Journal of Natural Gas Science and Engineering 21 (2014) 1184–1204

Contents lists available at ScienceDirect

Journal of Natural Gas Science and Engineering

journal homepage: www.elsevier.com/locate/jngse

Designing and optimizing deviated wellbore trajectories using novel particle swarm algorithms☆

Amin Atashnezhad [a, *], David A. Wood [b], Ali Fereidounpour [c], Rasoul Khosravanian [a]

[a] Petroleum Engineering Department, Amirkabir University of Technology, Tehran, Iran
[b] DWA Energy Limited, Lincoln, United Kingdom
[c] Petroleum Engineering Department, Tehran University, Iran

ARTICLE INFO

Article history:
Received 17 March 2014
Received in revised form
13 May 2014
Accepted 16 May 2014
Available online 17 July 2014

Keywords:
Meta optimization
Particle swarm optimization
Well trajectory analysis
Directional drilling optimization
MATLAB PSO algorithm codes

ABSTRACT

Wellbore trajectory design is a determinant issue in drilling engineering. This paper introduces a new stochastic approach for drilling trajectory design applying continuous particle swarm algorithms to find the optimum drilling measured depth of directional and horizontal wells in 3-D space. Considering all the constraints and limitations, the final goal is to determine all geometrical well parameters, in order to achieve the optimum measured depth to the desired target location. Particle swarm optimization is a computational algorithm inspired from natural behavior of some animal societies, such as flocks of birds and shoals of fish. In this review, the trajectory design method for an objective function, originally proposed by Adams and Charrier (1985), is explored and developed. Also, the attributes of the particle swarm optimization (e.g., Onwunalu, 2010) and Meta-optimization (e.g., Pedersen, 2010a,b) algorithms are considered and compared. These algorithms are then applied to demonstrate the determination of true measured depth of example horizontal wellbores as the objective function. Faster convergences, better final points which satisfy all constraints imposed on the drilling paths and population diversity maintenance to help the algorithms find better solutions, are positive characteristics of the solutions found using the algorithms proposed. These algorithms make promising new tools for designing economically-effective trajectories for deviated wells.

MATLAB codes for the PSO algorithms evaluated are provided as appendices to this article.

1. Introduction

The industry has recognized for more than a century the benefits of being able to drill and steer deviated wellbores with the early oil well application in the 1920s and the first horizontal well drilled in 1929 near Texon, Texas, USA (Kashikar, 2005). Prior to the 1950s wells were drilled with horizontal sections of just a few tens of meters, but the technology was gradually scaled up (Pratt, 2004). The first multi-lateral well was drilled in the Soviet Union in 1953, where that technology was further developed through to the 1980s (Kashikar, 2005). It was the 1980s and 1990s when directional drilling technology became a commercially-viable and the preferred option for drilling and developing many oil and gas fields. In the 1980s directional drilling was though quite different from today, without the benefits of steerable motors and bits and many of the other technologies subsequently developed, making it risky and costly. Of the many problems that have confronted directional drilling, selecting the optimum wellbore trajectory is one that has received much attention over the years.

One of the primary reasons for drilling directional boreholes is to reach and traverse subsurface objectives that could not easily be reached with a single or multiple vertical boreholes (Pratt, 2004). To optimize oil and gas recovery, wellbores must have the best possible exposure to the reservoir. Drilling sharply deviated or horizontal wells makes it possible to place the well path within the productive intervals over long distances. There are in fact several reasons why directional and horizontal drilling has become the drilling technology of choice since the 1990s (e.g. Short, 1993), including: draining wider areas of a reservoir to increase well productivity; increasing the net productive section length of target reservoir drilled; drilling multiple wells from a single well pad to minimize the surface footprint of drilling operations; production improvement by penetrating more fractures in fractured reservoirs; precise positioning of relief wells in instances of loss of well

☆ This article belongs to the Special Issue: *Gas Systems Optimization*.

* Corresponding author.
E-mail addresses: Atashnezhad1@gmail.com (A. Atashnezhad), dw@dwasolutions.com (D.A. Wood), A.fereidounpour@ut.ac.ir (A. Fereidounpour), Khosravanian@aut.ac.ir (R. Khosravanian).

http://dx.doi.org/10.1016/j.jngse.2014.05.029
1875-5100/

control. Some of these now-essential applications of directional drilling in the oil and gas industry are illustrated in Fig. 1.

Directional drilling typically is more expensive than vertical drilling, but its cost is justified by its potential to significantly increase well productivity and deliver a lower overall cost of supply on a cost/unit of gas (or oil) produced (e.g. Dalzell, 2013). Finding minimum trajectory length and more accurate well paths are two key factors in reducing drilling time and total well cost. The final goal is to determine the optimum directional well design parameters when the coordinates of wellhead and downhole target locations are specified, and some other geometrical and technical constraints are imposed on wellbore trajectory design (e.g. avoiding existing wellbores; avoiding shallow gas pockets; avoiding the intersection of certain faults, etc.).

Particle swarm optimization (PSO) is a metaheuristic optimization method that makes few assumptions about the problem being optimized with the ability to search large solution spaces to find the better solutions. It was first developed by Kennedy and Eberhart (1995) and Kennedy (1997). PSO optimizes a problem iteratively by monitoring and improving a population of candidate solutions, referred to as "particles", in terms of an objective function or fitness test (i.e. measure of quality). It does this by repeatedly adjusting the position of the particles in the search-space according to simple mathematical formulas that determine each particle's "position" and "velocity" (Shi and Eberhart, 1998a,b).

This optimization technique has in recent years been successfully applied, often in hybrid methodologies involving other optimization tools, to successfully solve varied optimization problems in the petroleum industry, for example: prediction of reservoir permeability (Ahmadi et al., 2013); prediction of minimum miscibility pressure for carbon dioxide injection (Sayyad et al., 2014); and, parameter estimation for a polypropylene reactor (Prata et al., 2009).

In fact, recent research has demonstrated that PSO applications include many diverse areas of potential application, such as: communication networks; robotics; signal processing; power generation, transmission and distribution systems and networks; prediction and forecasting, electronics and electromagnetics; meteorological predictions; investment decision-making; face detection and recognition, etc. For example, in the medical and pharmaceutical sectors, some PSO applications include: human tremor analysis for the diagnosis of Parkinson's disease (Eberhart and Hu, 1999); inference of gene regulatory networks (Rui et al., 2007); gene clustering (Xiao et al., 2003); and, DNA motif detection (Han et al., 2005). Hence, this relatively recent tool has seen rapid and diverse uptake which testifies to its powerful and desirable performance.

The PSO algorithm is applied to successfully design a wellbore trajectory, based on a real well drilled in Egypt, and previously used to demonstrate wellbore trajectory optimization using a genetic algorithm by Shokir et al. (2004).

