Computes binomial numbers \((n,k) = \text{numbers of combinations}\)

**Syntax**

```plaintext
b = nchoosek(n, k)
logb = nchoosek(n, k, logFormat)
[logb, b] = nchoosek(n, k, logFormat)
[logb, b] = nchoosek(n, k)
```

**Arguments**

- **n, k**
  arrays of positive decimal integers. If both \(n\) and \(k\) are not scalars, they must have the same size.
- **b**
  array of positive decimal integers, with the size of the biggest array \(n\) or \(k\) : \(C_n^k\)
- **logb**
  array of positive decimal numbers, with the size of \(b\): \(\log10(C_n^k)\)
- **logFormat**
  single word "log10" | "10.mant". "log10" by default.
  - If "log10" is used, \(\text{logb}\) returns \(\log10(b)\).
  - If "10.mant" is used, then \(\text{logb}\) returns \(\text{int}(\log10(b)) + 10^{\log10(b)-\text{int}(\log10(b))}/10\): The 10-exponent of \(b\) is the logb integer part, while its fractional part directly shows the mantissa/10 of \(b\), in \([1.0, 10)/10\).

**Description**

For every \((n(i),k(i))\) element-wise pair, \(\text{nchoosek}()\) computes the number of \(k\)-element subsets of an \(n\)-element set.

It is mathematically defined by

\[
C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{(n-k+1)\ldots(n-1)\ldots n}{1\ldots(k-1)\ldots k}
\]

The implementation, though, uses the \(\beta\)eta function.

If \(n < 0\), or \(k < 0\), or \(k > n\), then an error is generated.

**Note about floating point accuracy**

Despite \(n!\) overflows for \(n>170\), \(\text{nchoosek}(n,k)\) is computed within almost the \(\%\text{eps}\) relative accuracy for \(n\) much beyond 170.

For any given \(n\) value, we know that \(C_n^k\) is maximal for \(k=n/2\). Scilab computes \(b = \text{nchoosek}(n,n/2)\) close to the \(\%\text{eps}\) relative accuracy for \(n\) up to 1029. Beyond this value, \(b=\%\text{inf}\) is returned.

This narrow \(n=1029\) limit can be overcome by computing \(\log10(C_n^k)\) and returning it through the second output argument \(\text{logb}\). This allows to use still bigger \(n\) values, and to get some valid information about the exponent and the mantissa of the true result.

However, we must keep in mind that beyond \(n = 1/\%\text{eps} \approx 4.10^{15}\), the numerical step between consecutive integers \(m\) and \(m+1\) stored as floating point numbers become \(>1\). Then \(n^{n(n-1)}\) is numerically equal to \(n^n\), and getting reliable results becomes harder.

The integer part of \(\text{logb}\) represents the exponent (of 10) of \(C_n^k\), whereas its fractional part is the \(\log10\) of its mantissa. When the integer part of \(\text{logb}\) increases, the more digits it takes among the 16 available ones in floating point encoding, the less remain to encode the mantissa. Then, knowing the mantissa of \(C_n^k\) with at least one significant digit requires \(C_n^k < 10^{10^{14}}\).

**Relative accuracy on logb**: As a rule of thumb, except for special \(k\) cases, the relative uncertainty on \(\text{logb}\) is of the order of \(\%\text{eps} \times \frac{n}{k}\).
Examples

Simple examples

\[
c = \text{nchoosek}(4, 1) \\
c = \text{nchoosek}(5, 0) \\
c = \text{nchoosek}([4 5], [1 0])
\]

\[
\rightarrow c = \text{nchoosek}(4, 1) \\
c = 4.
\]

\[
\rightarrow c = \text{nchoosek}(5, 0) \\
c = 1.
\]

\[
\rightarrow c = \text{nchoosek}([4 5], [1 0]) \\
c = 4. 1.
\]

\[
\text{nchoosek}(10, 0:10) \\
\text{nchoosek}(4:12, 4)
\]

\[
\rightarrow \text{nchoosek}(10, 0:10) \\
\text{ans} = 1. 10. 45. 210. 715. 120. 210. 45. 10. 1.
\]

\[
\rightarrow \text{nchoosek}(4:12, 4) \\
\text{ans} = 1. 5. 15. 30. 70. 126. 210. 330. 495.
\]

For big values:

\[
\text{exact} = 2.050083865033972676e307; \\
c = \text{nchoosek}(10000, 134) \\
\text{relelror} = \text{abs}(c-\text{exact})/\text{exact}
\]

\[
\rightarrow c = \text{nchoosek}(10000, 134) \\
c = 2.05D+307
\]

\[
\rightarrow \text{relelror} = \text{abs}(c-\text{exact})/\text{exact} \\
\text{relelor} = 7.959D-14
\]

The exact result for \( c_n^k(n, 2) \) is \( n(n-1)/2 \). Now, for values \( n > 1/\text{eps} = 4e15 \), \( (n-1) \) is numerically identical to \( n \). In no way we can expect an exact result below, but rather \( n^2/2 \):

\[
n = 1e20; \\
c = \text{nchoosek}(n, 2) \\
c / (n^n/2) - 1
\]

\[
\rightarrow c = \text{nchoosek}(n, 2) \\
c = 5.000D+39
\]

\[
\rightarrow c / (n^n/2) - 1 \\
\text{ans} = -6.661D-15
\]

