DISTRIBUTION PACKAGE

DYNAMICALLY DIMENSIONED SEARCH (DDS) ALGORITHM

MATLAB 14 PC Version 1.1mp

by: Dr. Bryan Tolson

Assistant Professor
Department of Civil & Environmental Engineering
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada, N2L 3G1
Ph: 519 888 4567 ext. 33377
email: btolson@uwaterloo.ca

REFERENCE FOR THIS ALGORITHM:
Tolson, B. A., and C. A. Shoemaker (2007), Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, Water Resour. Res., 43, W01413, doi:10.1029/2005WR004723.

README_1.1mp FILE CONTENTS:

INTRODUCTION

This file contains the disclaimer and user instructions for the Dynamically Dimensioned Search (DDS) Algorithm version 1.1mp by Bryan A. Tolson. DDS is an n-dimensional, heuristic, probabilistic global optimization algorithm for continuous, box-constrained (bound-constrained) optimization problems. DDS is designed to find good solutions quickly and requires little if any algorithm parameter tuning. Note that the DDS algorithm used to be called ‘GGS’.

MATLAB Distribution package 1.1mp contains MATLAB 14 generated pcode files (*.p) to use DDS within MATLAB on a windows PC. Note that pcode is equivalent in all respects to MATLAB m-files except users cannot see the source code of pcode files. This version of DDS can be linked with any user-supplied objective function that is coded as a Matlab m-file. User objective function m-file must accept a <u>row</u> vector of decision variables as input and return a scalar value for the objective function. DDS v1.1 is also implemented in Fortran 90.

Please note this distribution package is currently (Jan, 07) being updated to include source code

WHO SHOULD TRY DDS?

- those who have a highly dimensional (6 or more decision variables) continuous global optimization problem (multiple local optima). I have found good DDS performance relative to other global optimization methods for many 6-30 dimensional (# of decision variables) optimization problems as well as up to a 300 dimensional problem.
- especially those who meet the criterion above AND are optimizing a computationally expensive objective function such as a spatially distributed environmental simulation model calibration problem (e.g. watershed model). In my research, DDS has generated excellent watershed model calibration results in as few as 200 model evaluations for a 14 dimensional problem (14 model parameters were calibrated).
- DDS was *not* developed for (and thus performance *not* tested relative to other algorithms) optimizing low-dimensional problems (5 or fewer decision variables). Currently, algorithm modifications are being tested to determine relative DDS performance on these lower dimensional problems.
- for convex optimization problems, derivative-based optimization algorithms are expected to be more effective and efficient than DDS (especially low to moderate dimensions)

DISCLAIMER AND LICENSE INFORMATION:

This DDS algorithm implementation is free for public use. However, I request that the user reference and/or acknowledge the use of this program in any papers/reports/articles which have results obtained from the use of this program. I would also appreciate a copy of such papers/articles/reports, or at least an e-mail message with the reference so I can get a copy.

REFERENCE FOR THIS ALGORITHM:
Tolson, B. A., and C. A. Shoemaker (2007), Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, Water Resour. Res., 43, W01413, doi:10.1029/2005WR004723.

Please report to me any bugs/problems you find with this package.
For companies wishing to link this optimization code with an existing code, I am available for some consulting work.

GENERAL DDS ALGORITHM DESCRIPTION:

- an intentionally incomplete description of the algorithm is given as follows (complete details and source code available in package 1.1m to be released soon).
- The DDS algorithm is a novel and simple stochastic single-solution based heuristic global search algorithm that was developed for the purpose of finding good global solutions (as opposed to globally optimal solutions) within a specified maximum objective function (or model) evaluation limit.
- The only algorithm parameter to set in the DDS algorithm is the scalar neighborhood size perturbation parameter (r_val). A default value of the r_val parameter is recommended as 0.2 because this yields a sampling range that practically spans the normalized decision variable range. Empirical testing also showed this value to be robust and enables the algorithm to easily escape regions around poor local minima.
- The algorithm is designed to scale the search to the user-specified number of maximum objective function evaluations and thus has no other stopping criteria.
- The maximum number of function evaluations (m) is an algorithm input (like the initial solution) rather than algorithm parameter because it should be set according to the problem specific available (or desired) computational time the user wishes to spend solving the optimization problem. The value of m therefore depends on the time to compute the objective function on the available computational resources. Except for the most computationally trivial objective functions, essentially 100% of DDS execution time is associated with objective function evaluations.
- It must be clarified that the DDS algorithm is not designed to converge to the precise global optimum. Instead, it is designed to converge to the region of the global optimum in the best case or the region of a good local optimum in the worst case.
- DDS has at least three advantages relative to most population-based, evolutionary algorithms (e.g. GA, SCE, PSO etc.):
a) It has an immediate efficiency advantage because it is not population-based
b) It is designed to find good solutions quickly and thus it adjusts to the user-specified computational scale to generate good solutions without requiring any algorithm parameter adjustment.
c) It has only one algorithm parameter (r_val) that is easily interpreted and has a reasonably well established default value that has been shown to produce good results over a range of test problems. For most problems, I have found that the default of 0.2 will produce good answers and thus no algorithm parameter fine-tuning is recommended (unless of course ample computational resources make this feasible – try 0.1).

