Практическая работа по курсу "Численные методы"

Худякова Екатерина, группа 209, ВМК МГУ

Постановка задачи

Найти решение задачи Коши

$\dfrac{du}{dx} = \displaystyle\int_0^x f(t)\,dt$, $u = u(x)$, $0 < x < l$

$u(0) = u_0$

методом Рунге – Кутта второго порядка

Функция $f(t)$ задана и может быть найдена как в точках сетки

$x_i = ih$, $i = 0, 1, \ldots, N$, $h = \dfrac{l}{N}$, так и в любой точке отрезка $[0, l]$

а) Исследовать поведение решения на сгущающихся сетках (при увеличении N).

б) Выяснить, будет ли сходимость.

Вариант задачи

Функция $f(t) = e^{-t^2}$

отрезок $[0, 1]$, начальное $N = 20$

$u(x)$ можно для проверки вычислять численным интегрированием. Для этого введем функцию

$g(x) = \int_0^x e^{-t^2}\,dt = \frac{\sqrt{\pi}}{2} erf(x)$

Тогда

$u(x_0) = \displaystyle\int_0^{x_0} g(x)\,dx$ (для $u(0) = 0$)

Используемые формулы и алгоритмы

Функция $g_1(x, u) = g(x)$ непрерывна и удовлетворяет условию Липшица по аргументу u в любой

окрестности начальной точки (поскольку функция от u не зависит). Следовательно, можно

указать отрезок $[a, b]$, $a < x_0 < b$, на котором решение вышеописанной задачи

существует и единственно.

Для интегрирования используется **метод трапеций**. Формула для равномерной сетки:

$$\int_a^b f(x)\,dx \approx h\left(\frac{f_0+f_n}{2}+\sum_{i=1}^{n-1} f_i\right)$$

Применим к $f(x)$, $g_1(x,u)$, так как они интегрируемы (как непрерывные).

Реккурентное соотношение для метода Рунге-Кутта:

$$y_{i+1} = y_i + h\left[(1-\alpha)g(x_i, y_i) + \alpha g\left(x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha}g(x_i, y_i)\right)\right]$$

Поскольку $g(x,u)$ в данном случае не зависит от u, формула может быть записана как

$$y_{i+1} = y_i + h\left[(1-\alpha)g(x_i) + \alpha g\left(x_i + \frac{h}{2\alpha}\right)\right]$$

Было исследовано 2 разностные схемы:

1. $\alpha = 0.5$ ("предиктор-корректор")

Фактическая формула:

$$y_{i+1} = y_i + h\frac{g(x_i) + g(x_{i+1})}{2}$$

2. $\alpha = 1$

Фактическая формула:

$$y_{i+1} = y_i + hg\left(x_i + \frac{h}{2}\right)$$

Для исследования было решено взять следующее количество точек (N): 10, 20, 40, 80, 160.

Программная реализация

In [1]:

```
from math import *
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import random

# параметры для графиков
plt.rcParams['figure.figsize'] = [10, 10]
plt.rcParams['font.size'] = '14'

# точность с которой считаю интеграл
eps = 0.000001

# функция f(t)
def f(t):
    return(exp(-t*t))

# Вычисление интеграла по формуле трапеций
def integral(x, op):
    n = 1
    h = x / n
    sum_tek = 0.0
    sum_next = 0.0
    for i in range(n):
        sum_next = sum_next + op(i * h)
```

```python
    sum_next = sum_next - op(0)/2 - op(x)/2
    while(1):
        sum_tek = sum_next
        h = h / 2;
        for i in range(n):
            sum_next += op(h * (i * 2 + 1))
        n = n * 2
        if(fabs(h * 2 * sum_tek - h * sum_next) < eps):
            break
    return(h * sum_next)

# функция g(x)
def g(x):
    return(integral(x, f))

# funct, и используются для проверки и расчета погрешности
# funct(x) - аналитически заданная g(x)
# u(x) - вычисляемая методом трапеций искомая функция

def funct(x):
    return(sqrt(pi) * erf(x) / 2)

def u(x):
    return(integral(x, funct))

# вычисление медодом Рунге-Кутта, параметры: a, b - начало и конец отрезка,
# N - число точек, arr - массив для записи результатов, function - функция
def rungekutta(a, b, N, alpha, arr, op):
    arr[0] = 0.0
    h = (b - a) / N
    for i in range(1, N + 1):
        inter = op(a + h * (i - 1))
        arr[i] = arr[i-1] + ((1-alpha)*inter + alpha*op(a+h*(i-1+1/2/alpha)))*h
```

