Metoda sítí

pro diferenciální rovnici 2. Śádu
y"=f(x,y,y')
a okrajové podmínky Ŭ1y1(a) + б1 y2(a) = ɔ1, Ŭ2 y1(b) + б2 y2(b) = ɔ2

Výpoļ et nelineárních rovnic je Śeġen metodou postupných iterací

```
DESite1 := proc(n, f, a, b, alfa1, alfa2, beta1, beta2, gama1, gama2, eps, y0, maxit)
    local h,  s, i, j, alfa, d1, d2, d3, p, q, F, y1, beta, x, y2, yp;
    h := evalf((b-a)/n);
    s := 99999999;
    alfa := 1/(h*h);
    beta := 1/(2*h);
    p := 1/(alfa1-beta1*3*beta);
    q := 1/(alfa2-beta2*3*beta);
    yp := [seq(0, i = 1..n+1)];
    x := [seq(a+(i-1)*h, i = 1..n+1)];
    d1 := [seq(alfa, i = 1..n-1)];
    d2 := [seq(-2*alfa, i = 1..n-1)];
    d3 := [seq(alfa, i = 1..n-1)];
    d1[1] := 0;
    d2[1] := d2[1]-4*alfa*p*beta*beta1;
    d3[1] := d3[1]+alfa*p*beta*beta1;
    d1[n-1] := d1[n-1]-alfa*q*beta*beta2;
    d2[n-1] := d2[n-1]+4*alfa*q*beta*beta2;
    d3[n-1] := 0;
    y1 := [seq(y0[i], i = 1..n+1)];
    y1[1] := (gama1-4*beta1*beta*y1[2]+beta1*beta*y1[3])*p;
    y1[n+1] := (gama2+4*beta2*beta*y1[n]-beta2*beta*y1[n-1])*q;
    print("iterace = ", 0);
    print("y = ", y1);
    j := 0;
    while s > eps and j < maxit do
        F := [seq(f(x[i+1], y1[i+1], beta*(y1[i+2]-y1[i])), i = 1..n-1)];
        F[1] := F[1]-alfa*p*gama1;
        F[n-1] := F[n-1]-alfa*q*gama2;
        # výpoļ et j +1 iterace y metodou postupných iterací
        y2 := TriDiagonalSolve(n-1, d1, d2, d3, F);
        for i from 2 to n do
            yp[i] := y2[i-1];
            end do;
        # výpoļ et krajních hodnot j +1 profilu
        yp[1] := (gama1-4*beta1*beta*yp[2]+beta1*beta*yp[3])*p;
        yp[n+1] := (gama2+4*beta2*beta*yp[n]-beta2*beta*yp[n-1])*q;
        unassign('i');
        s := sqrt(sum((yp[i]-y1[i])^2, i = 1..n+1)/sum((y1[i])^2, i = 1..n+1));
        for i from 1 to n+1 do
            y1[i] := yp[i];
        end do;
        j := j+1;
        print("iterace = ", j, "      s = ", s);
```

```
            print("y = ", y1);
        end do;
        if j ≥ maxit then print("Vyẹ erpán maximální poẹ et iterací, poẹ et iterací = ", j); end if;
        RETURN([seq([x[i], y1[i]], i = 1 ..n + 1)]);
    end proc:
        TriDiagonalSolve :=proc(n, a, d, b, f)
        local b1, f1, x, pom, i;
        b1 := [seq(0, i = 1 ..n)];
        f1 := [seq(0, i = 1 ..n)];
        x := [seq(0, i = 1 ..n)];
        b1[1] := b[1]/d[1];
        f1[1] := f[1]/d[1];
        for i from 2 to n-1 do
            pom := (d[i]-a[i]*b1[i-1]);
            b1[i] := b[i]/pom;
            f1[i] := (f[i]-a[i]*f1[i-1])/pom;
        end do;
        f1[n] := (f[n]-a[n]*f1[n-1])/(d[n]-a[n]*b1[n-1]);
        x[n] := f1[n];
        for i from n-1 by -1 to 1 do
            x[i] := f1[i]-b1[i]*x[i + 1];
        end do;
        RETURN(x);
    end:
```

