Использование компиляции «на лету» для быстрого и эффективного построения графиков в инструментальном средстве MathTools

Владимир Янушковский[*]
9 класс, лицей «Вторая школа», Москва

Научный руководитель: И. Р. Дединский, лицей «Вторая школа», Москва

Задачей работы являлось разработка приложения, которое позволило бы быстро строить график введённого пользователем выражения и наглядно наблюдать за его перемещением при изменении значений параметров. При этом особое внимание было уделено вопросам производительности вычислений и качества рисуемого графика.

Приложение разработано на языке C++ с использованием WinAPI. Для этого был использован собственный базовый класс Window, инкапсулирующий оконную процедуру, и все основные окна приложения реализованы в классах, унаследованных от него.

Один из самых важных компонентов программы – это класс Megacalculator, отвечающий за процесс компиляции. Строка, введённая в программу пользователем, отправляется в Megacalculator, который выполняет её синтаксический разбор и генерирует машинный код для процессора Intel 80386. Данный компонент позволяет компилировать выражения, содержащие различные арифметические операции, такие как сумма, произведение, возведение в степень и т.д. Присутствует возможность использования математических функций и констант.

Одной из приоритетных задач работы являлось увеличение качества рисуемого графика и избежание артефактов. Поэтому MathTools предлагает на выбор три различных алгоритма построения графика: с постоянным шагом, с переменным шагом и рекурсивный алгоритм. Все они отличаются скоростью построения и качеством графика.

Приложение MathTools обеспечивает быстрое и качественное построение графиков математических функций, по введённому пользователем выражению. Для повышения производительности используется компиляция в машинный код процессора. Программа позволяет добавлять в функцию дополнительные параметры и наглядно наблюдать зависимость графика от их значений. Для повышения качества рисуемого графика, программа предлагает три различных алгоритма его построения.

Аннотация

MathTools – это разработанное для платформы Win32 приложение, позволяющее строить графики математических функций и наблюдать за их поведением при различных значениях параметров. Для увеличения своей производительности, MathTools компилирует введённое пользователем выражение в исполняемый машинный код. Данный метод позволяет существенно увеличить скорость расчётов значения функции, и быстрее построить график, поэтому ему уделяется особое внимание. Приложение позволяет также задать функцию с дополнительными параметрами, быстро изменять их значения и наблюдать изменение графика. Приложение разработано на языке C++ с использованием Win32 API.

[*] E-mail для связи: error0x0@ya.ru.

Введение

Построение графиков является важной задачей во многих научных и образовательных областях. Наглядное исследование функции – очень полезная возможность, которая позволяет получить представление о поведении функции при различных значениях её параметров.

На первый взгляд, реализация приложения, строящего графики – несложная задача, и она часто встречается даже в учебных курсах программирования. Алгоритмы разбора математического выражения известны, а рисование графика кажется вообще очевидным. Однако, при написании полноценного приложения возникают различные проблемы: скорость построения становится совершенно неудовлетворительной, в графиках появляются разрывы, лишние точки и т.д. При написании MathTools были учтены эти проблемы, и, кроме того, эта программа предлагает эффективный способ увеличения производительности за счёт компиляции выражения в машинный код, а для улучшения качества рисуемого графика программа предлагает 3 различных алгоритма его построения.

Цель работы: разработка приложения, позволяющего быстро и качественно строить графики математических функций, а также отслеживать их поведение при различных значениях параметров. При этом особое внимание должно быть уделено следующим проблемам:

1. Скорость построения должна быть максимально высокой. Навигация по графику не должна сопровождаться постоянными задержками.
2. График не должен «рваться» там, где не надо и не должен содержать артефактов построения (как при построении, например, гиперболы при соединении точек линиями – ложное прохождение графика через начало координат).

Задачи работы:

1. Написание компонента-компилятора (класс Megacalculator), который, на основе строки с математическим выражением, будет генерировать соответствующий машинный код функции, выполняющей подсчёт значения этого выражения.
2. Написание графического интерфейса пользователя, который позволял бы легко и быстро перемещаться по графику, изменять значения параметров, и наблюдать изменение поведения функции.
3. Реализация трех алгоритмов рисования графика, позволяющих избежать «дырок» и лишних точек в нём.

