Rozwiązanie zadania N9

Krzysztof Waniak

Znaleźć wszystkie rozwiązania równania $\det(A-\lambda \mathrm{I})$ wszystkimi metodami z poprzedniego zadania z dokładnością 10^{-8}. A jest macierzą z zadania N8. Które metody działają najszybciej?

Macierz:

$$A=\begin{bmatrix}1 & 2 & 3\\ 2 & 4 & 5\\ 3 & 5 & -1\end{bmatrix}$$

Po przekształceniu:

$$\det(A-\lambda \mathrm{I})=\begin{vmatrix}1-\lambda & 2 & 3\\ 2 & 4-\lambda & 5\\ 3 & 5 & -1-\lambda\end{vmatrix}$$

Po uproszczenie jego jawna postać to:
$\det(A-\lambda \mathrm{I})= f(x)=x(39-(x-4)x)-1$

Kod programu:

```
#include<stdio.h>
#include<math.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("%2.16f",a)
#define wypisz2(a) printf("%i",a)
#define karetka printf("\n")
#define karetka2 printf("\n\n")
#define space printf(" ")

/* Sprawdzanie, czy wejściowy plik nie istnieje, jesli nie istnieje zwraca
wartosc "TRUE" */
int nieistnieje(const char* nazwa)
{
    return access(nazwa, F_OK);
}

int counter = 0;

/* Funcja det(A-lambda*I)*/
/*
|1 2 3|
|2 4 5|
|3 5 -1|
```

```
*/
double func(double x)
{
      return x * (39 - (x - 4) * x) - 1;
}

/* Pochodna */
double funcp(double x)
{
      return (-3) * x * x + 8 * x + 39;
}

/* Druga pochodna */
double funcpp(double x)
{
      return (-6) * x + 8;
}

double bisekcja(double lewy, double prawy, double eps)
{
      double pierwiastek;

      if (func(lewy) * func(prawy) < 0)
    {
            pierwiastek = lewy;

            while ((prawy - lewy) >= eps)
        {
                  pierwiastek = (prawy + lewy) / 2.0;
                  if (func(lewy) * func(pierwiastek) < 0)
                  {
                        prawy = pierwiastek;
            }
                  else if (func(pierwiastek) * func(prawy) < 0)
                  {
                        lewy = pierwiastek;
            }
                  else
                  {
                        break;
            }
                  counter++;
            }
      }
    else
    {
            pierwiastek = -111111;
    }
      return (pierwiastek);
}

double halley(double xold, double eps)
{
      double xnew = xold;

      do
    {
            xold = xnew;
            xnew = xold - (2 * func(xold) * funcp(xold))/ (2 * (funcp(xold)
* funcp(xold)) - func(xold)   * funcpp(xold));
```

```
            counter++;
      } while (fabs(xnew - xold) > eps);

      return xnew;
}

double newton(double xold, double eps)
{
      double xnew = xold;

      do {
            xold = xnew;
            xnew = xold - (func(xold) / funcp(xold));

            counter++;
      } while (fabs(xnew - xold) > eps);

      return xnew;
}

double FalsiMethod(double s, double t, double eps)
{
      int side = 0;
      double r, fr, fs = func(s), ft = func(t);

      do {
            r = (fs * t - ft * s) / (fs - ft);
            fr = func(r);

            if (fr * ft > 0) {
                  t = r;
                  ft = fr;
                  if (side == -1)
                        fs /= 2;
                  side = -1;
            } else if (fs * fr > 0) {
                  s = r;
                  fs = fr;
                  if (side == +1)
                        ft /= 2;
                  side = +1;
            } else
                  break;

            counter++;
      } while ((fabs(t - s)) > (eps * fabs(t + s)));
      return r;
}

double siecznych(double xn_1, double xn, double eps)
{
      double d;
      do {
            d = (xn - xn_1) / (func(xn) - func(xn_1)) * func(xn);
            xn_1 = xn;
            xn = xn - d;

            counter++;
      } while (fabs(d) > eps);
      return xn;
}
```

