Численные методы

Решение нелинейных уравнений

$$f(x) = 0, \quad f : \mathbb{R} \to \mathbb{R}$$

Говорят, что корень непрерывной функции f *локализован* на интервале (a, b), если $f(a)$ и $f(b)$ имеют разные знаки.

- На (a, b) нечётное число простых корней
- На (a, b)
  - нет корней
  - один кратный корень
  - чётное число простых корней
- На (a, b) (бесконечно) много простых корней $(\sin \frac{1}{x})$
- На (a, b) функция f имеет разрыв второго рода $(1/(x - c))$

Первоначальное определение отрезка (a, b)

```
function [res,bnds]=zbrac(f, x1, x2)
    n_max = 50;  FACTOR = 1.6;
    f1 = f(x1);  f2 = f(x2);
    res = 0;     bnds = [x1,x2];
    for i=1:n_max
        if f1*f2<0
            res = 1;  bnds = [x1,x2];
            break
        endif
        if abs(f1)<abs(f2)
            x1 = x1 + FACTOR*(x1-x2);
            f1 = f(x1);
        else
            x2 = x2 + FACTOR*(x2-x1);
            f2 = f(x2);
        endif
    endfor
endfunction
```

Метод деления отрезка пополам

```
function [x,y]=rtbis(f, bnds, err=1e-5)
    x1 = bnds(1); f1 = f(x1);
    x2 = bnds(2); f2 = f(x2);
    while abs(x1-x2)>err
        xm = (x1+x2)/2; fm = f(xm);
        if fm==0
            break;
        endif
        if f1*fm > 0
            x1 = xm; f1 = fm;
        else
            x2 = xm; f2 = fm;
        endif
    endwhile
    x = (x1+x2)/2; y = f(x);
endfunction
```

Использование линейной интерполяции

- Метод секущих

  x_{i-1}, x_i — самые «свежие» точки

  Более быстрый
- Метод хорд

  x_{i-1}, x_i локализуют корень

  Более устойчивый

Метод секущих

```
function [x,y] = rtsec(f, bnds, xacc=1e-5)
    x1 = bnds(1);  f1 = f(x1);
    x2 = bnds(2);  f2 = f(x2);
    for i=1:30
        dx = f1*(x2-x1)/(f2-f1);
        x3 = x1-dx; f3 = f(x3);

        if abs(dx)<xacc
            break
        endif

        x1 = x2; f1 = f2;
        x2 = x3; f2 = f3;
    endfor
    x = x3;
    y = f(x);
endfunction
```

Метод хорд

```
function [x,y] = rtchr(f, bnds, xacc=1e-5)
    x1 = bnds(1);  f1 = f(x1);
    x2 = bnds(2);  f2 = f(x2);
    for i=1:30
        dx = f1*(x2-x1)/(f2-f1);
        x3 = x1-dx; f3 = f(x3);

        if abs(dx)<xacc
            break
        endif

        if f1*f3>0
          x1 = x3; f1 = f3;
        else
          x2 = x3; f2 = f3;
        endif
    endfor
    x = x3;
    y = f(x);
endfunction
```

Метод Риддерса (Ridders' method)

- x_1, x_2 локализуют корень
- $x_3 = (x_1 + x_2)/2$
- вычисление экспоненциальной функции e^Q такой, что

$$f(x_1) - 2f(x_3)e^Q + f(x_3)e^{2Q} = 0,$$

$$e^Q = \frac{f(x_3) + \operatorname{sgn}\big(f(x_2)\big)\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}{f(x_2)}$$

- метод хорд применяется для точек $f(x_1)$, $f(x_3)e^Q$, $f(x_2)e^{2Q}$

$$x_4 = x_3 + (x_3 - x_1)\frac{\operatorname{sgn}\big(f(x_1) - f(x_2)\big)f(x_3)}{\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}$$

Метод Риддерса: реализация

```
function [x,y,n]=rtridd(f, bnds, err=1e-5)
    x1 = bnds(1);  f1 = f(x1);
    x2 = bnds(2);  f2 = f(x2);

    d = err*2;
    while abs(d)>err
        x3 = (x1+x2)/2; f3 = f(x3);
        d = (x3-x1)*f3/(f3^2-f1*f2)^0.5;
        if f1>f2
            x4 = x3+d;
        else
            x4=x3-d;
        endif
        f4 = f(x4);
        if f1*f4>0
            x1 = x4; f1 = f4;
        else
            x2 = x4; f2 = f4;
        endif
    endwhile
    x = x4; y = f(x);
endfunction
```