FILES & SUBDIRECTORIES INCLUDED IN THIS PACKAGE (1.1mp):

-> Readme_1.1mp.txt
The file you are currently reading.

The following 9 files in the directory with this README file are *always* needed to run the DDS program. They are described as follows:

-> DDS_inp.txt
NOTE: READ DDS_inp.txt WITH WORD WRAP OFF in NOTEPAD. This is the DDS algorithm input file. This is where user enters all DDS algorithm optimization inputs. This copy of the file is set to minimize the ackley10.m file. Algorithm inputs are defined in by the comments in this file. Read the instructions in the DDS_inp.txt file carefully! Further DDS input descriptions are given below in the USER INSTRUCTION SECTION.

-> 8 Matlab p-code files:
bounds.p
cleanInput.p
DDS.p
DDS_inout.p
MainDDS.p
neigh_value.p
read_DDS_inp.p
textread_bt.p

No explanation of the pfiles are necessary here since you can’t read their contents...

Ex1 Subdirectory:

This folder contains the files used for the Example 1 objective function - the Ackley 10-dimensional optimization test function (ackley10). The function is to be minimized and the global optimum is approx. -22.7183 at a solution of [0 0 … 0].

-> ackley10.m
mfile function code for the ackley 10-D optimization test function. Note the arguments used in ackley10.m are the I/O arguments you must use for your objective function. It may be important to note in some problems that the <u>decision variables are passed to your mfile as a row vector</u>.

-> Ex1/Ex1_out subdirectory
This subdirectory is the sample DDS output file folder created when the DDS program was run on the ackley10 problem. All output file contents are described in the output file called ‘Ex1_Input.txt’.

Ex2 Subdirectory:

This folder contains the files used for the Example 2 external objective function - the griewank 10-dimensional optimization test function compiled as an executable (Grie10_exe.exe).
* Only concern yourself with example 2 IF you are running DDS to optimize an external function – in which case you should consider the Fortran based PC executable DDS program as it has more user instructions to do this*
* This example is not needed for people who already have an m-file that calls various exe programs and then calculates and returns an objective function *

-> ext_function.m
template file for running external .exe for computing the objective function. See file for comments.

function_out.txt and variables_in.txt are for communication between DDS and external objective function (names are hardcoded in ext_function.m).

DDS_inp.txt – input file setup for this example.

ext_function.inp – file for bounds of decision variables. Fixed name no matter what .exe or .bat file name entered in DDS_inp.txt file.

EXAMPLES:

Follow the example below and you should be able to utilize this program for your own problem in Matlab. Note this program is written so that multiple DDS optimization runs can be executed.

Example 1
----
1. Create a new directory called 'Test1'.
2. Move the following 9 files from the unzipped directory:
   - DDS_inp.txt
   - bounds.p
   - cleanInput.p
   - DDS.p
   - DDS_inout.p
   - MainDDS.p
   - neigh_value.p
   - read_DDS_inp.p
   - textread_bt.p

   to your new Test1 directory.
3. Add the optimization problem specific files to your directory: In this case, they are ‘ackley10.m’ and ‘ackley10.inp’ from the Ex1 subdirectory.
4. Change the current directory in Matlab to ***/Test1
5. Run DDS with supplied inputs by typing ‘MainDDS’ at the command prompt

