Metody numeryczne w elektrotechnice

Ćwiczenie 9 — Optymalizacja i minimalizacja funkcji

Dariusz Borkowski

13 maja 2012

Wstęp

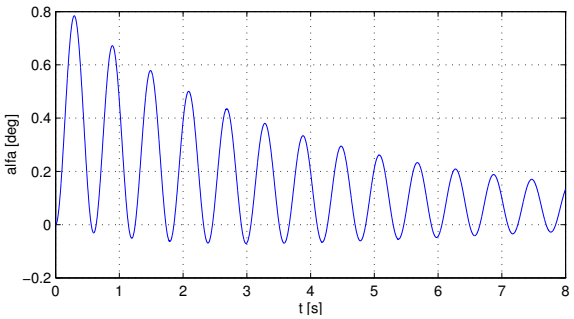
Na zajęciach poznasz typowe problemy związane z minimalizacją funkcji jak również kilka typowych zastosowań minimalizacji.

1 Szukanie minimum funkcji jednej zmiennej – 3 pkt

Podczas rozruchu prądnicy napędzanej silnikiem wyznaczono kąt skręcenia wału łączącego prądnicę i silnik. Ze względu na możliwość ukręcenia wału konieczne jest określenie maksymalnego i minimalnego kąta skręcenia i sprawdzenie czy kąty te nie przekraczają wartości dopuszczalnych. Przebieg kąta skręcenia α wału w funkcji czasu jest opisany zależnością

$$\alpha(t) = A + Be^{-Ct} - De^{-Et}\left(cos(\Omega t) - Fsin(\Omega t)\right) \tag{1}$$

i został pokazany na rysunku 1. Wartosci parametrów są następujące: $A = 0,0478$, $B = 0,376$, $C = 0,3823$, $D = 0,424$, $E = 0,1923$, $F = 0,01398$, $\Omega = 10,51$.



Rysunek 1: Przebieg czasowy skręcenia wału podczas rozruchu układu silnik–prądnica.

Wyznacz wartość minimalną i maksymalną kąta skręcenia $\alpha(t)$ w przedziale czasu od 0 do 8 sekund przy pomocy funkcji biblioteki GSL. Szkielet programu znajduje się w pliku `lab09z01.c`. Dodaj brakujące części programu i dla zadanego w programi przedziału poszukiwania minimum $(0,1$ do $6,8)$ przeprowadź minimalizację metodami Brenta i złotego podziału (ang. *golden section*). Zauważ, że obie metody, pomijając różną ilość iteracji znajdują inne minimum. Czy, któreś z nich jest minimum globalnym? Teraz wykonaj maksymalizację zmieniając znak funkcji celu przez zmianę wartości zmiennej `params.Sign` z 1 na -1. Działanie programu kończy się błędem, w którym GSL informuje użytkownika, że zadany przedział poszukiwania minimum nie zawiera minima, co jest nieprawdą, gdyż zawiera ich kilka. Na rysunku 1 widać, że funkcja celu 1 posiada wiele ekstremów (zarówno minimów jak i maksimów). Jak widzisz, obie testowane metody są jedynie metododami poszukiwania minimum lokalnego. Aby metody te działały dobrze należy tak dobrać krańce przedziału poszukiwania minimum, żeby wewnątrz tego przedziału znajdowało się dokładnie jedno minimum. Obiema w.w. metodami sprawdź wartośc minimalną i maksymalną skręcenia $\alpha(t)$ po uprzednim ograniczeniu obszaru poszukiwań, tak aby zawierał tylko jedno ekstremum globalne. Porównaj ilość iteracji obydwu metod.

Sposób korzystania z funkcji minimalizujących funkcję celu jednej zmiennej w GSL-u jest podobny do korzystania z funkcji znajdujących pierwiastki funkcji. Algorytmy poszukiwania minimum poszukują minimum w ograniczonym przedziale i wraz ze zbliżeniem się do minimum zawężają szerokość przedziału. W większości przypadków jesli w przedziale znajduje się więcej niż jedno minimum, nie otrzymamy ostrzeżenia. Znajdowane jest tylko jedno minimum. Obliczenia mają następujący przebieg:

1. inicjalizacja stanu minimizara s dla algorytmu T
2. aktualizacja (update) stanu minimizera s dla aktualnej iteracji
3. sprawdzenie s pod kątem zbieżności i jeśli potrzeba powtórzenie iteracji

