Spherical Bessel Functions in Mathematica

Here we look at spherical Bessel functions, seeing how to define them in terms of more general *Mathematica* functions (e.g., BesselJ), how to get a value printed out with more than the default number of digits (using NumberForm or ScientificForm), and how to make plots to analyze the behavior of the functions.

■ Definitions

Mathematica doesn't have built−in spherical Bessel function routines but you can define them easily in terms of the more general Bessel functions using expressions from Abramowitz and Stegun (or equivalent). If you don't know where to start, open the Help Browser, select "Master Index" and type in bessel.
Under the "Bessel functions" entry you'll find all of the info you need, including an entry for spherical Bessel functions.

If we know the name of the command, prepending a "?" tells us how to use it.

```
?BesselJ
```

```
BesselJ[n, z] gives the Bessel function of the first kind J(n, z). More…
```

```
?BesselY
```

```
BesselY[n, z] gives the Bessel function of the second kind Y(n, z). More…
```

Make the definitions. Remember to use "_" after the variables on the left side but not the right side. Also note the use of ":=" in the definition.

```
jl[x_, l_] := Sqrt[Pi / (2 x)] BesselJ[l + 1 / 2, x]
```

```
nl[x_, l_] := Sqrt[Pi / (2 x)] BesselY[l + 1 / 2, x]
```

■ Printing answers to full precision

If we want to know the value at x=36.2 and l=10, we don't get many digits and using N[ ] doesn't help:

```
jl[36.2, 10]
```

```
-0.000860425
```

```
N[jl[36.2, 10], 16]
```

```
-0.000860425
```

We can confirm that *Mathematica* calculates the result with more precision by using Precision[ ]:

```
Precision[jl[36.2, 10]]
```

```
MachinePrecision
```

One solution is to use NumberForm or ScientificForm to print out more digits:

```
? NumberForm
```

```
NumberForm[expr, n] prints with approximate real numbers in expr given to n-digit precision.
 More…
```

```
? ScientificForm
```

```
ScientificForm[expr] prints with all real numbers in expr given in scientific notation.
   ScientificForm[expr, n] prints with numbers given to n-digit precision. More…
```

```
NumberForm[jl[36.2, 10], 16]
```

```
-0.000860424929985335
```

```
ScientificForm[jl[36.2, 10], 16]
```

$$-8.60424929985335 \times 10^{-4}$$

Note that trying to print out 30 digits doesn't give more digits than the precision.

```
NumberForm[jl[36.2, 10], 30]
```

```
-0.000860424929985335
```

■ Generating a Log–Log plot

Suppose we want to examine the power–law dependence of the spherical Bessel functions with l=10. Start by defining "lval":

```
lval = 10;
```

We can check the first terms in the power series expansions about x=0 (even for negative powers of x!):

```
Series[jl[x, lval], {x, 0, 13}]
```

$$\frac{x^{10}}{13749310575} - \frac{x^{12}}{632468286450} + O[x]^{14}$$

```
Series[nl[x, lval], {x, 0, -9}]
```

$$-\frac{654729075}{x^{11}} - \frac{34459425}{2\,x^{9}} + \frac{1}{O[x]^{8}}$$

To make log–log plots, we need to load the "Graphics" package as follows. Be careful not to use the command "LogLog-Plot" before loading the package, or *Mathematica* will think you are defining a new name. If you make this mistake, use Remove[LogLogPlot] to recover.

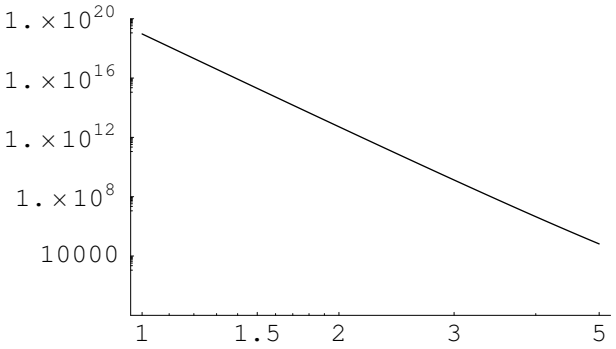
```
<< Graphics`Graphics`
```

```
? LogLogPlot
```

```
LogLogPlot[f, {x, xmin, xmax}] generates a plot of Log[f] as a function of Log[x]. More…
```

Check the relative scaling of nl and jl for 1<x<5:

```
LogLogPlot[Abs[nl[x, lval] / jl[x, lval]], {x, 1, 5}, PlotRange → {1, 10^(20)}]
```




```
- Graphics -
```

We see that nl is much larger than jl in this region.

Now suppose we want to look at jl/nl for fixed x (say x=0.1) as a function of integer l.

```
xval = 0.1;
```

First make a table of the values of l:

```
lvalues = Table[l, {l, 0, 10}]
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

If we evaluate a function with the table, we get a table of function values:

```
jl[xval, lvalues]
```

```
{0.998334, 0.0333, 0.000666191, 9.51852×10^-6, 1.05772×10^-7, 9.61631×10^-10,
 7.39754×10^-12, 4.93189×10^-14, 2.9012×10^-16, 1.52699×10^-18, 7.27151×10^-21}
```

```
nl[xval, lvalues]
```

```
{-9.95004, -100.499, -3005.01, -150150., -1.05075×10^7, -9.45525×10^8,
 -1.03997×10^11, -1.35187×10^13, -2.0277×10^15, -3.44696×10^17, -6.54901×10^19}
```

If we have numbers rather than functions to plot, we want to use a "ListPlot". Check the definition:

