Metody numeryczne w elektrotechnice

Ćwiczenie 8 — Rozwiązywanie równań i układów równań nieliniowych

Dariusz Borkowski

4 maja 2012

Wstęp

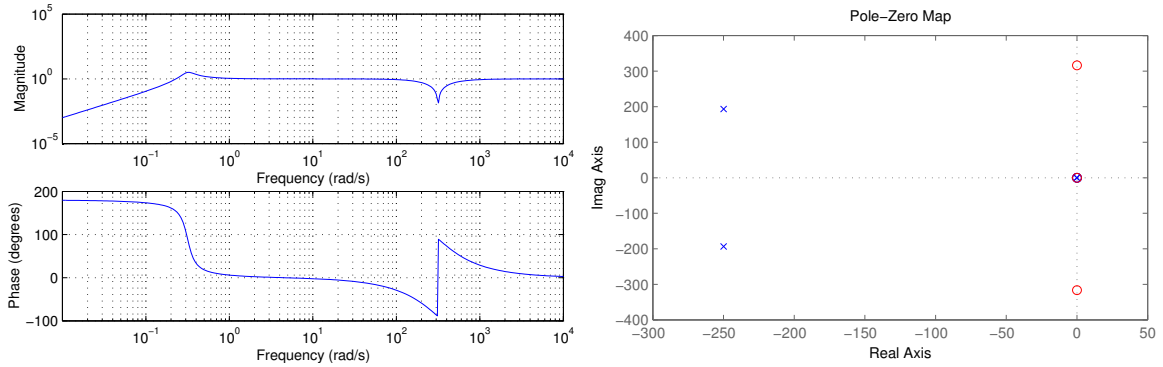
Na zajęciach zobaczycie jak wykorzystać metody poszukiwania miejsce zerowych funkcji do badania stabilnosci filtrów, rozwiązywania układów zawierających elementy nieliniowe (np. diody) oraz oceny czasu potrzebnego na zebranie odpowiedniej ilości funduszy na koncie bankowym.

1 Badanie stabilności filtrów przez znalezienie biegunów transmitancji – 2 pkt

Do miernika THD zaprojektowano analogowy filtr pasmowozaporowy 4–go rzędu (rys. 1) o transmitancji

$$G(s) = \frac{L(s)}{M(s)} = \frac{0,0001s^4 + 10s^2}{0,0001s^4 + 0,05s^3 + 10s^2 + s + 1} \tag{1}$$

służący do tłumienia częstotliwości sieciowej 50 Hz. Należy sprawdzić czy filtr ten jest stabliny. Filtr stabilny będzie jeśli wszystkie jego bieguny znajdują się w lewej części płaszczyzny zespolonej, t.j. części rzeczywiste wszytkich biegunów są ujemne. Położenie zer transmitancji nie wpływa na stabilność. Bieguny transmitancji to pierwiastki wielomianu $M(s)$ znajdującego się w mianowniku transmitancji. Zera transmitancji to pierwiastki wielomianu $L(s)$ znajdującego się w liczniku transmitancji.



Rysunek 1: Charakterystyki amplitudowo–częstotliwościowa i fazowo–częstotliwościowa rozpatrywanego filtra analogowego (po lewej) i położenie biegunów (x) i zer (o) jego transmitancji.

W Matlabie pierwiastki wielomianu oblicza się poleceniem `roots`. Położenie biegunów i zer można wyrysować poleceniem `pzmap`, której argumentami może być obiekt opisany bądź macierzami stanu $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ albo licznikiem i mianownikiem transmitancji $\boldsymbol{L}, \boldsymbol{M}$. Charakterystykę systemu ciągłego można narysować poeleceniem `freqs`, której oprócz systemu mozna podać wektor pulsacji, który wygodnie jest generować poleceniem `logspace`.

