Методы численного решения дифференциальных уравнений первого порядка

Дифференциальным уравнением первого порядка называют уравнение вида:

$$\frac{dY(x)}{dx} = F(Y(x), x) \quad (1)$$

При численном решении дифференциального уравнения первого порядка вида (1) с начальным условием $Y(x_0) = y_0$ (задача Коши) сначала выбираем фиксированное приращение аргумента $h = (x_n - x_0)/n$, где x_n - конечная точка интервала интегрирования, n - число шагов. Далее, используя один из итерационных методов численного интегрирования дифференциальных уравнений первого порядка, находим значение $Y(x_n)$.

Метод Эйлера

Метод Эйлера относится к наиболее простым, базовым методам решения дифференциальных уравнений. Этот метод основан на представлении производной в конечно-разностной форме:

$$\frac{dY(x)}{dx} = \frac{Y(x + dx) - Y(x)}{dx} \quad (2)$$

Так, для дифференциального уравнения вида (1) при достаточно малых значениях приращения аргумента x, можно, представив производную в виде отношения приращения функции к приращению аргумента, получить следующее приближенное представление в виде конечно-разностного уравнения:

$$\frac{dY(x + h) - Y(x)}{h} = F(Y(x), x) \quad (3)$$

Из этого представления легко выразить «новое» значение функции (значение функции в точке x+h) – расчетную формулу метода Эйлера:

$$Y(x + h) = Y(x) + h * F(Y(x), x) \quad (4)$$

Описание алгоритма численного интегрирования дифференциального уравнения первого порядка методом Эйлера:

1. Задаем начальное x_0 и конечное x_n значения отрезка интегрирования, начальное *y0,* т.е. значение функции y в точке x_0, а также количество шагов интегрирования n.
2. Вычисляем величину шага интегрирования h.
3. Задаем цикл по k от 2 до n+1.
4. В цикле на основе значения x_{k-1} вычисляем x_k, т.е. $x_1 = x_0 + h$, $x_2 = x_1 + h$ и т.д.
5. В цикле на основе значения Y_{k-1} по рекуррентной формуле вычисляем y_k, т.е. $Y_1=Y_0+h*F(x_0,Y_0)$, $Y_2=Y_1+h*F(x_1,Y_1)$ и т.д. Значение $F(x_k,Y_k)$ должно вычисляться в функции, соответствующей правой части дифференциального уравнения. Конец цикла.
6. Конец алгоритма.

Алгоритм численного интегрирования дифференциального уравнения первого порядка методом Эйлера на естественном языке:

```
x0=начальное_значение_x
xn=конечное_значение_x
y0=начальное_значение_Y
n=количество_шагов_интегрирования

h=(xn-x0)/n

ОПРЕДЕЛИТЬ МАССИВ X(n+1)
ОПРЕДЕЛИТЬ МАССИВ Y(n+1)
X(1)=x0
Y(1)=y0

Цикл по i от 2 до n+1
X(i)=X(i-1)+h
Y(i)=Y(i-1)+h*F(X(i-1),Y(i-1))
Конец цикла

Печать " Решение: Значения x      Значения Y(x) :"
Цикл по i от 1 до n+1
Печать X(i),Y(i)
Конец цикла

ОПРЕДЕЛИТЬ ФУНКЦИЮ F
ПАРАМЕТРЫ x,y
f=.... *** Задается вид функции
ВЕРНУТЬ f
КОНЕЦ ОПРЕДЕЛЕНИЯ ФУНКЦИИ
```

Покажем пример реализации метода Эйлера на примере уравнения

$$\frac{dY(x)}{dx} = Y(x), \text{ т.е. } F(Y(x), x) = Y(x) \quad (5)$$

При аналитическом интегрировании данного уравнения получаем начальное значение функции в точке x_0=0 и вид искомой функции:

$$Y_{test}(0) = 1, \; Y_{test}(x) = \exp(x).$$

Пример реализации алгоритма численного интегрирования дифференциального уравнения первого порядка методом Эйлера на VFP для уравнения (5):

```
SET DECIMALS TO 8
CLEAR
* Начальное и конечное значения x
x0=0
xn=1
* Начальное значение y
y0=1
* Количество итераций
n=10

h=(xn-x0)/n

DIMENSION X(n+1),Y(n+1)
X(1)=x0
Y(1)=y0

*Y(i+1)=Y(i)+h*F(X(i),Y(i))
FOR i=2 TO n+1
X(i)=X(i-1)+h
Y(i)=Y(i-1)+h*F(X(i-1),Y(i-1))
ENDFOR

? " Решение :”
? " Значения X      Известные значения Y    Вычисленные
значения Y :"
FOR i=1 TO n+1
?X(i),F_test(X(i)),Y(i)
ENDFOR


function F
lPARAMETERS x,y
f=y
RETURN f
ENDFUNC

function F_test
lPARAMETERS x
```

```
F_test=EXP(x)
RETURN F_test
ENDFUNC
```

Пример реализации алгоритма численного интегрирования дифференциального уравнения первого порядка методом Эйлера на VBA для уравнения (5):

```
Sub Euler_Test()
' Начальное и конечное значения x
x0 = 0
xn = 1
' Начальное значение y
y0 = 1
' Количество итераций
n = 10

h = (xn - x0) / n

ReDim x(n + 1)
ReDim y(n + 1)
x(1) = x0
y(1) = y0

'Y(i+1)=Y(i)+h*F(X(i),Y(i))
For i = 2 To n + 1
x(i) = x(i - 1) + h
y(i) = y(i - 1) + h * F(x(i - 1), y(i - 1))
Next i

Debug.Print "РЕШЕНИЕ :"
Debug.Print "X              Y             Y test
DaltaY"
For i = 1 To n + 1
delta = Abs(y(i) - Y_test(x(i)))
Debug.Print Format(x(i), "0.000"), _
        Format(y(i), "0.000 000"), _
        Format(Y_test(x(i)), "0.000 000"), _
        Format(delta, "0.000 000")
Next i


End Sub


Function F(x, y)
    F = y
End Function

Function Y_test(x)
    Y_test = Exp(x)
```

```
    End Function
```

