Университет ИТМО

Вычислительная математика

Лабораторная работа №2

ФИО студента: Готовко Алексей Владимирович
Направление подготовки: 09.03.04 (СППО)
Учебная группа: P32101
ФИО преподавателей:
Малышева Т.А.
Рыбаков С.Д.

Санкт-Петербург
2023г.

1 Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

2 Вычислительная реализация задачи

Функция по варианту:

$$x^3 + 2.84x^2 - 5.606x - 14.766$$

№ шага	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|a-b\|$
1	2.000	3.000	2.500	-6.618	20.976	4.594	1.000
2	2.000	2.500	2.250	-6.618	4.594	-1.611	0.500
3	2.250	2.500	2.375	-1.611	4.594	1.336	0.250
4	2.250	2.375	2.313	-1.611	1.336	-0.176	0.125
5	2.313	2.375	2.344	-0.176	1.336	0.570	0.063
6	2.313	2.344	2.328	-0.176	0.570	0.195	0.031
7	2.313	2.328	2.320	-0.176	0.195	0.009	0.016

№ шага	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\|x_{k+1}-x_k\|$	$f(x)$	$\|a-b\|$
0	-2.000	-0.194	-4.966	-2.039	0.039	4.594	1.000
1	-2.039	-0.005	-4.715	-2.040	0.001	-1.611	0.500
3	2.250	2.500	2.375	-1.611	4.594	1.336	0.250
4	2.250	2.375	2.313	-1.611	1.336	-0.176	0.125
5	2.313	2.375	2.344	-0.176	1.336	0.570	0.063
6	2.313	2.344	2.328	-0.176	0.570	0.195	0.031
7	2.313	2.328	2.320	-0.176	0.195	0.009	0.016

№ шага	x_k	x_{k+1}	$\varphi(x_{k+1})$	$f(x_{k+1})$	$\|x_{k+1}-x_k\|$	$f(x)$	$\|a-b\|$
0	-4.000	-3.444	-3.310	-2.623	0.556	4.594	1.000
1	-3.444	-3.310	-3.241	-1.361	0.134	-1.611	0.500
2	-3.310	-3.241	-3.200	-0.808	0.069	1.336	0.250
3	-3.241	-3.200	-3.174	-0.510	0.041	-0.176	0.125
4	-3.200	-3.174	-3.157	-0.335	0.026	0.570	0.063
5	-3.174	-3.157	-3.145	-0.224	0.017	0.195	0.031
6	-3.157	-3.145	-3.137	-0.152	0.011	0.009	0.016

3 Программная реализация задачи

```
def bisection(f, left, right, eps):
    result = validate_parameters(f, left, right, eps)
    if result["error_msg"] is not None:
        return result

    mid = (left + right) / 2

    val_left = f(left)
    val_right = f(right)
    val_mid = f(mid)

    # True  <=> (f(boundary) > 0)
    # False <=> (f(boundary) < 0)
    sign_left = val_left > 0
    sign_right = val_right > 0
    sign_mid = val_mid > 0
```

```
17
18      iter_cnt = 0
19      while (right - left) > eps:
20          if sign_left == sign_mid:
21              left = mid
22              sign_left = sign_mid
23          elif sign_mid == sign_right:
24              right = mid
25              sign_right = sign_mid
26          else:
27              print("wtf?")
28
29          mid = (left + right) / 2
30          val_mid = f(mid)
31          sign_mid = val_mid > 0
32
33          iter_cnt += 1
34
35      result["root"] = mid
36      result["value"] = val_mid
37      result["iterations"] = iter_cnt
38      return result
39
40
41  def secant(f, left, right, eps):
42      result = validate_parameters(f, left, right, eps)
43      if result["error_msg"] is not None:
44          return result
45
46      try:
47          derivatives_res = validate_derivatives_const_sign_and_get_init_value(f, left, right, eps)
48          if derivatives_res["is_error"]:
49              result["error_msg"] = \
50                  "At least one of the functions derivatives does not "\
51                  "have constant sign on the entire interval"
52              return result
53      except ValueError:
54          result["error_msg"] = \
55              "At least one of the functions derivatives is not defined on the entire interval"
56          return result
57
58      if derivatives_res["start_left"]:
59          x0 = left
60          x1 = left + eps
61      else:
62          x0 = right
63          x1 = right - eps
64
65      iter_cnt = 0
66      x2 = 0
67      while abs(f(x1)) > eps:
68          x2 = x1 - (x1 - x0) / (f(x1) - f(x0)) * f(x1)
69          x0 = x1
70          x1 = x2
71          iter_cnt += 1
72
73      result["root"] = x2
74      result["value"] = f(x2)
75      result["iterations"] = iter_cnt
76      return result
77
```

```python

def fixed_point_iteration(f, left, right, eps):
    result = validate_parameters(f, left, right, eps)
    if result["error_msg"] is not None:
        return result

    f_dx = get_derivative(f)
    f_dx_res = validate_func_max(f_dx, left, right, eps)
    if f_dx_res["is_error"]:
        result["error_msg"] = "Function's derivative is not defined on the entire interval"
        return result

    f_dx_max = f_dx_res["max_value"]
    lmd = -1 / f_dx_max
    phi = get_phi(f)

    x0 = left
    x1 = phi(x0, lmd)
    iter_cnt = 0
    while abs(x1 - x0) > eps:
        x0 = x1
        x1 = phi(x1, lmd)
        iter_cnt += 1

    result["root"] = x1
    result["value"] = f(x1)
    result["iterations"] = iter_cnt
    return result
# ----------------------------------------------------------------------------------------------------

def system_fixed_point_iteration(system, x_val, y_val, eps):
    result = {
        "root_x": None,
        "root_y": None,
        "value": None,
        "iterations": 0,
        "error_msg": None
    }

    iter_cnt = 0
    max_iter = 1000
    max_value = 1e+10

    while abs(system.eq_arr[0].func(x_val, y_val)) > eps:
        x_val = system.eq_arr[0].phi(x_val, y_val)
        y_val = system.eq_arr[1].phi(x_val, y_val)

        iter_cnt += 1

        if iter_cnt > max_iter or x_val > max_value or y_val > max_value:
            result["error_msg"] = "Does not converge"
            return result

    result["root_x"] = x_val
    result["root_y"] = y_val
    result["value"] = system.eq_arr[0].func(x_val, y_val)
    result["iterations"] = iter_cnt

    return result
```

Полный код программы доступен по ссылке.

4 Вывод

Для наиболее оптимального распределения человеческих ресурсов и поддержания количества нервных клеток в головном мозгу, настоятельно рекомендуется использовать готовые библиотеки, содержащие наиболее эффективные реализации алгоритмов, вместо самостоятельной реализации оных.

Иными словами, вообще лучше зачиллиться и не изобретать велосипед.