In [2]:

```python
# на скольких h рассматривается метод
M = 5
# на сколько умножается начальное значение точек
mnozh = [1, 2, 4, 8, 16]
# начальное значение точек
N = 10

# массив с количеством точек
num = [N * mnozh[i] for i in range(M)]
# массив с шагом
h = [(1.0 / num[i]) for i in range(M)]
# массив для результатов рассчетов
mas = [np.zeros((2, num[i] + 1)) for i in range(M)]

# итоговый массив по N + 1 точек для разных h и alpha
res = np.zeros((M, 2, N + 1))
# фактические значения u(x)
u_res = np.zeros(N + 1)
# значения для сетки
x_res = np.zeros(N + 1)
# ошибка (считаю как максимальное отклонение)
delta = np.zeros((M,2))

# считаем для разного количества точек и разных alpha
for i in range(M):
    rungekutta(0, 1.0, num[i], 1.0, mas[i][0], g)
    rungekutta(0, 1.0, num[i], 0.5, mas[i][1], g)

# заполняем массив x
for i in range(N + 1):
    x_res[i] = i * h[0]
```

```python
    # считаем значения u(x)
    for i in range(N + 1):
        u_res[i] = u(x_res[i])
    # заполняем массив результатов
    for i in range(M):
        for j in range(2):
            for e in range(N + 1):
                res[i][j][e] = mas[i][j][e * (num[i] // N)]
```

Цифровое и графическое представление результатов

In [3]:

```python
#Печать таблицы

# BOLD GREEN COLOR  "\033[1;32m"
# BOLD YELLOW COLOR "\033[1;33m"
# STANDARD COLOR    "\x1b[0m"

def printing():
    for i in range(13 + M * 16):
        print("-", end = '')
    print()

# Вывод таблицы значений

printing()
print("|   ", end = '')
for i in range(M):
    print("|N = {:11d}".format(num[i]), end = '')
print("|       |")
printing()
print("| x ", end = '')
for i in range(M):
    print("| a = 1 |a = 0.5", end = '')
print("|   u   |")
for i in range(N + 1):
    printing()
    print("|", end = '')
    print("{:3.1f}".format(x_res[i]), end = '')
    print("|", end = '')
    for j in range(M):
        for e in range(2):
            if(num[j] == 20):
                print("\033[1;32m{:7.5f}\x1b[0m".format(res[j][e][i]), end = '')
            else:
                print("\033[1;33m{:7.5f}\x1b[0m".format(res[j][e][i]), end = '')
            print("|", end = '')
            if(abs(u_res[i] - res[j][e][i]) > delta[j][e]):
                delta[j][e] = abs(u_res[i] - res[j][e][i])
    print("{:7.5f}".format(u_res[i]), end = '')
    print("|")
printing()
```

```
---------------------------------------------------------------------------------------------
|   |N =          10|N =          20|N =          40|N =          80|N =         160|       |
---------------------------------------------------------------------------------------------
| x | a = 1 |a = 0.5| a = 1 |a = 0.5| a = 1 |a = 0.5| a = 1 |a = 0.5| a = 1 |a = 0.5|   u   |
---------------------------------------------------------------------------------------------
|0.0|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|0.00000|
---------------------------------------------------------------------------------------------
|0.1|0.00500|0.00498|0.00499|0.00499|0.00499|0.00499|0.00499|0.00499|0.00499|0.00499|0.00499|
---------------------------------------------------------------------------------------------
|0.2|0.01988|0.01983|0.01987|0.01986|0.01987|0.01987|0.01987|0.01987|0.01987|0.01987|0.01987|
---------------------------------------------------------------------------------------------
|0.3|0.04437|0.04426|0.04435|0.04432|0.04434|0.04433|0.04434|0.04434|0.04434|0.04434|0.04434|
---------------------------------------------------------------------------------------------
```

```
|0.4|0.07799|0.07781|0.07795|0.07790|0.07794|0.07793|0.07793|0.07793|0.07793|0.07793|0.07793|
---------------------------------------------------------------------------------------------
|0.5|0.12013|0.11986|0.12006|0.11999|0.12005|0.12003|0.12004|0.12004|0.12004|0.12004|0.12004|
---------------------------------------------------------------------------------------------
|0.6|0.17006|0.16968|0.16996|0.16987|0.16994|0.16991|0.16993|0.16993|0.16993|0.16993|0.16993|
---------------------------------------------------------------------------------------------
|0.7|0.22695|0.22647|0.22683|0.22671|0.22680|0.22677|0.22680|0.22679|0.22679|0.22679|0.22679|
---------------------------------------------------------------------------------------------
|0.8|0.28998|0.28939|0.28983|0.28968|0.28979|0.28976|0.28978|0.28978|0.28978|0.28978|0.28978|
---------------------------------------------------------------------------------------------
|0.9|0.35828|0.35758|0.35810|0.35793|0.35806|0.35802|0.35805|0.35804|0.35805|0.35804|0.35805|
---------------------------------------------------------------------------------------------
|1.0|0.43103|0.43024|0.43083|0.43063|0.43078|0.43073|0.43077|0.43075|0.43076|0.43076|0.43076|
---------------------------------------------------------------------------------------------
```