2. Calculating wellbore trajectories

A number of methodologies have been applied to estimate wellbore trajectory in the planning stages and while a wellbore is being drilled Craig and Randall (1976). The most-widely used methods are the tangential, angle-averaging, minimum curvature and radius-of-curvature methods. The tangential method involves only the inclination and direction angles measured at the lower end of the wellbore length. The wellbore path is assumed to be a tangent to these angles throughout its length. Although this method has been widely used it is the least accurate approach. In the 1990s several models were developed to calculate detailed well trajectories of deviated wellbores (e.g. Hashim, 1995; Xiushan et al., 1997; Suryanarayana et al., 1998) and these have been further developed in the past decade (e.g. Shokir et al., 2004).

There are several methods used by the industry to calculate well trajectory, i.e., the tangential method, the angle-averaging method and the radius of curvature method are the more common ones. In this paper, we focus on the radius of curvature method to design our example well trajectories and are outlined below. In the 1990s several models were developed to calculate detailed well trajectories of deviated wellbores (e.g. Hashim, 1995; Xiushan et al., 1997; Suryanarayana et al., 1998) and have been further developed in the past decade (e.g. Shokir et al., 2004).

2.1. The radius of curvature method

The details of radius of curvature calculations have been well established since the 1980s (Adams and Charrier, 1985). In this study we focus on the radius of curvature method in order to compare the performance of PSO algorithm in determining the optimum trajectory parameters for a defined wellbore objective and constraints versus the performance of a genetic algorithm for the same wellbore objective and constraints, as originally presented by Shokir et al. (2004).

Considering a simple deviated wellbore the main parameters which affect the true measured depth (TMD) in the build-up phase calculated by the radius of curvature method are illustrated in Fig. 2. The effective parameters are vertical inclination hold angle(s), azimuth angles, dogleg severity, true vertical depths, and, lateral length. The problem is to find a good solution for effective parameters with the objective of minimizing true measured depth.

A schematic vertical plane cross-section of a generic 3D-horizontal well trajectory with more than one angle build-up and hold sections and an angle drop-off section is shown in Fig. 3. It is this well design that is used in the optimization analysis presented here.

Equation (1) calculates the constant curvature equation between two points in space:

$$a = \frac{1}{\Delta MD}\sqrt{(\theta_2-\theta_1)^2\sin^4\left(\frac{\varnothing_2+\varnothing_1}{2}\right)+(\varnothing_2-\varnothing_1)} \tag{1}$$

where "a" is the constant of curvature in degrees/foot and ΔMD is the change in measured depth.

Equation (2) calculates the radius of curvature, "r", in degrees/100 feet:

$$r = \frac{1}{a} = \frac{180*100}{\pi*T} \tag{2}$$

where "T" is the dog-leg severity

Considering the equations (1) and (2), equation (3) calculates the well path between two points in a three dimensional survey (Shokir et al., 2004):

$$\Delta MD = r \times \sqrt{(\theta_2-\theta_1)^2\cdot\sin^4\left(\frac{(\varnothing_1+\varnothing_2)}{2}\right)+(\varnothing_2-\varnothing_1)^2} \tag{3}$$

Equations (4)–(6) calculate offset distances in three specific directions (i.e. to the north, to the east and vertically) and are derived from the radius of curvature method (Adams and Charrier, 1985).

$$\Delta North = \frac{\Delta MD\cdot(\cos(\varnothing_1)-\cos(\varnothing_2))\cdot(\sin(\theta_2)-\sin(\theta_1))}{(\varnothing_2-\varnothing_1)\cdot(\theta_2-\theta_1)} \tag{4}$$

$$\Delta EAST = \frac{\Delta MD\cdot(\cos(\varnothing_1)-\cos(\varnothing_2))\cdot(\cos(\theta_1)-\cos(\theta_2))}{(\varnothing_2-\varnothing_1)\cdot(\theta_2-\theta_1)} \tag{5}$$

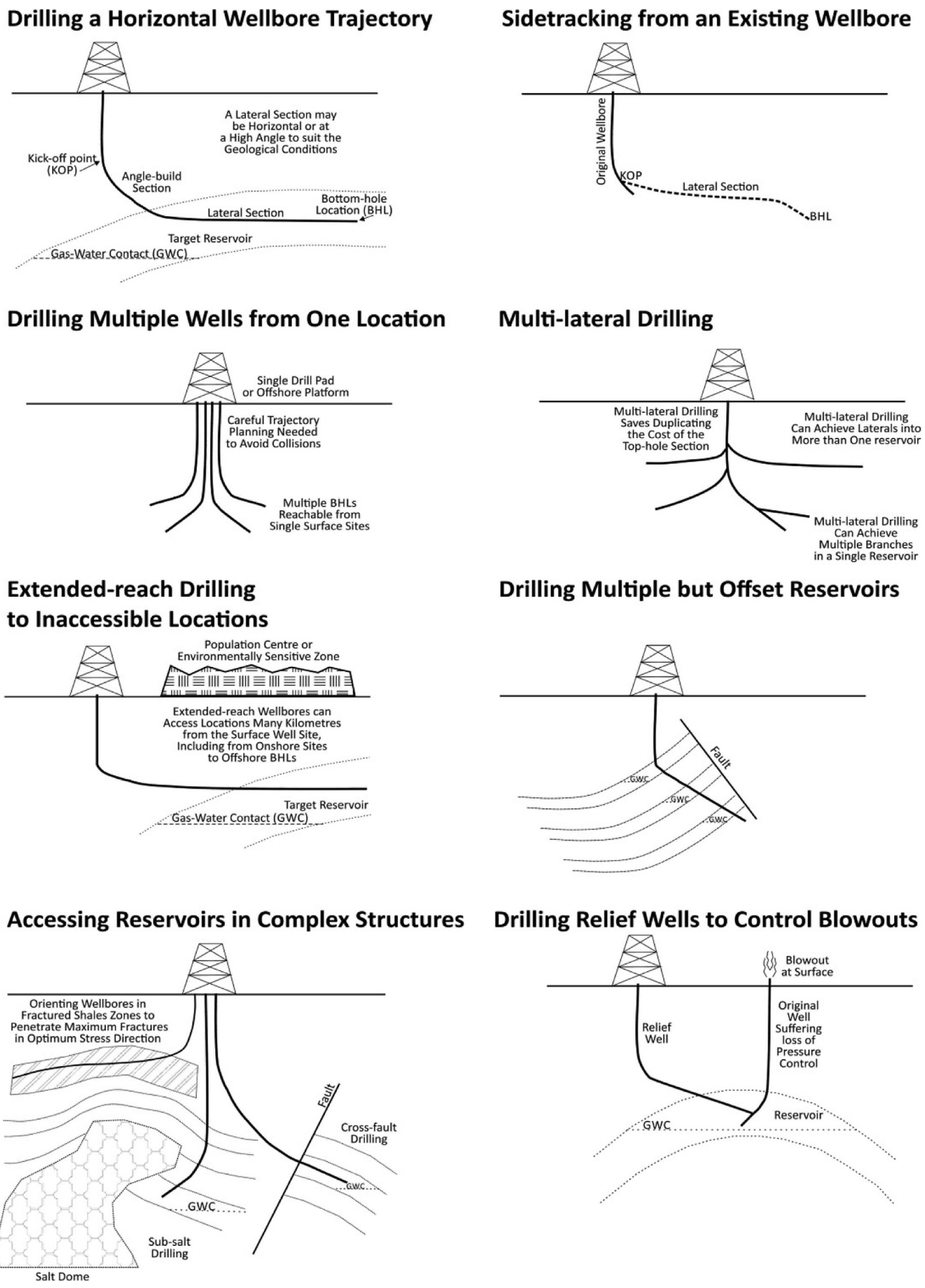


Fig. 1. Reasons for drilling directionally-deviated wellbores.