Результат работы программы:

```
РЕШЕНИЕ :
X              Y             Y test       DaltaY
0,000         1,000 000     1,000 000     0,000 000
0,100         1,100 000     1,105 171     0,005 171
0,200         1,210 000     1,221 403     0,011 403
0,300         1,331 000     1,349 859     0,018 859
0,400         1,464 100     1,491 825     0,027 725
0,500         1,610 510     1,648 721     0,038 211
0,600         1,771 561     1,822 119     0,050 558
0,700         1,948 717     2,013 753     0,065 036
0,800         2,143 589     2,225 541     0,081 952
0,900         2,357 948     2,459 603     0,101 655
1,000         2,593 742     2,718 282     0,124 539
```

Для данного уравнения решение дифференциального уравнения первого порядка с шагом 0,1 дает точность порядка 0,1 (на последнем шаге). При решении с шагом 0,01 точность возрастает до порядка 0,01 (на последнем шаге).

Метод Кранка-Николсона

Метод Кранка-Николсона представляет собой модификацию метода Эйлера, в которой правая часть разностной формы дифференциального уравнения представляется в виде среднего значения правой части в точке x и правой части в точке $x+h$:

$$\frac{Y(x+h) - Y(x)}{h} = \frac{F(Y(x),x) + F(Y(x+h),x+h)}{2} \qquad (6)$$

Поскольку искомое значение функции в точке $x+h$ присутствует и в левой и в правой части уравнения – эта формула содержит $Y(x+h)$ неявно, поэтому, для получения расчетной формулы необходимо выразить $Y(x+h)$ явно. Выразить явно $Y(x+h)$ возможно не для любого вида $F(Y(x),x)$. Если правая часть дифференциального уравнения зависит только от x – $F(x)$, тогда метод Кранка-Николсона можно применять для любого вида $F(x)$.

Рассмотрим пример дифференциального уравнения с разрешимой правой частью, зависящей от *x* и *Y(x)*:

$$\frac{dY(x)}{dx} = - Y(x) * x \quad , \quad Y(0) = 0.1 \tag{7}$$

Запишем разностный вид этого дифференциального уравнения:

$$\frac{Y(x+h) - Y(x)}{h} = - \frac{Y(x+h) * (x+h)}{2} - Y(x) * \frac{x}{2}$$

После простых преобразований:

1) $$Y(x+h) = Y(x) - \frac{h * Y(x+h) * (x+h)}{2} - \frac{h * Y(x) * x}{2} \quad ,$$

2) $$Y(x+h) + \frac{h * Y(x+h) * (x+h)}{2} = Y(x) - \frac{h * Y(x) * x}{2} \quad ,$$

3) $$Y(x+h) * (1 + \frac{h * (x+h)}{2}) = Y(x) - \frac{h * Y(x) * x}{2} \quad .$$

Получим расчетную формулу:

$$Y(x+h) = \frac{Y(x) - h * Y(x) * \frac{x}{2}}{1 + \frac{h * (x+h)}{2}} \quad . \tag{8}$$

В случае, когда правая часть дифференциального уравнения зависит только от x, рекуррентная формула метода Кранка-Николсона имеет вид:

$$Y(x+h) = Y(x) + h * \frac{F(x) + F(x+h)}{2} \tag{9}$$

Например, для дифференциального уравнения с правой частью, зависящей только от *x*:

$$\frac{dY(x)}{dx} = x^2 + 3 * x \tag{10}$$

формула Кранка-Николсона имеет вид:

$$Y(x+h) = Y(x) + h * \frac{x^2 + 3 * x + (x+h)^2 + 3 * (x+h)}{2} \quad . \tag{11}$$

Описание алгоритма численного интегрирования дифференциального уравнения первого порядка с правой частью уравнения, зависящей только от *x,* методом Кранка-Николсона:

7. Задаем начальное x_0 и конечное x_n значения отрезка интегрирования, начальное *y0,* т.е. значение функции *y* в точке x_0, а также количество шагов интегрирования *n.*
8. Вычисляем величину шага интегрирования *h*.
9. Задаем цикл по k от 2 до n+1.
10. В цикле на основе значения x_{k-1} вычисляем x_k, т.е. $x_1 = x_0 + h$, $x_2 = x_1 + h$ и т.д.
11. В цикле на основе значения Y_{k-1} по рекуррентной формуле вычисляем y_k, т.е. $Y_1=Y_0+h*(F(x_0,Y_0)+F(x_1,Y_1))/2$, $Y_2=Y_1+h*(F(x_1,Y_1)+F(x_2,Y_2))/2$ и т.д. Значение $F(x_k,Y_k)$ должно вычисляться в функции, соответствующей правой части дифференциального уравнения. Конец цикла.
12. Конец алгоритма.

Алгоритм численного интегрирования дифференциального уравнения первого порядка с правой частью уравнения, зависящей только от *x,* методом Кранка-Николсона на естественном языке:

```
x0=начальное_значение_x
xn=конечное_значение_x
y0=начальное_значение_Y
n=количество_шагов_интегрирования

h=(xn-x0)/n

ОПРЕДЕЛИТЬ МАССИВ X(n+1)
ОПРЕДЕЛИТЬ МАССИВ Y(n+1)
X(1)=x0
Y(1)=y0

Цикл по i от 2 до n+1
X(i)=X(i-1)+h
Y(i)=Y(i-1)+h*(F(X(i-1),Y(i-1))+ F(X(i),Y(i)))/2
Конец цикла

Печать " Решение: Значения x      Значения Y(x) :"
Цикл по i от 1 до n+1
Печать X(i),Y(i)
Конец цикла

ОПРЕДЕЛИТЬ ФУНКЦИЮ F
```

```
ПАРАМЕТРЫ x,y
f=.... *** Задается вид функции
ВЕРНУТЬ f
КОНЕЦ ОПРЕДЕЛЕНИЯ ФУНКЦИИ
```

Покажем пример реализации метода Кранка-Николсона на примере уравнения (10). При аналитическом интегрировании уравнения (10) получаем начальное значение функции в точке x_0=0 и вид искомой функции:

$$Y_{test}(0) = 3, \quad Y_{test}(x) = \frac{x^3}{3} + \frac{3}{2} * x^2 + 3. \tag{12}$$

