

ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ ДИНАМИКИ ИТЕРАЦИОННЫХ МЕТОДОВ НЬЮТОНА, ЧЕБЫШЕВА И ХЭЛЛИ НА КОМПЛЕКСНОЙ ПЛОСКОСТИ С ИСПОЛЬЗОВАНИЕМ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В.И.

Аннотация

В работе представлено численное исследование динамики итерационных методов Ньютона, Чебышева и Хэлли при решении нелинейных комплексных уравнений. Для визуализации поведения итерационных процессов построены фрактальные структуры бассейнов притяжения корней для различных функций, включая полиномиальные, тригонометрические, вложенные тригонометрические и гиперболические выражения. Производные первого и второго порядка вычислялись автоматически с использованием гипер-дуальных чисел. Особое внимание уделено областям расходимости, структуре границ бассейнов притяжения и влиянию параметра `maxIter` на форму фрактала.

1. Введение

Итерационные методы для решения нелинейных уравнений в комплексной области обладают богатой динамикой. Даже простые функции, такие как $f(z) = z^3 - 1$, порождают сложные фрактальные структуры, известные как фракталы Ньютона.

Методы более высокого порядка (Чебышева и Хэлли) — демонстрируют ещё более сложное поведение, включая:

- расширенные области расходимости,
- тонкие фрактальные границы,
- чувствительность к начальным условиям,
- появление хаотических зон.

Использование гипер-дуальных чисел над поле комплексных чисел (HDC) [1] позволяет автоматически получать значения $f(z)$, $f'(z)$ и $f''(z)$, что делает возможным построение фракталов для функций любой сложности.

2. Итерационные методы

В работе используются следующие итерационные методы [1, 2].

2.1. Метод Ньютона

$$z_{n+1} = z_n - K(z_n), \text{ где } K(z_n) = f(z_n)/f'(z_n)$$

Метод второго порядка, устойчивый и относительно простой.

2.2. Метод Чебышева

$$z_{n+1} = z_n - K(z_n)(1 + K(z_n)f''(z_n)/(2f'(z_n)))$$

Метод третьего порядка, более быстрый, но менее устойчивый.

2.3. Метод Хэлли

$$z_{n+1} = z_n - K(z_n)/(1 - f(z_n)f''(z_n)/(2f'(z_n)^2))$$

Метод четвёртого порядка, обладающий высокой скоростью сходимости, но чувствительный к малым значениям знаменателя.

3. Автоматическое дифференцирование с помощью гипер-дуальных чисел

Гипер-дуальные числа над полем комплексных чисел позволяют вычислять [1, 3]:

- значение функции $f(z)$,
- первую производную $f'(z)$,
- вторую производную $f''(z)$

за один вызов функции.

Это делает возможным построение фракталов для функций:

- $\sin(z)$
- $\sin(\cos(z))$
- $\sin(\cos(\tan(z)))$
- $\sin(\cos(\tan(\sinh(z))))$

и любых других вложенных выражений.

4. Бассейны притяжения и области расходимости

4.1. Бассейны притяжения

Для каждого корня уравнения существует область начальных значений, орбиты которых сходятся к этому корню. Эти области называются бассейнами притяжения. Визуально они образуют цветные зоны вокруг корней.

4.2. Области расходимости

Точки, для которых итерационный процесс:

- не достигает корня,
- уходит на бесконечность,
- попадает в циклы,
- или даёт слишком большие шаги,

образуют *множество расходимости*.

В визуализации они отображаются как *чёрные области*.

Интуитивно они напоминают «чёрные дыры», куда орбита «падает» и не возвращается — но в научной терминологии используется выражение *множество расходимости*.

