```
restart;
```

Aufgabe 1: Ausgleichsrechnung

```
with(plots):
with(Statistics):
X := ⟨90, 180, 270, 360, 450, 540, 630, 720⟩;
```

$$\begin{bmatrix} 90 \\ 180 \\ 270 \\ 360 \\ 450 \\ 540 \\ 630 \\ 720 \end{bmatrix} \tag{1.1}$$

```
Y := ⟨299.72, 723.33, 1178.98, 1711.08, 2161.69, 2260.98, 2418.65, 2502.74⟩;
```

$$\begin{bmatrix} 299.72 \\ 723.33 \\ 1178.98 \\ 1711.08 \\ 2161.69 \\ 2260.98 \\ 2418.65 \\ 2502.74 \end{bmatrix} \tag{1.2}$$

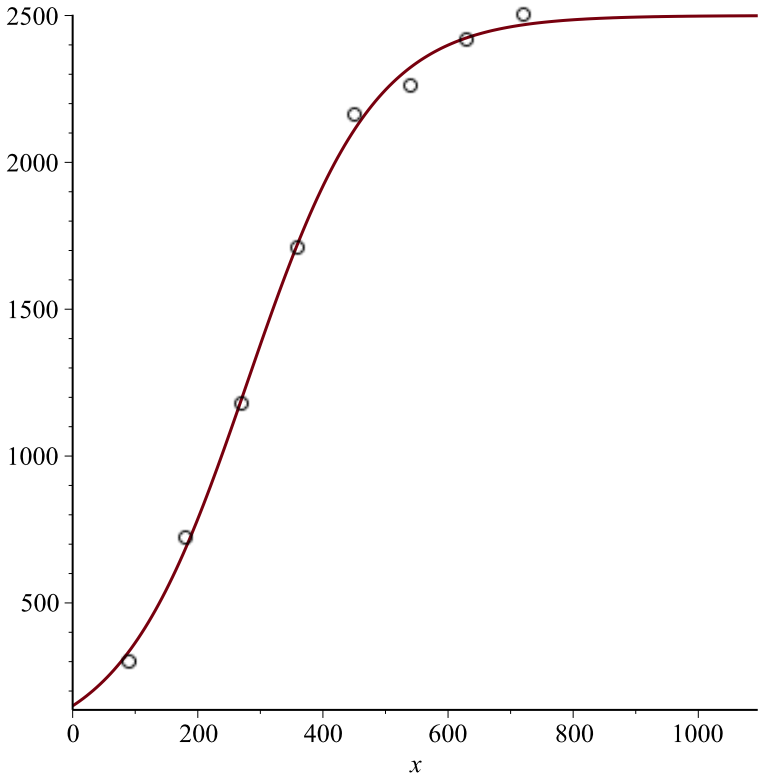
```
f := 2500·e^(a/100·x) / (b + e^(a/100·x));
```

$$\frac{2500\, e^{\frac{1}{100} a x}}{b + e^{\frac{1}{100} a x}} \tag{1.3}$$

```
fit := Fit(f, X, Y, x);
```

$$\frac{2500\, e^{0.00987599839115128\, x}}{15.7280539249178 + e^{0.00987599839115128\, x}} \tag{1.4}$$

```
display(plot(fit, x = 0..3·365), pointplot(X, Y, symbol = circle, symbolsize = 15));
```



Aufgabe 2: Gleichungssysteme

$$solve\left(\left[x^2+y^2=16, x+y=p\right], [x, y], Explicit\right);$$

$$\left[\left[x=\frac{1}{2}p-\frac{1}{2}\sqrt{-p^2+32}, y=\frac{1}{2}p+\frac{1}{2}\sqrt{-p^2+32}\right], \left[x=\frac{1}{2}p+\frac{1}{2}\sqrt{-p^2+32}, y=\frac{1}{2}p-\frac{1}{2}\sqrt{-p^2+32}\right]\right] \quad \textbf{(2.1)}$$

Aufgabe 3: Prozeduren

```
numbers := proc(n)
 local i;
 for i to n do
  print(i);
 end do; # Anmerkung: end do ≙ od
end proc;
```

```
proc(n) local i; for i to n do print(i) end do end proc
```
(3.1)

```
numbers(4);
```

