Function Addition

Michael Penna, Indiana University – Purdue University, Indianapolis

Objective

There are several ways in which new functions can be created from old ones. In this project we discuss function addition. We also illustrate how to write procedures in Maple. Procedures are a basic construct in many programming languages; they are useful since they (like do loops) allow us to perform repetitive tasks without duplicating code. Finally, we illustrate one way we can write and use lists.

Narrative

Analytically, the value of any function $f+g$ at x is given by $(f+g)(x) = f(x)+g(x)$. Geometrically, the value of $f+g$ can be computed, for any value of x, by adding the y-coordinates of $P(x, f(x))$ and $Q(x, g(x))$; this process is known as *ordinate addition*.

Task

1. Type the command lines below into Maple in the order in which they are listed. They use a procedure to produce two sets of points, one set on the graph of $f(x) = x+2$ and the other set on the graph of $g(x) = x^2$.

```
> # Your name, today´s date
> # Function Addition
> restart:
> with(plots):
> pointset := proc(f,a,b,N)
    local dx,mylist,n:
      dx := (b-a)/(N-1):
      mylist := [a,evalf(f(a))]:
      for n from 1 to N-1 do
        mylist := mylist,[a+n*dx,f(a+n*dx)]:
      end do:
    RETURN([mylist]):
  end:
> plot0 := pointplot(pointset(x -> x+2,-1.5,1.5,11),color=red,symbol=circle):
> plot1 := pointplot(pointset(x -> x^2,-1.5,1.5,11),color=green,symbol=circle):
> display({plot0,plot1},scaling=constrained);
```

At this time make a hard-copy of your typed input and Maple's responses. Then:

2. Using a ruler and ordinate addition, plot by hand the 11 points that lie on the graph of $f(x)+g(x) = x^2+x+2$.

3. Connect the dots that lie on the graph of $f(x)$, those that lie on the graph of $g(x)$, and those that lie on the graph of $f(x)+g(x)$. Finally, label the graphs of f, g, and $f+g$ by "$y = f(x)$", "$y = g(x)$", and "$y = f(x)+g(x)$".

Your lab report will be a hard copy of your typed input and Maple's responses (both text and hand-labeled graphics).

Comments

In Task 1, `[mylist]` is a list of coordinate pairs. In general, a list is an (ordered) collection of entries enclosed by brackets "`[`" and "`]`". It initially contains one coordinate pair (this is the effect of the command

"`mylist := [a,evalf(f(a))]`"), and subsequently, we add further coordinate pairs (this is the effect of the command "`mylist := mylist,[a+n*dx,f(a+n*dx)]`"). Finally we add the brackets "`[`" and "`]`" that enclose the entire list (when we return it from the procedure back to the main code with the command "`RETURN([mylist])`").

Alternately, we could have started off with an empty list (using a command such as "`foo := []`" or "`foo := null`"), opening it (with the `op` command), adding an entry to it, and then closing it (by enclosing it by brackets "`[`" and "`]`"):

```
> foo := [];
> foo := [op(foo),item];
```

Additional `items` could be added to `foo` by continuing with "`foo := [op(foo),items];`".

Lists, sets, and arrays are important data structures in Maple. If you plan on doing any of your own coding in Maple then you might want to investigate lists, sets, and arrays in Maple's Help pages.