5. Численные эксперименты

Исследование проводилось для функций:

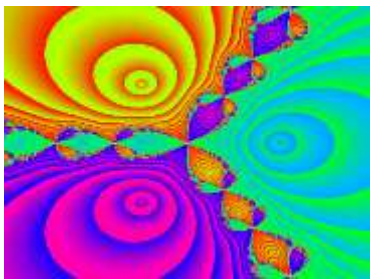
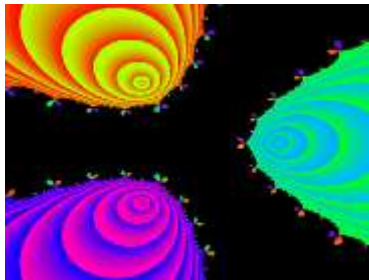
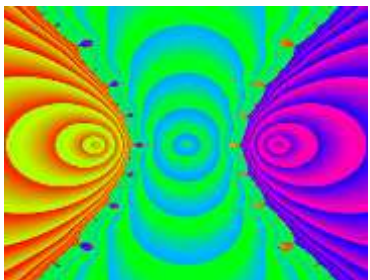
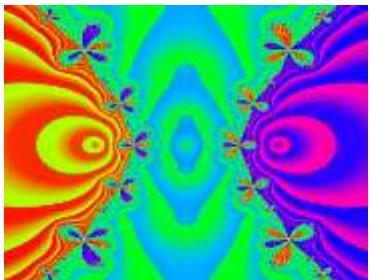
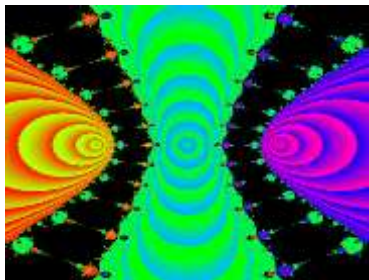
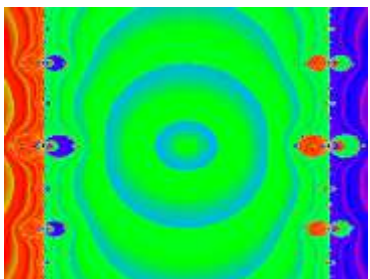
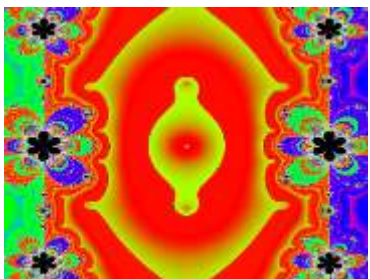
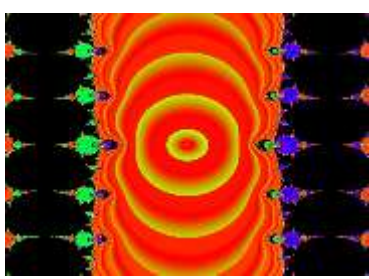
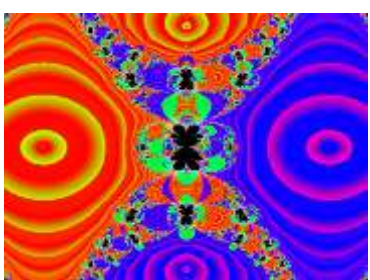
1. $f_1(z) = z^3 - 1$
2. $f_2(z) = z^3 - z$
3. $f_3(z) = \sin(z)$
4. $f_4(z) = \sin(\cos(z))$
5. $f_5(z) = \sin(\cos(\tan(z)))$
6. $f_5(z) = \sin(\cos(\tan(\sinh(z))))$

Для каждой функции строились фракталы методом:

- Ньютона,
- Чебышева,
- Хэлли.

Параметр $\max \text{Iter}$ варьировался от 20 до 100.

Ниже в табл.1 приводятся результаты расчёта для вышеуказанных функций.