Пример реализации алгоритма численного интегрирования дифференциального уравнения первого порядка с правой частью, зависящей только от x, методом Кранка-Николсона на VFP для уравнения (10):

```
SET DECIMALS TO 8
CLEAR
* Начальное и конечное значения x
x0=0
xn=1
* Начальное значение y
y0=3
* Количество итераций
n=10

h=(xn-x0)/n

DIMENSION X(n+1),Y(n+1)
X(1)=x0
Y(1)=y0

*Y(i+1)=Y(i)+h*F(X(i),Y(i))
FOR i=2 TO n+1
X(i)=X(i-1)+h
Y(i)=Y(i-1)+h*F(X(i-1),Y(i-1))
ENDFOR

? " Решение :"
? " Значения X      Известные значения Y    Вычисленные
значения Y :"
FOR i=1 TO n+1
?X(i),F_test(X(i)),Y(i)
ENDFOR


function F
lPARAMETERS x,y
```

```
f=x*x+3*x
RETURN f
ENDFUNC

function F_test
lPARAMETERS x
F_test=2*x+3
RETURN F_test
ENDFUNC
```

Пример реализации алгоритма численного интегрирования дифференциального уравнения первого порядка с правой частью, зависящей только от *x*, методом Кранка-Николсона на VBA для уравнения (10):

```
Sub Krank_Nikolson_Test()
' Начальное и конечное значения x
x0 = 0
xn = 1
' Начальное значение y
y0 = 3
' Количество итераций
n = 10

h = (xn - x0) / n

ReDim x(n + 1)
ReDim Y(n + 1)
x(1) = x0
Y(1) = y0

'Y(x+h) = Y(x) + h*(x^2 + 3*x + (x+h)^2 + 3*(x+h))/2
For i = 2 To n + 1
x(i) = x(i - 1) + h
Y(i) = Y(i - 1) + h * (f(x(i - 1), Y(i - 1)) + f(x(i),
Y(i))) / 2
Next i

Debug.Print "РЕШЕНИЕ :"
Debug.Print "X              Y             Y test
DaltaY"
For i = 1 To n + 1
delta = Abs(Y(i) - Y_test(x(i)))
Debug.Print Format(x(i), "0.000"), _
        Format(Y(i), "0.000 000"), _
        Format(Y_test(x(i)), "0.000 000"), _
        Format(delta, "0.000 000")
Next i

End Sub


Function f(x, Y)
```

```
    f = x ^ 2 + 3 * x
End Function

Function Y_test(x)
    Y_test = (x ^ 3) / 3 + 3 / 2 * (x ^ 2) + 3
End Function
```

Результат работы программы:

```
РЕШЕНИЕ :
X              Y              Y test         DaltaY
0,000          3,000 000      3,000 000      0,000 000
0,100          3,015 500      3,015 333      0,000 167
0,200          3,063 000      3,062 667      0,000 333
0,300          3,144 500      3,144 000      0,000 500
0,400          3,262 000      3,261 333      0,000 667
0,500          3,417 500      3,416 667      0,000 833
0,600          3,613 000      3,612 000      0,001 000
0,700          3,850 500      3,849 333      0,001 167
0,800          4,132 000      4,130 667      0,001 333
0,900          4,459 500      4,458 000      0,001 500
1,000          4,835 000      4,833 333      0,001 667
```

Как видно из приведенного примера, решение дифференциального уравнения первого (10) порядка с шагом 0,1 дает точность порядка 0,001 (на последнем шаге). При интегрировании с шагом 0,01 точность вычислений возрастает до 0,00001 (на последнем шаге).

Таким образом, сравнивая методы Эйлера и Кранка-Никольсона можно отметить следующее: метод Кранка-Никольсона как минимум в 100 раз точнее метода Эйлера, при этом точность вычислений существенно возрастает при уменьшении шага интегрирования.

Однако, метод Кранка-Никольсона удобно применять в тех случаях, когда правая часть дифференциального уравнения зависит только от *x*. Существуют также методы интегрирования, учитывающие зависимость правой части уравнения и от *x* и от *Y(x)*, например, методы Рунге-Кутты.

Метод Рунге-Кутты четвертого порядка для решения уравнения первого порядка

Наиболее употребительным методом Рунге-Кутты решения уравнения первого порядка вида (1) является метод четвертого порядка, в котором вычисления производятся по формуле:

$$Y_{i+1} = Y_i + (k_1 + 2*k_2 + 2*k_3 + k_4)/6, \qquad (13)$$

где:

$$
\begin{aligned}
k_1 &= F(x_i, Y_i) * h \\
k_2 &= F(x_i + h/2, Y_i + k_1/2) * h \\
k_3 &= F(x_i + h/2, Y_i + k_2/2) * h \\
k_4 &= F(x_i + h, Y_i + k_3) * h \\
i &= 0,\ldots, n - 1 \\
h &= \frac{(x_n - x_0)}{n}
\end{aligned}
\tag{14}
$$

Описание алгоритма метода Рунге-Кутты решения уравнения первого порядка:

1. Задаем начальное x_0 и конечное x_n значения отрезка интегрирования, начальное *y0,* т.е. значение функции *y* в точке x_0, а также количество шагов интегрирования *n.*
2. Вычисляем величину шага интегрирования *h.*
3. Цикл по *x* от x_0 до x_n с шагом *h* .
4. Вычисляем *k1 = F(x , y)*h.*
5. Вычисляем *x1 = x +h/2.*
6. Вычисляем *Y1 = Y +K1 /2.*
7. Вычисляем *k2 = F(x1,y1)*h.*
8. Вычисляем *k2 = F(x1,y1)*h.*
9. Вычисляем *Y2 = Y+K2 /2.*
10. Вычисляем *k3 = F(x1,y2)*h.*
11. Вычисляем *x4 = x+h.*
12. Вычисляем *y4 = Y+K3.*
13. Вычисляем *k4 = F(x4,y4)*h.*
14. Вычисляем *Y = Y +(k1 +2*k2 +2*k3 +k4)/6.*
15. Конец цикла.
13. Определить функцию F(x,y).
14. Задать вид функции F(x,y).
15. Конец функции.