1
2
3
4
(3.2)

Aufgabe 4: Prozeduren

a)

```
maxima := proc(f)
          local c, z, el;
          c := 0;
          z := [fsolve(f'(x) = 0, x)];
          for el in z do
              if f''(el) < 0 then c := c + 1;
              end if;
          end do;
          return c;
 end proc:
```

b)

```
f := x→ -x^2;
```
$x \to -x^2$ (4.2.1)

```
g := x→ -x^4;
```
$x \to -x^4$ (4.2.2)

```
h := x→ -x^4 - x^3 + 10·x^2 + 3;
```
$x \to -x^4 - x^3 + 10\,x^2 + 3$ (4.2.3)

```
i := x→sin(x);
```
$x \to \sin(x)$ (4.2.4)

```
maxima(f);
```
1 (4.2.5)

✓ Ergebnis ist korrekt.

```
maxima(g);
```
0 (4.2.6)

Hier greift das hinreichende Kriterium (2. Ableitung) nicht.

```
maxima(h);
```
2 (4.2.7)

✓ Ergebnis ist korrekt.

```
maxima(i);
```

$$1 \tag{4.2.8}$$

Hier gibt es unendlich viele Extrema, allerdings findet fsolve nur eines davon.

Aufgabe 5: Sequenzen

a)

```
a := 3, 4, 5;
```

$$3, 4, 5 \tag{5.1.1}$$

```
b := NULL, 1, 9;
```

$$1, 9 \tag{5.1.2}$$

```
c := a, b;
```

$$3, 4, 5, 1, 9 \tag{5.1.3}$$

```
c := c, 42;
```

$$3, 4, 5, 1, 9, 42 \tag{5.1.4}$$

b)

```
myfacs :=proc(n)
 local L, i;
 L := NULL;
 for i while i! < n do
  L := L, i!;
 end do; # Anmerkung: end do ≙ od
 return L;
end proc;
```

```
proc(n)
    local L, i;
    L := NULL; for i while factorial(i) < n do L := L, factorial(i) end do; return L
end proc
```

(5.2.1)

```
myfacs(7);
```

$$1, 2, 6 \tag{5.2.2}$$

Aufgabe 6: Polynome

```
g := x→ -x^4;
```

$$x \to -x^4 \tag{6.1}$$

```
h := x→ -x^4 - x^3 + 10·x^2 + 3;
```

$$x \to -x^4 - x^3 + 10\,x^2 + 3 \tag{6.2}$$

a)

```
if sqrt(4) > 0 then 1 else 0 end;
```

1 **(6.1.1)**

```
if sqrt(3) > 0 then 1 else 0 end;
```

```
Error, cannot determine if this expression is true or false:
0 < 3^(1/2)
```

```
if is(sqrt(4) > 0) then 1 else 0 end;
```

1 **(6.1.2)**

```
if is(sqrt(3) > 0) then 1 else 0 end;
```

1 **(6.1.3)**

b)

```
maxima := proc(f)
          local L, z, el;
          L := NULL;
          z := [fsolve(f'(x) = 0, x)];
          for el in z do
              if f''(el) < 0 then L := L, el;
              end if;
          end do;
          return [L];
 end proc;
```

```
proc(f)                                                                          (6.2.1)
    local L, z, el;
    L := NULL;
    z := [fsolve(diff(f(x), x) = 0, x)];
    for el in z do if eval(diff(f(x), x, x), x = el) < 0 then L := L, el end if end do;
    return [L]
end proc
```

```
maxima(h);
```