	Ньютон	Чебышев	Хэлли
1			
2			
3			
4			

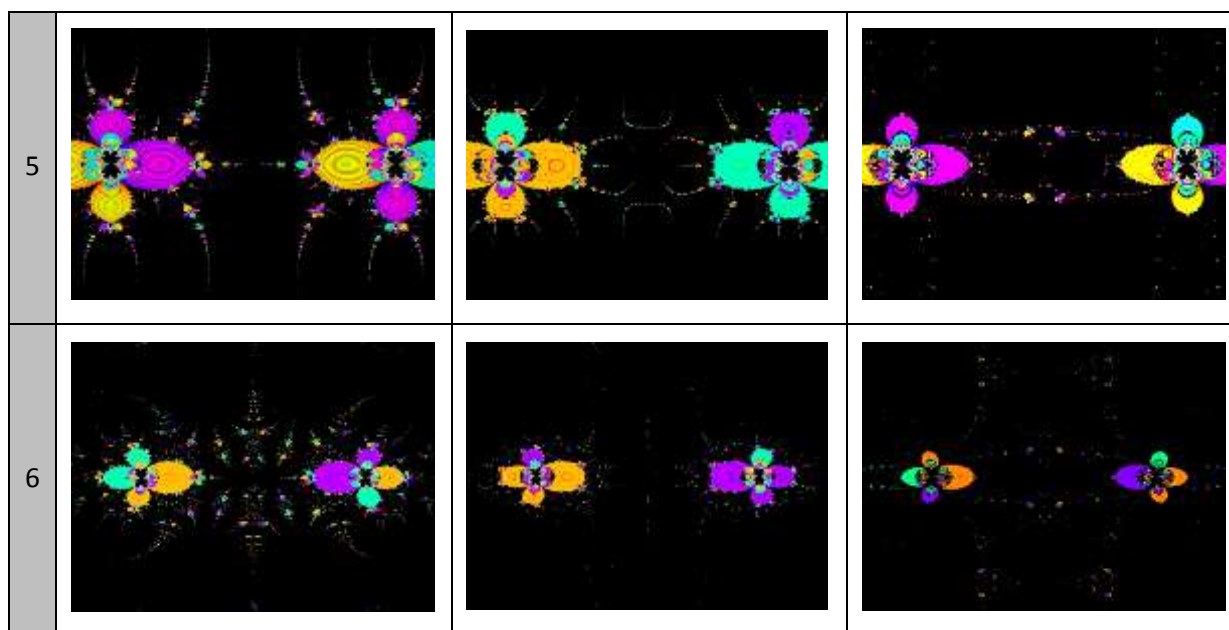


Табл.1. Фракталы для тестируемых функций.

6. Наблюдения и результаты

6.1. Влияние сложности функции

Чем сложнее функция (особенно с вложенными тригонометрическими и гиперболическими операциями), тем:

- больше полюсов,
- больше зон расходимости,
- более хаотичны границы бассейнов,
- выше чувствительность к начальным условиям.

6.2. Поведение методов

- Ньютон — наиболее устойчивый, даёт гладкие бассейны,
- Чебышев — более быстрый, но границы становятся тоньше и сложнее,
- Хэлли — самый быстрый в идеальных условиях, но часто «взрывается» из-за деления на малые значения,
- С увеличением порядка метода, как правило, растут области расходимости.

6.3. Влияние maxIter

Увеличение maxIter приводит к:

- появлению новых фрактальных деталей,

- расширению областей расходимости,
- увеличению времени вычислений (особенно для Хэлли),
- появлению тонких «нитей» и «лепестков» на границах.

7. Программная реализация

Реализация методов Ньютона, Чебышёва и Хэлли на базе HDC требует создания специализированного типа данных, обеспечивающего автоматическое вычисление производных первого и второго порядков. В рамках данной работы на языке C# была разработана структура **Cdn** (см. Приложение1), включающая перегруженные арифметические операции и функции элементарной математики для HDC. А также создан класс **Fractal**, содержащий процедуру **GetFractal(...)**, вычисляющую графический образ фрактала по заданной функции на заданной области комплексной плоскости. Эта процедура содержит три вложенных цикла анализирующих каждую точку исследуемой комплексной плоскости для выявления корней исходной функции и соответствующих областей притяжения. Перебор всех расчётных точек весьма ресурсоемкий, поэтому используются параллельные вычисления. Для вызова **GetFractal(...)** в список входных параметров входит указатель на исследуемую функцию, которая должна быть описана в терминах **Cdn**.

8. Вычислительная сложность

Метод Хэлли для функций $f_4(z)$, $f_5(z)$ и $f_6(z)$ оказался настолько тяжёлым, что при больших $\maxIter$ время вычисления на CPU достигало десятков минут.

Это связано с:

- большим числом тригонометрических операций,
- огромными значениями производных,
- частыми расходимостями,
- большим количеством итераций,
- размерами исследуемой области.