Poniżej znajduje się opis niezbędnych funkcji:

`gsl_min_fminimizer * gsl_min_fminimizer_alloc (const gsl_min_fminimizer_type * T)` zwraca wskaźnik do nowego minimizera wykorzystującego algorytm T. Przykładowe użycie to przygotowanie minimizera s wykorzystującego metodę złotego podziału odcinka:
`const gsl_min_fminimizer_type * T = gsl_min_fminimizer_goldensection;`
`gsl_min_fminimizer * s = gsl_min_fminimizer_alloc (T);` Sprawdź w dokumentacji GSL-a jakie inne algorytmy są dostępne.

`int gsl_min_fminimizer_set (gsl_min_fminimizer * s, gsl_function * f, double x_minimum, double x_lower, double x_upper)` funkcja inicjalizuje minimizer s dla funkcji celu f i przedziału poszukiwań od x_lower do x_upper.

`void gsl_min_fminimizer_free (gsl_min_fminimizer * s)` zwalnia pamięć wykorzystaną przez minimizer s.

`int gsl_min_fminimizer_iterate (gsl_min_fminimizer * s)` funkcja wykonuje pojedynczą iterację minimizera s. W przypadku problemów zwracane sa kody błędu GSL_EBADFUNC lub GSL_FAILURE. Minimizer s przechowuje informacje o postepie obliczeń. Do uzyskania tych informacji służą poniższe funkcje:

`double gsl_min_fminimizer_x_minimum (const gsl_min_fminimizer * s)` zwraca obecną estymatę położenia minimum.

`double gsl_min_fminimizer_x_upper (const gsl_min_fminimizer * s)` zwraca nową wartość dolnego końca przedziału poszukiwań minimum.

`double gsl_min_fminimizer_x_lower (const gsl_min_fminimizer * s)` zwraca nową wartość górnego końca przedziału poszukiwań minimum.

`double gsl_min_fminimizer_f_minimum (const gsl_min_fminimizer *s)` zwraca wartość funkcji celu dla obecnej estymaty położenia minimum.

`int gsl_min_test_interval (double x_lower, double x_upper, double epsabs, double epsrel)` sprawdza zbieżność w przedziale od x_lower do x_upper z błędem absoultnym epsabs i względnym epsrel. Jeśli zadana dokładność zostanie osiągnięta to funkcja zwraca GSL_SUCCESS, w przeciwnym razie zwraca GSL_CONTINUE. Przykład wykorzystania to `status = gsl_min_test_interval (N_lo, N_hi, 1e-4, 1e-4)`.

GSL udostępnia również metody minimalizacji funkcji wielu zmiennych.

2 Identyfikacja parametrów obiektu dynamicznego przez dopasowanie krzywej do punktów czyli regresję nieliniową – 2 pkt

Na rysunku 2 pokazano zarejestrowaną odpowiedź skokową obiektu oscylacyjnego drugiego rzędu. Odpowiedź składa się z N próbek wartości wyjścia y_i w chwilach czasowych t_i i zawiera pewną zawartość zakłóceń losowych. Zadanie identyfikacji parametrów obiektu można sprowadzić do dobrania takich parametrów A, B, C, D, E modelu

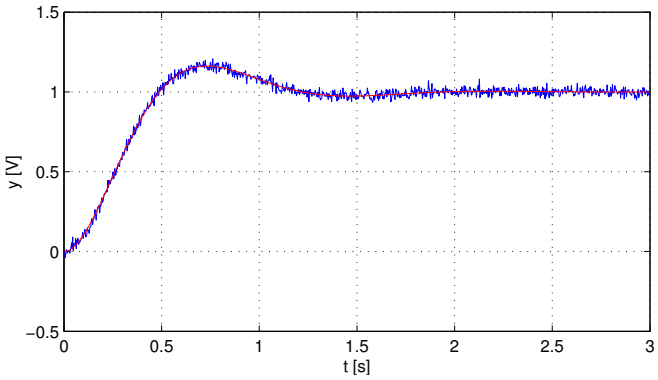
$$ym(t) = A - Be^{-Ct}sin(Dt + E) \tag{2}$$

