie przypadają w nowych punktach dyskretyzacji sterowania (wykres dolny). Takie postępowanie jest skutkiem, naiwnego skądinąd, założenia, że sterowanie optymalne na etapie następnym będzie takie samo, jak na poprzednim. Rzadko tak się zdarza, niemniej przyjęto powyższe rozwiązanie z braku rozsądnej alternatywy.

3.5 Całkowanie

Teoria

Wybierając metodę całkowania należy mieć na uwadze, by była ona niezawodna, szybka i wszechstronna, gdyż będzie uruchamiana wielo-

krotnie i dla rozmaitych zadań. Dlatego zdecydowano się na metodę Runge-Kutta. Jest wszechstronna i efektywna, a to istotne przy częstym całkowaniu podczas optymalizacji czy symulacji. Niestety, nie daje takiej dokładności, jak konkurencyjna metoda ekstrapolacji Richardsona czy metody typu predyktor-korektor. Na szczęście, nie ma to znaczenia. Wszak tylko część wyników jest wykorzystywana , a i te są stale zniekształcane przez zakłócenia.

Zdefiniujmy zadanie: znaleźć rozwiązanie równania różniczkowego

$$\frac{\mathbf{d}\,\boldsymbol{y}}{\mathbf{d}\,x} = \mathbf{f}(x, \boldsymbol{y}), \quad \boldsymbol{y}(x_0) = \boldsymbol{y}_0 . \tag{29}$$

Punktem wyjścia metody Runge-Kutta w pojedynczym kroku całkowania jest krok Eulera: $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\mathbf{f}(x_n, \boldsymbol{y}_n)$, gdzie h —długość kroku. Gdyby poprzestać na tak prymitywnym schemacie, to błędy w oszacowaniu $\boldsymbol{y}_{n+1}$ byłyby rzędu $\mathbf{O}(h^3)$ (z rozwinięcia Taylora). Dlatego metoda wykonuje pośrednich, mniejszych kroków Eulera a ich rezultaty służą do lepszego oszacowania $\boldsymbol{y}_{n+1}$. Szczególnie rozpowszechniona jest metoda Runge-Kutta 4-go rzędu która, przy zaledwie czterokrotnym oszacowaniu $\mathbf{f}$, pozwala zmniejszyć błąd do $\mathbf{O}(h^5)$. Niestety, brak informacji o samym błędzie nie pozwala na modyfikacje kroku całkowania h. Adaptacja h tak, by utrzymać stały, niewielki błąd, jest konieczna: pozwala znakomicie zmniejszyć ilość kroków na *monotonnej* części rozwiązania jednocześnie zachowując dokładność dla fragmentów *burzliwych*. Dlatego została zastosowana zmodyfikowana wersja Runge-Kutta umożliwiająca szacowanie błędu przy niewiele większym ($\times 1{,}375$) nakładzie obliczeń.

Adaptacja kroku całkowania polega na jego skracaniu bądź wydłużaniu tak, by oszacowany błąd pozostawał na wyznaczonym poziomie.

Realizacja

Całkowanie jest realizowane w uniwersalnym module całkującym. Służy on zarówno do całkowania równań stanu, równań sprzężonych, jak i wskaźnika jakości. Standardowe procedury z [2] zostały uzupełnione algorytmem wymuszającym pewną minimalną wartość h_i, gdy jest ono zbyt małe. Zapobiega to „dreptaniu" całkowania w miejscach, gdzie np. sterowanie zmienia się skokowo.

Powyżej zostały przedstawione podstawowe składniki algorytmu sterowania repetycyjnego. Ilość parametrów decydujących o jego działaniu jest znaczna: od najprostszych, dotyczących całkowania, przez optymalizację, aż po dyskretyzację sterowania. Następny rozdział jest próbą usystematyzowania tych parametrów i opracowania procedury doboru ich wartości.

4. Dobór parametrów

Poprawne, a w następnej kolejności szybkie, działanie regulatora repetycyjnego zależy od wielu wzmiankowanych parametrów. Stawiając sprawę jasno: nie można *stroić* algorytmu na najlepszy wskaźnik jakości sterowania manipulując nimi wszystkimi naraz czy też cyklicznie, po kolei. Taka strategia (Gaussa-Seidla) musi zawieść. Dlatego, że parametry nie mogą być traktowane równorzędnie. Przypomina to sytuację, gdy porównuje się wyniki minimalizacji pewnej funkcji prowadzonej w komputerze z *miernym emulatorem koprocesora* i wyniki minimalizacji tej samej funkcji w *innej maszynie wyposażonej w doskonały koprocesor numeryczny*. Zmiana dokładności podstawowych procedur numerycznych całkowicie uniemożliwia jakiekolwiek porównanie zastosowanych metod optymalizacji. W naszym przypadku rolę tej funkcji pełni wskaźnik jakości sterowania (uśredniony), a rolę precyzji obliczeń — dokładność całkowania. Dlatego należy przygotować procedurę doboru parametrów: poczynając od fundamentalnych, kończąc na ogólnych.

4.1 Dobór parametrów całkowania

Numeryczne znajdowanie rozwiązań równań różniczkowych zwyczajnych jest w mojej pracy zadaniem podstawowym. Stanowi bazę do obliczania wskaźników jakości, do symulowania zachowania obiektu, do wyznaczania $\lambda(t)$. Często interesuje nas jedynie rozwiązanie równania dla chwili końcowej: wówczas to, co się dzieje wewnątrz procedury jest nieistotne. Jednak bywają przypadki, gdy pożądana jest cała scałkowana trajektoria. Zastosowałem tylko jedną procedurę całkowania: ze zmiennym krokiem. To niewygodne w przypadkach, gdy potrzebna jest cała trajektoria. Żeby pogodzić efektywność i uniwersalność, dokonuję interpolacji trajektorii za pomocą linii łamanej i przechowuję ją w module całkującym.

Użytkownik może zmieniać wszystkie parametry algorytmu całkującego, ale nie jest to konieczne, bo większość jest już dobrze dobrana i odpowiednia dla szerokiej klasy zadań. Pozostają dwa istotne, które należy dopasować do konkretnego problemu. To *dokładność całkowania* ε i *przyrost zapisywany* δx. Dokładność ε wpływa na rzetelność rozwiązania: im ε mniejsze, tym rozwiązanie rzetelniejsze — ale całkowanie wolniejsze. Przyrost δx wpływa na dokładność zapisania tego rozwiązania: δx to minimalna odległość dwóch sąsiednich punktów trajektorii, przy której nowy punkt zostanie jeszcze zarejestrowany — ale zbyt małe δx to więcej zużytej pamięci i wolniejsze działanie.

Zaprezentuję podejście do problemu doboru parametrów na przykładzie dwóch układów dynamicznych.

Model wg Lotka-Volterra

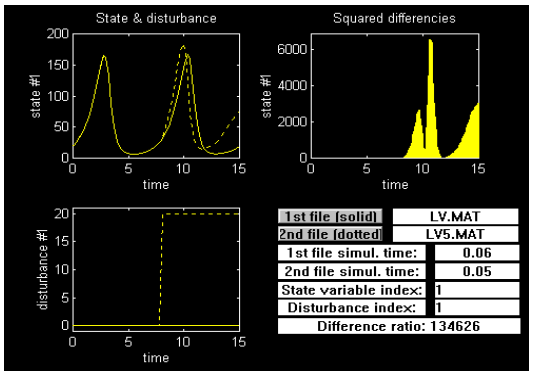
$$\dot{x}_1 = (1 - \alpha x_2) x_1 + z$$
$$\dot{x}_2 = (-1 + \beta x_1) x_2$$

gdzie $\alpha = 0{,}01$, $\beta = 0{,}02$.

Model obiektu z inercją

$\dot{x} = -ax + m + z$,
gdzie $a = 4$.

Pierwszym zadaniem jest oszacowanie dynamiki obiektu — przynajmniej rzędu wielkości. Dla modelu inercji jest ona dobrze określana przez stałą czasową (najmniejszą stałą dla układów wielowymiarowych); dla obiektów oscylacyjnych może być to okres drgań swobodnych. W tym celu stworzyłem mały program dokonujący całkowania oraz skrypt w *Matlabie* do wizualizacji rezultatów (7). Pozwala on na porównywanie wyników dwóch całkowań.



Rys. 7 Dobór parametrów całkowania

Następnym krokiem będzie sprawdzenie, jak zmieniają się uzyskiwane wykresy przy manipulacjach ε. Pierwotnie będę zwiększał dokładność o rząd wielkości, by wyznaczyć kres stabilności numerycznej algorytmu (i oszacować czasy obliczeń). Następnie, zmniejszając dokładność sprawdzę, kiedy algorytm traci stabilność. Zacznę od ustawień standardowych ($\varepsilon = 10^{-5}$, $\delta x = 10^{-2}$). Całość eksperymentu dokona się w obecności skoku jednostkowego na wejściu z, by symulować faktyczny charakter sterowania.

Model wg Lotka-Volterra

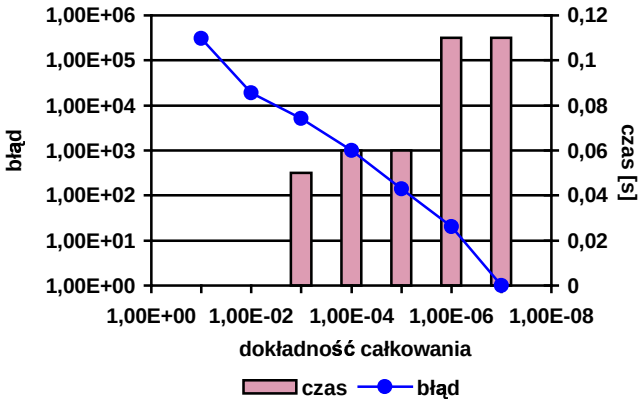
Zmienne stanu mają przebieg oscylacyjny o okresie około $7s$. Niech ta liczba opisuje dynamikę układu. Zwiększanie dokładności nie powoduje zmian w kształcie przebiegu, tylko wydłużenie obliczeń. Dla $\varepsilon = 10^{-9}$ został przekroczony limit kroków całkowania. Z kolei przy $\varepsilon = 0{,}1$ wykres jakościowo odbiega od tego najdokładniejszego.

Model obiektu z inercją

Zmienna stanu ma przebieg wykładniczy o stałej czasowej $0{,}25s$. Zwiększanie dokładności tylko wydłuża obliczenia — przebieg bez zmian. Górna granica dla $\varepsilon = 10^{-11}$; dolna dla $\varepsilon = 0{,}1$ — wówczas przebieg nie jest wiarygodny.

Należy zdecydować się, które ε wybrać. Nie ma sensu mierzyć w to, które daje najwierniejszą symulację, jeśli jest to okupione bardzo długim czasem obliczeń. Wystarczy zadowolić się parametrem, który da przebieg zbliżony do dokładnego, ale policzony o rząd wielkości szybciej. Wydaje się, że przyjęcie ε od 2 do 3 dekad lepszego niż to, które daje jeszcze dopuszczalną trajektorię, jest bezpiecznym rozwiązaniem. Przykładowe za-

leżności czasów całkowania i błędów od ε zostały przedstawione na wykresie poniżej (8).



Rys. 8 Czasy i błędy całkowania — model wg Lotka-Volterra

Następnie należy sprawdzić, czy domyślna wartość δx wystarcza, by zarejestrować wszystkie istotne elementy trajektorii. Przypuszczalnie δx rzędu 1/1000 - 1/100 parametru opisującego dynamikę wystarczy, by uchwycić *stromizny* trajektorii. Poniżej przedstawiam dalszy przebieg eksperymentu dla obu modeli.

Model wg Lotka-Volterra

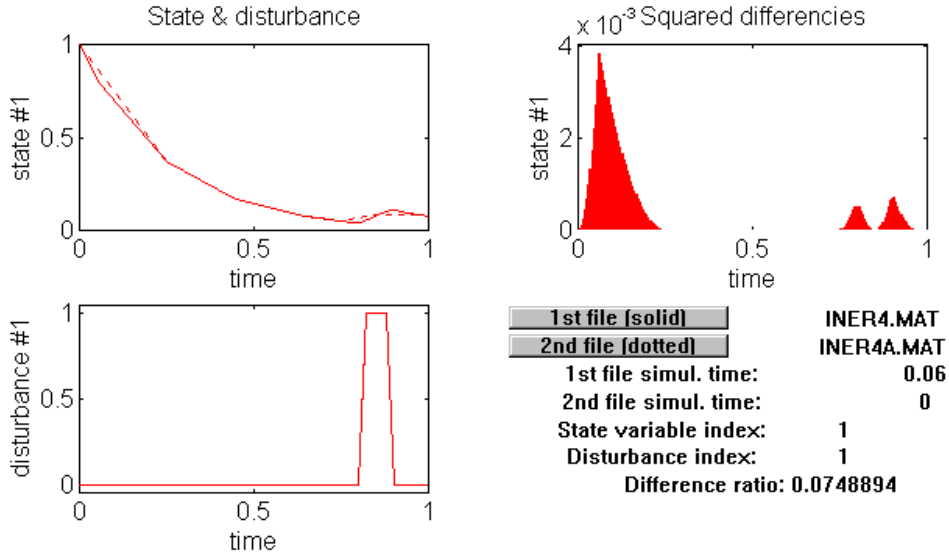
Przy $\varepsilon = 0{,}1$ nie do zaakceptowania, ale $\varepsilon = 0{,}01$ już do przyjęcia dobrym rozwiązaniem będzie przejść do doboru δx z $\varepsilon = 10^{-4}$.

Domyślna wartość minimalnego przyrostu, który jest zapisywany jest 0,01, co przy dynamice równej około $7s$ jest całkiem wystarczające. Nie ma potrzeby modyfikacji δx.

Model obiektu z inercją

Podobnie, jak dla Lotka-Volterra, przyjmuję $\varepsilon = 10^{-4}$.

Podobnie, jak dla przykładu obok, wartość domyślna δx wystarcza, by uchwycić wszystkie punkty trajektorii. Pokusiłem się jednak o sprawdzenie, co stanie się dla $\delta x = 0{,}1$. Wykresy dla tego przypadku są zaznaczone na rysunku poniżej liniami przerywanymi (9).

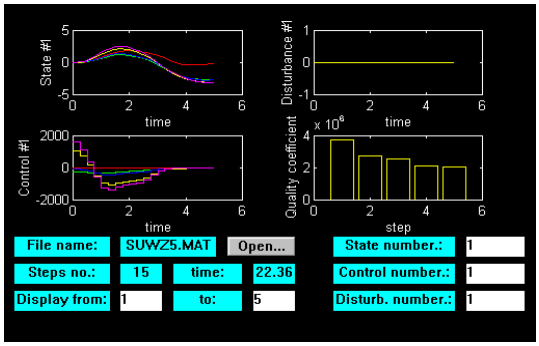


Rys. 9 Dobór minimalnego przyrostu rejestrowanego podczas całkowania

Okazało się, że parametry standardowe nie muszą być istotnie modyfikowane. Akurat dla tych przykładów. Mogą jednak trafiać się obiekty o zupełnie innej dynamice i wówczas opisane badania algorytmu całkującego okażą się nieodzowne.

4.2 Dobór parametrów optymalizacji

Zasadniczo procedura optymalizacyjna jest wszechstronna a jej parametry zostały sensownie dobrane i nie wymagają modyfikacji. Jednak użyte kryterium *stopu* jest wrażliwe na skalę wartości, jaką może przyjmować optymalizowana funkcja. Dotyczy to stopu ze względu na wartość gradientu. Kryterium stopu w punkcie $\boldsymbol{x}$ dla funkcji $\mathbf{f}_0(\boldsymbol{x})$ o gradiencie $\mathbf{g}(\boldsymbol{x})$ ma postać: $\frac{\max_i\left(\left|\mathbf{g}_i(\boldsymbol{x})\right|\cdot\max\left(\left|\boldsymbol{x}_i\right|,1\right)\right)}{\max\left(\mathbf{f}_0(\boldsymbol{x}),1\right)}<c$, gdzie c jest *tolerancją* określaną przez użytkownika. Jak widać, umiejętny dobór c określa, kiedy optymalizacja się zakończy, a zatem jak dobre będzie wyznaczone sterowanie. Warto zwrócić szczególną uwagę na przypadek, gdy $\mathbf{g}(\boldsymbol{x})$ jest bardzo małe a $\mathbf{f}_0(\boldsymbol{x})$ i $\boldsymbol{x}$ — mniejsze od jedności. Wówczas powyższe kryterium zamienia się w $\max_i\left|\mathbf{g}_i(\boldsymbol{x})\right|<c$, co generalnie dla małego $\mathbf{g}(\boldsymbol{x})$ powoduje przedwczesne przerwanie pętli optymalizacji. Dlatego drugim (a raczej równorzędnym) zadaniem jest takie przeskalowanie wskaźnika jakości, by w miarę zbliżania się do optimum nie ginął on przedwcześnie przy „1” w mianowniku.



Rys. 10 Dobór parametrów optymalizacji

Można więc przystąpić do doboru parametrów (standardowo $c=10^{-3}$, skala wskaźnika jakości jest dowolna). Stworzony przeze mnie osobny program do rejestracji kolejnych faz optymalizacji wraz ze skryptem wizualizującym wyniki (10) umożliwia oszacowanie wyników. Niestety, trudno jest przebadać działanie optymalizacji dla wszystkich stanów modelu. Dlatego wyniki dobre w jednym przypadku nie wykluczają możliwości, że w przyszłości obiekt znajdzie się w takim stanie, że optymalizacja zawiedzie. Podobnie trudno jest wybrać testowy przebieg zakłóceń. W eksperymentach zaczynałem symulację zawsze od stanu początkowego i wybierałem długi lokalny przedział sterowania a zakłócenia raz zdefiniowane były konsekwentnie używane w dalszych badaniach.

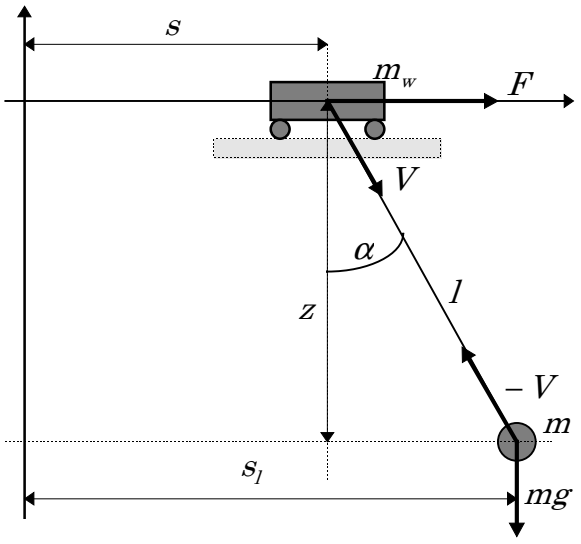
Próbowałem wypracować procedurę projektową posługując się dwoma nowymi modelami obiektów dynamicznych: wahadła i suwnicy.

Wahadło matematyczne. Model jest opisany równaniem stanu:

$$\begin{aligned} \dot{x}_1 &= x_2 & x_1(0) &= 1 \\ x_2 &= -225\sin(x_1) + 100z\cos(x_1) + m & x_2(0) &= 0. \end{aligned}$$

Wychylenie wahadła jest reprezentowane przez x_1 a prędkość kątowa przez x_2. Zakłócenia są symbolizowane przez z i mogą być interpretowane jako siła przyłożona do masy w kierunku poziomym. Sterowanie m to moment przyłożony do osi wahadła. Układ jest dla dużych wychyleń nieliniowy. Podcałkowy wskaźnik jakości: $\mathbf{f}_0(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{z}, t) = x_1^2$. Zakłócenia mają postać sinusoidy. Wskaźnik jakości na lokalnym przedziale sterowania nie zależy od stanu w horyzoncie lokalnym ani od sterowania. Obrałem lokalny przedział sterowania o długości $1{,}5s$ czyli trzykrotnie dłuższy niż okres wahań. Na tym przedziale jest 10, równomiernie rozmieszczonych, punktów dyskretyzacji sterowania.

Suwnica. Jest to model nieliniowy, czwartego rzędu, suwnicy bramowej z zawieszonym na niej ładunkiem na sztywnym, nieważkim pręcie (11). Sterowanie układem odbywa się przez przyłożenie siły F do wózka. Model uwzględnia opory toczne wózka i nieliniowości dla dużych wychyleń wahadła.



s	przesunięcie wózka
s_l	przes. ładunku wzdłuż s
z	przes. ładunku wzdłuż z
α	kąt odchylenia ładunku
l	długość liny
F	siła napędzająca wózek
V	siła naprężająca linę
m_w	masa wózka
m	masa ładunku
r_w	opory toczne wózka
g	przyspieszenie ziemskie

Rys. 11 Model suwnicy bramowej

Równania stanu modelu są następujące:

$$\begin{aligned} r_w\dot{s} + m_w\ddot{s} &= F + V\sin\alpha \\ m\ddot{s}_l &= -V\sin\alpha \\ m\ddot{z} &= -V\cos\alpha + mg \end{aligned} \qquad \text{przy czym} \qquad \begin{aligned} s_l &= s + l\sin\alpha \\ z &= l\cos\alpha, \end{aligned}$$

co prowadzi do

$$\begin{aligned} r_w\dot{s} + (m + m_w)\ddot{s} - ml\dot{\alpha}^2\sin\alpha + ml\ddot{\alpha}\cos\alpha &= F \\ \ddot{s}\cos\alpha + l\ddot{\alpha} + g\sin\alpha &= 0. \end{aligned}$$

Dążąc do ujednolicenia nazw zmiennych stanu ($x_1 = s, x_2 = \dot{s}, x_3 = \alpha, x_3 = \dot{\alpha}$) otrzymujemy po przekształceniach

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{-r_w x_2 + m \sin x_3 \cdot \left(l x_4^2 + g \cos x_3 \right) + F}{m_w + m \sin x_3} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{\cos x_3 \cdot \left(r_w x_2 - F - m l x_4^2 \sin x_3 \right) - g \left(m + m_w \right) \sin x_3}{l \left(m_w + m \sin^2 x_3 \right)}
\end{aligned}$$

, co przy

$$\begin{aligned}
r_w &= 100 && N \cdot s/m \\
m &= 200 && kg \\
g &= 10 && kg \cdot m/s^2 \\
m_w &= 600 && kg \\
l &= 5 && m
\end{aligned}$$

oraz standardowym oznaczeniu przez m sterowania, daje równania:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{-100 x_2 + 200 \sin x_3 \cdot \left(5 x_4^2 + 10 \cos x_3 \right) + m}{600 + 200 \sin x_3} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{\cos x_3 \cdot \left(100 x_2 - m - 1000 x_4^2 \sin x_3 \right) - 8000 \sin x_3}{l \left(3000 + 1000 \sin^2 x_3 \right)}
\end{aligned}$$

przy

$$\begin{aligned}
\dot{x}_1(0) &= 0 \\
\dot{x}_2(0) &= 0 \\
\dot{x}_3(0) &= 1 \\
\dot{x}_4(0) &= 0.
\end{aligned}$$

Ponieważ okres drgań własnych jest około $4{,}5s$ dlatego przeprowadzałem optymalizację na horyzoncie równym $10s$. Dyskretyzacja sterowania równomierna, w 20 punktach. Według nowych oznaczeń x_1 reprezentuje położenie wózka (w metrach) a x_3 — wychylenie wahadła (w radianach). Wskaźnik jakości

$$\mathbf{f}_0(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{z}, t) = \left(x_1 + 5 \sin x_3 \right)^2 + \tan^2 x_3$$

ma postać sumy. Jej pierwszy składnik rośnie, gdy odcięta położenia ładunku oddala się od zera. Drugi — gdy wychylenie ładunku przekroczy 45°. Drugi składnik zapobiega irracjonalnemu obracaniu ładunkiem w imię zoptymalizowania samego odchylenia po s (zdarzyło mi się takie „optymalne" sterowanie uzyskać).

Optymalizacja sterowania wahadłem została przeprowadzona najpierw dla domyślnych wartości parametrów. Postanowiłem przekonać się na ile uniwersalność metody BFGS jest prawdziwa. Oto zestawienie najistotniejszych wyników optymalizacji:

Krok	mnożnik wskaźnika jakości	tolerancja c	liczba kroków	czas obliczeń (s)	wskaźnik jakości unormowany*	uwagi dotyczące wyników
Sprawdzenie, jak zmieniają się wyniki optymalizacji przy skalowaniu $\mathbf{f}_0$ co dekadę:						
1	⚐ 10^0	⚐ 10^{-3}	21	39	0,141	istotne pierwsze 4 etapy
2	⇗ 10^1	10^{-3}	6	14	0,161	istotne pierwsze 3 etapy
3	⇗ 10^2	10^{-3}	29	49	0,088	istotne pierwsze 4 etapy
4	⇗ 10^3	10^{-3}	29	52	0,084	jak wyżej, bez zmian
Dwukrotna poprawa sterowania dla kroków 3 i 4 jest okupiona dużo dłuższym czasem obliczeń. Sprawdzenie, co się dzieje, gdy przeskalować $\mathbf{f}_0$ o dekadę w dół:						
5	⇘ 10^{-1}	10^{-3}	3	4	0,723	nie optymalizuje się

* Skalowanie $\mathbf{f}_0$ powoduje, że całkowy wskaźnik jakości zmienia się proporcjonalnie do mnożnika przy $\mathbf{f}_0$. Unormowany wskaźnik jakości to wskaźnik uzyskany w symulacji a następnie podzielony przez aktualną wartość mnożnika przy $\mathbf{f}_0$.

Wówczas przeskalowany wskaźnik jest zbyt mały i optymalizacja nie dokonuje się. Do manipulacji tolerancją c przechodzę z pierwotnym wskaźnikiem jakości pomnożonym przez 10 (krok 2.). Sprawdzenie, co się dzieje przy zmniejszającym się c:						
6	↓ 10^{1}	↑ 10^{-2}	6	11	0,161	wyniki jak dla kroku 2.
7	10^{1}	↑ 10^{-1}	2	1	0,723	nie optymalizuje się
Zwiększanie tolerancji gradientu c nie powoduje poprawy ani czasu optymalizacji, ani jej wyniku. Sprawdzenie, co się dzieje dla zmniejszającego się c:						
8	10^{1}	↓ 10^{-4}	6	11	0,161	wyniki jak w kroku 2.
9	10^{1}	↓ 10^{-5}	6	11	0,161	wyniki jak w kroku 2.

Tabela 1 Procedura doboru parametrów optymalizacji dla modelu wahadła

Opisywany test ma na celu jedynie zgrubne dobranie parametrów optymalizacji. Przez naprzemienne manipulowanie skalą wskaźnika i parametrem c znajduje się takie ustawienie, które daje najlepszy wynik. Nie można jednak, w pogoni za ścisłym optimum, zapomnieć o aspekcie czasowym: sterowanie repetycyjne będzie użyteczne tylko wtedy, gdy optymalizacja wykona się w czasie znacznie krótszym niż długość etapu. Dlatego dalej będę posługiwał się parametrami z kroku 2., a nie 3., gdzie dwukrotna poprawa jakości jest okupiona ponadtrzykrotnie dłuższymi obliczeniami.