In logarithmic formats:

\[
[\text{logb}, b] = \text{nchoosek}(10000, 1234); \ [b, \text{logb}]
\]

\[
\text{logb} = \text{nchoosek}(10000, 1234, "10.mant")
\]

\[
[\text{logb}, b] = \text{nchoosek}(1000, 500); \\
\text{logb2} = \text{nchoosek}(1000, 500, "10.mant"); \\
[b, \text{logb}, \text{logb2}] \\
\text{logb} = \text{nchoosek}(1000, 500, "10log")
\]

\[
\rightarrow [\text{logb}, b] = \text{nchoosek}(10000, 1234); \ [b, \text{logb}]
\]

\[
\text{ans} =
\]
INF   1620.803261
---> logb = nchoosek(10000, 1234, "10.mant")
logb =
1620.635713      // 6.35713D+1620
---> [logb, b] = nchoosek(1000, 500);
---> logb2 = nchoosek(1000, 500, "10.mant");
---> [b, logb, logb2]
ans =
2.7029D+299   299.4318272   299.2702882
---> logb = nchoosek(1000, 500, "log10")
logb =
299.4318272

Drawing \( n\choose k \) on the main floating point domain, up to \( 10^{300} \):

```plaintext
function ax=drawCnk(n)
    // Preparing data
    [N,K] = meshgrid(n);
    noOp = K>N;
    [logC, C] = nchoosek(N, K);
    C(noOp) = %nan;
    if max(n)<2000, logC = log10(C), end
    // Surface of Cnk data
    surf(N, K, logC);
    gce().color_mode = -2; // hiding the mesh
    plot2d([@ n],[@ n/2]);
    gce().children.data(:,3) = max(logC(logC<>%inf))
    // Axes settings
    xgrid(25,1,7)
    ax = gca();
    set(ax, "view","2d", "tight_limits",["on" "on" "off"], "grid_position","foreground");
    xtitle("","","");
    ax.margins(2) = 0.05;
    // Color bar
    colorbar();
    cb = gce().parent;
    cb.y_ticks.labels = msprintf("$10^{%s}$", cb.y_ticks.labels);
    title(cb,"$%c_n^k$", "fontsize",3)
endfunction
clf
drawlater
    // Figure settings
    f = gcf();
    f.color_map = jetcolormap(100);
    //f.axes_size = [840 570];
    // Main plot
    ax = drawCnk(0:10:1500); sca(ax);
    xtitle("n\choose k","n","k","$%c_n^k$");
    set([ax.title ax.x_label ax.y_label ax.z_label], "font_size",4);
    xstring(1250, 450, "Overflow")
    gce().font_size = 4;
    // Inset
    xsetech([0.11 0.11 0.37 0.42]);
    ax = drawCnk(0:100);
    ax.subplot = [3 3];
    gce().parent.subplot(2) = 4;
drawnow
```
Going beyond, in logarithmic mode:

```matlab
// // !\ : The drawCnk() function used here is defined in the previous example.
clf
drawlater

// Figure settings
f = gcf();
f.color_map = jetcolormap(100);
%f.axes_size = [840 570];

// Main plot
ax = drawCnk(0:10000:1e6); sca(ax);
xtitle("nchoosek(n, k)";
set([ax.title ax.x_label ax.y_label ax.z_label], "font_size", 4);

// Inset
xsetech([0.12 0.11 0.37 0.42]);
ax = drawCnk(0:10000:1e5);
ax.sub_ticks = [3 3];
gce().parent.sub_ticks(2) = 4;
drawnow
```
Top line $c_n^{n/2}$: $\binom{n}{n/2}$ and its known close approximation $2^n / \sqrt{\pi n/2}$

```
n = round(18^((1:0.1:8))^2);
logb = nchoosek(n, n/2, "log10");
clear
plot2d("ll", n, logb)
ax = gca();
xgrid(color("grey70"), 1, 7);
//ax.sub_ticks = [8 0];
ax.tight_limits = "on";
ax.y_ticks.labels = msprintf("\Large $10^{%d}$\n", ax.y_ticks.locations);
xlabel("n", "font_size", 4);
xstring(60, 1000, "nchoosek(n, n/2)", 270)
set(gca(), "clip_state", "off", "font_size", 3);
// Relative difference with the $2^n / \sqrt{\pi n/2}$ approximation:
e = abs(10.^((n"log10(2) - (log10(4*pi)+log10(n/2))/2 - logb) - 1);

ax = newaxes();
ax.filled = "off";
ax.y_location = "right";
ax.tight_limits = ["on" "off"];"
Bibliography

- Binomial coefficients (Wikipedia)
- "Introduction to discrete probabilities with Scilab", Michael Baudin, 2010

See also

- factorial
- gamma
- perms

History

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.0</td>
<td>nchoosek() introduced.</td>
</tr>
</tbody>
</table>

Report an issue

```
<< nchoosek
```

nchoosek