>

> *#Laboratory work №1*
version 6
Topic : *Methods numerical solution of first order ordinary differential equations.*
Student of group Tee−1 :
Коронкевича Д.А.

> *restart*; # *Runge-Kutta method*

> *Digits* := 5;

$$Digits := 5 \tag{1}$$

> *ya* := 1;# *Function value y(x=a)*

$$ya := 1 \tag{2}$$

> $f := (-1/x)*y + (\ln(x) + 1);$

$$f := -\frac{y}{x} + \ln(x) + 1 \tag{3}$$

> $n := 10;$

$$n := 10 \tag{4}$$

> $a := 1; b := 2;$

$$a := 1$$
$$b := 2 \tag{5}$$

> $h := (b-a)/n;$

$$h := \frac{1}{10} \tag{6}$$

> $X[0] := a; Y[0] := ya;$

$$X_0 := 1$$
$$Y_0 := 1 \tag{7}$$

> $X[0] := a; Y[0] := ya;$

>

```
> for i from 1 to n do
x := X[i-1] : y := Y[i-1] :
k1 := h*f :
x := X[i-1] + h/2 : y := Y[i-1] + k1/2;
k2 := h*f :
x := X[i-1] + h/2 : y := Y[i-1] + k2/2;
k3 := h*f :
x := X[i-1] + h : y := Y[i-1] + k3;
k4 := h*f :
Y[i] := Y[i-1] + 1/6*(k1 + 2*k2 + 2*k3 + k4) :
X[i] := X[i-1] + h :
end do:
```

```
> printf("Runge-Kutta method ") :
print( ) :
printf("  Xi         Yi") :
```

```
print( ) :
for i from 0 to n do
printf("%f %f \n", evalf(X[i]), evalf(Y[i])) :
endtable[1, i] := evalf(X[i]) :
endtable[2, i] := evalf(Y[i]) :
end do: print( ) :
```

```
Runge-Kutta method

  Xi          Yi

1.000000 1.000000
1.100000 1.009200
1.200000 1.034300
1.300000 1.072400
1.400000 1.121200
1.500000 1.179100
1.600000 1.244700
1.700000 1.317000
1.800000 1.395800
1.900000 1.479500
2.000000 1.568100
```

(8)

```
> # Euler method
> for i from 1 to n do
x := X[i-1] : y := Y[i-1] :
Y[i] := Y[i-1] + f* h :
X[i] := X[i-1] + h :
end do:

> print( ) :
printf("  Euler method ") :
print( ) :
printf("  Xi        Yi") :
print( ) :
for i from 0 to n do
printf("%f %f \n", evalf(X[i]), evalf(Y[i])) :
endtable[3, i] := evalf(Y[i]) :
end do:
print( ) :
```

```
  Euler method

  Xi          Yi

1.000000 1.000000
1.100000 1.000000
1.200000 1.018600
1.300000 1.051900
1.400000 1.097300
```

```
1.500000 1.152500
1.600000 1.216200
1.700000 1.287200
1.800000 1.364700
1.900000 1.447600
2.000000 1.535600
```

(9)

>

> *# Exact solution*
> $y := 'y'; x := 'x';$

$$y := y$$
$$x := x \quad \textbf{(10)}$$

> $deq1 := diff(y(x), x) = (-1/x) * y(x) + (\ln(x) + 1);$

$$deq1 := \frac{d}{dx} y(x) = -\frac{y(x)}{x} + \ln(x) + 1 \quad \textbf{(11)}$$

> $x0 := a;$

$$x0 := 1 \quad \textbf{(12)}$$

> $lnC := y(x0) = 1;$

$$lnC := y(1) = 1 \quad \textbf{(13)}$$

> $F00 := dsolve(\{deq1, lnC\}, y(x)) : evalf(\%);$

$$y(x) = 0.50000 \ln(x)\, x + 0.25000\, x + \frac{0.75000}{x} \quad \textbf{(14)}$$

> $yr := rhs(F00) : evalf(\%);$

$$0.50000 \ln(x)\, x + 0.25000\, x + \frac{0.75000}{x} \quad \textbf{(15)}$$

> *print*( ) :
> *printf*(" Exact solution ") :
> *print*( ) :
> *printf*(" Xi Yi") :
> *print*( ) :
> **for** i **from** 0 **to** n **do**
> $x := X[i]$:
> *printf*("%f %f \n", *evalf*($X[i]$), *evalf*(yr)) :
>
> $endtable[4, i] := evalf(yr)$:
> **end do**:
> *print*( ) :

```
Exact solution

Xi          Yi

1.000000 1.000000
1.100000 1.009200
1.200000 1.034400
1.300000 1.072400
```

```
1.400000 1.121200
1.500000 1.179100
1.600000 1.244800
1.700000 1.317200
1.800000 1.395700
1.900000 1.479500
2.000000 1.568200
```

(16)

```
> 
> # comparison table
> print( );
  printf("|          | Runge-Kutta method |    Euler method     |  Exact solution    |") :
  print( ) :
  printf("|   Xi     |         Yi         |         Yi          |         Yi         |") :
  print( ) :

  for i from 0 to n do
  printf("| %f |      %f       |      %f       |      %f       |   \n", endtable[1, i], endtable[2, i],
         endtable[3, i], endtable[4, i]) :
  end do:

  print( ) :
     printf("As a result of calculations using different methods, we can conclude that the most
     accurate is the ?????????????? method ") : print( ) :
```

```
|             |   Runge-Kutta method  |     Euler method      |
Exact solution    |

|    Xi       |         Yi            |         Yi            |
     Yi           |

|  1.000000  |       1.000000        |       1.000000        |
  1.000000        |
|  1.100000  |       1.009200        |       1.000000        |
  1.009200        |
|  1.200000  |       1.034300        |       1.018600        |
  1.034400        |
|  1.300000  |       1.072400        |       1.051900        |
  1.072400        |
|  1.400000  |       1.121200        |       1.097300        |
  1.121200        |
|  1.500000  |       1.179100        |       1.152500        |
  1.179100        |
|  1.600000  |       1.244700        |       1.216200        |
  1.244800        |
|  1.700000  |       1.317000        |       1.287200        |
  1.317200        |
|  1.800000  |       1.395800        |       1.364700        |
  1.395700        |
|  1.900000  |       1.479500        |       1.447600        |
  1.479500        |
```

```
|  2.000000  |       1.568100       |       1.535600       |
  1.568200       |

As a result of calculations using different methods, we can
conclude that the most accurate is the ??????????????? method
```

(17)

>

>