УЧЕБНАЯ МАСТЕРСКАЯ

*Очков Валерий Федорович*

# ИСТОРИЯ ОДНОГО ШЕДЕВРА

## MATHCAD И НЕСТАНДАРТНАЯ ГРАФИКА

Слово «шедевр» в названии статьи, конечно, должно быть взято в кавычки. Ведь заголовок статьи – это не что иное, как название серии TV-передач, в которых рассказывается об истории создания известных полотен, хранящихся в Третьяковской галерее (http://www.tretyakov.ru), в Эрмитаже (www.hermitage.museum.ru), в Русском музее (www.rusmuseum.ru) и в других местах. Наш «шедевр» также хранится в галерее, но не в простой, а в виртуальной – в галерее необычной трехмерной графики, созданной с помощью **Mathcad**[1] (рисунок 1). Адрес галереи в Internet: http://www.mathsoft.com/mathcad/library/3Dplots. А история «шедевра» (он в правом нижнем углу на рисунке 1 и назван lace.mcd) такова.

Одна из рутинных задач, решаемых в среде Mathcad, это *поиск корней алгебраических уравнений*. К этой проблеме очень часто сводятся разного рода физические, химические, экономические и прочие «предметные» задачи. Mathcad имеет довольно мощный встроенный инструмент решения и верификации (проверки) подобных задач – линейных и нелинейных, безразмерных и включающих размерные величины.

На рисунке 2 приведен фрагмент результата решения (были найдены все 24 корня) с помощью средств символьной математики Mathcad системы двух нелинейных алгебраических уравнений вида:

$$x^2 \cdot y^2 - 7 \cdot x^3 \cdot y^3 - 7 = 0$$

$$(x^2+y^2)^2 - 7 \cdot (x^2-y^2) = 0$$

Из найденных корней четыре – действительные, остальные – мнимые (комплексные). С решением, показанным на рисунке 2, нам, можно сказать, повезло: стоит слегка усложнить одно из уравнений системы, как символьная математика Mathcad откажется решать задачу. Оператор **solve** (решить), задействованный в задаче на рисунке 2, – это своего рода *максималист*, возвращающий либо все решения системы, либо ни одного. Человеку же, сидящему за компьютером, как правило, нужен один действительный корень вблизи точки, указанной самим пользователем.

На рисунке 3 действительные корни нашей системы ищутся средствами уже не символьной, а *вычислительной* математики Mathcad: встроенная функция **Find** возвращает значение своих аргументов (**x** и **y**), превращающих уравнения системы (они записаны ниже ключевого слова **Given** и выше функции **Find**) в *тождества*. Вернее, делающих отклонение между левыми и правыми частями уравнений системы меньше (по модулю), чем значение встроенной переменной **TOL** (TOLerance, погрешность – по умолчанию она равна 0.001). Отсюда становится понятным, почему у термина «вычислительная математика» есть синоним – «математика *приближенных* вычислений». Тут подразумевается, что символьная (*аналитическая*) математика, задействованная в решении задачи на рисунке 2, – это математика абсолютно точных вычислений. Вот

поэтому-то, а также из-за того, что символьная математика пытается *всегда* выдать *все* решения, она бессильна перед более-менее сложной задачей. Но человеку, как правило, не нужны *абсолютно* все *абсолютно* точные ответы (рисунок 2). Он обычно удовлетворяется приближенными ответами по одиночным корням (рисунок 3).

Полное решение задачи в среде Mathcad должно включать не только символьное[2] (рисунок 2) и не только приближенное (рисунок 3), но и *графическое*