Oto opis postępowania dla modelu suwnicy:

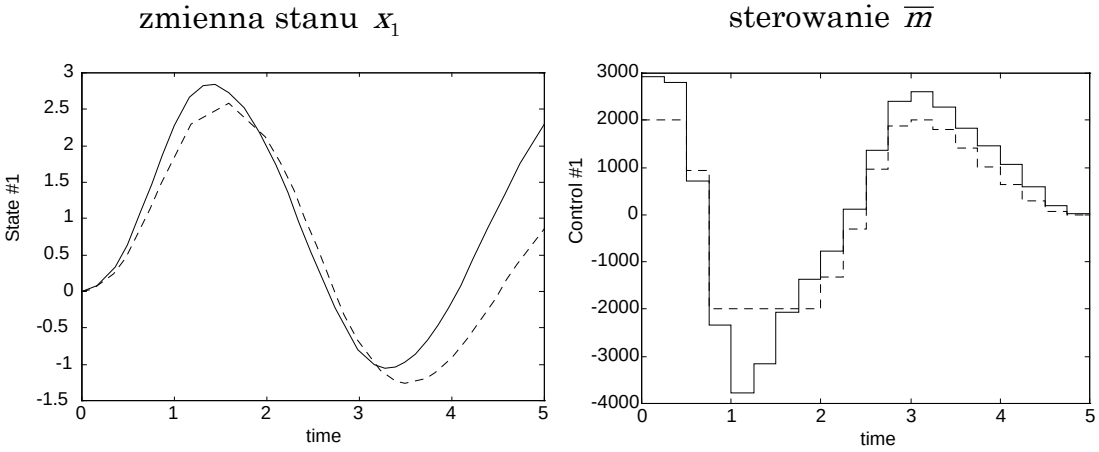
Krok	mnożnik wskaźnika jakości	tolerancja c	liczba kroków	czas obliczeń (s)	wskaźnik jakości unormowany	uwagi dotyczące wyników
Sprawdzenie, jak zmieniają się wyniki optymalizacji przy skalowaniu $\mathbf{f}_0$ co dekadę:						
1	→ 10^{0}	→ 10^{-3}	2	1	37,4	nie optymalizuje się
2	↑ 10^{1}	10^{-3}	2	1	37,3	nie optymalizuje się
3	↑ 10^{2}	10^{-3}	2	1	37,3	jak wyżej
4÷7	↑ $10^{3\div6}$	10^{-3}	2	1	37,3	jak wyżej
Optymalizacja kończy się zaraz po starcie we wszystkich powyższych przypadkach. Należy więc, operując tolerancją c, wprowadzić chociaż zmianę jakościową wyników:						
8	10^{0}	↓ 10^{-4}	4	5	37,3	optymalizacja ruszyła
9	10^{0}	↓ 10^{-5}	4	5	37,3	wyniki jak wyżej
10÷11	10^{0}	↓ $10^{-6\div7}$	4	5	37,3	bez zmian
Zmiana c spowodowała, że procedura optymalizacyjna ruszyła, ale nie można uzyskać poprawy sterowania. Można zatem spróbować znów skalować $\mathbf{f}_0$, wychodząc od kroku 9:						
12	↑ 10^{1}	↓ 10^{-5}	6	5	27,5	znaczna poprawa
13	↑ 10^{2}	10^{-5}	7	8	27,4	wyniki jak wyżej
14	↑ 10^{3}	10^{-5}	5	6	27,4	tak samo, ale 5 kroków
15	↑ 10^{4}	10^{-5}	10	13	20,8	lepszy wynik, ale dłużej
16	↑ 10^{5}	10^{-5}	15	24	16,9	jak wyżej, poprawa
17	↑ 10^{6}	10^{-5}	47	112	8,9	b. dobrze, ale b. długo
18	↑ 10^{7}	10^{-5}	31	67	10,7	gorzej — krócej
19	↑ 10^{8}	10^{-5}	42	113	9,6	bez zmian jakościowych
20	↑ 10^{9}	10^{-5}	55	154	9,9	długo i bez korzyści

Tabela 2 Procedura doboru parametrów optymalizacji dla modelu suwnicy

Optymalizacja sterowania modelu suwnicy nie powiodła się przy argumentach domyślnych. Konsekwentne manipulowanie skalą $\mathbf{f}_0$ też nie doprowadziło do sensownych wyników. Jednak próby zmian skali przeplatane z szukaniem c powiodły do zestawu akceptowalnych parametrów (kroki 15÷19). Postanowiłem, nieco arbitralnie, prowadzić dalsze obliczenia dla zestawu parametrów z kroku 16.

Jednym z nie wymienionych do tej pory, a istotnym, parametrem jest maksymalna liczba iteracji procedury BFGS. Standardowo jest to 200, ale przy gęstej dyskretyzacji sterowania wymiarowość zadania optymalizacji jest taka, że wartość standardowa jest z łatwością osiągana. Wypadało by więc zmieniać ten limit w zależności od dyskretyzacji. Ze względów praktycznych wszystkie dalsze symulacje odbywają się w zasadzie bez ograniczenia na maksymalną ilość iteracji BFGS.

Pominięta została jeszcze jedna kwestia: ograniczeń na sterowanie. We wszystkich dotychczasowych badaniach nie były one osiągane. Można sprawdzić, jak wpłyną na sterowanie, trajektorie stanu i wskaźnik jakości. Na poniższych wykresach (12) linie ciągłe odnoszą się do przypadku bez ograniczeń; linie kropkowane — z ograniczeniami na sterowanie (od -2000 do 2000):



Rys. 12 Model suwnicy. Zmienne stanu i odpowiadające im sterowania (ograniczane i nie ograniczane). Model suwnicy.

Tym razem ograniczenie sterowania nie ma wielkiego wpływu ani na kształt trajektorii stanu, ani na wskaźnik jakości (17,4 przy ograniczeniach, 16,9 — bez). Wynika to z natury obiektu: do jego ustabilizowania się potrzebny jest przede wszystkim czas, nie olbrzymie sterowania. Drugi rysunek pozwala, ponadto, zauważyć prawidłowość: niedostatki sterowania, wynikające z wystąpienia ograniczeń, są rekompensowane dłuższym stosowaniem sterowania o tym samym charakterze: widać to szczególnie w okolicach ekstremów sterowania — bez ograniczeń są one ostre i krótkie, z ograniczeniami — stępione. Sprawdziwszy, że algorytm umożliwia sterowanie z ograniczeniami, będę starał się dalej nie korzystać z tej

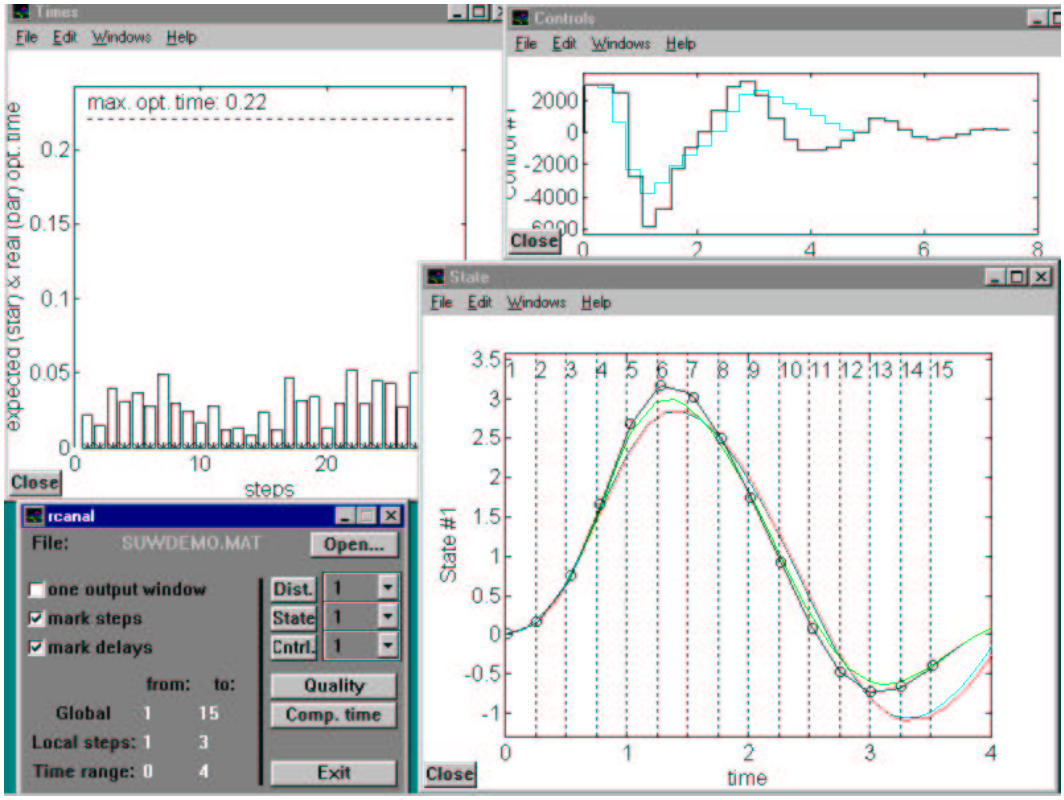
właściwości. Głównym tego powodem jest znacznie dłuższy (7-krotnie w tym przykładzie) czas obliczeń przy ograniczeniach.

Podsumowując: są dwa kryteria doboru parametrów optymalizacji: jak najlepszy wskaźnik jakości i jak najkrótszy czas obliczeń. Parametrami są: skala funkcji podcałkowej $\mathbf{f}_0$ i tolerancja zerowania się gradientu c. Manipulując na przemian oboma należy przede wszystkim doprowadzić do uruchomienia procedury optymalizacyjnej (*vide* przykład suwnicy), a później do uzyskania jak najlepszego wskaźnika jakości przy sensownym czasie obliczeń.

4.3 Dobór długości lokalnego przedziału sterowania

Dla całkowitego zestrojenia algorytmu należy dobrać wartości jeszcze kilku parametrów. Są one ze sobą powiązane, więc trzeba zdecydować, które określić na początku. Pierwszą jest długość lokalnego przedziału sterowania T_i^*. Oczywiście, długość etapu sterowania czy sposób rozmieszczania punktów dyskretyzacji są równie ważne, ale — moim zdaniem — długość przedziału stoi przed nimi. Dlatego, że aby algorytm działał poprawnie, przedział lokalny musi odkrywać przed algorytmem wystarczająco dużo z dynamiki obiektu. Krótkowzroczność optymalizacji sterowania może spowodować, że regulator, dążąc do znakomitych wyników bieżących, umieści obiekt w stanie, w którym nie uniknie się później znacznego pogorszenia jakości sterowania. Ryzyko takie nie istnieje w przypadku programowania dynamicznego oraz sterowania repetycyjnego z ograniczanym horyzontem lokalnym. Temat tego podrozdziału można więc tłumaczyć, jako szukanie kompromisu między *niezawodną ale czasochłonną repetycją z ograniczanym horyzontem lokalnym* a *szybkim ale ryzykownym sterowaniem z krótkim przedziałem lokalnym*.

Przedstawione wyniki uzyskałem za sprawą głównego produktu mojej praktyki programistycznej: programu symulującego przebieg sesji sterowania repetycyjnego. Ich wizualizacji dokonuje skrypt uruchamiany w *Matlabie*. Zapewnia to całkowitą przenośność oprogramowania. Skrypt umożliwia oglądanie wycinków trajektorii stanu, sterowania i zakłóceń; szacowanie średnioetapowego wskaźnika jakości i zestawienie czasów symulacji. Na poniższym rysunku (13) przedstawiam próbkę jego możliwości:



Rys. 13 Skrypt wizualizujący wyniki sterowania repetycyjnego*

Poniżej przedstawiam wyniki symulacji dla trzech modeli: wahadła, suwnicy i potrójnej inercji. Starałem się uchwycić konsekwencje, które przyniosło skracanie przedziału lokalnego w kolejnych eksperymentach.

Krok	długość przedz. lokalnego □s□	jakość od etapu 1.	jakość od etapu 5
1	1,50	0,324	0,0151
2	1,20	0,321	0,0178
3	0,90	0,321	0,0134
4	0,60	0,305	0,0285
5	0,45	0,343	0,0547
6	0,30	0,499	0,2285

Tabela 3 Procedura doboru T_i^* dla modelu wahadła

Kryterium niech stanowi średnioetapowy wskaźnik jakości liczony bądź od początku sesji (3, kol. 3.), bądź od etapu, na którym można uznać, że składowa przejściowa (od warunków początkowych) już zanikła i liczy się tylko wpływ zakłóceń. Ten drugi wynik jest bardziej miarodajny, gdyż odzwierciedla to, co dzieje się przez większość sesji. Na pierwszy składa się głównie zły (duży) wskaźnik jakości na kilku początkowych etapach.

* Wyniki symulacji wizualizowane przez skrypt w *Matlabie* są prezentowane zawsze w ten sam sposób. Na wykresach trajektorii oraz wskaźnika jakości czas (wg dynamiki obiektu) jest na osi odciętych. Na osi rzędnych mogą się znaleźć: sterowanie, zmienna stanu, zakłócenie i wskaźnik jakości. Na wykresach trajektorii kolorem czarnym oznaczono trajektorię zrealizowaną; kolorami innymi — trajektorie przewidywane w kolejnych etapach. Liczby oznaczające poszczególne etapy pojawiają się na górnej (wykresy trajektorii) oraz na dolnej (słupkowy wykres czasu obliczeń) osi odciętych.

Trzy inercje. Model obiektu jest opisany równaniem:

$\dot{x}_1 = -x_1 + m \qquad x_1(0) = 0$

$\dot{x}_2 = -x_2 + x_1 + z \qquad x_2(0) = 0$

$\dot{x}_3 = -x_3 + x_2 \qquad x_3(0) = 1$

Zakłócenia mają przebieg sinusoidalny, jakość sterowania określa $\mathbf{f}_0 = 100x_3^2$. Dyskretyzacja sterowania co 0,15.

W przypadku wahadła, o ile średni wskaźnik za całą sesję nie zmienia się znacząco, to jakość sterowania liczona od etapu 5. wyraźnie się pogarsza (krok 4. i następne). Prowadzi to do wniosku, że długość lokalnego przedziału sterowania nie powinna być mniejsza niż około $0{,}9s$. Dla modelu suwnicy sytuacja nie jest tak jasna. Otrzymane wyniki nie dają się usystematyzować. Ale oto w przypadku nowego modelu (trójinercyjnego) istota doboru długości przedziału staje się aż nadto wyraźna. Dla lokalnego przedziału sterowania o długości $1{,}5s$ wskaźnik jakości jest równy 0,291; dla $0{,}75s$ — już 0,743 przy bardzo gwałtownym sterowaniu; dla $0{,}3s$ — układ regulacji destabilizuje się.

Podsumowując można stwierdzić, że istnieją obiekty, dla których długość lokalnego przedziału sterowania może być bardzo istotna; parametr ten należy dobierać tak, by mieć gwarancję stabilności układu sterowania — ale jednocześnie, w imię stabilności, nie przeciągać obliczeń niepotrzebnie.

4.4 Dyskretyzacja sterowania na przedziale lokalnym

Jedną z cech charakterystycznych dla badanego algorytmu jest możliwość nierównomiernej dyskretyzacji sterowania na przedziale lokalnym. To usprawnienie uniemożliwiło traktowanie obiektu i regulatora jako dyskretnych; w zamian użytkownik ma możliwość dowolnego rozmieszczania chwil zmian sterowania na przedziale lokalnym. Uciąglenie układu pozwoliło ponadto na rzetelne badanie opóźnień decyzyjnych.

Właściwa lokalna dyskretyzacja sterowania ma na celu poprawę sterowania w dwóch aspektach. Pierwszy, oryginalny, jest reprezentowany przez średni wskaźnik jakości sterowania na całym przedziale globalnym bądź jego fragmencie. W zasadzie można by na nim poprzestać, gdyby założyć, że algorytm nie będzie stosowany w praktyce. Bo o jego użyteczności zadecyduje w równej mierze szybkość wykonywanych przezeń obliczeń sterowania — i to jest ów drugi aspekt. Mając ciągle na uwadze rolę czasów optymalizacji, w najbliższych paragrafach skupię się przede wszystkim na wpływie dyskretyzacji na wartość średnią wskaźnika jakości.

Istotę sterowania repetycyjnego stanowi cykliczna optymalizacja sterowania układem na przedziale lokalnym, by później niewielki fragment trajektorii sterowania wykorzystać w rzeczywistości. Lokalny horyzont sterowania nie jest zazwyczaj tożsamy z horyzontem globalnym a wynika to ze względów praktycznych. Dlatego lokalny przedział sterowania jest tylko niewielką (zwłaszcza, gdy $t_f = \infty$) częścią przedziału globalnego. Rzecz w tym, by warunki sterowania na przedziale lokalnym jak najlepiej odzwierciedlały to, co ma miejsce w rzeczywistości. Dlatego, dobierając parametry sterowania repetycyjnego, projektant może ustalać

dyskretyzację i kryteria jakości zupełnie niezależne od obowiązujących dla układu rzeczywistego.

Powstaje pytanie: czy (jak podpowiada intuicja) nie byłoby najlepiej, gdyby wszystkie uwarunkowania sterowania na przedziale lokalnym były wierną kopią tych dla horyzontu globalnego□A więc: równania stanu takie same, zakłócenia takie same; tak samo zdefiniowany podcałkowy wskaźnik jakości $\mathbf{f}_0$; lokalne punkty dyskretyzacji sterowania pokrywające się z globalnymi itp. Nie jest to oczywiste. O ile nie budzi wątpliwości wierne odwzorowanie równań stanu czy trajektorii zakłóceń, to identyczna dyskretyzacja sterowania wcale nie musi skutkować optymalnymi wskaźnikami.

Dla wyjaśnienia weźmy dowolny obiekt sterowany repetycyjnie. Na pewnym etapie przeprowadzana jest optymalizacja sterowania na przedziale lokalnym z dyskretyzacją i wskaźnikiem jakości takim samym, jak dla układu rzeczywistego. Jednak optymalizator „nie wie", że w normalnych warunkach sesja sterowania trwa także po horyzoncie lokalnym. Dlatego wyznacza sterowanie, które pod koniec przedziału lokalnego umieszcza obiekt w niekorzystnym, wysoko karanym stanie. Taki scenariusz jest prawdopodobny. Optymalizator „pokutuje" bowiem za niekorzystne sterowania z początku przedziału przez całą jego długość. Natomiast nierozsądne sterowania blisko horyzontu lokalnego uchodzą mu niemal bezkarnie.

Aby temu zaradzić, można usunąć kilka punktów sterowania, które leżą blisko horyzontu lokalnego. Wówczas ostatnie sterowanie na przedziale lokalnym pozostaje *zamrożone* przez czas dłuższy, niż poprzednie sterowania — zatem waga decyzji podjętej co do niego rośnie.

Istnieje jeszcze jeden sposób, aby zapobiec opisywanym niebezpieczeństwom: to funkcja kary $\mathbf{Q}$ za stan w horyzoncie lokalnym. Rozsądnie dobrana, powinna dostatecznie uświadamiać późniejsze skutki pozostawienia obiektu w niekorzystnym stanie pod koniec przedziału lokalnego.

Istnieją więc dwie dwa sposoby poprawy wad sterowania wynikających z jego repetycyjnej natury: miejscowe rozrzedzenie dyskretyzacji oraz wprowadzenie kary za stan końcowy. Można zastanawiać się, czy przedstawione problemy w ogóle mogą mieć wpływ na jakość sterowania, skoro powstają blisko horyzontu lokalnego. Temu poświęcone są następne paragrafy.

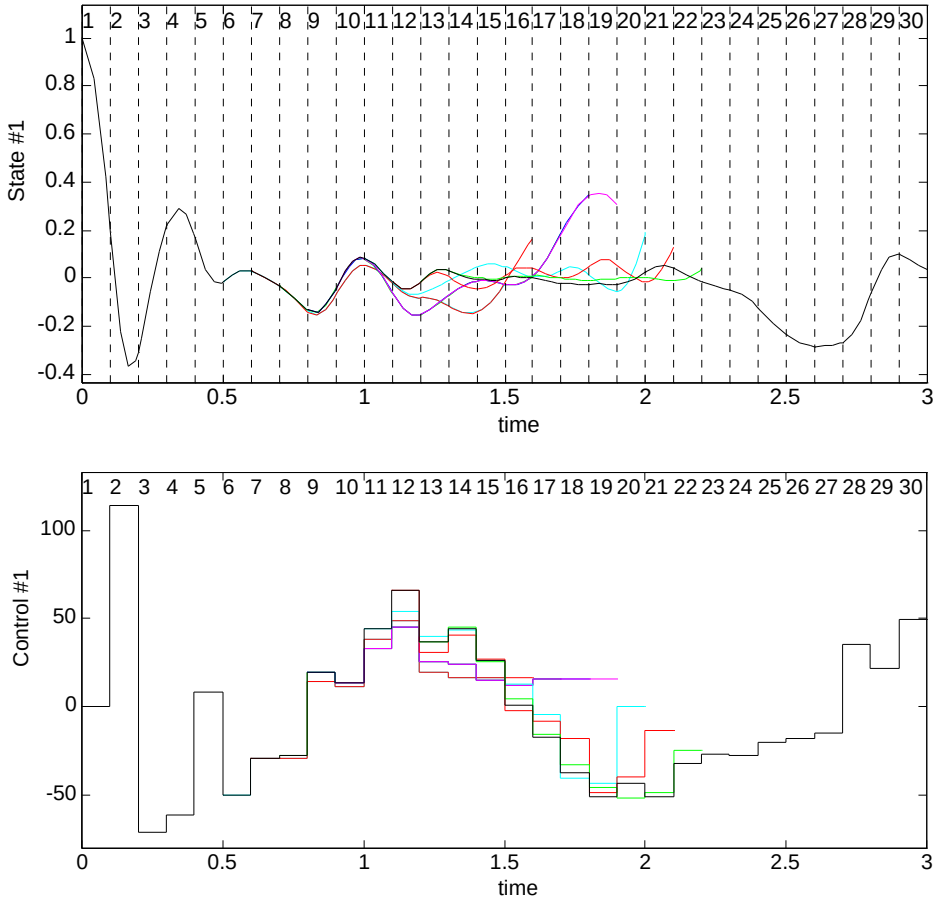
4.4.1 Model wahadła

Przedmiotem badań jest, poznany wcześniej, model wahadła. Bazując na wynikach z poprzedniego podrozdziału można uznać, że przedział lokalny o długości $1s$ wystarczy, by uchwycić dynamikę obiektu. Niech sesja sterowania składa się z 30 etapów o długości 0,1. Wartość ta jest dobrana arbitralnie — w warunkach przemysłowych częstość zmian sterowania będzie określona dynamiką elementów wykonawczych, dynamiką samego procesu, szybkością działania regulatora. Poza tym, dla sterowania repetycyjnego optymalizacja sterowania nie będzie musiała doko-

nywać się częściej niż pomiary stanu obiektu czy wyznaczenie nowej prognozy zakłóceń. I właśnie te parametry określą długość pojedynczego etapu.

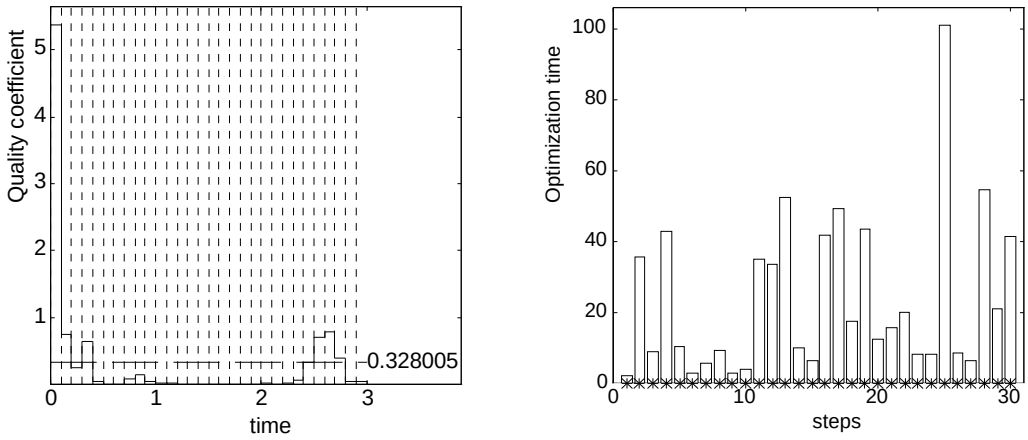
Dyskretyzacja na przedziale lokalnym jest w wersji wyjściowej testów równomierna; jest fragmentem dyskretyzacji na przedziale globalnym. Zakłócenia mają przebieg $z(t) = 0{,}5\sin 4t$. W przypadkach, gdy regulator nie zna w pełni zakłóceń, są one dla niego określone jako $z^*(t) = 0{,}5\sin 4t + 0{,}3\cos 5t$. Dla testów z funkcją kary za stan końcowy jest ona opisana przez $\mathbf{Q} = 10x_1^2$. □redni wskaźnik jakości jest liczony po ustaniu wpływu warunków początkowych, czyli od $t = 0{,}5\ s$.

Pełna znajomość zakłóceń. Oto przebiegi zmiennej stanu x_1 i sterowania m wraz z lokalnymi trajektoriami stanu i sterowania dla etapów od 6 do 13 (14).



Rys. 14 Stan i sterowanie dla wahadła — przebiegi wyjściowe

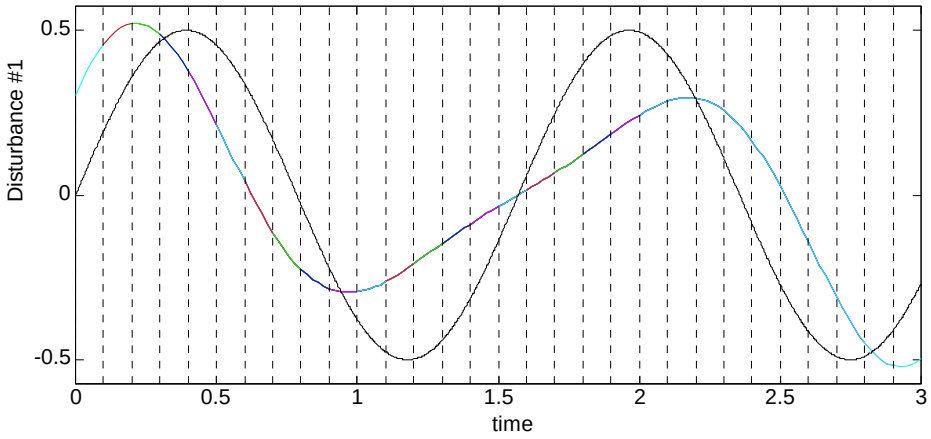
Przewidywane przebiegi zmiennych stanu i sterowań nie różnią się jakościowo od zrealizowanych. W swojej początkowej fazie są bardzo zgodne. Na kolejnych wykresach (15) zostały przedstawione etapowe wskaźniki jakości i czasy optymalizacji.