Сравнение результатов с предполагаемой функцией показывает, что

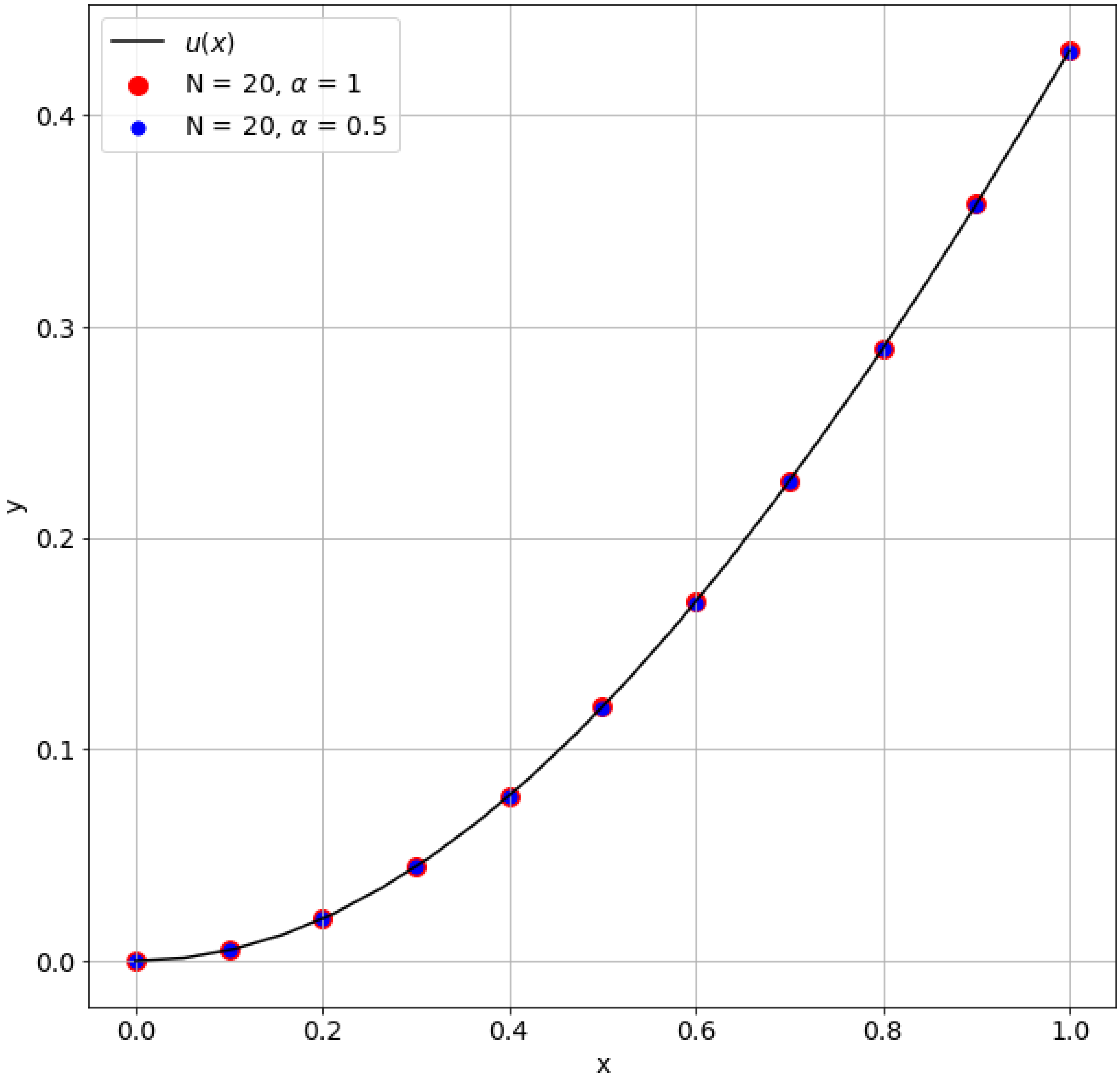
при N = 20 значения вычисляются достаточно точно:

