Лабораторaная работа 1

”Численные методы решения нелинейных уравнений”

Глеб Бузин - Б03-907

2021-09-24

Contents

1 Постановка задачи

Для двух уравнений

$$3x + 4x^3 - 12x^2 - 5 = 0$$
$$2\tan(x) - \frac{x}{2} + 1 = 0$$

1. Локализовать корни уравнения (найти непересекающиеся отрезки, каждый из которых имеет только один корень).
2. Формулы выбранных методов уточнения корней с обоснованием их сходимости.
3. Таблицы расчетных данных.
4. Оценка погрешности найденного решения (сравнение найденного решения с аналитическим решением).

2 Локализация

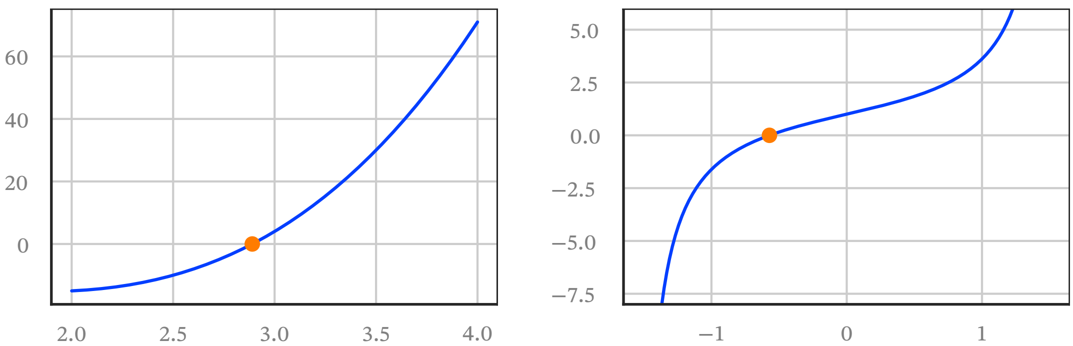
Требуется найти непересекающиеся отрезки $[a_i, b_i]$, каждому из которых принадлежит один и только один корень данного уравнения x_i^*.
Пусть $f(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n = 0$, тогда требуется

1. Определить число этих корней $N = N_+ + N_-$;
   - Т. Декарта: число положительных корней равно числу перемен знаков в последовательности коэффициентов $a_0, a_1, ..., a_n$ или на четное число меньше.
   - Все корни многочлена (включая комплексные) лежат в кольце

   $$\frac{|a_n|}{|a_n| + B} <= |z| <= 1 + \frac{A}{|a_0|}$$

   где $A = max\{|a_1|, |a_2|, ..., |a_n|\}$, $B = max\{|a_0|, |a_1|, ..., |a_{n-1}|\}$
2. Определить отрезки, на которых лежат все действительные корни;
3. Найти N непересекающихся отрезков $[a_i, b_i]$, для которых $f(a_i)f(b_i) < 0$.

2.1 Графики функций и выбор отрезков



(a) $3x + 4x^3 - 12x^2 - 5 = 0$ (b) $2\tan(x) - \frac{x}{2} + 1 = 0$

Figure 1: Графики

Первая функция возрастает во всей области определения. Корень находится в сегменте $[2.5, 3.5]$. Вторая функция периодическая. Один из корней находится в сегменте $[-1, 0]$.

3 Методы уточнения корней

Уточнение корней Для каждого корня $x_i^* \in [a_i, b_i]$ данного уравнения требуется найти $\tilde{x_i}$ такое, что $|\tilde{x_i} - x_i^*| < \varepsilon$, где ε - заданная погрешность.

3.1 Метод половинного деления

Отыскивается $\tilde{x}$ - приближение к корню $x^* \in [a, b]$ с точностью ε.