1. Применение компиляции для построения графиков

1.1. Задача вычислений

Пусть пользователь ввёл в программу строку с выражением. Приложение, чтобы построить график, должно рассчитать значение этого выражение для каждого значения параметра x, попадающего на рисунок, а затем нарисовать соответствующие точки на графике. Чтобы посчитать значение выражения в строке, необходимо провести её грамматический разбор. Для него можно использовать алгоритм рекурсивного спуска. Но можно по-разному организовать порядок вычислений.

1.2. Способы решения

1.2.1. Разбор для каждой точки

Наиболее примитивным из способов организации вычислений является разбор строки для каждого значения x. То есть функции подсчёта выражения передаётся значение x, она запускает рекурсивный спуск, в котором разбирает выражение и тут же посчитывает значение функции. Такой способ наиболее простой, но обладает одним недостатком: он катастрофиче-

ски медленный. При проверке, построение графика функции «$y = x + 2$» заняло порядка 30 секунд. Разумеется, такие задержки просто недопустимы.

1.2.2. Компиляция в байт-код для виртуального процессора

Можно поступить иначе: один раз разобрать выражение и сгенерировать байт-код для виртуального процессора со стековой архитектурой (напоминающего математический сопроцессор Intel). Затем для каждого значения x «запускать» этот процессор (как функцию компонента программы), который будет интерпретировать код и подсчитывать выражение. Такой способ существенно быстрее, т.к. позволяет обойтись без рекурсии и постоянного разбора, ограничиваясь однопроходным выполнением кода. Но, тем не менее, при проверке оказалось, что построение функции «$x^2 + tg\ (x)$» занимает порядка 2 секунд. Это гораздо меньше, чем при первом способе, но всё же слишком много.

1.2.3. Компиляция в машинный код

Этот способ заключается в том, что выражение один раз разбирается и генерируется *машинный* код функции для центрального процессора (Intel 80386 и выше) и математического сопроцессора (Intel 80387 и выше), которая возвращает значение этого выражения, при заданных значениях параметров. В этом случае в роли стекового процессора выступает сам процессор компьютера. Это позволяет очень сильно (в 4-5 раз) повысить скорость построения. На практике график строится так быстро, что навигация по нему не вызывает практически никаких задержек.

1.3. Компиляция – наибыстрейший способ

Итак, на основе приведённых данных, можно сделать вывод, что использование компиляции в машинный код позволяет серьёзно повысить скорость вычислений, которая так необходима для быстрого построения графиков.

2. Программа MathTools

2.1. Интерфейс программы

2.1.1. Запуск программы

При запуске приложения отображается заставка (см. рис. 1). Через несколько секунд открывается главное окно (см. рис. 2).

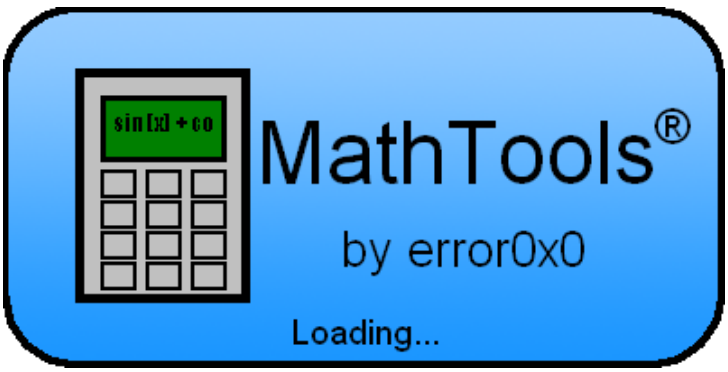

Рис. 1. Загрузочный экран программы.