výpoẹ et nelineárních rovnic Newtonovou metodou

```
DESite2 :=proc(n, f, a, b, alfa1, alfa2, beta1, beta2, gama1, gama2, eps, y0, maxit)
      local h,  s, i, j, alfa, d1, dn, dd1, dd2, dd3, d21, d31, d1n, d2n, p, q, F, y1, beta, x, dy, yp, df2 ,
    df3;
      df2 := D[2](f);
      df3 := D[3](f);
      h := evalf((b-a)/n);
      s := 99999999;
      alfa := 1/(h*h);
      beta := 1/(2*h);
      p := 1/(alfa1-beta1*3*beta);
      q := 1/(alfa2-beta2*3*beta);
      yp := [seq(0, i = 1 ..n + 1)];
      x := [seq(a + (i-1)*h, i = 1 ..n + 1)];
      dd1 := [seq(0, i = 1 ..n-1)];
      dd2 := [seq(0, i = 1 ..n-1)];
      dd3 := [seq(0, i = 1 ..n-1)];
      d21 := -2*alfa-4*alfa*p*beta*beta1;
      d31 := alfa + alfa*p*beta*beta1;
      d1n := alfa-alfa*q*beta*beta2;
      d2n := -2*alfa + 4*alfa*q*beta*beta2;
      d1 := p*beta*beta1;
      dn := q*beta*beta2;
      y1 := [seq(y0[i], i = 1 ..n + 1)];
      y1[1] := (gama1-4*beta1*beta*y1[2] + beta1*beta*y1[3])*p;
```

```
y1[n + 1] := (gama2 + 4 * beta2 * beta * y1[n] - beta2 * beta * y1[n - 1]) * q;
print("iterace = ", 0);
print("y = ", y1);
j := 0;
# Newtonova metoda
while s > eps and j < maxit do
    F := -[seq(alfa * y1[i] - 2 * alfa * y1[i + 1] + alfa * y1[i + 2] - f(x[i + 1], y1[i + 1], beta
* (y1[i + 2] - y1[i])), i = 1 ..n - 1)];
    for i from 2 to n - 2 do
        dd1[i] := alfa - df3(x[i + 1], y1[i + 1], beta * (y1[i + 2] - y1[i])) * (-beta);
        dd2[i] := -2 * alfa - df2(x[i + 1], y1[i + 1], beta * (y1[i + 2] - y1[i]));
        dd3[i] := alfa - df3(x[i + 1], y1[i + 1], beta * (y1[i + 2] - y1[i])) * (beta);
    end do;
    dd1[1] := 0;
    dd2[1] := d21 - df2(x[i], y1[i], beta * (y1[i + 1] - y1[i - 1])) - df3(x[i], y1[i], beta * (y1[i
+ 1] - y1[i - 1])) * 4 * d1 * beta;
    dd3[1] := d31 - df3(x[i], y1[i], beta * (y1[i + 1] - y1[i - 1])) * (1 - d1) * beta;
    dd1[n - 1] := d1n + df3(x[i], y1[i], beta * (y1[i + 1] - y1[i - 1])) * (1 + dn) * (beta);
    dd2[n - 1] := d2n - df2(x[i], y1[i], beta * (y1[i + 1] - y1[i - 1])) - df3(x[i], y1[i], beta
* (y1[i + 1] - y1[i - 1])) * 4 * dn * (beta);
    dd3[n - 1] := 0;
    # Śeġení  soustavy lineárních rovnic
    dy := TriDiagonalSolve(n - 1, dd1, dd2, dd3, F);
    for i from 2 to n do
        yp[i] := y1[i] + dy[i - 1];
    end do;
    # výpoļet krajních hodnot j +1 profilu
    yp[1] := (gama1 - 4 * beta1 * beta * yp[2] + beta1 * beta * yp[3]) * p;
    yp[n + 1] := (gama2 + 4 * beta2 * beta * yp[n] - beta2 * beta * yp[n - 1]) * q;
    unassign('i');
    s := sqrt(sum((yp[i] - y1[i])^2, i = 1 ..n + 1) / sum((y1[i])^2, i = 1 ..n + 1));
    for i from 1 to n + 1 do
        y1[i] := yp[i];
    end do;
    j := j + 1;
    print("iterace = ", j, "      s = ", s);
    print("y = ", y1);
end do;
if j ≥ maxit then print("Vyļerpán maximální poļet iterací, poļet iterací = ", j); end if;
RETURN([seq([x[i], y1[i]], i = 1 ..n + 1)]);
end proc:
```

PŚíklad 1 (vyuġití postupných aproximací)

