Содержание задания

1. Решить обратную задачу теории погрешностей для заданной функции $z(x)$ с допуском $\varepsilon = 10^{-6}$. Выполняется аналитически.
2. Найти с требуемой точностью приближенные значения $\overline{z}(x)$ этой функции (квадратный корень вычислять по формуле Герона, а остальные элементарные функции вычислять с использованием степенных рядов, указанных ниже).
3. Найти «точные» значения $z(x)$, используя встроенные функции языка программирования. Убедиться, что абсолютная погрешность вычислений удовлетворяет заданному допуску ε. Итоговый вывод программы должен представлять собой таблицу

$x \quad z(x) \quad \overline{z}(x) \quad |\triangle z(x)|$

. . . .

. . . .

. . . .

причём погрешности должны быть меньше 10^{-6}, однако не должны быть слишком маленькими (не меньше 10^{-10}).

Задание

№ 1:

$$z(x) = \frac{\sqrt{1 + arctg(16.7x + 0.1)}}{cos(7x + 0.3)}, \qquad x = 0.01(0.005)0.05; \tag{1}$$

Вычисление погрешностей

Источниками погрешности (при точных значениях x) являются три функции: арктангенс, косинус и квадратный корень. Положим:

$u(x) = arctg(16.7x + 0.1)$
$v(x) = cos(7x + 0.3)$
$p(x) = \sqrt{1 + u(x)}$
$f(x) = \frac{p(x)}{v(x)}$

Значит, считая x точным, получаем:
$z(x) = f(p(u(x)), v(x)) = \frac{p(u)}{v}$

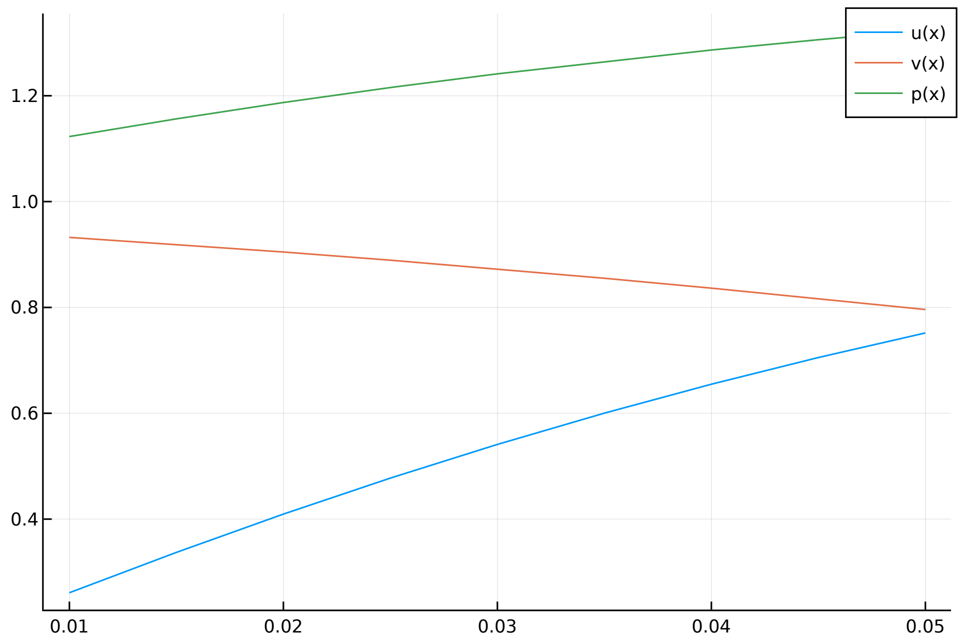
Рассмотрим функции $u(x)$, $v(x)$ и $p(x)$ на промежутке $x = 0.01(0.005)0.05$:

[ ]:
```
x = 0.01:0.005:0.05;
u(x) = atan.(16.7*x .+ 0.1);
v(x) = cos.(7x .+ 0.3);
```

```
p(x) = sqrt.(1 .+ u(x))

plot(x, u(x), label = "u(x)")
plot!(x, v(x), label = "v(x)")
plot!(x, p(x), label = "p(x)")
```

[ ]:



По графикам видим, что все три функции монотонны:

$u(0.01) \leq u(x) \leq u(0.05)$
$v(0.05) \leq v(x) \leq v(0.01)$
$p(0.01) \leq p(x) \leq p(0.05)$