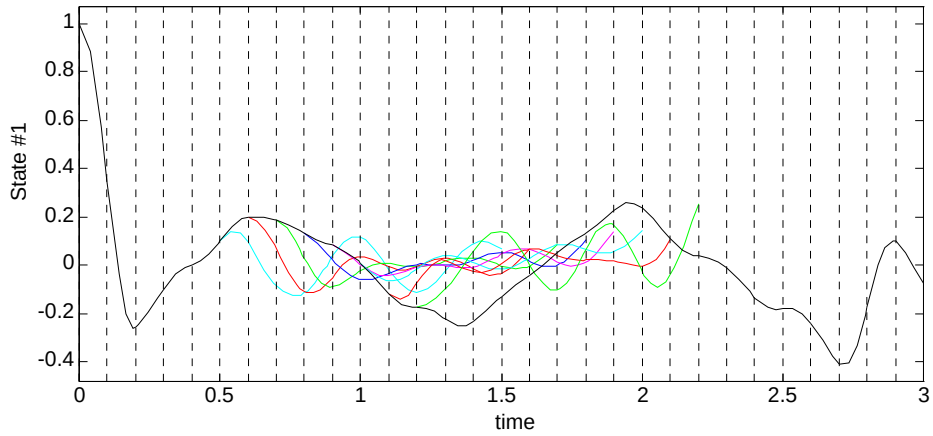


Rys. 15 Wyjściowy wskaźnik jakości i czasy optymalizacji (w sekundach) dla wahadła

Wykres wskaźnika jakości jeszcze raz potwierdza, że powinien on być liczony z pominięciem kilku pierwszych etapów. Z wykresu czasów optymalizacji na poszczególnych etapach można wywnioskować, że w większości są one umiarkowane (od 10 do 20*s*). Niestety, kilka z nich mocno odbiega od normy i podważa wskaźnik szybkości działania algorytmu. Obecnie jest nim całkowity czas symulacji. Powód stanowią prawdopodobnie pewne warunki początkowe, przy których procedura całkująca działa wyjątkowo powoli. Mocno wydłużone, odbiegające od średniego, czasy optymalizacji przestaną być problemem po wprowadzeniu do algorytmu ograniczeń na maksymalny czas trwania optymalizacji.

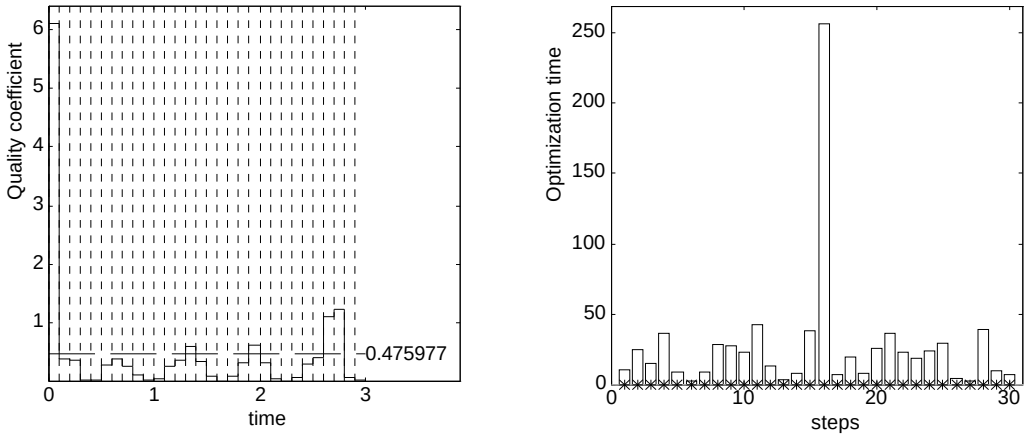
Niepełna znajomość zakłóceń. Dla porównania, na poniższych wykresach (16) zostały przedstawione przebiegi zakłóceń (kolorem czarnym oznaczono zakłócenia rzeczywiste) oraz zmiennej stanu.





Rys. 16 Wahadło — zakłócenia i stan przy ich niepełnej znajomości

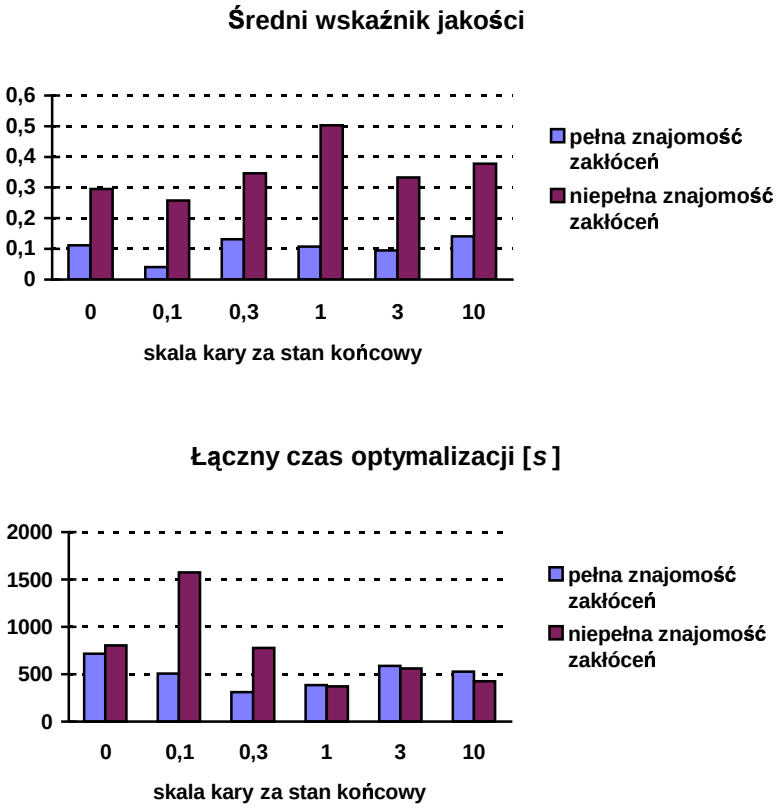
Niepełna wiedza o zakłóceniach natychmiast skutkuje większymi wahaniami zmiennych stanu. Natomiast wykresy wskaźnika jakości i czasów optymalizacji wyglądają następująco (17):



Rys. 17 Wahadło — wskaźnik jakości i czasy optymalizacji przy niepełnej wiedzy o zakłóceniach

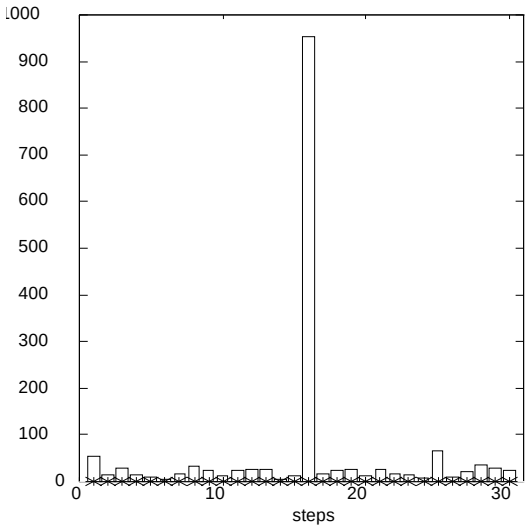
Na pierwszym z nich zaznaczają się cykliczne pogorszenia jakości sterowania. Można je skojarzyć z rozbieżnościami między przewidywanym przez optymalizator a rzeczywistym przebiegiem zakłóceń. Wykres czasów optymalizacji jeszcze raz potwierdza, że wśród ogólnie dobrych wyników trafiają się etapy z bardzo długim czasem obliczeń.

Badanie wpływu kary za stan końcowy. Jedną z metod poprawy jakości sterowania jest wprowadzenie kary za stan na końcu przedziału lokalnego. Dokonałem symulacji z karą $\mathbf{Q}$ (zdefiniowaną na początku) mnożoną przez 0,1, 0,3, 1, 3 i 10. Wpływ przeskalowania $\mathbf{Q}$ na czas i jakość sterowania przedstawiają poniższe wykresy (18):



Rys. 18 Wahadło — wyniki symulacji przy karze za stan końcowy

Okazuje się, że wprowadzenie umiarkowanej kary za stan końcowy poprawia wskaźnik jakości około dwukrotnie w przypadku pełnej wiedzy o zakłóceniach. Dalsze jej zwiększanie nie przynosi dalszej poprawy sterowania. Przy niepełnej wiedzy o zakłóceniach żaden z wariantów kary za stan końcowy nie przynosi istotnej poprawy.

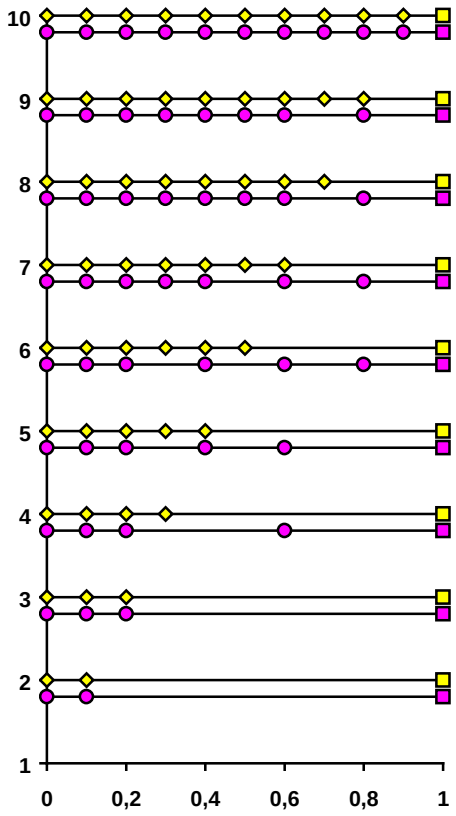


Rys. 19 Wahadło — przykładowy wykres czasów optymalizacji (w sekundach)

Analizując czasy obliczeń można zauważyć pewne ich skrócenie dla umiarkowanych kar za stan końcowy (0,1÷1) — ale tylko, gdy zakłócenia są w pełni znane. Inaczej czas się wydłuża, jednak jest to skutkiem bardzo przeciągających się obliczeń na jednym z etapów (19).

Można stwierdzić, że manipulowanie karą za stan końcowy nie daje radykalnej poprawy sterowania. Można, co prawda, dla przypadku ze znanymi zakłóceniami, mówić o poprawie wskaźnika dla pojedynczego przypadku, jednak nie jest to trend trwały. Natomiast o wynikach uzyskanych dla niepełnej znajomości zakłóceń nie można powiedzieć nic pozytywnego. Dlatego warto przebadać zachowanie się algorytmu przy rozrzedzaniu dyskretyzacji blisko horyzontu lokalnego.

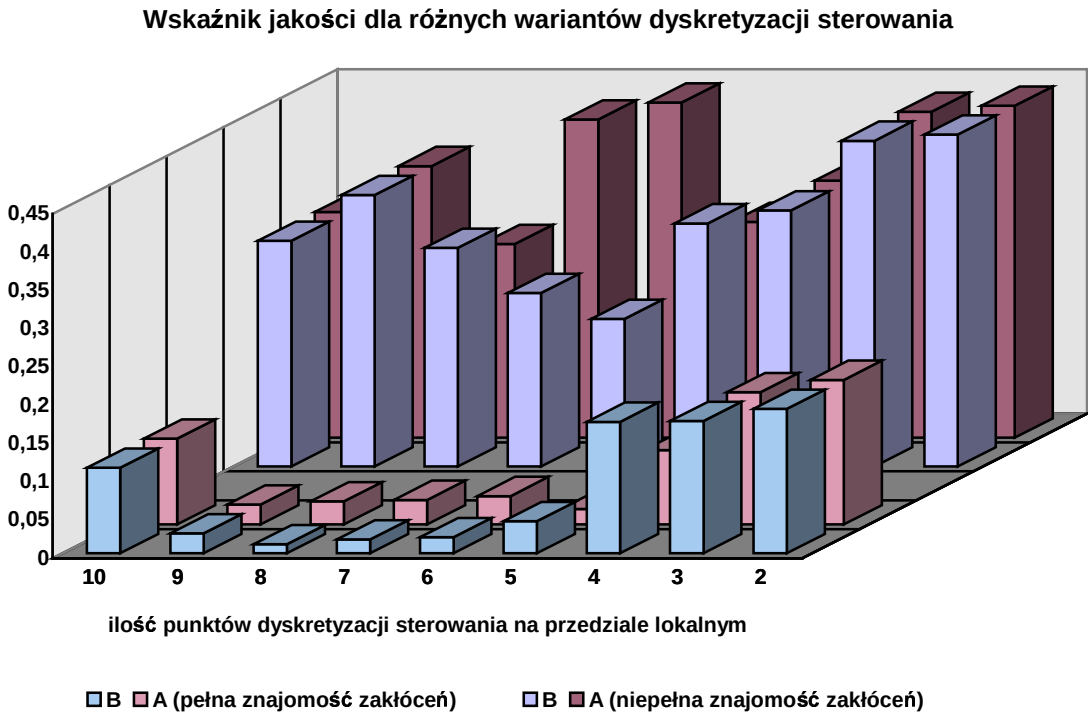
Badanie wpływu nierównomiernej dyskretyzacji sterowania. Niech stanem wyjściowym do badań będzie równomierna dyskretyzacja sterowania na przedziale lokalnym, zgodna z dyskretyzacją faktycznego sterowania. Można wówczas rozważyć różne strategie nierównomiernego rozmieszczania punktów zmian sterowania. Zasadniczo powinny one polegać na usunięciu ze zbioru podstawowego punktów, które leżą blisko horyzontu lokalnego.



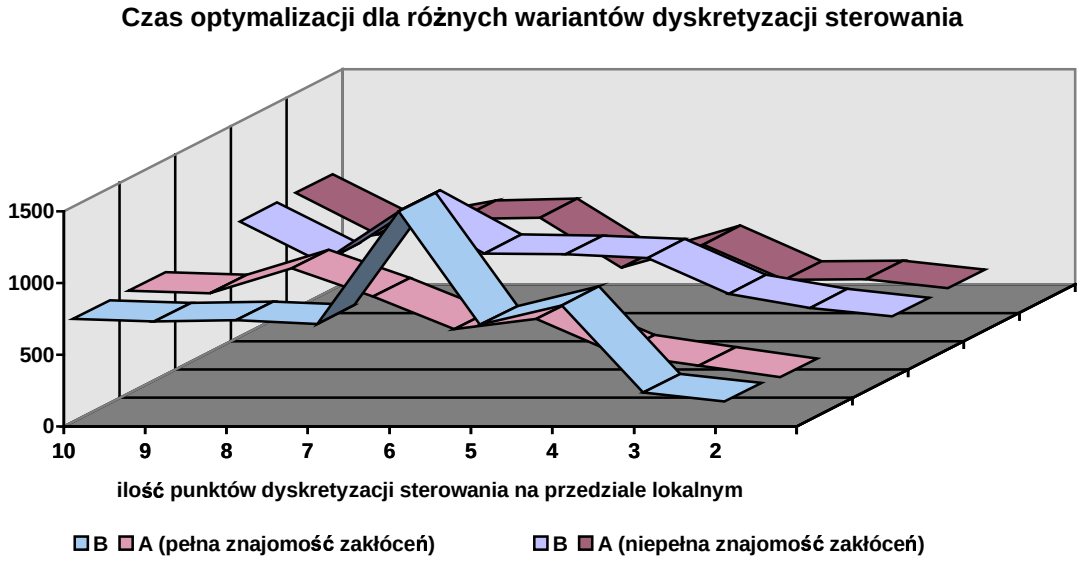
Rys. 20 Strategie rozrzedzania — przykład

Najprostszą strategią (strategia *A*) jest usuwanie punktów dyskretyzacji po kolei, poczynając od końca (20, kolor żółty). Wówczas strefa stałego sterowania powiększa się przy nie zmienionej dyskretyzacji na początku przedziału. Inna metoda (strategia *B*) może polegać na usuwaniu co drugiego punktu, począwszy od końca (kolor różowy). Gdy trzeba usunąć ich więcej niż 50□, wówczas od nowa zaczyna się usuwanie już przerzedzonych punków z końca przedziału, itd. W ten sposób rozrzedzenie dyskretyzacji dotyka punkty całego przedziału.

Poniższy rysunek (21) zawiera zestawienie jakości sterowania uzyskanej dla obu strategii rozrzedzania dyskretyzacji w przypadku pełnej (odcienie jaśniejsze) i niepełnej (odcienie ciemniejsze) wiedzy o zakłóceniach.



Rys. 21 Wahadło — jakość dla różnych dyskretyzacji

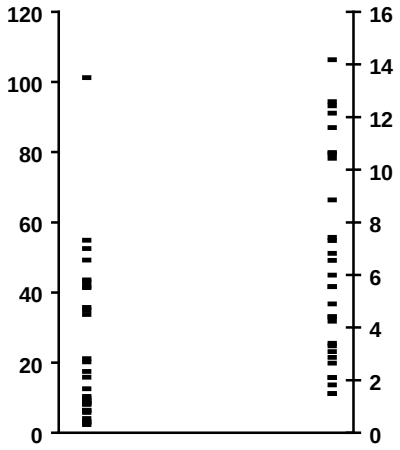


Rys. 22 Wahadło — czasy optymalizacji (w sekundach)

Okazuje się, że zmniejszenie liczby punktów lokalnej dyskretyzacji sterowania poprawia wyniki symulacji w sposób znaczny i trwały.

Zwłaszcza dla strategii *B* i przy pełnej wiedzy o zakłóceniach. Istotne jest to, że redukowanie liczby punktów dyskretyzacji nawet o połowę w obu strategiach *nie przynosi istotnego pogorszenia wskaźnika jakości* To upoważnia do śmiałego ograniczania liczby punktów, przy czym dla strategii *B* najlepszy wynik został osiągnięty przy ich liczbie równej 8; dla strategii *A* rezultaty są gorsze niż dla *B*, ale i tu nastąpiła poprawa (9÷4 punkty dyskretyzacji). Warto też przeanalizować wpływ obu strategii na czasy optymalizacji.

Powyższy rysunek (22) obrazuje ich zależność od strategii dyskretyzacji. Wyraźnie, dla małej liczby punktów, zaznacza się tendencja spadkowa; zdarzają się, niestety, wyjątki (dwa piki na pierwszej, błękitnej wstędze). Również one są wynikiem przeciągającego się całkowania; przyczyn tego nie potrafiłem zlokalizować.



Rys. 23 Wahadło — rozkład czasów optymalizacji (w sekundach)

Generalnie jednak, im mniej punktów, tym sprawniejsza optymalizacja i bardziej wyrównane czasy obliczeń. Oto porównanie (23) rozkładu czasów optymalizacji dla wyjściowych parametrów optymalizacji (z lewej) i dla optymalizacji z trzema punktami przy znanych zakłóceniach (z prawej): rzuca się w oczy większa jednostajność tego drugiego.

Wnioski. Analizowany przykład sugeruje, że zmniejszanie liczby punktów dyskretyzacji sterowania na przedziale lokalnym według strategii *B* rokuje najpomyślniej. Występuje, co prawda, konflikt dwóch kryteriów: z jednej strony rozrzedzanie dyskretyzacji to krótsze czasy obliczeń; z drugiej — gdy jest ono nadmierne — pogorszenie się wskaźnika jakości. Dlatego najlepsze rezultaty daje usunięcie około 20÷30□ punktów. Pozostaje zbadać na innych przykładach na ile konkluzje te są prawdziwe.

4.4.2 Model wahadła przy gęstej dyskretyzacji sterowania

Uważałem za celowe sprawdzenie, jak powyższe wnioski będą się miały do wyników sterowania tym samym modelem przy 10-krotnie krótszym etapie sterowania. Oznacza to, że przedział globalny ma 300 etapów a lokalny — 100. Pozostałe warunki symulacji takie, jak poprzednio. Próby symulacji z takimi parametrami spełzły w większości przypadków na niczym. A to za przyczyną przewlekającego się całkowania. Na pewnych fragmentach symulacji nawet zwiększenie limitu kroków całkowania nie przynosiło poprawy. Oznacza to, że dobór algorytmu nie był trafny albo jego implementacja — niepoprawna.

Obliczenia „ruszyły” po pięciokrotnym (wartości pośrednie nie sprawdzane) zmniejszeniu dokładności całkowania. Obecnie $\varepsilon = 5 \cdot 10^{-5}$.

Wariant	wskaźnik jakości przy znajomości zakłóceń	
	pełnej	niepełnej
wyjściowy, bez kary za stan w horyzoncie lokalnym	0,00751	0,0869
kara za stan $\mathbf{Q} = x_1^2$	0,0198	0,134
kara za stan $\mathbf{Q} = 3x_1^2$	0,113	0,151

Tabela 4 Wahadło (krótkie etapy) — wpływ □

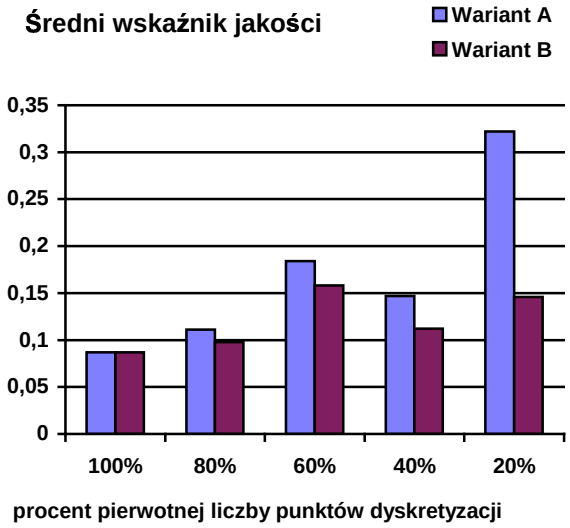
Wskaźnik jakości. Wprowadzanie kary za stan na horyzoncie lokalnym w żadnym z wariantów nie spowodowało poprawy jakości sterowania (4). Wniosek to mało optymistyczny, ale w zasadzie zgodny z wcześniejszymi. Nieco odmienne spostrzeżenia nasuwają się w przypadku rozrzedzania lokalnej dyskretyzacji sterowania. Tym razem wariant wyjściowy okazał się, pod względem wskaźnika jakości, nie do pobicia (5). Zmniejszenie liczby punktów już o 20□ spowodowało dwu- i trzykrotne pogorszenie sterowania zarówno dla preferowanego przeze mnie wariantu *B*, jak i *A*.

Wariant	wskaźnik jakości dla rozrzedzania metodą	
	A	*B*
wyjściowy, przy 100□ punktów dyskretyzacji	0,00751	
80□ punktów dyskretyzacji	0,0256	0,0149
60□ punktów dyskretyzacji	0,0245	0,0397
40□ punktów dyskretyzacji	0,263	0,0377
20□ punktów dyskretyzacji	0,316	0,166

Tabela 5 Wahadło (krótkie etapy) — rozrzedzanie dyskretyzacji

Wciąż rozrzedzanie metodą *B* daje lepsze wyniki. Przy 80□ pierwotnej liczby punktów daje znacznie lepsze sterowanie a przy 40□ — deklasuje *A* o rząd wielkości.

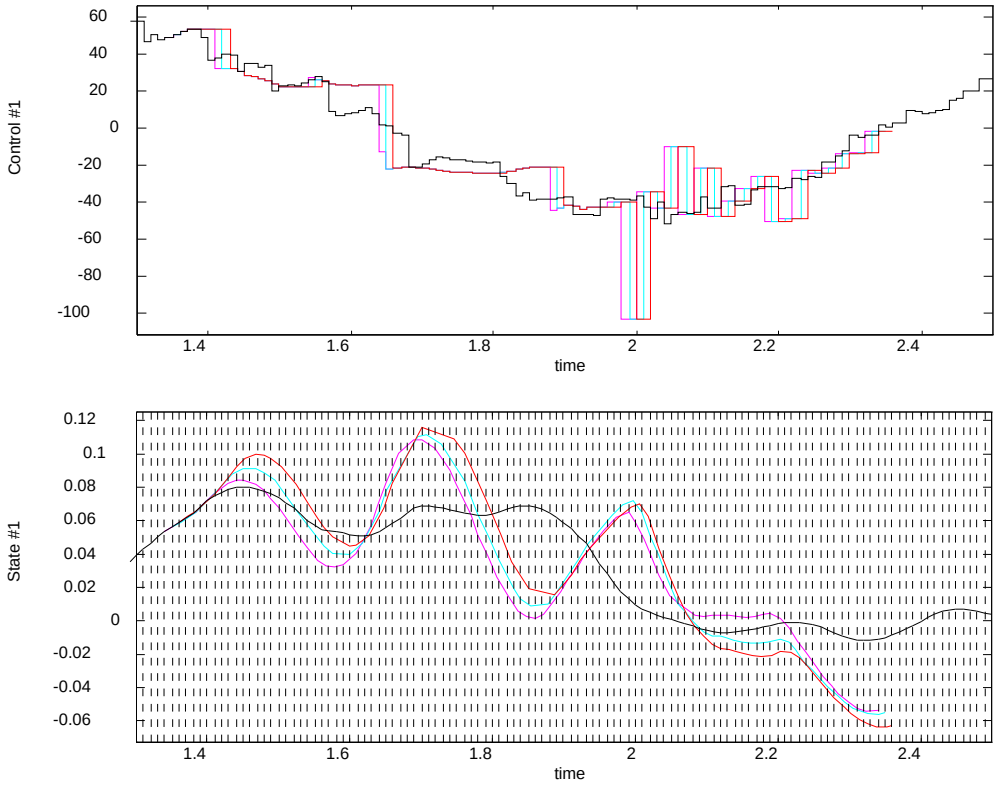
Dużo mniejsze dysproporcje w jakości sterowania występują przy niepełnej znajomości zakłóceń (24). Potwierdza to tezę, że sterowanie z powtarzaną optymalizacją jest wrażliwe na rzetelność prognozy $\boldsymbol{z}$, że wszelkie zabiegi zawodzą, gdy jest ona błędna.



Rys. 24 Wahadło (krótkie etapy, niepełna znajomość zakłóceń) — rozrzedzanie dyskretyzacji

Okazuje się, że nawet pięciokrotne rozrzedzenie dyskretyzacji według strategii B nie powoduje radykalnego pogorszenia jakości — w przeciwieństwie do strategii A. Warto tutaj przypomnieć, że wskaźnik jakości dla dziesięciu punktów dyskretyzacji w poprzednim podrozdziale był równy 0,295. Dziesięciokrotne skrócenie etapu sterowania zmniejszyło go do 0,0869. Oznacza to, że najlepszym sposobem poprawienia sterowania jest skrócenie pojedynczego etapu. Niestety, nie zawsze jest to możliwe.

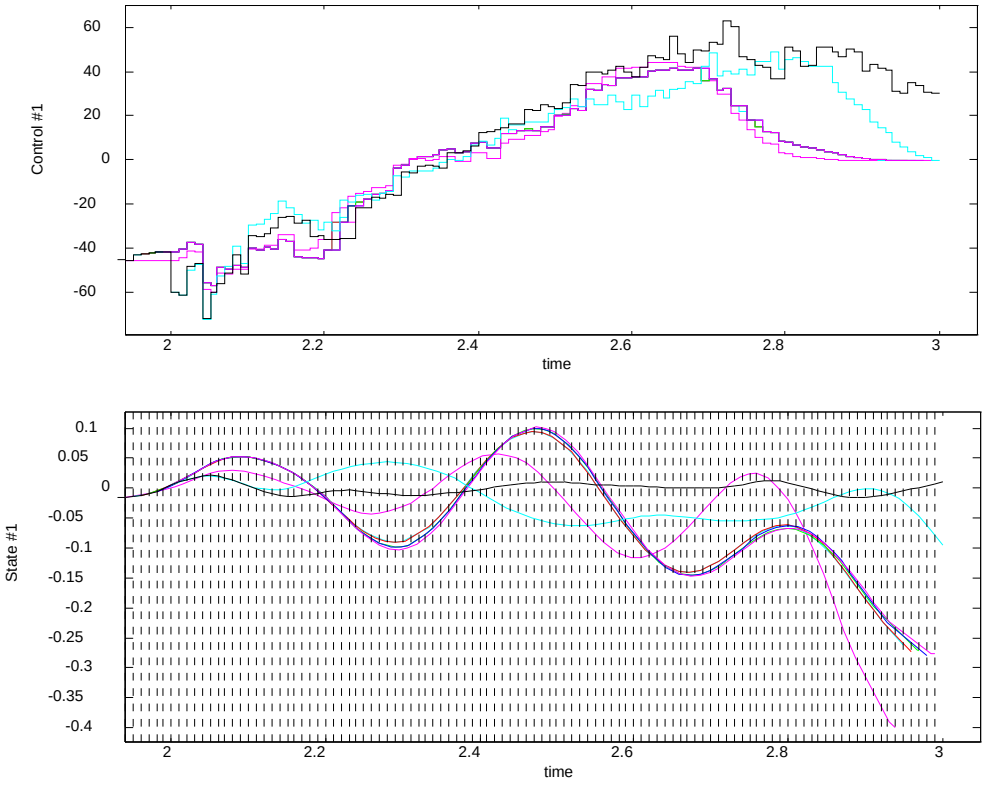
Przebiegi sterujące. Można doszukiwać się przyczyn niezgodności wyników z oczekiwaniami. Faktem jest, że teza o korzystnym wpływie rzadszej dyskretyzacji w tym przypadku się nie sprawdziła. Przeanalizowanie przebiegów sterujących i trajektorii zmiennych stanu (25) mogłoby nieco wyjaśnić sytuację.