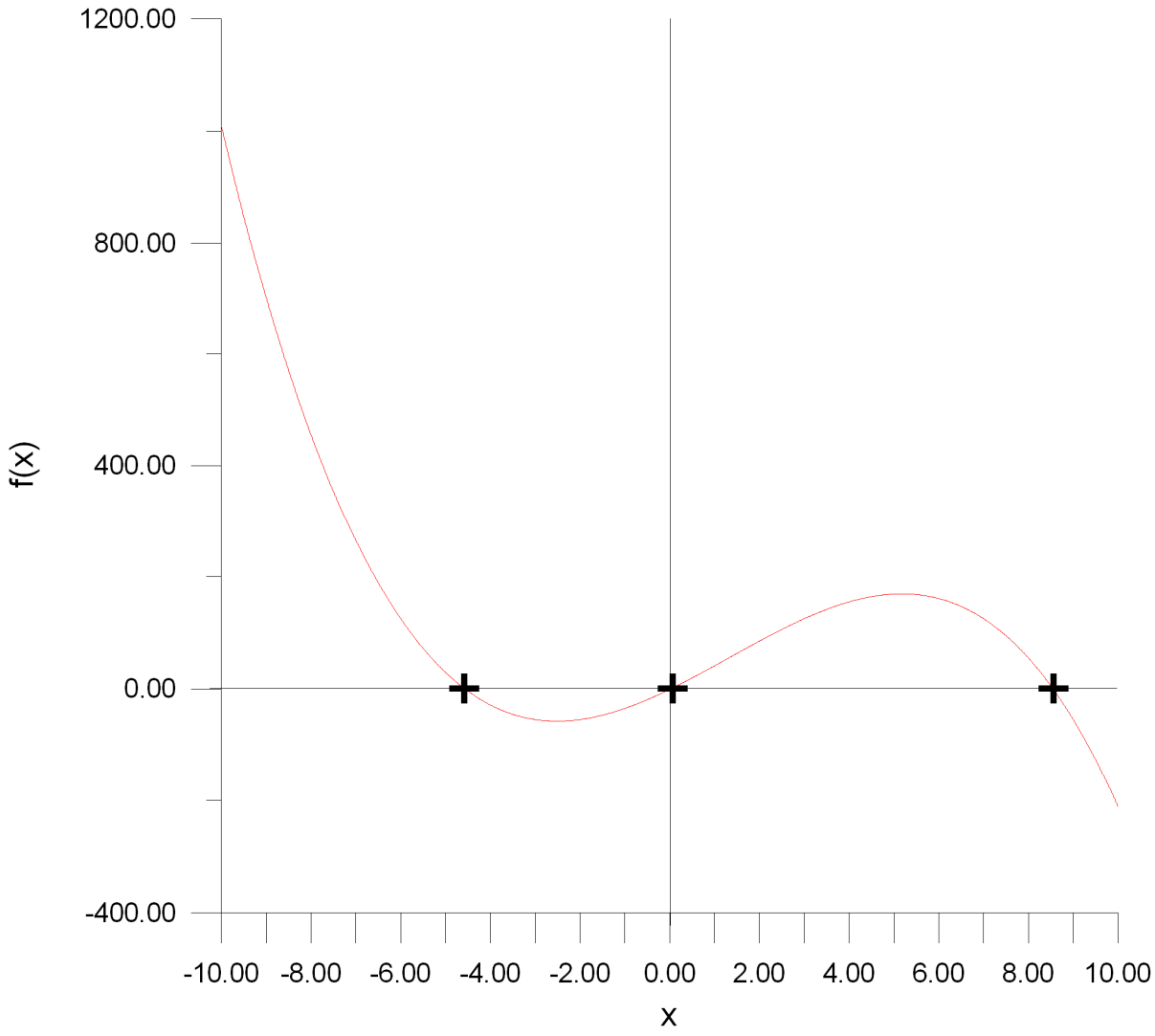
```
void szukajWPrzedzialach(double lewy, double prawy, double eps)
{
      counter = 0;
      wyp2(Metoda bisekcji: x = );
      wypisz(bisekcja(lewy, prawy, eps));
      karetka;
      wyp2(x znaleziono po);
      space;
      wypisz2(counter);
      space;
      wyp(iteracjach);

      counter = 0;
      wyp2(Metoda Newtona: x = );
      wypisz(newton(prawy, eps));
      karetka;
      wyp2(x znaleziono po);
      space;
      wypisz2(counter);
      space;
      wyp(iteracjach);

      counter = 0;
      wyp2(Metoda Halleya: x = );
      wypisz(halley(prawy, eps));
      karetka;
      wyp2(x znaleziono po);
      space;
      wypisz2(counter);
      space;
      wyp(iteracjach);

      counter = 0;
      wyp2(Metoda Regula falsi: x = );
      wypisz(FalsiMethod(lewy, prawy, eps));
      karetka;
      wyp2(x znaleziono po);
      space;
      wypisz2(counter);
      space;
      wyp(iteracjach);

      counter = 0;
      wyp2(Metoda siecznych: x = );
      wypisz(siecznych(lewy, prawy, eps));
      karetka;
      wyp2(x znaleziono po);
      space;
      wypisz2(counter);
      space;
      wyp(iteracjach);
}


int main(void)
{
    FILE *fwynik;
    char plik_b[30];
    int p1;
    double i;
```

```
    wyp(Podaj nazwe pliku wyjsciowego:);
    scanf("%s",  &plik_b[0]);
    while(!nieistnieje(plik_b))
    {
        karetka;
        wyp(Taka nazwa pliku juz istnieje);
