Расчетно-графическая работа №2

Фролова Алена, МХ-301

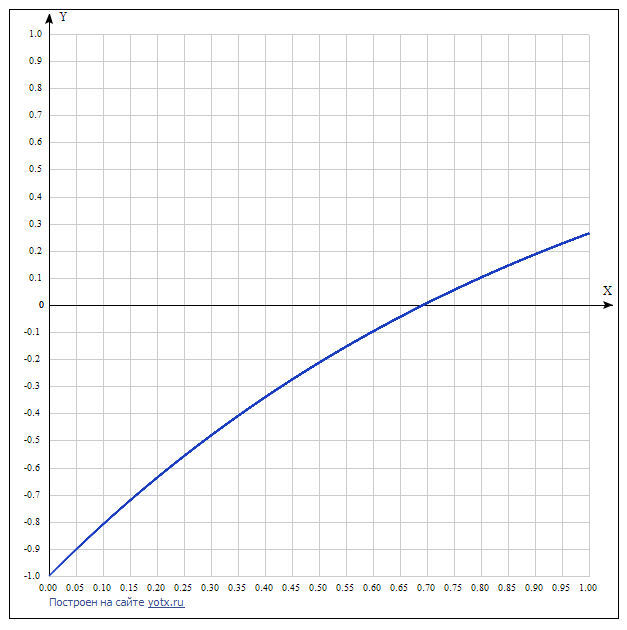
Вариант 8.

11 апреля 2017г.

Постановка задачи:

Решить задачу Коши: На отрезке [0,1] : , с шагом h=0.1, h=0.05.  
Необходимо решить явным методом Эйлера, методом Рунге-Кутты четвертого порядка и методом Симпсона.

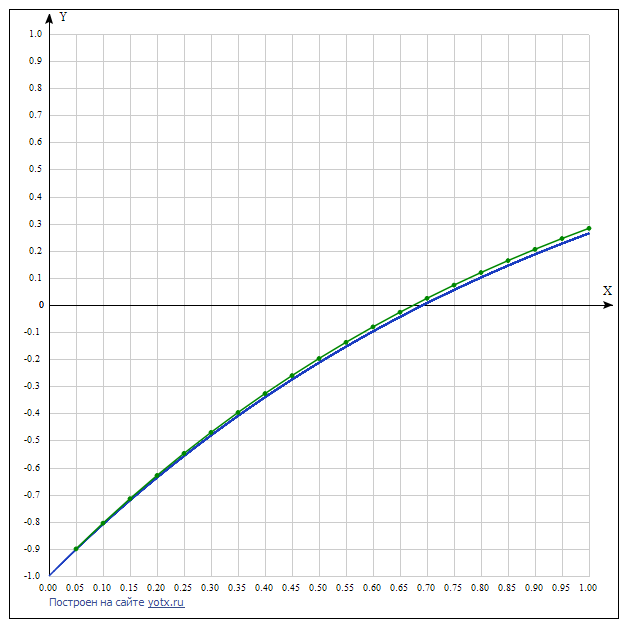
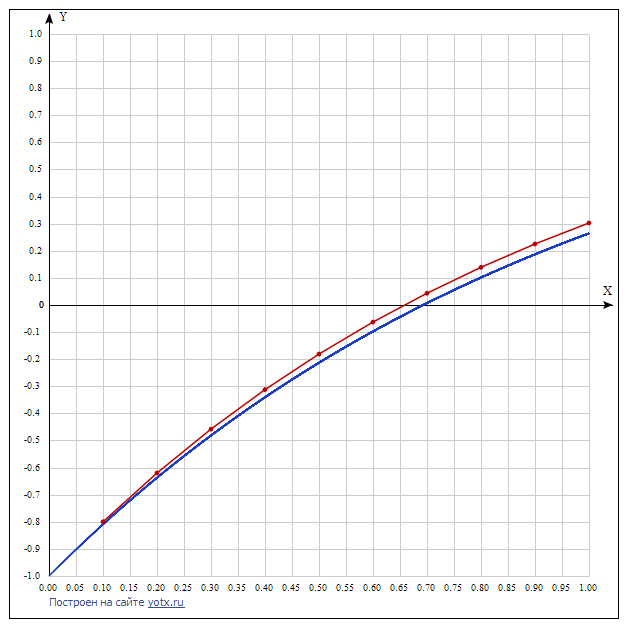
Точное решение:



Решение явным методом Эйлера:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.1* | *0.2* | *0.3* | *0.4* | *0.5* |
|  | *-1* | *-0.8* | *-0.62* | *-0.458* | *-0.3122* | *-0.18098* |
|  | *0.6* | *0.7* | *0.8* | *0.9* | *1* |
|  | *-0.062882* | *0.043406* | *0.139066* | *0.225159* | *0.302643* |