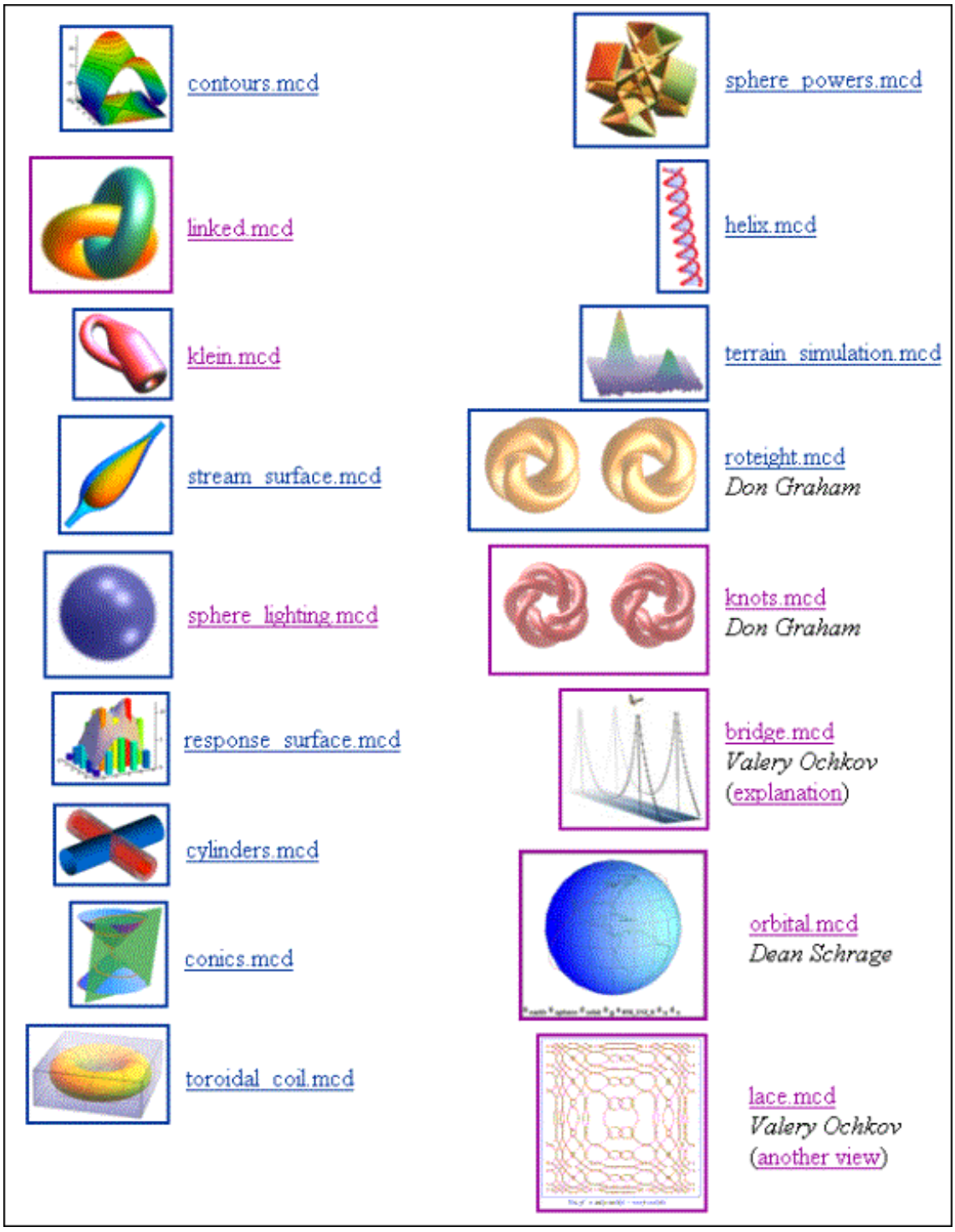


**Рисунок 1.** *Галерея необычной графики, созданной в среде Mathcad.*

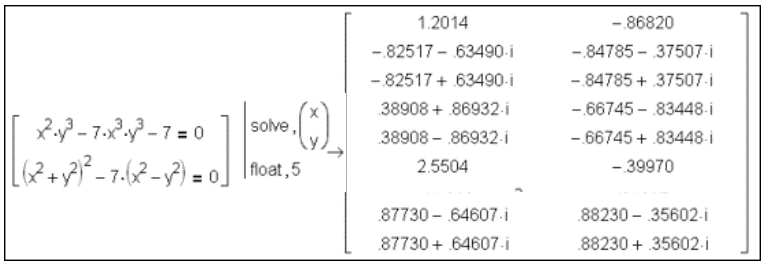


**Рисунок 2.** *Аналитическое решение системы двух алгебраических уравнений.*

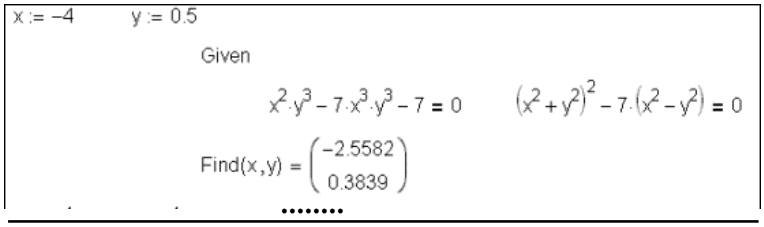


**Рисунок 3.** *Численное решение системы двух алгебраических уравнений.*

решение, которое не только визуализирует задачу, но и подтверждает правильность ее решения (верифицирует). Графическое решение нашей задачи – это вычерчивание на декартовом графике двух кривых, точки пересечения которых будут действительными корнями системы. Сложность тут лишь в том, что не всегда удается преобразовать систему (см. пример выше) к виду, готовому для построения декартовых графиков в среде Mathcad:

**y1(x) := f1(x)**

**y2(x) := f2(x)**

Здесь нужно средствами символьной математики попытаться решить уравнения **f1(x, y) = 0** и **f2(x, y) = 0** относительно **x**, что само по себе является довольно сложной задачей, которая очень часто просто не решается. А если исходное уравнение и решается, то и решений может быть много – $\mathbf{y1_1(x)}$, $\mathbf{y1_2(x)}$... $\mathbf{y1_N(x)}$... $\mathbf{y2_1(x)}$, $\mathbf{y2_2(x)}$... $\mathbf{y2_N(x)}$, каждое из которых требует отдельной кривой на графике. Получается некий *гордиев узел*, который мы развязывать не будем, а просто... разрубим его.

На рисунке 4 показана *универсальная* методика графического решения системы двух алгебраических уравнений: плоскость **x–y** *сканируется* (перебираются координаты **x** и **y** двумя циклами **for**) и запоминаются точки, где