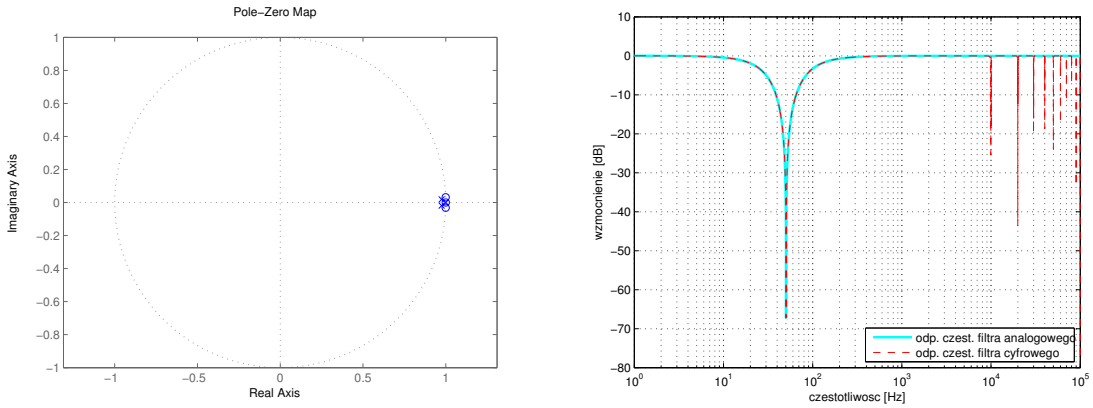
Wyznacz, w postaci liczb i w postaci wykresu, położenia biegunów i zer transmitancji (1). Następnie, za pomocą polecenia `c2dm`, dokonaj konwersji tego filtra analogowego do postaci dyskretnej (czyli przekonwertuj go filtr cyfrowy) opisanej transmitancją zmiennej zespolonej z. Przyjmij częstotliwość próbkowania równą $f_S = 10$ kHz (przy okazji przetestuj różne sposoby dyskretyzacji obiektów ciągłych udostępniane przez polecenie `c2dm`). W wyniku dostaniesz wektory L_d i M_d współczynników wielomianów $L_d(z)$ i $M_d(z)$ licznika i mianownika transmitancji dyskretnej. Przekształć licznik i mianownik do postaci transmitancji specyficznej dla Control System Toolbox Matlaba za pomocą polecenia `tf` podając okres próbkowania $T_S = 1/f_S$. Następnie narysuj położenie zer i biegunów tej dyskretnej transmitancji na płaszczyźnie zmiennej zespolonej z za pomocą polecenia `pzmap`

Wyznacz jak wcześniej pierwiastki licznika i mianownika (czyli zera i bieguny filtra). Oblicz moduły biegunów (czyli pierwiastków mianownika) i sprawdź czy filtr cyfrowy otrzymany przez dyskretyzację filtra analogowego jest nadal stabilny (nie musi tak być!). Jaki jest warunek stabilności filtra cyfrowego?

W celu porównania obu filtrów narysuj ich charakterystyki amplitudowo–częstotliwościowe (czyli moduły zespolonych odpowiedzi częstotliwościowych filtrów) w jednym półlograrytmicznym układzie współrzędnych (polecenie `semilogx`) tak jak pokazano na rysunku 2. Do obliczenia wartości odpowiedzi częstotliwościowych użyj poleceń `freqs` dla filtra analogowego i `freqz` dla filtra cyfrowego, zadając w obu przypadkach taki sam przedział częstotliwości. Do przeliczenia wartości wzmocnienia na decybele użyj polecenia `db`.

Zaobserwuj wpływ zmniejszenia precyzji współczynników filtra cyfrowego (wektory L_d i M_d) na jego odpowiedź częstotliwościową oraz na stabilność. Poprzez zaokrąglenie z pomocą funkcji `round` sprawdź ile cyfr po przecinku muszą mieć współczynniki filtra cyfrowego aby filtr ten był jeszcze stabilny. Poleceniem `step` sprawdź kształty odpowiedzi skokowych tego filtra dla przypadków stabilnych i niestabilnych.

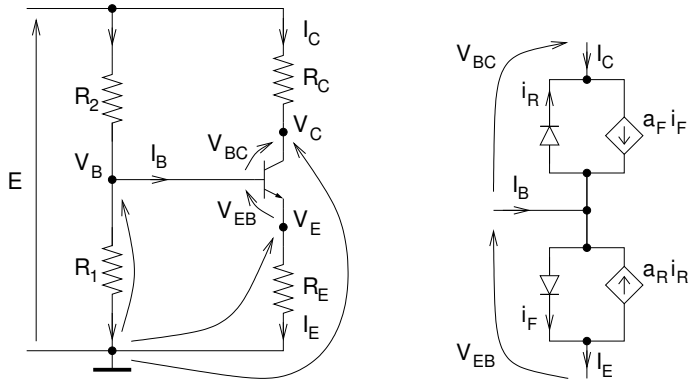
Na koniec zobacz źródło funkcji `roots` poleceniem `type` i sprawdź jaką metodą wyznaczane są pierwiastki wielomianu.