Rys. 25 Wahadło (krótkie etapy, rozrzedzona dyskretyzacja) — przebiegi wybranych zmiennych stanu i sterowań

Powyższy rysunek przedstawia część zrealizowanej i kilka przewidywanych trajektorii sterowania dla przypadku z pełną znajomością zakłóceń i liczbą punktów dyskretyzacji zredukowaną do 80□. Poniżej pokazane zostały przewidywane przebiegi zmiennych stanu. Okazuje się, że przewidywane sterowania (a więc i stany) mogą znacznie odbiegać od rzeczywistych. Szczególnie niepokojące są duże wahania sterowania przewidywane pod koniec przedziałów lokalnych, a nigdy nie zrealizowane. Tego rodzaju sytuacje nie mają miejsca przy 100□ punktów dyskretyzacji sterowania. Istnieje podejrzenie, że takie zachowanie algorytmu optymalizacji w istotny sposób wpływa na początkowy fragment sterowania, który zostanie zrealizowany.

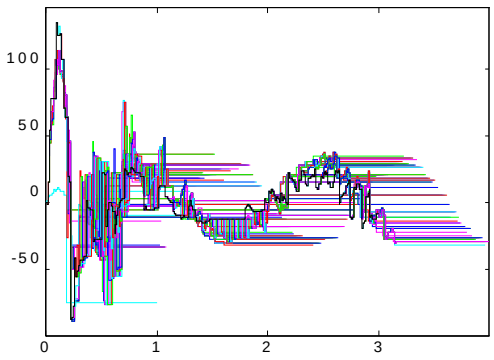
Przewidywane sterowania są dla wersji pierwotnej (100□) zgoła odmienne (26).

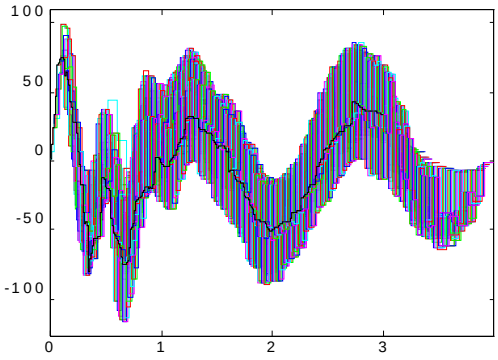


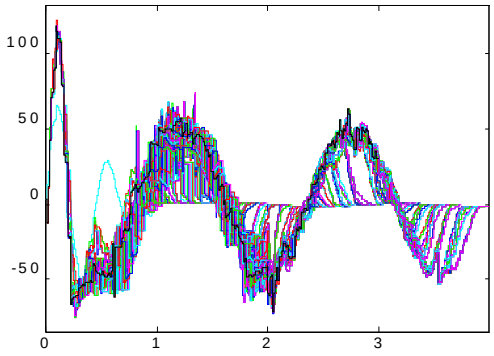
Rys. 26 Wahadło (krótkie etapy, pierwotna dyskretyzacja) — przebiegi wybranych zmiennych stanu i sterowań

W tym przypadku lokalne trajektorie sterowania nie kończą się gwałtownymi oscylacjami ale powoli opadają do zera. Sterowanie zanika. Zmienne stanu zaczynają, pod wpływem zakłóceń, odbiegać od trajektorii zadanej. Słowem: zaniedbania w sterowaniu blisko horyzontu lokalnego skutkują karami za stan — czyli wszystko zgadza się z przypuszczeniami z poprzedniego podrozdziału. Z dokładnością do skutków. Bowiem dla tej serii symulacji żadne manipulacje dyskretyzacją sterowania nie dają poprawy.

Ogólnie jednak we wszystkich zbadanych przypadkach (27) algorytm układ zachowuje stabilność a przewidywane sterowania w zasadzie przypominają kształtem przebiegi rzeczywiste.







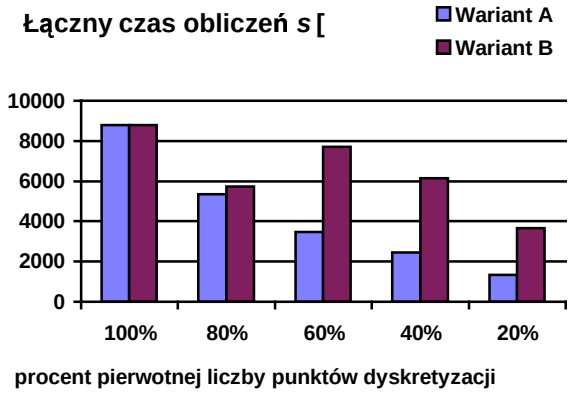
Sterowania przewidywane na wszystkich etapach (kolorowe) i zrealizowane (czarne) dla wersji (kolejno):

- ze zredukowaną do 20□ liczbą punktów dyskretyzacji sterowania — wersja *A*
- ze zredukowaną do 20□ liczbą punktów dyskretyzacji sterowania — wersja *B*.
- oryginalnej, z zachowanymi wszystkimi punktami dyskretyzacji sterowania.

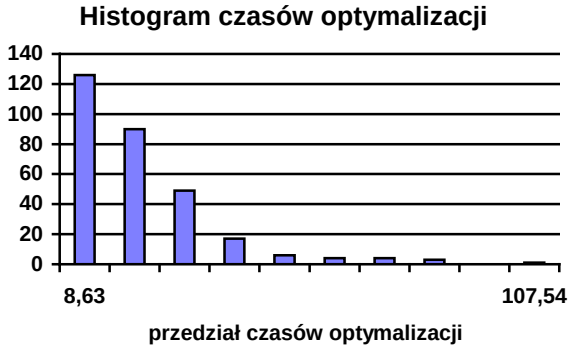
Rys. 27 Wahadło — zestawienie sterowań dla różnych dyskretyzacji

Czasy obliczeń. Sesje sterowania omawiane w tym podrozdziale trwały o wiele dłużej niż ich odpowiedniki w poprzednim. Na przykład łączny czas obliczeń dla wersji podstawowej wynosił uprzednio $713s$ (pełna informacja o zakłóceniach) i $803s$ (informacja niepełna). Obecnie jest on równy odpowiednio $6272s$ i $8782s$, czyli około 10 razy więcej. Owo wydłużenie nie wynika z przeciągającej się procedury całkowania: ta wykonuje się tak samo a nawet szybciej, biorąc pod uwagę zmniejszone wymagania na dokładność. Przyczyną jest większa liczba kroków optymalizacji wynikająca z gęściejszej dyskretyzacji. Przy niezbyt dużej dokładności całkowania obliczenia nie są precyzyjne, co może doprowadzić (i parokrotnie, dla innych modeli, doprowadziło) do sytuacji, gdy moduł optymalizujący wyznacza kolejne, odmienne trajektorie sterowania, które dają taki sam wskaźnik jakości. Policzony następnie gradient jest zbyt duży, by przerwać obliczenia — w ten sposób optymalizator generuje w pętli nieskończonej kolejne, równie nieudane, trajektorie sterowania, nie mogąc osiągnąć stanu, gdy co najmniej jeden z testów stopu byłby spełniony.

O ile trendy dla symulacji przy znanych w pełni zakłóceniach nie są wyraziste, to w wersji drugiej (nieznane zakłócenia) potwierdziły się wyniki z podrozdziału poprzedniego (28).



Rys. 28 Wahadło (zakłócenia nie w pełni znane) — rozrzedzanie

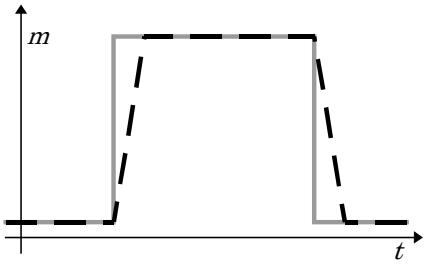


Rys. 29 Wahadło (zakłócenia nie w pełni znane) — histogram czasów obliczeń

Ponadto, uprzednio o łącznym czasie obliczeń przesądzał jeden bądź kilka feralnych etapów, dla których zoptymalizowanie sterowania trwało grubo ponad średnią. Dopiero przy zredukowanej liczbie punktów dyskretyzacji anomalie te znikały.

Tutaj czasy symulacji są bardziej wyrównane i niezależne od dyskretyzacji sterowania (29). Być może można to przypisać liberalnemu podejściu do całkowania, które nie „potyka się" już tak na sokowych zmianach sterowania.

Wnioski. Dokonane porównanie symulacji sterowania przy znacznie różniących się długościach pojedynczego etapu przynosi kilka ważkich wniosków. Widać jasno, że najskuteczniejszym sposobem poprawy jakości sterowania jest... częstsze sterowanie, zwłaszcza przy niewielkiej wiedzy o zakłóceniach. Okazuje się też, że rozrzedzenie dyskretyzacji na przedziale lokalnym może, ale wcale nie musi prowadzić do poprawy sterowania. Sprawdzają się natomiast spostrzeżenia o niewielkiej wadze takiego czynnika, jak kara za stan w horyzoncie lokalnym. Ponadto, dalsze rozrzedzanie dyskretyzacji prowadzi w końcu do pogorszenia się sterowania, przy czym dysproporcje są mniejsze przy niepełnej wiedzy o zakłóceniach. Redukcja liczby punktów dyskretyzacji według strategii B pozwala na utrzymanie przyzwoitego wskaźnika jakości przy mniejszej liczbie punktów, niż A.

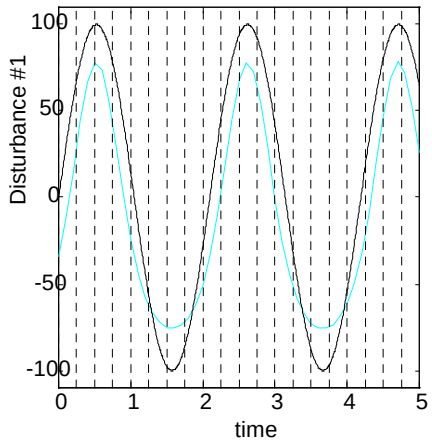


Rys. 30 Przybliżenie przebiegu schodkowego

Ta seria symulacji pokazała też słabości zaimplementowanego algorytmu. Czasy optymalizacji są długie a ponadto nie można dostatecznie zwiększyć dokładności całkowania — wówczas algorytm całkujący nie radzi sobie ze skokową naturą sterowania. □rodkiem zaradczym, stosowanym w praktyce, byłoby przybliżenie przebiegu schodkowego — trapezowym (30). Jest to rozwiązanie całkowicie uzasadnione, bowiem elementy wykonawcze nigdy nie działają natychmiastowo.

4.4.3 Model suwnicy

Warto zweryfikować spostrzeżenia z dwóch ostatnich podrozdziałów badając model trzeci: suwnicy. Niech symulacja składa się z 50 etapów o długości 0,25*s*, co daje horyzont globalny równy 12,5*s*. Lokalny przedział ma 20 punktów sterowania, równo rozmieszczonych, czyli jego długość jest równa 5*s*. Zakłócenia sumują się ze sterowaniami i mają przebieg $z(t) = 100\,\mathbf{sin}\,3t$. Przy niepełnej znajomości zakłóceń regulator dysponuje ich prognozą w postaci $z^*(t) = -100 + \mathbf{exp}(\mathbf{sin}\,3t + 4{,}2)$. Porównanie obu przebiegów z poniżej (31).

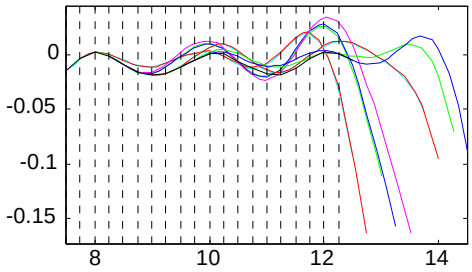


Rys. 31 Suwnica — zakłócenia faktyczne i przewidywane

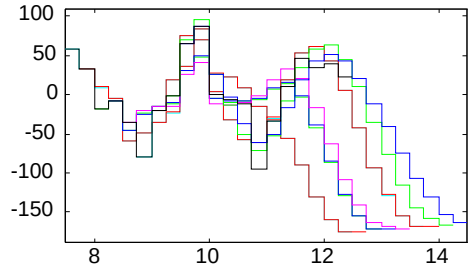
Ponadto przyjmuje się, że składowa przejściowa zmiennych ustaje począwszy od 30. etapu i dlatego średni wskaźnik jakości będzie liczony od tej właśnie chwili.

Jakość — wersja wyjściowa. Już symulacje przy 100□ punktów dyskretyzacji sterowania dały zaskakujące rezultaty. Oto wskaźnik jakości przy niepełnej znajomości zakłóceń okazał się lepszy niż gdy są one znane do końca. Natomiast wskaźniki jakości liczone od początku sesji sterowania układają się według przewidywań, tzn. sterowanie przy nie w pełni znanych zakłóceniach daje gorsze rezultaty. Poniżej (6) przedstawione są przykładowe wykresy zmiennych stanu, sterowań i zakłóceń wraz z komentarzami.

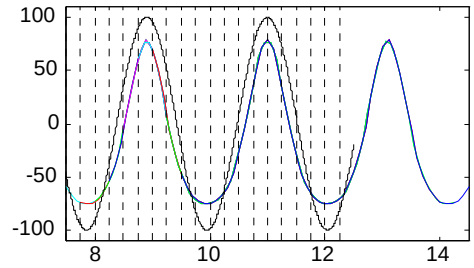
Sterowanie przy zakłóceniach w pełni znanych



Fragment trajektorii zmiennej stanu nr 1. dla sterowania przy znanych zakłóceniach. W początkowej części lokalnego przedziału sterowania trajektorie przewidywane i zrealizowane są zgodne. Później wszystkie trajektorie przewidywane opadają, bez względu na wartość zakłócenia. Cząstkowy średni wskaźnik jakości dla tego przypadku wynosi 9,2.

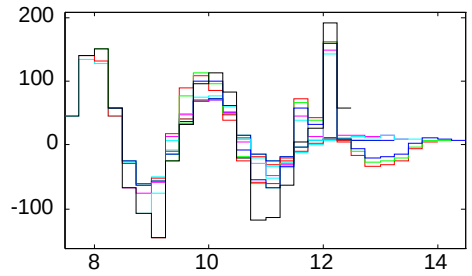


Zrealizowany i przewidywane przebiegi sterujące dla przypadku opisanego powyżej. Charakterystyczny, bezzasadny spadek sterowań blisko horyzontu lokalnego, i to nie do zera, ale pewnej ujemnej wartości. Początkowe fragmenty przewidywanych sterowań działają tak, by zneutralizować przebieg zakłócający, natomiast kształt fragmentów końcowych nie jest racjonalny.

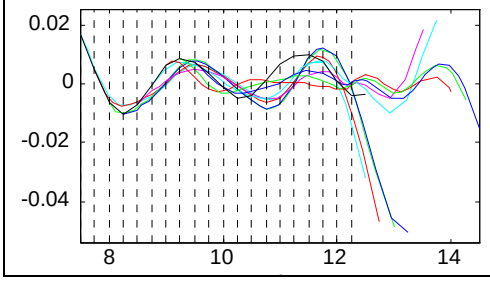


Trajektorie zakłóceń. Kolorem czarnym pokazano rzeczywisty ich przebieg. Innymi kolorami — prognozę, jaką widzi optymalizator przy niepełnej wiedzy o zakłóceniach. Dla przypadku opisanego powyżej istotna jest tylko trajektoria czarna; dla tego poniżej — wszystkie.

Sterowanie przy niepełnej wiedzy o zakłóceniach



Przebiegi sterowań przy nie znanych w pełni zakłóceniach. Ich amplituda jest większa niż w poprzednim przypadku, ale nie przeszkadza to w uzyskaniu lepszego wskaźnika jakości. Przewidywane sterowania dużo lepiej pokrywają się ze zrealizowanymi.

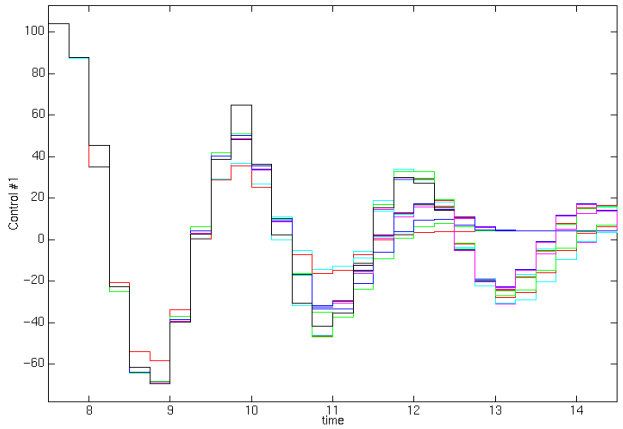


Rzeczywista i planowane trajektorie stanu przy nie w pełni znanych zakłóceniach. Tylko część z nich rozbiega się, w dodatku w różnych kierunkach (por. przykład poprzedni). Cząstkowy średni wskaźnik jakości dla tego przypadku jest równy 1,37 — prawie siedmiokrotnie mniej niż przy znanych zakłóceniach.

Tabela 6 Suwnica — stany i sterowania przy pełnej i niepełnej znajomości zakłóceń

Zachowanie się układu sterowania w pierwszym przypadku jest niepokojące. Sterowania blisko końca przedziału lokalnego przyjmują pewną niezerową wartość, co nie może być tłumaczone ich małym oddziaływa-

niem na J_i^*. Ta anomalia może stanowić pierwszy krok do odsłonięcia jakiejś reguły, może jednak wynikać z pospolitych błędów obliczeniowych. Dlatego dokonano takich samych symulacji na *Sunie* —niestety, ich wyniki wskazują ta tę drugą ewentualność.



Rys. 32 Suwnica — sterowanie uzyskane na platformie *Sun*

Otóż okazało się, że sterowania uzyskane tam dla przypadku z pełną znajomością zakłóceń nie opadają (32) a uzyskany wskaźnik jakości jest równy 0,842. (Wskaźnik jakości przy niepełnej znajomości zakłóceń wyniósł 0,957.) A zatem polepszenie precyzji obliczeń o rząd wielkości (odmienna reprezentacja liczb zmiennoprzecinkowych) może w pewnych przypadkach całkowicie zmienić znalezione rozwiązanie. Wniosek jest oczywisty — trzeba zwiększyć dokładność obliczeń — ale zastosowane procedury *są* rekomendowane przez autora □2□ jako uniwersalne i sprawdzone. Niestety — w *tym* problemie — nie zawsze.

Jakość — rzadsza dyskretyzacja. □amiąc zasadę jednakowych warunków symulacji, przedstawiono poniżej skutki rozrzedzania dyskretyzacji uzyskane na *Sunie* (7). Zakłócenia nie są w pełni znane.

Wariant	wskaźnik jakości dla rozrzedzania metodą	
	A	*B*
wyjściowy, przy 100□ punktów dyskretyzacji	0,975	
80□ punktów dyskretyzacji	1,51	1,30
60□ punktów dyskretyzacji	2,34	289
40□ punktów dyskretyzacji	1,96	186

Tabela 7 Suwnica — rozrzedzanie dyskretyzacji

□adna ze strategii nie dała poprawy wskaźnika jakości sterowania. Tym razem lepszą okazała się *A*, pozwalająca utrzymać wskaźnik jakości tego samego rzędu, co oryginalny, po usunięciu aż 80□ punktów dyskretyzacji.

4.4.4 Wnioski dotyczące dyskretyzacji sterowania

Powyższe badania dowodzą, że jest możliwe poprawienie jakości sterowania bez naruszania zadanej długości pojedynczego etapu. Udało się dokonać tego wprowadzając funkcję kary za stan w horyzoncie lokalnym; druga metoda polega na zmniejszeniu gęstości dyskretyzacji sterowania na horyzoncie lokalnym (2 strategie). Niestety, nie można wypracować

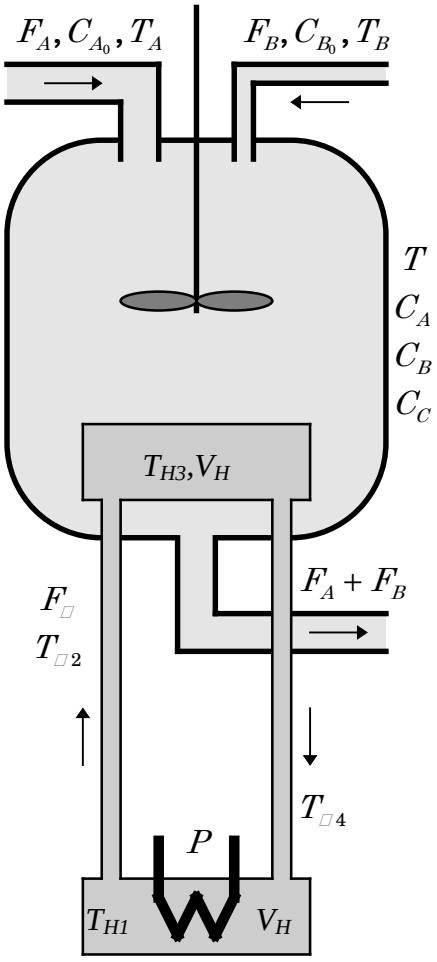
uniwersalnej receptury poprawiającej sterowanie. Wybór metody, a później skala modyfikacji warunków początkowych, zależą od konkretnego przypadku. Generalnie, zmniejszenie liczby lokalnych punktów dyskretyzacji skutkowało częściej niż wprowadzanie dodatkowej kary. Ponadto, starania o poprawę jakości sterowania drogą rozrzedzania lokalnej dyskretyzacji dają lepsze rezultaty przy trafnych prognozach zakłóceń.

Dokonane symulacje odsłoniły wady zaimplementowanych algorytmów. Schodkowy charakter przebiegów sterujących powodował częste grzęźnięcie programu w fazie całkowania równań stanu. Postuluje się więc aproksymowanie tych przebiegów trapezami. Ponadto, dokładność numeryczna *Intela* jest niekiedy zbyt mała i algorytm domaga się przeróbek, które umożliwiłyby dorównanie precyzją obliczeń platformie *Sun*.

Zdaję sobie sprawę, że liczba przebadanych przeze mnie modeli i konfiguracji parametrów sterowania jest niewielka. Pozwala jednak uświadomić sobie, że czasami zmiana lokalnej dyskretyzacji sterowania może się sowicie opłacić. Dlatego zachęcam do niepomijania tej fazy procedury projektowej. W ostatecznym rozrachunku powiększa ona wiedzę o obiekcie i, być może, pozwoli z czasem na wypracowanie ściślejszych reguł dyskretyzacji na lokalnym horyzoncie sterowania.

4.5 Przykład

Wnioski zwarte w tym rozdziale można złożyć w regułę projektową, która dla konkretnego obiektu doprowadzi do dobrania najlepszych parametrów algorytmu sterowania repetycyjnego. Jest to pierwsza próba określenia ścieżki postępowania w celu uzyskania optymalnych wartości parametrów: z uwagi na moje skąpe doświadczenie i niewielką liczbę przebadanych modeli ścieżka ta może się zmienić.



Rys. 33 Model reaktora

Niech przedmiotem sterowania będzie reaktor chemiczny o objętości V (33). Do reaktora doprowadzane są rurami dwa ciekłe składniki. Z lewej strony z szybkością F_A dopływa ciecz o temperaturze T_A i stężeniu czynnika A równym C_{A_0}. Rura prawa doprowadza z szybkością F_B czynnik B o temperaturze T_B. Jego stężenie jest równe C_{B_0}.

Wewnątrz reaktora następuje idealne mieszanie obu składników. Nie dochodzi do wymiany ciepła z otoczeniem. Panuje tam temperatura T a stężenie substancji A, B i C jest równe odpowiednio C_A, C_B i C_C. Pod wpływem temperatury i obecności składnika B dochodzi do zamiany składnika A w produkt finalny C z szybkością opisaną przez zależność $\mathbf{k}(C_A, C_B, T)$. Aby utrzymać odpowiednią temperaturę, reaktor jest podgrzewany cieczą przepływającą przez wymiennik ciepła. Następuje w nim zrównanie temperatur cieczy grzewczej i cieczy w reaktorze ($T_{□3} = T$). Reakcja $A \xrightarrow{B,T} C$ nie wymaga dostarczania ciepła z zewnątrz.

Układ ogrzewania składa się ze wspomnianego idealnego wymiennika ciepła oraz komory grzejnej w której, zanurzona w cieczy, znajduje się grzałka dostarczająca moc P. Oba elementy są połączone rurami. Prędkość przepływu czynnika grzewczego jest równa $F_□$. Także tutaj nie dochodzi do wymiany ciepła z otoczeniem.