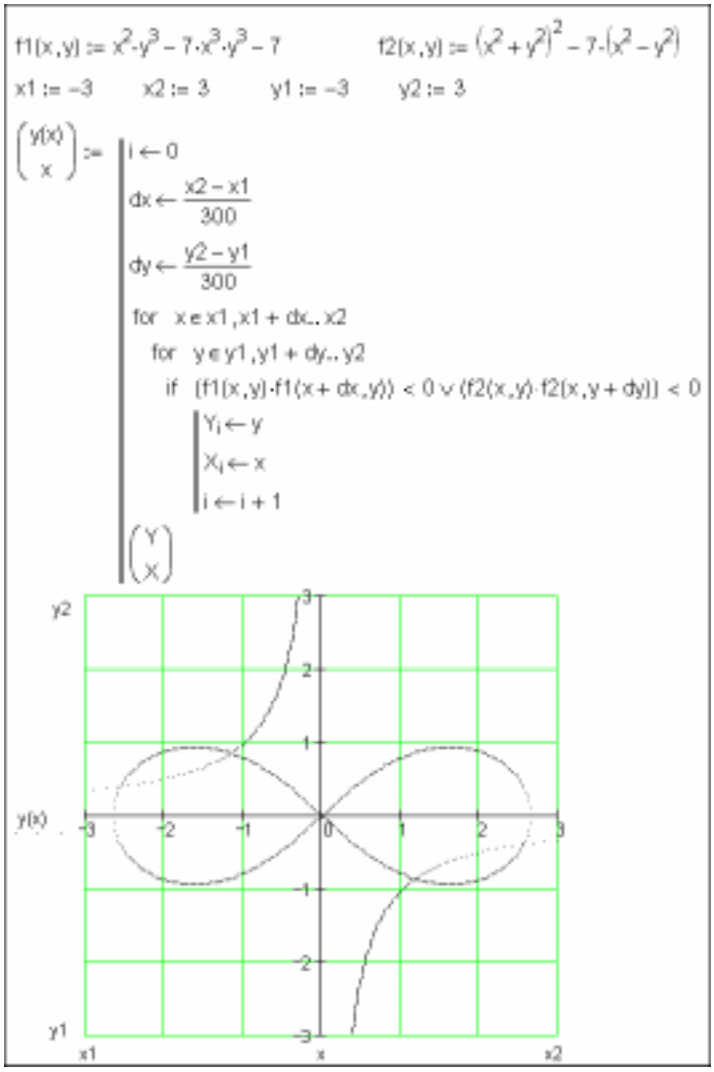


**Рисунок 4.** *Графическое решение системы двух алгебраических уравнений.*


**Рисунок 5.** *Графическое решение алгебраического уравнения.*

функция **f1** или (см. оператор ∨ – оператор логического сложения **ИЛИ**) функция **f2** близка к нулю. В момент прогонки нашей Mathcad-программы на рисунке 4 генерируется два вектора **y(x)** и **x** одной длины[3], элементы которых хранят графическое решение нашей задачи: на графике пересечение кривых (в центре знаменитая лемниската[4] Бернулли) – это корни нашей системы. Их можно уточнить, взглянув на рисунки 2, 3, либо изменив значения переменных **x1**, **x2**, **y1** и **y2** (zooming графика).

Вот теперь мы и подобрались к вышеупомянутому «шедевру», выставленному в галерее MathSoft, Inc. (рисунок 1).

Как понимает читатель, методика построения графика, показанная на рис. 4, годится и для системы и для одиночного уравнения вида **f(x, y) = 0**. Тут автору попалась на глаза проблема, «выложенная» в форуме Collaboratory (см. статью «Mathcad и Internet, или Сетевой колхоз», КомпьютерПресс, № 3, 2000 г. – http://twt.mpei.ac.ru/ochkov/Collab/Collab.htm): один пользователь Mathcad попросил помочь ему построить график уравнения

**sin(x×sin(x)) – cos(y×cos(y)) = 0**.

Стандартные средства Mathcad позволяют решить эту задачу только через построение линий одного уровня (Contour Plot – «контурная карта»). Решение задачи – линия нулевого уровня (**f(x, y) = 0** – береговая линия на географической карте). Но беда в том, что по вышеописанному конкретному уравнению четкую линию **f(x, y) = 0** получить никак не удавалось – все было как в тумане[5]: видно, что кривая очень сложная, но какая она именно, неизвестно.