        wyp(Wprowadz inna nazwe pliku);
        karetka;
        scanf("%s",  &plik_b[0]);
    }
    karetka;

    fwynik = fopen(plik_b, "w");
    for(i = -20.0; i <= 20.0; i+=0.01)
    {
        /*printf("%1.2f %f \n",i,func(i));*/
        fprintf(fwynik, "%1.2f %f \n",i,func(i));
    }

    fclose(fwynik);

    p1=0;
    wyp2(Funkcja obliczona i zapisana do pliku o nazwie:);
    space;
    while(plik_b[p1]!='\0')  printf("%c",plik_b[p1++]);
    karetka2;

    karetka;
    wyp(Miejsca zerowe w przedziale <-7: -3>);
    szukajWPrzedzialach(-7.0, -3.0, 10e-8);
    karetka2;
    wyp(Miejsca zerowe w przedziale <-3: 3>);
    szukajWPrzedzialach(-3.0, 3.0, 10e-8);
    karetka2;
    wyp(Miejsca zerowe w przedziale <7: 10>);
    szukajWPrzedzialach(7.0, 10.0, 10e-8);
    karetka;

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

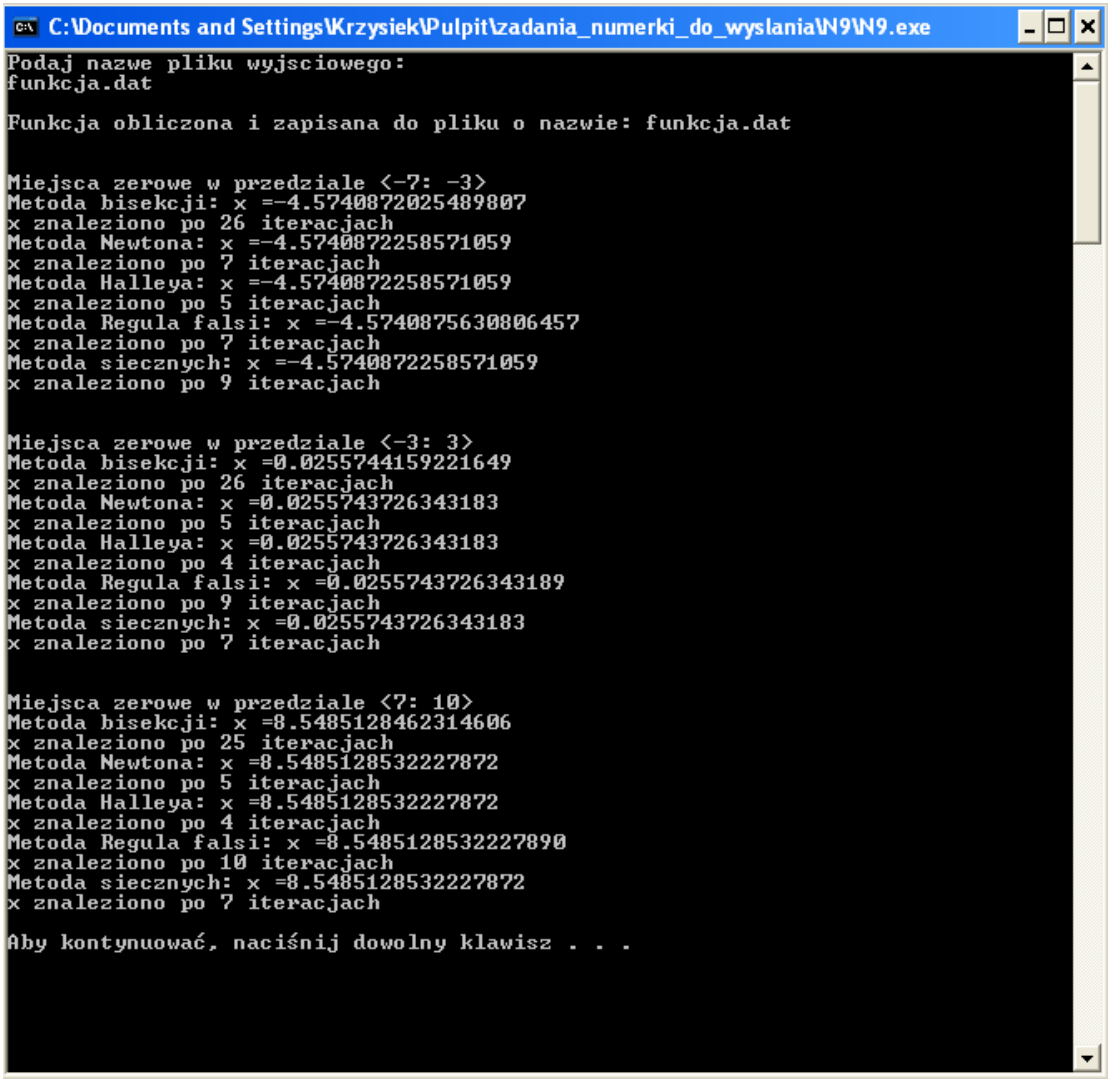
Wartości podanej funkcji zapisane do wskazanego pliku. W archiwum przykładowy wynik programu zapisany do pliku funkcja.dat.

Wybrano trzy przedziały, w których możemy znaleźć miejsca zerowe funkcji $\det(A - \lambda \mathrm{I})$; widoczne na przykładowym wykresie utworzonym w programie Grapher. W przedziałach tych będziemy szukać miejsc zerowych za pomocą metod z N8.

Wykres na podstawie funkcja.dat:



Wynik działania programu:



Wyniki zgodne z przewidywaniami; metoda Newtona oraz Halley’a jest najszybsza.