Z uwagi na to , że program symulacyjny nie pozwala na definiowanie opóźnień w propagacji ciepła, jakie mają miejsce w obu rurach, musiałem dokonać uproszczenia modelu. Rury są przybliżane serią 10 małych zbiorników; temperatura w każdym z nich jest w jego całej objętości jednakowa. Powyższe ustalenia prowadzą to następującego układu równań stanu:

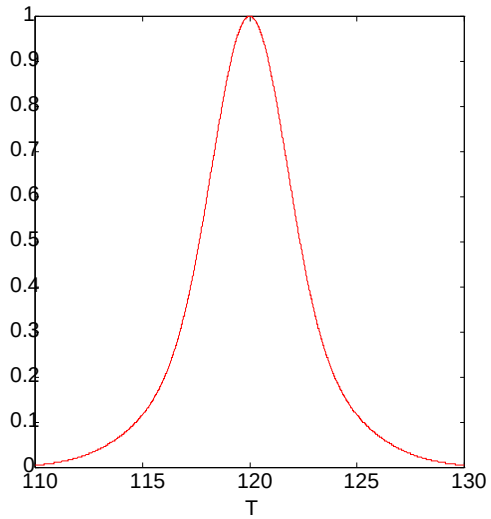
$$\dot{C}_C = C_A \mathbf{k}(C_A, C_B, T) - C_C(F_A + F_B) \square V \qquad C_C(0) = 0{,}15$$
$$\dot{C}_A = C_{A_0} F_A \square V - C_A \mathbf{k}(C_A, C_B, T) - C_A(F_A + F_B) \square V \qquad C_A(0) = 0{,}05$$
$$\dot{C}_B = \left(C_{B_0} F_B - C_B(F_A + F_B)\right) \square V \qquad C_B(0) = 0{,}025$$
$$\dot{T} = \left(T_A F_A + T_B F_B - T(F_A + F_B) + T_{\square 2} F_{\square} - T F_{\square}\right) \square (V + V_{\square}) \qquad T(0) = 120$$
$$\dot{T}_{\square 1} = \left(T_{\square 4e} F_{\square} - T_{\square 1} F_{\square} + P \square \square\right) \square V_{\square} \qquad T_{\square 1}(0) = 121{,}2$$
$$\dot{T}_{\square 1a} = \left(T_{\square 1} F_{\square} - T_{\square 1a} F_{\square}\right) \square V_{\square} \qquad T_{\square 1a}(0) = 121{,}2$$
$$\dot{T}_{\square 1b} = \left(T_{\square 1a} F_{\square} - T_{\square 1b} F_{\square}\right) \square V_{\square} \qquad T_{\square 1b}(0) = 121{,}2$$
$$\dots$$
$$\dot{T}_{\square 1e} = \left(T_{\square 1d} F_{\square} - T_{\square 1e} F_{\square}\right) \square V_{\square} \qquad T_{\square 1e}(0) = 121{,}2$$
$$\dot{T}_{\square 2} = \left(T_{\square 1e} F_{\square} - T_{\square 2} F_{\square}\right) \square V_{\square} \qquad T_{\square 2}(0) = 121{,}2$$
$$\dot{T}_{\square 4} = \left(T F_{\square} - T_{\square 4} F_{\square}\right) \square V_{\square} \qquad T_{\square 4}(0) = 120$$
$$\dot{T}_{\square 4a} = \left(T_{\square 4} F_{\square} - T_{\square 4a} F_{\square}\right) \square V_{\square} \qquad T_{\square 4a}(0) = 120$$
$$\dot{T}_{\square 4b} = \left(T_{\square 4a} F_{\square} - T_{\square 4b} F_{\square}\right) \square V_{\square} \qquad T_{\square 4b}(0) = 120$$
$$\dots$$
$$\dot{T}_{\square 4e} = \left(T_{\square 4d} F_{\square} - T_{\square 4e} F_{\square}\right) \square V_{\square} \qquad T_{\square 4e}(0) = 120$$

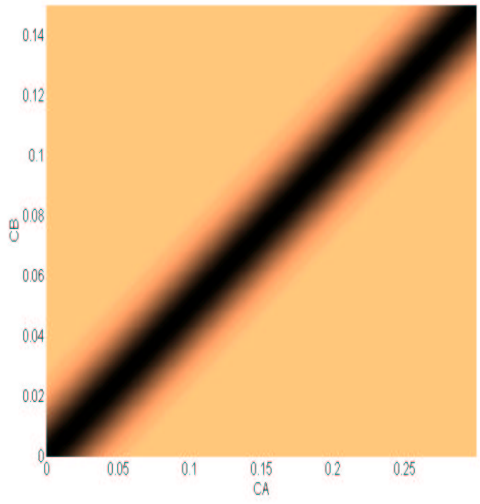
Stała □ to ciepło właściwe czynnika grzewczego i wszystkich składników reakcji; $V_{\square}$ to masa cieczy grzewczej w wymienniku, w komorze grzewczej i we wszystkich zbiornikach modelujących rury doprowadzające ciepło.

Funkcja $\mathbf{k}(C_A, C_B, T)$ wyraża się wzorem:

$$\left(0{,}7 \cdot 6^{-0{,}1(T-120)^2} + 0{,}3 \cdot 1{,}5^{-0{,}1(T-120)^2}\right) \cdot e^{-1000(C_A - 2C_B)^2}.$$

Wynika stąd, że szybkość reakcji jest największa przy temperaturze w reaktorze równej 120°C i przy stężeniu czynnika B dokładnie dwa razy mniejszym niż czynnika A. Poniżej (34) zostały przedstawione orientacyjne wykresy zależności dla obu czynników w $\mathbf{k}(C_A, C_B, T)$.





Rys. 34 Orientacyjna zależność **k** od C_A, C_B **i** T □°C□.

W omawianym przykładzie przyjąłem, że F_A, C_A i T_{A_0} są wielkościami niesterowanymi a P i F_B — sterującymi. Pozostałe wielkości nie będące zmiennymi stanu są stałe i wynoszą: $V = 50[kg]$, $V_{\square} = 20[kg]$, $F_{\square} = 5[kg \square s]$, $T_B = 20[^\circ C]$, $C_{B_0} = 0{,}9$, $\square = 5000[J/kg \cdot ^\circ C]$. Dla utrzymania tego samego rzędu wielkości zmiennych sterujących wyrażam P w megawatach.

Po przedstawieniu modelu można przystąpić do syntezy wszystkich wysnutych dotychczas wniosków w ciąg zaleceń projektowych prowadzących do wyboru najkorzystniejszych wartości parametrów sterowania repetycyjnego. Każde z zaleceń zostanie zastosowane wobec modelu reaktora.

1. **Określenie dokładności całkowania** ε.

Należy dokonać symulacji sterowania obiektu i przeanalizować przebiegi zmiennych stanu. Podczas symulacji zakłócenia powinny być tak dobrane, by swoim charakterem odpowiadały późniejszym warunkom roboczym, oraz by powodowały zmiany wszystkich zmiennych stanu, co pozwoli oszacować dynamikę układu.

Należy, poczynając od standardowej wartości $\varepsilon = 10^{-5}$, zmniejszać je w kolejnych krokach 10-krotnie, notując czasy obliczeń i uwagi o kształcie przebiegu. Należy dokonywać tego aż osiągnie się najdokładniejszy możliwy do uzyskania, przebieg.

Należy, poczynając od standardowej wartości ε, zwiększać je w kolejnych krokach 10-krotnie. Wyniki trzeba porównywać z najdokładniejszym uzyskanym przebiegiem. Należy zatrzymać się wówczas, gdy uzyskany przebieg swoim charakterem będzie odbiegał od najdokładniejszego w stopniu nie do zaakceptowania.

Należy przyjąć ε o 2 lub 3 rzędy wielkości lepsze (mniejsze) od tego, które daje ów niedopuszczalny wynik. Możliwe jest przyjęcie innego (jeszcze mniejszego) ε o ile czasy obliczeń dla niego nie są znacząco dłuższe od ε zalecanego standardowo.

Ad 1. Program do badania dynamiki (`simsys.exe`) wymaga stałych wartości sterujących podczas symulacji. Niech więc $F_B = 0{,}06[kg/s]$ a $P = 0{,}0275[MW]$. Zakłócenia są zrealizowane w postaci trzech, następujących kolejno po sobie skoków od wartości ustalonych. Wartości ustalone to $F_A = 2[kg/s]$, $T_A = 120[^\circ C]$ i $C_{A_0} = 0{,}2$. Dla takich zakłóceń i sterowań obiekt jest w stanie stabilnym. Na przedziale symulacji $\langle 0{,}1500 \rangle$ są trzy takie skoki: dla $t \in \langle 200{,}300 \rangle$ $F_A = 2{,}3$, dla $t \in \langle 500{,}600 \rangle$ $C_{A_0} = 0{,}25$ i dla $t \in \langle 800{,}900 \rangle$ $T_A = 115$.

Stopniowe zwiększanie dokładności całkowania powoduje wzrost czasów obliczeń, a dla $\varepsilon = 10^{-8}$ zostaje przekroczona dopuszczalna liczba iteracji procedury całkującej. Można więc uznać, że przebieg zmiennych stanu uzyskany dla $\varepsilon = 10^{-7}$ to najlepszy z możliwych.

Stopniowe zmniejszanie dokładności całkowania powoduje nieznaczne zmniejszenie się czasów obliczeń. Dla $\varepsilon = 10^{-2}$ na przebiegu zmiennej stanu T pojawiają się znaczne oscylacje, obserwowane także w mniejszym stopniu przy $\varepsilon = 10^{-3}$, ale nie w żadnym innym przypadku. Można więc uznać, że wynik dla $\varepsilon = 10^{-2}$ jest już niedopuszczalny. Zatem $\varepsilon = 10^{-4}$ (100 razy większe) jest dobrą wartością tym bardziej, że redukuje czas obliczeń o 30□ w stosunku do $\varepsilon = 10^{-7}$. (Niestety, poprzednie symulacje pokazały, że niekiedy trzeba i tę dokładność obniżyć, nie ma jednak powodu, by robić to teraz). Z taką wartością ε przechodzimy do kroku następnego.

2. Określenie minimalnego rejestrowanego kroku δx.

Używając środowiska z punktu 1. należy zwiększać δx (standardowo 0,01) aż zaczną pojawiać się rozbieżności między trajektorią dokładną a bieżącą. Najlepszym sygnałem zakłócającym przy wyznaczaniu δx jest przebieg schodkowy. Należy przyjąć δx około 100 razy większe od tego, które daje przebieg nie do zaakceptowania z uwagi na nierejestrowanie wszystkich szczegółów trajektorii.

Można też postąpić w sposób uproszczony i przyjąć za δx około $1/100$ najmniejszej stałej czasowej w modelu.

Ad 2. Szacowania poszczególnych stałych czasowych modelu dla przyjętych warunków początkowych, sterowań i zakłóceń wykazują, że najmniejsza stała czasowa jest rzędu dziesiątek sekund. Pozwala to przyjąć $\delta x = 0{,}1$. Dla takiej wartości została przeprowadzona symulacja a uzyskane wyniki różniły się od oryginalnych ($\delta x = 0{,}01$) w sposób znikomy.

3. Określenie skali wskaźnika jakości (mnożnika przy $\mathbf{f}_0$).

Dobierając parametry optymalizacji należy mieć na uwadze dwa kryteria jakości: właściwy wskaźnik jakości oraz czas potrzebny do jego uzyskania. Należy przeprowadzić serię optymalizacji sterowania przy zmieniającym się mnożniku przy $\mathbf{f}_0$. Warunki optymalizacji powinny być maksymalnie zbliżone do rzeczywistych: zakłócenia powinny mieć taką samą naturę i rząd wielkości; długość pojedynczego etapu sterowania powinna być identyczna jak w rzeczywistości, długość lokalnego przedziału sterowania powinna być tak duża, by uchwycić całą dynamikę obiektu.

Należy dokonać kilku optymalizacji sterowania. W każdej z nich należy zmieniać mnożnik przy $\mathbf{f}_0$ i notować uzyskane rezultaty: wskaźnik jakości, czas optymalizacji oraz cechy charakterystyczne uzyskanej trajektorii. Należy wybrać taki mnożnik, który gwarantuje kompromis

między dobrym a szybkim sterowaniem.

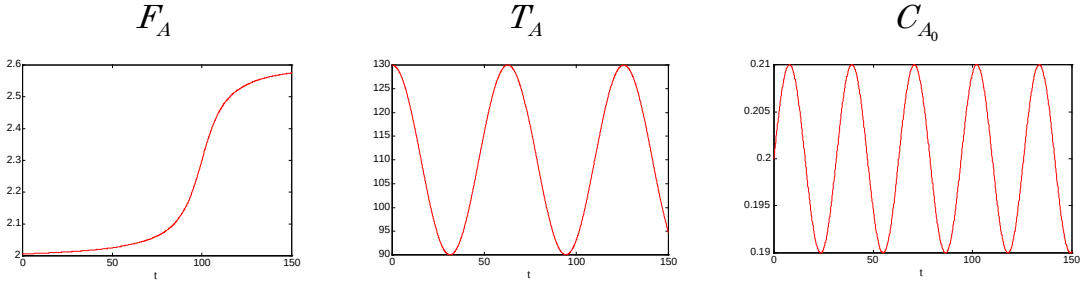
Gdyby manipulowanie skalą $\mathbf{f}_0$ nie przynosiło zadowalających rezultatów (optymalizacja kończyła się po pierwszym kroku lub dopuszczalna liczba iteracji była notorycznie przekraczana), wówczas należy dobrać tolerancję c według zalecenia 4. i powrócić do doboru skali $\mathbf{f}_0$.

Ad 3. Do optymalizacji sterowania na jednym etapie posłużyłem się programem `ctlsys.exe`, który znajduje się w pakiecie. Zapisuje on wyniki z kolejnych kroków optymalizacji do pliku. Są one następnie wizualizowane przez skrypt w *Matlabie*. Wskaźnik jakości dla omawianego przykładu ma postać: $\mathbf{f}_0 = -C_C$. Celem sterowania jest więc, by stężenie produktu C w wypływającej z reaktora cieczy było jak największe. Przyjmijmy ponadto, że etap sterowania ma długość $2s$. Optymalizacja dokonuje się na przedziale globalnym o długości $150s$ przy równomiernej dyskretyzacji. Przy dwóch wejściach sterujących daje to wymiar zadania optymalizacji równy 150.

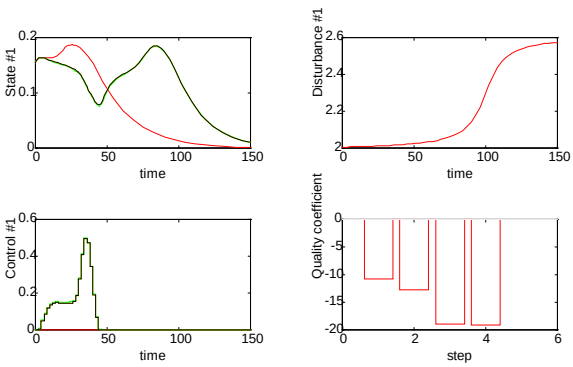
Na wielkości sterujące są nałożone ograniczenia. Dla uproszczenia nie zmieniają się one w czasie. $F_B \in \langle 0;0,5 \rangle \left[\frac{kg}{s} \right]$, $P \in \langle 0,1 \rangle [MW]$. Początkowo wszystkie optymalizowane sterowania mają wartość 0.

Wartości początkowe zmiennych stanu są nieco inne niż przy badaniu całkowania i wynoszą: $C_C = 0{,}155$, $C_A = 0{,}038$, $C_B = 0{,}035$, $T, T_{\square 4}, T_{\square 4a+e} = 120$, $T_{\square 1}, T_{\square 1a+e}, T_{\square 2} = 126$. Ich odmienność nie ma podłoża merytorycznego: jest pozostałością po serii prób prowadzących do uzyskania odpowiedniego modelu dla tego podrozdziału.

Zakłócenia mają tym razem przebieg ciągły (35).



Rys. 35 Zakłócenia działające na reaktor



Rys. 36 Wykresy uzyskiwane podczas doboru parametrów optymalizacji dla modelu reaktora (kolejno: C_C, F_A, F_B i jakość).

Obok (36) przedstawione są wybrane przebiegi w kolejnych krokach optymalizacji. Są to rezultaty dla funkcji $\mathbf{f}_0$ nie przeskalowanej. Optymalizacja zapewnia dobre sterowanie do momentu, gdy F_A rośnie o 30□. Wówczas (a jest to już blisko

horyzontu lokalnego) jakiekolwiek sterowanie zostaje zaniechane.

Taki wynik jest zdecydowanie niezadowalający. Sensowne sterowanie obiektem do samego końca polepszyłoby wskaźnik jakości jeszcze o około 25□. Jest więc o co walczyć.

Kolejne symulacje dla $\mathbf{f}_0$ pomnożonej przez 10, 100 i 1000, oraz przez $\frac{1}{10}$, $\frac{1}{100}$ i $\frac{1}{1000}$ nie przyniosły zasadniczych zmian jakościowych. Sterowanie było optymalizowane tylko do 80. sekundy — tak, jak w przypadku wyjściowym. Te zmiany skali $\mathbf{f}_0$ miały jedynie istotny wpływ na czasy obliczeń oraz powodowały niewielkie różnice w trajektoriach stanów i sterowań. Wskaźnik jakości pozostawał praktycznie bez zmian.

Ta seria symulacji każe przewartościować wpływ skali $\mathbf{f}_0$ na optymalizację. Nie jest on tak duży, jak sądziłem na początku — to dobrze, bo o znalezieniu punktu optymalnego nie powinno przesądzać to, że $\mathbf{f}_0$ jest wyrażona w *milach*, czy w *milimetrach*. Niemniej jednak skala $\mathbf{f}_0$ ciągle ma wpływ choćby na czasy obliczeń i dlatego nie można zupełnie zlekceważyć tego punktu procedury. Dlatego pozostaję przy $\mathbf{f}_0 = -C_C$: wówczas obliczenia trwały najkrócej. Poprawy obecnej sytuacji należy szukać w następnym punkcie procedury.

4. Określenie tolerancji c w kryterium stopu ze względu na gradient.

Przy ustalonej (bądź jeszcze nie) w poprzednim punkcie skali wskaźnika jakości, należy w takich samych warunkach dokonać optymalizacji sterowania przy zmieniającym się c.

Jeśli w poprzednim punkcie został już uzyskany pewien zadowalający poziom sterowania, to należy dążyć do jego polepszenia bądź do zmniejszenia czasu optymalizacji.

Jeśli w poprzednim punkcie wystąpiły problemy z uzyskaniem jakiegokolwiek zadowalającego wyniku, to należy dążyć do osiągnięcia zmiany ilościowej (choćby wykonania kilku, zamiast jednego kroku optymalizacji) i powrócić do punktu poprzedniego w celu ponownego dobrania skali $\mathbf{f}_0$.

Ad 4. Standardowo tolerancja c jest równa 10^{-3}. Cztery optymalizacje sterowania (dla $c \in \{10^{-1}, 10^{-2}, 10^{-4}, 10^{-5}\}$) nie przyniosły wyników różniących się od wyników dla tolerancji standardowej. I przynieść nie mogły, bowiem każdorazowo optymalizacja kończyła się spełnieniem warunków *stopu* względem sterowania ($\left|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\right| < \varepsilon_x$), a nie jego gradientu. W tej sytuacji zostały wykorzystane wszystkie możliwości poprawy optymalizacji, przewidziane w zaimprowizowanej procedurze projektowej. Ale nie przyjęła ona jeszcze wersji ostatecznej: skoro optymalizacja zatrzymuje się na $\left|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\right| < \varepsilon_x$, to warto sprawdzić wpływ ε_x i wzbogacić procedurę o wnioski.

Wnioski okazały się bardzo istotne. Otóż, zmniejszając ε_x (kolejno: $3 \cdot 10^{-8}$, 10^{-8}, $3 \cdot 10^{-9}$, 10^{-9}, ...), wyniki pozostawały takie same, aż do

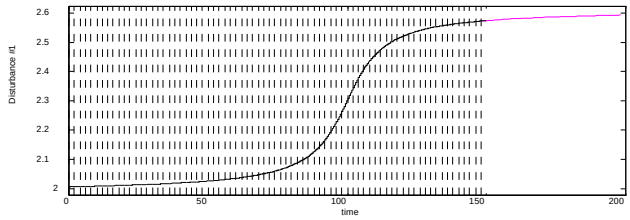
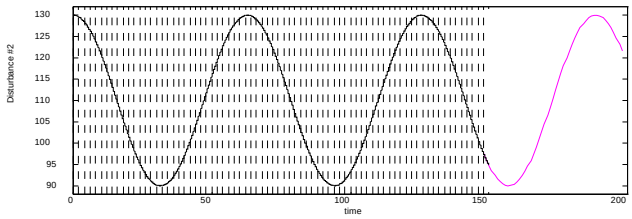
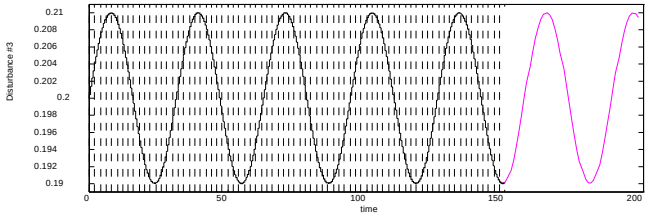
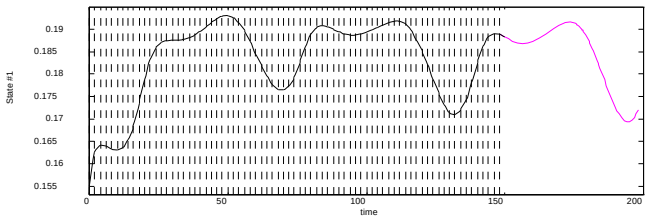
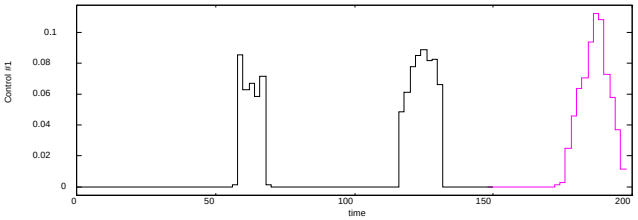
przypadku, gdy $\varepsilon_x = 10^{-10}$. Wówczas nastąpił przełom: obliczenia były znacznie dłuższe, ale optymalizacja sterowania wykonała się do samego końca. Dalsze zmniejszanie ε_x nie przyniosło zmian. Można się zastanawiać, czy uzyskana poprawa (o 25□) wskaźnika jakości w zamian za (10-krotnie) czas optymalizacji, to dobra transakcja□Dobra. Dlatego, że uzyskano wyraźną, *jakościową* zmianę sterowania, że wreszcie jest ono optymalizowane rzetelnie. Dopiero mając tę pewność można szukać sposobów przyspieszenia obliczeń.

Do punktu następnego przechodzę więc z „bezpiecznie" dobranym $\varepsilon_x = 3 \cdot 10^{-11}$.

5. Określenie długości lokalnego przedziału sterowania.

Należy przeprowadzić serię symulacji sterowania repetycyjnego. W kolejnych sesjach należy skracać lokalny horyzont sterowania, aż do momentu, gdy układ sterowania utraci stabilność lub ilość punktów dyskretyzacji sterowania zmaleje do 2. W ten sposób zostanie wyznaczona *krytyczna długość lokalnego przedziału sterowania*, przy której układ jeszcze nie traci stabilności. Dla bezpieczeństwa można do dalszych obliczeń przyjąć przedział lokalny o 50÷100□ dłuższy od krytycznego.

Ad 5. Dla skrócenia procedury rozpocznę badanie wpływu długości przedziału lokalnego od 80*s* (40 punktów dyskretyzacji sterowania). Pozostałe warunki symulacji — jak poprzednio. Symulacja na horyzoncie globalnym $\langle 0{,}200 \rangle$. Poniżej (8) przedstawiam najważniejsze wyniki oraz komentarze.

Krok	Dł. przedziału lokalnego □s□	Liczba punktów dyskr. sterowania	Wskaźnik jakości	Uwagi
1	50	25	-0,1856	Poniższe wykresy przedstawiają ważniejsze zrealizowane trajektorie: zakłóceń, zmiennych stanu i sterowań (wg oznaczeń w kolumnie z lewej), wraz trajektoriami przewidywanymi na etapie ostatnim (kolor różowy): F_A□  T_A□  C_{A_0}□  C_C□  F_B□ 

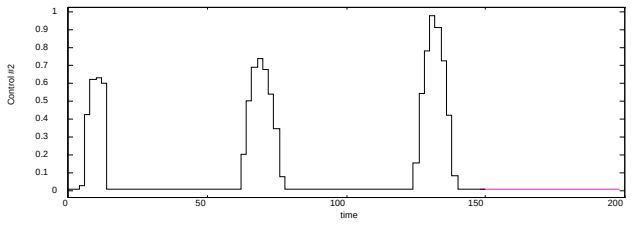
			P□	 Stężenie C_C nieco spada po wzroście F_A. Oba sterowania często osiągają dopuszczalne minima; nie osiągają maksimów. Wyraźnie zaznacza się wzrost F_B po wzroście F_A. P ma charakter impulsowy: jest duże tylko gdy T_A spada.
2	40	20	-0,1844	Trajektoria stanu bardzo podobna do tej z p. 1. Impulsy na przebiegach sterujących niższe, ale szersze.
3	30	15	-0,1838	Impuls 3. (przewidywany) przebiegu F_B ma dużo mniejszą amplitudę. Impulsy P jeszcze szersze i niższe.
4	20	10	-0,1710	Od 60. sekundy zaczynają się duże oscylacje trajektorii stanu; wartość średnia C_C zaczyna spadać. Sterowanie F_B ma więcej impulsów, co wynika z oscylacji stanu.
5	10	5	-0,1267	Od 80. sekundy C_C opada asymptotycznie do 0. Od tej chwili nie są wystawiane nowe sterowania.

Tabela 8 Procedura doboru długości lokalnego przedziału sterowania dla modelu reaktora

Okazuje się, że zmniejszenie długości przedziału lokalnego powoduje załamanie się sterowania. Tylko na początku potrafi ono sprostać zadaniu optymalizacji; później obiekt „wypada" z punktu pracy a gradient funkcji celu jest tak mały, że nie można wprowadzić go weń ponownie. Krytyczna długością lokalnego przedziału sterowania leży między 30*s*. a 20*s*. Przyjęcie lokalnego przedziału sterowania o długości 30*s* jest wystarczająco bezpieczne.

6. Dyskretyzacja sterowania na przedziale lokalnym i kara za stan w na jego końcu.

Dalszą poprawę jakości sterowania i szybkości obliczeń można uzyskać poprzez:

- wprowadzenie kary za stan na końcu lokalnego horyzontu sterowania; należy dokonać symulacji dla funkcji □ z różnymi współczynnikami skali: od kary znikomej w porównaniu z całką z $\mathbf{f}_0$ — do kary znacznie ją przewyższającej,
- nierównomierną dyskretyzację sterowania na przedziale lokalnym; należy stopniowo usuwać niektóre punkty dyskretyzacji (strategia *A* lub *B*) i dokonywać symulacji sprawdzając, jak zabiegi te wpływają na czas obliczeń i jakość sterowania.

Należy wybrać, która z metod daje najlepsze rezultaty.

Strategia *A*: punkty dyskretyzacji sterowania są usuwane kolejno, począwszy od horyzontu lokalnego w kierunku początku przedziału.
Strategia *B*: usuwany jest co drugi punkt dyskretyzacji sterowania; począwszy od horyzontu lokalnego w kierunku początku przedziału. Po osiągnięciu początku przedziału procedura jest powtarzana na przerzedzonym zbiorze punktów dyskretyzacji.

Ad 6(□). Przy obecnie osiąganych wynikach całka z $\mathbf{f}_0$ na horyzoncie lokalnym jest rzędu -2,76. Zatem przyjęcie $\Box(C_C) = -15C_C$ da karę za stan równą w przybliżeniu tej całce. Postanowiłem przebadać zachowanie się układu przy karze za stan równej 0,1, 0,3, 1, 3 i 10 □ (9).

Krok	□ mnożone przez	Czas obliczeń □s□	Wskaźnik jakości
1	0	1904	-0,1838
2	0,1	1583	-0,1838
3	0,3	1764	-0,1838
4	1	1738	-0,1838
5	3	1690	-0,1838
6	10	2218	-0,1838

Tabela 9 Wpływ □ na jakość sterowania w przypadku reaktora

Jakość sterowania jest niewrażliwa na □ zmieniające się w dość szerokim zakresie. Zmiany funkcji kary wpływają jedynie w niewielkim stopniu na kształty trajektorii stanu i w większym (ale bez wyraźnej reguły) na czasy obliczeń. Pozostaje próbować poprawić sterowanie stosując rzadszą lokalną dyskretyzację sterowania.

Ad 6(□). Zostały dokonane symulacje sterowania dla obu strategii (*A* i *B*) dla 80, 60, 40 i 20 procent pierwotnej liczby punktów dyskretyzacji. □adna nie przyniosła poprawy jakości sterowania (dla wersji wyjściowej jest ona najlepsza z możliwych). Interesujące okazały się natomiast czasy obliczeń, przedstawione poniżej (10).