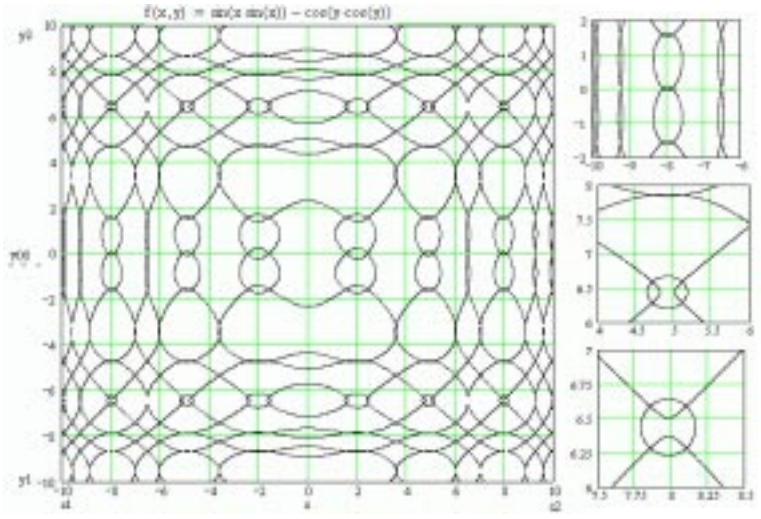

**Рисунок 6.**

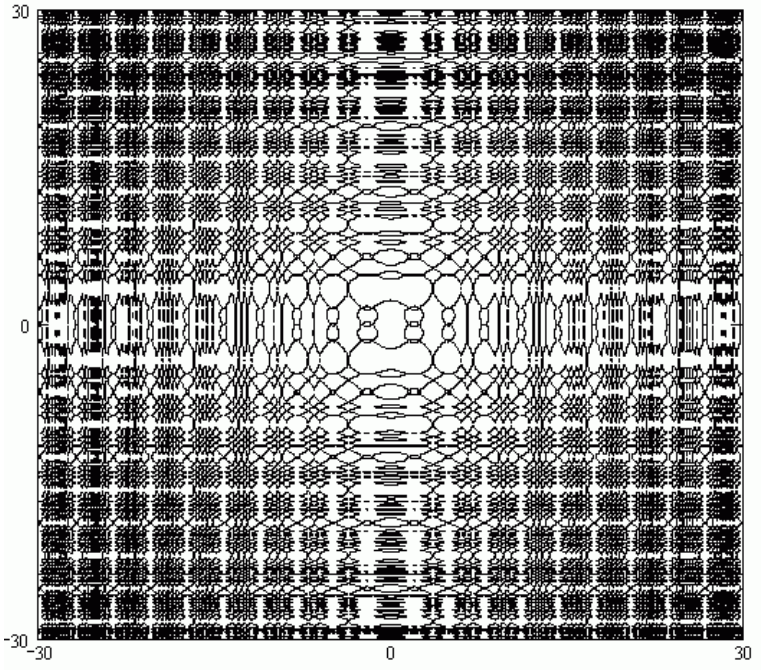


**Рисунок 7.**

На рисунке 5 наша программа сканирования плоскости **x**–**y** из рисунка 4 несколько усложнена (продублирована): сканирование ведется не только «по вертикали» (**for x**... **for y**... – см. рисунок 4), но и «по горизонтали» **for y**... **for x**...). Это делается для того, чтобы повысить четкость графиков, которые на рисунке 4 разрывались на отдельные точки при малом подъеме по оси **y**.[6]

Программа на рисунке 5 работает медленно[7], но верно – на рисунке 6 и рисунке 7 можно увидеть четкие графики нашей функции, сплетающиеся в кружево (*lace*). Под этим названием (*lace*) наше «живописное полотно» и было вывешено в галерее MathSoft, Inc. Получилась квадратная кружевная шаль.

На рисунке 8 можно увидеть полярные графики нашей функции, также кривые, сплетающиеся в кружево – в кружевную салфетку для круглого стола.

Предлагаю читателям «интересное задание» – поиск уравнений вида **f(x, y) = 0**, по которым можно построить оригинальные рисунки.

---

[1] В слово «необычный» вложен следующий смысл. В среде Mathcad есть семь кнопок, нажатие на которые создает на экране дисплея заготовки семи типов графиков: декартов график, полярный график, поверхность, линии уровня, векторное поле, график рассеяния (трехмерный декартов график) и трехмерная столбчатая диаграмма. Если «графическая» задумка пользователя не укладывается в рамки «великолепной семерки Mathcad», то он (пользователь) может либо попытаться поискать специализированный пакет научной и деловой графики (Axum – http://www.mathsoft.com/Axum, например), либо все-таки построить необычные графические объекты – см. рисунок 1.

[2] Хорошее правило: нужно начинать решать задачу с ее аналитического анализа.

[3] **y(x)** – это имя *переменной*, а не *функции*. Читатель, не верь глазам своим: в среде Mathcad возможны имена переменных со скобками, что недопустимо в традиционных языках программирования. В биологии есть явление «мимикрия», когда одно живое существо притворяется не тем, что оно есть на самом деле! Богомол, например, «косит» под сучок, говоря языком современной молодежи. Наша переменная **y(x)**, вернее, не переменная, а вектор, принимает вид функции (мимикрирует). Это сделано для того, чтобы маркировка графика на рисунке 4 выглядела естественной.

[4] Лемниската – «увитая лентами».

