Лабораторная работа №6

«Метод градиентного спуска»

выполнил Пажитных Иван, 2-й курс, 1-я группа

1) Постановка задачи

Необходимо найти решение системы линейных алгебраических уравнений вида $Ax = b$, где А- квадратная матрица n-ого порядка, x и b – столбцы размеров n×1.

$$\begin{bmatrix} a_{11}\ a_{12}\ a_{13} & \cdots & a_{1n} & |\ b_1 \\ a_{21}\ a_{22} a_{23} & \dots & & |b_2 \\ \vdots & \ddots & & \vdots \\ a_{n1}\ a_{n2} a_{n3} & \cdots & a_{nn} & |\ b_n \end{bmatrix}$$

Предполагается, что det A=|A|≠0, A – симметрическая и положительно определённая ($A*A^T>0$). Тогда решение системы существует и оно единственно и по Теореме о сходимости метода градиентного спуска, он будет сходящимся.

2)Алгоритм решения

В методе градиентного спуска нахождение решения системы $Ax = b$ связано с задачей минимизации квадратичного функционала $F(x) = (Ax, x) - 2(b, x)$. Метод градиентного спуска состоит в последовательном вычислении значения вектора x^k:

Положим начальное приближение $x^0=b$. Далее будем вычислять x^k по итерационной формуле $x^k = x^{k-1} - \tau_k r^{k-1}$, где $r^{k-1} = Ax^{k-1} - b$, $\tau_k = \frac{(r^{k-1}, r^{k-1})}{(Ar^{k-1}, r^{k-1})}$, до тех пор, пока не выполнится условие: $||x^{k+1} - x^k|| \leq \varepsilon$.

3) Листинг программы

```
E = np.identity(n)
eps = 10 ** (-5)
a = np.array(A)
At = a.transpose()                          # находим At
b = np.array(f).transpose()

a = np.dot(At, a)                    # перемножаем A на At, теперь A - симметрическая
b = np.dot(At, b)                    # то же самое с b

xk = b                               # начальное приближение
x = zeros(n)
k = 0

while True:                                                  # итерационный процесс
   rk = np.dot(a, xk) - b                                    # считаем rk
   x = xk - np.dot(rk, np.dot(rk, rk) / np.dot(np.dot(a, rk), rk))
   if abs(np.linalg.norm(x, inf) - np.linalg.norm(xk, inf)) < eps:    # условие окончания итерации
      break
   xk = x
   k += 1

r = np.dot(A, x) - f                                         # находим вектор невязки
rnorm = np.linalg.norm(r, 1)                                 # находим норму невязки
```

4) Результат и его анализ

Матрица коэффициентов A:

```
[[ 0.6897 -0.0908  0.0182  0.0363  0.1271]
 [ 0.0944  1.0799  0.     -0.0726  0.0726]
 [ 0.0545  0.      0.8676 -0.2541  0.1452]
 [-0.1089  0.2287  0.      0.8531 -0.0363]
 [ 0.4538  0.      0.1634  0.0182  1.0164]]
```

Столбец свободных членов b:

```
[ 4.2108  4.6174 -5.877   2.7842  0.2178]
```

Симметрическая AA^T:

```
[[ 0.70536135  0.01441237  0.13398766 -0.08030921  0.5676231 ]
 [ 0.01441237  1.22673234 -0.00165256  0.11340719  0.05855825]
 [ 0.13398766 -0.00165256  0.77976056 -0.21682262  0.2943685 ]
 [-0.08030921  0.11340719 -0.21682262  0.79926611 -0.0500214 ]
 [ 0.5676231   0.05855825  0.2943685  -0.0500214   1.07689486]]
```

Столбец свободных членов bA^T:

```
[ 2.81541308  5.24073616 -4.98666012  3.69013948  0.13738098]
```

Вектор решений x:

```
[ 7.00092471  3.99994451 -6.00024734  2.99987223 -2.00056371]
```

Вектор невязки r:

```
[ -3.79741772e-05  -4.27900748e-06  -1.35803157e-05  -1.92819049e-06
  3.93901948e-06]
```

Норма $||r||$:

6.17007103671e-05

Количество итераций k:

31

Эпсилон ε:

1e-05

Метод градиентного спуска сходится, так как A – симметрическая и положительно определённая. Точность решения и скорость сходимости зависят от эпсилона. Сравнивая с МПИ метод градиентного спуска даёт чуть худшую точность (невязка порядка 10^{-5} против 10^{-6}) и меньшую скорость сходимости (31 итерация против 10). Для увеличения точности решения необходимо задать меньший эпсилон.