4. DDS writes screen output for optimization function.
In no more than a few seconds, you should see on the screen:

```
Trial number 1 executing ...
     Best objective function value is -22.673051
Trial number 2 executing ...
     Best objective function value is -22.666134
```

- The final screen message also specifies a subdirectory where the DDS output files are found
5. DDS created an output subdirectory called ' Ex1' within the Test1 directory.
- open this output subdirectory and there are 11 files here: 2 of which are simply copies of DDS input files and 9 of which are DDS output files.
- !! FIRST !! read the 'Ex1_Input.txt' created here. This file completely describes the DDS program output files.
- as you read the 'Ex1_Input.txt' file descriptions, open and browse the contents of the other files.
6. The files in your new Ex1 subdirectory should have exactly the same contents as the sample output files provided in …/Ex1/Ex1_out directory.

Example 2
----
* Only concern yourself with example 2 IF you are running DDS to optimize an external function (e.g. running an exe or batch file) – in which case you should consider the Fortran based PC executable DDS program as it has more user instructions to do this than the Matlab DDS*
* This example is not needed for people who already have an m-file that accepts decision variable value inputs and then calls various exe programs and then calculates and returns an objective function *

Instructions here are limited…

1. Move the 6 files in 'Ex2' directory to 'Test1' directory (replace others is fine).
2. Run MainDDS and it should work…

Main things to remember if you are use this template is the following:
- see the code in ext_function.m to understand communication strategy
- all file names are currently hardcoded but you can change ext_function.m if you wish
- DDS is writing variables_in.txt in %12.5f format (each line is one DV value, line 1 is DV 1, line 2 is DV2 … etc
- DDS is reading function_out.txt as %f
- your objective function executable must be OK with these formats (unless you change them)

USING DDS TO OPTIMIZE YOUR OWN M-FILE-BASED PROBLEM:

- You will now likely want to look at the DDS input file to see how to control the program (DDS_inp.txt). Also open the ackley10.m and ackley10.inp files to see what they look like.
- Note that the ranges/limitations for the DDS program inputs described below are also briefly described in the comments of the DDS_inp.txt file. So do not delete the comments in the DDS_inp.txt file!
- Hopefully after working through the example above, you have an initial idea how to run DDS on a given problem. Let me clarify how to do this for your problem in what follows:

- FIRST note that you must provide 2 files to use DDS on your problem: say you have ‘user.m’ to calculate your objective function value, then you must have a file called ‘user.inp’ that provides your decision variable info (name, lower and upper bounds). The filenames before the file extension must be exactly the same.

IMPORTANT: user.m must accept a vector of decision variable values and return a scalar value of the objective function. It may be important in some problems to note that DDS passes the decision variables to your m-file as a row vector.

NEXT, I describe the program inputs in the DDS_inp.txt file in more detail:

- LINE 3, you must tell the program the name of the m-file (no file extension) located in the same directory as the DDS program (e.g. MainDDS.p) that accepts new decision variable values and returns the objective function value. Users have the option of entering an executable file name or a batch file name here (with file extension) if their objective function evaluation requires invoking a file of this type (in this case, see Example 2 above)
- LINE 5, Controls how many times DDS solves the optimization problem. Since DDS is a stochastic or probabilistic algorithm, algorithm performance comparisons requires evaluating performance across multiple optimization trials. When you use DDS for your own problem and are not comparing algorithms, you may only want to use 1 or 2 long optimization trials if computation time is limited.
- LINE 6, the number of objective function evaluations to user per optimization trial. This input is equivalent to the terminology 'number of iterations used by DDS'. You will want to set this input for your own problems according to how long each objective function evaluation takes and how quickly you need an answer. The more objective functions you use, the better your estimate of the globally optimal solution will be. DDS was created from the perspective that the total available objective function evaluations for each optimization trial was limited (not infinite).
- on LINE 7 & 8 you enter the random seed input to fix the Matlab random number generators used in this program. Note that for your own problems, you should enter this input as a large (<10^8), randomly selected, positive integer. This input is what allows DDS users to replicate exact optimization results (often important for algorithm comparison purposes).
- on LINE 9, you enter a flag ("0" or "1") to change the files printed.

Users may want to enter "1" here if they are doing hundreds of optimization trials and/or have hundreds of decision variables otherwise you should keep this at "0".

