Интерполяционный полином Лагранжа в Excel

В рамках подготовки курса для бакалавров МФТИ я понял, что в моем блоге не так много заметок по использованию Excel в математике и физике. Каково же было мое удивление, когда я обнаружил, что книг по этой теме на русском языке буквально единицы. Ранее я опубликовал Вильям Дж. Орвис. Excel для ученых, инженеров и студентов. В заметке представлены три варианта нахождения интерполяционного полинома Лагранжа: таблица на листе Excel, функция VBA, функция листа Excel на основе REDUCE и LAMBDA. В заметке использованы материалы книги Алексея Васильева «Числовые расчеты в Excel». Бумажная и электронная версии книги доступны на сайте издательства.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Интерполяционный полином Лагранжа**													
2														
3			**Индекс узл. точки**	0	1	2	3	4	5	6	7	8	9	10
4			**Узловая точка**	-6,2832	-5,0265	-3,7699	-2,5133	-1,2566	0,0000	1,2566	2,5133	3,7699	5,0265	6,2832
5			**Значение функции**	0,0000	0,0362	0,0386	-0,0803	-0,3687	0,0000	0,3687	0,0803	-0,0386	-0,0362	0,0000
6	Аргумент	Функция	Полином Лагранжа											
7	-6,2832	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
8	-6,1575	0,0032	0,3308	0,7400	0,8222	-1,7527	3,0622	-3,9847	3,8058	-2,6340	1,2870	-0,4215	0,0831	-0,0075
9	-6,0319	0,0067	0,5022	0,5377	1,3442	-2,6884	4,6087	-5,9428	5,6456	-3,8935	1,8977	-0,6204	0,1222	-0,0110
10	-5,9062	0,0103	0,5606	0,3821	1,6376	-3,0343	5,0947	-6,5061	6,1461	-4,2232	2,0531	-0,6699	0,1318	-0,0118
11	-5,7805	0,0140	0,5429	0,2642	1,7610	-2,9717	4,8767	-6,1636	5,7884	-3,9623	1,9211	-0,6256	0,1229	-0,0110
12	-5,6549	0,0178	0,4781	0,1762	1,7620	-2,6430	4,2287	-5,2859	4,9335	-3,3638	1,6264	-0,5286	0,1036	-0,0093
13	-5,5292	0,0217	0,3880	0,1119	1,6783	-2,1578	3,3566	-4,1464	3,8449	-2,6107	1,2587	-0,4082	0,0799	-0,0071
14	-5,4035	0,0255	0,2891	0,0660	1,5399	-1,5991	2,4103	-2,9398	2,7073	-1,8304	0,8799	-0,2848	0,0557	-0,0050
15	-5,2779	0,0293	0,1928	0,0342	1,3698	-1,0274	1,4943	-1,7979	1,6438	-1,1064	0,5302	-0,1712	0,0334	-0,0030
16	-5,1522	0,0328	0,1070	0,0132	1,1856	-0,4850	0,6775	-0,8031	0,7287	-0,4882	0,2332	-0,0751	0,0146	-0,0013
17	-5,0265	0,0362	0,0362	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

Рис. 1. Вычисление интерполяционного полинома по методу Лагранжа; график табулируемой функции (штрихованная линия) и интерполяционного полинома (сплошная линия)

Алгоритм

Задача интерполирования обычно решается для того, чтобы «восстановить» по набору дискретных данных аналитическую функциональную зависимость. Нередко в качестве функции, на основе которой строится интерполяционная зависимость, выбирают полиномиальные выражения. Предположим, имеется набор узловых точек $x_1, x_2, \ldots, x_n$ и набор значений $y_1, y_2, \ldots, y_n$ неизвестной функции в этих точках. Задача – построить зависимость $f(x)$ такую, чтобы соответствующая кривая проходила через все точки (x_k, y_k) $(k = 1, 2, \ldots, n)$, т.е. необходимо, чтобы для всех k выполнялись соотношения $f(x_k) = y_k$.

Для однозначного определения коэффициентов интерполяционного полинома его степень должна быть на единицу меньше, чем количество точек, по которым выполняется интерполирование. Технически коэффициенты полинома можно вычислить несколькими методами. Наиболее популярны полином Лагранжа и полином Ньютона.