[ ]:
```
println("u(0.01) = ", u(0.01), ", u(0.05) = ", u(0.05))
println("v(0.05) = ", v(0.05), ", v(0.01) = ", v(0.01))
println("p(0.01) = ", p(0.01), ", p(0.05) = ", p(0.05))
```

```
u(0.01) = 0.26091356923294057, u(0.05) = 0.7518190586029678
v(0.05) = 0.7960838022471798, v(0.01) = 0.9323273456192719
p(0.01) = 1.1229040783757713, p(0.05) = 1.323563016483525
```

Получаем после округления:

$0.26 \leq u(x) \leq 0.76$

$0.79 \leq v(x) \leq 0.94$
$1.12 \leq p(x) \leq 1.33$

Оценим теперь частные производные функций:

$$\frac{\partial f}{\partial p} = \frac{1}{v}, \quad \frac{\partial f}{\partial v} = -\frac{p}{v^2}, \quad \frac{\partial f}{\partial u} = \frac{1}{2v\sqrt{1+u}}$$

$$|\frac{\partial f}{\partial p}| \leq \frac{1}{0.79} < 1.27 = c_p, \quad |\frac{\partial f}{\partial v}| \leq \frac{1.33}{0.79^2} < 2.14 = c_v, \quad |\frac{\partial f}{\partial u}| \leq \frac{1}{2 \cdot 0.79 \cdot \sqrt{1+0.26}} < 0.57 = c_u$$

Т.к. $\varepsilon_f = \varepsilon = 10^{-6}$ и $\varepsilon_f = c_u\varepsilon_u + c_v\varepsilon_v + c_p\varepsilon_p$, то, применяя принцип равных влияний, получаем:

$$\varepsilon_u = \frac{\varepsilon}{3c_u} = \frac{10^{-6}}{3 \cdot 0.57} \approx 5.85 * 10^{-7}$$
$$\varepsilon_v = \frac{\varepsilon}{3c_v} = \frac{10^{-6}}{3 \cdot 2.14} \approx 1.56 * 10^{-7}$$
$$\varepsilon_p = \frac{\varepsilon}{3c_p} = \frac{10^{-6}}{3 \cdot 1.27} \approx 2.63 * 10^{-7}$$

Степенные ряды для элементарных функций и оценки их остатков

Через u_k обозначен k-й член ряда, а через R_n — остаток частичной суммы $S_n = \sum_{k=0}^{n} u_k$.

$$1. \quad arctg(x) = \begin{cases} \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k+1}}{2k+1}, & |x| < 1 \\ \frac{\pi}{2}sgn(x) - \sum_{k=0}^{\infty}(-1)^k \frac{x^{-(2k+1)}}{2k+1}, & |x| \geq 1 \end{cases} \tag{2}$$

в обоих случаях $|R_n(x)| \leq |u_n(x)|$.

$$2. \quad cos(x) = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{(2k)!}, \quad |R_n(x)| \leq |u_n(x)| \tag{3}$$

Реализация программы

[ ]:
```
function my_sqrt(x, eps)
    last = (1. - x) / 2.;
    num = 1. - last;
    while last > eps || last < -eps
        last = (num / 2.) - (x / (2. * num));
        num -= last;
    end;
    return num;
end;
```

[ ]:
```
function my_cos(x, eps)
    sum = 1.;
    last = 1.;
    step = 1;

    while last > eps || last < -eps
        last *= (-x * x) / ((2. * step)*(2. * step - 1));
        sum += last;
        step += 1;
    end;

    return sum;
end;
```

[ ]:
```
function my_atan(x, eps)
    if x >= 1
        sum = 1/x;
        last = 1/x;
        step = 1;

        while last > eps || last < -eps
            last *= (2step - 1) / ((2step + 1)*(-x * x));
            sum += last;
            step += 1;
        end;

        return pi/2 * sign(x) - sum;
    end;

    sum = x;
    last = x;
    step = 1;

    while last > eps || last < -eps
        last *= (-x * x)*(2step - 1) / (2step + 1);
        sum += last;
        step += 1;
    end;

    return sum;
end;
```