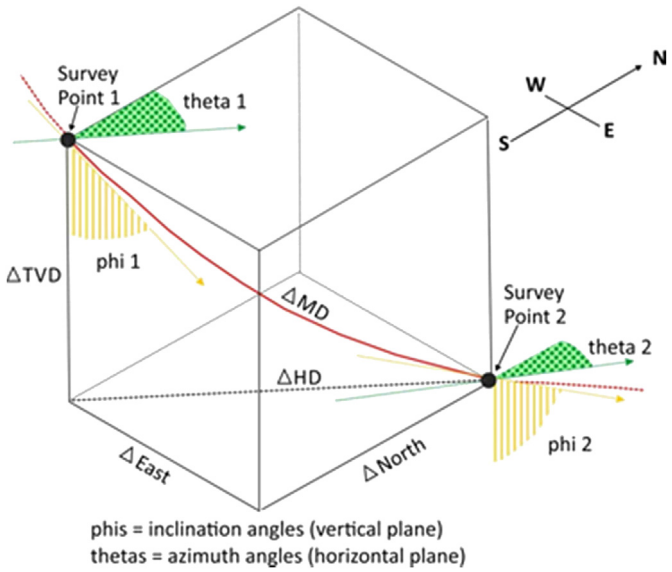


Fig. 2. Deviated well trajectory survey-calculation variables. Modified after Adams and Charrier, 1985.

$$\Delta\text{Vertical} = \frac{\Delta\text{MD}\cdot(\sin(\varnothing_2) - \sin(\varnothing_1))}{(\varnothing_2 - \varnothing_1)} \quad (6)$$

Equations (4)–(6) provide the north-south, east-west and true vertical depth (TVD) at any point along specific curved segments of a wellbore trajectory.

Equation (7) then calculates the true measured depth (TMD) of the wellbore trajectory, which is the objective function for this study:

$$\text{True Measured Depth (TMD)} = D_{\text{KOP}} + D_1 + D_2 + D_3 + D_4 + D_5 + \text{HD} \quad (7)$$

D_1 to D_5 are calculated measured depths of specific segments of the wellbore trajectory illustrated in Fig. 2 applying was calculated considering the above equations with the detailed calculations explained by equations (4)–(18) published by Shokir et al. (2004).

2.2. Example wellbore objectives and constraints

When drilling a deviated wellbore typically drilling engineers are provided with a target objective in terms of geographic co-ordinates, a true vertical depth (TVD) to an objective reservoir formation (with some error limits on that TVD), a surface location from which to commence the wellbore, and an objective lateral length (perhaps horizontal, or perhaps at a specified angle to match the dip of the objective reservoir formation) to drill within the reservoir. Such objectives are typically transformed into a set of constraints applied to establish acceptable wellbore trajectories to reach the specified target. The objective function then becomes the minimization of true measured depth for wellbore trajectories that satisfy all of the imposed constraints.

The constraints are typically classified into two groups:

1) *Operational constraints*, which normally are taken into consideration when a well trajectory is designed. The operational constraints include geological and geometrical limitations, technological shortages, controlling buildup and drop-off rates, managing the hold angle value, etc.
2) *Non-negative constraints*, are those which prevent negative values being considered for certain variables, e.g. measured depth.

The constraints imposed on the objective function of the example wellbore trajectory modeled in this study are presented in Table 1 and are the same as those studied by Shokir et al. (2004).

3. Particle swarm optimization (PSO) methodology

3.1. PSO algorithms

The PSO algorithm was inspired from the observed social behaviors of some animals, such as birds and fish. PSO is a computational algorithm which is based on iteration (Kennedy and Eberhart, 1995). This algorithm uses a matrix of random of

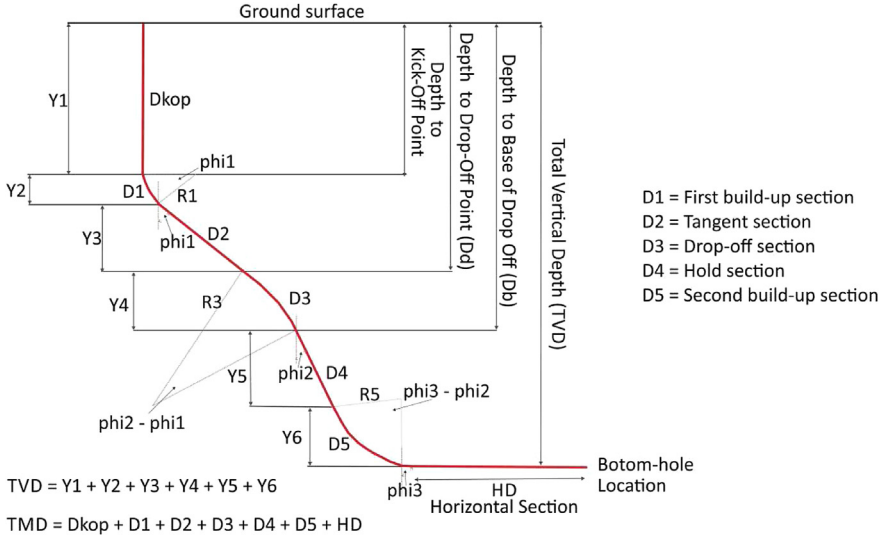


Fig. 3. Vertical plane cross-section of a generic 3D-deviated-and-partially-horizontal well trajectory (modified after Shokir et al., 2004). See nomenclature for definitions/explanations of labeled angles and wellbore section lengths.

Table 1
Constraints imposed on the objective function of the example wellbore trajectory modeled (as specified by Shokir et al., 2004).

