Компьютерные технологии в научных исследованиях и образовании

Юрий Анатольевич Богоявленский, заведующий кафедрой Информатики и математического обеспечения, к.т.н., доцент, ybgv

Примеры применения функций численных методов

Решение системы линейных уравнений методом Гаусса

Поиск корней полинома

Методы интерполяции

Аппроксимация по методу наименьших квадратов

Решение системы линейных уравнений методом Гаусса

Этот метод реализован в функции `rref`, которую можно вызвать в формах:

```
rref (A)
rref (A, TOL)
[R, K] = rref (...)
```

Назначение параметров.

- `A` — расширенная матрица системы линейных уравнений;
- `TOL` — параметр неустойчивости, по умолчанию имеет значение `eps * max (size (A)) * norm (A, inf);`
- `R` — результат - матрица приведённого ступенчатого вида по строкам;
- `K` — необязательный параметр содержащий вектор "связанных переменных", которые указывают на столбцы, для которых было выполнено исключение.

Значение функции — матрица приведённого ступенчатого вида по строкам.

```
ybgv@ybgv-home:~> octave  -q -p ~/MyOct -p ~/MyOct/Plotting/
octave:1> echo on
octave:2> Paslau
+ %Решение СЛАУ методом Гаусса
+ disp('Решение СЛАУ Ax=b методом Гаусса');
Решение СЛАУ Ax=b методом Гаусса
+
+ disp('Матрица системы:');
Матрица системы:
+ A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6]
A =

   2   1  -5   1
   1  -3   0  -6
   0   2  -1   2
   1   4  -7   6

+
+ disp('Вектор свободных коэффициентов:');
Вектор свободных коэффициентов:
+ b=[8;9;-5;0]
b =

   8
   9
  -5
   0

+
```

```
+ disp('Расширенная матрица системы:');
Расширенная матрица системы:
+ [A b]
ans =

   2   1  -5   1   8
   1  -3   0  -6   9
   0   2  -1   2  -5
   1   4  -7   6   0

+
+ disp("Матрица приведённого ступенчатого вида по строкам")
Матрица приведённого ступенчатого вида по строкам
+ C=rref([A b])
C =

   1   0   0   0   3
   0   1   0   0  -4
   0   0   1   0  -1
   0   0   0   1   1

+
+ disp('Размерность матрицы C:');
Размерность матрицы C:
+ n=size(C)
n =

   4   5

+
+ disp('Вектор решений СЛАУ Ax=b');
Вектор решений СЛАУ Ax=b
```

```
+ x=C(:,n(2))
x =


   3
  -4
  -1
   1


+
+ disp('Проверка Ax-b');
Проверка Ax-b
+ A*x-b
ans =


            0
   1.7764e-15
            0
  -2.6645e-15


+
octave:3> ^C
octave:3> eps * max (size (A)) * norm (A, inf)
ans = 1.5987e-14
octave:4>
```