Схема Лагранжа предполагает, что соответствующее полиномиальное выражение, построенное по точкам (x_0, y_0), (x_1, y_1), ..., (x_n, y_n), ищется в виде

$$(1)\ L_n(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + \cdots + y_n\varphi_n(x) = \sum_{m=0}^{n} y_m\varphi_m(x)$$

Полином имеет степень n, поскольку строится по $n + 1$ точке: индексация точек (x_m, y_m) начинается с нуля, а последний индекс равен n. Функции $\phi_m(x)$ являются полиномами степени n, причем такими, что в узловых точках x_k имеют место соотношения: $\phi_m(x_k) = 0$, если $k \neq m$, и $\phi_m(x_k) = 1$, если $k = m$. Эти полиномы вычисляются в виде произведений

$$(2)\ \varphi_m(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{m-1})(x - x_{m+1})\ldots(x - x_n)}{(x_m - x_0)(x_m - x_1)\ldots(x_m - x_{m-1})(x_m - x_{m+1})\ldots(x_m - x_n)} = \prod_{\substack{k=0\\k\neq m}}^{n} \frac{(x - x_k)}{(x_m - x_k)}$$

для всех $m = 0, 1, 2, \ldots, n$.

Метод Лагранжа в таблице на листе Excel

Реализуем метод Лагранжа в Excel для табулированной по 11 равноудаленным узловым точкам (на интервале значений аргумента от -2π до 2 π) функции

$$(3)\ f(x) = \frac{sin(x)}{1 + x^2}$$

См. рис. 1. В столбце А – аргумент, В – значение функции, С – значения интерполяционного полинома построенного по методу Лагранжа. Количество точек в столбце А зависит от целей интерполяции. Поскольку мы далее хотим построить график, точек выбрано много и они расположены равномерно на интервале интерполяции. Частота этих точек значительно выше, чем частота узловых точек, на основе которых создается полином.

Ячейки D4:N4 содержат значения узловых точек, на основе которых мы создаем интерполяционный полином. Значения полинома в узловых точках отображаются в ячейках D5:N5. В ячейках D3:N3 указаны индексы узловых точек.

Основные вычисления выполняются в ячейках D7:N107 и, как результат, в ячейках C7:C107 вычисляются значения интерполяционного полинома в точках, которые указаны в ячейках A7:A107. В ячейках D7:N107 содержатся значения функций $\phi_m(x)$ для разных аргументов интерполяционного полинома *x* и разных индексов узловых точек *m*. В каждой строке диапазона D7:N107 находятся значения функций $\phi_m(x)$ для одного и того же аргумента *x*, но разных индексов *m*. В столбцах диапазона D7:N107 содержатся значения для разных аргументов *x*, и одного и того же индекса *m*. В диапазоне D7:N107 три типа формул: для левого D7: D107 и правого N7:N107 краев и остальных столбцов E7:M107. С формулами можно ознакомиться в приложенном Excel-файле.

Выделите диапазон A7:C107 и постройте график. Чтобы добавить узловые точки, скопируйте диапазон D4:N5 в буфер обмена, выделите диаграмму, пройдите *Главная –> Вставить –> Специальная вставка*. Настройте параметры в окне *Специальная вставка.* Нажмите Ok. Отформатируйте вставленный ряд: отмените линию и добавьте встроенный маркер.