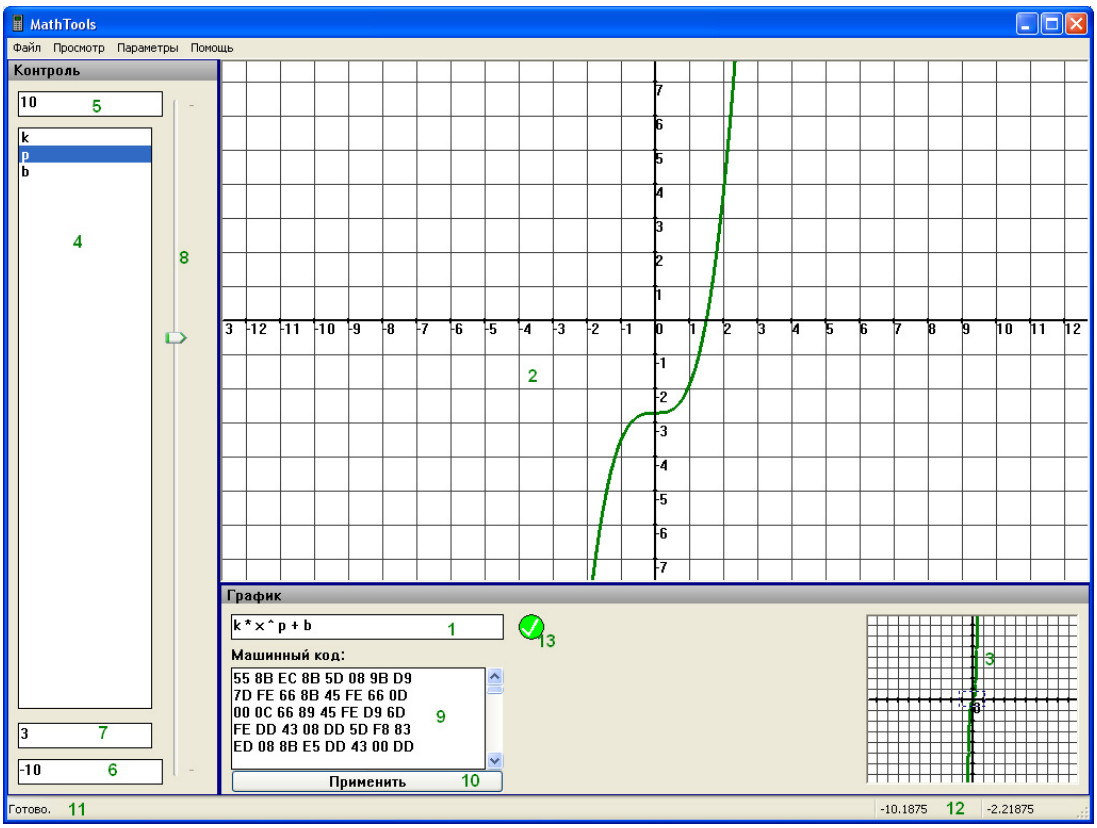

Рис.2. Главное окно программы.

Здесь перечислены элементы главного окна:

1) Поле ввода формулы. При вводе, график перестраивается автоматически.

2) Поле просмотра графика. Перемещаться по графику можно с помощью мыши (Перетаскиванием и колёсиком) или с клавиатуры (Стрелки – перемещение, P – увеличение, L – уменьшение).

3) Поле навигатора. Здесь отображается график в уменьшенном масштабе. Чтобы попасть в нужную позицию, достаточно просто щёлкнуть по ней мышью.

4) Список параметров. (см. ниже)

5) Максимальное значение параметра. Используется для ползунка и при построении графиков семейства функций.

6) Минимальное значение параметра.

7) Текущее значение параметра.

8) Ползунок. Служит для быстрого изменения значения параметра.

9) Поле машинного кода. Отображает сгенерированный компилятором машинный код функции в шестнадцатеричном представлении. Машинный код можно изменить.

10) Кнопка подтверждения изменения кода. Если после изменения машинного кода нажать эту кнопку, то будет подставлен новый код и построен график функции.

11) Статус программы. Отображает текущее состояние приложения.

12) Координаты курсора мыши. Координаты относительно точки (0; 0).