Шаг 1. $m = 0$
Шаг 2. $a_m = a, b_m = b$
Шаг 3. $c = \frac{a_m + b_m}{2}$
Шаг 4. (а) Если $f(c)f(a_m) > 0$, то $a_{m+1} = c, b_{m+1} = b_m$ (б) Если $f(c)f(b_m) > 0$, то $a_{m+1} = a_m, b_{m+1} = c$
Шаг 5. Если $|b_{m+1} - a_{m+1}| > \varepsilon$, то $m = m + 1$, перейти к Шагу 2, иначе $\tilde{x} = \frac{a_{m+1} + b_{m+1}}{2}$

3.2 Метод простой итерации

$$f(x) = 0 \to x = g(x) \to x_{n+1} = g(x_n)$$

Условие сходимости: $\forall x \in [a, b] : |g'(x)| < 1$ или $\forall x', x'' \in [a, b] : |g(x') - g(x'')| <= q|x' - x''|, q < 1$

3.3 Метод Ньютона

$$f(x) = f(x_n) + f'(x_n)(x - x_n) \to x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Условие сходимости: $\frac{1}{2}\frac{M_2}{m_1}|x_0 - x^*|^2 < 1$

$$|x_{n+1} - x^*| < \frac{1}{2}\frac{M_2}{m_1}|x_n - x^*|^2 < C^{-1}(C|x_0 - x^*|)^{2^n}$$

Выбор начального приближения: $f(x_0)f''(x_0) > 0$. Практический критерий оценки достижения заданной точности:

$$|x_{n+1} - x^*| < \frac{1}{2}\frac{M_2}{m_1}|x_{n+1} - x_n|^2 < \epsilon$$

3.4 Метод Секущих

В методе Ньютона подставим: $f'(x_n) \approx \dfrac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

4 Уточнение корней

4.1 Метод половинного деления

Будем уточнять все корти с точностью $\epsilon = 10^{-4}$.

bisection.go (GitHub)

```go
package numerical

// https://en.wikipedia.org/wiki/Bisection
func Bisection(f func(float64) float64,
	a, b, tolerance float64,
	maxIterations int) (float64, int, error) {
	if f(a)*f(b) >= 0 {
		panic("oh no")
	}

	for n := 1; n <= maxIterations; n++ {
		c := (a + b) / 2
		if f(c) == 0 || (b-a)/2 < tolerance {
			return c, n, nil
		}
		if f(c)*f(a) > 0 {
			a = c
		} else {
			b = c
		}
	}

	panic("maxIterations exceeded")
}
```

Для первой функции

$$f(x) = 3x + 4x^3 - 12x^2 - 5$$

потребовалось 14 итераций до нужной точности на отрезке $[2.5, 3.5]$ с найденным значением $x = 2.89019775390625$.

Для второй функции

$$f(x) = 2\tan(x) - \frac{x}{2} + 1 = 0$$

потребовалось 15 итераций на отрезке $[-1, 1]$ с найденным значением $x = -0.57135009765625$.

4.2 Метод секущих

secant.go

```go
package numerical

func Secant(f func(float64) float64,
        a, b, tolerance float64,
        maxIterations int) (float64, int, error) {
        var a_n, b_n, m_n, f_m_n float64
        if f(a)*f(b) >= 0 {
                panic("oh no")
        }
        a_n = a
        b_n = b
        for n := 1; n <= maxIterations; n++ {
                m_n = a_n - f(a_n)*(b_n-a_n)/(f(b_n)-f(a_n))
                f_m_n = f(m_n)
                if f(a_n)*f_m_n < 0 {
                        b_n = m_n
                } else if f(b_n)*f_m_n < 0 {
                        a_n = m_n
                } else if f_m_n == 0 {
                        return m_n, n, nil
                } else {
                        panic("oh no")
                }
        }
        return a_n - f(a_n)*(b_n-a_n)/(f(b_n)-f(a_n)), maxIterations, nil
}
```

Для достижения нужной точности потребовалось 30 итераций для обеих функций.

4.3 Метод простой итерации

fixed_point_iteration.go

```go
package numerical

import (
	"fmt"
	"math"
)

func FixedPointIteration(fn, fng func(float64) float64,
	x0, tolerance float64,
	maxIterations int) {
	step := 1
	flag := 1
	condition := true
	var x1 float64
	for condition {
		x1 = fng(x0)
		fmt.Printf("Iteration-%d, x1 = %0.6f and fn(x1) = %0.6f\n",
			step, x1, fn(x1))
		x0 = x1

		step = step + 1

		if step > maxIterations {
			flag = 0
			break
		}

		condition = math.Abs(fn(x1)) > tolerance
	}

	if flag == 1 {
		fmt.Println("\nSolution: ", x1)
	} else {
		fmt.Printf("\nNot Convergent.")
	}
}
```

Для метода простой итерации приведем выражение вида $f(x) = 0$ к виду $x = g(x)$.

