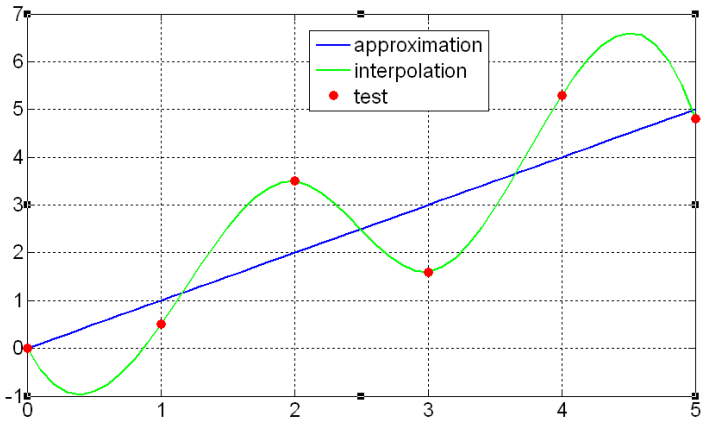
Некоторые задачи вычислительной математики

Аппроксимация и интерполяция

Экспериментальные данные, сведенные в таблицы и графики, проигрывают в наглядности аналитическим решениям. Подобные проблемы возникают и в случае, если данные получены расчетным путем, но существуют для ограниченного числа точек. Причиной тому может быть сложность и трудоемкость расчетов. Можно выделить два подхода к решению данной задачи.

Можно построить кривую, не требуя прохождения ее через все имеющиеся точки, в каком-то смысле соответствующей этим точкам. Эта задача аппроксимации. На рисунке ей соответствует синяя линия. Красные маркеры – результаты измерений.

Если требовать, чтобы построенная кривая точно проходила через точки, как зеленая линия на рисунке, то это задача интерполяции.



MatLab

Аппроксимация

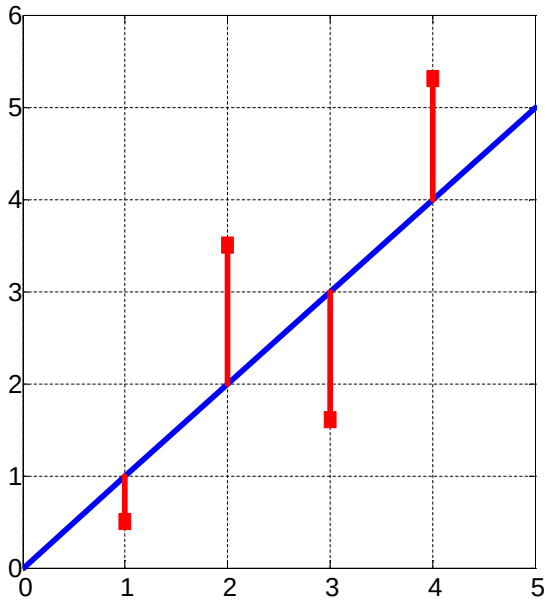
Аппроксимация (приближение) – научный метод, состоящий в замене объектов более простыми, но близкими к исходным.

При интегрировании сложная функция аппроксимируется (заменяется) набором прямоугольников. Аппроксимация функции *sin(x)~x*.

Пусть имеются экспериментальные данные, заданные таблично. Аппроксимация заключается в нахождении гладкой кривой, которая, не обязательно проходя через все узловые точки, наилучшим образом соответствует им. Для экспериментальных данных, полученных в результате измерений, актуально сглаживание случайных ошибок.

Метод наименьших квадратов

Метод наименьших квадратов (МНК) – аппроксимирующую кривую следует провести так, чтобы сумма квадратов её отклонений от табличных значений по всем узловым точкам была минимальна. На рисунке квадратными маркерами показаны результаты измерений, которые аппроксимирует линейная функция.



Matlab

Пусть результаты измерений представлены N парами значений (x_i, y_i). Найдем линейную функцию, аппроксимирующую эти точки в соответствии с методом МНК. Запишем эту функцию в виде

$$y = a_1 + a_2 x.$$

Задача заключается в определении коэффициентов a_1, a_2.