Поиск корней полинома

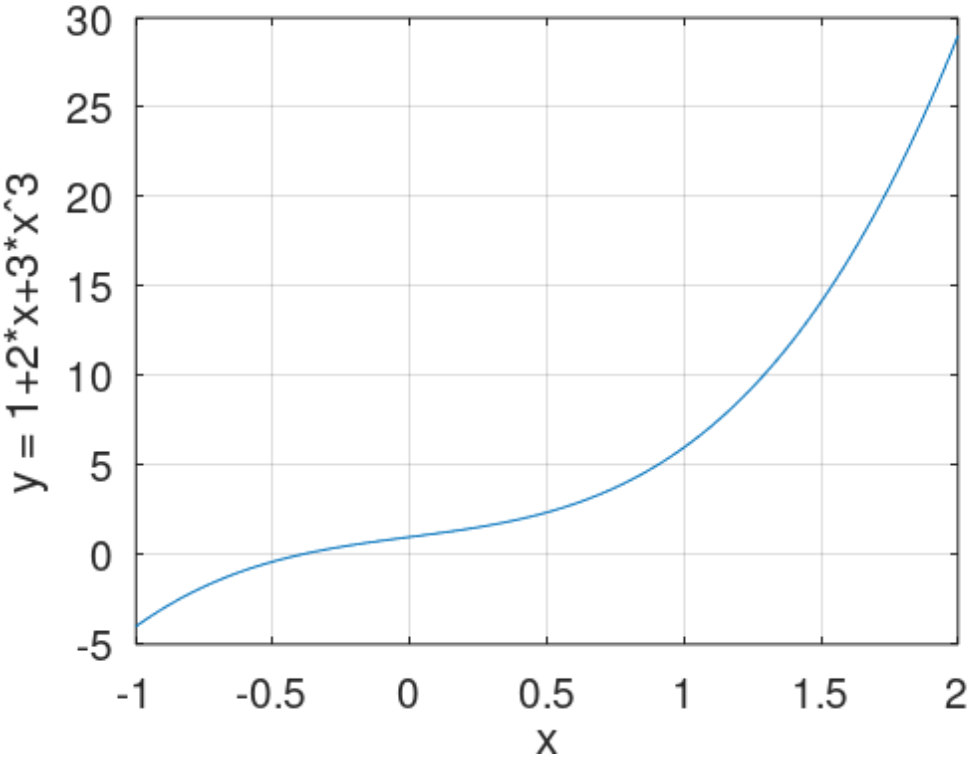
Функция `roots` для нахождения корней полинома, заданного вектором значений коэффициентов вызывается в форме:

`roots (c)`, где `c` — вектор коэффициентов полинома:

$$c_1 x^{N-1} + \cdots + c_{N-1} x + c_N .$$

Рассмотрим пример.

```
octave:3> echo on
octave:4> Paroot
+ # поиск корней уравнения p(z) = 1 + 2*z + 3*z^3
+
+ roots([3 0 2 1])
ans =


   0.2012 + 0.8877i
   0.2012 - 0.8877i
  -0.4023 +      0i


+
+ x = -1:0.01:2;  # x меняется от 1 до 2
+ y = polyval([3 0 2 1],x); # значение полинома в этих точках
+
+ plot(x,y);
+ xlabel("x"); ylabel("y = 1+2*x+3*x^3");
+ grid on
+ print -svgconvert root.pdf;print -svgconvert root.png
+
octave:5>
```



На графике видно наличие единственного действительного корня — при z = -0.4023.

Методы интерполяции

Фувнкция `interp1` реализует несколько методов одномерной интерполяции и может быть вызвана в формах:

```
yi = interp1 (x, y, xi)
yi = interp1 (y, xi)
yi = interp1 ( . . . , method)
yi = interp1 ( . . . , extrap)
yi = interp1 ( . . . , "left")
yi = interp1 ( . . . , "right")
pp = interp1 ( . . . , "pp")
```

Назначение параметров.

- `xi` — вектор значений аргумента для которых вычисляются значения интерполяционного полинома.

- `yi` — выходной параметр — вектор или матрица значений интерполяционного полинома для значений аргумента в точках `xi`. Если не указан входной параметр `x`, то вместо него используется массив индексов вектора `y`.
- `x` — вектор, задающие значение абсцисс для котрых заданы значения ординат кривой, подлежащей интерполяции.
- `y` — вектор, матрицы или n-мерный массив, задающие значения ординат кривой, подлежащей интерполяции. Если задан матрицей или n-мерным массивом, интерполяция выполняется для каждого столбца `y`.
- `"pp"` — строковый параметр. Если он задан, то входной параметр `xi` задавать не следует т. к. при этом функция не будет получать значения `yi`, а построит кусочно-полиномиальный объект с именем `pp`, позволяющий вычислять значения интерполяционного полинома с помощью функции `ppval`.
- `method` — текстовый параметр, задающий метод интерполяции, принимает одно из значений:
  - ✔ `"nearest"` — метод «ближайшего соседа»;
  - ✔ `"previous"` — «метод предыдущего соседа»;
  - ✔ `"next"` — «метод следующего соседа»;
  - ✔ `"linear"` — линейная интерполяция (по умолчанию);
  - ✔ `"pchip"` — кусочно—кубическая полиномиальная интерполяция с гладкой первой производной.
  - ✔ `"cubic"` — синоним `"pchip"`;
  - ✔ `"spline"` - кубический сплайн с гладкими первой и второй производными по всей кривой.

