DISTRIBUTION PACKAGE

DYNAMICALLY DIMENSIONED SEARCH (DDS) ALGORITHM

MATLAB PC Version 1.2_all
Last update: Feb 18, 2015

Runs on version R2014b for sure and hopefully all others

by: Dr. Bryan Tolson

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

REFERENCES FOR THIS ALGORITHM:

For continuous optimization problems:
Tolson, B. A. and C. A. Shoemaker (2007), Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, Water Resources Research, 43, W01413, doi:10.1029/2005WR004723.

For discrete/mixed integer optimization problems (Discrete DDS):
Tolson, B. A., M. Asadzadeh, H. R. Maier, and A. Zecchin (2009), Hybrid discrete dynamically dimensioned search (HD-DDS) algorithm for water distribution system design optimization, Water Resources Research, 45, W12416, doi:10.1029/2008WR007673.

This file contains the user instructions for the Dynamically Dimensioned Search (DDS) Algorithm version 1.2 by Bryan A. Tolson. DDS is an n-dimensional, heuristic, probabilistic global optimization algorithm for continuous/discrete/mixed decision-variable, box-constrained (bound-constrained) optimization problems. DDS is designed to find good solutions quickly and requires no algorithm parameter tuning.

README_1.2_all FILE CONTENTS:

DISCLAIMER AND LICENSE INFORMATION:

Licence information: http://www.gnu.org/licenses/gpl.txt

This DDS algorithm implementation is free for public use and you are free to redistribute it. However, the user needs to reference 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.

Please report to me any bugs you find in this code.

For companies wishing to link this optimization code with an existing simulation model generating objective/constraint values, I am available for some consulting work. I suggest users alter this code as little as possible to make future updates I make to DDS easier to incorporate.

DISCLAIMER: this program is not guaranteed to be free of error (although it is believed to be free of error). This software is provided 'AS IS', without warranty of any kind, expressed or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort of otherwise, arising from, out or in connection with the software or the use or other dealings in the software.

CHANGES RELATIVE TO RELEASE 1.1

The primary change to the algorithm is just the added flexibility to tackle binary/integer variable decision variables. Users now specify the bounds of decision variables and identify if the variable is an integer-valued or not in the decision variable input file. The decision-variable perturbation distribution is now a discretized normal distribution. Users make continuous/integer specification for each decision variable and hence this release enables DDS to solve mixed integer optimization problems.

The only other noteworthy change is the use of MATLABs recommended updated random number generation routines. The random number generator call has been updated to reflect new MATLAB syntax. The ‘rng’ function controls the shared generator used by rand, randn,randi and all other random number generation functions and is now used to fix the random number generator. This function automatically uses the default Mersenne Twister algorithm

Users of release version 1.1 should be able to simply update their

decision variable bound input file (***.inp* where *** is the name of their MATLAB objective function m-file) to specify their decision variables are continuous and then this code will run fine. Each line in this file must now end with 0 (continuous variable) or 1 (Integer valued variable) Note that users of version 1.1 will not be able to recreate their results exactly due to new random number generation.

Non algorithmic changes:
- 'CleanInput.m' has been removed. The call to this function under read_DDS_inp.m has been replaced with the MATLAB built-in function 'strtrim'to accommodate for newer release versions of MATLAB
- 'Textread_bt.m' has been removed. The call to this function under the 'bounds.m' file has been replaced with the MATLAB built-in function 'textscan' to accommodate for newer release versions of MATLAB.
- The latest version has been updated to include both discrete and continuous decision variables.

INTRODUCTION

These codes run 1 or more DDS optimization trials and save and manage all outputs. This version of DDS can be linked with any user-supplied objective function that is coded as a Matlab m-file. Template provided for m-file that needs to call an .exe or .bat program. User defined objective function m-file must accept a vector of decision variables and return a value for the objective function.

WHO SHOULD TRY DDS?