13) Индикатор правильности формулы. Отображает правильность введённого выражения. При измененном машинном коде отображает букву R, и при щелчке восстанавливает функцию из формулы.

Примечание: все поля, требующие ввод числа могут принимать формулу.

2.1.3. Работа с параметрами

Вводимое в приложение выражение помимо x может содержать и другие параметры. При появлении в выражении нового идентификатора, в список параметров добавляется новый, с данным именем и значением 0. При удалении его из выражения, он удаляется и из списка параметров. Щёлкнув по параметру, можно изменить его значение ползунком или, непосредственно введя его в поле текущего значения. При этом график будет перестроен автоматически. А выбрав пункт «Просмотр» -> «График семейства функций», можно построить множество графиков данной функции при разных значениях выбранного параметра от минимального до максимального. Эта процедура может занять некоторое время, но позволяет получить очень красивые изображения.

2.2. Структура программы

Так как приложение разрабатывалось с помощью Win32 API, который не содержит в себе объектно-ориентированных возможностей, и целиком основан на функциях и структурах, потребовалось определённое время, чтобы адаптировать программу к данному интерфейсу.

Основные окна в приложении реализованы с помощью классов, унаследованных от собственного базового класса Window, который инкапсулирует в себе оконную процедуру и позволяет в производных классах её не использовать.

Был написан менеджер панелей, который содержит в себе панели навигации и параметров и управляет их расположением и перемещением.

Одним из самых важных компонентов приложения является класс *Megacalculator*. Он занимается компиляцией выражения в машинный код. Подробное его описание см. ниже.

Координатное поле является дочерним окном приложения и включает в себя непосредственно само рисование графика различными алгоритмами.

2.3. Компонент Megacalculator

Этот компонент занимается генерацией машинного кода на основе введённого выражения и является одним из самых важных компонентов программы.

2.3.1. Разбор и компиляция строки

Когда пользователь вводит в MathTools выражение, оно сразу же направляется в *Megacalculator*. Он выполняет синтаксический разбор строки алгоритмом рекурсивного спуска. Обрабатывая строку, он генерирует байты машинного кода, которые записывает в память. Вот пример:

Функция: $y = x^2$

Код в шестнадцатеричном виде и в дизассемблере (листинг 1):

```
00000200: 55        push ebp
00000201: 8BEC      mov ebp,esp
00000203: 8B5D08    mov ebx,dword ptr [ebp+08]
00000206: 9B        wait
00000207: D97DFE    fstcw dword ptr [ebp-02]
0000020A: 668B45FE  mov ax,word ptr [ebp-02]
0000020E: 660D000C  or ax,0C00
00000212: 668945FE  mov word ptr [ebp-02],ax
00000216: D96DFE    fldcw dword ptr [ebp-02]
00000219: DD4300    fldq dword ptr [ebx]
0000021C: DD5DF8    fstpq dword ptr [ebp-08]
0000021F: 83ED08    sub ebp,00000008
00000222: 8BE5      mov esp,ebp
00000224: DD4300    fldq dword ptr [ebx]
00000227: DD5DF8    fstpq dword ptr [ebp-08]
0000022A: 83ED08    sub ebp,00000008
0000022D: 8BE5      mov esp,ebp
0000022F: DD4508    fldq dword ptr [ebp+08]
```

```
00000232: DD4500    fldq dword ptr [ebp]
00000235: DEC9      fmulp st(1),st
00000237: DD5D08    fstpq dword ptr [ebp+08]
0000023A: DDD8      fstp st(0)
0000023C: 83C508    add ebp,00000008
0000023F: 8BE5      mov esp,ebp
00000241: 9B        wait
00000242: D97DFE    fstcw dword ptr [ebp-02]
0000245: 668B45FE  mov ax,word ptr [ebp-02]
00000249: 6625FFF3  and ax,F3FF
0000024D: 668945FE  mov word ptr [ebp-02],ax
00000251: D96DFE    fldcw dword ptr [ebp-02]
00000254: DD4500    fldq dword ptr [ebp]
00000257: 83C508    add ebp,00000008
0000025A: 8BE5      mov esp,ebp
0000025C: 5D        pop ebp
0000025D: C3        ret
```

Листинг 1. Результат компиляции функции $y = x^2$.