Рассчитаем сумму квадратов отклонений аппроксимирующей функции от табличных значений

$$F = \sum_{i=1}^{N} (y - y_i)^2 = \sum_{i=1}^{N} (a_1 + a_2 x_i - y_i)^2 .$$

Полученная форма есть функция двух неизвестных a_1, a_2. Минимум ее определим из условия

$$\frac{\partial F}{\partial a_1} = \frac{\partial F}{\partial a_2} = 0 .$$

Получающуюся в результате дифференцирования линейную систему алгебраических уравнений запишем в матричной форме

$$\begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

где $s_{11} = N$, $s_{12} = s_{21} = \sum_{i=1}^{N} x_i$, $s_{22} = \sum_{i=1}^{N} x_i^2$, $b_1 = \sum_{i=1}^{N} y_i$, $b_2 = \sum_{i=1}^{N} x_i y_i$.

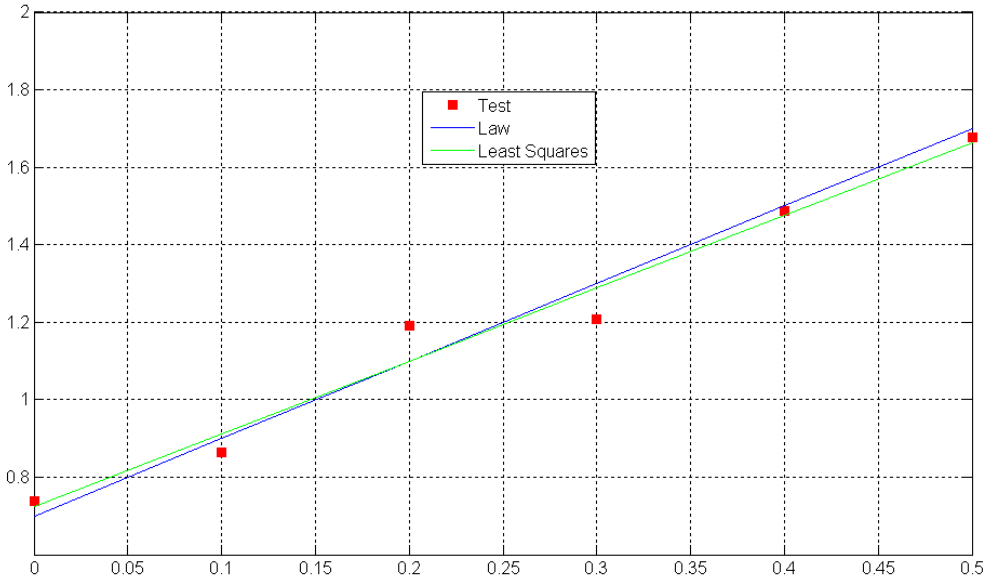
Решив систему, получим искомое – коэффициенты a_1, a_2 аппроксимирующей прямой.

Данный алгоритм, реализованный средствами MatLab, приведен ниже. В качестве истинной выбирается прямая с коэффициентами a_1=*0.7* и a_2=*2*. Экспериментальные данные получаем, искажая истинную функцию генератором случайных чисел *rand*.

```
% метод наименьших квадратов
% Ordinary Least Squares
N=6; % количество измерений
x = 0:0.1:0.5; % вектор-строка абсцисс измерений
x=x';          % вектор-столбец абсцисс измерений
y1=0.7+2*x;    % вектор-столбец ординат истиной прямой
```

```
y=0.7+2*x+(rand(N,1)-0.5)/5; % зашумленные ординаты
% рассчитываем матрицу системы линейных алгебраических уравнений
s(1,1) = N;
s(2,1) = sum(x);
s(1,2) =s(2,1);
s(2,2) = sum(x.^2);
% вектор-столбец правой части системы линейных алгебраических уравнений
b=[sum(y);sum(y'*x)];
a=s\b; % решаем систему
y2=a(1)+a(2)*x; % аппроксимирующая прямая
% строим график
hL=plot (x,y ,x,y1,x,y2); grid;
set(hL(1), 'LineStyle', 'none','Marker', 's', 'MarkerSize', 7, 'Color', 'r',
'MarkerFaceColor', 'r' );
set(hL(2), 'LineWidth', 1, 'Color', 'b');
set(hL(3), 'LineWidth', 1, 'Color', 'g');
set(gca,'fontsize',12)
legend('Test', 'Law', 'Least Squares',-1)
```

На рисунке приведены: истинная функция, аппроксимирующая прямая и результаты измерений.