Krok	□ pierwotnej liczby punktów dyskretyzacji	□łączny czas obliczeń □*s*□dla rozrzedzania z użyciem strategii	
		A	*B*
1	100	1904	
2	80	1196	1618
3	60	1215	1559
4	40	1165	1410
6	20	1047	1008

Tabela 10 Wpływ rozrzedzania optymalizacji na czasy obliczeń w przypadku reaktora

Stosując strategię *A* osiąga się radykalne skrócenie czasu obliczeń już w kroku 2. Później zmiany są już minimalne. Strategia *B* systematycznie, w kolejnych krokach, powoduje skrócenie czasów obliczeń by, w kroku ostatnim, minimalnie prześcignąć *A*. Dlatego w dalszych symulacjach sterowania reaktorem będą stosowane takie same ustawienia, jak dla kroku 5. przy strategii *B*.

Analizując wyniki badań w punkcie 6. ma się wrażenie, że najistotniejsze jest przede wszystkim „uświadomienie" układowi sterującemu natury modelu; tego, że sterowanie nie kończy się wraz z końcem etapu. Zagadnienia dyskretyzacji sterowania czy istnienia kary za stan końcowy mają tutaj znaczenie drugorzędne. Sterowanie z najbardziej zredukowaną

liczbą punktów dyskretyzacji (w obu strategiach) dało najlepsze wyniki dlatego, że nie obciążając modułu całkującego nadmiarem skokowych zmian sterowania potrafiło odsłonić przed modułem optymalizującym istotną część dynamiki reaktora.

W tym rozdziale zostało przeprowadzonych wiele testów mających na celu poznanie wpływu parametrów algorytmu na jakość sterowania. Dotyczyły one zaledwie kilku klas obiektów dynamicznych; wystarczyło to jednak by odsłonić liczne wady w implementacji algorytmu. Wypracowane zalecenia projektowe wciąż domagają się weryfikacji i każde nowe zadanie projektowe wzbogaci wiedzę o zagadnieniu. Przykładem może być ostatni model reaktora. Po raz pierwszy zaimplementowany algorytm musiał być zastosowany do regulacji wielowymiarowej (C_C bardzo silnie zależy od T, C_A i C_B). Po raz pierwszy sterowano obiektem o 17 zmiennych stanu. Po raz pierwszy okazało się, że dobór wartości parametru w kryterium *stopu* ze względu na sterowanie jest tak samo ważny, jak dobór innych parametrów optymalizacji.

Wypada ubolewać, że optymalizacja sterowania przebiega tak powoli. Wciąż potrzeba znacznie skrócić czasy obliczeń, by algorytm mógł być zastosowany do układów rzeczywistych.

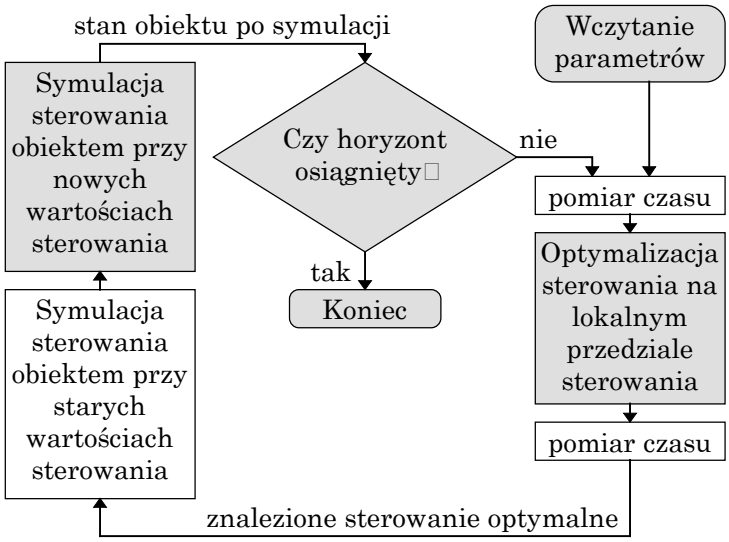
Rozdział następny to różne inne badania algorytmu repetycyjnego. Część z nich sprawdza, co by było, gdyby optymalizacja dokonywała się tak szybko, jak to jest pożądane — to część poświęcona opóźnieniom decyzyjnym. Inne testy to próba porównania tego algorytmu z najprostszym i najbardziej niezawodnym: PID.

5. Inne właściwości

Po opracowaniu podstawowych zasad doboru parametrów algorytmu repetycyjnego można przystąpić do zbadania niektórych jego właściwości. Warto sprawdzić, jak zachowuje się on w symulacji w czasie rzeczywistym, gdy występują opóźnienia decyzyjne a dopuszczalny czas na znalezienie nowego sterowania jest ograniczony. Interesujące może być ponadto sprawdzenie działania algorytmu przy stałym horyzoncie lokalnym (czyli przy skracającym się, w trakcie symulacji, lokalnym przedziale sterowania). Wskazane jest ponadto dokonanie porównania algorytmu repetycyjnego z innym znanym, na przykład PID.

5.1 Opóźnienia decyzyjne

Do tej pory czynione było założenie, że wyznaczenie nowego sterowania odbywa się natychmiast po uzyskaniu prognozy zakłóceń i informacji o stanie obiektu. W praktyce jednak może ono trwać bardzo długo. Aby móc dokonywać symulacji z uwzględnieniem opóźnień decyzyjnych, potrzebna jest modyfikacja głównego algorytmu symulacji (37, por. też z



Rys. 37 Algorytm symulacji z uwzględnieniem opóźnień decyzyjnych

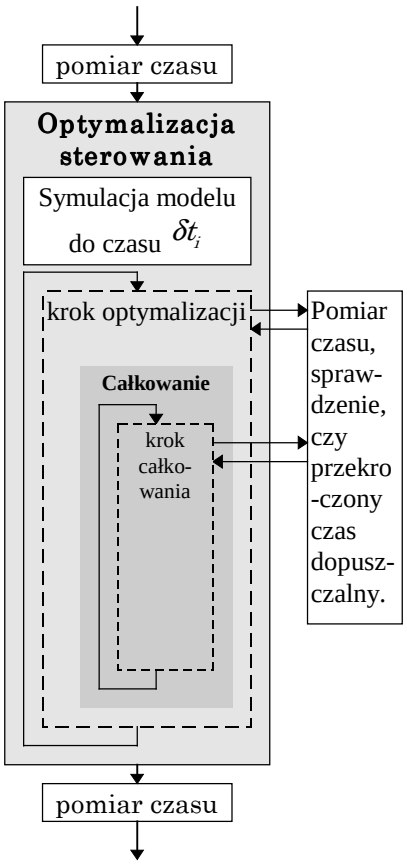
Rys. .).

W jego nowej wersji dokonuje się pomiaru czasu (funkcja *ftime*) przed przystąpieniem do optymalizacji sterowania i po jego zakończeniu. Zmierzony w ten sposób czas optymalizacji Δt_i to faktyczne opóźnienie decyzyjne, które musi być wyegzekwowane po stronie obiektu dynamicznego.

Ze względu na zmiany w algorytmie optymalizacji (o których mowa później), opisywana procedura charakteryzuje się jeszcze jednym parame-

trem: *przewidywanym czasem optymalizacji* δt_i. Opóźnienie wyegzekwowane nie może być mniejsze niż δt_i. Zatem dodatkowa symulacja ze starymi wartościami sterowania nastąpi na odcinku od t_i do $t_i + \mathbf{max}(\Delta t_i, \delta t_i)$. Później zostanie zastosowane nowo znalezione sterowanie, które będzie aplikowane aż do zakończenia optymalizacji i wyznaczenia następnego.

Przejście do symulacji w czasie rzeczywistym wymaga zmian również w algorytmie optymalizacji. Dotychczas mogła się ona zakończyć pomyślnie i wówczas symulacja była kontynuowana; mogła też zawieść z powodu utraty dokładności numerycznej w całkowaniu bądź przekroczenia limitu kroków optymalizacji. Jednak w każdym z tych przypadków czas nie odgrywał roli. Obecnie, w sytuacji, gdy opóźnienie decyzyjne dłuższe niż długość etapu jest niedopuszczalne, trzeba zastosować mechanizm ograniczania czasu obliczeń.



Rys. 38 Kontrola czasu trwania optymalizacji

W każdym kroku optymalizacji i w każdym kroku całkowania jest sprawdzane, czy nie został przekroczony *dopuszczalny czas obliczeń* $t_{i\,\mathbf{max}}$ a jeśli tak, to bieżąca operacja jest przerywana i cała procedura optymalizacji sterowania zwraca najlepsze znalezione do tej pory sterowanie (38). W szczególności ograniczenia czasowe mogą być tak rygorystyczne, że nie zostanie wykonany ani jeden krok i zrealizowany będzie kolejny fragment trajektorii sterowania z poprzedniej optymalizacji.

Ponadto, została dokonana modyfikacja pozwalająca uwzględnić opóźnienie decyzyjne już w procesie optymalizacji sterowania. Procedura optymalizacji jest uruchamiana z dodatkowymi parametrami: przewidywanym czasem optymalizacji δt_i i ostatnio zastosowanym sterowaniem. Na ich podstawie, jeszcze przed przystąpieniem do właściwej optymalizacji dokonuje się wstępnej symulacji modelu obiektu do czasu $t_i + \delta t_i$ i przesunięcia lokalnego przedziału sterowania o δt_i. Następnie algorytm działa w sposób standardowy. Dodatkowa symulacja (podobnie, jak nieustanne pomiary czasu) oznacza dalsze spowolnienie algorytmu, ale może się opłacić.

Takie rozwiązanie pozwala jeśli nie na uniknięcie opóźnień decyzyjnych, to przynajmniej na obliczanie sterowań ze świadomością ich wystę-

powania. Ta świadomość jest jednak dobra na tyle, na ile dobre jest oszacowanie δt_i. Należy pamiętać, że musi być ono dokonane *przed* samą optymalizacją. Jeśli więc $\delta t_i < \Delta t_i$, to uzyskane sterowanie i tak „spóźni się" w sensie i fizycznym, i logicznym. Jeśli $\delta t_i > \Delta t_i$, to oznacza, że optymalizator zwrócił sterowanie właściwe dla chwili późniejszej i należy z nim poczekać do $t_i + \delta t_i$ — co też się zdarza.

Zatem symulacje sterowania w czasie rzeczywistym wymogły wprowadzenie dwóch nowych parametrów:

$t_{i\,\mathbf{max}}$ maksymalnego czasu trwania optymalizacji, po którym *musi* zostać wystawione sterowanie; ze względów bezpieczeństwa $t_{i\,\mathbf{max}}$ nie może być dłuższy niż $9/10$ długości etapu;

δt_i przewidywanego czasu trwania optymalizacji; sterowanie jest znajdowane z uwzględnieniem tego parametru; znalezione sterowanie nie powinno być stosowane wcześniej niż $t_i + \delta t_i$.

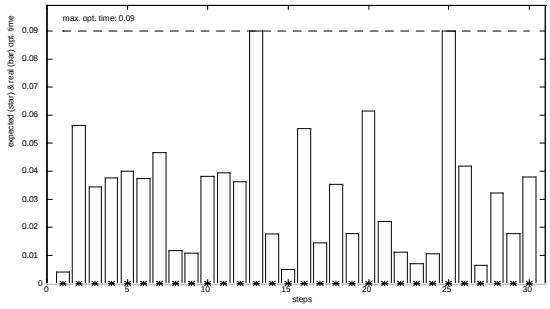
Niestety, cały powyższy wywód wydaje się być bez sensu, jeśli wziąć po uwagę dotychczasową szybkość działania algorytmu. Znalezienie sterowania na pojedynczym etapie trwa dziesiątki, a nierzadko i setki sekund przy założeniu, że etap ma długość rzędu kilku sekund lub ułamka sekundy. Do prawdziwego sterowania w czasie rzeczywistym potrzeba albo obiektu o bardzo wolnej dynamice, albo wielu przeróbek w istniejącym programie, albo rewelacyjnie szybkiego komputera. Albo wszystkich tych czynników naraz.

Jednak, dla celów wyłącznie symulacyjnych, można pokusić się o przeskalowanie osi czasu i wyniki wszystkich pomiarów dokonanych *ftime* mnożyć przez R, nazywane *skalą czasu*. W ten sposób można dowolnie przyspieszać (bądź spowalniać) proces optymalizacji. Takie rozwiązanie nie przybliża dokładnie sytuacji, w której algorytm wykonywał się na szybszym komputerze; wpływ zdarzeń losowych (przerwania, procedury wykonywane okresowo) jest tutaj jednak dużo mniejszy, bo uśredniony. Niemniej jednak jest to obecnie najprostsza metoda symulowania działania w warunkach rzeczywistych.

5.1.1 Wpływ czasu optymalizacji

Interesujące jest zbadanie, jaki wpływ na jakość sterowania ma szybkość komputera, lub inaczej — zależne od niej opóźnienie decyzyjne. W tym celu dokonano sterowania poznanymi już modelami przy R zmieniającym się niekiedy w bardzo szerokich granicach. Dla tej serii symulacji przewidywany czas optymalizacji δt_i jest 0 (optymalizacja bez uwzględnienia opóźnienia) a $t_{i\,\mathbf{max}}$ przyjmuje największą dopuszczalną wartość: 0,9 długości etapu.

Rys. 39 Jakość sterowania a opóźnienia decyzyjne (wahadło)

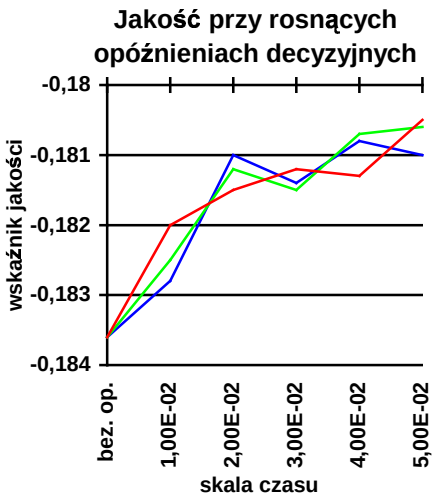


Rys. 40 Czasy optymalizacji ($R = 0{,}00144$)

Symulacje były wykonywane przy pełnej znajomości zakłóceń i przy R dających opóźnienia od bardzo małych do znaczących (gdy czas obliczeń sięgał $t_{i\,\max}$). W celu uśrednienia wyników dokonano po 3 serie symulacji dla każdego modelu. **Dla wahadła** (39) zwiększanie opóźnienia generalnie powoduje pogorszenie się wskaźnika jakości. Jego wzrost nie jest, co prawda, jednostajny ani wysoce powtarzalny w kolejnych seriach; niemniej jednak dla dużych opóźnień (gdy na niektórych etapach jest osiągane $t_{i\,\max}$) jakość sterowania pogarsza się ponad dwukrotnie.

Niestety, nie wszystkie symulacje się powiodły (stąd przerwy w wykresach). W niektórych przypadkach znów zawodziła procedura całkująca, znów była przekraczana dopuszczalna liczba kroków całkowania. Wszystkie dotychczasowe wyniki zostały uzyskane z programów skompilowanych przez *Microsoft Visual C□□ v. 2.0*, więc było możliwe, że nie jest to najlepszy produkt do implementacji algorytmów numerycznych. Jednak te same symulacje przeprowadzone w *Linuxie* po skompilowaniu przez *gnu* dały taki sam skutek. Dopiero symulacje na komputerze *Sun* zakończyły się sukcesem. Okazuje się więc, że wciąż jeszcze sporo zależy od platformy sprzętowej, na której wykonuje się algorytm. Nie można zatem mówić o całkowitej przenaszalności programów między systemami, nawet jeśli są one napisane zgodnie ze standardem *ANSI C*. Dalsze symulacje były konsekwentnie prowadzone na komputerze typu *PC*, by zapewnić porównywalność uzyskiwanych wyników.

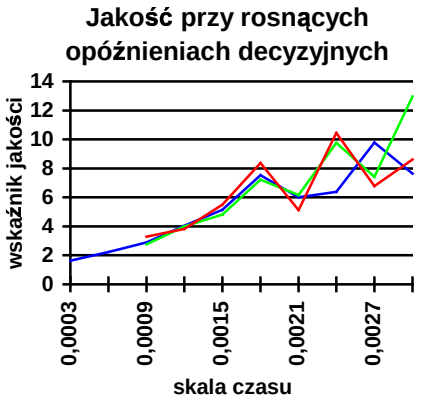
Problemy z procedurą całkującą najbardziej dały się we znaki przy symulacjach **modelu suwnicy**. Praktycznie tylko połowa z nich zakończyła się pomyślnie, a i te nie dały jednoznacznych wyników. Można jedynie stwierdzić, że zwiększanie opóźnień powoduje od pewnego momentu bardzo znaczne pogorszenie się wskaźnika jakości (liczonego po ustaniu od-



Rys. 41 Jakość sterowania a opóźnienia decyzyjne (reaktor)

działywania warunków początkowych). Dalsze zwiększanie R prowadzi do błędów procedury całkującej i fiaska symulacji.

Model reaktora przysporzył o wiele mniej problemów. Okazało się, że nawet dla dużych opóźnień ($R = 0{,}05$) jakość sterowania nie ulega znacznemu pogorszeniu (). Należy, owszem, brać pod uwagę naturę obiektu: uzyskiwane do tej pory wskaźniki sterowania reaktorem były albo bardzo wysokie, albo zerowe. Niemniej pogorszenie się wskaźnika o 1□ przy średnim opóźnieniu decyzyjnym rzędu 50□ długości etapu znaczy, że mamy do czynienia z przykładem obiektu bardzo odpornego na opóźnienie decyzyjne. W dotychczasowych badaniach tego modelu przełomowe znaczenie miał dobór długości lokalnego przedziału sterowania; żadne późniejsze zabiegi potencjalnie osłabiające moc algorytmu (rzadsza dyskretyzacja, opóźnienia decyzyjne) nie wywarły już takiego wpływu na jakość sterowania.



Rys. 42 Jakość sterowania a opóźnienia decyzyjne (inercja)

Większość z omówionych dotychczas modeli zachowywała się w zasadzie zgodnie z oczekiwaniami, tzn. wzrost opóźnienia decyzyjnego powodował pogarszanie się wskaźnika jakości. Dla żadnego modelu nie jest ono jednak tak wyraźne, jak dla **modelu pojedynczej inercji**. W dotychczasowych badaniach nie zyskał on wiele uwagi; jednak jego prostota ułatwia dostrzeżenie kolejnych skutków „spóźniania się" sterowania.

Poniżej (11) zostały przedstawione przebiegi zmiennej stanu i czasy obliczeń dla małych i dużych opóźnień decyzyjnych (w drugim przypadku ograniczenie na czas trwania obliczeń uaktywnia się na większości etapów).

małe opóźnienia decyzyjne duże opóźnienia decyzyjne

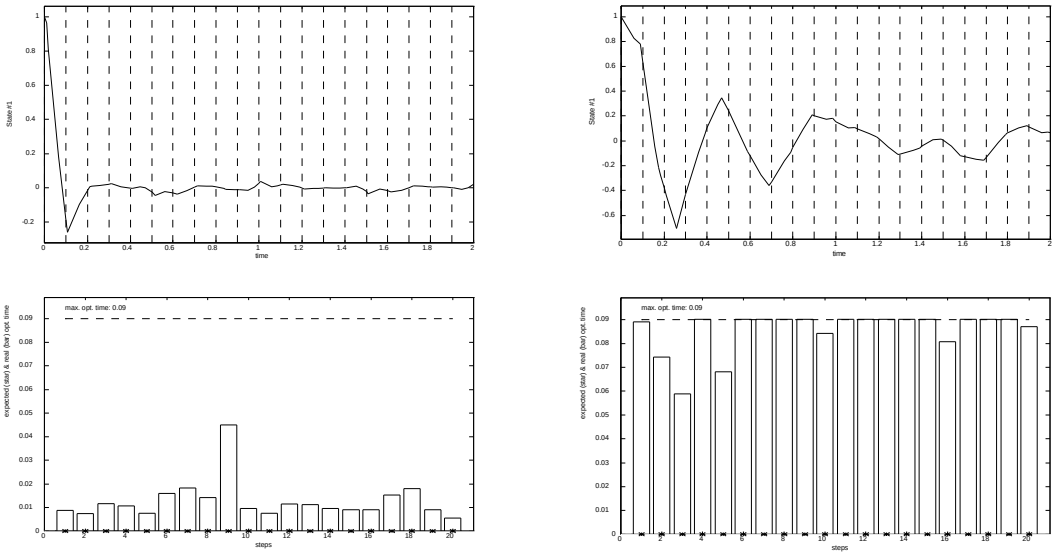


Tabela 11 Zestawienie przebiegów zmiennej stanu dla różnych czasów optymalizacji (inercja)

Okazuje się, że znaczny poślizg we wprowadzaniu nowego sterowania spowodował duże oscylacje zmiennej stanu; wręcz trudności w uzyskaniu wartości zadanej. A wszystko to dla obiektu, którego stan można łatwo sprowadzić do zera praktycznie w dwóch posunięciach. Ta łatwość okazuje się bardzo niewygodna, jeśli opóźnienia są duże i, co gorsza, optymalizator nie jest ich świadom.

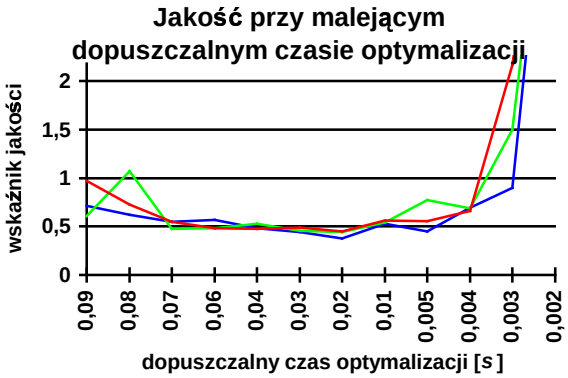
Podsumowując: wzrost opóźnień decyzyjnych ma zawsze niekorzystny wpływ na jakość sterowania. Skala tego oddziaływania zależy jednak w istotnym stopniu od samej natury obiektu: duża dla obiektów prostych (inercja), mniejsza dla bardziej złożonych (wahadło), prawie niezauważalna dla skomplikowanych i z opóźnieniami (reaktor). Osobną kwestię stanowi niezawodność modułu całkującego (a raczej jej częsty brak). Wpływa na to nie tylko niedopracowanie algorytmu ale i specyfika platformy sprzętowej.

5.1.2 Wpływ ograniczania czasu optymalizacji

W poprzednim paragrafie próbowałem określić jak szybki powinien być komputer na którym wykonywany jest algorytm (parametr R), by optymalizacja sterowania miała szansę zakończyć się przed końcem etapu sterowania i jakie skutki wywoła tak duże opóźnienie decyzyjne. Okazuje się, że nie zawsze duże opóźnienia decyzyjne powodują znaczne pogorszenie jakości sterowania. Ale w przypadkach, gdy powodują, należy znaleźć sposób na jego redukcję. Najprostszym wydaje się skrócenie dopuszczalnego czasu optymalizacji $t_{i\,\mathbf{max}}$.

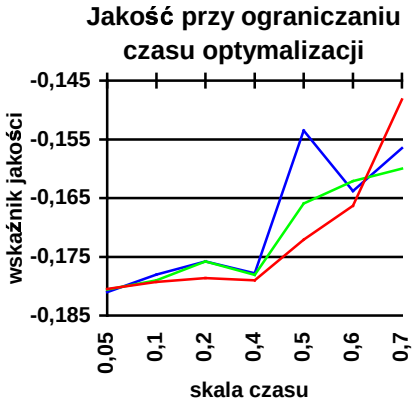
Trzeba jednak zauważyć, że skracanie opóźnienia decyzyjnego tą drogą odbywa się kosztem optymalności wyznaczonego rozwiązania.

Mniej czasu na optymalizację oznacza gorszy jej wynik. Warto więc pokusić się o sprawdzenie, na ile można skrócić czas dopuszczalny optymalizacji, by nie doprowadzić do pogorszenia sterowania. Poniżej przedstawiam wyniki symulacji.



Rys. 43 Jakość a dopuszczalny czas optymalizacji (wahadło)

Wahadło (43). Symulacje potwierdzają przypuszczenia. Umiarkowane ograniczanie czasu optymalizacji (pierwszych 6 punktów na wykresach) nie wpływa znacznie na optymalność rozwiązania; za to istotnie redukuje opóźnienia decyzyjne. Wypadkowo, wskaźnik jakości poprawia się (w tym przypadku nawet o 50□). Dalsze ograniczanie $t_{i\,\max}$ powoduje zmniejszenie opóźnień o pojedyncze procenty długości etapu; natomiast czas na zoptymalizowanie sterowania kurczy się każdorazowo o dziesiątki procent. Dla ostatnich punktów na wykresach zdołały się wykonać jeden, dwa lub trzy kroki optymalizacji. To zbyt mało, by dotrzeć w okolice minimum i dlatego w układzie na całym horyzoncie globalnym występowały bardzo silne oscylacje zmiennych stanu i gwałtowne skoki sterowań.



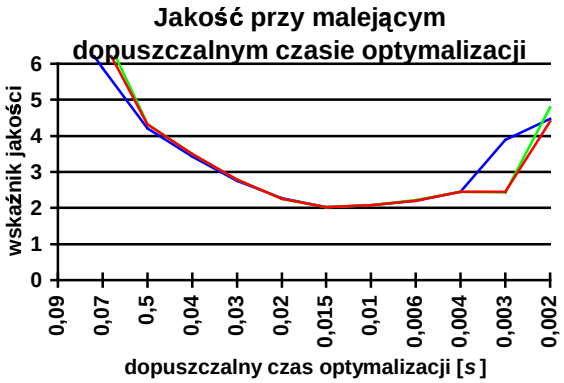
Rys. 44 Jakość a dopuszczalny czas optymalizacji (reaktor)

Reaktor (44). Dla modelu reaktora nie ma sensu jawne skracanie $t_{i\,\max}$. O wiele lepiej będzie pozostawić je prawie równe długości etapu sterowania (i tak opóźnienie decyzyjne nie ma tu wielkiego wpływu na jakość) i kontynuować symulacje przy rosnącym R. Także w ten sposób można redukować $t_{i\,\max}$.

Uzyskane wyniki prowadzą do dwóch wniosków. Po pierwsze, duże (cztero-, ośmiokrotne) skrócenie dopuszczalnego czasu optymalizacji nie wpływa wyraźnie negatywnie na jakość sterowania. Na przykład, dla $R = 0{,}4$ wystarcza czasu na dokonanie tylko czterech kroków optymalizacji a mimo to układ sterowania spełnia swoje zadanie. Niestety, dalsze zwiększanie R skutkuje pojawieniem się dużych oscylacji C_C, aż do zaprzestania sterowania w połowie sesji

($R = 0{,}7$, wykres czerwony). Ograniczenia czasowe są tak silne, że nie na wszystkich etapach zdoła wykonać się choćby jeden krok optymalizacji.

Po drugie, okazuje się, że tylko dwukrotne zwiększenie szybkości wykonywania obliczeń dzieli nas od zastosowania regulatora go w układzie rzeczywistym. To bardzo optymistyczna konkluzja zważywszy obecny postęp technologii i możliwość dalszego cyzelowania algorytmu.



Rys. 45 Jakość a dopuszczalny czas optymalizacji (inercja)

Inercja (45). Także w tym przypadku skrócenie $t_{i\,\max}$ przynosi korzyści. Sześciokrotna redukcja $t_{i\,\max}$ daje jakość sterowania prawie taką, jak dla przypadku sterowania bez opóźnień — ale przy R 10-krotnie większym niż pierwotne. Oznacza to, że można sterować obiektem tak samo dobrze za pomocą algorytmu działającego dziesięciokrotnie wolniej. Dalsze zmniejszanie $t_{i\,\max}$ powoduje ponowne pogorszenie sterowania — nic w tym dziwnego: dla $t_{i\,\max} = 0{,}004$ wykonuje się jedynie jeden krok optymalizacji na każdym etapie. Dla $t_{i\,\max}$ jeszcze mniejszych optymalizacja sterowania po prostu nie wykonuje się na wszystkich etapach i stąd znaczne pogorszenie wskaźnika jakości.