Метод Брента

Комбинация

- локализации корней
- метода деления отрезка пополам
- обратной квадратической интерполяции

Метод Брента

Обратная квадратическая интерполяция:
(a, f_a), (b, f_b), (c, f_c)

$$x = \frac{(y - f_a)(y - f_b)c}{(f_c - f_a)(f_c - f_b)} + \frac{(y - f_b)(y - f_b)a}{(f_a - f_b)(f_a - f_b)} + \frac{(y - f_a)(y - f_c)b}{(f_b - f_a)(f_b - f_c)}$$

$$y = 0 \quad \Rightarrow \quad x = b + P/Q$$

$$P = S\big(T(R - T)(c - b) - (1 - R)(b - a)\big), \quad Q = (T - 1)(R - 1)(S - 1)$$

$$R \equiv \frac{f(b)}{f(c)}, \quad S \equiv \frac{f(b)}{f(a)}, \quad T \equiv \frac{f(a)}{f(c)}$$

Метод Брента: реализация

```
function [x,y] = rtbrent(f, bnds, err=1e-5)
    xa = bnds(1);  fa = f(xa);
    xc = bnds(2);  fc = f(xc);

    d = err*2;
    while abs(xc-xa)>err && d>err^2
      # деление отрезка пополам
      xb = (xa+xc)/2;  fb = f(xb);
      # обратная квадратичная интерполяция
      R = fb/fc;  S = fb/fa;  T = fa/fc;
      P = S*(T*(R-T)*(xc-xb)-(1-R)*(xb-xa));
      Q = (T-1)*(R-1)*(S-1);
      xd = xb+P/Q;      fd = f(xd);

      if xa<=xd && xd<=xc && abs(fd)<abs(fb)
        # лучше обратная квадратичная интерполяция
        if fa*fd>0
          d = abs(xd-xa); xa = xd;  fa = fd;
        else
          d = abs(xd-xc); xc = xd;  fc = fd;
        endif
```

Метод Брента: реализация (продолжение)

```
    else
      # лучше деление отрезка пополам
      if fa*fb>0
        d = abs(xb-xa); xa = xb;  fa = fb;
      else
        d = abs(xb-xc); xc = xb;  fc = fb;
      endif
    endif
  endwhile

  if d<=err^2
    if abs(fa)<abs(fc)
      x = xa; y = fa;
    else
      x = xc; y = fc;
    endif
  else
    x = (xa+xc)/2; y = f(x);
  endif
endfunction
```

Метод Ньютона

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{f''(x)}{2!}\delta^2 + \dots$$

$$f(x+\delta) = 0 \quad \Rightarrow \quad \delta = \frac{f(x)}{f'(x)}$$

$$x_0, \qquad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 1, 2, \dots$$

Метод Ньютона: реализация

```
function [x,y]=rtnewton(f,df,x0,xacc)
    x = x0;
    for i=1:30
        dx = f(x)/df(x);
        x = x-dx;
        if abs(dx)<xacc then
            break
        end
    end
    y = f(x);
endfunction
```

Сравнение методов

Уравнение и точное решение:

$$\sin(x) - 0.9 = 0, \qquad x = 1.1197695149986\ldots$$

Начальное приближение:

$$[0, 1.18813760] \text{ или } 0.59406880$$

Метод	x	$f(x)$	**Итераций**
деления пополам	1.11976656844482	-1.284×10^{-6}	18
секущих	1.11976951499863	0	6
хорд	1.11976951500357	2.151×10^{-12}	30
Риддерса	1.11976951499863	2.220×10^{-16}	21
Брента	1.11976951502245	1.038×10^{-11}	17
Ньютона	1.11976951499863	0	5

Метод простых итераций

Уравнение

$$f(x) = 0$$

приводится к виду

$$x = \phi(x).$$

$$x_0, \qquad x_i = \phi(x_{i-1}), \quad i = 1, 2, \ldots$$

$$\begin{aligned} x_i - x_{i-1} =& \phi(x_{i-1}) - \phi(x_{i-2}) \\ =& \phi'(x_{i-2})(x_{i-2} - x_{i-1}) + O\big((x_{i-2} - x_{i-1})^2\big) \end{aligned}$$

Метод Ньютона для системы нелинейных уравнений

Не существует хорошего общего метода для решения систем, состоящих более чем из одного нелинейного уравнения.