Мы не будем рассматривать другие свойства этой функции, а также других функций интерполяции, представленный в [Док, с. 813 — 822].

Рассмотрим пример.

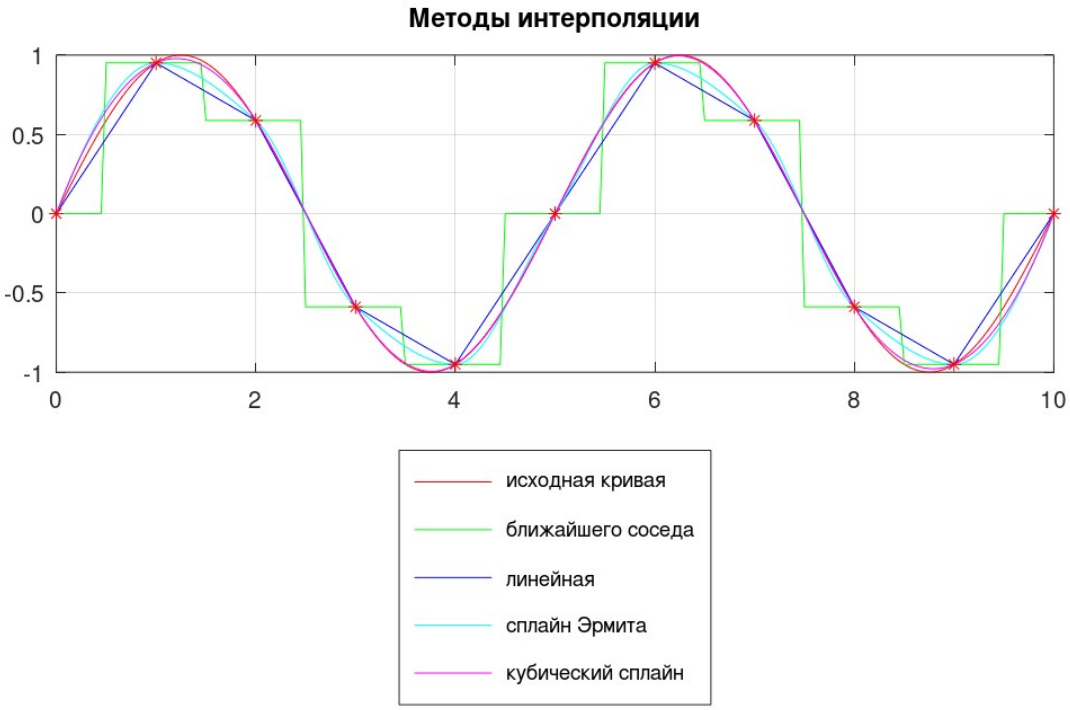
```
octave:5> Paitrp
```

```
+
+ # Методы интерполяции
+
+ # аргументы и значения функции
+
+ xf = [0:0.05:10];
+
+
+ newwhos
+  whos_line_format ("%la:5; %ln:6; %cs:14:6:1; %lt:6; %rb:12;
%lc:-1;\n");
+
+
+ yf = sin (2*pi*xf/5);
+
+ # координаты точек интерполяции
+
+   xp = [0:10];
+   yp = sin (2*pi*xp/5);
+
+ # знвчения интерполяционных полиномов
+
+   lin = interp1 (xp, yp, xf);           # линейная
+   near = interp1 (xp, yp, xf, "nearest"); # ближ. соседа
+   pch = interp1 (xp, yp, xf, "pchip");    # сплайн Эрмита
+   spl = interp1 (xp, yp, xf, "spline");   # куб. сплайн
+
+ # график
+
+   f = figure('position', [10 10 1200 500]);
+
+   plot (xf,yf,"r", xf,near,"g", xf,lin,"b", xf,pch,"c",...
```

```
+         xf,spl,"m",xp,yp,"r*");
+
+   legend ("исходная кривая", "ближайшего соседа",...
+    "линейная", "сплайн Эрмита", "кубический сплайн");
+   legend("location", "southoutside")
+
+   title   ("Методы интерполяции");
+   grid on
+
+   print -svgconvert int.pdf;print -svgconvert int.png
+
+   whos
Variables visible from the current scope:

variables in scope: top scope

Attr   Name        Size              Type               Bytes Class
====   ====        ====              ====               ===== =====
c      ans         3x1               complex matrix        48 double
       f           1x1               scalar                 8 double
       lin         1x201             matrix              1608 double
       near        1x201             matrix              1608 double
       pch         1x201             matrix              1608 double
       spl         1x201             matrix              1608 double
       x           1x301             double_range          24 double
       xf          1x201             matrix              1608 double
       xp          1x11              matrix                88 double
       y           1x301             matrix              2408 double
       yf          1x201             matrix              1608 double
       yp          1x11              matrix                88 double

Total is 1834 elements using 12312 bytes
```

```
+
octave:6>
```