Все приведенные численные расчёты производились в среде Windows 10 с i5-4670 CPU @3.4 GHz.

Для дальнейшего исследования требуется переход на GPU (ComputeShader), где каждый пиксель обрабатывается отдельным потоком.

9. Заключение

В работе показано, что динамика итерационных методов на комплексной плоскости обладает чрезвычайно богатой структурой. Даже простые функции порождают сложные

фракталы, а методы высокого порядка демонстрируют хаотическое поведение вблизи полюсов и областей, где знаменатели итерационных формул малы.

Использование HDC делает возможным автоматическое дифференцирование функций любой сложности, а параллельные вычисления позволяют существенно ускорить построение фракталов.

Для функций с высокой степенью вложенности рекомендуется использование GPU вычислений.

10. Благодарности

Автор выражает благодарность всем принявшим участие в обсуждении вопросов, связанных с комплексной динамикой, программным обеспечением для фракталов с использованием параллельных процессов.

ПРИЛОЖЕНИЕ 1.

Код программы на C#

```
using System;
using System.Numerics;
using System.Numerics.Complex;
using System.Diagnostics;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Drawing;

public struct Cdn {
    public Complex Re, Im1, Im2;
    public Cdn(Complex x, Complex x1, Complex x2) { Re = x; Im1 = x1; Im2 = x2; }
    public Cdn(Complex x) { Re = x; Im1 = Complex.Zero; Im2 = Complex.Zero; }
    public Cdn(Complex x, Complex x1) { Re = x; Im1 = x1; Im2 = Complex.Zero; }
    #region OPERATORS (overloading):
    public static Cdn operator -(Cdn X) => new Cdn(-X.Re, -X.Im1, -X.Im2);
    public static Cdn operator +(Cdn X, Cdn Y) =>
        new Cdn(X.Re + Y.Re, X.Im1 + Y.Im1, X.Im2 + Y.Im2);
    public static Cdn operator -(Cdn X, Cdn Y) =>
        new Cdn(X.Re - Y.Re, X.Im1 - Y.Im1, X.Im2 - Y.Im2);
    public static Cdn operator *(Cdn X, Cdn Y)
    { return new Cdn(X.Re * Y.Re,
        X.Re * Y.Im1 + Y.Re * X.Im1,
        X.Re * Y.Im2 + 2.0 * X.Im1 * Y.Im1 + Y.Re * X.Im2); }
    public static Cdn operator *(Cdn X, Complex x) =>
        new Cdn(X.Re * x, X.Im1 * x, X.Im2 * x);
    public static Cdn operator *(Complex x, Cdn X) => X * x;
    public static Cdn operator /(Cdn X, Cdn Y) => X * Inverse(Y);
    public static implicit operator Cdn(double value) =>
        new(new Complex(value, 0), Complex.Zero);
    public static implicit operator Cdn(Complex value) => new(value, Complex.Zero);
    #endregion
    #region PUBLIC FUNCTIONS:
    public static Cdn Inverse(Cdn X)
```

```

{
    Complex x0 = X.Re, x1 = X.Im1, x2 = X.Im2,
    x0_2 = x0 * x0, x0_3 = x0_2 * x0,
    y0 = Complex.One / x0, y1 = -x1 / x0_2,
    y2 = (2.0 * x1 * x1 - x0 * x2) / x0_3;
    return new Cdn(y0, y1, y2);
}
public static Cdn Pow(Cdn Z, int n)
{
    if (n == 0) return new Cdn(Complex.One);
    Complex z = Z.Re, c1 = Complex.Pow(z, n),
    k1 = n * Complex.Pow(z, n - 1),
    k2 = (n > 1) ? n * (n - 1) * Complex.Pow(z, n - 2) : Complex.Zero,
    c2 = k1 * Z.Im1,
    c3 = k1 * Z.Im2 + k2 * Z.Im1 * Z.Im1;
    return new Cdn(c1, c2, c3);
}
public static Cdn exp(Cdn X)
{
    Complex ex = Complex.Exp(X.Re),
    c1 = ex * X.Im1, c2 = ex * X.Im2 + ex * X.Im1 * X.Im1;
    return new Cdn(ex, c1, c2);
}
public static Cdn sin(Cdn X)
{
    Complex x = X.Re, s = Complex.Sin(x), c = Complex.Cos(x),
    c1 = c * X.Im1, c2 = c * X.Im2 - s * X.Im1 * X.Im1;
    return new Cdn(s, c1, c2);
}
public static Cdn cos(Cdn X)
{
    Complex x = X.Re, s = Complex.Sin(x), c = Complex.Cos(x),
    c1 = -s * X.Im1, c2 = -s * X.Im2 - c * X.Im1 * X.Im1;
    return new Cdn(c, c1, c2);
}
public static Cdn tg(Cdn X) { return sin(X) / cos(X); }
public static Cdn log(Cdn X) // ln
{
    Complex x = X.Re, c1 = X.Im1 / x,
    c2 = X.Im2 / x - (X.Im1 * X.Im1) / (x * x);
    return new Cdn(Complex.Log(x), c1, c2);
}
public static Cdn sinh(Cdn X)
{
    Complex x = X.Re, sh = Complex.Sinh(x), ch = Complex.Cosh(x),
    c1 = ch * X.Im1, c2 = ch * X.Im2 + sh * X.Im1 * X.Im1;
    return new Cdn(sh, c1, c2);
}
public static Cdn cosh(Cdn X)
{
    Complex x = X.Re, sh = Complex.Sinh(x), ch = Complex.Cosh(x),
    c1 = sh * X.Im1, c2 = sh * X.Im2 + ch * X.Im1 * X.Im1;
    return new Cdn(ch, c1, c2);
}
public static Cdn th(Cdn X) { return sinh(X) / cosh(X); }
public static Cdn sqrt(Cdn X)
{
    Complex x = X.Re, r = Complex.Sqrt(x),
    c1 = X.Im1 / (2.0 * r),
    c2 = X.Im2 / (2.0 * r) - (X.Im1 * X.Im1) / (4.0 * r * x);
    return new Cdn(r, c1, c2);
}
public override string ToString() => $"{Re} + {Im1}ε + {Im2}ω";
}
#endregion
}

class Fractal
{