Rysunek 2: Położenie biegunów (x) i zer (o) transmitancji filtra cyfrowego (po lewej); porównanie odpowiedzi częstotliwościowych obu filtrów (po prawej).

2 Obliczanie prądu źródła prądowego z tranzystorem NPN – 3 pkt

Na ryunku 3 pokazano najprostsze źródło prądowe wykorzystujace jeden tranzystor NPN. Prąd kolektora I_C (źródła) jest stały w dość dużym zakresie zmian rezystancji R_C. Ponieważ jednak tranzystor zawiera dwa złącza półprzewodnikowe, których opis jest nieliniowy, rozwiązanie takiego układu można wykonać rozwiązując układ równań nieliniowych np. metodą Newtona–Raphsona.



Rysunek 3: Schemat najprostszego źródła pradowego i model Ebersa-Molla tranzystora NPN.

Do opisu tranzystora pracującego w obszarze aktywnym wykorzystano model Ebersa–Molla pokazany na rysunku 3 po prawej. Składa się z dwóch diod i dwóch sterowanych źródeł prądu. Jedna z diod w czasie

pracy jest spolaryzowana w kierunku przewodzenia (prąd i_F), a druga w kierunku zaporowym (prąd i_R). Zależności łączące prądy kolektora I_C i emitera I_E w modelu tranzystora z napięciem baza–kolektor V_{BC} i napięciem emiter–baza V_{EB} podano poniżej

$$i_F = I_{ES}(e^{\frac{V_{EB}}{V_T}} - 1) \qquad i_R = I_{CS}(e^{\frac{-V_{BC}}{V_T}} - 1) \tag{2}$$

$$I_E = i_F - a_R i_R \qquad I_C = a_F i_F - i_R \tag{3}$$

Znając model zależności (2–3) i wartości parametrów za pomocą metody węzłowej można opisać układ trzema równaniami opisującymi prądy w węzłach za pomocą trzech niewiadomych napięć wezłowych V_B, V_C, V_E jak poniżej

$$\begin{cases} (E - V_B)/R_2 + I_C &= V_B/R_1 + I_E \\ (E - V_C)/R_C &= I_C \\ V_E/R_E &= I_E \end{cases} \tag{4}$$

Po podstawieniu zależności (2–3) do (4) układ równań nieliniowych zapisuje sie jako

$$\begin{cases} f_1(V_C, V_B, V_E) &= 0 \\ f_2(V_C, V_B, V_E) &= 0 \\ f_3(V_C, V_B, V_E) &= 0 \end{cases} \tag{5}$$

lub w postaci macierzowej

$$\boldsymbol{F}(\boldsymbol{V}) = \boldsymbol{0} \tag{6}$$

gdzie $\boldsymbol{V} = [\ V_C \quad V_B \quad V_E\ ]^T$ jest wektorem niewiadomych napięć węzłowych, natomiast $\boldsymbol{F} = [\ f_1(\boldsymbol{V}) \quad f_2(\boldsymbol{V}) \quad f_3(\boldsymbol{V})\ ]^T$ jest wektorem funkcji.

Metoda Newtona–Raphosona dla układów n równań nieliniowych n zmiennych jest uogólnieniem tej samej metody szukania zer (przejść przez zero) jednej funkcji jednej zmiennej[1]. Jest to metoda iteracyjna, w której każdym kroku obliczne jest nowe przybliżenie rozwiązania $\boldsymbol{V}_R$ układu równań (6) dane zależnością

$$\boldsymbol{V_{k+1}} = \boldsymbol{V_K} - \boldsymbol{J}^{-1}(\boldsymbol{V_K})\boldsymbol{F}(\boldsymbol{V_K}) \tag{7}$$

gdzie $\boldsymbol{J}$ jest macierzą Jacobiego obliczoną dla $\boldsymbol{V}_k$, czyli macierzą pochodnych cząstkowych funkcji $\boldsymbol{F}(\boldsymbol{V})$ po wszystkich zmiennych

$$\boldsymbol{J}(\boldsymbol{V}) = \begin{bmatrix} \frac{\partial f_1}{\partial V_C} & \frac{\partial f_1}{\partial V_B} & \frac{\partial f_1}{\partial V_E} \\ \frac{\partial f_2}{\partial V_C} & \frac{\partial f_2}{\partial V_B} & \frac{\partial f_2}{\partial V_E} \\ \frac{\partial f_3}{\partial V_C} & \frac{\partial f_3}{\partial V_B} & \frac{\partial f_3}{\partial V_E} \end{bmatrix} \tag{8}$$