Примечание: Приведенный код – настоящий машинный код функции, выведенный программой. Он содержит все операции со стеком, с параметрами, и, безусловно, команду возврата RET (0C3h). Если вручную занести этот код в байтовый массив, а затем выполнить вызов по указателю *(*) (double*),* передав указатель на число, то функция вернёт квадрат этого числа.

2.3.2. Возможности компонента

Класс *Megacalculator* предоставляет следующий набор операций (табл. 1):

Таблица 1. Набор компилируемых операций.

Операция	**Описание**	**Порядок выполнения**	**Приоритет**
^	Степень	Справа налево	max
*,/	Умножение, деление	Слева направо	
+,-	Унарные + и –	Справа налево	
+,-	Сумма, разность	Слева направо	min

Megacalculator позволяет использовать в выражении математические константы π и *e.* Как было ранее сказано, он позволяет работать с выражениями, имеющими дополнительные параметры.

2.3.3. Использование готового кода

Когда код сгенерирован и сохранён в памяти, программе для подсчёта значения выражения достаточно использовать обычный вызов функции. То есть программа создаёт обычный указатель на функцию *double (* fptr) (double*),* которому присваивает адрес местоположения кода. Так как с точки зрения времени выполнения указатель на сгенерированный код – то же самое, что и указатель на функцию, скомпилированную с языка C++, то достаточно просто выполнить вызов, чтобы посчитать значение выражения. А, чтобы посчитать функцию для другого значения параметра, достаточно просто вызвать её с другими параметрами.

2.4. Отображение графика

2.4.1. Задача и основные проблемы

Как было уже отмечено, на первый взгляд отображения графика кажется задачей простой. Ведь достаточно перебрать все значения параметра *x*, попадающие на экран и отметить точки, с координатами, соответствующими значениям функции. Но при таком её решении могут возникнуть неприятности: график может рваться, содержать лишние точки и т.д. Чтобы избежать подобных артефактов MathTools предлагает на выбор 3 различных по скорости и качеству алгоритма рисования графика.

2.4.2. Настройка рисования

Выбрав пункт меню «Параметры» -> «Алгоритм построения…» можно настроить параметры рисования графика (рис.3).

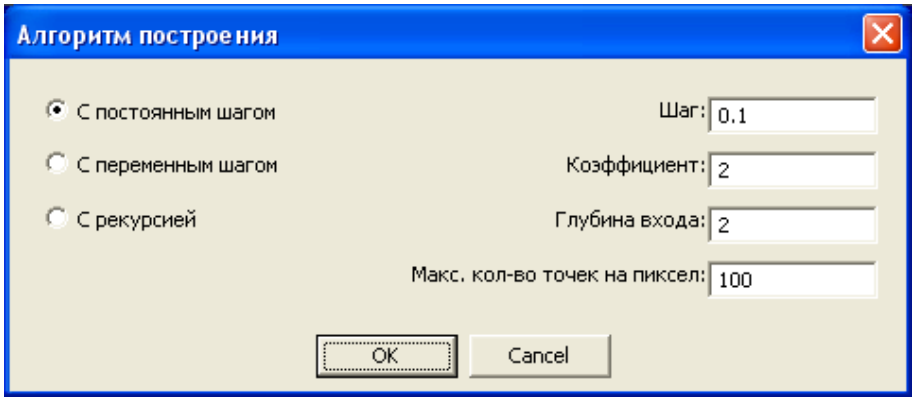

Рис.3. Окно настройки алгоритма рисования.

Этот диалог позволяет выбрать один из трёх алгоритмов построения графика функции и настроить его параметры.

2.4.3. Алгоритм с постоянным шагом

Если строить по одной точке графика на каждый пиксель оси x, то можно прийти к следующей ситуации (см. рис. 4).