```

```
public enum Method
{
    Newton,      // 0
    Chebyshev,   // 1
    Halley       // 2
}

public static Bitmap GetFractal(
    int width, int height, int maxIter,
    double xmin, double xmax, double ymin, double ymax,
    Func<Cdn, Cdn> Ф, Method m = Method.Newton)
{
    // Потокбезопасный буфер пикселей
    Color[,] pixels = new Color[width, height];
    // Список корней – общий ресурс
    List<Complex> roots = new List<Complex>();
    object rootLock = new object();
    double epsRoot = 1e-6;
    // Параллельный проход по строкам
    Parallel.For(0, height, py =>
    {
        double y = ymin + (ymax - ymin) * py / (height - 1);
        for (int px = 0; px < width; px++)
        {
            double x = xmin + (xmax - xmin) * px / (width - 1);
            Complex z = new Complex(x, y);
            int iter = 0;
            int rootIndex = -1;
            Complex root = Complex.Zero;
            while (iter < maxIter)
            {
                Cdn F = Ф(new Cdn(z, Complex.One));
                Complex f = F.Re, fp = F.Im1, fpp = F.Im2;
                Complex r = f / fp;
                Complex ratio = Complex.Zero, corr = ratio;
                switch (m)
                {
                    case Method.Newton:
                    {
                        z = z - r;
                        break;
                    }
                    case Method.Chebyshev:
                    {
                        corr = Complex.One + 0.5 * (f * fpp) / (fp * fp);
                        z = z - r * corr;
                        break;
                    }
                    case Method.Halley:
                    {
                        ratio = f * fpp / (fp * fp);
                        corr = Complex.One + 0.5 * ratio;
                        z = z - r / corr;
                        break;
                    }
                }
                iter++;
                // проверяем попадание в существующий корень
                for (int k = 0; k < roots.Count; k++)
                {
                    if ((z - roots[k]).Magnitude < epsRoot)
                    {
                        rootIndex = k;
                        root = roots[k];
                        break;
                    }
                }
            }
            if (rootIndex >= 0) break;
            // новый корень
            if (f.Magnitude < epsRoot)
            {
                lock (rootLock)
                {
                    roots.Add(z);
                    rootIndex = roots.Count - 1;
                    root = z;
                }
            }
        }
    });
}
```

```
        break;
    }
    // расходимость
    if (z.Magnitude > 1e6) break;
}
// паскраска
Color color;
if (rootIndex < 0){ color = Color.Black; }
else
{
    double dist = (z - root).Magnitude;
    dist = Math.Max(dist, 1e-12);
    double mu = iter + 1 - Math.Log(dist);
    double t = mu / maxIter;
    t = Math.Max(0, Math.Min(1, t));
    int k = rootIndex % 3;
    int r, g, b;
    if (k == 0)
    {
        r = (int)(255 * t);
        g = (int)(128 + 127 * Math.Sin(6.28 * t));
        b = 0;
    }
    else if (k == 1)
    {
        r = 0;
        g = (int)(255 * t);
        b = (int)(128 + 127 * Math.Sin(6.28 * t));
    }
    else
    {
        r = (int)(128 + 127 * Math.Sin(6.28 * t));
        g = 0;
        b = (int)(255 * t);
    }
    color = Color.FromArgb(r, g, b);
}
pixels[px, py] = color;
}
});
// Собираем Bitmap
Bitmap bmp = new Bitmap(width, height);
for (int py = 0; py < height; py++)
    for (int px = 0; px < width; px++)
        bmp.SetPixel(px, py, pixels[px, py]);
return bmp;
}
}

//ПРИМЕР РАСЧЁТА
Cdn F1(Cdn z) { return z * z * z - new Cdn(new Complex(1, 0)); }
Cdn F3(Cdn z){ return Cdn.sin(Cdn.cos(z)); }

var bmp = testFractalV.GetFractal(
    width: 800, height: 600, maxIter: 40,
    xmin: -2, xmax: 2, ymin: -2, ymax: 2,
    F1, testFractalV.Method.Halley);
bmp = new Bitmap(bmp, new Size(bmp.Width / 2, bmp.Height / 2));
pictureBox1.Image = bmp;
```