[5] Туман (Fog), кстати говоря, – это одно из средств повышения объемности трехмерных графиков. Так, на рисунке 1 можно видеть еще один «шедевр» автора – мост (bridge) в тумане. Два других инструмента повышения объемности графиков – это *перспектива* и *освещение*. Перспектива, наряду с туманом, была использована при строительстве моста (bridge.mcd): полотно проезжей части вдали сужается (Марк Твен говорил, что многие составные немецкие слова так длинны, что их нужно писать, придерживаясь законов перспективы). Освещение (lighting) графиков можно увидеть на семи «шедеврах» нашей виртуальной галереи (рисунок 1): linked.mcd (сцепление двух бубликовторов), klein.mcd (односторонняя бутылка – объемный аналог ленты Мебиуса), sphere lighting (осве-

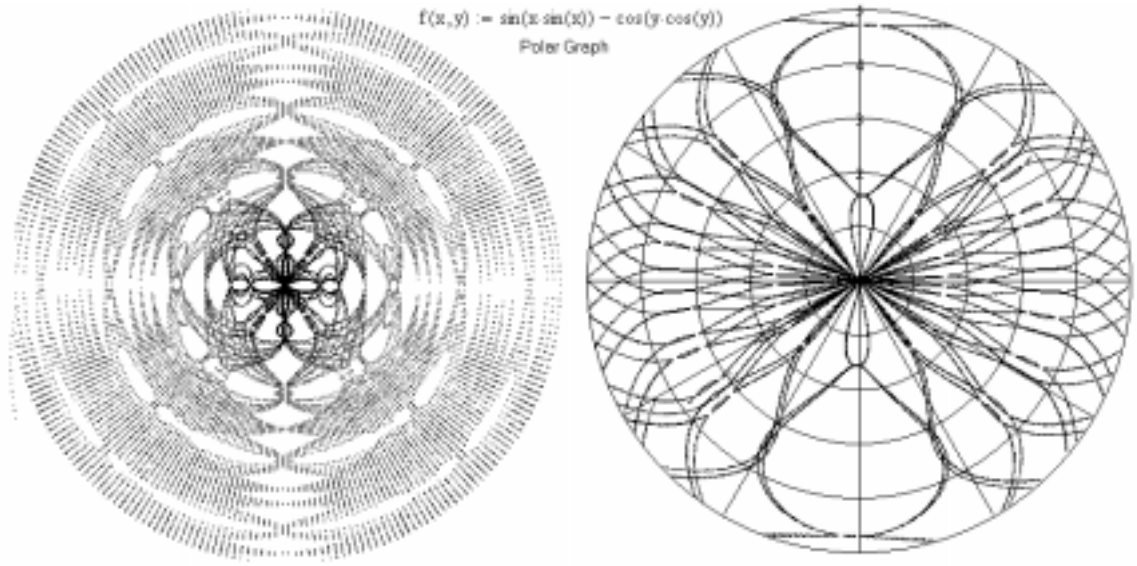


**Рисунок 8.**

щенная сфера – здесь без света от трех источников не обойтись: шар превратился бы в плоский блин), узлы (roteight.mcd и knots.mcd – интересно, можно ли визуализировать гордиев узел, упоминавшийся в статье) и orbital.mcd (это уже не рисунок, а *анимация* полета Шатла вокруг Земли).

[6] Кроме того, в программе на рисунке 5 переменные были «англизированы», так как программа готовилась к публикации в форуме **Collaboratory**.

[7] На рисунке 5 с помощью недокументированной встроенной Mathcad-функции time зафиксировано время счета – время генерации двух векторов (в них по 28828 элементов) – 124 секунды (Pentium II, 233 МГц). Программы, показанные на рисунках 4 и 5, можно оптимизировать по скорости работы. В частности, сильно тормозит работу то, что внутри двойного цикла с параметром (**for x**... **for y**...) дважды вычисляется значение анализируемой функции (в точке **x**, **y** и в точке **x**, **y+dy**: сверху и снизу от предполагаемого корня), хотя это можно делать один раз, используя при этом результат предыдущего вычисления. Изменяя программу в этом направлении, мы ускорим ее работу, но сделаем более сложной для понимания читателями.

***Очков Валерий Федорович, кандидат техн. наук, доцент Московского энергетического института (Технического университета).***

***НАШИ АВТОРЫ***