Iteracyjne metody rozwiązywania układów równań nieliniowych są bardzo wrażliwe na wybór punktu startowego. Uproszczona analiza obwodu z rysunku 3 pozwala na wybór punktu startowego $V_{C0} = 9,5[\mathrm{V}]$, $V_{B0} = 8[\mathrm{V}]$, $V_{E0} = 7,5[\mathrm{V}]$ dla następujących danych: $I_{ES} = 10^{-12}[\mathrm{A}]$, $I_{CS} = 2\,10^{-12}[\mathrm{A}]$, $V_T = 0,026[\mathrm{V}]$, $a_F = 0,99$, $a_R = 0,5$, $R_1 = 4[\mathrm{k\Omega}]$, $R_2 = 1[\mathrm{k\Omega}]$, $R_E = 5[\mathrm{k\Omega}]$, $R_C = 100[\Omega]$, $E = 10[\mathrm{V}]$.

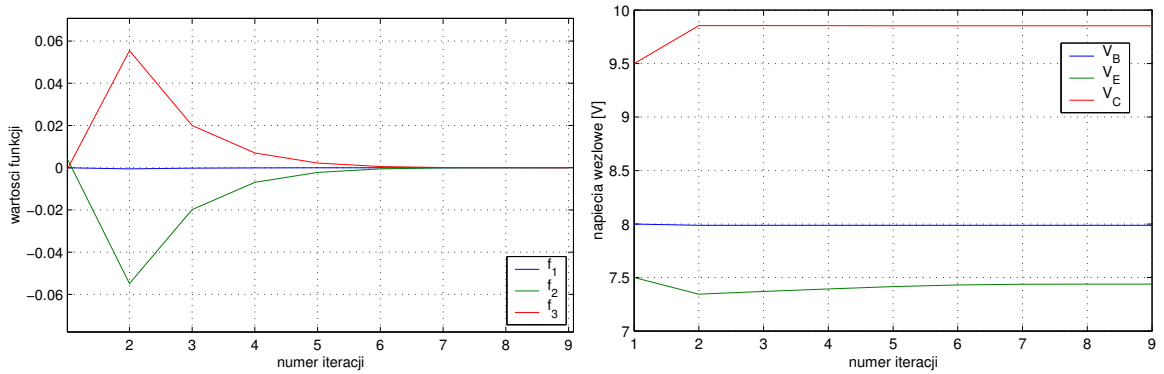
Wyznacz w Matlabie metodą Newtona–Raphsona wartości napięć węzłowych, napięć V_{EB}, V_{BC}, prądów I_C, I_E, I_B. Sprawdź dla jakiego zakresu rezystancji R_C nie zmienia się prąd projektowanego źródła I_C. Znajdź inne punkte startowe, dla których metoda iteracyjna nie będzie zbieżna rysując, wartości zmiennych (napięć węzłowych) i wartości funkcji f_1, f_2, f_3 w kolejnych iteracjach. Sprawdź ile iteracji potrzeba dla osiągnięcia zadanej dokładności.

Do wykonania zadania najwygodniej zdefiniować funkcje $\boldsymbol{F}$ i macierz $\boldsymbol{J}$ za pomocą komend Symbolic Math Toolbox, a w kolejnych iteracjach jedynie obliczać ich wartości dla podstawionych wartosci zmiennych t.j. napięć węzłowych. Najpierw zadeklaruj zmienne symboliczne za pomocą polecenia `syms` (np. `syms VC VB VE`). Następnie zdefiniuj analityczne postaci funkcji `f1 f2 f3` zadeklarowanych zmiennych (np. `f1 = VE - VB`). Ostatecznie zdefiniuj macierz Jacobiego zawierające pochodne cząstkowe wyznaczane analitycznie poleceniem `diff` (np. `diff(f1,VE)`).