In [4]:

```python
# график функции u(x)
vis_u = np.vectorize(u)
x=np.linspace(0,1,20)
plt.plot(x, vis_u(x), c = 'black', label = '$u(x)$')
for i in range(M):
    if(num[i] == 20):
        plt.scatter(x_res, res[0, 0], c = 'r', s = 100, label = r'N = 20, $\alpha$ = 1')
        plt.scatter(x_res, res[0, 1], c = 'b', s = 50, label = r'N = 20, $\alpha$ = 0.5')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
```

Out[4]:

```
<matplotlib.legend.Legend at 0x7f178b926e50>
```



Ошибка $\Delta_i = |z_i| = |y_i - u_i|$ (где y_i вычисляется методом Рунге-Кутта, u_i методом трапеций)

отрицательно коррелирует с шагом h (т.е. чем больше точек, тем точнее расчет и меньше погрешность).

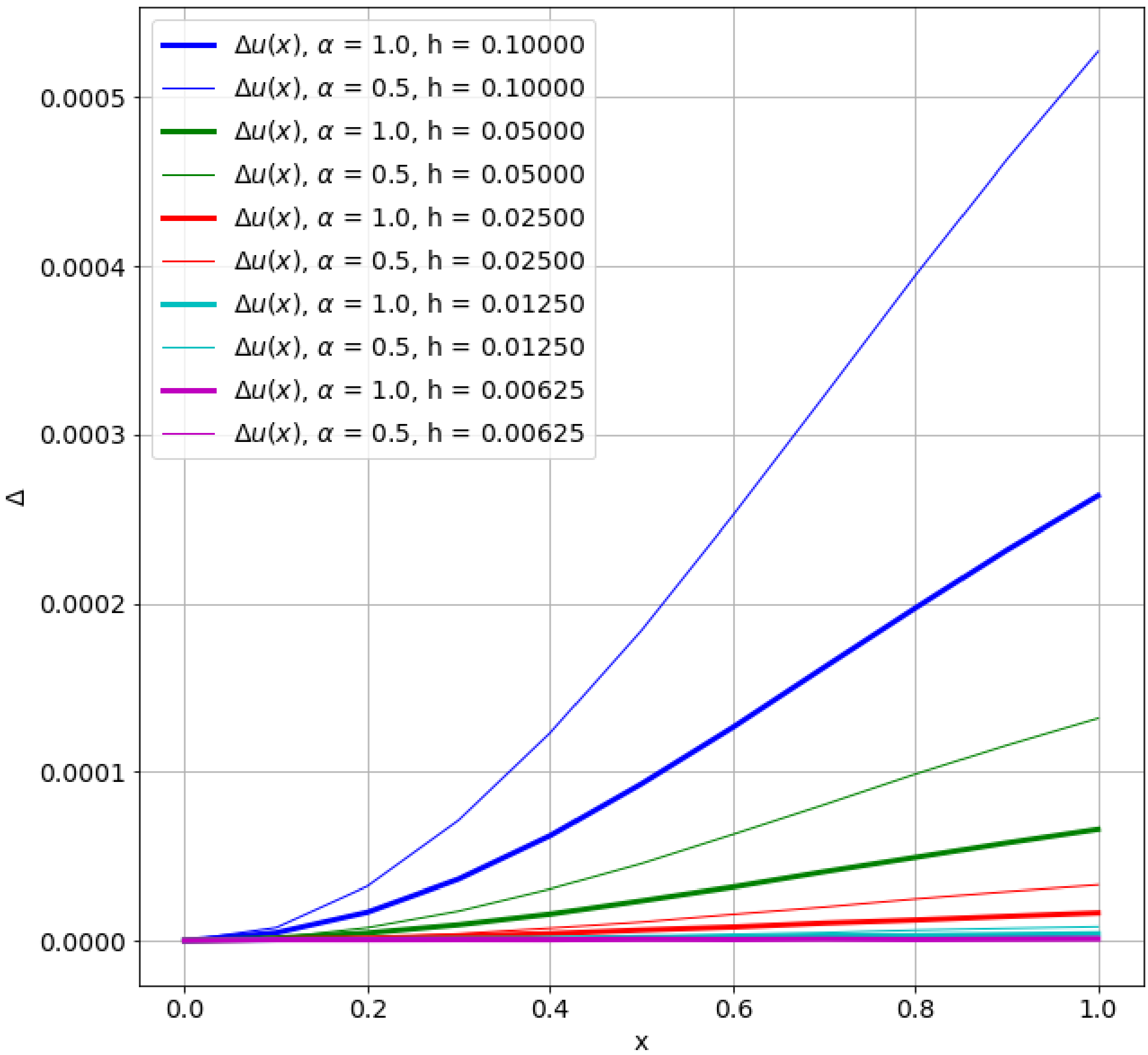
In [5]:

```
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k']
x=np.linspace(0,1,10)

for i in range(M):
    plt.plot(x_res, np.abs(res[i,0] - u_res) , c = colors[i], linewidth = 3,
             label = r'$\Delta u(x)$, $\alpha$ = 1.0, h = {:7.5f}'.format(h[i]))
    plt.plot(x_res, np.abs(res[i,1] - u_res) , c = colors[i], linewidth = 1,
             label = r'$\Delta u(x)$, $\alpha$ = 0.5, h = {:7.5f}'.format(h[i]))

plt.grid()
plt.xlabel('x')
plt.ylabel(r'$\Delta$')
plt.legend()
```

Out[5]: `<matplotlib.legend.Legend at 0x7f178ab5afd0>`



Норма $\|\Delta\|$ соответствует ожидаемой оценке $\|\Delta\ \| \leq Mlh^2e^{Cl}$