dla których jego odpowiedź jest najlepiej, w sensie najmniejszej sumy kwadratów, dopasowana do zarejestrowanej odpowiedzi obiektu. Jest to w pewnym sensie aproksymacja średniokwadratowa, przy czym funkcja aproksymujaca jest nieliniowa względem parametrów. Dobór parametrów modelu odbywa się poprzez minimalizację wskaźnika jakości J, będącego sumą kwadratów odchyłek odpowiedzi modelu od odpowiedzi

obiektu. Wskaźnik jakości nazywany jest również funkcją celu (ang. *goal function*) lub funkcją błędu. Zadanie minimalizacji można zapisac jako

$$\min_{\boldsymbol{x}} J(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) = \min_{\boldsymbol{x}} \sum_{i=1}^{N} \left(ym(t_i, x) - y_i\right)^2 \tag{3}$$

gdzie $\boldsymbol{x}$ jest wektorem poszukiwanych parametrów $\boldsymbol{x} = [\ A \quad B \quad C \quad E \quad D\ ]^T$.



Rysunek 2: Zarejestrowana odpowiedź obiektu i dopasowana do niej odpowiedź modelu.

Do wyznaczania parametrów można wykorzystać procedury optymalizacyjne `lsqnonlin` i `fminunc` zaimplementowane w Optimization Toolbox Matlaba. Zapoznaj się z opisem tych funkcji. Procedura `fminunc` implementuje wiele różnych algorytmów (m.in. BFGS, DFP) minimalizacji bez ograniczeń. Funkcja opisująca wskaźnik jakości ma zwracać sumę kwadratów odchyłek $\sum_{i=1}^{N} \left(ym(t_i, x) - y_i\right)^2$. Procedura `lsqnonlin` wykorzystuje metodę minimalizacji Levenberga–Marquardta, które jest przeznaczona jedynie do rozwiązywania problemów najmniejszej sumy kwadratów. Z tego powodu funkcja opisująca wartość wskaźnika jakości ma zwracać wektor odchyłek $\boldsymbol{ym} - \boldsymbol{y}$, a nie jak poprzednio sumę kwadratów odchyłek.

Dla przykładu podaję sposób wywołania funkcji `lsqnonlin`:
`x = lsqnonlin('wsk_jakosci', x0, lb, ub, options, t, y)`, gdzie `'wsk_jakosci'` jest nazwą funkcji zwracającej wartośc funkcji celu, `x0` jest punktem startowym, `lb`, `ub` to wektory opisujące dopuszczone przedziały zmienności wyznaczanych parametrów `x`, `options` to struktura zawierająca opcje pracy procedury minimalizacyjnej (np. zadane dokładności, kryteria zakończenia, wybór algorytmu), `t`, `y` to opcjonalne parametry funkcji celu lub modelu. Jeśli zamiast wektorów `lb`, `ub` podasz `[]`, `[]` to wymusisz minimalizację bez ograniczeń. Jeśli zamiast `options` podasz `[]` to zostaną użyte domyślne ustawienia procedur. Ostatnie dwa parametry `t`, `y` nie sa obowiązkowe, ale czesto bywają potrzebne, może ich być więcej niż 2. `x` jest wektorem poszukiwanych parametrów i jest zwracany dopiero po zakończeniu wszystkich iteracji.

Strukturę zawierającą opcje minimalizacji tworzy się za pomocą funkcji `optimset` np. `options = optimset('MaxIter',500)`. Poszczególne elementy stuktury `options` można potem modyfikować w nastepujacy sposób: `options.MaxIter = 600`. Zapoznaj się z opisem funkcji `optimset` oraz zestawem możliwych ustawień.

Wyznacz w Matlabie wartości parametrów modelu. Niezbędne będzie do tego napisanie programu głównego oraz dwóch funkcji Matlaba w osobnych plikach: (1) zwracającej wartość minimalizowanej funkcji celu, (2) zwracającej wartość wyjścia modelu. Program głowny używa jednej z opisanych wyżej funkcji `lsqnonlin` lub `fminunc`, które wywołują funkcję (1), która wywołuje funkcję (2). Program ma rysować odpowiedź modelu ym na tle zarejestrowanej odpowiedzi obiektu y za pomocą polecenia `drawnow`. Dane (wektory wartości i czasu) znajdują się w pliku `lab09z02.mat`.

