Лабораторная работа №3

Численные методы

Помимо аппарата линейной алгебры, Matlab включает в себя множество библиотек, предназначенных для решения нелинейных задач. С некоторыми из них мы познакомимся в этой лабораторной работе.

Практически любой численный алгоритм можно описать следующим образом: `y = F(f,x0, options)`, где `f` — функция, «порождающая» задачу (которую интегрируют, чей ноль ищут, которая задаёт правую часть системы уравнений, ...), `x0` — начальное приближение, `y` — выходное значение (или сетка выходных значений), `options` — внутренние параметры алгоритма (максимальное число итераций, минимальная допустимая точность, ...). Поскольку в Matlab, как вы уже знаете, функции могут выступать в роли переменных, практически все алгоритмы его библиотек имеют такой простой синтаксис вызова. Обычно переменная, отвечающая за параметры, является структурой (для наших целей структуры в Matlab ни чем не отличаются от структур в C/C++).

Эта лабораторная работа посвящена решению задач, с которыми вы сталкивались в процессе обучения на первых двух курсах. В процессе её выполнения рекомендуется освежить знания по этим курсам.

Напомним несколько важных принципов использования численных методов на практике.

1. В большинстве теорем о сходимости алгоритмов к «истинным» решениям (например, метод Ньютона поиска корней или метод Зейделя решения систем линейных уравнений) на задачу накладывается ряд предположений, проверка которых не всегда возможна на практике или может является нецелесообразной. Так, например, обычно оценки на погрешность метода интегрирования выводятся через максимум модуля одной из производных функции. На практике поиск этой производной (и потом — её максимума) может оказаться задачей гораздо более трудоёмкой, чем сам процесс интегрирования. Поэтому на практике, увы, мы не всегда можем гарантировать правильность решений за счёт выполнение условий теорем; с другой стороны, поскольку эти предположения явно не влияют на сам алгоритм, мы всегда можем поставить алгоритм на счёт и получить какой-то результат, который может претендовать на то, чтобы быть правильным[1]. Этот факт имеет два важных последствия: во-первых, результат работы алгоритма надо всегда проверять на правильность в том или ином смысле (например, если алгоритм поиска корней уравнения $f(x) = y$ с известной функцией f выдал корень z, то величина $|f(z) - y|$ будет говорить нам о величине ошибки), и, во-вторых, очень важны те случаи, когда задача, которую решает алгоритм, имеет аналитическое решение: на таких примерах мы можем многое понять о свойствах алгоритма.
2. Вычислительные системы (в отличие от, например, систем символьной алгебры) *всегда* хранят дробные и иррациональные числа в обрезанном виде. Это приводит к тому, что значения, которые «на бумаге» должны совпадать, на практике могут отличаться. Так, в Matlab 2013 выражение `sqrt(2) == 8^(1/6)` равно нулю (т.е. логическому «ложь»), в то время как `|sqrt(2) - 8^(1/6)|` равен 2.2204×10^{-16}, очень маленькому, но ненулевому числу. При этом не стоит безусловно отождествлять «маленькие значения» и отсутствие каких-либо свойств у объектов как таковых. Так, в теории, вырожденность матрицы эквивалентна равенству нулю её детерминанта, поэтому если численно детерминант «маленький», то естественно ожидать больших погрешностей. Однако, скажем, у матрицы с элементами

$$H_{ij} = (-1)^{i+j}(i+j-1)C_{n+i-1}^{n-j}C_{n-i}^{n+j-1}\left(C_{i-1}^{i+j-2}\right)^2,\ i,j = 1,\dots,n,$$

   определитель весьма далёк от нуля, однако для случайной правой части b уже при $n = 9$ величина `norm(H*inv(H)*b - b)` может быть весьма значительной[2]. В качестве необязательного задания читателю предлагается подумать и дать объяснение этому эффекту.
3. Важно понимать, что многие алгоритмы являются по сути своей *локальными*. Так, если у функции есть несколько корней, то конкретный корень, к которому сойдётся алгоритм (и сойдётся ли он вообще), обычно *существенно* зависит от начального приближения. Поэтому часто на практике имеет смысл выделить поиск начального приближения в отдельную задачу.

Некоторые полезные команды для этой лабораторной работы.

Решение уравнений: `fzero`, `fsolve`

Численное интегрирование и дифференцирование: `diff`, `trapz`, `ode45`

Визуализация векторных полей: `quiver`

Для выполнения задания 4 рекомендуется почитать статью в руководстве про Event Handling (механизм прерывания процесса интегрирования до окончания промежутка интегрирования).

[1] Большинство встроенных алгоритмов Matlab замечают «странные» вещи, возникающие при расчёте, вроде отсутствия сходимости, и предупреждают выводом в консоль.

[2] Дело здесь не только в использовании `inv`. При использовании более продвинутых методов эффект по-прежнему будет иметь место, но при больших размерностях