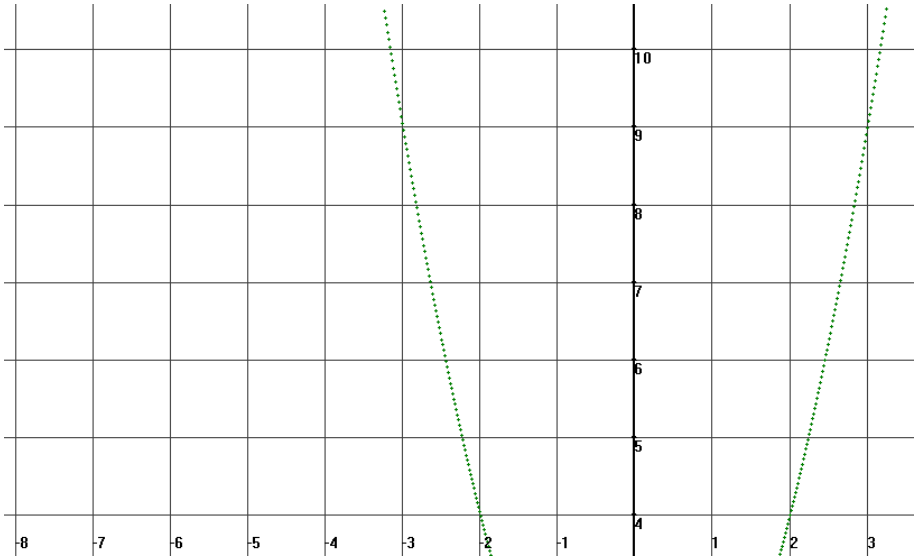

Рис.4. График функции $y = x^2$. 1 пиксель – точка.

В общем, суть алгоритма с постоянным шагом такая же, но можно установить параметр «Шаг» в значения меньшие 1. Тогда можно добиться сколь угодно высокого качества, правда, когда шаг достигнет значения порядка 0.0001, скорость построения станет очень низкой. К плюсам данного алгоритма относится та же скорость, она постоянна для любого введённого пользователем выражения.

2.4.4. Алгоритм с переменным шагом

Как видно из данной ситуации, в точках, где функция растёт со скоростью большей единицы, недостаточно нарисовать одну точку, требуется посчитать функцию в промежуточных значениях и поставить дополнительные точки. Собственно, данный алгоритм и занимается этим. Он берёт разность между значениями функции на соседних пикселях и вычисляет соответствующее количество значений для промежуточных точек. Параметр «Коэффициент» используется для устранения незначительных артефактов, которые могут возникнуть в некоторых функциях. Стоит заметить, что, так как шаг переменный, то скорость тоже становится переменной.

2.4.5. Рекурсивный алгоритм

Некоторые функции могут сильно и часто колебаться в определённых местах. В этом случае предыдущий алгоритм может не учесть резких изменений функции на малом интервале, что может сказаться на качестве нарисованного графика. Рекурсивный алгоритм позволяет этого избежать, т.к. он, как и предыдущий алгоритм, считает разность между значениями в соседних пикселях и вычисляет значения в промежуточных точках, но затем рекурсивно вызывает ту же самую процедуру уже для промежуточных точек, и так до тех пор, пока различия между значениями не станут меньше 1. Этот алгоритм имеет параметр «Глубина входа», который показывает, сколько раз можно рекурсивно вызывать процедуру вычислений. Как правило, достаточно значения 2, иногда 3. Этот алгоритм самый медленный из всех, но самый качественный.

Примечание: четвёртый параметр «Макс. кол-во точек на пиксел» определяет максимальное количество точек на один пиксель оси x. Это позволяет избежать зависания программы в случае слишком сложных функций.

2.4.6. Сравнение алгоритмов

На рис. 5, 6 и 7 приведены результаты построения графика функции $f(x) = sin\ (1/x)$.