The variables	Variable constraints imposed on generic wellbore design
Target true vertical depth (TVD)	Min. TVD = 10,850 ft. Max. TVD = 10,900 ft.
Lateral section length (HD)	2500 ft.
Dogleg severity	$T_1 \leq 5°/100$ ft., $T_2 \leq 5°/100$ ft., $T_3 \leq 5°/100$ ft., $T_4 \leq 5°/100$ ft. $T_5 \leq 5°/100$ ft.
Minimum value of inclination angles (phis)	$\phi_1 = 10°$, $\phi_2 = 40°$, $\phi_3 = 90°$
Maximum value of inclination angles (phis)	$\phi_1 = 20°$, $\phi_2 = 70°$, $\phi_3 = 95°$
Minimum value of azimuth angles (thetas)	$\theta_1 = 270°$, $\theta_2 = 270°$, $\theta_3 = 270°$ $\theta_4 = 330°$, $\theta_5 = 330°$, $\theta_6 = 355°$
Maximum value of azimuth angles (thetas)	$\theta_1 = 280°$, $\theta_2 = 280°$, $\theta_3 = 280°$ $\theta_4 = 340°$, $\theta_5 = 340°$, $\theta_6 = 360°$
Kick-off point depth (TVD)	Min. $D_{KOP} = 600$ ft. Max. $D_{KOP} = 1000$ ft.
Second build point depth (TVD)	Min. $D_D = 6000$ ft. Max. $D_D = 7000$ ft.
Third build point depth (TVD)	Min. $D_B = 10{,}000$ ft. Max. $D_B = 10{,}200$ ft.
Casing setting depth after first build (TVD)	Min. $C_1 = 1800$ ft. Max. $C_1 = 2200$ ft.
Casing setting depth after 2nd build (TVD)	Min. $C_2 = 7200$ ft. Max. $C_2 = 8700$ ft.
Casing setting depth after 3rd build (TVD)	Min. $C_3 = 10{,}300$ ft. Max. $C_3 = 11{,}000$ ft.

candidates. Hence, it is similar to other evolutionary algorithms such as Genetic Algorithms (GA) and Imperialist Competitive Algorithms (ICA). In contrast with GA, PSO has no operators, such as mutation and cross-over, so the PSO solution process is more similar to ICA than GA. Each particle, called an "element", which are comparable with the terms "chromosome" used in GA and "country" used in ICA.

The PSO algorithm is composed of a number of swarms of particles which stochastically are assigned initial values, either randomly or as seed values. A velocity vector and position vector are attributed to each particle. Each particle moves around the n-dimensional space of the feasible solutions to find the next optimum candidate considering pre-determined competence criteria. Problem space dimensions are equal to the number of parameters (i.e., unknown variables), which define the problem. A memory is addressed to the best previous position of each particle, and a general memory is assigned as the best position found by the last iteration. Taking into account these memories, the particle positions for the next step in the next iteration are selected. In any iteration, particles move around in n-dimensional space of the feasible solutions until finding the best global position. Each particle updates its velocity and position considering the best previous position for the swarm and the best global position found up to that point. As and when an improved position is discovered in each iteration, it replaces the previous position, and these steps are repeated. Finding the best global position is an aspiration, but is not guaranteed (Sharma and Khurana, 2013).

PSO algorithms refer to topologies or neighborhoods in which particles are grouped into swarms (Onwunalu, 2010). Particles exchange information freely with other particles in their neighborhood. The concept of an adjacency matrix helps to identify computationally how particles interact, where particles in the rows of the matrix are in the same swarm, the *informing* particles communicating information to each other, but the particles in the columns represent the *informed* particles that contain the information gathered by other swarms evaluated so far, including the global best position. The informing particles (rows) can communicate with the informed particles (columns) to establish a new global optimum. On the other hand the informed particles typically influence the informing particles via the best global position found so far. There are several ways in which the neighborhoods (topologies) can be defined in terms of particle communication rules leading to different PSO algorithms (e.g. Onwunalo and Durlofsky, 2010, their Fig. 2).

Assume f is our objective function which must be minimized and defined in n-dimensional space (e.g. f: $R^n \ldots R$). The objective function is assigned a vector and calculates an output for which its veracity and precision is measured in relation to its appropriateness. The goal is to find a vector such as "a" where $f(a) < f(b)$ for all candidates in the search space. Such a vector is called the best global position. In the PSO algorithm, in order to determine the velocity vector, the Inertia component, Cognitive component and the Social component are three determinant parameters calculated. The Inertia component (ω) applies a degree of continuity from one iteration to the next and helps prevent particles moving outside the problem boundaries. The Cognitive component (φ_p) moves a particle toward its best previous position in the swarm. The Social component (φ_g) moves the particle toward the best global position found up to that point. These three components have different roles in the PSO optimization algorithm. In each iteration, the next position of particles is based on the calculation of equations (8) and (9) (Engelbrecht, 2005; Onwunalu, 2010; Onwunalu and Durlofsky, 2010). There are three components used to calculate the velocity of each particle in equation (8): the Inertia component (ω); the

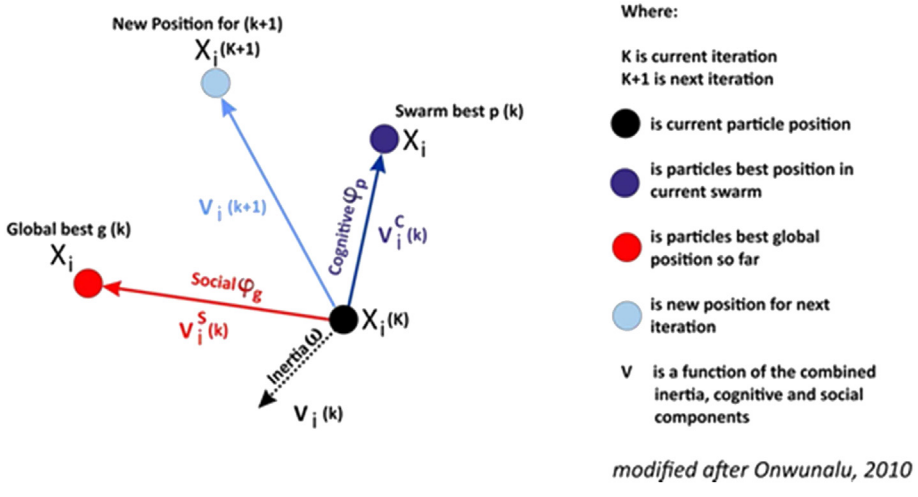


Fig. 4. PSO velocity components and update position for a particle X_i from iteration (k) to iteration ($k + 1$).

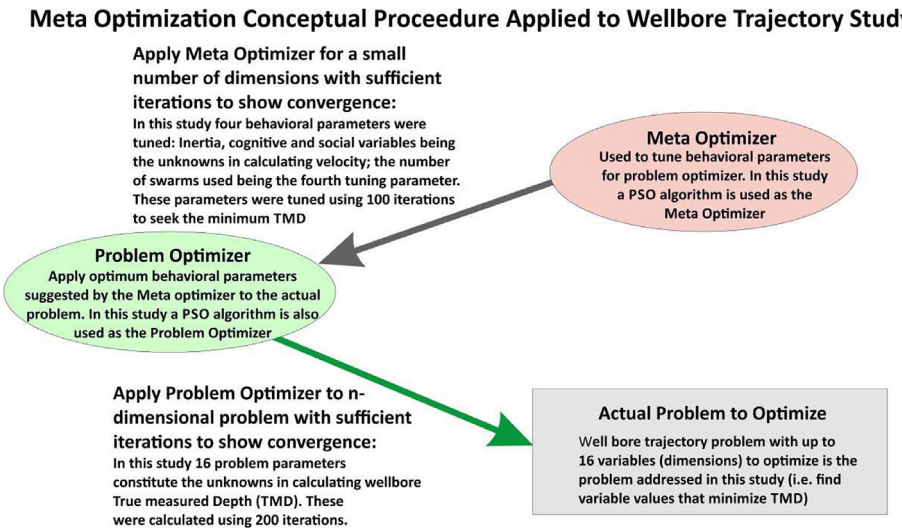


Fig. 5. The concept of Meta optimization (modified after Pedersen, 2010a,b). One optimizer is used in an offline manner to tune the behavioral parameters of the optimizer that is ultimately to be used to optimize the actual problem. In this study a PSO algorithm is used for the Meta optimizer and the problem optimizer in order to optimize wellbore trajectory.