```
?LogListPlot
```

```
LogListPlot[{y1, y2, ...}] or LogListPlot[{{x1,
   y1}, {x2, y2}, ...}] generates a plot of Log[yi] against the xi. More...
```

This construction forms pairs from our lvalues and jl lists:

```
jlist = Abs[Transpose[{lvalues, jl[xval, lvalues]}]]
```

```
{{0, 4.99167}, {1, 0.1665}, {2, 0.00333095}, {3, 0.0000475926},
 {4, 5.2886×10^-7}, {5, 4.80816×10^-9}, {6, 3.69877×10^-11}, {7, 2.46594×10^-13},
 {8, 1.4506×10^-15}, {9, 7.63493×10^-18}, {10, 3.63576×10^-20}}
```

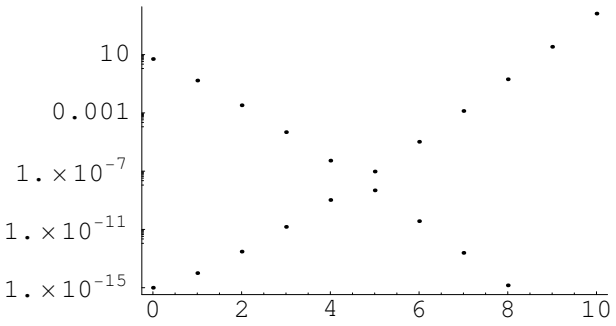
Multiply nl by machine precision for the comparison (and take absolute values):

```
nlist = Abs[Transpose[{lvalues, 10^(-16) * nl[xval, lvalues]}]]
```

```
{{0, 9.95004×10^-16}, {1, 1.00499×10^-14}, {2, 3.00501×10^-13},
 {3, 1.5015×10^-11}, {4, 1.05075×10^-9}, {5, 9.45525×10^-8}, {6, 0.0000103997},
 {7, 0.00135187}, {8, 0.20277}, {9, 34.4696}, {10, 6549.01}}
```

Build the plots first and then display them. The DisplayFunction stuff just prevents the p1 and p2 plots from being shown by themselves.

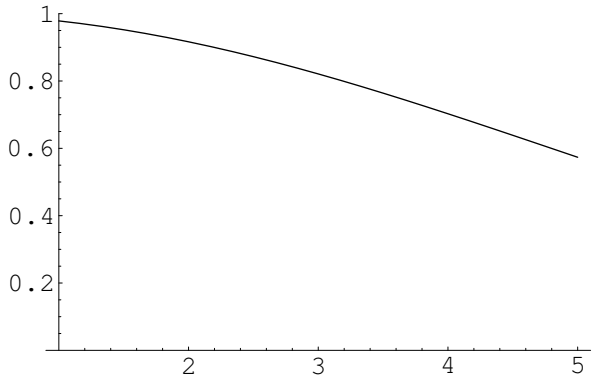
```
p1 = LogListPlot[nlist, DisplayFunction → Identity];
p2 = LogListPlot[jlist, DisplayFunction → Identity];
Show[p1, p2, DisplayFunction → $DisplayFunction]
```



- Graphics -

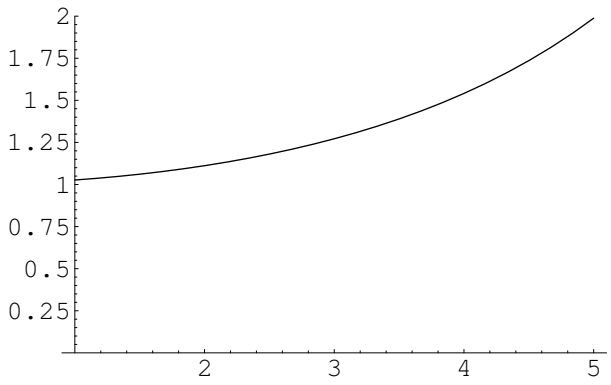
Now check how good the leading term for small x works for 1<x<5:

```
Plot[(2 * lval + 1)!! jl[x, lval] / x^(lval), {x, 1, 5}, PlotRange → {0, 1}]
```



- Graphics -

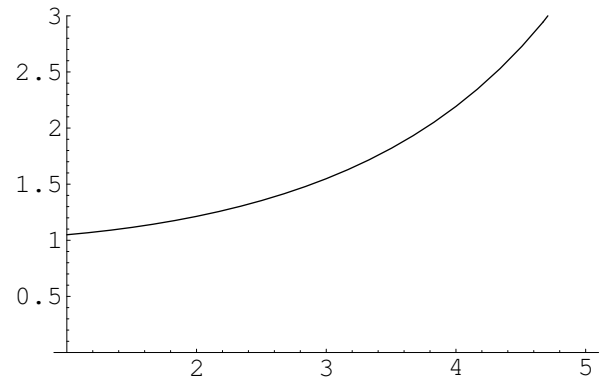
```
Plot[-nl[x, lval] * x^(lval + 1) / (2 * lval - 1)!!, {x, 1, 5}, PlotRange → {0, 2}]
```




```
- Graphics -
```

Finally, check out the ratio of these approximations:

```
Plot[(-nl[x, lval] * x^(lval + 1) / (2 * lval - 1)!!) / ((2 * lval + 1)!! jl[x, lval] / x^(lval)),
 {x, 1, 5}, PlotRange → {0, 3}]
```




```
- Graphics -
```