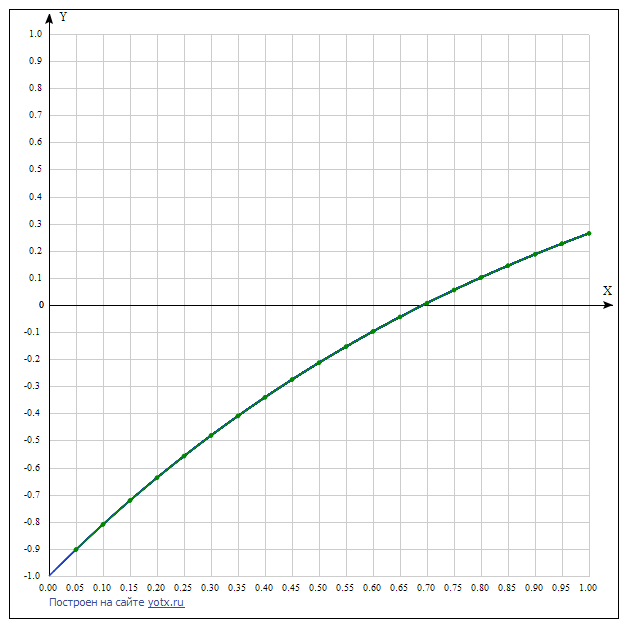
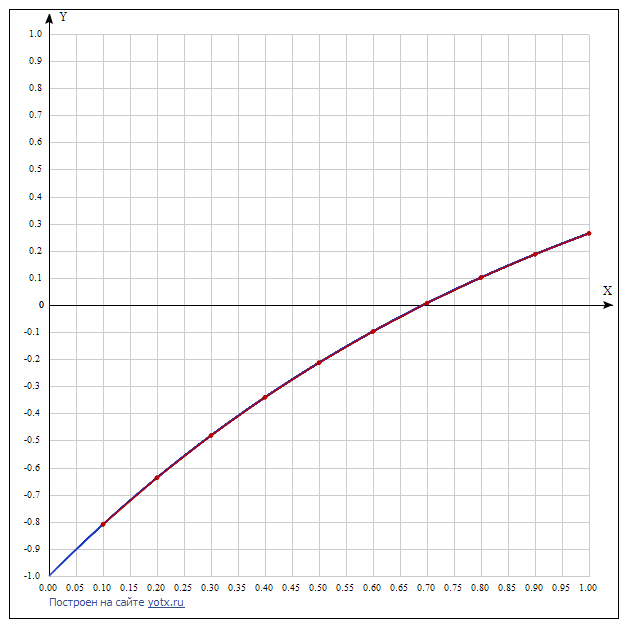
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.05* | *0.1* | *0.15* | *0.2* | *0.25* | *0.3* |
|  | *-1* | *-0.9* | *-0.805* | *-0.71475* | *-0.629013* | *-0.547562* | *-0.470184* |
|  | *0.35* | *0.4* | *0.45* | *0.5* | *0.55* | *0.6* | *0.65* |
|  | *-0.396675* | *-0.32684* | *-0.26049* | *-0.197474* | *-0.1376* | *-0.08072* | *-0.026684* |
|  | *0.7* | *0.75* | *0.8* | *0.85* | *0.9* | *0.95* | *1* |
|  | *0.02465* | *0.073418* | *0.119747* | *0.163759* | *0.205571* | *0.245293* | *0.283028* |

**

Решение методом Рунге-Кутты четвертого порядка:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.1* | *0.2* | *0.3* | *0.4* | *0.5* |
|  | *-1* | *-0.809675* | *-0.637462* | *-0.481637* | *-0.340641* | *-0.213062* |
|  | *0.6* | *0.7* | *0.8* | *0.9* | *1* |
|  | *-0.097624* | *0.006829* | *0.101341* | *0.186860* | *0.264240* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.05* | *0.1* | *0.15* | *0.2* | *0.25* | *0.3* |
|  | *-1* | *-0.902459* | *-0.809675* | *-0.721416* | *-0.637462* | *-0.557602* | *-0.481636* |
|  | *0.35* | *0.4* | *0.45* | *0.5* | *0.55* | *0.6* | *0.65* |
|  | *-0.409376* | *-0.34064* | *-0.275256* | *-0.213061* | *-0.1539* | *-0.097623* | *-0.044092* |
|  | *0.7* | *0.75* | *0.8* | *0.85* | *0.9* | *0.95* | *1* |
|  | *0.006829* | *0.055267* | *0.101342* | *0.14517* | *0.186861* | *0.226518* | *0.264241* |