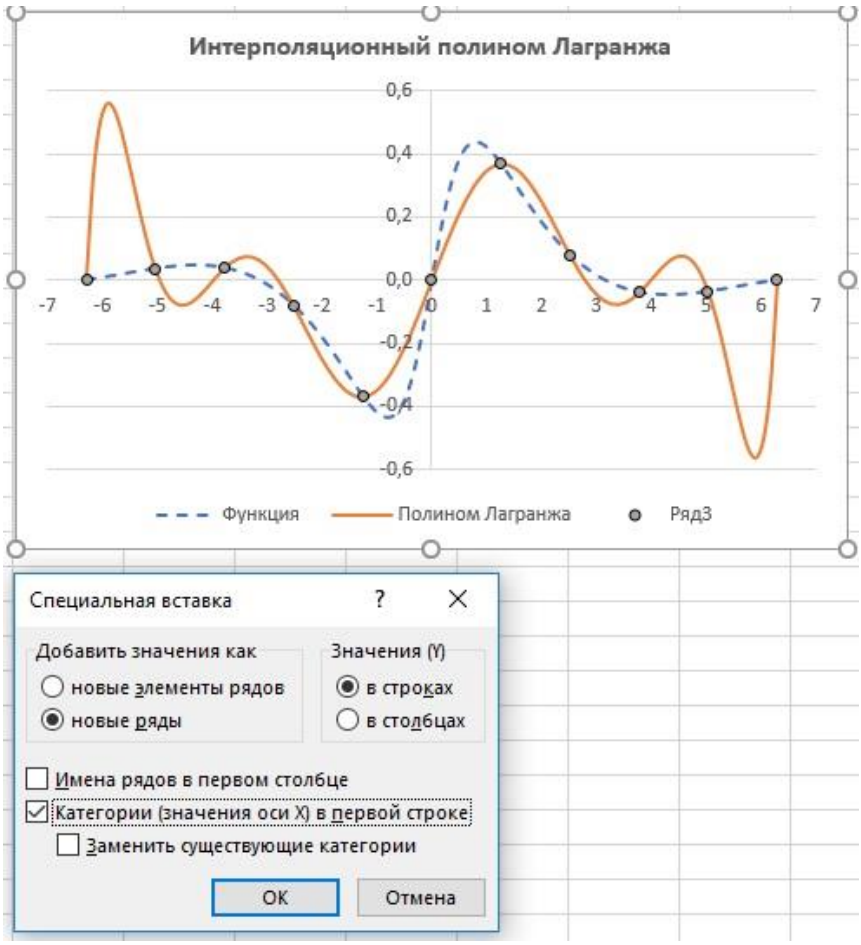

Рис. 2. График функции $f(x) = sin(x)/(1 + x^2)$, интерполяционный полином Лагранжа и узловые точки

Видно, что вычисленные нами значения интерполяционного полинома в узловых точках совпадают со значениями функции. В других точках совпадение не столь хорошее. К сожалению, значительные осцилляции на границах интервала интерполирования является типичными для этого метода.

Функция VBA

Можно упростить вычисления и отказаться от громоздкой промежуточной таблицы с вычислениями, если воспользоваться кодом VBA.

```
Function ЛАГРАНЖ(Px As Range, Py As Range, z As Variant) As Double
' Переменная для запоминания аргумента полинома
Dim x As Double
' Значение аргумента полинома
x = z
' Переменная для запоминания количества узловых точек
Dim n As Integer
' Вычисляем количество узловых точек
n = Px.Count
' Целочисленные переменные для операторов цикла
Dim i As Integer, j As Integer
' Переменная для вычисления значения полинома
Dim L As Double
' Начальное значение для полиномиальной суммы
L = 0
' Переменная для вычисления произведения
Dim phi As Double
' Внешний цикл
For i = 1 To n
' Начальное значение для произведения
phi = 1
' Первый внутренний цикл
For j = 1 To i - 1
' Умножаем на разность аргумента и узловой точки
phi = phi * (x - Px.Cells(j).Value)
' Делим на разность узловых точек
phi = phi / (Px.Cells(i).Value - Px.Cells(j).Value)
Next j
' Второй внутренний цикл
For j = i + 1 To n
' Умножаем на разность аргумента и узловой точки
phi = phi * (x - Px.Cells(j).Value)
' Делим на разность узловых точек
phi = phi / (Px.Cells(i).Value - Px.Cells(j).Value)
Next j
' К полиномиальной сумме добавляем очередное слагаемое
L = L + phi * Py.Cells(i).Value
Next i
' Результат вычислений
ЛАГРАНЖ = L
End Function
```

Пояснение кода

Функция ЛАГРАНЖ() имеет три аргумента: диапазон ячеек со значениями узловых точек (*Px*), диапазон ячеек со значениями интерполируемой функции в узловых точках (*Py*), а также аргумент, для которого вычисляется значение интерполяционного полинома Лагранжа (*z*). Функция возвращает числовой результат типа Double.

Поскольку третий аргумент функции может быть как числовым значением, так и ссылкой на ячейку, в теле функции значение этого аргумента записываем в переменную *x*. Если третий аргумент – число, то процесс «переписывания» ничего не добавляет. Если же третий аргумент – ссылка на ячейку, то присваивание значения ячейки локальной числовой переменной позволяет снять неоднозначность в определении типа третьего аргумента.

