ЗАДАНИЯ 2

1. Построить семейства графиков кривых при различных значениях параметра заданных уравнением в декартовых координатах, или параметрически, или в полярных координатах (составлять программы так, чтобы пределы и шаг изменения параметра *a* можно было задавать произвольно).

Существует понятие текущего графического окна и понятие текущих координатных осей. В одном графическом окне может располагаться сразу несколько координатных осей (как этого можно добиться, см., например, ниже, в п. 7). Метод Pero\draw() отображает график в текущих координатных осях или инициирует создание новых координатных осей в текущем графическом окне, если координатных осей в нем до этого не существовало. Если и графического окна до этого не существовало, то сначала автоматически инициируется его создание, затем инициируется создание в нем координатных осей (графического объекта axes), а затем уже в них отображается график. Таким образом, если мы желаем, чтобы каждый новый график строился в отдельном графическом окне (в размещенных в нем новых координатных осях), а не строился в прежних координатных осях поверх старого графика, то перед построением каждого нового графика с помощью Pero\draw() следует предварительно создавать новое графическое окно, которое при этом будет становиться текущим, с помощью встроенной функции figure.

А) – уравнение окружности,

При построении графиков концентрических окружностей следует периодически изменять их цвет, а толщину линии каждого следующего графика - увеличивать с помощью некоторого коэффициента, большего 1, для чего следует воспользоваться методом Pero\set.

Б) – параметрическое описание «астроиды»;

каждую «астроиду» вписать в окружность радиуса *a*, линия окружности должна быть пунктирной,

В) – «улитка» Паскаля в полярных координатах, (при *a*=0 имеем окружность, при *a*=1 – «кардиоиду», при *a*=2 - трисектриссу; вид этих кривых и дополнительную информацию см., например, на странице <https://ru.wikipedia.org/wiki/Улитка_Паскаля>)

Г) , (в частности, при , - это уравнение окружности)

Все эти кривые – замкнутые, поэтому каждый раз следует убедиться, что, во-первых, отрисовываются именно замкнутые кривые (в виде замкнутых многоугольников), и, во-вторых, что, при отрисовке, не происходит «самоналожения».

Д)

можно ограничиться только случаем *a*.

Обратите внимание, что выражение, стоящее под знаком произведения не определено при *x*=0, и это может привести к неприятности при построении графика. Поэтому целесообразно с самого начала доопределить это выражение по непрерывности единицей при *x* = 0 (вспомните 1-ый замечательный предел), написав соответствующую вспомогательную функцию, или воспользовавшись уже имеющейся встроенной библиотечной функцией sinc – это очень полезная, часто применяемая функция (надо сделать и так, и так).

Особенность данного графика состоит в том, что выражение под знаком логарифма в некоторых точках обращается в ноль, поэтому значения логарифма в окрестностях этих точек достигают неограниченно больших отрицательных значений. В следствии этого верхняя, наиболее информативная, часть графика будет получаться слишком сжатой по вертикали. Поэтому будет необходимо верхнюю часть графика растянуть до нужного масштаба, обрезав малоинформативную нижнюю его часть. Сделать это можно воспользовавшись встроенной функцией axis([xmin,xmax,ymin,ymax]), которая при таком фактическом параметре (массив double из 4-х элементов) позволяет отображать в координатных осях нужную часть графика, ограничивая её заданной рамкой. В данном случае потребуется только выбрать (в результате эксперимента) подходящий диапазон значений по оси Oy, а диапазон значений по оси Ox уже определен: следует положить xmin = -10, xmax = 10.

1. – описание трехлепестковой «розы» в полярных координатах; можно считать *a.*

Построить семейство графиков трехлепестковой «розы», например, при *a*=1, повернутых относительно исходного (соответствующего данному уравнению) на все углы, кратные 30 градусам. Толщины линий графиков следует выбирать так, чтобы соседние графики имели разную толщину линии (достаточно 2-х различных значений толщины).

Для этого воспользоваться методом Pero\transform, запрограммировав преобразование координат точки при повороте ее радиус-вектора на заданный угол. При этом понадобится возможность копирования объектов класса Pero с помощью копирующего конструктора (ВНИМАНИЕ: такая возможность отсутствует в старой версии класса Pero):

если переменная p1 – содержит ссылку на некоторый объект класса Pero, то создать НОВУЮ КОПИЮ этого объекта и получить на него ссылку можно так:

p2 = Pero(p1);

Теперь p2 – это уже НОВАЯ ссылка на НОВЫЙ (физически другой) объект, но с теми же самыми свойствами, что и исходный.