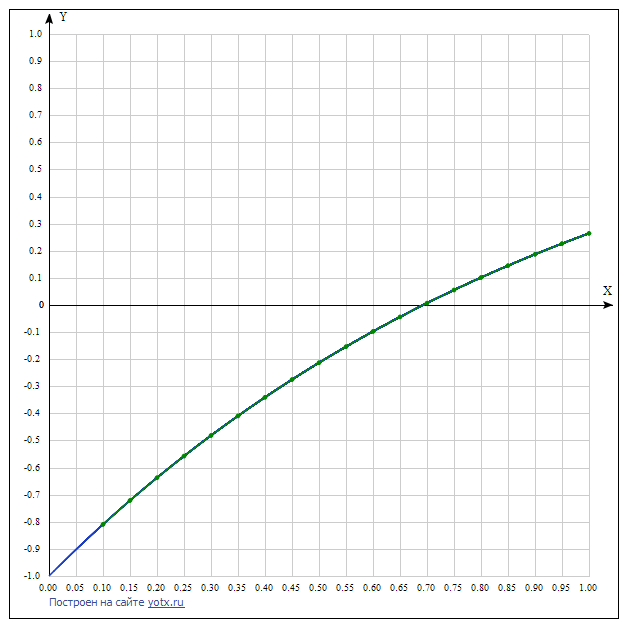
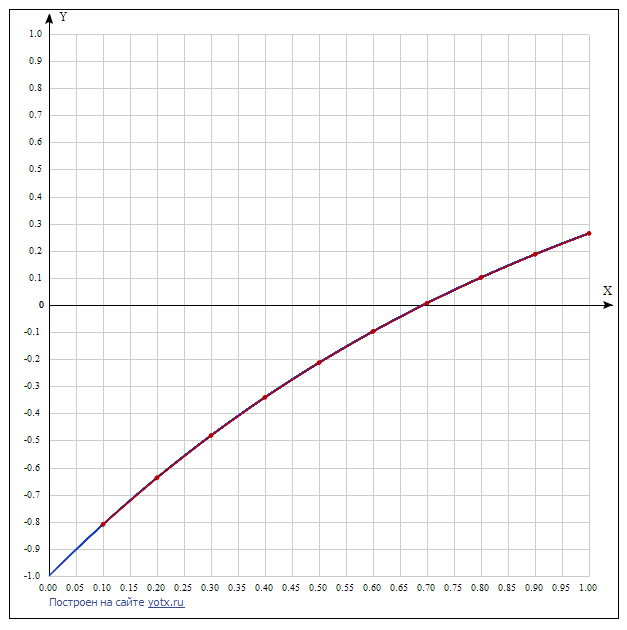


Решение методом Симпсона:

Будем проводить разгонку методом Рунге-Кутты четвертого порядка.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.1* | *0.2* | *0.3* | *0.4* | *0.5* |
|  | *-1* | *-0.809675* | *-0.637461* | *-0.481636* | *-0.34064* | *-0.213061* |
|  | *0.6* | *0.7* | *0.8* | *0.9* | *1* |
|  | *-0.097623* | *0.00683* | *0.101343* | *0.186861* | *0.264242* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *0.05* | *0.1* | *0.15* | *0.2* | *0.25* | *0.3* |
|  | *-1* | *-0.902459* | *-0.809675* | *-0.721416* | *-0.637461* | *-0.557602* | *-0.481636* |
|  | *0.35* | *0.4* | *0.45* | *0.5* | *0.55* | *0.6* | *0.65* |
|  | *-0.409376* | *-0.34064* | *-0.275256* | *-0.213061* | *-0.1539* | *-0.097623* | *-0.044092* |
|  | *0.7* | *0.75* | *0.8* | *0.85* | *0.9* | *0.95* | *1* |
|  | *0.006829* | *0.055267* | *0.101342* | *0.14517* | *0.186861* | *0.226518* | *0.264241* |



Вывод: как и ожидалось методы Рунге-Кутты четвертого порядка и Симпсона сходятся быстрее, так как при пересчете следующих значений учитываются предыдущие шаги.

Приложение:

**static** IEnumerable<(**double** x, **double** y)> ExplicitEulerMethod(**double** y0, **double**[] segment, **double** h)

{

**double** GetNextY(**double** y) => y + (-y + 1) \* h;

var a = segment[0];

var b = segment[1];

var currentX = a;

var currentY = y0;

**while** (Math.Abs(currentX - b) > 1e-5)

{

currentX += h;

currentY = GetNextY(currentY);

**yield** return (currentX, currentY);

}

}

**static** IEnumerable<(**double** x, **double** y)> RungeKuttaMethods(**double** y0, **double**[] segment, **double** h)

{

**double** F(**double** y) => -y + 1;

var a = segment[0];

var b = segment[1];

var currentX = a;

var currentY = y0;

**while** (Math.Abs(currentX - b) > 1e-5)

{

var k1 = F(currentY);

var k2 = F(currentY + h / 2 \* k1);

var k3 = F(currentY + h / 2 \* k2);

var k4 = F(currentY + h \* k3);

currentX += h;

currentY = currentY + h / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4);

**yield** return (currentX, currentY);

}

}

**static** IEnumerable<(**double** x, **double** y)> SimpsonsRule(**double** y0, **double** y1, **double**[] segment, **double** h)

{

var a = segment[0];

var b = segment[1];

var currentX = a + h;

var previousY = y0;

var currentY = y1;

**while** (Math.Abs(currentX - b) > 1e-5)

{

currentX += h;

var y = (previousY + h / 3 \* (-previousY - 4 \* currentY + 6)) / (1 + h / 3);

previousY = currentY;

currentY = y;

**yield** return (currentX, currentY);

}

}