Алгоритм использует количество узловых точек. Явно этот параметр в аргументе функции ЛАГРАНЖ() отсутствует. Но он вычисляется, как количество ячеек в диапазоне, переданном первым аргументом функции ЛАГРАНЖ(). Поэтому мы объявляем переменную *n = Px.Count*. Свойство Count для диапазона возвращает количество ячеек.

Основные вычисления производятся в блоке из вложенных условных операторов. Для использования в этих операторах объявляются две целочисленные переменные *i* и *j*. В переменной *L* накапливается полиномиальная сумма, а в переменной *phi* произведение в соответствии с формулой (2).

Перед началом выполнения вложенных операторов цикла переменной *L* присваивается значение 0. Индексная переменная *i* во внешнем цикле пробегает значения от 1 до *n*. В начале каждой итерации переменной *phi* присваивается значение 1. После этого последовательно запускаются два идентичных цикла. Основное различие между ними – диапазон изменения индексной переменной *j*. Для первого цикла она изменяется от 1 до *i-1*, а для второго цикла — от *i+1* до *n*. Таким образом, при фиксированном значении *i* переменная *j* пробегает все значения от 1 до *n*, за исключением значения *i*. За каждую такую итерацию переменная *phi* сначала умножается на величину (x-Px.Cells(j).Value), а затем делится на величину (Px.Cells(i).Value-Px.Cells(j).Value). Здесь следует учесть, что Px.Cells(индекс).Value – это значение ячейки с указанным индексом в диапазоне Px (т.е. это значение узловой точки).

После того как значение переменной *phi* (для данного значения *i*) вычислено, командой L = L+phi * Py.Cells(i).Value к полиномиальной сумме добавляем очередное слагаемое. Здесь Py.Cells(i).Value – ссылка на значение ячейки в диапазоне Py со значениями табулированной функции. В итоге значение переменной L возвращается как результат функции (команда ЛАГРАНЖ = L).

Работа функции ЛАГРАНЖ()

Функцию ЛАГРАНЖ() можно использовать на рабочем листе. Рассмотрим новый пример. Ячейки A4:B9 содержат данные об узловых точках и значениях функции *f(x) = x*exp(-x)*. При этом в ячейках A4:A9 указаны несколько неравномерно распределенных на интервале от 0 до 7 точек.

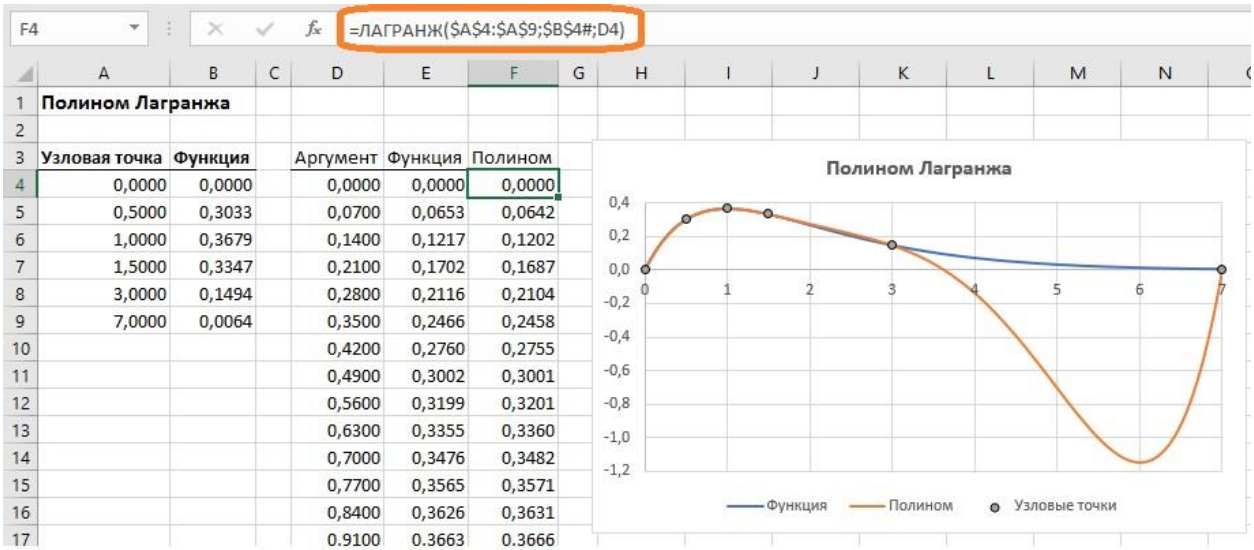

Рис. 3. Пользовательская функции для вычисления интерполяционного полинома Лагранжа