Table 2
PSO behavioral parameter ranges for a problem with five dimensions suggested by Pedersen and Chipperfield (2010). Behavioral parameters used in that study were the number of particles per swarms (S), inertia (W), cognitive (C_1) and social (C_2) values shown.

Problem dimensions	Fitness evaluations	PSO parameters			
		S	W	C_1	C_2
5	1000	63/47	−0.3593/ −0.1832	−0.7238/0.5287	2.0289/3.1913

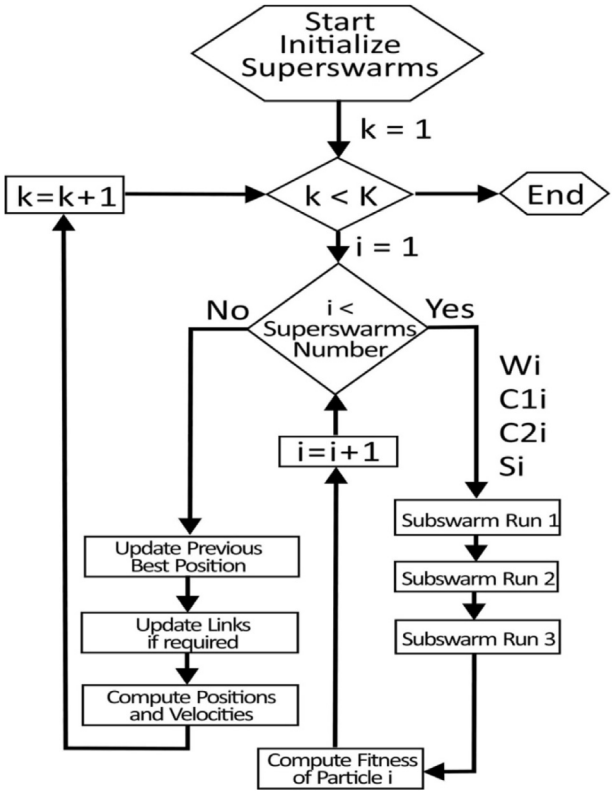


Fig. 6. Flowchart illustrating the overall Meta optimization procedure. Modified after Onwunalu, 2010, Fig. 4.2.

Cognitive component (φ_p); and, the Social component (φ_g) as illustrated in Fig. 4 for a two dimensional search space.

$$V_{i,d} = \omega V_{i,d} + \varphi_p r_p \left(p_{i,d} - X_{i,d}\right) + \varphi_g r_g \left(g_d - X_{i,d}\right) \quad (8)$$

where:

$V_{i,d}$: *Velocity* of ith particle and for dth dimension
ω: *Inertia* component (referred to as W in following tables and figures)
φ_p: *Cognitive* component (referred to as C_1 in following tables and figures)
φ_g: *Social* component (referred to as C_2 in following tables and figures)
r_p, r_g: are random numbers uniformly distributed between 0 and 1, applied to the cognitive and social components, respectively.

Table 3
Behavioral parameters recommended for the PSO Meta optimization to solve a 5-dimensional problem (Pedersen, 2010a,b). These were used initially to search for the optimal PSO behavioral parameters to apply when optimizing the defined wellbore trajectory problem.

Meta optimizer algorithm behavioral parameters	S	W	C_1	C_2
	47	−0.2	0.5	2.5

Table 4
Superswarm parameter ranges, selected based upon conclusions of Pedersen (2010a,b), searched by the Meta optimization study to identify the optimal behavioral parameters to use as input for the tuned PSO optimization analysis of the defined wellbore trajectory problem. The values shown represent upper and lower value constraints applied to the parameters.

Superswarm parameter ranges searched by meta optimization	S	W	C_1	C_2
	20–70	−0.5 to 0	−0.5 to 0	0–4

Table 5
Number of iterations and overall runs used to derive the average optimum behavioral parameter results for the Meta optimization.

Subswarm code parameters	Iterations	Number of runs (used to calculate an average)
	100	3

Table 6
Behavioral parameters established by three tuned runs of the Meta optimization algorithm. The non-tuned behavioral parameters, shown for comparison, are those suggested by Pedersen (2010a,b) for solving 10–20 dimensional problems.

PSO behavioral parameters	Tuned Run 1	Tuned Run 2	Tuned Run 3	Non-tuned
C_1	−0.414658484	0.006843348	−0.059926157	−0.2699
C_2	2.354873241	3.999823467	3.812480878	4.2373
W	0	−0.127441595	0	−0.67
Number of swarms	65	50	20	45
Optimized TMD (feet)	15,023.55	15,023.63	15,023.73	15031.31

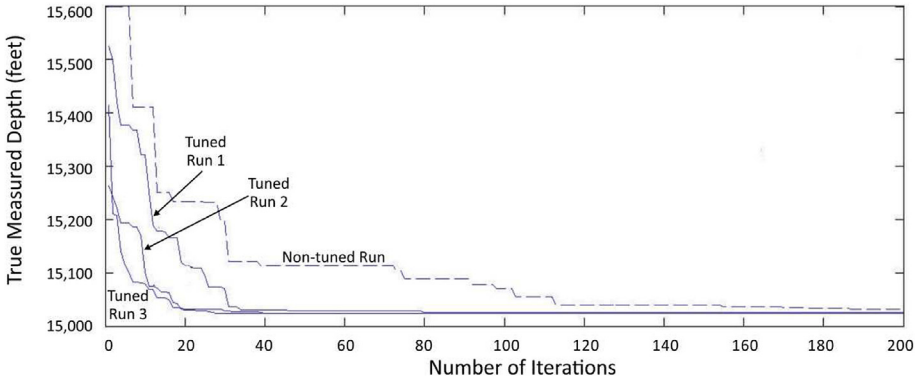


Fig. 7. Comparison between tuned and non-tuned parameter operations in two hundred iterations of the PSO algorithm, applying the behavioral parameters listed in Table 6 for wellbore trajectory optimization.