MatLab имеет встроенные функции, позволяющие решить данную задачу. Функция *p=polyfit(x,y,n)* рассчитывает коэффициенты полинома *p(x)* степени *n*, аппроксимирующего функцию *y(x)* в смысле МНК. Результатом работы функции является строка *p* длины *n +1*, содержащая коэффициенты аппроксимирующего полинома.

Зная строку коэффициентов полинома *p*, с помощью функции *y = polyval(p, s)* можно вычислить значение этого полинома в точке *s*.

В приведенном примере аппроксимирующий полином первой степени, поэтому код будет следующий:

```
% встроенные функции
ap = polyfit(x,y,1); % вычисление коэффициентов полинома
y2 = polyval(ap,x); % расчет ординат аппроксимирующей прямой
```

Ниже приведены коэффициенты аппроксимирующего полинома первой степени, рассчитанные по приведенному алгоритму и встроенными функциями. В MatLab коэффициенты пронумерованы в обратном порядке.

```
a = 0.7617    1.8951
ap =1.8951    0.7617
```

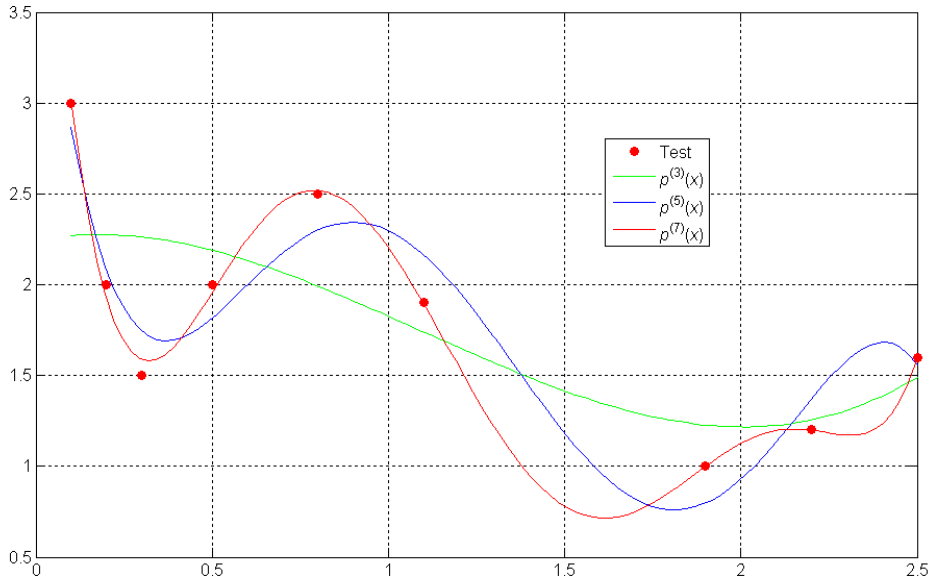
Полиномиальная аппроксимация

Линейная аппроксимация далеко не всегда применима. Хотя в некоторых случаях, зная физический закон, лежащий в основе исследуемой зависимости, надлежащим выбором переменных можно линеаризировать эту зависимость.

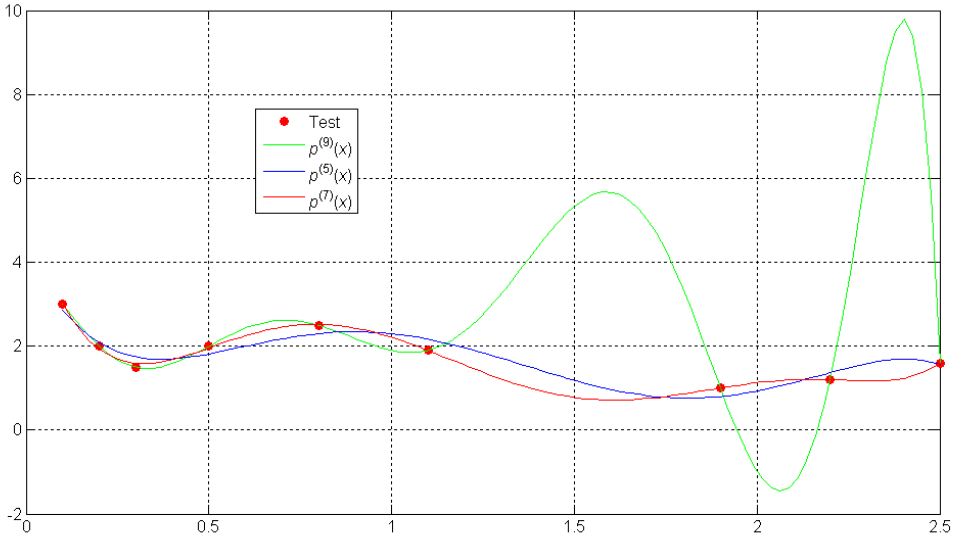
Аппроксимация в смысле МНК полиномами более высоких порядков, чем первый имеет широкую область применения.

