mathcad data analysis extension pack

APPENDICES

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)

```
$\mathrm{filterNaN1}(\mathrm{data}) :=$
    $\mathrm{data} \leftarrow (\mathrm{data})^{T}$
    $\mathrm{trimi} \leftarrow 0$
    for $i \in 0 .. \mathrm{cols}(\mathrm{data}) - 1$
      if $\mathrm{IsScalar}\left(\mathrm{matchNaN}\left(\mathrm{data}^{\langle i \rangle}\right)\right)$
          $\mathrm{trimmed}^{\langle \mathrm{trimi} \rangle} \leftarrow \mathrm{data}^{\langle i \rangle}$
          $\mathrm{trimi} \leftarrow \mathrm{trimi} + 1$
    return $(\mathrm{trimmed})^{T}$
```

localmax(Data,n)

```
$\mathrm{localmax1}(\mathrm{Data}, n) :=$
    count ← 0
    error("window must be integer greater than or equal to 1")  if  $n < 1 \vee (\mathrm{trunc}(n) \neq n)$
    error("window cannot be wider than half of data range")  if  $\frac{n}{2} > \mathrm{rows}(\mathrm{Data}) - 1$
    $\Delta 1 \leftarrow n$
    $\Delta 2 \leftarrow \mathrm{rows}(\mathrm{Data}) - 1 - n$
    for  $i \in 0 .. \mathrm{rows}(\mathrm{Data}) - 1$
        $\mathrm{Start} \leftarrow i - n$  if  $i \geq \Delta 1$
        Start ← 0  otherwise
        $\mathrm{Stop} \leftarrow i + n$  if  $i \leq \Delta 2$
        $\mathrm{Stop} \leftarrow \mathrm{rows}(\mathrm{Data}) - 1$  otherwise
        compoints ← submatrix(Data, Start, Stop, 1, 1)
        if  $\sum \left(\mathrm{compoints} > \mathrm{Data}_{i,1}\right) = 0$
            $\mathrm{Val}_{\mathrm{count},0} \leftarrow \mathrm{Data}_{i,0}$
            $\mathrm{Val}_{\mathrm{count},1} \leftarrow \mathrm{Data}_{i,1}$
            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  $n < 1 \vee (\mathrm{trunc}(n) \neq n)$
  error("window cannot be wider than half of data range")  if  $\frac{n}{2} > \mathrm{rows}(Data) - 1$
  $\Delta 1 \leftarrow n$
  $\Delta 2 \leftarrow \mathrm{rows}(Data) - 1 - n$
  for  $i \in 0 .. \mathrm{rows}(Data) - 1$
    $Start \leftarrow i - n$  if  $i \geq \Delta 1$
    $Start \leftarrow 0$  otherwise
    $Stop \leftarrow i + n$  if  $i \leq \Delta 2$
    $Stop \leftarrow \mathrm{rows}(Data) - 1$  otherwise
    $compoints \leftarrow \mathrm{submatrix}(Data, Start, Stop, 1, 1)$
    if  $\sum\left(compoints < Data_{i,1}\right) = 0$
      $Val_{count,0} \leftarrow Data_{i,0}$
      $Val_{count,1} \leftarrow Data_{i,1}$
      $count \leftarrow count + 1$
  return  Val
```

markNaN(data, index)

```
markNaN1(data, index) :=
  if  $\mathrm{IsArray}\left(index_0\right)$
    "case where index is an array of nested matrix indices"
    for  $i \in 0 .. \mathrm{rows}(index) - 1$
      $data_{\left(index_i\right)_0, \left(index_i\right)_1} \leftarrow NaN$
  otherwise
    "case where index is a vector of vector indices"
    for  $i \in 0 .. \mathrm{last}(index)$
      $data_{index_i} \leftarrow NaN$
  data
```

matchNaN(v)

```
matchNaN1(v) := | match ← 0
                | for  i ∈ 0 .. last(v)
                |   if  IsNaN($v_i$)
                |       | $index_{match}$ ← i
                |       | match ← match + 1
                | index
```

order(v)

```
order1(x) := | sorted ← sort(x)
             | ind ← 0
             | for  i ∈ 0 .. last(x)
             |   | m ← match($x_i$, sorted)
             |   | if  length(m) > 1
             |   |     | $found_{ind}$ ← $x_i$
             |   |     | repeats ← length(match($x_i$, found))
             |   |     | $order_i$ ← $m_{repeats-1}$
             |   |     | ind ← ind + 1
             |   | $order_i$ ← $m_0$  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 ← $sorted_{floor(position)}$ + between_entries·($sorted_{ceil(position)}$ − $sorted_{floor(position)}$)
                    | percentile
```

polyiter(vx,vy,x,N,ε)

```
$\text{polyiter1}(vx, vy, x, N, \varepsilon) :=$
  for  $k \in 0 .. N - 1$
    $Y_{1,k} \leftarrow \frac{vy_k \cdot \left(x - vx_{k+1}\right) - vy_{k+1} \cdot \left(x - vx_k\right)}{vx_k - vx_{k+1}}$
  $\text{Converged} \leftarrow 0$
  for  $m \in 2 .. N$
    if  $\text{Converged} = 0$
      $m_{max} \leftarrow m$
      $k \leftarrow 0$
      while  $k \le N - m$
        if  $\text{Converged} = 0$
          $Y_{m,k} \leftarrow \frac{Y_{m-1,k} \cdot \left(x - vx_{k+m}\right) - Y_{m-1,k+1} \cdot \left(x - vx_k\right)}{vx_k - vx_{k+m}}$
          $y \leftarrow Y_{m,k}$
          $\text{Converged} \leftarrow 1 \text{  if } \left(\left|Y_{m,k} - Y_{m-1,k}\right| < \varepsilon\right) \wedge k = 0$
        $k \leftarrow k + 1$
  $\begin{pmatrix} \text{Converged} \\ m_{max} \\ y \end{pmatrix}$
```