$p_{i,d}$: Best particle value found in the current swarm for the dth dimension of ith particle, up to current iteration
$X_{i,d}$: Value of the dth dimension of ith particle
g_d: Best global value found for the dth dimension by all particles in the current and previous swarms up to current iteration, which, at the end of the last iteration, contains the optimal values the dth dimension.

$$X_i + X_i + v_i \tag{9}$$

In equation (9), the value of the ith particle, X_i, for each dimension, is adjusted by the calculated velocity to provide the particle values to test in the next iteration of the model.

3.2. Meta optimization algorithms

Tuning the particle swarm optimization algorithm parameters has great impact on its efficiency. This fact is well established in the literature (e.g. Pedersen, 2010a,b). Meta optimization is an optimization method to tune the behavioral parameters of another optimization method (Fig. 5). Behavioral parameters are those that control the search behavior of an optimization algorithm (e.g. the velocity components in the PSO algorithm).

Mercer and Sampson (1978) used Meta optimization to find optimum genetic algorithm (GA) parameters. Meta optimization may be interchangeably referred to by other names such as "Meta evaluation", "Super optimization", "Automated parameters calibration" and "Hyper heuristic".

Some optimization methods such as GA and differential evolution (DE) have some parameters specifying the behavior and efficiency of the optimization algorithm for a certain problem, which must be determined. Determining these parameters manually can be a difficult and time consuming procedure. Furthermore, it often fails to provide a good perspective regarding their effects on the algorithm's efficiency. By changing behavioral parameters of an optimization method and graphing the algorithm operation, their impacts on the searching algorithm's performance can be seen. Such procedures are only realistic for problems which can be solved quickly. When a problem involves many parameters, the required time for solving the calculations increases exponentially as the number of parameters increases. So a powerful and effective method is absolutely needed to search the n-dimensional space of the behavioral parameters. Meta optimization is a simple way of finding good behavioral parameters for an optimizer by applying another optimizer to tune the behavioral parameters.

Meta optimized GA techniques were applied by Grefenstette (1986) and Keane (1995). Meissner et al. (2006) and Pedersen and Chipperfield (2010) applied Meta optimized PSO techniques. Pedersen and Chipperfield (2010) also applied Meta optimized DE.

Table 7
Meta and tuned optimization setup and assumptions for the wellbore trajectory analysis.

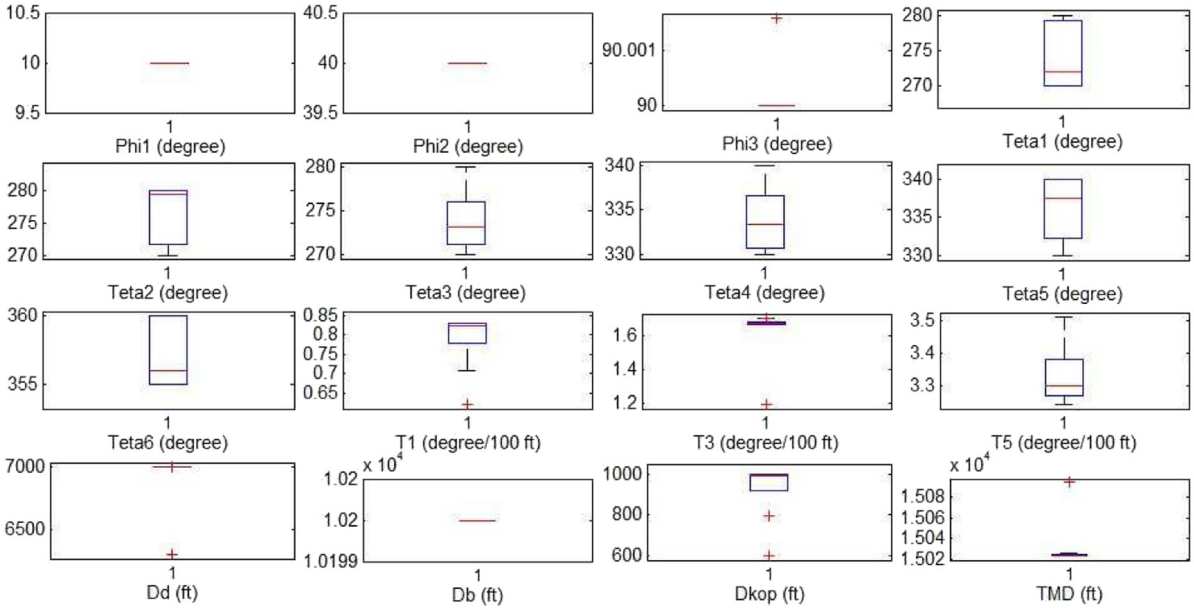
	Meta (tuning) optimizer	Applying tuned optimizer
Optimization steps	Step 1	Step 2
Optimization algorithm applied	PSO	PSO
Algorithm behavioral parameters	C_1, C_2, ω, S Selected from Table 2, based upon those suggested by Pedersen and Chipperfield (2010)	C_1, C_2, ω, S Selected from the optimum results of three Meta runs of step 1
Objective function	Search a 5-dimensional problem to find best behavioral parameters with which to optimize TMD	Search a 16-dimensional problem to find best (minimum) TMD
Number of particle positions or superswarms (S)	Initially 47 (from Table 3), but ultimately reduced to 5 based on computation time versus results	65 (from Meta Run 1) 50 (from Meta Run 2) 20 (from Meta Run 3)
Number of iterations or subswarms (K)	100	200
Unknowns and variables evaluated	S, ω, C_1, C_2, PSO behavioral parameters	ϕ_1, ϕ_2, ϕ_3, θ_1, θ_2, θ_3, θ_4, θ_5, θ_6, T_1, T_3, T_5, TVD, D_{KOP}, D_B and D_D Wellbore trajectory variables
Constraints	S: 20 to 70 ω: −0.5 to 0 C_1: −0.5 to 0 C_2: 0–4	Presented in Table 2

Meta optimized ant colony optimization was applied by Birattari et al. (2002), and Smit and Eiben (2009) compared various techniques of Meta optimization.

