SEP # : Scilab interface to the Dakota software

Title: Scilab interface to the Dakota software
Version: 1.0
Author: Yann Chapalain (yann.chapalain@gmail.com)
Review:
Commented:
State: alpha
Scilab-Version: 5.3.1
Vote:
Created: May 4th, 2011

Abstract

The aim of this SEP is to add a Scilab interface to the Dakota software, to solve optimisation problems

Rationale

Dakota is a software developped by SANDIA laboratories. It contains some methods and algorithms for optimization problems. Nevertheless, without an interface with a computation software, Dakota can't solve any problems. That's why there are also two interfaces : Python and Matlab.

This project is an opportunity to add a Scilab interface, especially as it exists an API call_scilab. It is therefore expected to implement three interfaces called : « compiled » interface, « linked » interface and « scripted » interface.

Compiled interface

Simpler to implement than the linked interface, it is a good way to familiarize with the differents tools, which will be used. The operation of this interface is based on a new driver, which will be called « dakscilab ». Dakscilab will be used by Dakota to communicate with Scilab through pipe. During the optimization method, Scilab will permit to compute the objective function, and if necessary, hessian and gradient function. After some iterations, Dakota will can find the solution.

Thanks to API Scilab, Dakscilab will can given some instructions to Scilab, and request it to compute the different functions. So, Scilab will launch in background through fork. Concretly, Scilab will be called with :

- `BOOL StartScilab(char *SCIpath, char *ScilabStartup, int *Stacksize);`
- `int SendScilabJob(char *job);`
- `BOOL TerminateScilab(char *ScilabQuit);`

About Dakota, it needs to be launch with a configuration file, which describe the optimisation method, driver used for computations (dakscilab), etc… An example of a configuration file :

```
method,
  optpp_newton
```

```
   max_iterations=50,
   convergence_tolerance=1e-12

variables,
  continuous_design=2
   cdv_initial_point -1.2  1.0
   cdv_lower_bounds -2.0  -2.0
   cdv_upper_bounds 2.0  2.0
   cdv_descriptor      'x1'  'x2'

interface,
  fork
   analysis_driver='./dakscilab',
   parameters_file='r.in'
   results_file='r.out'
   deactivate active_set_vector

responses,
  num_objective_functions=1
   analytic_gradients #0,#1,#2,#3
    analytic_hessians #0
```

Scripted interface

The scripted interface resembles the previous one. It will provide a communication file. A script shell will create a directory (workdir) with used files in it. Next, it will call Scilab, which will read the params.in file, and write in the results.out.

Linked interface

The linked interface won't use fork or pipe. The driver specified in the configuration file will be Scilab. So, Dakota will be connected directly with an embeded Scilab server. The Scilab interface script will be send a specific Scilab structure. Via this structure, the Scilab interface script will be able to know what is needed to be computed (objective function, gradient, hessian).

Example of use

It is planned to develop some examples of use. The example planned are:

- optimization of the Rosenbrock function ;
- optimization of a PID regulator (using xcos) ;
- optimization of a truss structure (using the Femtruss module).

Example Usage

A user will launch Dakota with the config file which correspond to example. After some iterations, solution will appear if one exists.

Changelog

```
1.0 – initial release of the SEP.
```

Copyright