KSN — III FK — teoria 9
Rozwiązywanie równań nieliniowych

Zajmujemy się znajdowaniem pierwiastków równania

$$f(x) = 0. \tag{1}$$

Większość metod rozwiązywania równania (1) jest iteracyjna a pierwszym krokiem jest zgrubne (najczęściej graficzne) określenie przedziału, w którym znajduje(ą) się miejsca zerowe funkcji $f(x)$.

Ideę poszukiwania pierwiastków równania (1) pozwolę sobie przedstawić (jak zwykle) na najprostszym przykładzie. Wyobraźmy sobie, że możemy stwierdzić, iż poszukiwany przez nas pierwiastek równania (1) α znajduje się wewnątrz przedziału (x_1, x_2) takiego, że $f(x_1) \cdot f(x_2) < 0$ (między nami mówiąc funkcja ciągła **musi** mieć przynajmniej jeden rzeczywisty pierwiastek w takim przedziale). Oznacza to, że funkcja f ma **różne** znaki na końcach przedziału (x_1, x_2). Sprawdzając znak funkcji w połowie przedziału $x_3 = (x_1 + x_2)/2$ jesteśmy w stanie dwukrotnie zmniejszyć długość przedziału podejrzanego o ukrywanie pierwiastka α — będzie on między x_1 i x_3 bądź x_3 i x_2 w zależności od tego, dla którego z tych przedziałów iloczyn $f(x_i) \cdot f(x_j)$ znów okaże się ujemny. Jeśli będziemy powtarzać sukcesywnie operację połowienia przedziału (*bisekcji* — stąd nazwa metody) jesteśmy w stanie osaczyć pierwiastek α do przedziału o dowolnie małej długości ε — a więc oszacować go z taką właśnie niepewnością.

Na podobnej „zasadzie działania" oparte są metody *siecznych* i *regula falsi*.

Natomiast metoda Newtona–Raphsona generuje kolejne przybliżenia x_{i+1} pierwiastka α poprzez punkty przecięcia *stycznej* do $y = f(x)$ w punkcie x_i z osią OX:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

co wymaga znajomości nie tylko funkcji $f(x)$ ale i jej pochodnej $f'(x)$.

W bibliotece *Numerical Recipes* funkcja

```
FUNCTION rtbis(func,x1,x2,xacc)
INTEGER JMAX
REAL rtbis,x1,x2,xacc,func
...
END
```

poszukuje w podejrzanym o ukrywanie pierwiastka przedziale (`x1`,`x2`) zer funkcji `func` metodą bisekcji z dokładnością do `xacc`.

Z takim samym zestawem parametrów wywoływana jest procedura

```
FUNCTION rtflsp(func,x1,x2,xacc)
INTEGER MAXIT
REAL rtflsp,x1,x2,xacc,func
...
END
```

Natomiast metoda Newtona–Raphsona

```
FUNCTION rtnewt(funcd,x1,x2,xacc)
INTEGER JMAX
REAL rtnewt,x1,x2,xacc


...
END
```

potrzebuje oprócz wartości funkcji w bieżącym przybliżeniu rozwiązania również informacji o wartości tam jej pochodnej, Obie je zwraca procedura `funcd`.

Procedura `zrhqr`

```
SUBROUTINE zrhqr(a,m,rtr,rti)
INTEGER m,MAXM
REAL a(m+1),rtr(m),rti(m)
...
END
```

znajduje zera wielomianu stopnia `m`, którego współczynniki zawiera tablica `a`. W wyniku działania procedury część rzeczywista i urojona kolejnych piewiastków trafia do `rtr` i `rti`.

Biblioteka *GSL* umożliwia znajdowanie zer zdefiniowanych przez nas funkcji (obiektów typu `gsl_function`) m.in. metodami bisekcji (`gsl_root_fsolver_bisection`), *falsi* (`gsl_root_fsolver_falsepos`) i Newtona (`gsl_root_fdfsolver_newton`).

Metoda postępowania składa się z 3 głównych faz:

- zainicjalizowania obiektu typu `gsl_root_fsolver` dla wybranego algorytmu,
- przeprowadzenia pojedynczej iteracji (np. za pomocą `gsl_root_fsolver_iterate`),
- sprawdzenia osiągniętej dokładności (np. funkcją `gsl_root_test_interval`) i ew. powtarzania iteracji aż do uzyskania zamierzonej dokładności.

Krzysztof Malarz & Maciej Wołoszyn, Kraków, 4 stycznia 2006