$$x = \sqrt[3]{3x^2 - 0.75x + 1.25}$$

```
Iteration-1, x1 = 2.626794 and fn(x1) = -7.420175
Iteration-3, x1 = 2.772188 and fn(x1) = -3.686390
Iteration-5, x1 = 2.837943 and fn(x1) = -1.706944
Iteration-7, x1 = 2.867166 and fn(x1) = -0.766414
Iteration-9, x1 = 2.880054 and fn(x1) = -0.339508
Iteration-11, x1 = 2.885718 and fn(x1) = -0.149510
Iteration-13, x1 = 2.888204 and fn(x1) = -0.065670
Iteration-15, x1 = 2.889294 and fn(x1) = -0.028812
Iteration-17, x1 = 2.889772 and fn(x1) = -0.012634
Iteration-19, x1 = 2.889982 and fn(x1) = -0.005539
Iteration-21, x1 = 2.890074 and fn(x1) = -0.002428
Iteration-23, x1 = 2.890114 and fn(x1) = -0.001064
Iteration-25, x1 = 2.890132 and fn(x1) = -0.000467
Iteration-27, x1 = 2.890139 and fn(x1) = -0.000205
Iteration-29, x1 = 2.890143 and fn(x1) = -0.000090
```

Для заданной точности потребовалось 29 итераций.

4.4 Метод Ньютона

newtons_method.go

```go
package numerical

import (
	"fmt"
	"math"
)

// x0 - the initial guess
// fn - the function whose root we are trying to find
// fnPrime - the derivative of the function
// tolerance - tolerance
// epsilon - do not divide by a number smaller than this
// todo make it a library function
func NewtonsMethod(fn, fnPrime func(float64) float64,
	x0, tolerance, epsilon float64, maxIterations int) {
	solutionFound := false
	var y, yPrime, x1 float64

	for i := 1; i <= maxIterations; i++ {
		fmt.Printf("Iteration-%d, x1 = %0.6f and fn(x1) = %0.6f\n",
			i, x1, fn(x1))
		y = fn(x0)
		yPrime = fnPrime(x0)

		// Stop if the denominator is too small
		if math.Abs(yPrime) < epsilon {
			break
		}

		// Do Newton's computation
		x1 = x0 - y/yPrime

		// Stop when the result is within the desired tolerance
		if math.Abs(x1-x0) <= tolerance {
			solutionFound = true
			break
		}

		// Update x0 to start the process again
		x0 = x1
	}

	if solutionFound {
		// x1 is a solution within tolerance and maximum number of iterations
		fmt.Println("\nSolution: ", x1)
	} else {
		// Newton's method did not converge
		fmt.Println("Did not converge")
	}
}
```

Для первого уравнения потребовалось 5 итераций:

```
Newton's method:
Iteration-1, x1 = 0.000000 and fn(x1) = -5.000000
Iteration-2, x1 = 3.055556 and fn(x1) = 6.241427
Iteration-3, x1 = 2.905894 and fn(x1) = 0.539087
Iteration-4, x1 = 2.890309 and fn(x1) = 0.005540
Iteration-5, x1 = 2.890145 and fn(x1) = 0.000001
```

Для второго - 7:

```
Iteration-1, x1 = 0.000000 and fn(x1) = 1.000000
Iteration-2, x1 = -0.764296 and fn(x1) = -0.535176
Iteration-3, x1 = -0.624857 and fn(x1) = -0.130106
Iteration-4, x1 = -0.582066 and fn(x1) = -0.025216
Iteration-5, x1 = -0.573268 and fn(x1) = -0.004543
Iteration-6, x1 = -0.571664 and fn(x1) = -0.000806
Iteration-7, x1 = -0.571379 and fn(x1) = -0.000143
```

5 Оценка погрешности найденного решения

Для первого уравнения

$$3x + 4x^3 - 12x^2 - 5 = 0$$

аналитическое решение

$$x = \frac{1}{2}(2 + \sqrt[3]{10 - \sqrt{73}} + \sqrt[3]{10 + \sqrt{73}})$$
$$x \approx 2.8901$$

Для второго уравнения

$$2\tan(x) - \frac{x}{2} + 1 = 0$$

Одно из численных решений с большей точностью: $x = -0.571317902831$.

Все методы успешно добились заданной точности $\epsilon = 10^{-4}$ в пределах погрешности $\pm 5 * 10^{-5}$.

Меньше всего итераций потребовалось используя метод Ньютона.