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  (IsArray(y2) ∨ IsString(y2)) = 0
  if IsArray(y2)
    len1 ← length(y1)
    len2 ← length(y2)
    pts ← if(len1 < len2, len1, len2)
    for j ∈ 0 .. pts − 1
      $p_j \leftarrow \dfrac{\dfrac{100 \cdot j}{pts - 1} + 0.3175}{100 + 0.365}$
      $Plot_{j,0} \leftarrow percentile(y1, p_j)$
      $Plot_{j,1} \leftarrow percentile(y2, p_j)$
  if IsString(y2)
    pts ← length(y1)
    for j ∈ 0 .. pts − 1
      $p_j \leftarrow \dfrac{\dfrac{100 \cdot j}{pts - 1} + 0.3175}{100 + 0.365}$
      if y2 = "weibull"
        $Plot_{j,0} \leftarrow \ln(percentile(y1, p_j))$
        $Plot_{j,1} \leftarrow \ln(-\ln(1 - p_j))$
      otherwise
        $Plot_{j,0} \leftarrow percentile(y1, p_j)$
        $Plot_{j,1} \leftarrow qnorm(p_j, 0, 1)$
  Plot
```

Rank(v)

```
Rankdemo1(x) :=
  sorted ← sort(x)
  for i ∈ 0 .. length(x) − 1
    $m \leftarrow match(x_i, sorted) + 1$
    $rank_i \leftarrow mean(m)$
  rank
```

Supsmooth(v)

```
$LocLin(x,y,n) :=$
    $medsmooth(x,n)$  if  $mod(n,2) = 0$
    otherwise
        $sy \leftarrow y$
        $N \leftarrow last(x)$
        for  $i \in \frac{n-1}{2} .. N - \left(\frac{n-1}{2}\right)$
            $xtemp \leftarrow submatrix\left(x, i - \frac{n-1}{2}, i + \frac{n-1}{2}, 0, 0\right)$
            $ytemp \leftarrow submatrix\left(y, i - \frac{n-1}{2}, i + \frac{n-1}{2}, 0, 0\right)$
            $m \leftarrow slope(xtemp, ytemp)$
            $b \leftarrow intercept(xtemp, ytemp)$
            $sy_i \leftarrow m \cdot x_i + b$
        $sy$
```

Thielecoeff(vx,vy)

```
$Thielecoeff1(x,y) :=$
    $\varepsilon \leftarrow 10^{-18}$
    $n \leftarrow rows(x)$
    for  $i \in 0 .. n - 1$
        $M_i \leftarrow y_i$
    for  $j \in 1 .. n - 1$
        for  $i \in j .. n - 1$
            $dM \leftarrow M_i - M_{j-1}$
            $dM \leftarrow \varepsilon$  if  $|dM| < \varepsilon$
            $M_i \leftarrow \frac{x_i - x_{j-1}}{dM}$
    for  $i \in 0 .. n - 1$
        $c_{i,0} \leftarrow x_i$
        $c_{i,1} \leftarrow M_i$
    return  $c$
```

trim(vdata,vindex)

```
$\mathrm{trim1}(\mathrm{data}, \mathrm{index}) :=$
    $\mathrm{data} \leftarrow \mathrm{data}^{T}$
    $\mathrm{trimi} \leftarrow 0$
    for $i \in 0 .. \mathrm{cols}(\mathrm{data}) - 1$
      on error $\mathrm{match}(i, \mathrm{index})$
            $\mathrm{trimmed}^{\langle \mathrm{trimi} \rangle} \leftarrow \mathrm{data}^{\langle i \rangle}$
            $\mathrm{trimi} \leftarrow \mathrm{trimi} + 1$
    return $\mathrm{trimmed}^{T}$
```

VSmooth(v, w)

```
$\mathrm{Smooth}(x, w) :=$
    $\mathrm{maxit} \leftarrow 100$
    $\mathrm{sx} \leftarrow \mathrm{medsmooth}(x, w)$
    $\mathrm{sx2} \leftarrow \mathrm{medsmooth}(\mathrm{sx}, w)$
    $\mathrm{counter} \leftarrow 1$
    while $[1 - (\mathrm{sx} = \mathrm{sx2})] \cdot (\mathrm{counter} < \mathrm{maxit})$
        $\mathrm{sx} \leftarrow \mathrm{sx2}$
        $\mathrm{sx2} \leftarrow \mathrm{medsmooth}(\mathrm{sx}, w)$
        $\mathrm{counter} \leftarrow \mathrm{counter} + 1$
    $\mathrm{sx2}$
```

```
$\mathrm{VSmooth1}(x, W) :=$
    $n \leftarrow \mathrm{length}(W)$
    $\mathrm{ax} \leftarrow x$
    for $i \in 1 .. n$
      $\mathrm{ax} \leftarrow \mathrm{Smooth}\left(\mathrm{ax}, W_{i-1}\right)$
    $\mathrm{ax}$
```