[-2.642294643, 1.892294643] **(6.2.2)**

c)

```
maxima := proc(f)
          local L, z, el;
          L := NULL;
          if degree(f(x)) > 5 then
            z := [fsolve(f'(x) = 0, x)];
          else
            z := [solve(f'(x) = 0, x, DropMultiplicity)];
          end if;
```

```
            for el in z do
                if is(f''(el) < 0) then L := L, el;
                end if;
            end do;
            return [L];
end proc;
```

```
proc(f)                                                                    (6.3.1)
    local L, z, el;
    L := NULL;
    if 5 < degree(f(x)) then
        z := [fsolve(diff(f(x), x) = 0, x)]
    else
        z := [solve(diff(f(x), x) = 0, x, DropMultiplicity)]
    end if;
    for el in z do if is(eval(diff(f(x), x, x), x = el) < 0) then L := L, el end if end do;
    return [L]
end proc
```

```
maxima(h);
```

$$\left[-\frac{3}{8}+\frac{1}{8}\sqrt{329},\ -\frac{3}{8}-\frac{1}{8}\sqrt{329}\right] \qquad (6.3.2)$$

d)

```
maxima := proc(f)
            local L, z, el, n, ii;
            L := NULL;
            n := degree(f(x));
            if n ≤ 5 then
             z := [solve(f'(x) = 0, x, DropMultiplicity)];
             for el in z do
              for ii from 2 to n do
               if D^(ii)(f)(el) ≠ 0 then
                if ii mod 2 = 0 and is(D^(ii)(f)(el) < 0) then
                 L := L, el;
                end if;
                break;
               end if;
              end do;
             end do;
            else
             z := [fsolve(f'(x) = 0, x)];
             for el in z do
                if is(f''(el) < 0) then L := L, el;
                end if;
             end do;
```

```
            end if;
            return [L];
 end proc;

proc(f)                                                                    (6.4.1)
    local L, z, el, n, ii;
    L := NULL;
    n := degree(f(x));
    if n <= 5 then
        z := [solve(diff(f(x), x) = 0, x, DropMultiplicity)];
        for el in z do
            for ii from 2 to n do
                if `@@`(D, ii)(f)(el) <> 0 then
                    if mod(ii, 2) = 0 and
                    is(`@@`(D, ii)(f)(el) < 0) then
                        L := L, el
                    end if;
                    break
                end if
            end do
        end do
    else
        z := [fsolve(diff(f(x), x) = 0, x)];
        for el in z do
            if is(eval(diff(f(x), x, x), x = el) < 0) then L := L, el end if
        end do
    end if;
    return [L]
end proc
```

e)

```
maxima(g);
```

$$[0] \qquad (6.5.1)$$

```
maxima(h);
```

$$\left[-\frac{3}{8}+\frac{1}{8}\sqrt{329}, -\frac{3}{8}-\frac{1}{8}\sqrt{329}\right] \qquad (6.5.2)$$

f)

Nullstellen sind im Allgemeinen nur für Polynome von Grad ≤ 4 analytisch bestimmbar. Die Ableitung eines Polynoms von Grad 5 ist ein Polynom von Grad 4.

Aufgabe 7 (Intensivaufgabe): Bildverarbeitung

```
restart;
```

a)

```
with(ImageTools) :
```

c)

```
org := Read("d:/IMS_tmp/Image1.jpg");
```

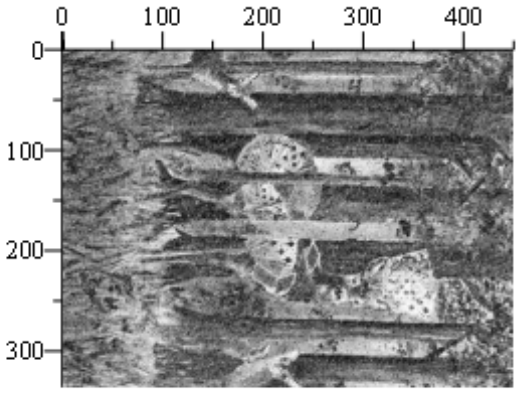
$$\begin{bmatrix} 1..336 \text{ x } 1..450 \text{ x } 1..3 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(7.2.1)}$$

d)

```
gray := ToGrayscale(org);
```

$$\begin{bmatrix} 1..336 \text{ x } 1..450 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(7.3.1)}$$

```
Preview(gray);
```