Пример такой аппроксимации полиномами 3, 5 и 7 степени, использующий встроенные функции, описанные ранее, приведен ниже.

```
% полиномиальная аппроксимация (метод наименьших квадратов)
x = [0.1 0.2 0.3 0.5 0.8 1.1 1.9 2.2 2.5];
y = [3 2 1.5 2 2.5 1.9 1 1.2 1.6];
% вычисление коэффициентов разных порядков
p3 = polyfit(x, y, 3);
p5 = polyfit(x, y, 5);
p7 = polyfit(x, y, 7);
xx = linspace(x(1), x(end), 100);
% расчет ординат аппроксимирующих полиномов
yy3 = polyval(p3, xx);
yy5 = polyval(p5, xx);
yy7 = polyval(p7, xx);
% строим график
figure('Color',[1 1 1]);
hL=plot(x, y, xx, yy3, xx, yy5, xx, yy7); grid;
set(gca,'fontsize',12)
legend('Test', '{\itp}^{(3)}({\itx})', '{\itp}^{(5)}({\itx})',
'{\itp}^{(7)}({\itx})',-1)
set(hL(1), 'LineStyle', 'none','Marker', 'o', 'MarkerSize', 7, 'Color', 'r',
'MarkerFaceColor', 'r' );
set(hL(2), 'LineWidth', 1, 'Color', 'g');
set(hL(3), 'LineWidth', 1, 'Color', 'b');
set(hL(4), 'LineWidth', 1, 'Color', 'r');
```



Результаты работы алгоритма приведены на рисунке. Повышение степени полинома улучшает качество аппроксимации. Возникает желание применять полином еще более высокого порядка. Однако повышение степени полинома приводит к обратному результату. На следующем рисунке полином третьей степени заменен полиномом девятой степени.



Удачной такую аппроксимацию назвать нельзя. На некоторых интервалах между точками полином начинает осциллировать, что объясняется погрешностью расчета коэффициентов при высоких степенях.

Интерполяция

Интерполяция – нахождения промежуточных значений по имеющемуся дискретному набору точных числовых значений.

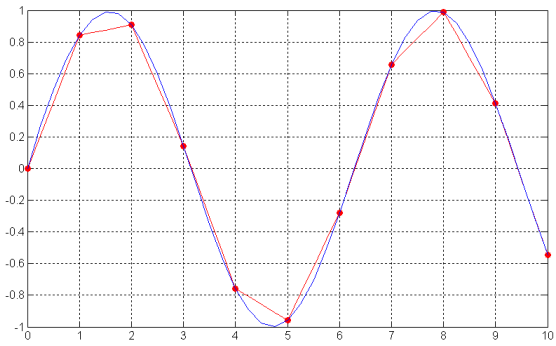
Постановка задачи интерполяции:

На отрезке $[a,b]$ в точках $\{x_1,x_2,...,x_n\}$ известны значения функции $f(x)$. Требуется построить функцию $g(x)$, совпадающую с заданной функцией $f(x)$ в этих точках.

$$g(x_k)=f(x_k),\ k=1,2,...,n,$$

Точки $\{x_1,x_2,...,x_n\}$ называются узлами интерполяции, а построение функции – интерполированием.

Линейная интерполяция – узлы соединяются прямолинейными отрезками. Два соседних узла позволяют рассчитать параметры этих отрезков. Построенная таким образом функция $g(x)$ – ломаная с вершинами в узлах. Точность интерполяции в промежуточных точках невысока. Представляют интерес гладкие функции, имеющие непрерывные производные.



Интерполяция полиномом высокой степени на всем отрезке называется глобальной. Теоретически точность интерполяции возрастает с ростом степени полинома, однако на практике погрешность расчета коэффициентов при высоких степенях приводят к осцилляциям интерполяционной кривой.