Алгоритм метода Рунге-Кутты 4-го порядка для решения дифференциального уравнения первого порядка на естественном языке:

```
x0 = значение_левой_границы
xn = значение_правой_границы
Y = начальное_значение_функции
n=количество_шагов_интегрирования
h = (xn-x0)/n
Цикл по x от x0 до xn с шагом h
     k1 = F(x , y )*h
     x1 = x +h/2
     Y1 = Y +K1 /2
     k2 = F(x1,y1)*h
     Y2 = Y+K2 /2
     k3 = F(x1,y2)*h
     x4 = x+h
     y4 = Y+K3
     k4 = F(x4,y4)*h
     Y = Y +(k1 +2*k2 +2*k3 +k4 )/6
КонецЦикла

ОПРЕДЕЛЕНИЕ ФУНКЦИИ F
ПАРАМЕТРЫ x,y
F=… *** задается вид функции
КонецФункции
```

Приведем примеры реализаций метода Рунге-Кутта для ранее рассмотренного уравнения (см. метод Эйлера) и сравним точность вычислений методом Эйлера и Рунге-Кутта.

Пример реализация алгоритма численного интегрирования дифференциального уравнения первого порядка на VFP для уравнения (5):

```
CLEAR

x0 = 0
xn = 1
y = 1

n = 10

h = (xn-x0)/n

?"            X                              Тестовые Y_test
Вычисленные Y                  Delta(Y_test-Y)"

for x=x0 to xn step h
```

```
k1 = F(x , y )*h
x1 = x +h/2
Y1 = Y +K1 /2
k2 = F(x1,y1)*h
Y2 = Y+K2 /2
k3 = F(x1,y2)*h
x4 = x+h
y4 = Y+K3
k4 = F(x4,y4)*h
Y = Y +(k1 +2*k2 +2*k3 +k4 )/6
delta=ABS(Y_test(x)-Y)
? x,Y_test(x),Y,delta
ENDFOR

Function F(x, y)
F=y
RETURN F
EndFunc

Function Y_test(x)
Y_test=EXP(x)
RETURN Y_test
EndFunc
```

Пример реализации алгоритма численного интегрирования дифференциального уравнения первого порядка методом Рунге-Кутта на VBA для уравнения (5):

```
Sub Test_Runge_Kytt()
Dim x(), y()

x0 = 0: xn = 1
y0 = 1

n = 100

ReDim y(n + 1): ReDim x(n + 1)

Call runge(0, 1, y0, x(), y(), n)

For i = 1 To n
ActiveSheet.Cells(i, 1).Value = x(i)
ActiveSheet.Cells(i, 2).Value = y(i)

    y_test = f_test(x(i))
    delta_y = Abs(y_test - y(i))

Debug.Print "x= " & Format(x(i), "0.000") & " : y= " &
Format(y(i), "0.000 000 000 000") & " : y_test= " _
    & Format(y_test, "0.000 000 000 000") & " : delta_y= "
& Format(delta_y, "0.000 000 000 000")
```

```
Next i

Call AddChart(1, 2)

End Sub


Sub runge(x0, xn, y0, x(), y(), n)
Dim i, h, dk

h = (xn - x0) / n
k1 = h * f(x0, y0)
k2 = h * f(x0 + h / 2, y0 + k1 / 2)
k3 = h * f(x0 + h / 2, y0 + k2 / 2)
k4 = h * f(x0 + h, y0 + k3)
dk = 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
y(1) = y0 + dk
x(1) = x0 + h

For i = 1 To n
k1 = h * f(x(i), y(i))
k2 = h * f(x(i) + h / 2, y(i) + k1 / 2)
k3 = h * f(x(i) + h / 2, y(i) + k2 / 2)
k4 = h * f(x(i) + h, y(i) + k3)
dk = 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
y(i + 1) = y(i) + dk
x(i + 1) = x(i) + h
Next i

End Sub

Function f(x, y)
f = y
End Function

Function f_test(x)
f_test = Exp(x)
End Function

Sub AddChart(Col1, Col2)

    ASheetName = Application.ActiveSheet.Name


    Charts.Add
    ActiveChart.ChartType = xlLine
                                    ActiveChart.SetSourceData
Source:=Sheets(ASheetName).Range("A1:B100"), PlotBy _
        :=xlColumns
    ActiveChart.SeriesCollection(1).Delete
        ActiveChart.SeriesCollection(1).XValues  =  "="  &
ASheetName & "!C1"
```

```
            ActiveChart.Location   Where:=xlLocationAsObject,
Name:=ASheetName
    With ActiveChart
        .HasTitle = False
        .Axes(xlCategory, xlPrimary).HasTitle = False
        .Axes(xlValue, xlPrimary).HasTitle = False
    End With

End Sub
```

В данной реализации все значения вычисляемой функции и ее аргумента хранятся в массивах, также использована процедура AddChart MS EXCEL для построения графика значений вычисляемой функции.

Результаты работы программы на VBA:

```
x=0,10:y=1,105 170 833 333:y_test=1,105 170 918 076:delta_y=0,000 000 084 742
x=0,20:y=1,221 402 570 851:y_test=1,221 402 758 160:delta_y=0,000 000 187 309
x=0,30:y=1,349 858 497 063:y_test=1,349 858 807 576:delta_y=0,000 000 310 513
x=0,40:y=1,491 824 240 081:y_test=1,491 824 697 641:delta_y=0,000 000 457 561
x=0,50:y=1,648 720 638 597:y_test=1,648 721 270 700:delta_y=0,000 000 632 103
x=0,60:y=1,822 117 962 092:y_test=1,822 118 800 391:delta_y=0,000 000 838 299
x=0,70:y=2,013 751 626 597:y_test=2,013 752 707 470:delta_y=0,000 001 080 874
x=0,80:y=2,225 539 563 292:y_test=2,225 540 928 492:delta_y=0,000 001 365 200
x=0,90:y=2,459 601 413 780:y_test=2,459 603 111 157:delta_y=0,000 001 697 377
x=1,00:y=2,718 279 744 135:y_test=2,718 281 828 459:delta_y=0,000 002 084 324
```

Как видно из результатов программы, метод Рунге-Кутты 4-го при шаге интегрирования 0,1 дает точность порядка 0,00001. При интегрировании с шагом 0,01 данный метод дает точность порядка 10^{-9}.