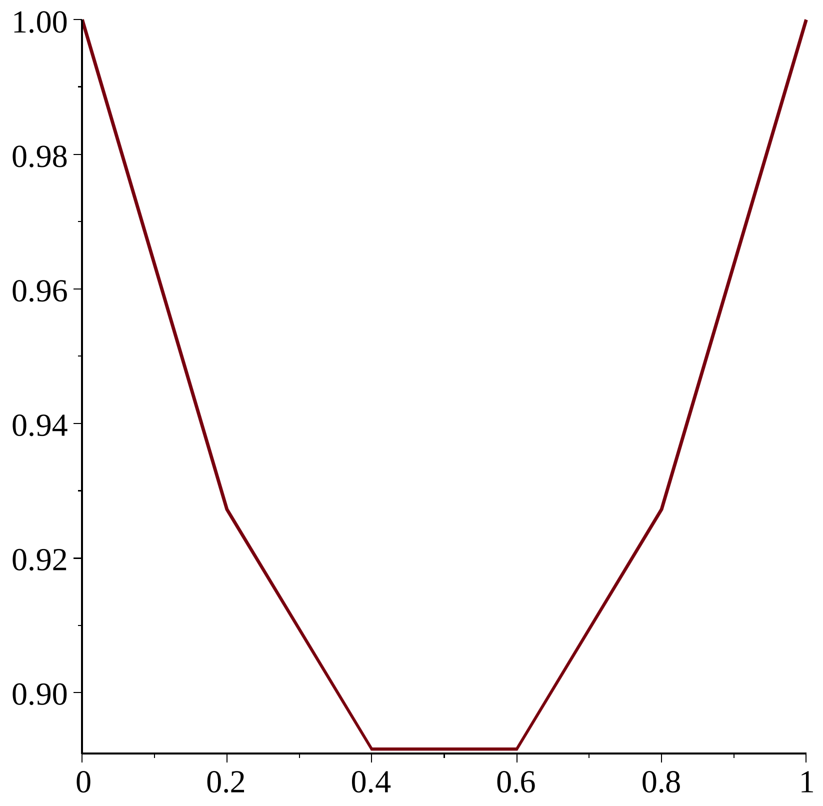
y" = y,   y(0)=1 y(1)=1

```
> f := (x, y, dy) → y;
```

$$f := (x, y, dy) \to y \tag{1.1}$$

```
> a := 0 :
  b := 1 :
  alfa1 := 1 :
  alfa2 := 1 :
  beta1 := 0 :
```

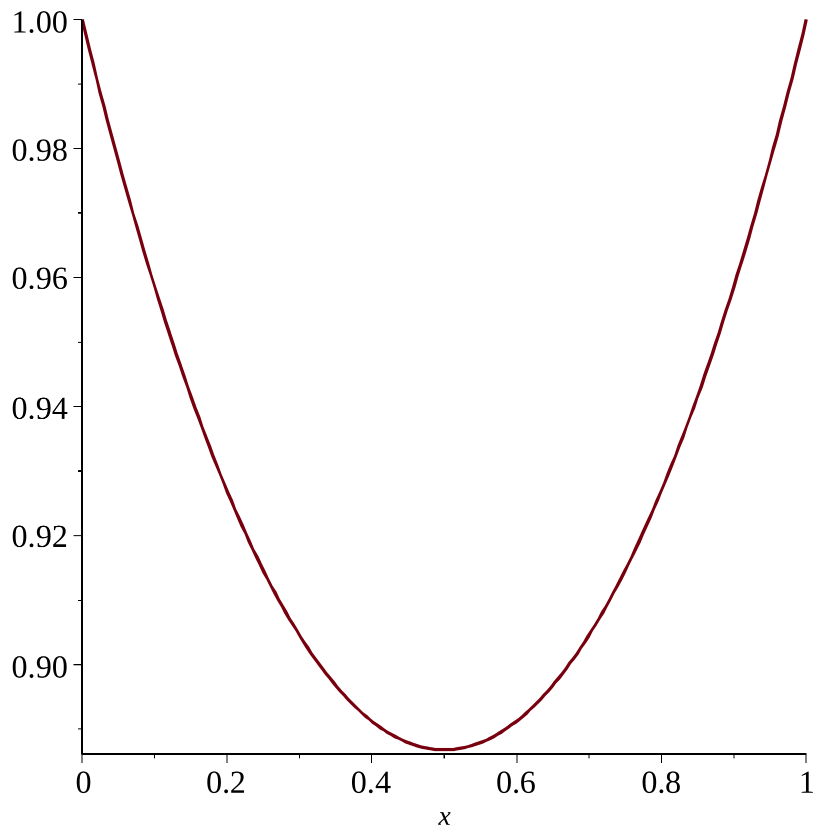
```
beta2 := 0 :
gama1 := 1 :
gama2 := 1;
eps := 0.00001 :
n := 5 :
y0 := evalf([seq(1.0, i = 1 ..n + 1)]) :

                                gama2 := 1                                    (1.2)
> v := DESite1(n, f, a, b, alfa1, alfa2, beta1, beta2, gama1, gama2, eps, y0, 15);
                              "iterace = ", 0
                     "y = ", [1., 1.0, 1.0, 1.0, 1.0, 1.]
               "iterace = ", 1, "      s = ", 0.08326663991
   "y = ", [1., 0.9200000000, 0.8800000001, 0.8800000002, 0.9200000000, 1.]
              "iterace = ", 2, "      s = ", 0.009323909467
   "y = ", [1., 0.9280000001, 0.8928000002, 0.8928000002, 0.9280000000, 1.]
             "iterace = ", 3, "      s = ", 0.0009694934279
   "y = ", [1., 0.9271680001, 0.8914560002, 0.8914560002, 0.9271680000, 1.]
             "iterace = ", 4, "      s = ", 0.0001016019537
   "y = ", [1., 0.9272550400, 0.8915968001, 0.8915968002, 0.9272550400, 1.]
            "iterace = ", 5, "      s = ", 0.00001063907251
   "y = ", [1., 0.9272459264, 0.8915820545, 0.8915820546, 0.9272459264, 1.]
            "iterace = ", 6, "      s = ", 0.000001114161703
   "y = ", [1., 0.9272468808, 0.8915835987, 0.8915835988, 0.9272468808, 1.]
v := [[0., 1.], [0.2000000000, 0.9272468808], [0.4000000000, 0.8915835987],   (1.3)
     [0.6000000000, 0.8915835988], [0.8000000000, 0.9272468808], [1.000000000,
     1.]]
>
> # Graf funkce y(x)
> plot(v);
```



```
> res:=dsolve({diff(y(x),x$2)=y(x),y(0)=1,y(1)=1},y(x));
```