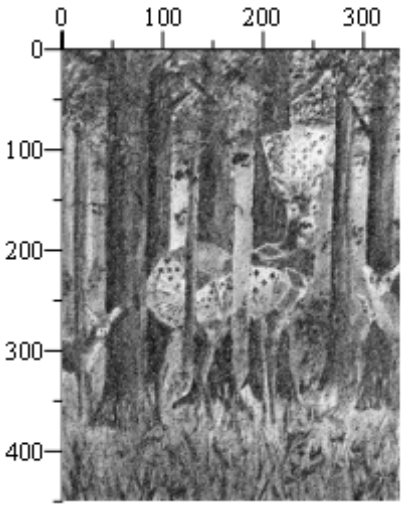


e)

$rot := Rotate(gray, -90);$

$$\begin{bmatrix} 1..450\ x\ 1..336\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: C\_order \end{bmatrix} \quad \textbf{(7.4.1)}$$

$Preview(rot);$



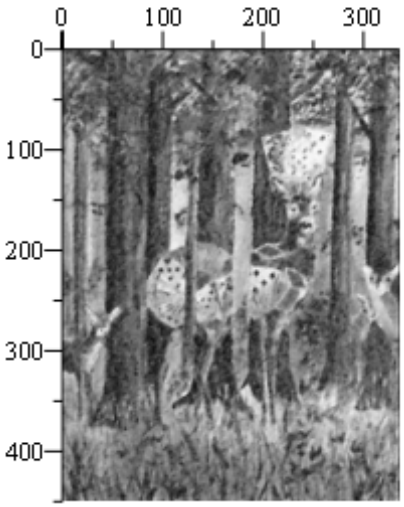
f)

$filtered := Create(Height(rot), Width(rot))$;

$$\begin{bmatrix} 1..450\ x\ 1..336\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: C\_order \end{bmatrix} \quad \textbf{(7.5.1)}$$

```
for i from 2 to Height(filtered) − 1 do
  for ii from 2 to Width(filtered) − 1 do
    filtered[i, ii] := Statistics[Median]([rot[i − 1, ii − 1], rot[i − 1, ii], rot[i − 1, ii + 1], rot[i, ii
        − 1], rot[i, ii], rot[i, ii + 1], rot[i + 1, ii − 1], rot[i + 1, ii], rot[i + 1, ii + 1]]) :
  od:
od:

Preview(filtered);
```



View(*filtered*); # *Schönere Ausgabe, öffnet aber extra Fenster.*

g)

$$SobelX := \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix};$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \textbf{(7.6.1)}$$

SobelY := *LinearAlgebra*[*Transpose*](*SobelX*);

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \textbf{(7.6.2)}$$

GX := *Convolution*(*filtered*, *SobelX*);

$$\begin{bmatrix} 1..450\ x\ 1..336\ Array \\ Data\ Type:\ float_8 \\ Storage:\ rectangular \\ Order:\ C\_order \end{bmatrix} \quad \textbf{(7.6.3)}$$

$GY := Convolution(filtered, SobelY);$

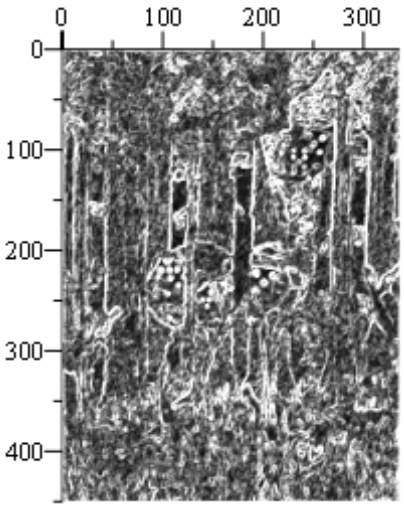
$$\begin{bmatrix} 1..450\ x\ 1..336\ Array \\ Data\ Type:\ float_8 \\ Storage:\ rectangular \\ Order:\ C\_order \end{bmatrix} \quad \textbf{(7.6.4)}$$

h)

$GS := \text{sqrt}\sim(GX^2 + GY^2);$

$$\begin{bmatrix} 1..450\ x\ 1..336\ Array \\ Data\ Type:\ float_8 \\ Storage:\ rectangular \\ Order:\ C\_order \end{bmatrix} \quad \textbf{(7.7.1)}$$

$Preview(GS);$



$View(GS);$