Идея локальной интерполяции заключается в том, что между соседними узлами строится свой отдельный интерполяционный полином невысокой степени. Затем эти полиномы «сшиваются». Такой метод называется сплайн- интерполяцией. Если использовать полином третьего порядка, то в узлах можно удовлетворить как условию непрерывности функции, так и непрерывности её первой и второй производных.

Сплайн – гибкая линейка, рейка, зафиксированная в отдельных точках. Рейка – физическая модель сплайн-функции, сплайн-функция – математическая модель рейки.

Функция Matlab *interp1* строит интерполирующую кривую для одномерного массива. Синтаксис функции

yi = interp1 (x,y,xi, metod),

x – массив узлов (абсцисс известных значений),
y – массив значений функции (ординат известных значений),
metod – определяет метод построения сплайна. Принимает следующие значения:

1. **‘nearest’** – интерполяция по соседним точкам – этот метод построения кусочной функции, при котором значение в любой точке равно значению в ближайшей узловой точке – интерполяция полиномами 0-ой степени;
2. **‘linear’** – линейная сплайн-интерполяция — интерполяция полиномами 1-ой степени (применяется по умолчанию, если способ интерполирования не задан);
3. **‘cubic’** – интерполяция кубическим полиномом;
4. **‘spline’** – интерполяция кубическим сплайном;
5. **‘pchip’** — интерполяция кубическим эрмитовым сплайном.

Приводимый ниже пример кода демонстрирует использование линейной и сплайн-интерполяции.

```
% сплайн интерполяция
x = 0:10; y = sin(x);
xx = 0:.25:10;
yy = interp1(x, y, xx, 'linear');
yy1 = interp1(x, y, xx, 'spline');
% строим график
figure('Color',[1 1 1]);
hL=plot(x, y, xx, yy, xx, yy1); grid;
set(gca,'fontsize',12)
set(hL(1), 'LineStyle', 'none','Marker', 'o', 'MarkerSize', 7, 'Color', 'r',
'MarkerFaceColor', 'r' );
set(hL(2), 'LineWidth', 1, 'Color', 'r');
set(hL(3), 'LineWidth', 1, 'Color', 'b');
```

Так же построение сплайн-интерполяции возможно и с помощью функции *yy=spline(x, y, xx),* более простой в реализации.

Вычисление корней функций одной переменной

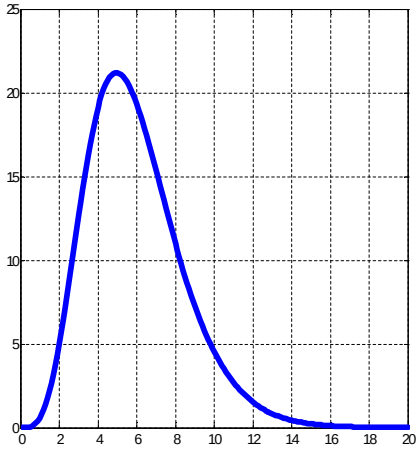
В задачах физики нередко возникает необходимость решения уравнений с одной переменной. Класс точных решений для нелинейных уравнений весьма ограничен. Использование численных методов позволяет добиться искомого результата.

Рассмотрим эту проблему на примере вывода из формулы Планка для излучения черного тела закона смещения Вина.

Планк предположил, что классические представления о том, что энергия осциллятора изменяется непрерывным образом, неприменимы к процессам излучения – поглощения. Согласно гипотезе Планка энергия гармонического осциллятора принимает не произвольные значения, а образует дискретный ряд 0, ε_0, $2\varepsilon_0$, $3\varepsilon_0$, где ε_0– минимальный квант энергии, определяемый характеристиками осциллятора. Основываясь на этой гипотезе, Планк получил выражение для спектральной плотности энергетической светимости абсолютно черного тела. Приведем формулу Планка в переменных λ, T

$$r_{\lambda,T} = D(T)\cdot r(x),\ \text{где}\ D(T) = 2\pi\frac{k^5T^5}{h^4c^3},\ x = \frac{hc}{\lambda kT},\ \text{а}\ r(x) = \frac{x^5}{\exp(x)-1}.$$

Приведя к безразмерному виду, нарисуем эту функцию



Matlab

Закон смещения Вина устанавливает, что длина волны, соответствующая спектральной плотности энергетической светимости абсолютно черного тела, обратно пропорциональна температуре. Покажем, что этот закон можно вывести, основываясь на формуле Планка. Из приведенного графика следует, что функция достигает максимума при значении, $x_0 \approx 5$, но тогда

$$\lambda_{\max} T = \frac{hc}{x_0 k} = b .$$

Полученный результат есть ничто иное как математическое выражение закона Вина, а если точно рассчитать входящую в него константу, то можно убедиться в согласии этого выражения с экспериментом.