Minimalizowana funkcja celu zależy od pięciu parametrów. Ustal punkt startowy `x0` czyli początkowe wartości wszystkich parametrów równe 1. Włącz diagnostykę i wypisywanie informacji o postępie obliczeń w kolejnych iteracjach (opcje `options.Diagnostics` i `options.Display`). Przeprowadź minimalizację metodą Levenberga–Marquardta oraz metodami zmiennej metryki BFGS i DFP (opcja `options.HessUpdate`). Zapisz ilość iteracji i ilość obliczeń funkcji celu potrzebnych do osiągnięcia minimum oraz wartość funkcji celu w ostatniej iteracji. Który warunek zakończenia obliczeń został spełniony w różnych metodach? Wyjaśnij co znaczą poszczególne informacje zwracane przez procedurę optymalizacyjną w kolejnych iteracjach. Sprawdź czy wybór punktu startowego równego `[5 5 5 5 5]'` wpływa na

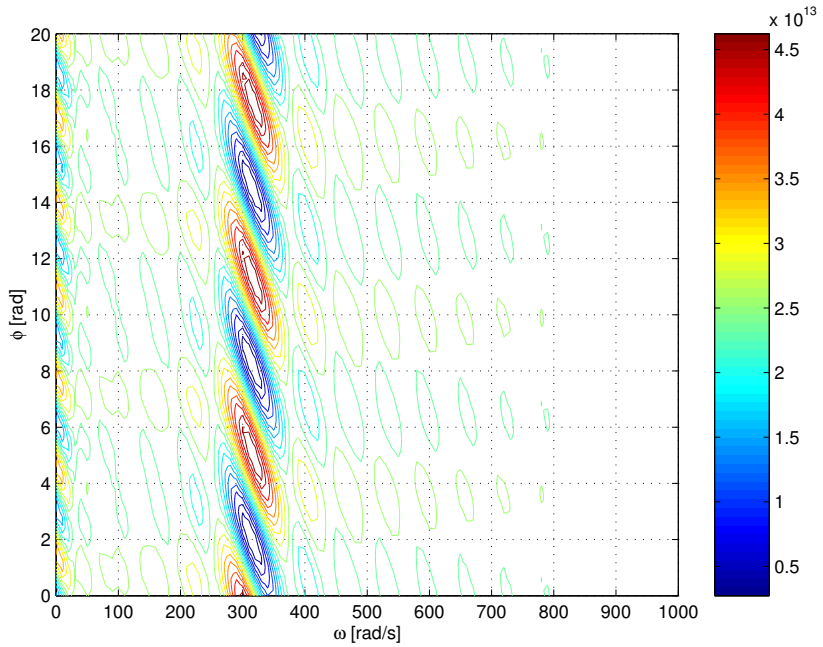
ilośc iteracji oraz na wartości wyznaczanych parametrów. Jeśli tak, to dlaczego pomimo różnych wartości parametrów lub parametru otrzymujemy identyczną odpowiedź modelu jak przy poprzednim punkcie startowym `[1 1 1 1 1]'`?

3 Wyznaczanie fazy i częstotliwości przebiegu okresowego, minimalizacja z ograniczeniami, pochodne funkcji celu – 3 pkt

Przykładem zadania optymalizacji jest estymacja fazy i częstotliwości podstawowej harmonicznej na podstawie rejestracji odkształconego przebiegu okresowego. Tu również stosowanym kryterium jest suma kwadratów odchyłek przebiegu przybliżającego przebieg zarejestrowany od przebiegu zarejestrowanego. Jeśli przyjąć znaną amplitudę A sygnału, to optymalizacja polega na znalezieniu minimum funkcji celu dwóch zmiennych:

$$S(\omega, \varphi) = \sum_{i=1}^{N} (y_i - A\sin(\omega t_i + \varphi))^2 \tag{4}$$

gdzie ω — pulsacja sygnału, φ — faza sygnału względem początku próbkowania, y_i — i–ta próbka czasowa zarejestrowanego sygnału w chwili t_i. Jako że funkcja celu (4) jest funkcją jedynie dwóch zmiennych, to można ją przedstawić w postaci wykresu (rysunek 3) i analizować wizualnie przebieg optymalizacji.



Rysunek 3: Wartości funkcji celu (4).