Сравнивая результаты работы трех приведенных методов решения дифференциальных уравнений первого порядка, приходим к выводу, что метод Рунге-Кутты является гораздо более точным, чем методы Эйлера и Кранка-Никольсона.

Методы численного решения систем дифференциальных уравнений

Гармонические колебания

Уравнение, описывающее идеальный периодический процесс (гармонические колебания), представляет собой дифференциальное уравнение второго порядка:

$$\frac{d^2Y(t)}{dt^2} = -W^2Y(t) \tag{15}$$

Это уравнение легко можно записать в виде системы дифференциальных уравнений первого порядка:

$$\begin{cases} \frac{dY(t)}{dt} = Y_1(t) \\ \frac{dY_1(t)}{dt} = -W^2 * Y(t) \end{cases} \tag{16}$$

Эту систему можно записать в матричном виде:

$$\frac{d}{dt}\begin{bmatrix} Y \\ Y_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -W^2 & 0 \end{bmatrix} * \begin{bmatrix} Y \\ Y_1 \end{bmatrix}, \text{ или } \frac{dY}{dt} = A * Y, \tag{17}$$

где A – матрица постоянных коэффициентов, Y – искомый вектор.

Эта система дифференциальных уравнений относится к линейным системам, поскольку в правой части отсутствуют нелинейные слагаемые по искомой функции (нет квадратов, кубов и других нелинейных выражений от Y). Кроме того, эта система относится к однородным системам дифференциальных уравнений, поскольку в правой части отсутствуют свободные слагаемые, зависящие только от аргумента.

Интересной особенностью однородных систем дифференциальных уравнений, как в прочем и однородных систем алгебраических уравнений, является наличие тривиального – нулевого решения – Y(t)=0. Это решение соответствует состоянию покоя (равновесия) моделируемого объекта.

Затухающие колебания

Периодический процесс с затуханием описывается дифференциальным уравнением гармонических колебаний с учетом силы сопротивления. Силу сопротивления можно описать различными способами, в зависимости от ее характера. Рассмотрим возможный вариант силы сопротивления. Скорость

изменения моделируемой величины - $\frac{dY}{dx}$. Пусть сила сопротивления будет направлена против скорости и пропорциональна скорости изменения по величине, то есть, чем больше скорость изменения, тем больше сила сопротивления, если скорость изменения моделируемой величины равна нулю, тогда и сила сопротивления будет равна нулю. Введем коэффициент пропорциональности силы сопротивления скорости – k, тогда получим выражение для силы сопротивления:

$$F_{\tilde{n}\hat{i}\hat{i}\eth} = - k * \frac{dY}{dt} \tag{18}$$

Таким образом, дифференциальное уравнение, описывающее затухающие колебания, имеет вид:

$$\frac{d^2Y(t)}{dt^2} = - W^2 Y(t) - k * \frac{dY}{dt} \tag{19}$$

Это дифференциальное уравнение легко представить в виде системы дифференциальных уравнений первого порядка:

$$\begin{cases} \frac{dY(t)}{dt} = Y_1(t) \\ \frac{dY_1(t)}{dt} = - W^2 * Y(t) - k * Y_1(t) \end{cases} \tag{20}$$

Матричная форма дифференциального уравнения:

$$\frac{d}{dt}\begin{bmatrix} Y \\ Y_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ - W^2 & - k \end{bmatrix} * \begin{bmatrix} Y \\ Y_1 \end{bmatrix} \tag{21}$$

Система дифференциальных уравнений, описывающая затухающие колебания также относится к линейным однородным дифференциальным уравнениям, имеет тривиальное решение и, более того, любое решение этой системы стремится к тривиальному – то есть к состоянию равновесия.

Периодический процесс с затуханием и с внешней силой

Уравнение, описывающее периодический процесс с затуханием и с внешней возбуждающей силой.

$$\frac{d^2Y(t)}{dt^2} = -W^2Y(t) - k*\frac{dY}{dt} + F(t) \tag{22}$$

Иногда, если внешняя сила *F(t)* периодическая, такую систему называют системой вынужденных колебаний.

Запишем это дифференциальное уравнение в виде системы дифференциальных уравнений первого порядка:

$$\begin{cases} \frac{dY(t)}{dt} = Y_1(t) \\ \frac{dY_1(t)}{dt} = -W^2 * Y(t) - k * Y_1(t) + F(t) \end{cases} \tag{23}$$

Запись системы дифференциальных уравнений в матричном виде:

$$\frac{d}{dt}\begin{bmatrix} Y \\ Y_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -W^2 & -k \end{bmatrix} * \begin{bmatrix} Y \\ Y_1 \end{bmatrix} + \begin{bmatrix} 0 \\ F(t) \end{bmatrix} \tag{24}$$

Эта система дифференциальных уравнений уже не является однородной. В правой части имеется свободное, зависящее от аргумента, слагаемое – столбец $\begin{bmatrix} 0 \\ F(t) \end{bmatrix}$, представляющий собой вектор возбуждающей силы. Неоднородные линейные системы дифференциальных уравнений уже не обязаны иметь тривиальное решение. Более того, по истечении достаточно большого промежутка времени, когда начальное возмущение затухнет, останется решение, определяющееся возбуждающей силой.

Для решения систем дифференциальных уравнений можно применять такие же методы, как и для дифференциальных уравнений первого порядка, только при этом, необходимо модифицировать расчетные формулы с учетом того, что искомая функция является вектором, и правая часть дифференциального уравнения представляет собой векторное выражение.

Решение системы ДУ методом Эйлера

Рассмотрим систему дифференциальных уравнений вида (17):

$$\frac{dY}{dt} = A * Y$$

где Y – вектор решений, A – матрица. В общем случае коэффициенты матрицы могут зависеть от t.