Для определения максимума функции $r(x)$, приравняем первую производную нулю. Проделав это, получаем трансцендентное уравнение

$$5\left(\exp(x)-1\right) = x \cdot \exp(x) .$$

Уравнение, имеющее вид $g(x) = u(x)$, преобразуем к эквивалентной форме

$$f(x) = g(x) - h(x) = 0 .$$

Рассмотрим методы численного решения нелинейных уравнений вида $f(x) = 0$. Нахождение корней уравнения осуществляется в два этапа. Первоначально следует отделить корни – разбить всю область допустимых значений на отрезки, в каждом из которых содержится один корень. Отрезок, содержащий только один корень x_0, называется отрезком локализации корня x_0. Затем производится расчет корня до заданной точности.

Метод деления отрезка пополам (метод бисекции)

Пусть функция *f(x)* непрерывна на интервале *[a, b]*. Метод решения уравнения *f(x)=0* применим если на границах интервала *[a, b]* функция *f(x)* имеет разные знаки. Как проверить? Если

$$f(a) \cdot f(b) < 0,$$

то функция *f(x)* имеет разные знаки, а внутри интервала имеется хотя бы один корень.

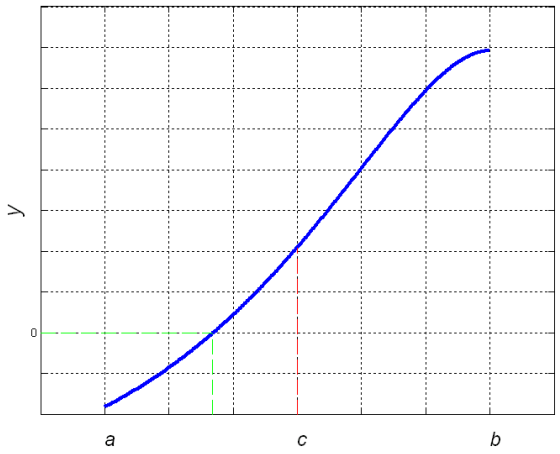
Поделим интервал пополам. Тогда координаты находящейся в середине интервала точки *c=(a+b)/2*, а значение функции в ней *f(c)*.

Если $f(a)\cdot f(c)>0$, то функция на интервале $[a,c]$ не меняет знак. Следовательно, корень уравнения находится на интервале $[c, b]$. Интервал $[a, c]$ можно исключить из дальнейшего анализа. Точку a перенести в точку c. Тогда

$$a=c,\ f(a)=f(c).$$

Если $f(a)\cdot f(c)<0$, то функция на интервале $[a,c]$ меняет знак, корень находится на этом интервале. Интервал $[c, b]$ можно исключить из дальнейшего анализа. Точку b перенести в точку c. Тогда

$$b=c,\ f(b)=f(c).$$



Matlab

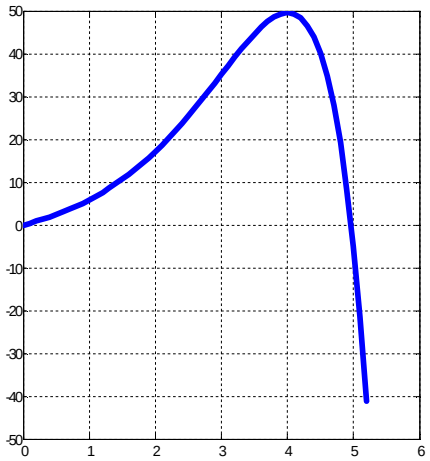
Именно этот случай представлен на рисунке. Исключив половину интервала, продолжим деление пока длина оставшегося интервала $[a, b]$ не станет меньше некоторой наперед заданной малой величины ε

$$|\ b-a\ |<\varepsilon$$

В качестве корня принимают середину отрезка $(a+b)/2$.