где $\left|\frac{\partial f}{\partial u}(x,u)\right| \leq C$ (для расчета взята C = 0)

$$\left|\frac{1}{6}u''(x_i) - \frac{1}{8\alpha}\left[\frac{\partial^2 f}{\partial x^2}(x_i,u_i) + 2\frac{\partial^2 f}{\partial x \partial u}(x_i,u_i)f(x_i,u_i) + \frac{\partial^2 f}{\partial u^2}(x_i,u_i)f^2(x_i,u_i)\right]\right| =$$

$$= \left|\left(\frac{1}{6} - \frac{1}{8\alpha}\right)u''(x_i)\right| \leq M$$

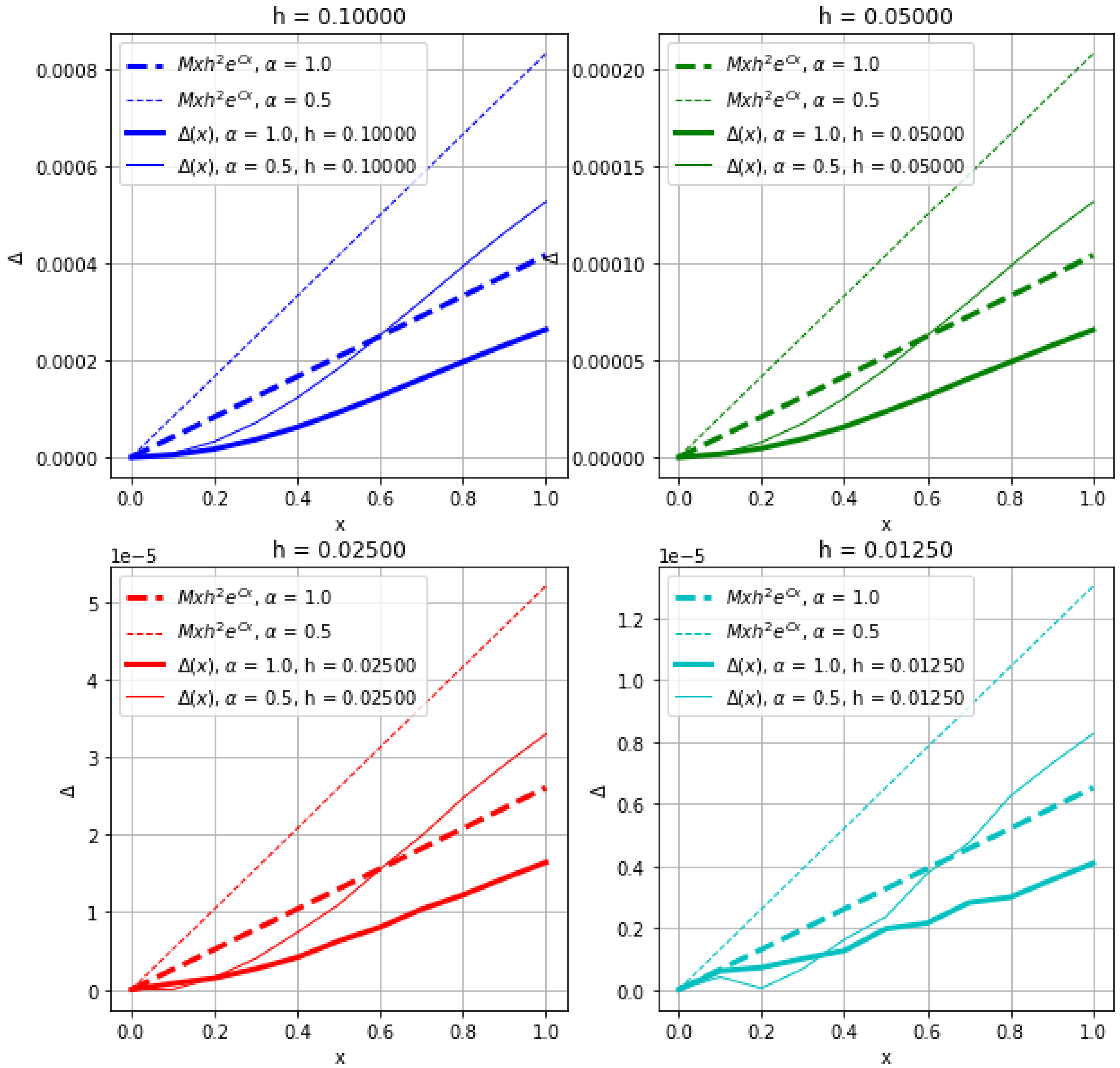
для расчета принято что $u''(x_i) = e^{-x_i^2} \leq 1.0$