Funkcja celu jest dana wzorem analitycznym (4) co pozwala na wykorzystanie informacji o pochodnych funkcji celu w niektórych (gradientowych) metodach optymalizayjnych[1].

W ćwiczeniu tym sprawdzisz m.in. jak wykorzystanie analitycznej postaci pochodnych wpływa na zbieżność metod optymalizacyjnych. Sprawdzisz również niepożądane skutki istnienia wielu minimów w obszarze poszukiwań minimum.

Matlab udostępnia wiele metod optymalizacyjnych. My skoncentrujemy się na dwóch z nich. `fminunc` wykorzystana w poprzednim zadaniu procedura minimalizacji bez ograniczeń oraz `fmincon` procedura minimalizacji z ograniczeniami. Obie, pomimo odmiennej składni wywołania, pozwalają na podanie analitycznie wyznaczonego gradientu:

$$G(\omega, \varphi) = \begin{bmatrix} \dfrac{\partial S}{\partial \omega} \\ \\ \dfrac{\partial S}{\partial \varphi} \end{bmatrix} \tag{5}$$

[1] W przypadku gdy nie dysponujemy zależnością analityczną gradientu (np. gdy znamy funkcję celu w postaci zbioru jej wartości), wtedy pochodne funkcji wyznacza się na podstawie ilorazów różnicowych

Metody te mogą wykorzystywać algorytmy dużej skali (bez minimalizacji kierunkowej, z której rezygnuje się ze względu na długi czas obliczeń w problemach o dużej liczbie wymiarów) lub algorytmy średniej skali (wykorzystujące minimalizację kierunkową).

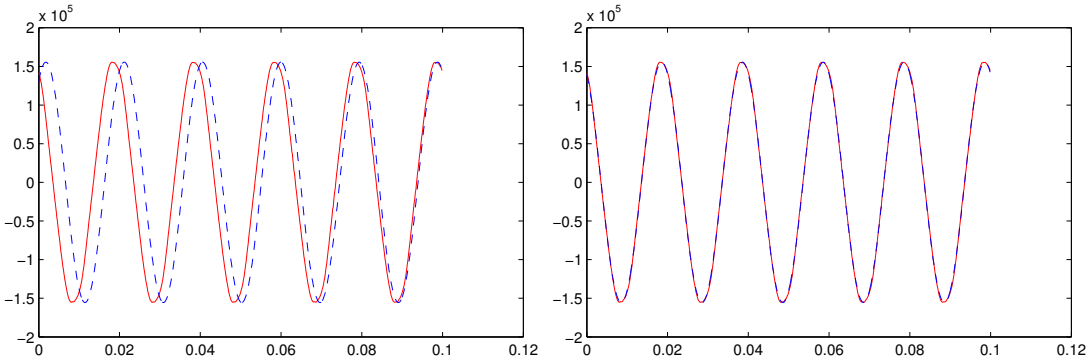
Matlabowskie metody optymalizacji dobierają stosowany algorytm na podstawie podanych opcji optymalizacji, jednak w zgodzie z wymaganiami algorytmu, co czasem nie jest zgodne z wolą użytkownika. Wynika to z faktu, iż niektóre z metod nie pozwalają na rozwiązanie problemów pewnych klas lub użytkownik nie dostarczył wymaganych informacji. Np. wybór algorytmu stosowanego przez polecenie `fmincon` zależy od rodzaju podanych ograniczeń oraz dostępność analitycznego gradientu funkcji celu.

`fminunc` oraz `fmincon` automatycznie wybierają najszybsze algorytmy dużej skali (odmiany metody regionu zaufania, opisanej w wykładzie), jeśli użytkownik dostarczy gradient w postaci analitycznej i ustawił opcję `GradObj='on'` oraz nie ustawił opcji `LargeScale='off'`. Gdy Matlab nie może zastosować najlepszego algorytmu, wtedy zmienia metodę na inną i zgłasza informację o wybranej metodzie oraz przyczynie takiego wyboru. Jeśli mimo dostarczenia gradientu chcemy wykorzystać algorytm średniej skali trzeba ustawić w opcjach `options.Large.Scale = 'off'`. Możliwe jest także dostarczenie przez użytkownika analitycznej postaci hesjanu, o czym informujemy procedurę włączając `options.Hessian = 'on'`.