Расчетная формула по методу Эйлера для такого уравнения имеет вид:

$$Y(t+h) = Y(t) + h * A * Y(t) \qquad (25)$$

Описание алгоритма решения системы ДУ методом Эйлера:

1. Задаем размерность системы *N*.
2. Объявляем вектор решений *Y(N)*.
3. Объявляем матрицу *A(N,N)*.
4. Объявляем вспомогательный вектор *P(N)*.
5. Задаем начальное t_0 и конечное t_m значения отрезка интегрирования, массив начальных значений Y_0, т.е. значения системы $Y(t_0)$ ….. $Y(t_m)$ в точке t_0, а также количество шагов интегрирования *m*
6. Вычисляем величину шага интегрирования *h*.
7. Цикл по *t* от t_0 до t_m с шагом *h*
8. Цикл по *i* от 1 до *N*
9. Вызвать процедуру Получить_Матрицу(*A,t*)
10. Задать *P(i)*=0.
11. Цикл по *j* от 1 до *N*.
12. Вычислить *P(i)=P(i)+A(i,j)*Y(t)*.
13. Конец цикла по *j*.
14. Конец цикла по *i*.
15. Цикл по *i* от 1 до *N*.
16. Вычислить *Y(i)=Y(i)+h*P(i)*.
17. Конец цикла по *i*.
18. Вызвать процедуру Печать(*Y,t*)
19. Конец цикла по *t*.

Алгоритм решения системы ДУ методом Эйлера на естественном языке:

```
N=размерность_системы
Объявить Массив Y(N),P(N),A(N,N)
t0=0
tm=100
m=1000
H = (tm-t0)/m

Получить_вектор (Y,t0)
```

```
Цикл по t от t0 до tm с шагом h

      Цикл по i=1 до N
         Получить_матрицу (A,t)

         P(i)=0
         Цикл по j=1 до N
         P(i) = P(i) + A(i,j)*Y(j)
         КонецЦикла
      КонецЦикла

      Цикл по i=1 до N
         Y(i) = Y(i) + h*P(i)
      КонецЦикла
Печать_Вектор(Y, t)

КонецЦикла
```

Пример реализации алгоритма решения системы дифференциальных уравнений методом Эйлера на VFP:

```
SET DECIMALS TO 8
CLEAR

N=2
DIMENSION Y(N),P(N),A(N,N)
t0=0
tm=10
m=100
H =(tm-t0)/m

Получить_вектор(@Y,N,t0)

FOR t=t0 to tm STEP h

      FOR i=1 TO N
         Получить_матрицу(@A,N,t)

         P(i)=0
         FOR j=1 TO N
         P(i) = P(i) + A(i,j)*Y(j)
         ENDFOR
      ENDFOR

      FOR i=1 TO N
         Y(i) = Y(i) + h*P(i)
      ENDFOR
Печать_Вектор(@Y, t)

ENDFOR

PROCEDURE Получить_вектор
```

```
PARAMETERS Y,N,t0
nf=FOPEN("U:\Prog\VFP\Euler_Sys\Y.txt",0)
     FOR i=1 TO N
     Y(i)=VAL(FGETS(nf))
     ENDFOR
FCLOSE(nf)
ENDPROC

PROCEDURE Получить_матрицу
PARAMETERS A,N,t
nf=FOPEN("U:\Prog\VFP\Euler_Sys\A.txt",0)
     FOR j=1 TO N
      FOR i=1 TO N
     A(i,j)=VAL(FGETS(nf))
      ENDFOR
     ENDFOR
FCLOSE(nf)
ENDPROC

PROCEDURE Печать_Вектор
PARAMETERS Y,t
          ?" t= " + ALLTRIM(STR(t))
     FOR i=1 TO alen(y)
          ? " y(" + ALLTRIM(STR(i)) + ")= " +
ALLTRIM(STR(y(i)))
     ENDFOR
ENDPROC
```

Пример реализации алгоритма решения системы дифференциальных уравнений методом Эйлера на VBA:

```
Sub Euler_Sys()

n = 2
ReDim Y(n): ReDim P(n): ReDim A(n, n)
t0 = 0
tm = 100
m = 1000
h = (tm - t0) / m

Call Получить_вектор(Y(), n, t0)

For t = t0 To tm Step h

     For i = 1 To n
        Call Получить_матрицу(A(), n, t)

        P(i) = 0
        For j = 1 To n
        P(i) = P(i) + A(i, j) * Y(j)
        Next j
     Next i
```

```
      For i = 1 To n
         Y(i) = Y(i) + h * P(i)
      Next i
Call Печать_Вектор(Y(), t)

Next t

End Sub

Sub Получить_вектор(Y(), n, t0)
Open "U:\Prog\VFP\Euler_Sys\Y.txt" For Input As #nf
    For i = 1 To n
Input #nf, Y(i)
    Next i
Close #nf
End Sub

Sub Получить_матрицу(A(), n, t)
Open "U:\Prog\VFP\Euler_Sys\A.txt" For Input As #nf
    For j = 1 To n
     For i = 1 To n
Input #nf, A(i, j)
     Next i
    Next j
Close #nf
End Sub


Sub Печать_Вектор(Y(), t)
        Debug.Print " t= " & CStr(t)
    For i = 1 To UBound(Y)
        Debug.Print " y(" & CStr(i) & ")= " & CStr(Y(i))
    Next i
End Sub
```

Решение системы ДУ методом Кранка-Николсона

Рассмотрим систему дифференциальных уравнений вида (17):

$$\frac{dY}{dt} = A * Y$$

где Y – вектор решений, A – матрица. В общем случае коэффициенты матрицы могут зависеть от t.

