Rozwiązanie zadania N12

Krzysztof Waniak

Znaleźć wszystkie rozwiązania układu równań:

$$\begin{cases} \sin\left(x + y^2 + 1\right) = 0, \\ xy - 1 = 0 \end{cases}$$

w kwadracie $(-2,2)\times(-2,2)$. Można korzystać z gotowych bibliotek numerycznych, ale nie z gotowych programów typu Mathematica.

Wiadomo nam, że układ ma jedno rozwiązanie (źródło: WolframAlpha) w wyznaczonym przedziale. Tak, więc trzeba je iteracyjnie znaleźć w zadanym kwadracie. Do rozwiązania powyższego układu równań wykorzystano funkcje biblioteczne GSL (szukanie miejsc zerowych wielu zmiennych).

Kod programu:

```
#include<stdlib.h>
#include<stdio.h>
#include<gsl/gsl_vector.h>
#include<gsl/gsl_multiroots.h>
#include<ctype.h>          /* zawiera F_OK itp.    */
#include<unistd.h>        /* zawiera funkcje access(), usleep()  */
#define wyp(a) printf(#a "\n")
#define wyp2(a) printf(#a)
#define wypisz(a) printf("%.3f",a)
#define wypisz1(a) printf("%3u",a)
#define wypisz2(a) printf("%.13f",a)
#define karetka printf("\n")
#define karetka2 printf("\n\n")
#define space printf(" ")
#define space2 printf("  ")

void Wynik(gsl_multiroot_fsolver * a)
{
    wyp2(Rozwiazanie: x =);
    space;
    wypisz(gsl_vector_get(a -> x, 0));
    wyp2(: y =);
    space;
    wypisz(gsl_vector_get(a -> x, 1));
    karetka2;
}

void wypisywanie(size_t krok, gsl_multiroot_fsolver * a)
{
    wyp2(Krok =);
    wypisz1(krok+1);
    space2;
    wyp2(x =);
    space;
    wypisz(gsl_vector_get(a -> x, 0));
    space2;
    wyp2(y =);
```

```
    space;
    wypisz(gsl_vector_get(a -> x, 1));
    karetka;

    wyp2(f1(x) =);
    space;
    wypisz2(gsl_vector_get(a -> f, 0));
    space;
    space2;
    wyp2(f2(x) =);
    space;
    wypisz2(gsl_vector_get(a -> f, 1));
    karetka;
}
/* dokladnosc do ok. 10^-8 - 10^-10*/
int zero(const gsl_vector * x, void *params, gsl_vector * f)
{
      const double x0 = gsl_vector_get(x, 0);
      const double x1 = gsl_vector_get(x, 1);
      const double y0 = sin(x0 + (x1 * x1) + 1);
      const double y1 = x0 * x1 - 1;

      gsl_vector_set(f, 0, y0);
      gsl_vector_set(f, 1, y1);

      return GSL_SUCCESS;
}

int main(void)
{
      const gsl_multiroot_fsolver_type *T;
      gsl_multiroot_fsolver *a;

      int status;
      size_t krok = 0;

      const size_t n = 2;
      gsl_multiroot_function f = { &zero, n };

      double x_init[2] = { -2.0, -2.0 };
      gsl_vector *x = gsl_vector_alloc(n);

      gsl_vector_set(x, 0, x_init[0]);
      gsl_vector_set(x, 1, x_init[1]);

      T = gsl_multiroot_fsolver_hybrids;
      a = gsl_multiroot_fsolver_alloc(T, 2);
      gsl_multiroot_fsolver_set(a, &f, x);

      wypisywanie(krok, a);

      do
    {
            krok++;
            status = gsl_multiroot_fsolver_iterate(a);
            wypisywanie(krok, a);

            if (status)
            {
                  break;
        }
```

```
            status = gsl_multiroot_test_residual(a -> f, 1e-8);
      } while (status == GSL_CONTINUE && krok < 1000);

      /* printf("%s\n", gsl_strerror(status)); */
      Wynik(a);

      gsl_multiroot_fsolver_free(a);
      gsl_vector_free(x);

      /*system("pause");*/
      return 0;
}
```

Wynik działania programu: