

Appendix B: Functions as Mathcad Programs

Although these functions are now built into Mathcad for you to use in any of your documents, these prototype Mathcad programs may help you to see ways in which you could write new functions.

filterNaN(v)

```

filterNaN1(data) :=
    data ← (data)T
    trimi ← 0
    for i ∈ 0..cols(data) - 1
        if IsScalar(matchNaN(data⟨i⟩))
            | trimmed⟨trimi⟩ ← data⟨i⟩
            | trimi ← trimi + 1
    return (trimmed)T

```

localmax(Data,n)

```

localmax1(Data, n) :=
  count ← 0
  error("window must be integer greater than or equal to 1" ) if n < 1 ∨ (trunc
  error("window cannot be wider than half of data range" ) if  $\frac{n}{2} > \text{rows}(\text{Data})$ 
  Δ1 ← n
  Δ2 ← rows(Data) - 1 - n
  for i ∈ 0.. rows(Data) - 1
    Start ← i - n if i ≥ Δ1
    Start ← 0 otherwise
    Stop ← i + n if i ≤ Δ2
    Stop ← rows(Data) - 1 otherwise
    compoints ← submatrix(Data, Start, Stop, 1, 1)
    if  $\sum(\text{compoints} > \text{Data}_{i,1}) = 0$ 
      Valcount,0 ← Datai,0
      Valcount,1 ← Datai,1
      count ← count + 1
  return Val

```

localmin(Data,n)

```
localmin1(Data,n) := | count ← 0
                      | error("window must be integer greater than or equal to 1") if n < 1 ∨ (tr
                      | error("window cannot be wider than half of data range") if  $\frac{n}{2} > \text{rows}(\text{Da}$ 
                      |
                      |  $\Delta 1 \leftarrow n$ 
                      |  $\Delta 2 \leftarrow \text{rows}(\text{Data}) - 1 - n$ 
                      | for i ∈ 0.. rows(Data) - 1
                      |   | Start ← i - n if i ≥ Δ1
                      |   | Start ← 0 otherwise
                      |   | Stop ← i + n if i ≤ Δ2
                      |   | Stop ← rows(Data) - 1 otherwise
                      |   | compoints ← submatrix(Data, Start, Stop, 1, 1)
                      |   | if  $\sum(\text{compoints} < \text{Data}_{i,1}) = 0$ 
                      |   |   | Valcount,0 ← Datai,0
                      |   |   | Valcount,1 ← Datai,1
                      |   |   | count ← count + 1
                      | return Val
```

markNaN(data, index)

```
markNaN1(data, index) := | if IsArray(index0)
                          |   | "case where index is an array of nested matrix indices"
                          |   | for i ∈ 0.. rows(index) - 1
                          |   |   | data(indexi)0, (indexi)1 ← NaN
                          |   | otherwise
                          |   |   | "case where index is a vector of vector indices"
                          |   |   | for i ∈ 0.. last(index)
                          |   |   |   | dataindexi ← NaN
                          | data
```

matchNaN(v)

```
matchNaN1(v) := | match ← 0
                  | for i ∈ 0..last(v)
                  |   if IsNaN(vi)
                  |     | indexmatch ← i
                  |     | match ← match + 1
                  | index
```

order(v)

```
order1(x) := | sorted ← sort(x)
              | ind ← 0
              | for i ∈ 0..last(x)
              |   | m ← match(xi, sorted)
              |   | if длина(m) > 1
              |   |   | foundind ← xi
              |   |   | repeats ← длина(match(xi, found))
              |   |   | orderi ← mrepeats-1
              |   |   | ind ← ind + 1
              |   | orderi ← m0 otherwise
              | order
```

percentile(v,p)

```
percentile1(x,p) := | n ← rows(x)
                    | pos1 ← (p·n + p) - 1
                    | position ← | 0 if pos1 ≤ 0
                    |           | pos1 if 0 < pos1 < n - 1
                    |           | n - 1 if pos1 ≥ n - 1
                    | between_entries ← position - floor(position)
                    | sorted ← sort(x)
                    | percentile ← sortedfloor(position) + between_entries · (sortedceil(position) - sortedfloor(position))
                    | percentile
```

polyiter(vx,vy,x,N,ε)

```
polyiter1(vx,vy,x,N,ε) := for k ∈ 0..N-1
  Y1,k ←  $\frac{vy_k \cdot (x - vx_{k+1}) - vy_{k+1} \cdot (x - vx_k)}{vx_k - vx_{k+1}}$ 
  Converged ← 0
  for m ∈ 2..N
    if Converged = 0
      mmax ← m
      k ← 0
      while k ≤ N - m
        if Converged = 0
          Ym,k ←  $\frac{Y_{m-1,k} \cdot (x - vx_{k+m}) - Y_{m-1,k+1} \cdot (x - vx_k)}{vx_k - vx_{k+m}}$ 
          y ← Ym,k
          Converged ← 1 if (|Ym,k - Ym-1,k| < ε) ∧ k = 0
        k ← k + 1
      (Converged)
      mmax
      y
```

qqplot(v1,[v2/"distribution"])

```
qqplot1(y1, y2) := error("The first argument must be a vector" ) if (IsArray(y1)) = 0
                  error("The second argument must be a vector or a recognized distribution" ) if (
if IsArray(y2)
    len1 ← длина(y1)
    len2 ← длина(y2)
    pts ← if(len1 < len2, len1, len2)
    for j ∈ 0.. pts - 1
        
$$p_j \leftarrow \frac{\frac{100 \cdot j}{pts - 1} + 0.3175}{100 + 0.365}$$

        Plotj,0 ← percentile(y1, pj)
        Plotj,1 ← percentile(y2, pj)
if IsString(y2)
    pts ← длина(y1)
    for j ∈ 0.. pts - 1
        
$$p_j \leftarrow \frac{\frac{100 \cdot j}{pts - 1} + 0.3175}{100 + 0.365}$$

        if y2 = "weibull"
            Plotj,0 ← ln(percentile(y1, pj))
            Plotj,1 ← ln(-ln(1 - pj))
        otherwise
            Plotj,0 ← percentile(y1, pj)
            Plotj,1 ← qnorm(pj, 0, 1)
Plot
```

Rank(v)

```
Rankdemo1(x) := sorted ← sort(x)
                for i ∈ 0.. длина(x) - 1
                    m ← match(xi, sorted) + 1
                    ranki ← mean(m)
                rank
```

Supsmooth(v)

```
LocLin(x, y, n) := | medsmooth(x, n) if mod(n, 2) = 0
                   | otherwise
                   | sy ← y
                   | N ← last(x)
                   | for i ∈  $\frac{n-1}{2} .. N - \left(\frac{n-1}{2}\right)$ 
                   |   | xtemp ← submatrix(x, i -  $\frac{n-1}{2}$ , i +  $\frac{n-1}{2}$ , 0, 0)
                   |   | ytemp ← submatrix(y, i -  $\frac{n-1}{2}$ , i +  $\frac{n-1}{2}$ , 0, 0)
                   |   | m ← slope(xtemp, ytemp)
                   |   | b ← intercept(xtemp, ytemp)
                   |   | syi ← m · xi + b
                   | sy
```

Thielecoeff(vx, vy)

```
Thielecoeff1(x, y) := | ε ← 10-18
                      | n ← rows(x)
                      | for i ∈ 0 .. n - 1
                      |   | Mi ← yi
                      |   | for j ∈ 1 .. n - 1
                      |   |   | for i ∈ j .. n - 1
                      |   |   |   | dM ← Mi - Mj-1
                      |   |   |   | dM ← ε if |dM| < ε
                      |   |   |   | Mi ←  $\frac{x_i - x_{j-1}}{dM}$ 
                      |   | for i ∈ 0 .. n - 1
                      |   |   | ci,0 ← xi
                      |   |   | ci,1 ← Mi
                      | return c
```

trim(vdata, vindex)

```
trim1(data, index) := | data ← dataT  
                      | trimi ← 0  
                      | for i ∈ 0.. cols(data) - 1  
                      |   on error match(i, index)  
                      |     | trimmed⟨trimi⟩ ← data⟨i⟩  
                      |     | trimi ← trimi + 1  
                      | return trimmedT
```

VSmooth(v, w)

```
Smooth(x, w) := | maxit ← 100  
                | sx ← medsmooth(x, w)  
                | sx2 ← medsmooth(sx, w)  
                | counter ← 1  
                | while [1 - (sx = sx2)] · (counter < maxit)  
                |   | sx ← sx2  
                |   | sx2 ← medsmooth(sx, w)  
                |   | counter ← counter + 1  
                | sx2
```

```
VSmooth1(x, W) := | n ← длина(W)  
                  | ax ← x  
                  | for i ∈ 1.. n  
                  |   ax ← Smooth(ax, Wi-1)  
                  | ax
```

$(n) \neq n$

- 1

$\text{mc}(n) \neq n$

ta) - 1

$\text{ed}_{\text{floor}(\text{position})}$)

$$\text{IsArray}(y2) \vee \text{IsString}(y2) = 0$$