Предложить и реализовать другое решение этой задачи, в котором бы не использовался метод Pero\transform.

1. Мысленно поместим центр окружности радиуса в точку с координатами . Пометим точку окружности с координатами и представим, что окружность подобно колесу без проскальзывания покатилась по оси Ox вправо (или влево). Тогда помеченная точка окружности будет описывать кривую в виде периодической последовательности дуг, соприкасающаяся с осью Ox при каждом полном повороте окружности вокруг своей оси. Эта кривая называется *циклоидой*. Можно убедиться, что циклоида задается параметрически так

где – угол поворота окружности при ее качении; можно считать *a.*

.

Требуется составить программу, которая бы визуализировала описанный выше процесс построения циклоиды (по прямой катится окружность и одна выделенная ее точка описывает циклоиду).

Для этого следует воспользоваться, во-первых, встроенной функцией pause(dt) с фактическим параметром dt, смысл которого – интервал времени, выраженный в секундах, на который приостанавливается выполнение программы каждый раз, при выполнении MATLAB’ом функции pause. Во-вторых, для визуализации качения окружности потребуется ее перерисовывать с соответствующим смещением (после задержки на dt секунд), т.е. удалять старый график и строить новый. В MATLAB’е это можно сделать разными способами. Самый простой для объяснения (но не всегда самый лучший) выглядит так. Чтобы удалить старый график надо располагать ДЕСКРИПТОРОМ этого графика (в MATLAB’е все графические объекты имеют дескрипторы – это их числовые идентификаторы, т.е. числа под которыми они регистрируются системой MATLAB при их построении). Получить дескриптор графика и записать его в какую-либо переменную (типа double) можно, если при построении графика с помощью Pero\draw использовать предусмотренный на этот случай фактический выходной параметр:

h = p.draw(); % p – ссылка на объект класса Pero

В результате в переменную h будет записан дескриптор построенной кривой. Теперь при необходимости эту кривую можно будет, например, удалить с помощью встроенной функции delete:

delete(h)

Однако для смещения графика можно было бы просто изменить нужным образом соответствующие свойства этого графического объекта (эти свойства называются xdata и ydata – это массивы с координатами точек графика) с помощью пары встроенных функций set() и get(). Для этого тоже понадобился бы дескриптор графика.

Для визуализации помеченной точки окружности следует ее отрисовывать в виде маленькой окружности, полностью «залитой» цветом (например, красным). Для построения фигуры, «залитой» цветом следует вызывать метод draw() c соответствующим фактическим параметром:

p.draw(‘patch’)

(при этом просто p.draw() было бы эквивалентно p.draw(‘line’))

1. Известно, что при выполнении определенных условий любое периодическое колебание может быть представлено в виде суммы гармонических колебаний с некоторыми частотами и амплитудами. Такая сумма, в общем случае содержащая бесконечное число слагаемых, называется рядом Фурье рассматриваемого периодического колебания, а соответствующие суммы, содержащие только первые *n* слагаемых (все члены ряда занумерованы определенным образом), называются частичными суммами этого ряда.

Требуется построить семейство графиков частичных сумм ряда Фурье заданной периодической четной функции (ее график тоже надо построить):

где – коэффициенты, которые вычисляются по специальным формулам, приведенным ниже, *Т* – период функции , к которой сходится (в определенном смысле) последовательность функций при .

1. Пусть на главном периоде, т.е. при ,

где *A* – амплитуда прямоугольного импульса, его длительность, *T* – период функции , причем . Т.е., говоря техническим языком, имеем периодическую последовательность прямоугольных импульсов положительной полярности. В этом случае , (). Величины *A* и *Т* можно положить равными 1, а величины *n* и должны быть параметрами программы.

Обратите внимание, что функция была определена только при условии . Продолжить её на всю числовую ось можно, имея в виду ее периодичность, положив, при произвольном , , где rem() – встроенная библиотечная функция, вычисляющая симметричный (т.е. со знаком + или -) остаток от целочисленного деления 1-го своего аргумента на 2-ой, что в данном случае означает требуемое приведение *t* к интервалу (*-T*/2, *T*/2) (для справки: rem(a,b) == a – fix(a/b)\*b, где встроенная функция fix(x) вычисляет целую часть х и при этом fix(-х) == -fix(х)).