- those who have a highly dimensional (6 or more decision variables) global optimization problem (multiple local optima). I have found good DDS performance relative to other global optimization methods for many 6-60 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 reasonably good quality watershed model calibration results in as few as 200 model evaluations for a 58 dimensional problem (58 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).
- for convex optimization problems, derivative-based optimization algorithms are expected to be more effective and efficient than DDS (especially low to moderate dimensions)

GENERAL DDS ALGORITHM DESCRIPTION:

- 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. My research group has conducted probably a 100 or more different comparisons of DDS against other algorithms and for all of these comparisons, we used *r_val*=0.2 for the parameter value and found very favourable performance. Since the one example optimization problem in Tolson and Shoemaker (2007), we have never utilized anything but *r_val*=0.2 in all other DDS publications.
- 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 well established default value that has been shown to produce good results over a range of test problems. No algorithm parameter fine-tuning is recommended.

FILES & SUBDIRECTORIES INCLUDED IN THIS PACKAGE (1.2_all):

-> Readme_1.2_all.pdf or Readme_1.2_all.docx
The file you are currently reading.

The following 11 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/WORDPAD. 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 Griewank10.m test function – run MainDDS function to run algorithm. Algorithm inputs are defined briefly in the comment section of the file - read these carefully! Further DDS input descriptions are given below in the USER INSTRUCTION SECTION.

-> Matlab m-files:
*bounds.m (reads decision variable bounds file)
*DDS.m (DDS algorithm code)
*DDS_inout.m (records all DDS inputs to an output file)
*ext_function.m (only used if you specify your obj function value is returned by exe program or batch file)
*get_objfunc.m (function to call objective function m-file)
*MainDDS.m (main driver of program calling DDS for multiple trials)
*neigh_value_mixed.m (determines whether a decision variable selected for perturbation variable is discrete or continuous)
*neigh_value_continuous.m (perturbs current decision variable if it is continuous)
*neigh_value_discrete.m (perturbs current decision variable if it is discrete/integer)
*read_DDS_inp.m (reads program input file)

*all above files are always needed to run DDS.

plot_v1.m (just a program to plot up optimization results, if you don't like, delete call to it in MainDDS.m)
Griewank10.m (just an example optimization test function)

The source codes are commented fairly well.

Ex1 Subdirectory:

This folder contains the files used for the Example 1 objective function - the Ackley 10-dimensional optimization test function (ackley10).

-> 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 decision variables are passed to your m-file as a row vector.

-> 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_out_Input.txt'.

Ex2 Subdirectory:

This folder contains the files used for the Example 2 external objective function - the griewank 10-dimensional optimization test function (Grie10_exe.exe).

* Only concern yourself with example 2 IF you are running DDS to optimize an external function *
* This is not needed for people who already have an m-file that calls various exe programs and then calculates and returns an objective function *

→ to run this example, users must CHANGE 'Grie10_exe.zzz' to 'Grie10_exe.exe' (extension modified for email filters)

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 examples 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 0
---
1. Open up and run the MainDDS.m script.
2. Enter 'no' at user prompt
3. If you see on the screen:

---------- DDS Optimization Algorithm Main Program ----------

Trial number 1 executing ...
Best objective function value is 1.030610
Trial number 2 executing ...
Best objective function value is 1.084289

Etc.
The program has run as expected → go to next example.

Example 1
---
1. Create a new directory called 'Test1'.
2. Copy the following files from the unzipped directory (containing this README file):
   DDS_inp.txt
   All the *.m files (Griewank10.m not needed)
   to your new Test1 directory.
3. Add the optimization problem specific files to your Test1 directory: In this case, they are 'ackley10.m' and 'ackley10.inp' from the Ex1 subdirectory.
4. Open DDS_inp.txt file and change:
   a. Line 3 input to tell DDS the objective function it is

evaluating: "ackley10"
b. Line 4 input to "ack10"

5. Change the current directory in Matlab to ***/Test1
6. Run DDS with supplied inputs by typing 'MainDDS' at the MATLAB command prompt (MATLAB must be in the right directory)
7. DDS writes screen output for optimization function.
8. You should see on the screen:

---------- DDS Optimization Algorithm Main Program ----------

Trial number 1 executing ...
Best objective function value is -22.566564
Trial number 2 executing ...
Best objective function value is -22.601321
Etc. ...