ЛИТЕРАТУРА

1. Олифер В.И. Численное решение комплексных уравнений методом Чебышева 3-го порядка. URL: http://viosolutions.amerihomesrealty.com/pdf/решение_комплексных_уравнений.pdf, – 2026.
2. Олифер В.И. Implementation of Chebyshev-Halley type methods based on hyper-dual numbers. URL: http://viosolutions.amerihomesrealty.com/pdf/implementation_of_chebyshev-halley_type_methods_based_on_hyper-dual_numbers.pdf, – 2023.
3. Naumann U. The Art of Differentiating Computer Programs. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, – 2012.
4. Минькова Р.М. Функции комплексного переменного и операционное исчисление. – Екатеринбург: УГТУ-УПИ, 2013. – 72 с.

Абстракт

В работе представлено численное исследование динамики итерационных методов Ньютона, Чебышева и Хэлли при решении нелинейных комплексных уравнений. Дана визуализация поведения итерационных процессов, построены фрактальные структуры для различных функций, включая полиномиальные, тригонометрические, вложенные тригонометрические и гиперболические выражения. Использовано автоматическое дифференцирование. Приведен код программного обеспечения и рассмотрены численные примеры, полученные на его основе.

This paper presents a numerical study of the dynamics of Newton's, Chebyshev's, and Halley's iterative methods for solving nonlinear complex equations. Visualizations of the iterative processes are provided, and fractal structures are generated for various functions, including polynomial, trigonometric, nested trigonometric, and hyperbolic expressions. Automatic differentiation is employed. The software code is included, and numerical examples obtained using it are discussed.

Ключевые слова комплексные уравнения, гипер-дуальные числа над полем комплексных чисел, методы Ньютона, Чебышева, Хэлли, автоматическое дифференцирование, фракталы, complex equations, hyper-dual numbers over the field of complex numbers, Newton's, Chebyshev's, and Halley's methods, automatic differentiation, fractals.