Б) Пусть , т.е., говоря техническим языком – это «выпрямленная» косинусоида (имеется в виду аналогия с работой специальной электронной схемы, выпрямляющей переменное напряжение). Тогда , и, при .

В) Пусть

,

т.е., говоря техническим языком – это «детектированная» косинусоида (имеется в виду аналогия с работой полупроводникового детектора в цепи переменного тока, предотвращающего протекание тока в обратном направлении). Тогда , и, при .

Во всех пунктах задания А), Б) и В) для вычисления коэффициентов соответствующего ряда Фурье, целесообразно написать отдельную короткую функцию, т. к. тогда отличия решений всех этих заданий будут сосредоточенны лишь в этой маленькой функции. Но если бы требовалось выполнить только какой-то один из этих 3-х пунктов, то все равно следовало бы сделать именно так.

1. Построить семейство графиков следующих функций:

где , а функции определены рекуррентно:

, , .

Программа должна допускать произвольное задание числа *n*. Все эти графики построить в одних координатных осях, различая их цветом. Шаг изменения аргумента при построении графика выбрать таким, чтобы на нем не было видно изломов.

Интересное и важное свойство функций состоит в том, что (но , т.е. эти функции имеют порядок гладкости, в точности равный *k*; убедитесь в этом).

1. При заданных значениях параметров m, n построить график, так называемой, up-функции (читается: ап-функция) на отрезке [-1; 1], приближенно определяемой выражением

где можно положить (в действительности up-функция определяется как предел данного выражения при , но при указанных конечных значениях погрешность результата получается уже незначительной). Шаг изменения аргумента при построении графика выбрать таким, чтобы на нем не было видно изломов.

1. Построить также графики функций и в разных координатных осях, причем координатные оси с этими графиками расположить под координатными осями графика функции .

Для получения такого расположения графиков можно воспользоваться встроенной функцией subplot(i,j,k), которая автоматически разбивает все поле графического окна на равные прямоугольные области, в которых могут размещаться координатные оси, расположенные друг под другом в i строках, по j – в каждой строке, причем все эти прямоугольные области имеют линейную нумерацию и k – это порядковый номер той из них, в которой в данный момент требуется отобразить график (эту функцию надо выполнять перед выводом каждого очередного графика, с тем, чтобы она создавала соответствующие координатные оси, или, если они уже существуют, делала их текущими).

1. Написать функцию, возвращающую для заданного аргумента (входной параметр функции) значение многочлена Чебышёва 1-го или 2-го рода  *n*-го порядка (порядок многочлена также должен передаваться в функцию через соответствующий входной параметр), воспользовавшись соответствующей рекуррентной формулой. А именно,

и, соответственно,

Построить графики этих многочленов порядков 1,2,…,*n* на [-1, 1] (вид этих графиков можно посмотреть, например, на <https://ru.wikipedia.org/wiki/Многочлены_Чебышёва>).

1. Пусть (по определению, 0!=1); известно, что . Написать функцию, получающую значение *n* и возвращающую . Требуется, чтобы при этом число используемых арифметических операций оценивалось как .
2. Многие функции, такие как тригонометрические функции, показательная функция, логарифмическая функция и другие, в том числе те, которые изучаются уже в курсе высшей математики, могут быть выражены в виде сумм с бесконечным числом слагаемых, представляющих собой одночлены, зависящие от целых неотрицательных степеней аргумента функции. Такие суммы называются степенными рядами. Они понимаются как пределы соответствующих частичных сумм при неограниченном возрастании числа слагаемых их составляющих; примеры см. в таблице ниже.

Построить семейство графиков частичных сумм (n=1,2,…,10) данного степенного ряда как функций аргумента *x* на указанном интервале сходимости ряда и график соответствующей предельной функции , при заданном выражении для *k*-го члена степенного ряда и заданном выражении соответствующей предельной функции.

(Возможно, что при некоторых значениях *x* последовательность частичных сумм не имеет конечного предела; в этом случае говорят, что область сходимости степенного ряда ограничена; в таблице указаны интервалы сходимости соответствующих рядов)

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  | Область сходимости ряда |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |

В каждом случае предварительно требуется получить (вывести) рекуррентную формулу (или, в №10, несколько рекуррентных формул) для последовательности членов ряда. Тогда ее можно будет использовать при вычислении частичных сумм, вычисляя как саму частичную сумму, так и очередное слагаемое этой суммы, в одном простом цикле. При этом верхняя оценка временнОй сложности алгоритма вычисления суммы ряда должна быть , а не , как если бы коэффициенты ряда вычислялись непосредственно по заданной формуле.