Получим расчетную формулу метода для систем ДУ:

1) $$Y(y+h) - Y(t) = \frac{h * A * Y(y)}{2} + \frac{h * A * Y(t+h)}{2},$$

2) $$E*Y(y+h)-\frac{h*A*Y(t+h)}{2}=Y(t)+\frac{h*A*Y(t)}{2},$$

3) $$(E-\frac{h}{2}*A)*Y(t+h)=Y(t)*(E+\frac{h}{2}*A).$$

В результате проделаных упрощений получаем расчетную формулу для решения системы ДУ методом Кранка-Николсона:

$$Y(t+h)=Y(t)*(E+\frac{h}{2}*A)*(E-\frac{h}{2}*A)^{-1} \quad (26)$$

Алгоритм решения системы ДУ методом Кранка-Николлсона на естественном языке:

```
N=значение_размерности_системы_ДУ
Массив A(N,N), Y(N), E(N,N),B1(N,N),B2(N,N),B3(N,N)
Получить_единичную_матрицу(@E,N)
Задать_значения_матрице(@B1,0,N)
Задать_значения_матрице(@B2,0,N)
Задать_значения_матрице(@B3,0,N)

A=левая_граница
B=правая_граница
H=значение_шага
X=a
Получить_начальные_значения(@Y,N,a)
Цикл по x от a до b с шагом h
     Получить_матрицу_правых_частей(@A,x,N)
     Цикл по i=1 до N
     Цикл по j=1 до N
          B1(i,j)=E(i,j)-0.5*h*A(i,j)
     КонецЦикла
     КонецЦикла

     Обратить_матрицу(@B1,N)

     Цикл по i=1 до N
     Цикл по j=1 до N
          B2(i,j)=E(i,j)+0.5*h*A(i,j)
     КонецЦикла
     КонецЦикла

     Умножить_матрицы(@B1,@B2,@B3,N)
     Умножить_матрицу_на_вектор(@B3,@Y,@Y1,N)
     Печать_решения(@Y1,x+h,N)
КонецЦикла

Процедура Получить_единичную_матрицу(E,N)
Локальные переменные i,j
Цикл по i=1 до N
     Цикл по j=1 до N
```

```
          Если i=j тогда
                    E(i,j)=1
          Иначе
               E(i,j)=0
          КонецЕсли
     КонецЦикла
КонецЦикла
КонецПроцедуры

Процедура Задать_значения_матрице(B,X,N)
Локальные переменные i,j
Цикл по i от 1 до N
     Цикл по j от 1 до N
          B(i,j)=X
     КонецЦикла
КонецЦикла


Процедура Получить_начальные_значения(Y,N,a)
     Y(1)=нач.значение_1
     Y(2)=нач.значение_2
          …
     Y(N)=нач.значение_N
КонецПроцедуры

Процедура Получить_матрицу_правых_частей(A,x,N)
     A(1,1)= значение 1_1 как функция от x
     A(1,2)= значение 1_2 как функция от x
     …
     A(N,N)= значение N_N как функция от x
КонецПроцедуры
Процедура Обратить_матрицу(B1,N)
     ***  выполняется обращение матрицы B1
     ***  обращенная матрица подставляется вместо B1
КонецПроцедуры

Процедура Умножить_матрицы(A,B,C,N)
Локальные переменные i,j,k
     Цикл по i от 1 до N
          Цикл по j от 1 до N
               C(i,j)=0
               Цикл по k от 1 до N
                    C(i,j) = C(i,j)+A(i,k)*B(k,j)
               КонецЦикла
          КонецЦикла
     КонецЦикла
КонецПроцедуры

Процедура Умножить_матрицу_на_вектор(A,B,C,N)
Локальные переменные i,j,k
     Цикл по i от 1 до N
          C(i)=0
          Цикл по j от 1 до N
```

```
                C(i)= C(i)+A(i,j)*B(j)
            КонецЦикла
        КонецЦикла
КонецПроцедуры

Процедура Печать_решения(Y,x,N)
*** выполняется вывод значений x и компонент вектора Y
КонецПроцедуры
```

Метод Рунге-Кутта четвертого порядка для решения системы уравнений первого порядка

Система обыкновенных дифференциальных уравнений высшего порядка путем введения новых неизвестных может быть сведена к системе уравнений первого порядка. Рассмотрим такую систему

$y_i' = F_i(x, y)$

где $i = 1, ..., n$;

$y = (y1, y2, ..., yn)$;

$y(x0) = y(0) = (y(0)1, ..., y(0)n)$.

Методе Рунге-Кутта вычисляет значение y_i^k по формуле:

$$y_i^{(k+1)} = y_i^{(k)} + (k_1^{(i)}+2*k_2^{(i)}+2*k_3^{(i)}+k_4^{(i)})/6$$

где:

$$k_1^{(i)} = F_i(x^{(k)}, y^{(k)})*h$$

$$k_2^{(i)} = F_i(x^{(k)}+h/2, y^{(k)}+k_1^{(i)}/2)*h$$

$$k_3^{(i)} = F_i(x^{(k)}+h/2, y^{(k)}+k_2^{(i)}/2)*h$$

$$k_4^{(i)} = F_i(x^{(k)}+h, y^{(k)}+k_3^{(i)})*h,$$

$$h = (x_n - x_0)/m$$

n – количество уравнений системы, m - количество шагов интегрирования. Процедура использует набор функций $F(i, x, y)$, которые соответствуют функциям $F_i(x, y)$ описанным выше.

Описание алгоритма метода Рунге-Кутта решения систем дифференциальных уравнений первого порядка на естественном языке:

1. Задаем начальное x_0 и конечное x_n значения отрезка интегрирования, массив начальных значений y_0, т.е. значения системы *y0(i)… y0(n)* в точке x_0, а также количество шагов интегрирования *m*.
2. Вычисляем шаг интегрирования *h*.
3. Задаем цикл по *x* от x_0 до x_n с шагом $\underline{h}$.
4. На основе известных x_0 и *y0(1)… y0(n)* вычисляем правые части системы для всех *n* уравнений.
5. Вычисляем коэффициенты k_1 и для всех *n* уравнений.
6. Шаги 4 и 5 проделываются для вычисления коэффициентов k_2, k_3, k_4 для всех *n* уравнений системы в соответствии с формулами метода.
7. Вычисляем $y_1 \ldots y_m$ для первого шага интегрирования в соответствии с главной формулой метода.
8. Шаги 4, 5, 6, 7 повторяются для вычисления состояний системы на каждом шаге интегрирования.
9. Конец цикла по x.
10. Конец алгоритма.