Wybór konkretnego algorytmu średniej skali zależy od wartosci `options.HessUpdate`, która może przyjąć wartości `bfgs`, `dfp`, `steepdesc`, gdzie dwie pierwsze to metody zmiennej metryki, a ostatnia to najprostsza metoda największego spadku, charakteryzująca się najwolniejszą zbieżnością ze względu na niewykorzystywanie Hesjanu. Opcja `options.HessUpdate` ma zastosowanie tylko w algorytmach średniej skali.

Szkielet programu znajduje się w pliku `lab09z03.m`. W pliku `lab09z03.mat` zapisane zostało około 5 okresów napięcia fazy R pola 110 kV pewnej krakowskiej rozdzielni NN/SN. Znajdują się w nim wektory wartości napięcia y i czasu t. Napisz funkcję zwracającą wartość funkcji celu $S(\omega, \varphi)$ oraz jej gradient $G(\omega, \varphi)$ w następującej kolejności `[S, G] = function(x,t,y)`. Przyjmij amplitudę A równą maksymalnej wartości sygnału.

Do wyznaczenia analitycznej postaci gradientu możesz wykorzystać Symbolic Math Toolbox (patrz poprzednie zajęcia) lub wyznaczyć wzory na pochodne ręcznie. Do funkcji obliczającej wartość funkcji celu dopisz rysowywanie przebiegów zarejestrowanego i przybliżającego w każdej iteracji jak na rysunku 4 (patrz poprzednie zadanie). Na końcu programu głównego dodaj rysowanie przebiegów wykresu funkcji celu (polecenie `contour`) jak na rysunku 3. Jeśli w każdym wywołaniu funkcji wyznaczającej funkcję celu dopiszesz odczyt (`load`), a następnie zapis (`save`) macierzy złożonej z wektorów wyznaczaych parametrów w kolejnych iteracjach, to będziesz mógł wyrysować na w.w. wykresie kolejne kroki procedury optymalizacyjnej.



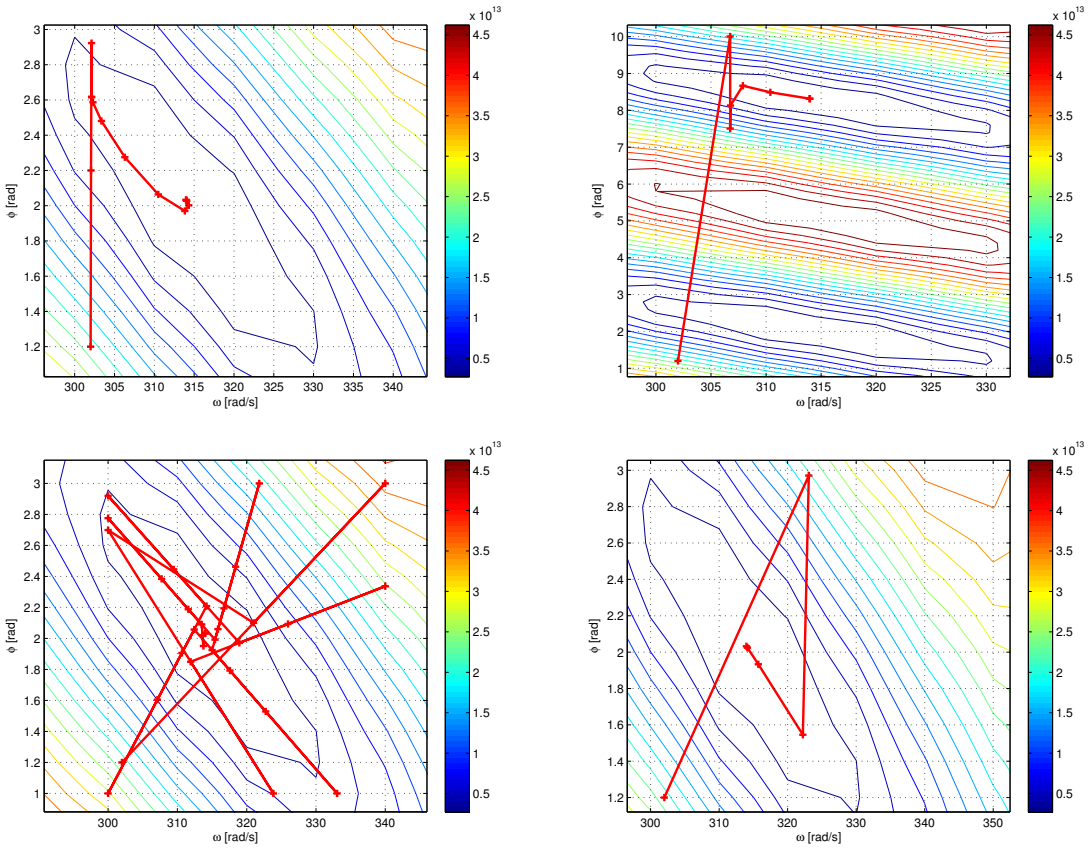
Rysunek 4: Przykładowe przybliżenia (przerywana) sygnału zarejestrowanego (ciągła) w trzeciej i ostatniej iteracji.