$$res := y(x) = -\frac{\left(e^{-1}-1\right) e^{x}}{e-e^{-1}} + \frac{(-1+e)\, e^{-x}}{e-e^{-1}} \qquad \textbf{(1.4)}$$

```
> plot(rhs(res),x=0..1);
```



```
> # Tabulka hodnot funkce y1(x)
> linalg[matrix](v);
```

$$\left[\begin{array}{cc} 0. & 1. \\ 0.2000000000 & 0.9272468808 \\ 0.4000000000 & 0.8915835987 \\ 0.6000000000 & 0.8915835988 \\ 0.8000000000 & 0.9272468808 \\ 1.000000000 & 1. \end{array}\right] \tag{1.5}$$

PŚíklad 1 (vyuģití Newtonovy metody)

y'' = y,   y(0)=1 y(1)=1

> $f := (x, y, dy) \to y;$

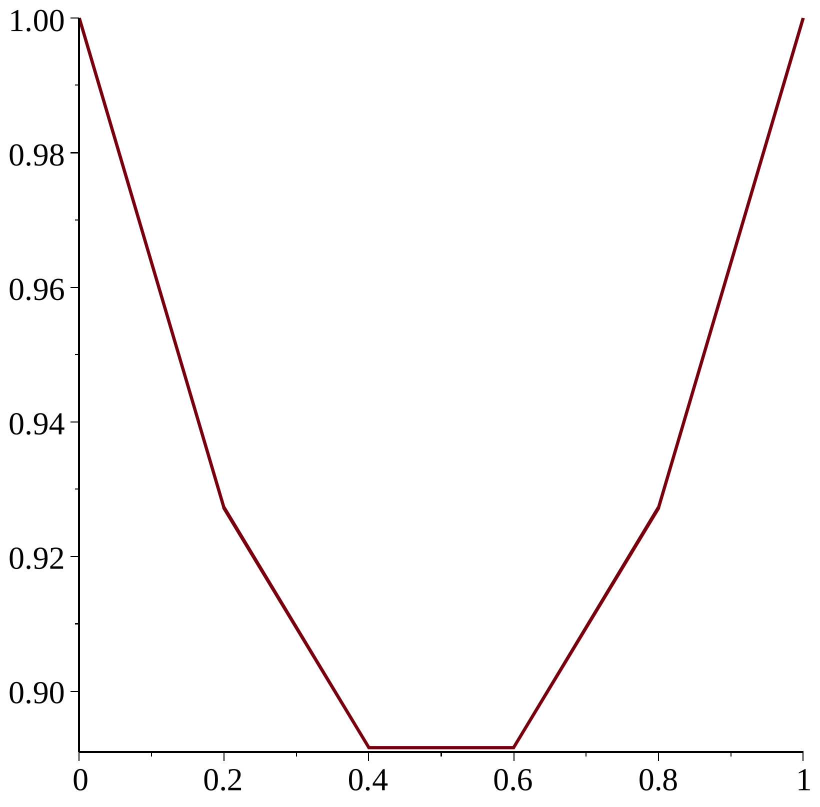
$$f := (x, y, dy) \to y \tag{2.1}$$

> $a := 0:$
$b := 1:$
$alfa1 := 1:$
$alfa2 := 1:$
$beta1 := 0:$

```
beta2 := 0 :
gama1 := 1 :
gama2 := 1;
eps := 0.00001 :
n := 5 :
y0 := evalf ( [seq(1.0, i = 1 ..n + 1 ) ]) :
                                   gama2 := 1                                        (2.2)
```

```
> v := DESite2(n, f, a, b, alfa1, alfa2, beta1, beta2, gama1, gama2, eps, y0, 15);
                              "iterace = ", 0
                      "y = ", [1., 1.0, 1.0, 1.0, 1.0, 1.]