Далее, для вычисления *n*-ой частичной суммы на основе полученных формул должна быть составлена специальная подпрограмма (функция, возвращающая значение этой суммы). Желательно получить такой алгоритм вычисления частичных сумм, в котором бы число требуемых арифметических операций было минимальным. Графики частичных сумм и предельной функции отобразить разными цветами. Пределы и шаг изменения аргумента выбрать самостоятельно.

При отладке программы следует сначала убедиться, что *n*-ая частичная сумма вычисляется правильно. Для этого следует задаться каким-нибудь подходящим значением аргумента *x* (зафиксировать его) и повторно вычислять частичную сумму с помощью составленной подпрограммы, каждый раз увеличивая значение *n* до тех пор, пока получаемое значение не станет совпадать во всех (!) десятичных разрядах со значением, получаемом с помощью имеющегося выражения предельной функции.

Чтобы в командном окне при выводе значений типа double отображались все 16 десятичных разрядов (а не только 5, как имеет место по умолчанию) необходимо предварительно исполнить команду format long (прямо из командной строки; чтобы потом вернуться к стандартному формату вывода, потребуется снова выполнить команду format short или просто format). Разумеется, у функции, возвращающей значение частичной суммы, для величин *x* и *n* должны быть предусмотрены соответствующие входные формальные параметры.

1. В математике и ее приложениях важную роль играют, так называемые, функции Бесселя.

Функцию Бесселя первого рода порядка *m*, если ограничиться только целыми положительными значениями порядка ( ), можно определить с помощью следующего степенного ряда:

Построить семейство графиков функций Бесселя для при , используя для каждого *x* приближение значения функции с помощью некоторой частичной суммы данного степенного ряда (образцы требуемых графиков см., например, на странице <https://ru.wikipedia.org/wiki/Функции_Бесселя>).

Если обозначить , т.е. , и вести величину 𝜀>0 – максимально допустимую погрешность искомого значения функции , то алгоритм вычисления соответствующей частичной суммы можно записать так: пока (*k*=0,1,2,…), суммировать последовательность членов ряда . Отметим, что указанный алгоритм опирается на один широко известный факт из математического анализа.

За 𝜀 можно принять любое сколь угодно малое число, например, можно положить или и, в любом случае, при каком-то *k*, этот процесс суммирования обязательно остановится. Однако можно обойтись и без явного использования величины 𝜀. Если обозначить, то алгоритм вычисления требуемой частичной суммы можно записать еще и так: пока , суммировать последовательность членов ряда . Конечно, с математической точки зрения равенство невозможно. Но программная реализация алгоритма суммирования с таким условием завершения опирается на особенность машинной арифметики в формате с «плавающей точкой». А именно, при сложении двух значений типа double сначала производится выравнивание порядков, т.е. приведение меньшего по порядку слагаемого к порядку большего, после чего уже складываются мантиссы. В результате, из-за того, что длина мантиссы фиксирована и в пересчете на десятичные разряды равна 16, значащие младшие разряды меньшего числа будут уходить за разрядную сетку (т.е. отбрасываться), вплоть до полного обнуления мантиссы, если только десятичные порядки слагаемых различаются не менее чем на 16.

1. Фигу́ры Лиссажу́ — замкнутые траектории, прочерчиваемые точкой, совершающей одновременно два гармонических колебания в двух взаимно перпендикулярных направлениях. Параметрическое описание фигур Лиссажу имеет вид:

где *A*, *B* — амплитуды колебаний, *a*, *b* — их частоты, δ — сдвиг фаз.

Фигуры Лиссажу вписываются в прямоугольник, центр которого совпадает с началом координат, а стороны параллельны осям координат и расположены по обе стороны от них на расстояниях, равных амплитудам колебаний (см., например, <https://ru.wikipedia.org/wiki/Фигуры_Лиссажу>).

Вид фигур зависит от соотношения между периодами (частотами), фазами и амплитудами обоих колебаний. В простейшем случае, когда частоты колебаний совпадают, фигуры представляют собой эллипсы, которые при вырождаются в отрезки прямых, а при и превращаются в окружность, при — это парабола.