Построенные интерполяционные полиномы представлены на графике



Аппроксимация по методу наименьших квадратов

Полноценная поддержка в octave матричных типов данных, матричных операций и функций для решения систем линейных уравнений позволяет компактно реализовать алгоритм нахождения коэффициентов аппроксимирующих полиномов.

Рассмотрим пример.

```
octave:10> Parfit
+ # Аппроксимация набора точек полиномом 2-й степени
+
+ # y = a + bx + cx^2
```

```
+
+ # задание точек
+
+ D = [1 1;2 2;3 5;4 4;5 2;6 -3];
+ xdata = D(:,1)
xdata =

   1
   2
   3
   4
   5
   6

+ ydata = D(:,2)
ydata =

   1
   2
   5
   4
   2
  -3

+
+ # Построение матрицы смстемы превичных уравнений
+
+ A = ones(6,3);
+ A(:,2) = xdata; A(:,3) = xdata.^2;
+ A
A =
```

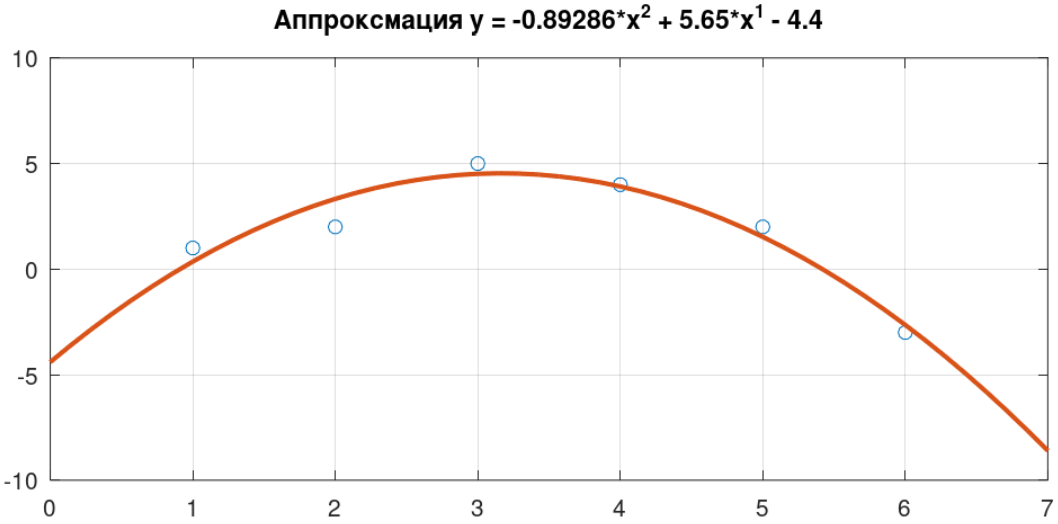
```
     1     1     1
     1     2     4
     1     3     9
     1     4    16
     1     5    25
     1     6    36

+
+ # Обозначим B = [a b c] вектор искомых коэффициентов
+
+ # Тогда они могут быть найдены по методу наименьших квадратов
+ # как решение системы нормальных уравнений:
+
+ #     A'AB = A'y
+
+ # Построим эту систему
+
+ D = A'*A      # ее левая часть
D =

       6     21     91
      21     91    441
      91    441   2275

+ A'*ydata      # ее правая часть
ans =

    11
    28
    60