$$f(x, y) = 0, \quad g(x, y) = 0$$

Метод Ньютона для системы нелинейных уравнений

$$f(x) = 0, \quad f : \mathbb{R}^n \to \mathbb{R}^n$$

$$f_i(x + \delta x) = f_i(x) + \sum_{j=1}^{n} \frac{\partial f_i(x)}{\partial x_j} \delta x_j + O(\delta x^2)$$

J — матрица Якоби системы уравнений:

$$J_{ij}(x) = \frac{\partial f_i(x)}{\partial x_j}$$

$$f(x + \delta x) = f(x) + J(x) \cdot \delta x + O(\delta x^2)$$

Метод Ньютона для системы нелинейных уравнений

$$J(x) \cdot \delta x = -f(x)$$

$$x_{\text{нов}} = x_{\text{пред}} + \delta x$$

Глобально сходящийся метод для системы нелинейных уравнений

$$f(x) = 0$$

$$x_{\text{нов}} = x_{\text{пред}} + \delta x$$

$$\delta x = -J(x)^{-1} \cdot f(x)$$

$$F = \frac{1}{2} f \cdot f \to \min$$

Глобально сходящийся метод для системы нелинейных уравнений

Шаг метода Ньютона δx — направление убывания для F:

$$\nabla F \cdot \delta x = (f \cdot J) \cdot (-J^{-1} f) = -f \cdot f < 0$$

Глобально сходящийся метод для системы нелинейных уравнений

- Сделать полный шаг по методу Ньютона
- Если F уменьшилась — ок
- Если F не уменьшилась — поиск в обратном направлении (между x и $x + \delta x$) минимума F

Локализация корней многочлена

$$f(z) = z^n + a_1 z^{n-1} + \cdots + a_n, \quad a_i \in \mathbb{C}$$

Внутри круга $|z| = 1 + \max_i |a_i|$ расположено ровно n корней многочлена f (с учётом их кратностей).

Метод Мюллера

Обобщение метода секущих с использованием квадратичной интерполяции

Дано: x_{i-2}, x_{i-1}, x_i.

$$q = \frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}}$$

$$A = qf_i - q(1+q)f_{i-1} + q^2 f_{i-2}$$
$$B = (2q+1)f_i - (1+q)^2 f_{i-1} + q^2 f_{i-2}$$
$$C = (1+q)f_i$$

$$x_{i+1} = x_i - \frac{2C(x_i - x_{i-1})}{B \pm \sqrt{B^2 - 4AC}}$$

Метод Лагерра

$$P_n(x) = (x - x_1)(x - x_2)\dots(x - x_n)$$

$$\ln|P_n(x)| = \ln|x - x_1| + \ln|x - x_2| + \dots + \ln|x - x_n|$$

$$\frac{d\ln|P_n(x)|}{dx} = \frac{1}{x - x_1} + \frac{1}{x - x_2} + \dots + \frac{1}{x - x_n} = \frac{P_n'(x)}{P_n(x)} \equiv G$$

$$-\frac{d^2\ln|P_n(x)|}{dx^2} = \frac{1}{(x - x_1)^2} + \frac{1}{(x - x_2)^2} + \dots + \frac{1}{(x - x_n)^2} =$$

$$= \left(\frac{P_n'(x)}{P_n(x)}\right)^2 - \frac{P_n''(x)}{P_n(x)} \equiv H$$

Локализация корней многочлена

Метод Лагерра

Предположение: искомый корень x_1 находится на расстоянии a от текущего приближения x, тогда как *все остальные корни* находятся на расстоянии b:

$$x - x_1 = a, \qquad x - x_i = b, \quad i = 2, 3, \ldots, n.$$

$$\frac{1}{a} + \frac{n-1}{b} = G$$
$$\frac{1}{a^2} + \frac{n-1^2}{b} = H$$

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}$$

Знак выбирается так, чтобы модуль числителя был наибольшим.

Локализация корней многочлена

$$
\begin{aligned}
f_0 &= f(x), \\
f_1 &= f'(x), \\
f_{i-2} &= q_{i-1} f_{i-1} - f_i, \quad i = 2, \ldots, n
\end{aligned}
$$

$f_0, f_1, \ldots, f_n$ — последовательность Штурма

Теорема Штурма

Пусть $w(x)$ — число перемен знака в последовательности Штурма. Тогда количество (вещественных) корней многочлена f (без учёта их кратности), заключённых между a и b, где $f(a) \neq 0$ и $f(b) \neq 0$ и $a < b$, в точности равно $w(a) - w(b)$.