Алгоритм метода Рунге-Кутта решения систем дифференциальных уравнений первого порядка на естественном языке:

```
N=размерность_системы
Массивы Y(N),F(N),K1(N),K2(N),K3(N),K4(N)
a = значение_левой_границы
b = значение_правой_границы
h = значение_шага
Цикл по i=1 до N
     F(i)=0
     K1(i)=0
     K2(i)=0
     K3(i)=0
     K4(i)=0
     Y(i) = начальное_значение_i-ой_компоненты_функции
КонецЦикла

Цикл по x от a до b с шагом h
       RP(x,@F,@Y,N)
     Цикл по I от 1 до N
          K1(i)  = F(i)*h
          Y1(i)= Y(i) +K1(i) /2
```

```
      КонецЦикла
            x1= x +h/2
            RP(x1,@F,@Y1,N)
      Цикл по i от 1 до N
                  K2(i) = F(i)*h
            Y2(i)= Y(i)+K2(i) /2
      КонецЦикла
            RP(x1,@F,@Y2,N)
      Цикл по i от 1 до N
                        K3(i)=F(i)*h
            Y4(i)=Y(i)+K3(i)
      КонецЦикла
            x4=x+h
            RP(x4,@F,@Y4,N)
      Цикл по i от 1 до N
            K4(i) = F(i)*h
            Y(i)  = Y(i) +(K1(i) +2*K2(i) +2*K3(i) +K4(i) )/6
      КонецЦикла
      Печать_решения(@Y,N,x4)
КонецЦикла

Процедура RP(x,@F,@Y,N)
      Локальные переменные i
      Цикл по i=1 до N
             F(i) = Выражение от Y(k), x
      КонецЦикла

КонецПроцедуры
```

Реализация алгоритма численного интегрирования системы дифференциальных уравнений первого порядка методом Рунге-Кутта на VFP:

```
set decimals to 10
clear

n=100
y0=0.5
m=2*n

DIMENSION y(m)

FOR i=1 TO m STEP 2
y(i)=0
y(i+1)=1
NEXT

runge_sys(0,2*PI(),@y,m,n)
```

```
PROCEDURE runge_sys
PARAMETERS xn,xe,y0,m,n
DIMENSION y0(m)
LOCAL ARRAY k1(m),k2(m),k3(m),k4(m),y1(m),y0_(m),f(m)
LOCAL i,x0,h

h=(xe-xn)/(n-1)

FOR x0 = xn TO xe STEP h

правые_части(x0,@y0,@f,m)

FOR i=1 TO m
k1(i)=h*f(i)
y0_(i) = y0(i)+k1(i)/2
NEXT

правые_части(x0+h/2,@y0_,@f,m)

FOR i=1 TO m
k2(i)=h*f(i)
y0_(i) = y0(i)+k2(i)/2
NEXT

правые_части(x0+h/2,@y0_,@f,m)

FOR i=1 TO m
k3(i)=h*f(i)
y0_(i) = y0(i)+k3(i)
NEXT

правые_части(x0+h,@y0_,@f,m)

FOR i=1 TO m
k4(i)=h*f(i)
dk=1/6*(k1(i)+2*k2(i)+2*k3(i)+k4(i))
y1(i)=y0(i)+dk
NEXT

вывод_решения(x0+h,@y1,m)


FOR i=1 TO m
y0(i) = y1(i)
NEXT

NEXT

RETURN


PROCEDURE правые_части
```

```
LPARAMETERS x,y,f,n
DIMENSION y(n),f(n)
*? "x=",x
FOR i=1 TO 200 STEP 2
f(i)=y(i+1)
f(i+1)=-y(i)
NEXT
RETURN
ENDPROC

PROCEDURE вывод_решения
PARAMETERS x,y,m
DIMENSION y(m)
? x,STR(y(1),10,7),STR(f(x),10,7),STR(ABS(y(1)-f(x)),10,7)
RETURN
ENDPROC


FUNCTION f
PARAMETERS x
RETURN SIN(x)
ENDFUNC
```

Реализация алгоритма численного интегрирования системы дифференциальных уравнений первого порядка методом Рунге-Кутта на VBA:

```
Sub runge_sys()
Const PiNumber = 3.14159265358979

Dim y()
Dim k1(), k2(), k3(), k4(), y1(), y_tmp(), f()

dec = 8
n = 100
m = 2 * n

x1 = 0: xn = 2 * PiNumber

ReDim k1(m): ReDim k2(m): ReDim k3(m): ReDim k4(m)
ReDim y1(m): ReDim y_tmp(m): ReDim f(m)
ReDim y(m)


For i = 1 To m Step 2
y(i) = 0
y(i + 1) = 1
Next i

h = (xn - x1) / (n - 1)
```

```
For x0 = x1 To xn Step h

Call правые_части(x0, y(), f, m)

For i = 1 To m
k1(i) = h * f(i)
y_tmp(i) = y(i) + k1(i) / 2
Next i

Call правые_части(x0 + h / 2, y_tmp, f, m)

For i = 1 To m
k2(i) = h * f(i)
y_tmp(i) = y(i) + k2(i) / 2
Next i

Call правые_части(x0 + h / 2, y_tmp, f, m)

For i = 1 To m
k3(i) = h * f(i)
y_tmp(i) = y(i) + k3(i)
Next i

Call правые_части(x0 + h, y_tmp, f, m)

For i = 1 To m
k4(i) = h * f(i)
dk = 1 / 6 * (k1(i) + 2 * k2(i) + 2 * k3(i) + k4(i))
y1(i) = y(i) + dk
Next i

Call вывод_решения(x0 + h, y1, m, dec)

For i = 1 To m
y(i) = y1(i)
Next i

Next x0

End Sub


Sub правые_части(x, y, f, n)
'Debug.Print "x=", x
For i = 1 To 200 Step 2
f(i) = y(i + 1)
f(i + 1) = -y(i)
Next i

End Sub

Sub вывод_решения(x, y, m, dec)
```

```
dStr = "0."
For i = 1 To dec
dStr = dStr + "0"
Next i
Debug.Print    Format(x,    dStr),    Format(y(1),    dStr),
Format(f(x), dStr), Format(Abs(y(1) - f(x)), dStr)
'Return
End Sub


Function f(x)
f = Sin(x)
End Function
```