Jeszcze raz potwierdza się spostrzeżenie, że liczba wykonywanych dotychczas kroków optymalizacji była nadmiarowa, że warto ją drastycznie ograniczać, by kosztem optymalności wyznaczonego sterowania (wahadło, inercja) zmniejszyć opóźnienia decyzyjne. Tam zaś, gdzie opóźnienie decyzyjne nie odgrywa zbytniej roli (reaktor), skrócenie czasu optymalizacji oznacza znaczne przyspieszenie algorytmu tak, że realne staje się zastosowanie go w rzeczywistości.

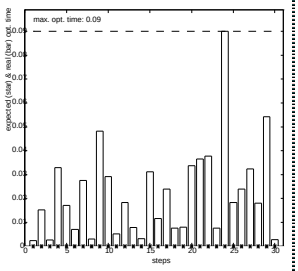
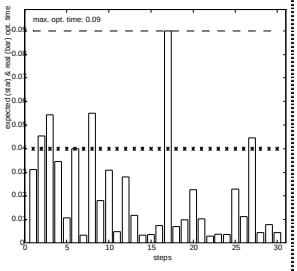
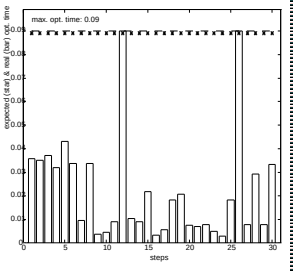
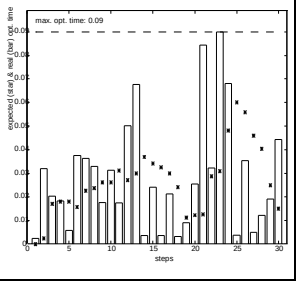
5.1.3 Wpływ przewidywania czasu optymalizacji

Dobór przewidywanego czasu trwania optymalizacji δt_i może mieć bardzo istotny wpływ na jakość uzyskiwanego sterowania. Jest to parametr przekazywany każdorazowo do modułu sterującego przed rozpoczęciem optymalizacji sterowania. Możliwe jest określenie jednakowego δt_i dla wszystkich etapów na podstawie wcześniejszych wyników symulacji (np. podanie wartości średniej czasu obliczeń na pojedynczym etapie); można też zdać się na procedurę szacowania δt_i dynamicznie, na każdym kroku (średnia krocząca z 5 ostatnich etapów). Należy zdawać sobie sprawę, że zbyt małe δt_i oznacza, że sterowanie będzie przykładane z

opóźnieniem; zbyt duże — że procedura symulująca będzie zwlekać, a w tym czasie stan obiektu oddali się od zamierzonego.

Poniżej (12) przedstawiłem wyniki symulacji przy różnych δt_i: małym, średnim i dużym w porównaniu z rzeczywistymi czasami obliczeń, oraz wyznaczanym w sposób automatyczny. Gwiazdki wskazują δt_i na poszczególnych etapach.

Wahadło. Przypadek, gdy nie zachodzi ograniczanie czasu optymalizacji. Wskaźniki jakości dla coraz większego δt_i (3 pierwsze) i dla δt_i dobieranego automatycznie. Okazuje się, że automatyczny dobór δt_i jest bardzo nieefektywny. Znacznie lepsze wyniki uzyskuje się stosując δt_i stałe, najlepiej niezbyt duże.

Wsk. jakości: 0,378	0,351	0,411	0,850
			

Reaktor. Przypadek, gdy ograniczanie czasu optymalizacji jest dość powszechne. Wskaźniki jakości (2 pierwsze) dla δt_i zerowego i maksymalnego oraz dla δt_i dobieranego automatycznie. Także w tym przypadku wyspecyfikowanie δt_i zbyt dużego bądź estymowanie go na bieżąco szkodzi, ale mniej niż w przypadku wahadła.

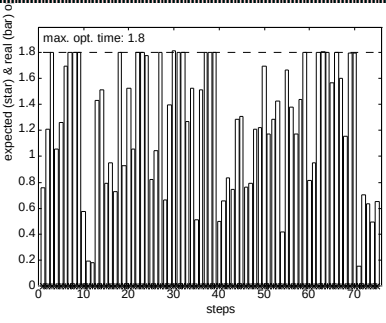
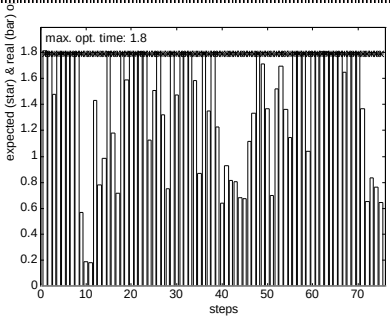
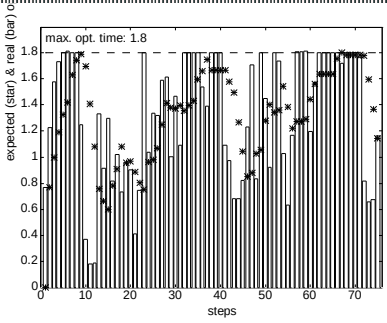
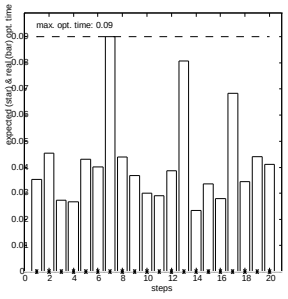
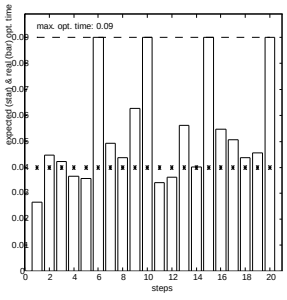
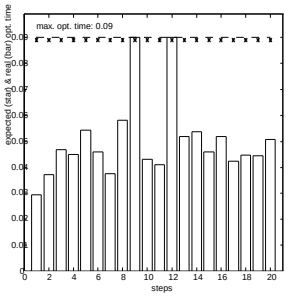
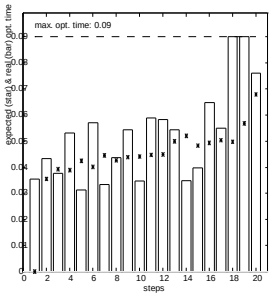
Wsk. jakości: -0,1793	-0,1789	-0,1783
		

Tabela 12 Wpływ δt_i na jakość (modele wahadła i reaktora)

Powyższe wyniki są raczej kontrprzykładem celowości stosowania parametru δt_i. Niemniej jednak, dla tak prostego obiektu, jak inercja, rezultaty idą w parze z oczekiwaniami.

Inercja. Przypadek, gdy nie zachodzi ograniczanie czasu optymalizacji. Wskaźniki jakości dla coraz większego δt_i (3 pierwsze) i dla δt_i dobieranego automatycznie. W przypadku tego obiektu dobór δt_i zbliżonego do rzeczywistego czasu optymalizacji może istotnie poprawić jakość sterowania. Zarówno zbyt małe (1), jak i zbyt duże (3) δt_i skutkuje znacznym pogorszeniem wskaźników. Tym razem najkorzystniej jest oprzeć się na wartości średniej czasów obliczeń (2) bądź zastosować automatyczną procedurę doboru δt_i (4).

Wsk. jakości: 3,5 (1)	2,89 (2)	4,25 (3)	2,72 (4)
			

Wyniki poniżej dotyczą sytuacji, gdy zachodzi ustawiczne ograniczanie czasu optymalizacji. Tym razem dla $\delta t_i = 0{,}4$, idealnego dla krótszych czasów obliczeń, wynik nie jest już optymalny; znakomicie natomiast wypada typowanie przewidywanego czasu obliczeń równego czasowi maksymalnemu (3). Również tutaj automatyczny dobór δt_i sprawdza się.

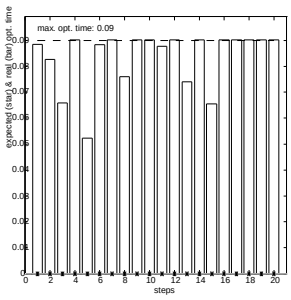
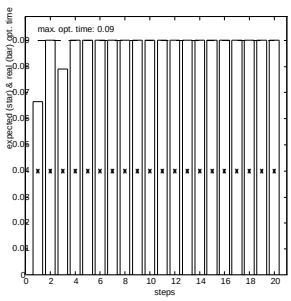
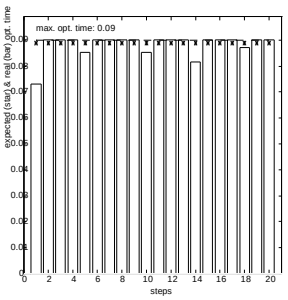
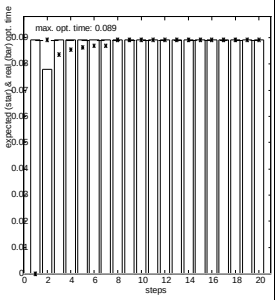
Wsk. jakości: 8,06 (1)	5,09 (2)	4,26 (3)	4,32 (4)
			

Tabela 13 Wpływ δt_i na jakość w przypadku inercji

Przytoczone przykłady wskazują, że sens stosowania optymalizacji z uwzględnieniem opóźnienia decyzyjnego istnieje tylko dla niektórych obiektów. W większości zbadanych przeze mnie przypadków korzyści z tego przewidywania były nikłe (jeśli w ogóle były). Natomiast istnieją obiekty, jak choćby opisany tutaj najprostszy inercyjny, dla których gra ono bardzo istotną rolę. Przypuszczalnie znaczenie przewidywania powstania opóźnienia jest niebagatelne dla obiektów dających się doprowadzić do stanu zadanego w kilku zaledwie krokach. Wówczas sterowanie uzyskane z optymalizatora jest skrojone na tyle precyzyjnie, by „trafić" zadany stan już w np. 2 ruchach i każda niespodzianka w postaci opóźnienia potrafi poważnie popsuć szyki. Dlatego tak ważna jest trafna prognoza czasu trwania optymalizacji.

5.2 Stały lokalny horyzont sterowania

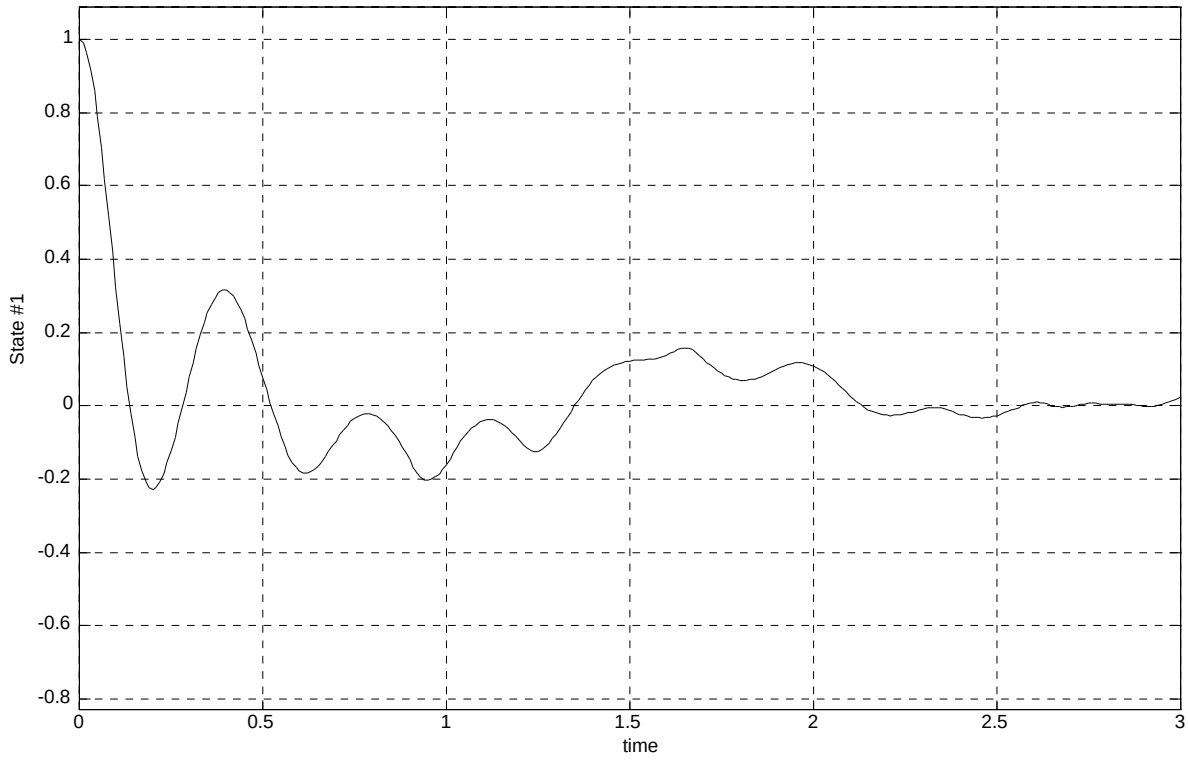
W podrozdziałach 4.3 i 4.4, traktujących o doborze długości lokalnego przedziału sterowania i rozmieszczaniu na nim punktów dyskretyzacji, przypominałem o uproszczeniu algorytmu; o tym, że optymalizowanie sterowania jedynie na fragmencie przedziału globalnego może w istotny sposób ujemnie wpływać na wskaźniki jakości. Jednak ze względu na spodziewane znaczne wydłużenie czasów obliczeń poprzestałem na symulacjach właśnie w takich warunkach.

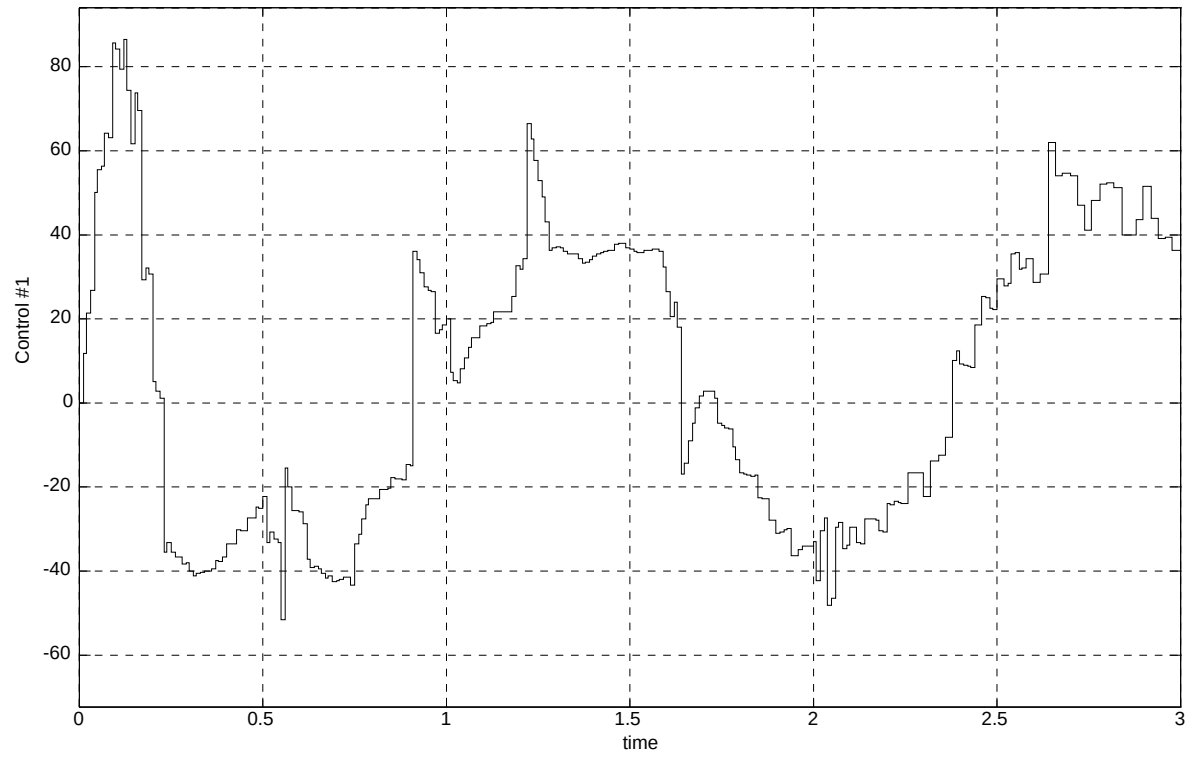
Teraz przyszła pora, by empirycznie zweryfikować przekonania. Poniższe wyniki dotyczą symulacji, w których horyzont lokalny $t_i + T_i^*$ jest taki sam dla wszystkich i i równy horyzontowi globalnemu t_f. To oznacza, że T_i^* maleje w kolejnych etapach symulacji. Zmniejsza się więc wymiar zadania optymalizacji, zatem należy się spodziewać malenia czasu obliczeń. Jeśli założymy ponadto, że układ sterujący dysponuje pełną wiedzą o zakłóceniach, to (korzystając z zasady optymalności Bellmana) trajektoria sterowania wyznaczona na etapie i powinna być częścią trajektorii z etapu poprzedniego. Okazuje się, że wcale tak nie jest.

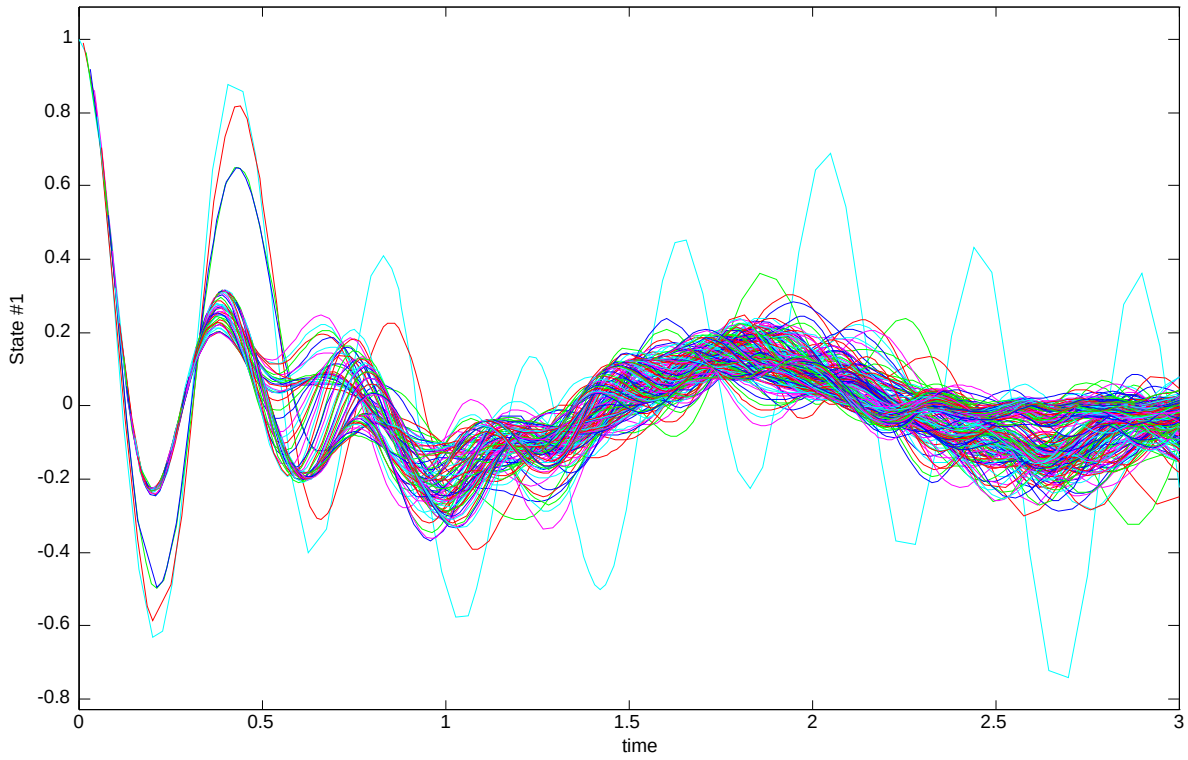
Wykresy na dwóch następnych stronach przedstawiają przebiegi zmiennej stanu i sterowania dla **modelu wahadła** przy gęstej dyskretyzacji sterowania (por. model w paragrafie 4.4.2). Zakłócenia są w pełni znane, lokalny horyzont sterowania jest zawsze równy t_f. Na stronie pierwszej zostały umieszczone wszystkie trajektorie przewidywane, na drugiej — zrealizowane. Wszystkie porównania odnoszą się do wyników z podrozdziału 4.4, więc całość opisywanych symulacji odbywała się bez uwzględniania opóźnień decyzyjnych.

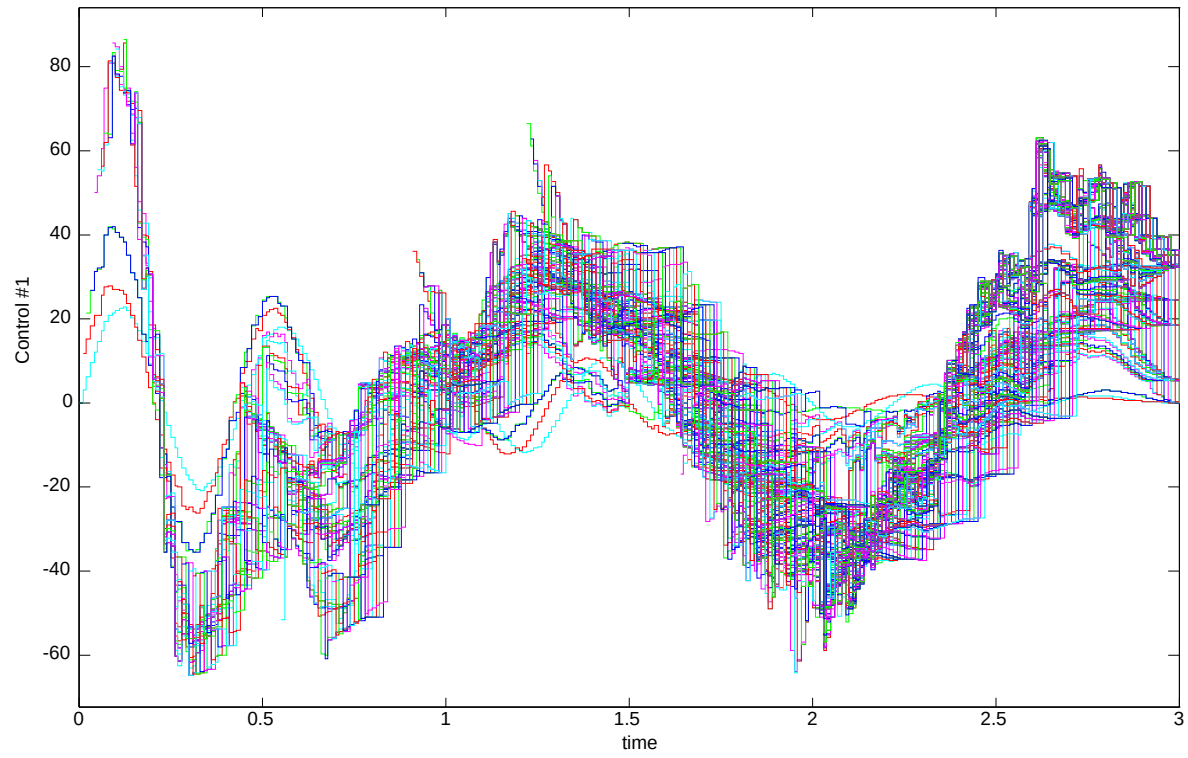
Zamiast oczekiwanej wiązki kolorowych trajektorii pokrywających się z trajektorią czarną widać, że każda przewidywana trajektoria stanu i każde sterowanie biegnie swoją drogą. Przyczyny tego mogą być następujące:

- w każdym kolejnym kroku zmienia się wymiar zadania optymalizacji, ale jej parametry pozostają takie same — może to skutkować osiągnięciem nieco innego punktu optymalnego;
- procedura całkująca ciągle jeszcze działa niepewnie — może być wrażliwa na niewielkie zmiany punktu startowego całkowania;
- wynik całkowania (trajektoria) jest aproksymowany łamaną — uzyskiwanie wartości pośrednich jest więc obarczone błędem zależnym od miejsca, w którym ma być zdjęta próbka.


State #1
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3


Control #1
time
80
60
40
20
0
-20
-40
-60
0
0.5
1
1.5
2
2.5
3


State #1
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3


Control #1
time
80
60
40
20
0
-20
-40
-60
0
0.5
1
1.5
2
2.5
3

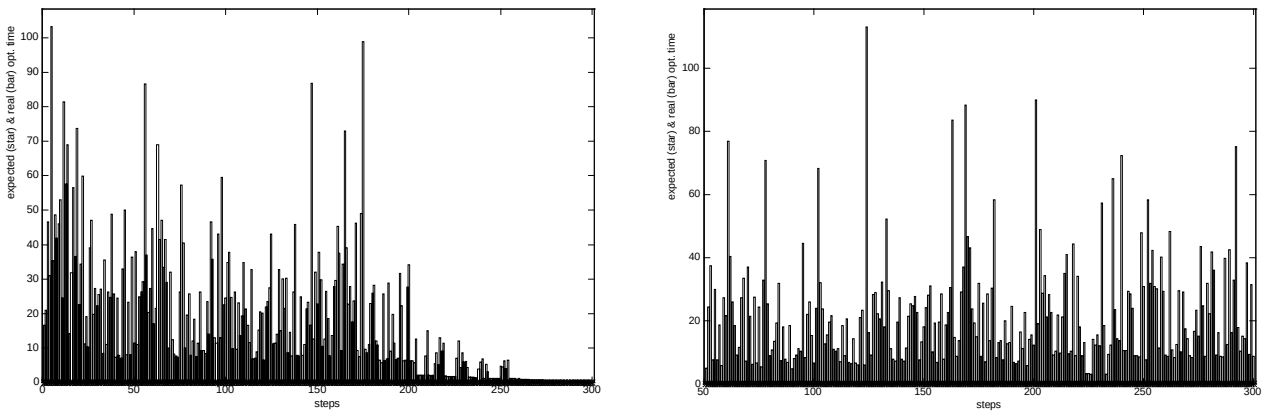
Rys. 46 Zrealizowane trajektorie stanu i sterowania dla modelu wahadła

Rys. 47 Przewidywane trajektorie stanu i sterowania dla modelu wahadła

Z natłoku informacji na opisywanych wykresach można wyłania się kilka budujących wniosków. Wystarczy spojrzeć na przewidywane zmienne stanu „odrywające się" od trajektorii zrealizowanej w przedziale $\langle 0{,}4;\ 0{,}6\rangle$. W zasadzie różnice między trajektoriami sąsiadującymi są niewielkie (i tak jest aż do t_f). Jednak rozbieżności się kumulują, co daje pod koniec symulacji szeroką wstęgę. Wstęga ta cały czas otacza dość ściśle trajektorię zrealizowaną. Nie zdarza się też (por. 46), żeby choć jedno sterowanie opadało w środku przedziału globalnego do zera. Ciekawe jest także to, że tych 300 przewidywanych sterowań przyjmuje na ostatnim etapie tylko 5 wartości.