Как мы отмечали ранее, интерполяционный полином не везде дает хорошее приближение для табулированной функции, даже с учетом того, что последняя достаточно плавная.

Вычисление полинома Лагранжа на основе функций REDUCE и LAMBDA

В декабре 2020 года Microsoft анонсировал функцию LAMBDA, которая позволяет определять пользовательские функции, написанные на языке формул Excel. А в июле 2021 г. объявил о

создании новых функций, основанных на LAMBDA. Я недавно описал работу с LAMBDA и новыми функциями Excel. Не могу сказать, что написание сложных конструкций на основе этих функций проще, чем кода VBA, но как учебный пример, это весьма интересно.[1]

Код функции REDUCE

```
=REDUCE(
  0;
  $B$4#;
  LAMBDA(L;Py;
    L+REDUCE(
      1;
      $A$4:$A$9;
      LAMBDA(phi;Px;
        LET(
          Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
          Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
          x_i; ИНДЕКС($A$4:$A$9;Ind_y);
          ЕСЛИ(
            Ind_x = Ind_y;
            phi;
            phi*(D4#-Px)/(x_i-Px)
          )
        )
      )
    )*Py
  )
)
```

Описание работы функции REDUCE

Функция REDUCE работает, как обычная функция листа. Внутри ее могут располагаться ссылки на ячейки и динамические массивы. В отличие от функции LAMBDA, функция REDUCE не требует предварительного именования. Алгоритм работы функции REDUCE похож на алгоритм кода VBA в функции ЛАГРАНЖ(): один внешний цикл и один внутренний (в коде VBA два внутренних цикла). Внешний цикл перебирает все i значений диапазона B4:B9 (см. рис. 4), умножая на значения phi, определяемые для каждого i во внутреннем цикле. Во внутреннем цикле задается стартовое значение phi = 1. Далее для фиксированного i перебираются все j значений диапазона A4:A9, и для всех i ≠ j значение phi, полученное на предыдущем шаге, умножается на некое значение, а для i = j значение phi не изменяется. (Благодаря такой проверке, вместо двух внутренних циклов в коде VBA, здесь используется один.)

Посмотрим, как этот алгоритм реализован в коде функции REDUCE. Цель функции REDUCE – обработать массив, и вернуть одно число. Функции REDUCE имеет три аргумента. Первые два – начальное значение (0) и обрабатываемый массив ($B$4#).

```
=REDUCE(
  0;
  $B$4#;
```

Третий аргумент – функция обработки элементов массива

```
  LAMBDA(L;Py;
    L+REDUCE(
      1;
      $A$4:$A$9;
      LAMBDA(phi;Px;
        LET(
          Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
```

[1] Это оригинальный метод. Он не описан в книге Алексея Васильева «Числовые расчеты в Excel».

```
            Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
            x_i; ИНДЕКС($A$4:$A$9;Ind_y);
            ЕСЛИ(
              Ind_x = Ind_y;
              phi;
              phi*(D4#-Px)/(x_i-Px)
            )
          )
        )
      )*Py
  )
```

G4

```
=REDUCE(
  0;
  $B$4#;
  LAMBDA(L;Py;
    L+REDUCE(
      1;
      $A$4:$A$9;
      LAMBDA(phi;Px;
        LET(
          Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
          Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
          x_i; ИНДЕКС($A$4:$A$9;Ind_y);
          ЕСЛИ(
            Ind_x = Ind_y;
            phi;
            phi*(D4#-Px)/(x_i-Px)
          )
        )
      )
    )*Py
  )
)
```