+
```

```
+ # Соответствующая матрица расширенной системы
+
+ D(:,4) = A'*ydata
D =


      6     21     91     11
     21     91    441     28
     91    441   2275     60


+
+ C=rref(D) # Решаем ее методом исключения Гаусса с получением
# C - матрицы приведённого ступенчатого вида по строкам
C =


   1.0000         0         0   -4.4000
        0    1.0000         0    5.6500
        0         0    1.0000   -0.8929
+
+
+ # Искомые коэффициенты полинома - c, b и a
+
+ B = C(:,4)
B =


  -4.4000
   5.6500
  -0.8929


+
+ # Можно было проще:  B = rref(D)(:,4)
+
+ # Меняем порядок элементов B на обратный, проверяем
```

```
+ # символьный вид полинома
+
+ polyout(flipud(B),"x")
-0.89286*x^2 + 5.65*x^1 - 4.4
+
+ # задаем значения x
+
+ x = linspace (0,7,50);
+
+ # Вычисляем значения полинома
+
+ y = polyval (flipud(B),x)
y =

 Columns 1 through 6:

  -4.400000  -3.611079  -2.858601  -2.142566  -1.462974  -0.819825

 Columns 7 through 12:

  -0.213120   0.357143   0.890962   1.388338   1.849271   2.273761

 Columns 13 through 18:

   2.661808   3.013411   3.328571   3.607289   3.849563   4.055394

 Columns 19 through 24:

   4.224781   4.357726   4.454227   4.514286   4.537901   4.525073

 Columns 25 through 30:

   4.475802   4.390087   4.267930   4.109329   3.914286   3.682799

 Columns 31 through 36:
```

```
   3.414869   3.110496   2.769679   2.392420   1.978717   1.528571

 Columns 37 through 42:

   1.041983   0.518950  -0.040525  -0.636443  -1.268805  -1.937609

 Columns 43 through 48:

  -2.642857  -3.384548  -4.162682  -4.977259  -5.828280  -6.715743

 Columns 49 and 50:

  -7.639650  -8.600000

+
+ # Строим график результата
+
+ f = figure('position', [10 10 1200 500]);
+
+ plot (xdata, ydata,"o", x, y, 'linewidth',2);
+ grid on
+
+ # В квадратных скобках конкатенация двух строк, разделенных
+ # символом пробел
+
+ title (["Аппроксммация y = " polyout(flipud(B),"x")])
+
+ print -svgconvert lin.pdf;print -svgconvert lin.png
+
```

Построенный график




```
+ %{
+  В octave есть функция polyfit, которая автоматизирует
+ получение коэффициэнтов приближение полином по координатам
+ точек и его степени
+
+ %}
+
+ B = polyfit(xdata,ydata,2)
B =

  -0.8929   5.6500  -4.4000

+
+ # Порядок коэффициэнтов правильный:
+
+ polyout(B,"x")
-0.89286*x^2 + 5.65*x^1 - 4.4
+
+ # Ординаты точек полинома:
+
+ y = polyval (B,xdata)
y =
```

```
   0.3571
   3.3286
   4.5143
   3.9143
   1.5286
  -2.6429


+
+ f = figure('position', [10 10 1200 500]);
+
+ plot (xdata,ydata,"o-", xdata,y,'+-')
+ grid on
+ legend ("Исходные данные", "Полином")
+ title (["Аппроксммация y = " polyout(B,"x")])
+
+ print -svgconvert pfit.pdf;print -svgconvert pfit.png
+
octave:11>
```

График после работы функции `polyfit`.