[ ]:
```
my_u(x, eps_u) = my_atan.(16.7*x .+ 0.1, eps);
my_v(x, eps_v) = my_cos.(7x .+ 0.3, eps);
my_p(x, eps_p, eps_u) = my_sqrt.(1 .+ my_u(x, eps_u), eps_p);
my_f(x, eps_u, eps_v, eps_p) = my_p(x, eps_p, eps_u) ./ my_v(x, 
 ↪eps_v);
```

```
function my_z(x, eps)
    eps_u = eps / (3*0.57);
    eps_v = eps / (3*2.14);
    eps_p = eps / (3*1.27);

    return my_f(x, eps_u, eps_v, eps_p);
end;

u(x) = atan.(16.7*x .+ 0.1);
v(x) = cos.(7x .+ 0.3);
p(x) = sqrt.(1 .+ u(x))

z(x) = p(x) ./ v(x);
```

Проверка и демонстрация результатов

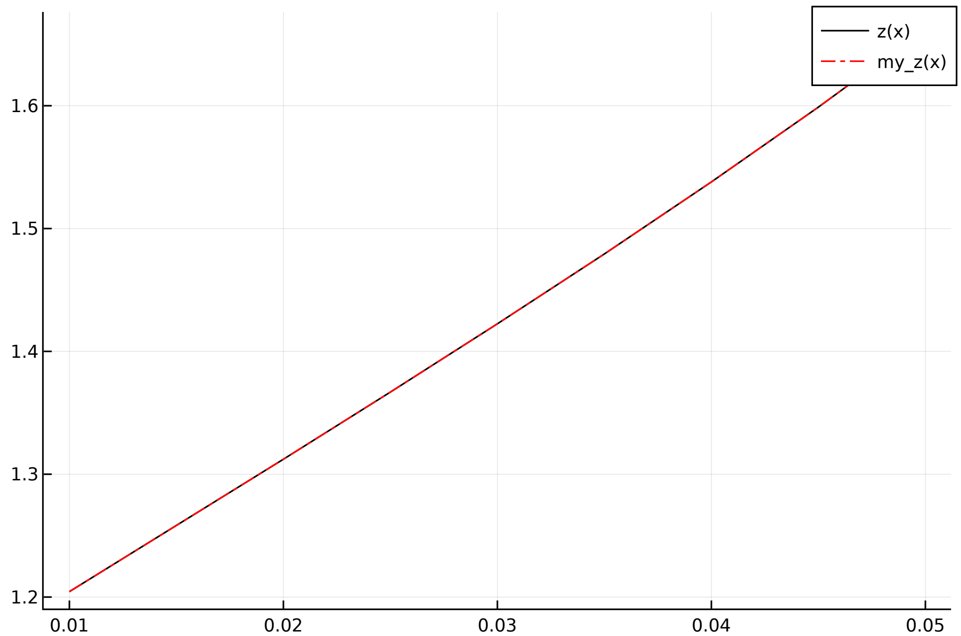
[ ]:
```
x = 0.01:0.005:0.05;
eps = 1e-6;

abs.(my_z(x, eps) .- z(x)) .< 1e-6
```

[ ]:
```
9-element BitVector:
 1
 1
 1
 1
 1
 1
 1
 1
 1
```

[ ]:
```
plot(x, z(x), linecolor = :black, label = "z(x)")
plot!(x, my_z(x, eps), line = :dashdot, linecolor = :red, label =
 ↪"my_z(x)")
```

[ ]:




```
[ ]: println("  i\t\tz(i)\t\t    my_z(i, eps)       abs(my_z(i, eps) -␣
      ↪z(i))\n")
     for i in x
         println(i, '\t', z(i), '\t', my_z(i, eps), '\t', abs(my_z(i, eps)␣
      ↪- z(i)))
     end;
```

```
  i              z(i)                    my_z(i, eps)       abs(my_z(i,␣
 ↪eps) - z(i))

0.01    1.2044096782658609      1.2044096983978654      2.
 ↪0132004507900092e-8
0.015   1.2581178482475504      1.2581178088330358      3.
 ↪941451454103628e-8
0.02    1.3121949697163693      1.3121949536005124      1.
 ↪6115856871223855e-8
0.025   1.366867024093271       1.3668670959762919      7.
 ↪188302086902354e-8
0.03    1.4224465640692616      1.4224465230548875      4.
 ↪101437411918596e-8
0.035   1.4793222591232789      1.4793223407457339      8.
 ↪162245501708298e-8
```

```
0.04    1.5379499479031917      1.5379498031936734      1.
  ↪4470951836820234e-7
0.045   1.5988472167969436      1.5988473914495742      1.
  ↪7465263058014102e-7
0.05    1.6625925722234023      1.6625927888271876      2.
  ↪1660378535237612e-7
```