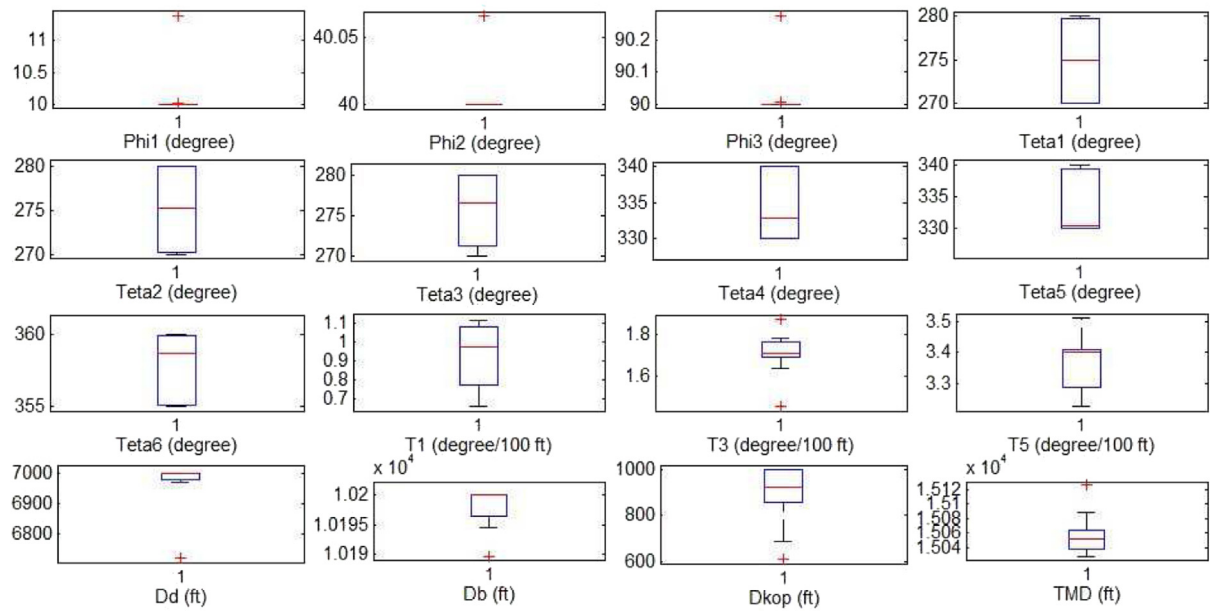
3.3. Tuning behavioral parameters in PSO meta optimization

The operation and efficiency of PSO depends upon the behavioral algorithm parameters. Meta optimization is applied in the model to tune the behavioral parameters of PSO. In Meta optimization for the wellbore trajectory design explored here, two PSO algorithms are applied. The first PSO algorithm with its swarms (i.e., referred to here as the "Superswarm"), tune and optimize the PSO behavioral parameters, while the second PSO algorithm (i.e., referred to here as the "Subswarm") tries to find optimum parameters to solve the objective function. This follows the methodology proposed by Onwunalu (2010, his chapter 4), where a more detailed description of the Superswarm and Subswarm components are provided. Meta optimization can either be used to evaluate certain benchmark or known reference problems (e.g., Pedersen, 2010a,b) and extrapolate behavioral parameters from those problems, or, it can be applied in methodologies that aim to solve the problem directly.

In the first approach, it is assumed that behavioral parameters that have previously been applied to benchmark problems are suitable for other problems, and specifically the problem in hand. This was the



(a) tuned PSO-algorithm variable distributions from ten evaluations



(b) Non-tuned PSO-algorithm variable distributions from ten evaluations

Fig. 8. Box-and-whisker plots for (a) tuned-PSO algorithm run 1 and (b) non-tuned PSO algorithm each run ten times with 200 iterations. These plots show the statistical dispersion for each of fifteen variables and the TMD objective function in both cases, with the red horizontal line within the box representing the median value and the box showing the middle two quartiles of the distribution (i.e. the edges of the box are the 25th and 75th percentiles). Whiskers extend to the outer limits of the distribution excluding outliers, which are shown as red crosses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Comparison between PSO, GA, WELLDES optimizations and conventional method for designing the trajectories of deviated wellbores.

Method	PSO design This study	Conventional design From Table 4 Shokir et al. (2004)	WELLDES design From Table 4 Shokir et al. (2004)	GA design From Table 4 Shokir et al. (2004)
HD, ft.	2500	2500	2500	2500
ϕ_1, °	10.0	15	13.92	13.77
ϕ_2, °	40.0	45	40.02	42.131
ϕ_3, °	90.0	90	90.05	90
θ_1, °	270.0	–	280.1	279.976
θ_2, °	280.0	–	280.1	279.72
θ_3, °	275.953	–	280.1	275.02
θ_4, °	331.545	–	332.4	332.3
θ_5, °	340.0	–	332.3	334.32
θ_6, °	355	–	332.5	355
T_1, °/100 ft.	0.829	0.9	0.7	1.675
T_3, °/100 ft.	1.666	2	1.2	1.431
T_5, °/100 ft.	3.243	3.5	2.6	2.413
D_D, ft. (TVD)	7000	6600	6498.7	6804.37
D_B, ft. (TVD)	10,200	11,260	10,003.5	10,004.48
D_{KOP}, ft. (TVD)	1000	625	627	987.975
TMD, ft.	15,023.6	15,565	15,498	15,496.7

approach used in this study (Table 2). A Meta optimization consists of three elements; Superswarm, Subswarm and an objective function, or functions, which are optimized (in the model described that is a mini mization) by Subswarms. Each set of Superswarm particles is related to a set of Subswarms. To evaluate a Superswarm's quality, the sub-sw arm algorithm is run and the final optimum value found is attributed to its super-swarm, as shown in the optimization flowchart (Fig. 6). Table 2 shows the Meta optimization parameter setup for a 5-dimensi onal problem presented by Pedersen and Chipperfield (2010). The nu mber of swarms (S) represents the number of times particle positions are adjusted by the velocity function in each iteration of the algorithm.

4. Optimization results for wellbore trajectory study

Applying a Meta-heuristic approach involving a PSO Meta optimization algorithm (i.e., using a higher-level PSO algorithm designed to optimize a lower-level PSO algorithm), and considering a series of constraints imposed on the example wellbore objective (Table 1), the results presented below highlight how this approach can be used to effectively minimize true measured depth (TMD), the objective function, of the wellbore using the radius of curvature method.

4.1. PSO behavioral parameters established by superswarms

The values of the inertia (W), cognitive (C_1), social (C_2) and number of swarms (S) behavioral parameters for the particle swarm algorithm used for the Superswarm in the Meta optimization are listed in Table 3. These behavioral parameters are constrained in the Meta analysis to the ranges specified in Table 4 and those ranges are searched via repeated iterations to find the optimum values for each of these behavioral parameters.

In the results of the Meta analysis reported in this study, just five particle positions ($N^{super} = 5$) searched the five-dimensional space of the model (i.e. four behavioral parameters plus the unknown TMD) to find optimum behavioral parameters and the Meta model was iterated one hundred times ($K = 100$). The lower number for S ultimately used than the 47 (i.e., $N^{super} = 47$) suggested in Table 3 was adopted because it reduced computation time, yet still produce an improvement on the non-tuned PSO algorithm. The behavioral-parameter space searched by the Meta optimization, i.e. the constraints applied to S, W, C_1 and C_2 are listed in Table 4.

To derive more statistically-reliable optima, the Meta optimization algorithm is iterated one hundred times in each run and the process duplicated in three separate runs to demonstrate that the results are reproducible (Table 5).

The three optimum points obtained for the behavioral parameters represents the "tuning" process (Table 6). Tuning refers to the procedure of finding the most appropriate input/behavioral parameters for an evolutionary optimization algorithm. Because of implicit answers in Meta-heuristic optimization, the small variations in the optima values obtained by each of the three runs of the Meta optimization algorithm is acceptable, and to be expected.