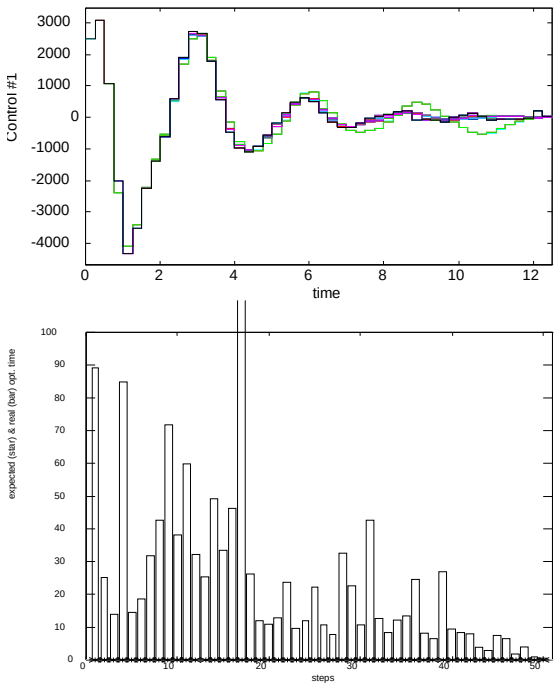
Jeszcze bardziej zaskakująco wypada porównanie czasów optymalizacji tu i w komplementarnej symulacji z 4.4.2. (48). Można by się spodziewać, że czasy te w pierwszym przypadku będą sukcesywnie malały na kolejnych etapach. I rzeczywiście: widać ich radykalne zmalenie, ale na zaledwie kilkunastu ostatnich etapach. Na pozostałych są one porównywalne z czasami dla horyzontu przesuwanego. Pod względem szybkości działania wersja algorytmu ze stałym horyzontem lokalnym nie ustępuje tej z horyzontem przesuwanym.

Czasy optymalizacji w symulacji z lokalnym horyzontem sterowania:
ustalonym przesuwanym



Rys. 48 Porównanie czasów optymalizacji dla modelu wahadła

Nie ustępuje, ale tylko pod względem szybkości. Podobne zestawienie wskaźników jakości wypada zdecydowanie (pogorszenie o 40□) na niekorzyść dla wersji z horyzontem ustalonym. Nie można zatem powiedzieć, by stosowanie algorytmu z ustalonym horyzontem lokalnym w jakikolwiek sposób polepszało sterowanie. (Zestawienie dokonane dla przypadku z niepełną znajomością zakłóceń potwierdziło tę opinię). Warto jednak sprawdzić, jak wnioski te mają się do innych, poznanych modeli.

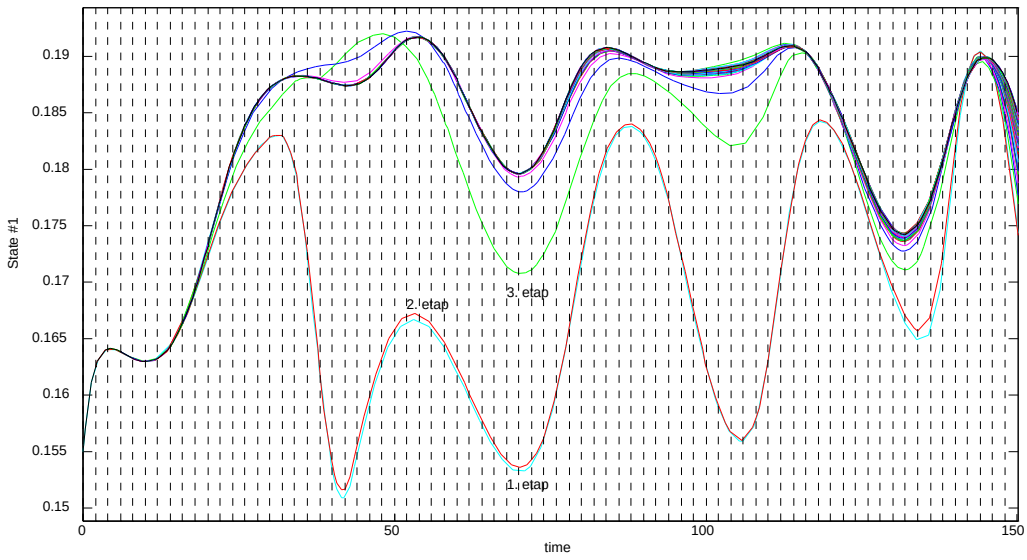


Rys. 49 Sterowanie i czasy optymalizacji przy ustalonym horyzoncie lokalnym (suwnica)

W przypadku **suwnicy** przewidywane przebiegi sterowań są bardziej zgodne; układają się zasadniczo w dwóch spójnych wiązkach (49). Czasy optymalizacji na poszczególnych etapach również maleją w miarę zbliżania się do t_f. Tendencja ta zaznacza się równie silnie na całym przedziale, nie tylko blisko końca. Niestety, wciąż trafiają się sytuacje, w których całkowanie trwa bezzasadnie długo. Taki przypadek można obserwować w kroku 17. Słupek wykracza poza wykres, gdyż optymalizacja w tym kroku wykonywała się aż przez 760*s*. Poza tym przypadkiem czasy obliczeń nie odbiegają od czasów dla przypadku ze zmiennym horyzontem lokalnym; są nawet nieco mniejsze. Ponownie, znaczne wydłużenie lokalnego przedziału sterowania na początku symulacji nie skutkuje drastycznym wydłużeniem czasów obliczeń.

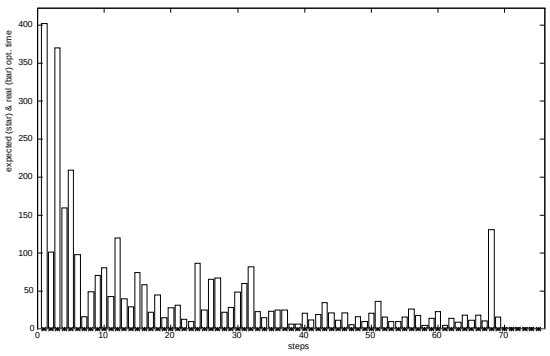
Uzyskane wskaźniki jakości są niemal identyczne, jak te z paragrafu 4.4.3. (Porównywano wyniki symulacji na platformie *Sun*.)

Sterowanie z ustalonym horyzontem lokalnym zdało się wywrzeć najmniejszy wpływ na model **reaktora** (50).



Rys. 50 Zrealizowane i przewidywane trajektorie stanu (model reaktora)

Uzyskany wskaźnik jakości był dobry (-0,1856), ale niewiele lepszy od wyników z podrozdziału 4.5 p. 6. Większość trajektorii przewidywanych pokrywała się ze zrealizowanymi. Kształty trajektorii przewidywanych na trzech pierwszych etapach różnią się od reszty ze względu na niekorzystny punkt startowy optymalizacji. (W miarę postępów symulacji początkowe przebiegi sterowań bazują na przebiegach zoptymalizowanych w poprzednim etapie.) Zestawienie czasów obliczeń w poszczególnych etapach (51) po raz pierwszy spełnia oczekiwania: sterowanie solidnie (i długo) optymalizowane w kilku pierwszych procentuje znikomymi czasami obliczeń w następnych.



Rys. 51 Czasy optymalizacji przy ustalonym horyzoncie lokalnym (reaktor)

Z przeprowadzonych analiz wynikają dwa główne wnioski. Po pierwsze, konsekwentna optymalizacja sterowania aż po t_f na każdym z etapów wcale nie powoduje poprawy jakości sterowania, nawet w sytuacji, gdy zakłócenia są w pełni znane. Dlatego, że długość lokalnego przedziału optymalizacji jest tak dobrana, by układ sterujący poradził sobie ze sprowadzeniem zmiennych stanu do wartości zadanych. Dalsze wydłużanie T_i^* nie jest potrzebne. Po drugie,

raz jeszcze została obnażona słabość algorytmu całkującego. Bo jak inaczej tłumaczyć to, że czasy optymalizacji maleją niekiedy dopiero na kilku ostatnich etapach□ Uważam, że czasy optymalizacji obserwowane do tej pory zależą głównie od tych sporadycznych sytuacji, gdy algorytm całkujący wręcz się zatrzymuje. (Przykład: pik na 51.)

5.3 Repetycja kontra PID

Poprzedni rozdział miał znaczenie raczej dydaktyczne, gdyż proponowane zmiany nie wpłynęły korzystnie na jakość sterowania i nie poszerzyły pola zastosowań algorytmu. Za to skłoniły do zastanowienia się nad przyczyną różnic między faktycznym zachowaniem się algorytmu a zachowaniem wynikającym z teorii. W tej części spróbuję dokonać porównania sterowania repetycyjnego ze sterowaniem za pomocą algorytmu PID. Jest to najprostszy i bardzo miarodajny sposób oceny jego przydatności.

Porównanie to będzie tendencyjne. Musi być, gdyż PID jest algorytmem o ugruntowanej, dobrej pozycji, mającym wiele zastosowań, wciąż modernizowanym. Jego prostota i szybkość działania już na wstępie deklasują algorytm repetycyjny. Dlatego poszukiwał będę prostych i często występujących przykładów, w których badany przeze mnie algorytm okaże się zdecydowanie lepszy od PID i na tyle szybki, że można poważnie zastanawiać się nad jego praktycznym zastosowaniem.

Wszystkie symulacje sterowania za pomocą regulatora PID zostały dokonane przez mały, pomocniczy program symulacyjny, napisany w stylu głównego programu do symulacji sterowania repetycyjnego. Nic nie stało na przeszkodzie, by wszystkich obliczeń dokonać np. w *Matlabie*; lepiej jednak zachować maksymalnie zbliżone do siebie warunki symulacji.

Potrójna inercja. Weźmy przedstawiany w podrozdziale 4.3 model obiektu w postaci trzech połączonych ze sobą modułów inercyjnych i zakłóceniach sinusoidalnych. Brakujące parametry dla algorytmu sterowania repetycyjnego zostały dobrane po zastosowaniu się do wypracowanej procedury projektowej. Nastawy PID zostały określone w eksperymencie Zieglera—Nicholsa. Zważywszy naturę obu algorytmów zdecydowałem się przyjąć dla sterowania repetycyjnego etap dłuższy (0,25s), zaś dla sterowania PID — krótszy (0,025s). Takie rozwiązanie służy obu algorytmom. Ponadto, algorytm repetycyjny zna w pełni zakłócenia.

Jeśli by nie uwzględniać opóźnień decyzyjnych, to w tej konfrontacji wygrywa algorytm sterowania repetycyjnego. Z trajektorii stanu (14) można odczytać sinusoidalny charakter zakłóceń. Algorytm repetycyjny radzi sobie z nimi o wiele lepiej (pomimo rzadszej sposobności do interwencji), gdyż zna ich trafną prognozę. W tym przypadku wystarczyły tylko dwa punkty dyskretyzacji sterowania na (dostatecznie długim) przedziale lokalnym.

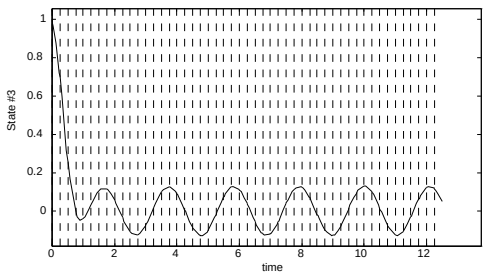
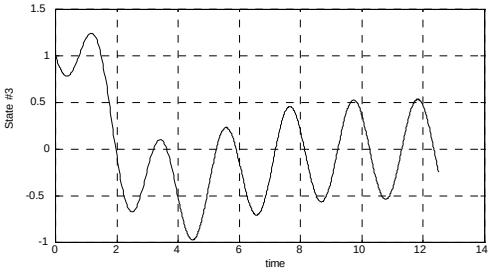
Przebieg zmiennej stanu x_3 przy sterowaniu algorytmem	
z powtarzaną optymalizacją	PID
	

Tabela 14 Jedna ze zmiennych stanu obiektu sterowanego algorytmem repetycyjnym, a następnie PID

Okazuje się również, że symulując działanie algorytmu repetycyjnego z uwzględnieniem opóźnień decyzyjnych, nie traci się na jakości sterowania. Na przykład, dla $R = 0{,}1$ wykonuje się tylko 4÷5 kroków optymalizacji w etapie, lecz wystarcza to, by osiągnąć taki sam (a czasami nawet lepszy) wskaźnik jakości. Dopiero skala czasu powyżej 0,2 skutkuje znaczną przypadkowością rozwiązań i zdecydowanie gorszymi wskaźnikami.

Pojedyncza inercja. Powyższy przykład zademonstrował siłę sterowania repetycyjnego, gdy zakłócenia są dobrze prognozowane. Następny pokaże ją w sytuacji, gdy do porządnego modelu liniowego wkrada się „drobna" nieliniowość. Oto model układu inercyjnego opisywany w rozdziale 4.1. Zastosowanie procedury projektowej prowadzi do konkretnego zestawu parametrów algorytmu repetycyjnego; natomiast regulator PID redukuje się do członu proporcjonalnego. Tym razem opóźnienia decyzyjne nie będą uwzględniane a długość etapu jest taka sama i równa 0,25. Zakłócenia są nieobecne.

Dopóki model (równania stanu) jest zgodny z pierwotnym (rozdz. 4.1), dopóty stany i sterowania zrealizowane przez oba algorytmy są niemal identyczne (15). Wystarczy jednak zmodyfikować równanie stanu tak, by $\dot{x} = -4x + m\mathbf{h}(m) + z$, gdzie

$$\mathbf{h}(m) = 1 + 0{,}7\left[\mathbf{exp}\left(-4(m-1)^2\right) + \mathbf{exp}\left(-4(m+1)^2\right)\right]$$

i wówczas czas regulacji dla PID znacznie się wydłuża (tabelka poniżej). Wynika to z tego, że PID nie jest predestynowany do zadań nieliniowych, w wersji podstawowej nie ma możliwości uwzględnienia dwóch „garbów" wprowadzonych przez funkcję $\mathbf{h}(m)$. Przyłożone sterowanie jest więc zawsze za duże — stąd oscylacje i długi czas oczekiwania na ustabilizowanie się stanu.

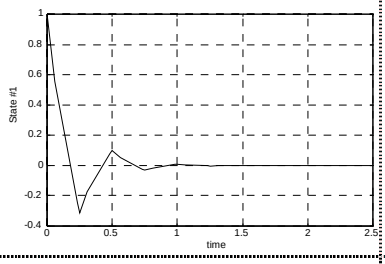
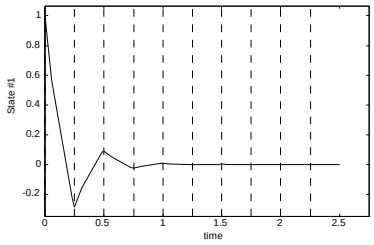
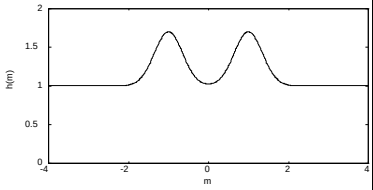
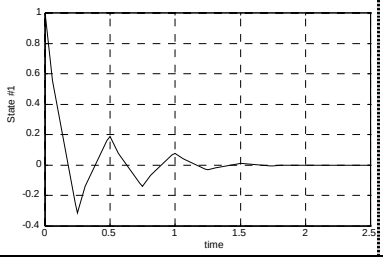
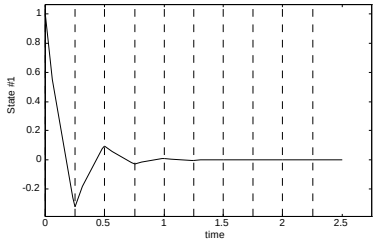
	Przebieg zmiennej stanu x_3 przy sterowaniu algorytmem	
Równania stanu	PID	repetycyjnym
pierwotne		
z funkcją $\mathbf{h}(m)$ 		

Tabela 15 Wrażliwość PID na nieliniowości modelu

Dwa powyższe przykłady miały na celu zasygnalizowanie, że istnieją sytuacje, w których sterowanie repetycyjne oddać może duże przysługi. Stworzone i opisywane przeze mnie środowisko symulacyjne, jakkolwiek powolne i zawodne, w tych przypadkach prawie nadaje się do zastosowania w praktyce. Skala czasu osiągnięta dla modelu trójinercyjnego (0,1) pomyślnie rokuje na przyszłość. Także w przykładzie ostatnim obliczenia przebiegały sprawnie a uzyskane wyniki zachęcają do popracowania nad kodem programu.

6. Podsumowanie

Celem opisanych badań było poznanie właściwości algorytmu sterowania z powtarzaną optymalizacją. Ze szczególną uwagą miały być potraktowane skutki stosowania nierównomiernej dyskretyzacji sterowania na przedziale lokalnym. Uciąglenie modelu obiektu pozwoliło dodatkowo na rzetelne zajęcie się opóźnieniami decyzyjnymi.

Stosowanie się do obecnych tendencji w inżynierii oprogramowania ukształtowało koncepcję środowiska do symulacji. Parametry procedur numerycznych są umieszczone w plikach, dzięki czemu czas, który zostałby poświęcony na oprogramowanie interfejsu użytkownika, spożytkowany był na wnikliwym badaniu działania algorytmu. Wszystkie symulacje dokonują się w programach napisanych w języku *C□□*, w standardzie *ANSI*, co gwarantuje ich szybkie działanie oraz przenośność. Wyniki symulacji są zapisywane w plikach w formacie *Matlaba*. Dane z tych plików są wizualizowane dzięki skryptom działającym w *Matlabie*. W ten sposób zostaje utrzymana całkowita przenośność oprogramowania oraz minimalny nakład pracy na efektowną i standaryzowaną prezentację wyników.

Owocem badań algorytmu repetycyjnego powinien stać się pewien zbiór zasad, pewna procedura projektowa pozwalająca dobrać parametry tak, by wyniki sterowania były jak najlepsze. Pod tym kątem były prowadzone eksperymenty symulacyjne. Wprowadzone liczne i różnorodne modele obiektów miały zapewnić, że uzyskana procedura będzie uniwersalna.

Algorytm został podzielony na poziomy, począwszy od najbardziej fundamentalnych (całkowanie), skończywszy na najbardziej ogólnych (strategia dyskretyzacji sterowania). Z każdym z nich powiązano zestaw parametrów. Ich dobór i synteza procedury projektowej odbywały się poprzez intensywne symulacje dla coraz wyższych poziomów algorytmu, przy czym modele były dobierane tak, by uświadomić wagę każdego poziomu.

Parametry dotyczące całkowania to dokładność ε i minimalny przyrost rejestrowany. Znaczenie tego pierwszego okazało się aż nadto istotne w dalszych badaniach. Dokładność całkowania przesądza bowiem o wszystkich innych wynikach. Zbyt mała powoduje „błądzenie" procedury optymalizacji; zbyt duża — utknięcie obliczeń lub, w najlepszym przypadku, bardzo duże opóźnienia decyzyjne. Wyszła na jaw niedoskonałość implementacji procedury całkowania: nieumiejętność radzenia sobie ze skokowymi zmianami sterowań i zakłóceń. Nie bez znaczenia okazała się platforma sprzętowa, na jakiej dokonuje się obliczeń. Te same symulacje, wykonywane poprawnie na *Sunie*, kończyły się niepowodzeniem na platformie *Intela*.

Procedura optymalizacji była strojona przez dobór skali $\mathbf{f}_0$ i tolerancji c gradientu funkcji celu. Przedstawiony przykład zwrócił uwagę także na inny parametr: dokładność w kryterium *stopu* ze względu na sterowa-

nie. Jest on niekiedy tak ważny, że dobór dwóch poprzednich staje się przy nim nieistotny. Nie rozwiązanym problemem pozostaje potrzeba dynamicznej zmiany maksymalnej liczby kroków optymalizacji i tolerancji w zadaniach z ustalonym horyzontem lokalnym.

Niezwykle istotny jest dobór długości lokalnego przedziału sterowania T_l^*. Musi być ona na tyle duża, by ukazać modułowi sterującemu istotę dynamiki obiektu. Spełnienie tego wymogu jest dla większości przypadków warunkiem funkcjonowania optymalizacji w ogóle, a dla niektórych (reaktor) jedynym istotnym uwarunkowaniem, by uzyskać dobre, stabilne sterowanie. Wszelkie próby sterowania z ustalonym horyzontem lokalnym ustępują w jakości algorytmowi wyjściowemu z dobrze dobranym T_l^*.

Istniały dwa powody wyjątkowego zainteresowania nierównomierną dyskretyzacją sterowania na przedziale lokalnym. Pierwszy, to możliwość zmniejszenia liczby punktów dyskretyzacji sterowania, a zatem skrócenia czasu optymalizacji bez pogorszenia wskaźnika jakości. Drugi, to sposobność obarczenia sterowań blisko horyzontu lokalnego większą „odpowiedzialnością". Tak postępując likwiduje się efekt zaniechania sterowania w końcowej części przedziału lokalnego.

Największy wpływ rozrzedzania dyskretyzacji obserwuje się w przypadku pełnej znajomości zakłóceń. Wówczas, w zależności od stosowanej strategii i od modelu, możliwa jest nawet trzykrotna poprawa wskaźnika jakości. Udało się też, w pewnych przypadkach, pięciokrotnie skrócić czas obliczeń — ale kryteria jakości i czasu obliczeń pozostają zasadniczo sprzeczne. Możliwe jest także zastosowanie funkcji kary □, ale nie przynosi ono poprawy jakości, a jedynie skrócenie czasów obliczeń. Ponadto, nie zawsze.

Seria symulacji „w czasie rzeczywistym" wykazała, że jest możliwe dalsze skrócenie czasów optymalizacji tak, że od zastosowania algorytmu w praktyce dzieli nas bardzo niewiele. Przyczyna sukcesu tkwi w tym, że największą poprawę sterowania liczoną na horyzoncie lokalnym uzyskuje się w kilku pierwszych krokach optymalizacji. Nakładając drastyczne ograniczenia na czas optymalizacji można niekiedy uzyskać bardzo dobry wskaźnik jakości przy skali czasu $R = 0{,}4$. Zastosowanie praktyczne jest więc w zasięgu ręki.

Okazuje się też, że wpływ opóźnień decyzyjnych na jakość wielce zależy od badanego modelu. Modele skomplikowane, o wielu inercjach są mniej wrażliwe na duże opóźnienia (rzędu długości etapu). Modele małe, którymi łatwo się steruje, gwałtownie reagują na jakąkolwiek zwłokę.

Można by pomyśleć, że te ostatnie mogą być sterowane z powodzeniem przez proste regulatory typu PID, skoro sterowanie repetycyjne jest takie powolne. Jednak kolejne symulacje pokazały, że wystąpienie nawet niedużej nieliniowości znacznie osłabia pozycję PID. Lecz by tego dokonać algorytm sterowania repetycyjnego musi działać znacznie, znacznie szybciej. Przyspieszenie obliczeń jest możliwe jeszcze na wiele sposobów. Po-

może temu nieuchronny wzrost szybkości komputerów, ale nie tylko. Poznawszy już wady zastosowanych algorytmów można pokusić się o ich ulepszenie (szczególnie dotyczy to całkowania). Ale algorytmy nie stanowią jedynej rezerwy. Warto zastosować inne zabiegi, by skrócić obliczenia: jednym z nich jest opisywany wcześniej pomysł przybliżania przebiegów prostokątnych trapezowymi.

Możliwe jest ponadto usprawnienie działań na wektorach i macierzach — operacji najprostszych i najczęściej wykonywanych. W wykonaniu przemysłowym wymiar zadania i definicja problemu są znane, nie ma więc potrzeby częstego przydzielania i zwalniania pamięci operacyjnej a równania stanu (oraz inne wyrażenia) można wkompilować.

Obecna postać programu jest podyktowana wymaganiami uniwersalności, dydaktyki, łatwości uruchamiania i śledzenia błędów. Wyniki symulacji dają wszakże nadzieję, że znajdzie on zastosowanie praktyczne — wówczas należy pomyśleć o wersji dedykowanej dla konkretnego problemu. Mam nadzieję, że dalsze badania zagadnienia sterowania repetycyjnego przyczynią się do pogłębienia wiedzy o algorytmie, usystematyzowania uzyskanych wyników. Ranking dokonany w podrozdziale 5.3 stanowi dostateczną argumentację w sporze o przydatność sterowania z powtarzaną optymalizacją.

Oznaczenia

We wzorach stosowana jest pisownia *kursywą*. Odstępstwa od tej reguły są następujące:

- *macierze i wektory są oznaczane czcionką* **wytłuszczoną**,
- *funkcje są oznaczane czcionką* prostą,
- *symbole o podobnym znaczeniu, ale mające charakter cząstkowy bądź przejściowy są niekiedy odróżniane symbolem* $*$ *(gwiazdki).*

Oto objaśnienia najczęściej używanych oznaczeń:

c	tolerancja optymalizacji;
$\mathbf{F}$	uogólnione równanie stanu;
$\mathbf{f}$	równanie stanu;
$\mathbf{f}_0$	podcałkowy wskaźnik jakości;
$\mathbf{f}_k$	równanie stanu obiektu zdyskretyzowanego;
$\mathbf{g}$	równanie wyjścia obiektu;
J	wskaźnik jakości;
J_k	wskaźnik jakości za etap k;
J^*, J_k^*	cząstkowy wskaźnik jakości;
$\mathbf{M}$	funkcja wyznaczająca zbiór sterowań dopuszczalnych;
$\boldsymbol{m}$	trajektoria sterowania, wektor wejść sterujących;
$\boldsymbol{m}_i^*$	sterowanie na przedziale lokalnym;
N	liczba etapów sterowania;
N^*	liczba lokalnych punktów dyskretyzacji sterowania;
n	wymiar wektora zmiennych stanu;
p	wymiar wektora wejść niesterowanych;
□	funkcja kary za stan w horyzoncie lokalnym;
R	skala czasu;
r	wymiar wektora wejść sterowanych;
T	długość etapu sterowania;
T_s	zbiór punktów pomiaru stanu;
T^*, T_i^*	długość lokalnego przedziału sterowania;
t_0, t_f	początek, koniec sesji sterowania;
t_i	punkty pomiaru stanu;
$t_{i\,\mathbf{max}}$	maksymalny czas trwania optymalizacji;
$\boldsymbol{U}$	uogólnione sterowanie;
□	uogólniony stan;
$\boldsymbol{x}$	trajektoria stanu, wektor stanu;
$\boldsymbol{x}_i^*$	przewidywana trajektoria stanu na przedziale lokalnym;
$\boldsymbol{y}$	wyjście obiektu;
$\hat{\boldsymbol{y}}$	wartość zadana wyjścia;
$\boldsymbol{z}$	trajektoria zakłóceń, wektor wejść niesterowanych;
$\boldsymbol{z}^*, \boldsymbol{z}_i^*$	prognoza zakłóceń;
Δt_i	czas trwania optymalizacji;
δt_i	przewidywany czas trwania optymalizacji;
δx	minimalny przyrost zapisywany;
ε	dokładność całkowania;
φ	uogólniony wskaźnik jakości;
Λ	równanie stanu sprzężone;
λ	sprzężona zmienna stanu;
μ	reguła decyzyjna.

Literatura

1 *Computational Optimal Control* ed. R. Bulirsch, Birkhäuser Verlag 1994r., str. 163-175

2 *Numerical Recipes in C*, Cambridge University Press, 1993r.

3 J. Pułaczewski *Zasady automatyki*, WNT, Warszawa 1974r.

4 K.J. Kurman *Teoria regulacji. Podstawy, analiza, projektowanie*, WNT Warszawa 1975r.

5 M. Siomak *Synteza optymalnych reguł decyzyjnych*, praca dypl. w Inst. Automatyki, 1996r.

6 P. Jakomulski *Repetitive Control Algorithms and Software*, praca dypl. w Inst. Automatyki, 1993r.