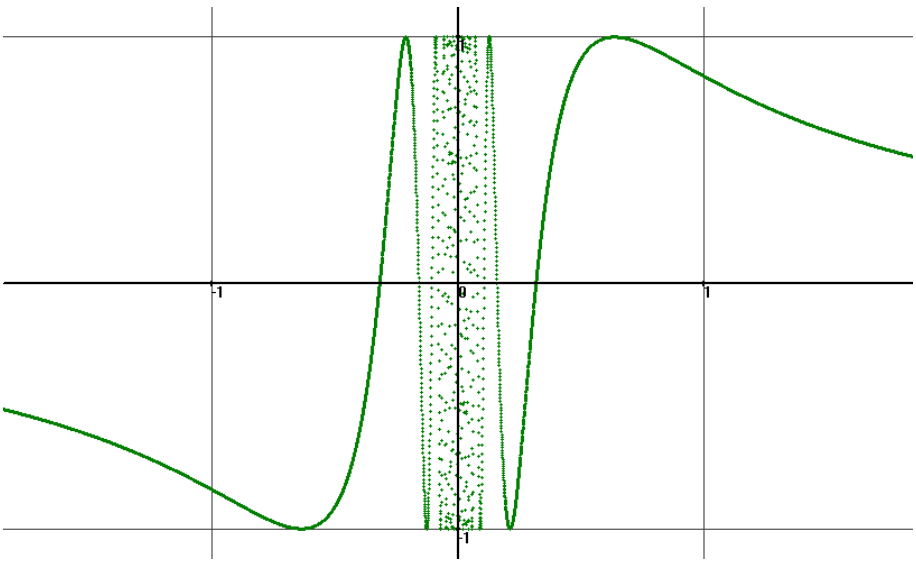

Рис.5. Функция $f(x) = sin\ (1/x)$. Алгоритм с постоянным шагом. Шаг = 0.1.

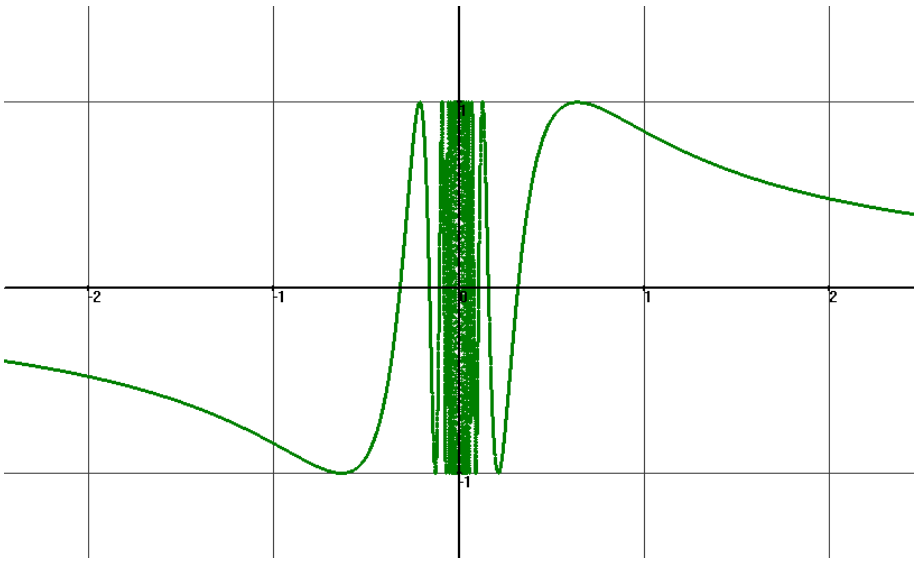

Рис.6. Функция $f(x) = sin(1/x)$. Алгоритм с переменным шагом. Коэффициент = 1.