9. The screen message also tells user a subdirectory where the DDS output files are found.
10. You are then also asked if you want to plot DDS results. I suggest you say 'no' to start and instead investigate the output files in the Ex1 output folder. Try the plot option later.
11. DDS created an output subdirectory called 'ack10' within the Test1 directory.
    - open this output subdirectory and there are 11 files here
    - !! FIRST !! read the 'ack10_Input.out' created here. This file completely describes the other program output files.
    - as you read the 'ack10_Input.out' file descriptions, open and browse the contents of the other files.

Example 1b

Assuming you do not want to have to put your objective function code into the same folder as the DDS code, try the following:

Move ackley10.m to a different directory.
See line 11 of the DDS_inp.txt file and edit accordingly.
[note ackley10.inp still has to stay with DDS code folder]
Run MainDDS again.
You will note DDS runs much slower on this same problem now due to lots of directory changing (see comments in get_objfunc.m).

Example 2

Only concern yourself with example 2 IF you are trying to link DDS with an external program to calculate the objective function value. Instructions here are limited…

1. Reuse files in Test1 directory from Example 1.
2. Look in Ex2 directory.
   a. Copy the model directory to Test1 directory
   b. To run this example users must CHANGE 'Grie10_exe.zzz' to 'Grie10_exe.exe' (extension modified for email filters) in the model directory.

c. Double-click on this .exe file and test to see it produces an output file. Now you know DDS can invoke this and get a result.
d. Copy the DDS_inp.txt and ext_function.inp files to Test1 directory (replace any existing files with these names).

3. Ensure the current directory in Matlab to ***/Test1
4. Run DDS with supplied inputs by typing 'MainDDS' at the command prompt
5. First, be sure to open the DDS_inp.txt file to see what inputs have changed to run this new problem.
6. As in above example, check out the outputs in Ex2 folder

Main things to remember if you are to use this ext_function.m template is the following:

- 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 decision variable value, line 1 is DV 1, line 2 is DV2 … etc
- DDS is reading function_out.txt as %f format for the objective function value
- your objective function executable or batch file must be OK with these formats (unless you change them)
- SEE source code of ext_function.m for all logic

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

- You will now likely want to look closely 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 noted 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 examples above, you have a good idea how to use DDS for your own problem. Let me clarify how to do this 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, and a 0/1 integer flag where 1 means decision variable is integer). The filenames before the file extension must be exactly the same.

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

- LINE 4, you must tell the program the name of the function that accepts new decision variable values and returns the objective function value. There is an option to use .exe or .bat file rather than an m-file (see Ex2 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. Two is always best to guard against single unlucky result.

- 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 is limited.

- 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 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.

- LINE 11 can be blank (if user m-file objective function is in same directory as the DDS code) but is included so that users 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 if your objective function mfile (or model exe/batch file) is not in the DDS directory. Note that switching directories increases runtime but the increase should be negligible relative increase if your objective function takes more than a 0.5 seconds to run. On the other hand if you are optimizing a function that is very fast and you input a directory location here, DDS is working slowly due solely to the directory switching.

- LINE 12 is not used by the program, a spacing 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"). DDS is coded to minimize but minimizes -1*Fvalue (equivalent to maximize Fvalue) when this flag is "-1". Only original obj function values written as output (eg Fvalue rather than -1*Fvalue).

- LINE 15 is the *r_val* DDS algorithm parameter. It has a very 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. I would suggest that experimenting with different *r_val* values is NOT 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 initial solution, then users may want to try *r_val*=0.1 in order to focus the search more closely around the initial solution.