Расчет экстремума формулы Планка

В рассматриваемом примере $f(x)=5\left(\exp(x)-1\right)-x\cdot\exp(x)$, график которой приведен ниже.



Matlab

Ниже реализован алгоритм бисекций, а также проведено сравнение со встроенной функцией *fzero* решающую ту же задачу. Показаны два способа передачи переменной в функцию *fzero*.

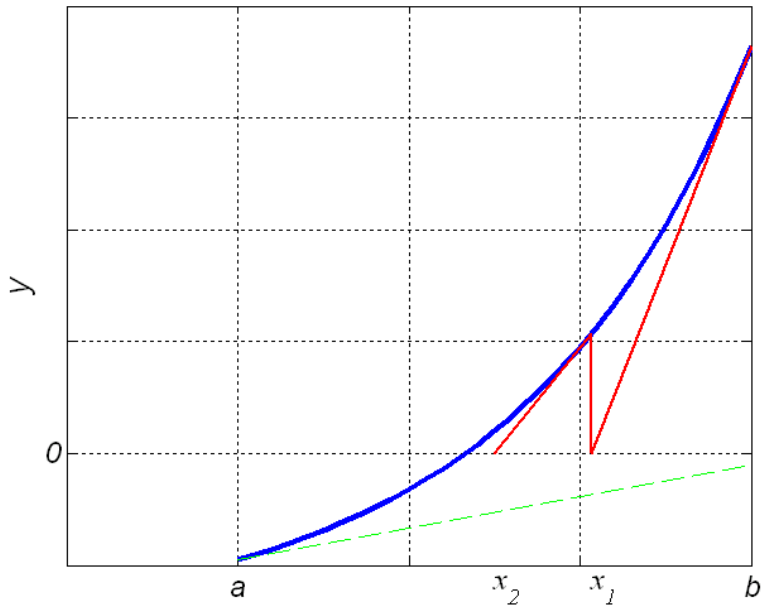
```
% нахождение корней уравнения f(x)=0 методом бисекции
f=@(x) 5*(exp(x)-1)-x.*exp(x);  %это функция из закона смещения Вина
a=1; b=6; % зададим интервал
e=0.0001; % зададим точность решения
% Основной цикл
while abs(b-a)>e
    c=(b+a)/2;
    if sign(f(c))==sign(f(a))
        a=c;
    else
        b=c;
    end
end
disp(['Корень x1=' num2str(c,8)]);
% нахождение корней встроенной функцией (два способа)
disp(['Корень x2=' num2str(fzero (f,5),8)]);
disp(['Корень x3=' num2str(fzero ('5*(exp(x)-1)-x*exp(x)', [1 6]),8)]);

Корень x1=4.9650726
Корень x2=4.9651142
Корень x3=4.9651142
```

Если при задании интервала положить $a=0$, то правильное решение построенным алгоритмом в отличие от встроенной функции не достигается. Этот пример показывает важность правильного отделения корней.

Метод Ньютона (метод касательных)

Функция $f(x)$ непрерывна на интервале $[a, b]$. На границах интервала $[a, b]$ функция имеет разные знаки. Проведем в точке x_0 , принадлежащей интервалу $[a, b]$, касательную к кривой $y=f(x)$. Уравнение этой прямой:

$$y = f(x_0) + (x - x_0) f'(x_0).$$



Matlab

При построении рисунка полагалось, что $x_0=b$. Построенная касательная пересекает ось абсцисс в точке x_1, так, что $y(x_1) = 0$. Тогда

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \ .$$

Перенесем точку x_0 в x_1, тем самым уменьшив интервал, содержащий корень, и проделаем аналогичную процедуру. При этом для любой итерации справедливо

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

Корень найден с заданной точностью, если $| x_{n+1} - x_n | < \varepsilon$.

При неудачном выборе начального приближения, например как показано на рисунке, при $x_0=a$, следующее приближение – x_1 окажется вне отрезка $[a, b]$, на котором отделен корень (зеленая линия). В этом случае сходимость итерационного процесса не гарантируется.

Чтобы избежать такого развития событий, за начальную точку следует принять такую, в которой знаки функции и ее второй производной, совпадают,

$$f(x_0) \cdot f''(x_0) > 0 .$$

Для проведения расчетов необходимо вычислять производную функции, поэтому для функций, заданных таблично, метод не применим. Вместе с тем, сходимость метода касательных лучше, чем метода бисекций.