При *a* = 1, *b = N* и (*N* — натуральное число) фигуры Лиссажу являются полиномами Чебышёва первого рода степени *N*.

Фигуры Лиссажу являются замкнутыми при условии, что отношение частот *a/b* есть рациональное число. Если, например, периоды обоих колебаний достаточно близки, но в точности не совпадают, то разность фаз всё время меняется, вследствие чего «эллипс» всё время деформируется. Если отношение периодов (частот) ­— целое число, то через промежуток времени, равный наименьшему кратному обоих периодов, движущаяся точка снова возвращается в то же положение.

А) Составить программу, строящую график Фигуры Лиссажу, заданной своими параметрами. Построить (автоматически) фигуры Лиссажу при *A=B*, δ = π/2 и

*a =* 1*, b =* 2 или *a =* 3*, b =* 2 или *a* = 3*, b =* 4 или *a =* 5*,*  *b =* 4 или *a =* 5*,*  *b* = 6 или *a* = 9*,*  *b* = 8.

Графики этих фигур разместить в одном графическом окне в 6 различных координатных осях, создать которые можно с помощью встроенной функции subplot (подробнее см. выше в задании 7).

Б) Создать анимацию, которая показывает плавные превращения кривых Лиссажу при монотонном (на периоде) изменении соотношения в пределах от 0 до 1 с шагом 0.01, при условии (подобно тому, как это сделано на странице <https://ru.wikipedia.org/wiki/Фигуры_Лиссажу>).

1. А) Построить семейства софокусных гипербол и эллипсов.

Для этого воспользоваться следующим уравнением:

где *c* – расстояние от начала координат до любого из фокусов (лежащих на оси Ох), *a* – параметр:, , при - уравнение описывает некоторый эллипс, а при – некоторую гиперболу. Фиксируя *c*, положив, например, , и изменяя параметр *a* в необходимых пределах с каким-либо шагом, и для каждого значения параметра вычерчивая соответствующую кривую, можно получить требуемый результат.

Б) Построить семейства софокусных парабол. Для этого воспользоваться следующими уравнениями: где *p* – расстояние от вершины соответствующей параболы до ее директрисы. Фокус каждой такой параболы будет находиться в начале координат, при этом в первом случае ветви парабол направлены вправо, а во втором – влево. Величину *p* надо рассматривать как переменный параметр, определяющий семейства кривых.

1. Построить семейство овалов Кассини. Овал Кассами состоит из всех точек плоскости, произведение расстояний до которых от двух фиксированных точек с координатами , называемых фокусами, есть величина постоянная и равная . В таких обозначениях уравнение овалов Кассами в полярных координатах имеет вид

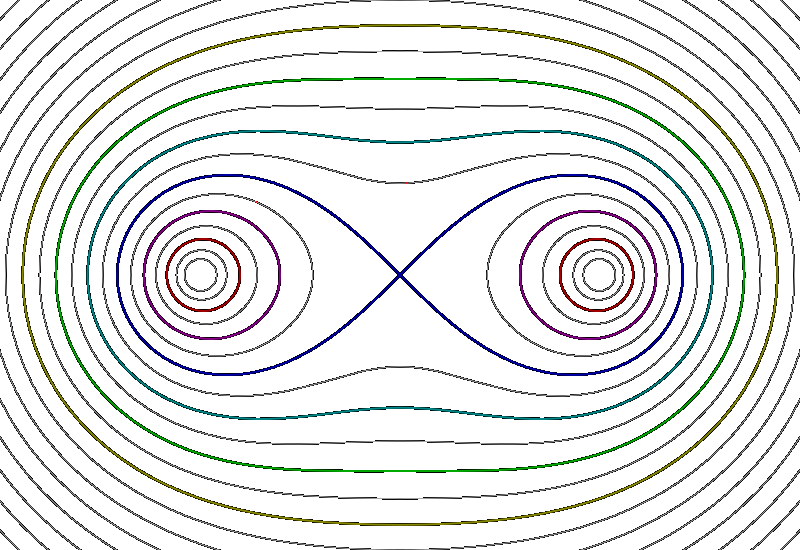
(Вывод уравнения можно найти, например, на странице <https://ru.wikipedia.org/wiki/Овал_Кассини>.)