[1] Dla jednej zmiennej reguła wyznaczania kolejnego przybliżenia rozwiązania jest następująca $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

Ze względu na zmniejszenie ryzyka pomyłki stosuj kolejne podstawienia, jak w zależnościach (2–3), zamiast długich wzorów opisujących funkcje. Następnie w pętli wykonuj kolejne iteracje, aż norma wektora $\boldsymbol{F}(\boldsymbol{V})$ będzie mniejsza niż 10^{-9}. Maksymalną liczbę iteracji ustal na 40. W każdej iteracji wartości liczbowe $\boldsymbol{F}(\boldsymbol{V})$ i $\boldsymbol{J}(\boldsymbol{V})$ wyznaczaj komendą `eval` i podstawiaj do pomocniczych zmiennych, które wykorzystasz do obliczenia wartości liczbowych wektora napięć węzłowych według (7). Obliczone wartości liczbowe podstawiaj następnie pod zmienne `VB`, `VC`, `VE`, aby w kolejnej iteracji można było ponownie zaktualizować wartości liczbowe $\boldsymbol{F}(\boldsymbol{V})$ i $\boldsymbol{J}(\boldsymbol{V})$.

Początkowe wartości liczbowe (punkt startowy) wektora $\boldsymbol{V}$ nadaj zmiennym tuż przed wejściem do pętli iteracji. Jeśli wszystko wykonasz poprawnie to wartości zmiennych w kolejnych iteracjach powinny zmieniać się jak na rysunku 4 (możesz otrzymać przeciwne znaki pojedycznych funkcji, ale nie wpływa to na rozwiązanie zadania). W pliku `lab08z02.m` znajduje się szkielet programu oraz wartosci elementów i parametrów modelu tranzystora.



Rysunek 4: Wartości funkcji w kolejnych iteracjach (po lewej) i wartości zmiennych (napięć węzłowych) w kolejnych iteracjach (po prawej).

3 Analiza dochodów na oprocentowanym rachunku bankowym – max 3 pkt

Typowym przykładem nieliniowej zależności jest zależność kwoty C zgromadzonej na rachunku o oprocentowaniu rocznym P przy regularnych (comiesięcznych) wpłatach W od ilości wpłat N

$$C = W + W\left(1+\frac{P}{12}\right) + W\left(1+\frac{P}{12}\right)^{N} + \ldots + W\left(1+\frac{P}{12}\right)^{N-1} \tag{9}$$

Częstym problemem jest określenie ilości wpłat N, potrzebnych do zgromadzenia określonej ilość pieniędzy C. Wykorzystując wzór na sumę szeregu

$$1 + r + r^2 + r^3 + \ldots + r^{N-1} = \frac{1-r^N}{1-r} \tag{10}$$

i podstawiając $r = 1 + P/12$ otrzymujemy

$$C(N) = W\frac{1-\left(1+\frac{P}{12}\right)^{N}}{1-\left(1+\frac{P}{12}\right)} = 12\frac{W}{P}\left(\left(1+\frac{P}{12}\right)^{N} - 1\right) \tag{11}$$

Problem znalezienia wartości N dla której $C(N) = C_Z$, można sprowadzić do rozwiązania nieliniowego równania $C(N) - C_Z = 0$ ze względu na N. Przyjmując wysokość comiesięcznego wkładu $W = 1000$, i roczne oprocentowanie $P = 11\% = 0.11$ oblicz ilu wpłat N należy dokonać, aby zgromdzić środki w wysokości $C_0 = 60000$.

Do wykonania obliczeń wykorzystaj Matlaba i funkcje biblioteki GSL. Najpierw w Matlabie rozwiąż zadanie, aby poznać wartość poszukiwanego pierwiastka równania. Zdefiniuj badaną funkcję funkcję $C(N) - C_Z$ za pomocą komendy `inline` podając listę wszystkich czterech parametrów funkcji (pierwszym z parametórw jest zmienna N). Następnie rozwiąż równanie poleceniem `fzero` z punktu starowego $N_0 = 0$, podając

wartości parametrów W, P, C_Z. Zadaj dokładność rozwiązania równą 10^{-4} i włącz wypisywanie informacji o kolejnych iteracjach za pomocą funkcji `optimset` (trzeba ustawić opcje `disp` oraz `tolX`). Uzyskany wynik wykorzystaj jako punkt odniesienia dla dunkcji GSL-a. Jak długo trzeba oszczędzać, aby odłożyć zadaną kwotę?