- LINE 10 is the initial solution *filename* input line. A blank entry indicates that the DDS algorithm is to be initialized using uniform random sampling. Users can provide initial solutions if they wish by entering a filename here (e.g. initials.txt). Format of the file must be that each row contains one initial solution, the columns are the decision variables – col 1 is DV 1 etc. So if you specify 5 optimization trials you need 5 rows with no blank line at top. If you have 8 decision variables you need 8 columns etc…
- LINE 11 should be blank if user supplied mfile for objective function is used (and thus in same directory as the DDS pcode) but is included so that users with simulation model programs (i.e. specifying a .exe or .bat file on LINE 3) can keep the DDS code in one directory and their model and objective function code in any other directory. Simply enter the full directory path here that contains your objective function dependent files if they are not in the MainDDS.m directory.
- LINE 12 is not used by the program, a comment line.
- LINE 13 is a line for the user to enter comments describing/distinguishing the optimization run (up to 100 characters). It can be blank. Users may want to comment here if they modify their executable program (or any non-optimized inputs of their executable program) and optimize multiple slightly different optimization problems.
- LINE 14 is where the user tells DDS whether the objective function is to be minimized (enter "1") or maximized (enter "-1"). I call this input ‘to_max’. DDS is coded to minimize the value (to_max*Fvalue) where Fvalue is the user defined objective function. Thus, DDS maximizes objective function when this input is "-1". Only original objective function values (what you care about) are written in DDS output files.
- LINE 15 is the r_val DDS algorithm parameter. It has a fairly well-established default value of 0.2. Range is 0.0 < r_val <= 1.0. As r_val increases to 1, the sampling becomes more and more spread out from the current best value of the decision variable. Although I would suggest that experimenting with different r_val values is NOT really necessary to find good solutions, if users do choose to experiment with the r_val, I would first reduce the r_val from 0.2 and I would also not set r_val > 0.3. If DDS is initialized to what the user believes to be a relatively good quality starting solution, then users may want to try r_val=0.1 in order to focus the search more closely around the initial solution. However, reducing r_val increases chance DDS is stuck near a poor local minimum.

THAT’S IT – I hope this is enough to get you going.

OTHER DDS PROGRAM NOTES WORTH READING:

- For long optimization runs (many hrs or more) with multiple trials, users will want to set the print input to "0" to ensure that output is saved after each optimization trial is completed. Print flag set to "1" only writes out the outputs after ALL optimization trials are

completed. So in event of power outage before program termination, output files summarizing completed optimization trials are saved.
- In addition, during each optimization trial, every time a new best solution is found, the solution is written to a temporary file called 'status.out'. The status.out file is deleted if the optimization trial is completed (and thus is summary output files are then written as described above). So after the program finishes, this file is not available. You can also use this file to check the optimization progress as the program is running for computationally expensive problems.

ERROR MESSAGES or APPARENT ERRORS:

- If DDS executes with NO error message BUT the stest or Jtest or Jbest values never change in the output files, then your mfile is most likely not interpreting the vector of incoming decision variables from DDS correctly.
- if the program returns: "??? Error using ==> feval" and says your objective function is undefined, the program did not find your m-file in the DDS directory (where MainDDS.p is). Your m-file must be there!

DDS for Constrained Problems:

- DDS has only been tested on 2 or 3 constrained global optimization problems.
- In one example (Keane 20-dimensional bump function), where approximately 25% of DDS sampled decision variable sets were infeasible due to additional linear and non-linear constraints, simply setting infeasible solutions a constant value of 0 produced excellent results (it was a maximization problem, feasible solutions had positive objective function values).
- In the above example, it was quite easy to find initial feasible solutions (most infeasible samples occurred later in the search)
- In more heavily constrained problems, where it may be difficult to find a starting feasible solution, I would expect a penalty-function approach would be necessary. Currently, I am testing and determining how best to handle constraints with DDS so I should have more to say about this eventually.

DDS ALGORITHM MODIFICATIONS/IMPROVEMENTS/VERSIONS:

- I am currently testing a number of variants to DDS and you should periodically check my webpage (assuming it will soon be up to date!) for future versions.
- For any Fortran users, I have v1.1 of DDS implemented in Fortran as well and like this version, the source code is not yet available. However, this Fortran version is available as a compiled .exe to optimize an external program (.exe or batch file) so you do not have to run Matlab to use DDS.