Решая это биквадратное уравнение, находим что

или

Откуда видно, что форма этой кривой зависит только от отношения , причем *c* – это просто масштабный коэффициент, который при построении графиков удобно положить равным 1, и тогда можно считать, что *a* – это независимый параметр, определяющий форму кривой. Для получения полной картины достаточно проследить зависимость формы кривой от параметра *a*, изменяя его значения в диапазоне от 0 до 2 с шагом 0,1.

Семейство этих кривых имеет вид, показанный на рисунке, центр которого находится в начале координат. При малых значениях параметра *a* овал Кассами распадается на две замкнутые кривые, близкие к окружностям малых радиусов с центрами в фокусах. С ростом значений параметра эти кривые деформируются и сближаются друг с другом, а потом сливаются в одну замкнутую кривую с центром в начале координат, и при дальнейшем увеличении значений параметра эта кривая по форме постепенно приближается к окружности радиуса *a* с центром в начале координат. Желающие проверить свои силы, могут пока дальше не читать.



Из приведенных формул (и рисунка) видно, что не всем значениям угла будет соответствовать некоторое значение радиуса , а только тем из них, для которых выполнены два условия, ограничивающих область допустимых значений (ОДЗ)

При построении графиков также важно учесть, что при некоторых значениях параметра кривая распадается на две несвязных замкнутых кривых. И поэтому в таких случаях при перемещении пера вдоль кривой нужно каким-то образом избегать ситуации, когда перо перескакивало бы с одной такой ее части на другую (в противном случае, эти перескоки могли бы привести к появлению на графике лишних линий). Понятно, что указанные перескоки могут получаться только при достижении углом границ ОДЗ (при условии, что в рассматриваемый момент используется какая-то одна ветвь графика, определяемая выбором знака на месте символа ). Поэтому возможным решением проблемы будет изменение знака приращения угла при достижении им границы ОДЗ с одновременным переходом на вторую ветвь функции, определяющей радиус (в соответствующей формуле знаки соответствуют как раз этим двум ветвям; при этом необходимо соблюдать соответствие этих знаков в выражении для и в одном из неравенств, определяющих ОДЗ).

Некоторый дополнительный анализ приведенных формул поможет упростить вычисления. А именно, можно заранее выяснить, при каких значениях параметра *a* кривая не распадается на две части, и ее график определяется лишь одним уравнением

, где .

(Ответ: при . Кривая, соответствующая граничному случаю, похожая на «восьмерку», называется лемнискатой Бернулли.)

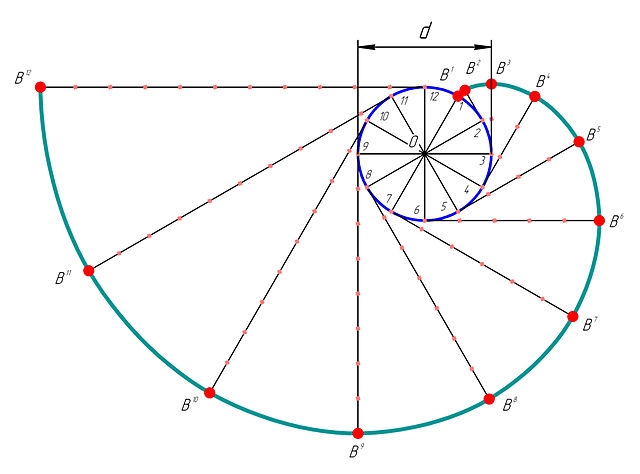
1. Эвольвентой (разверткой) окружности называют плоскую кривую, которую можно построить следующим образом. Представим себе окружность, на которую намотана тонкая нерастяжимая нить (нулевой толщины). Если взять эту нить за конец, и с натяжением начать разматывать (окружность при этом остается неподвижной), то ее конец будет описывать кривую, называемую эвольвентой. Интересно, что профили рабочих частей зубьев зубчатых колес в механизмах для обеспечения идеального сцепления должны иметь вид эвольвенты.

Параметрическое уравнение эвольвенты имеет вид

где r – радиус окружности, – параметр (угол сматывания нити), пробегающий значения от 0 до бесконечности.

Вычертить эвольвенту окружности в виде, поясняющем ее геометрический смысл, подобно тому, как это сделано на рисунке (см. также <https://ru.wikipedia.org/wiki/Эвольвента_окружности>)

.