                 "iterace = ", 1, "      s = ", 0.07538164533
   "y = ", [1., 0.9272467903, 0.8915834522, 0.8915834522, 0.9272467903, 1.]
                "iterace = ", 2, "      s = ", 2.455000810 10^-10
   "y = ", [1., 0.9272467903, 0.8915834526, 0.8915834526, 0.9272467903, 1.]
v := [[0., 1.], [0.2000000000, 0.9272467903], [0.4000000000, 0.8915834526],          (2.3)
     [0.6000000000, 0.8915834526], [0.8000000000, 0.9272467903], [1.000000000,
     1.]]
```

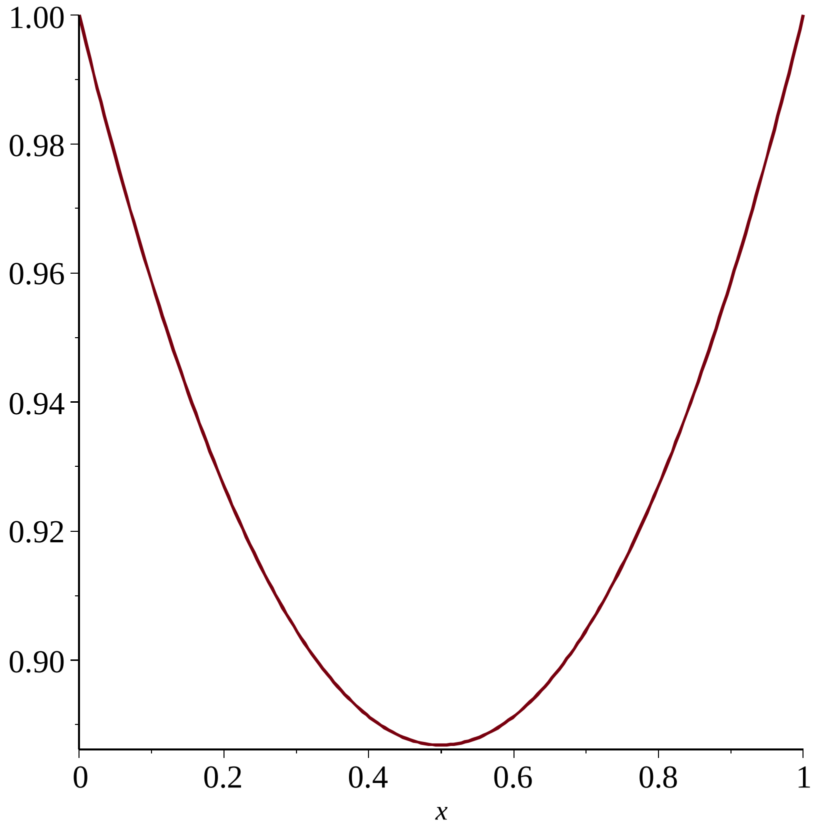
```
> # Graf funkce y(x)
> plot(v);
```



```
> res:=dsolve({diff(y(x),x$2)=y(x),y(0)=1,y(1)=1},y(x));
```

$$res := y(x) = -\frac{\left(e^{-1}-1\right)e^{x}}{e-e^{-1}} + \frac{(-1+e)\,e^{-x}}{e-e^{-1}} \qquad (2.4)$$

```
> plot(rhs(res),x=0..1);
```



```
> # Tabulka hodnot funkce y1(x)
> linalg[matrix](v);
```

$$\begin{bmatrix} 0. & 1. \\ 0.2000000000 & 0.9272467903 \\ 0.4000000000 & 0.8915834526 \\ 0.6000000000 & 0.8915834526 \\ 0.8000000000 & 0.9272467903 \\ 1.000000000 & 1. \end{bmatrix} \qquad (2.5)$$