Rozwiąż to samo zadanie metodami bisekcji, fegula falsi oraz Brenta wykorzystując funkcje biblioteki GSL. Porównaj zbieżność tych metod na podstawie ilości iteracji, dla zadancyh wartości błędu względnego i bezwzględnego 10^{-4}. Szkielet programu znajduje się w pliku `lab08z03.c`. Zawiera on definicje badanej funkcji oraz jej pochodnych, które nie są wykorzystywane przez wymienione metody). Uzupełnij brakujące elmenty programu zgodnie z poniższym opisem niezbędnych funkcji.

Działanie algorytmów GSL-a do poszukiwania pierwiastków składa się z 3 etapów:

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

Solvery ograniczające przedział poszukiwań (ang. *root bracketing solvers*) przechowują stan w strukturze `gsl_root_fsolver`, a solvery wykorzystujące również pochodne funkcji przechowują stan w strukturze `gsl_root_fdfsolver`. Użytkownik musi wcześniej zdefiniować badaną funkcję i ewentualnie jej pochodne.

`gsl_root_fsolver * gsl_root_fsolver_alloc (const gsl_root_fsolver_type * T)` zwraca wskaźnik do solvera typu (metody) `T`. Przykładowe użycie to przygotowanie struktury `s` solvera wykorzystującego metodę bisekcji:
```
const gsl_root_fsolver_type * T = gsl_root_fsolver_bisection;
gsl_root_fsolver * s = gsl_root_fsolver_alloc (T);
```
Sprawdź w dokumentacji GSL-a jak nazywają się inne typy (metody) rozwiązywania równań, które masz wykorzystać w tym zadaniu.

`int gsl_root_fsolver_set (gsl_root_fsolver * s, gsl_function * f, double x_lower, double x_upper)` funkcja inicjalizuje solver `s` do użycia zdefiniowanej przez użytkownika `f` dla początkowego przedziału poszukiwań od `x_lower` do `x_upper`.

`void gsl_root_fsolver_free (gsl_root_fsolver * s)` zwalnia pamięć używaną przez solver `s`.

`int gsl_root_fsolver_iterate (gsl_root_fsolver * s)` funkcja wykonuje pojedynczą iterację solvera `s`. W przypadku nieoczekiwanego błędu (np. natrafienia na osobliwość funkcji) zwraca kody błędów opisane w dokumentacji. Przykład wykorzystania `status = gsl_root_fsolver_iterate (s);`. Solver `s` przechowuje informację o aktualnej estymacie rozwiązania oraz o nowych granicach przedziału w którym znajduje się minimum. Informacje te można uzyskać za pomocą poniższych funkcji:

`double gsl_root_fsolver_root (const gsl_root_fsolver * s)` zwraca estymatę rozwiązania w obecnej iteracji.

`double gsl_root_fsolver_x_lower (const gsl_root_fsolver * s)` zwraca nową wartość dolnego końca przedziału w którym poszukiwane jest rozwiązanie.

`double gsl_root_fsolver_x_upper (const gsl_root_fsolver * s)` zwraca nową wartość górnego końca przedziału w którym poszukiwane jest rozwiązanie.

`int gsl_root_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_root_test_interval (N_lo, N_hi, 1e-4, 1e-4)`.

Na koniec rozwiąż problem za pomocą metod wykorzystujących pochodne analizowanej funkcji np. metodą Newtona, Siecznych lub Steffensona. Pochodne są już zdefiniowane w szkielecie programu. Opis niezbędnych działań oraz przykładowe programy są dostępne pod adresem http://www.gnu.org/software/gsl/manual/gsl-ref_31.html#SEC418.

Na następne zajęcia

Optymalizacja funkcji, szukanie minimów funkcji jednej i wielu zmiennych. Metody przeszukiwania siatki, złotego podziału i bisekcji do szukania minimum w kierunku. Metody gradientowe i bezgradientowe, newtonowskie, wykorzystujące hesjan. Kryteria zakończenia obliczeń. Ograniczenia obszaru poszukiwań.