l - длина отрезка

In [6]:
```
plt.rcParams['font.size'] = '10'
x=np.linspace(0,1,10)
def O(x, h, alpha):
    return(1.0 * x * h ** 2 * abs((1/6 - 1/8/alpha)))

vis_O = np.vectorize(O)
#vis_O_0_5 = np.vectorize(O_0_5)
for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.plot(x, vis_O(x, h[i], 1.0), c = colors[i],
             label = r'$Mxh^2e^{Cx}$, $\alpha$ = 1.0', linewidth = 3, linestyle = '--')
```

```
    plt.plot(x, vis_O(x, h[i], 0.5), c = colors[i],
             label = r'$Mxh^2e^{Cx}$, $\alpha$ = 0.5', linewidth = 1, linestyle = '--')
    plt.plot(x_res, np.abs(res[i,0] - u_res) , c = colors[i], linewidth = 3,
             label = r'$\Delta(x)$, $\alpha$ = 1.0, h = {:7.5f}'.format(h[i]))
    plt.plot(x_res, np.abs(res[i,1] - u_res) , c = colors[i], linewidth = 1,
             label = r'$\Delta(x)$, $\alpha$ = 0.5, h = {:7.5f}'.format(h[i]))
    plt.grid()
    plt.title(r'h = {:7.5f}'.format(h[i]))
    plt.xlabel('x')
    plt.ylabel(r'$\Delta$')
    plt.legend()
plt.rcParams['font.size'] = '14'
```



На сгущающихся сетках погрешность стремится к нулю, то есть показано, что данный метод сходится.

При этом погрешность $||\Delta||$, как видно из графика ниже, стремится к 0 со скоростью не более h^2,

что соответствует оценке выше.

Коэффициент при расчетах в данном случае подбирается по последней точке.

In [7]:

```
x=np.linspace(0,0.105,50)

def k1(t):
    return(delta[0][0] / (h[0] ** t))
def k2(t):
    return(delta[0][1] / (h[0] ** t))
```

```python
plt.plot(x, k1(1) * x, c = 'r', linestyle = '--',
         label = r'$k_1h$, $\alpha$ = 1')
plt.plot(x, k2(1) * x, c = 'b', linestyle = '--',
         label = r'$k_1h$, $\alpha$ = 0.5')
plt.plot(x, k1(2) * x ** 2, c = 'r',  linewidth = 3,
         label = r'$k_1h^2$, $\alpha$ = 1')
plt.plot(x, k2(2) * x ** 2, c = 'b',  linewidth = 3,
         label = r'$k_1h^2$, $\alpha$ = 0.5')
plt.plot(x, k1(3) * x ** 3, c = 'r', linestyle = '-.',
         label = r'$k_1h^3$, $\alpha$ = 1')
plt.plot(x, k2(3) * x ** 3, c = 'b', linestyle = '-.',
         label = r'$k_1h^3$, $\alpha$ = 0.5')

plt.scatter(h[:], delta[:, 0], c = 'red', s = 50,
            label = r'$\Delta(h)$, $\alpha$ = 1')
plt.scatter(h[:], delta[:, 1], c = 'blue', s = 50,
            label = r'$\Delta(h)$, $\alpha$ = 0.5')

plt.grid()
plt.xlabel('h')
plt.ylabel(r'$\Delta$')
plt.legend()
```

Out[7]: <matplotlib.legend.Legend at 0x7f178ac1e3a0>