	A	B	C	D	E	F	G	H
3	Узловая точка	Функция		Аргумент	Функция	Лагранж	REDUCE	
4	0,0000	0,0000		0,0000	0,0000	0,0000	0,0000	
5	0,5000	0,3033		0,0700	0,0653	0,0642	0,0642	
6	1,0000	0,3679		0,1400	0,1217	0,1202	0,1202	
7	1,5000	0,3347		0,2100	0,1702	0,1687	0,1687	
8	3,0000	0,1494		0,2800	0,2116	0,2104	0,2104	
9	7,0000	0,0064		0,3500	0,2466	0,2458	0,2458	
10				0,4200	0,2760	0,2755	0,2755	
11				0,4900	0,3002	0,3001	0,3001	
12				0,5600	0,3199	0,3201	0,3201	
13				0,6300	0,3355	0,3360	0,3360	

Рис. 4. Работа функции REDUCE

Функция LAMBDA принимает столько же параметров, сколько передает функция REDUCE:

LAMBDA(L;Py;

L – накопитель; Py – массив. L – это то значение, которое вернется функцией после обработки всех элементов массива. Начальное значение L = 0. Это значение определено первым аргументом функции REDUCE**.** Py – массив $B$4#. Функция LAMBDA накапливает значения L в элементе формулы L + REDUCE(… Эта запись аналогична более привычной для программистов L = L + REDUCE(…

Функция LAMBDA реализует внешний цикл. Она стартует со значения L = 0 и для всех элементов массива Py (он же $B$4#) выполняет действие

```
L+REDUCE(…
)*Py
```

Здесь реализована сумма произведений элементов Py и рассчитанных коэффициентов, которые возвращаются внутренним циклом на основе REDUCE(…

Внутренний цикл также основан на REDUCE

```
REDUCE(
  1;
  $A$4:$A$9;
  LAMBDA(phi;Px;
    LET(
      Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
      Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
      x_i; ИНДЕКС($A$4:$A$9;Ind_y);
      ЕСЛИ(
        Ind_x = Ind_y;
        phi;
        phi*(D4#-Px)/(x_i-Px)
      )
    )
  )
)
```

Результат REDUCE – коэффициент для умножения на элемент массива Py. Функция REDUCE принимает два аргумента

```
REDUCE(
  1;
  $A$4:$A$9;
```

Начальное значение 1 и массив $A$4:$A$9. Внутри функции массив $A$4:$A$9 понижается до одного значения, путем последовательной обработки всех элементов массива $A$4:$A$9. Обработка выполняется по правилам, описанным внутри LAMBDA:

```
  LAMBDA(phi;Px;
    LET(
      Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
      Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
      x_i; ИНДЕКС($A$4:$A$9;Ind_y);
      ЕСЛИ(
        Ind_x = Ind_y;
        phi;
        phi*(D4#-Px)/(x_i-Px)
      )
    )
  )
```

LAMBDA принимает два параметра от REDUCE: начальное значение phi = 1 и массив Px = $A$4:$A$9. Функция LET не выполняет расчеты, а позволяет упростить восприятие этого фрагмента формулы (подробнее см. здесь). LET определяет три переменные – индекс элемента массива $A$4:$A$9, обрабатываемого на текущем шаге:

```
      Ind_x; ПОИСКПОЗ(Px;$A$4:$A$9;0);
```

… индекс элемента массива $B$4#, переданного из внешнего цикла:

```
      Ind_y; ПОИСКПОЗ(Py;$B$4#;0);
```

… и значение элемента массива $A$4:$A$9, соответствующего индексу Ind_y из внешнего цикла:

```
x_i; ИНДЕКС($A$4:$A$9;Ind_y);
```

Расчет функции LET происходит внутри оператора ЕСЛИ:

```
ЕСЛИ(
   Ind_x = Ind_y;
   phi;
   phi*(D4#-Px)/(x_i-Px)
)
```

Для всех элементов массива, где Ind_x ≠ Ind_y накапливается новое значение phi:

```
   phi*(D4#-Px)/(x_i-Px)
```

Как обычно, эта запись эквивалента phi = phi*(…)

Если же индексы равны Ind_x = Ind_y, оставляем значение phi без изменения.

Ссылка на ячейку листа (D4#) – это ссылка на динамический массив аргументов, для которых вычисляется полином Лагранжа. На рис. 4 они размещены в столбце D.