The import buffering mechanism

The GroovyShell lacks a feature of keeping the imports from the previous statements. This can be a rather inconvenient in many cases, since we must issue manually the same import statements, before executing commands depending on them. For that reason, we implemented in GroovyLab a simple import buffering, that we think can facilitate the user's work.

GroovyLab prepends at the user's scripts some basic imports that extend the functionality, by performing a lot of import and import static statements, and allowing for example to use commands as *figure()*, *plot()* etc. In order to allow specialized import statements to be kept during a user's session, the user can **buffer** these import statements for that particular working session only. That can be accomplished by **selecting the relevant import statements** and then using **Buffer selected Imports** option from the **Imports** menu.

As an example suppose that we have the following script

```
// SECTION 1:  import section
import static com.nr.test.NRTestUtil.maxel
import static com.nr.test.NRTestUtil.vecsub
import static java.lang.Math.cos
import com.nr.interp.RBF_gauss
import com.nr.interp.RBF_interp
import com.nr.interp.RBF_inversemultiquadric
import com.nr.interp.RBF_multiquadric
import com.nr.interp.RBF_thinplate
import com.nr.ran.Ran
// SECTION 2: import independent commands section
  NPTS=100; NDIM=2; N=10; M=10;
  sbeps=0.05
  pts =new double[NPTS][NDIM]
  y = new double[NPTS]
  actual = new double[M]
  estim = new double[M]
  ppt = new double[2]
```

```
 globalflag=false

// SECTION 3:  import dependent command section
// Test RBF_interp
 myran = new Ran(17)
 pt = new double[M][2]
 for (i=0;i<M;i++) {
  pt[i][0]=(double)(N)*myran.doub()
  pt[i][1]=(double)(N)*myran.doub()
  actual[i]=cos(pt[i][0]/20.0)*cos(pt[i][1]/20.0)
}
 for (i=0;i<N;i++) {
  for (j=0;j<N;j++) {
   k=N*i+j
   pts[k][0]=(double)(j)
   pts[k][1]=(double)(i)
   y[k]=cos(pts[k][0]/20.0)*cos(pts[k][1]/20.0)
  }
}
 println("Testing RBF_interp with multiquadric function")
 scale=3.0
 multiquadric = new RBF_multiquadric(scale)
 myRBFmqf = new RBF_interp(pts,y,multiquadric,false)

 for (i=0;i<M;i++) {
  ppt[0]=pt[i][0]
  ppt[1]=pt[i][1]
  estim[i]=myRBFmqf.interp(ppt)
}
```

In order to execute first the import independent commands (i.e. SECTION 2), and after them in a separate script, the import dependent commands (i.e. SECTION 3) the user should *select the imports* and buffer them by using from the "*Imports*" menu the "*Buffer selected imports*" option. Issuing multiple "*Buffer selected imports*" commands, accumulate all the import statements, unless we clear the import buffer.

As we expect the script as a whole executes correctly. However, if we do not buffer the imports and try to execute statements that depend on the imports (e.g. myran = new Ran(17)), without prepending the code with them, we fail. However, if we buffer the necessary imports, we can execute our script conveniently **command-by-command**.