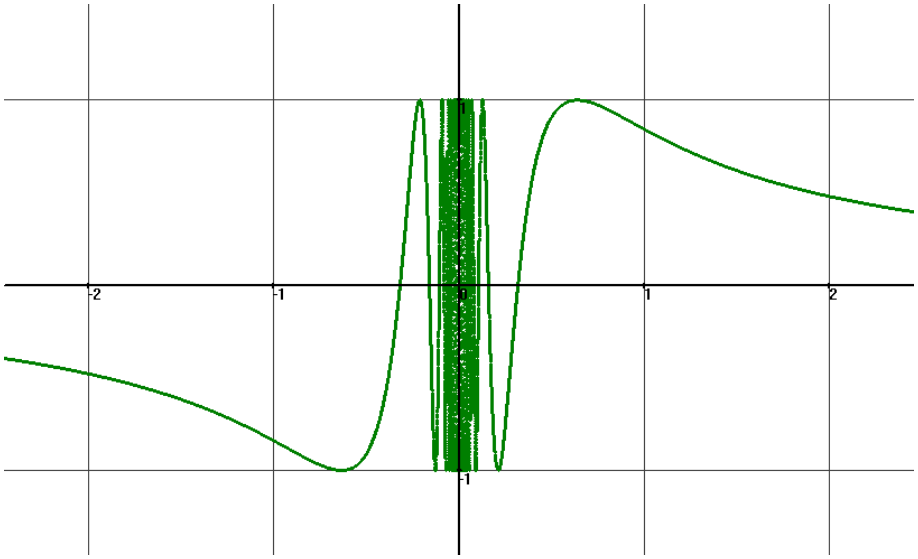

Рис.7. Функция f (x) = sin (1/x). Рекурсивный алгоритм. Глубина входа = 2.

Стоит заметить, что ни один из трёх алгоритмов не идеален, но рекурсивный обеспечивает наивысшее качество построения. Конечно, всё зависит от ситуации, для каких-то функций подойдёт простейший алгоритм с единичным шагом, а для некоторых без рекурсии не обойтись.

2.5. Дополнительные возможности

В MathTools представлены несколько интересных возможностей:

1) Построение семейства графиков. Достаточно выбрать параметр, указать наименьшее и наибольшее значение и выбрать «Просмотр» -> «График семейства функций…». Вот, результат подобного действия:
2) Сохранение изображения. Ctrl + S или «Файл» -> «Сохранить изображение…». Сохраняет изображение в формате .bmp.
3) Настройка цвета графика с помощью «Параметры» -> «Цвет графика».

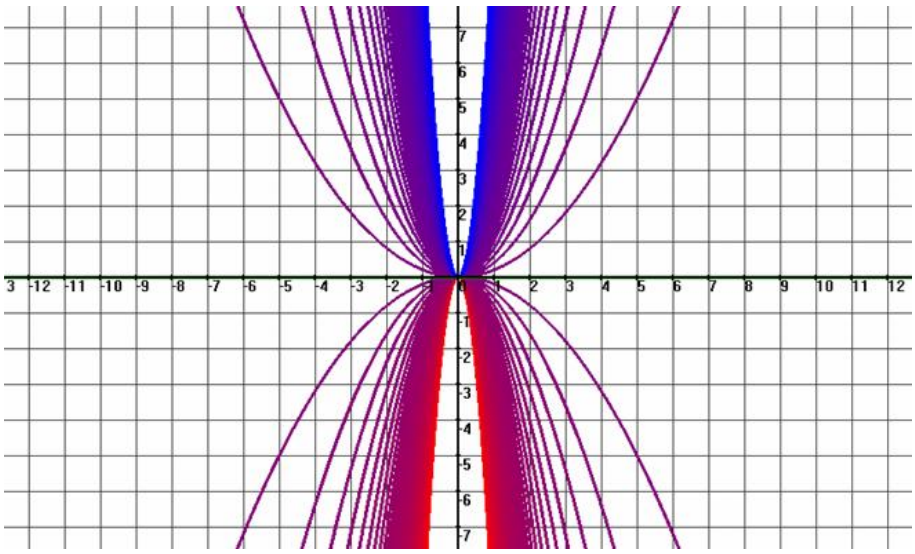

Рис.8. Семейство графиков $f(x) = ax^2$.

Результаты

Разработано приложение MathTools, позволяющее исследовать графики функций, их зависимость от значений параметров. Приложение обеспечивает хорошее качество рисования графиков и высокую скорость работы. Приложение использует компиляцию в машинный код для ускорения работы и предоставляет 3 различных алгоритма рисования.