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 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.
- If DDS executes with NO error message BUT the stest or Jtest or Jbest values never change in the outputs, then your mfile is most likely not interpreting the vector of incoming decision variables from DDS correctly.
- If DDS executes and you see <u>all</u> objective function values in outputs printed as 'INF' or '-INF' then there is an issue. See ext_function.m comments: Your exe/batch file is not being invoked or Your exe/batch coding is not properly writing function_out.txt
- If DDS executes and you see <u>some</u> objective function values in outputs printed as 'INF' or '-INF' then there is likely <u>no</u> issue with coding logic but your simulation model is probably crashing periodically. See ext_function.m comments.
- DDS can be modified to accept both discrete and continuous decision variables. This can be achieved by modifying the objective function '.INP' file. Note that column 4, entitled 'Integer?', is used to delineate whether a decision variable is 'continuous' or 'integer'. A flag of '0' is used to signify a continuous variable whereas '1' signifies an integer variable.

Related DDS Publications to be Aware of:

When solving model calibration problems with long simulation model run times, you will likely want to use the model pre-emption concept. See the following publication for details:

Razavi, S., B. A. Tolson, L. S. Matott, N. R. Thomson, A. MacLean, and F. Seglenieks (2010), Reducing the computational cost of automatic calibration via model pre-emption, Water Resources Research, 46, W11523, doi:10.1029/2009WR008957.

This source code is basically ready to use in pre-emption context – email me if necessary. To use pre-emption, you will want your code to open up and check 'status.out' file for best objective function value found so far.

Using DDS to approximate environmental simulation model prediction uncertainty (DDS-AU algorithm):

Tolson, B. A., and C. A. Shoemaker (2008), Efficient prediction uncertainty approximation in the calibration of environmental simulation models, Water Resources Research, 44, W04411, doi:10.1029/2007WR005869.

The DDS code distributed to you can be used to implement DDS-AU. You just need to write some supplemental code to process outputs, save time series etc.

DDS applied to constrained discrete decision variable value problems:

Matott, L. S., B. A. Tolson and M. Asadzadeh, (2012), A benchmarking framework for simulation-based optimization of environmental models, Environmental Modelling and Software, 35, 19-30. doi:10.1016/j.envsoft.2012.02.002.

DDS compares very favourably to metamodel enabled optimizers:

Razavi, S. S., B. A. Tolson and D. H. Burn (2012), Numerical assessment of metamodelling strategies in computationally intensive optimization, Environmental Modelling and Software. 34(0), 67-86. doi:10.1016/j.envsoft.2011.09.010.

DDS for constrained single-objective water distribution network design (discrete problem):

Tolson, B. A., M. Asadzadeh, H. R. Maier, and A. Zecchin (2009), Hybrid discrete dynamically dimensioned search (HD-DDS) algorithm for water distribution system design optimization, Water Resources Research, 45, W12416, doi:10.1029/2008WR007673.

DDS for multi-objective optimization (PA-DDS algorithm):

Asadzadeh, M. and B. A. Tolson (2013), Pareto archived dynamically dimensioned search with hypervolume-based selection for multi-objective optimization, Engineering Optimization. DOI:10.1080/0305215X.2012.748046.

Comparing DDS to SCE (response to comment on original 2007 paper):

Tolson, B. A., and C. A. Shoemaker (2008), Reply to comment on "Dynamically dimensioned search algorithm for computationally efficient watershed model calibration" by Ali Behrangi et al., Water Resources Research, 44, W12604, doi:10.1029/2008WR006862.

DDS for Constrained Problems:

- DDS has been tested on a few constrained global optimization problems. See for example Matott et al. (2012) and Asadzadeh et al (2009) above.
- 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, a penalty-function approach may be necessary. However, one of the problems in Asadzadeh et al (2009) has a very large infeasible region and a penalty-function approach is avoided in that paper.