Przyjmij punkt startowy $x_0 = [\ \omega_0 \quad \varphi_0\ ]^T = [\ 302 \quad 1,2\ ]^T$. Włącz następujące opcje procedury optymalizacyjnej: diagnostyka, wyświetlanie informacji o iteracjach, analityczny gradient funkcji, algorytm średniej skali, metoda największego spadku. Przeprowadź minimalizację z wykorzystaniem procedury `fminunc`. Otrzymane wyniki nie są poprawne, a zastosowana metoda przeskakuje tam i z powrotem pomiędzy różnymi dolinami nie osiągając minimum. Wynika to z faktu wyboru kierunku najlepszego spadku, bez uwzględnienia informacji o lokalnej wypukłości funkcji zawartej w Hesjanie. Na wykresie funkcji celu można zobaczyć, że algorytm zbłądził, mimo iż punkt startowy znajdował się blisko właściwego minimum.

Poeksperymentuj z różnymi innymi punktami startowymi i zobacz, że wyniki nie zawsze są poprawne. Przetestuj lepsze algorytmy średniej skali (DFP, ten ma kiepską zbieżność bo krąży wokół minimum, lub

BFGS najlepszy skali średniej). Porównaj ilość obliczeń funkcji celu przy gradiencie analitycznym lub przybliżonym numerycznie. Na koniec porównaj liczbę iteracji w metodzie BFGS z liczbą iteracji algorytmu dużej skali (trust-region). Czy w lepszych metodach wyznaczona częstotliwość jest poprawna? Czy wyznaczone przesunięcie fazowe zostało wyznaczone prawidłowo, uzasadnij?

Teraz wykorzystaj procedurę `fmincon` minimalizacji z ograniczeniami do rozwiązania tego samego problemu. Na początek zastosuj algorytm średniej skali, a następnie dużej skali (trust-region-reflective). Tu nie ma już zastosowania opcja `options.HessUpdate`, bo w skali średniej stosowana jest metoda SQP (opisana w wykładzie). Ograniczenia dobierz na podstawie wizualnej oceny położenia minimum na wykresie funkcji celu. Aby mieć pewność, że algorytm będzie zbieżny do właściwego minimum, obszar poszukiwań powinien być tak dobrany, żeby funkcja celu w tym obszarze była unimodalna czyli zawierała tylko jedno minimum. Sprawdź czy wyłączenie algorytmu dużej skali lub wyłączenie gradientu wpływa teraz na poprawność wyników. Porównaj czas obliczeń, liczbę iteracji i obliczeń funkcji celu w powyższych warunkach. Zobacz (w powiększeniu) kroki algorytmu dużej i średniej skali.



Rysunek 5: Przykład kolejnych iteracji w minimalizacji funkcji celu z zadania 3 różnymi metodami: po lewej algorytmy średniej skali, po prawej algorytmu dużej skali, u góry minimalizacja bez ograniczeń, a u dołu z ograniczeniami, w każdym przypadku stosowany jest inny algorytm.

Na następne zajęcia

Algorytmy genetyczne.

Literatura

[1] Guziak T., Kamińska A., Pańczyk B., Sikora J. *Metody numeryczne w elektrotechnice* Wydawnictwo Politechniki Lubelskiej, Lublin 2002.

[2] Romuald Wit *Metody programowania nieliniowego : minimalizacja funkcji gładkich* Wydawnictwa Naukowo–Techniczne, Warszawa 1986.