1. Написать функцию, реализующую алгоритм быстрого возведения в степень (алгоритм см., например, в книге В.В. Борисенко «Основы программирования» <http://www.intuit.ru/studies/courses/2193/67/lecture/1968?page=3>).
2. Написать функцию, возвращающую *n*-ый член последовательности Фибоначчи. Требуется получить сложность алгоритма порядка , и для этого воспользоваться формулой Бине, явно выражающей *n*-ый член последовательности Фибоначчи: (). Из этой формулы следует (почему?), что , где . (В языке Матлаб есть встроенная функция round(), округляющая свой аргумент до ближайшего целого.)

Сравните результаты, получаемые с помощью быстрого алгоритма на основе формулы Бине с результатами, получаемыми с помощью обычного алгоритма на основе рекуррентной формулы: При каких значениях *n* появляются различия? Объясните полученный результат. Каково наибольшее значение *n*, при котором можно рассчитывать на абсолютно точный результат?

См. также <http://e-maxx.ru/algo/fibonacci_numbers>.

1. Написать функцию, реализующую алгоритм решения нелинейных уравнений методом бисекций (алгоритм см., например, в книге В.В. Борисенко «Основы программирования» <http://www.intuit.ru/studies/courses/2193/67/lecture/1968?page=6>)
2. Методом бисекций найти корень уравнения .
3. Методом бисекций найти наименьший по модулю корень уравнения . Построить график зависимости значения этого корня уравнения от параметра *a*.
4. Используя метод бисекций составить программу, возвращающую значение функции, обратной к (из задания 5). Построить график этой обратной функции.
5. Используя метод бисекций, составить программу, возвращающую значение функции, обратной к функции, являющейся сужением функции (из задания 6) на отрезок [0; 1]. Построить график этой обратной функции.
6. Написать функцию, реализующую алгоритм приближенного вычисления логарифма с произвольно заданным основанием без использования степенного ряда (алгоритм см., например, в книге В.В. Борисенко «Основы программирования» <http://www.intuit.ru/studies/courses/2193/67/lecture/1968?page=4>).
7. Если имеется программа, вычисляющая значения некоторой дифференцируемой функции, то для вычисления ее производной можно воспользоваться следующими приближенными формулами:

При переходе к пределу обе дадут правильный результат. Первая из них вытекает непосредственно из определения производной. Однако вторая формула будет приводить к меньшим вычислительным погрешностям, в силу ее симметричности.

Исследовать влияние величины на точность результата. Для этого можно взять, например, и для некоторого заданного значения построить график зависимости от *x* (достаточно на одном периоде) модуля разности между точным значением производной (в данном случае ) и значением производной, вычисленным по приближенной формуле. Значения , следует изменять в пределах, начиная с больших значений, скажем, 0.1 и кончая весьма малыми, например, 1e-16. При этом можно пользоваться любой из двух приближенных формул. Для этого удобно будет написать функцию, которая бы строила указанный график в специально создаваемом новом окне (для создания которого использовать функцию figure), а величину сделать ее входным параметром.

Вопреки ожиданию (быть может), окажется, что существует оптимальное значение величины (где-то порядка 1e-8), обеспечивающее наибольшую точность вычислений. И, что самое удивительное, при достаточно малых значениях могут получаться катастрофически большие погрешности (казалось бы, теория говорит, что должно быть наоборот). Попытайтесь объяснить полученный результат, связанный с особенностями машинной арифметики с плавающей точкой.

Тут заданий с избытком (в первую очередь, это задания 11-15), для того чтобы дать поработать сильным студентам. Причем и сильные студенты могут не успеть сделать все. Поэтому можно разным студентам давать разные задания, с тем, чтобы потом они могли продемонстрировать свои результаты товарищам. Количество однотипных пунктов (А, Б, В…) в некоторых заданиях может быть уменьшено или увеличено (в зависимости от уровня студентов: чем ниже уровень, тем больше однотипных пунктов).

На будущее все задания надо будет еще перегруппировать:

- постороение кривых (+ гипоциклоида, гиперциклоида, лемниската Бернули)

- анимации (+ добавить еще анимацию построения улитки Паскаля при некоторых параметрах: подеры окружности и трисектриссы)

- криволинейные координатные сетки (+ добавить еще биполярные координаты); тут можно добавить еще такие задания повышенной сложности: отметить маленькими кружками узлы координатной сетки, раскрасить клетки координатной сетки в шахматном порядке

- частичные суммы рядов Фурье

- частичные суммы степенных рядов

- быстрое возведение в степень

- метод бисекций

- вычисление логарифма

- исследование простых формул численного дифференцирования