4.2. Tuned PSO optimization results from subswarms

One set of the behavioral parameters listed in Table 6 is then used as input for the tuned PSO algorithm for the sixteen-variable dimensions of the well trajectory problem. In the analysis shown in this study the results of "Meta run 1" were used as behavioral-parameter

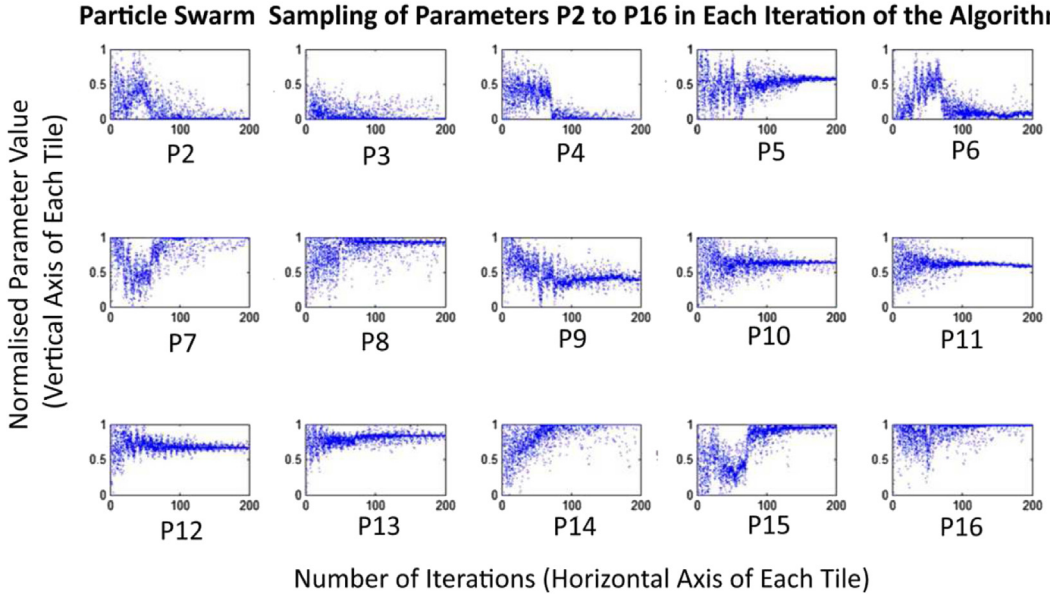


Fig. 9. Sampling history for the particle swarm optimization algorithm applied in this study to the generic wellbore trajectory optimization problem. Note parameter $P1$ in this case, the length of the horizontal is a constant, so it does not vary through the optimization process, although in other cases it could be varied. The example shown is for a non-tuned case.

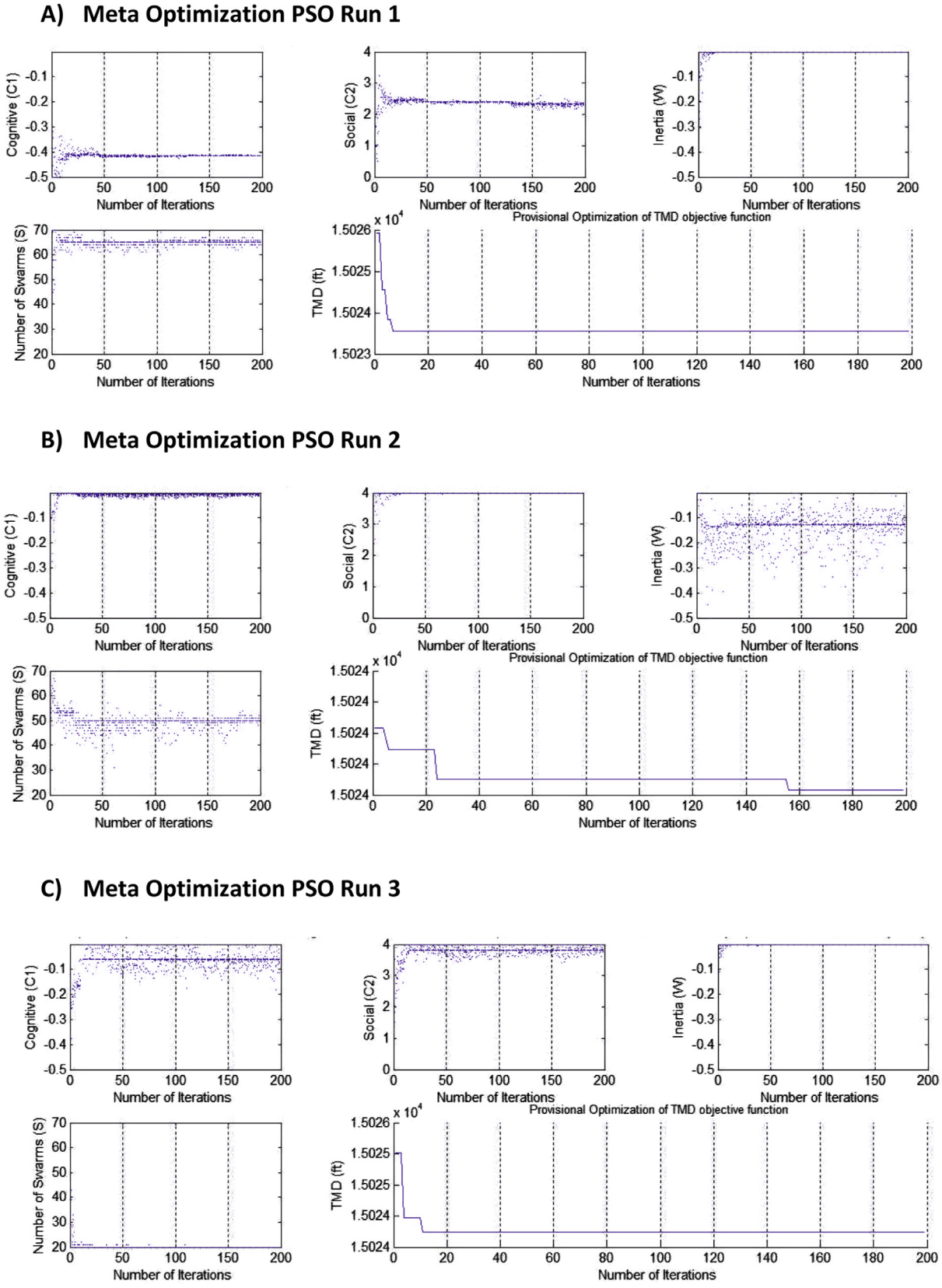


Fig. 10. A, B and C represent Meta-optimization Runs 1, 2 and 3, respectively. These runs were conducted to establish optimal PSO behavioral parameters for the wellbore trajectory algorithm. The horizontal axes show the number of iterations in all component graphics.

input for the tuned PSO optimizer. In the tuned PSO analysis reported in this study sixty-five particle positions ($N^{super} = 65$) searched the sixteen-dimensional space of the model to find the optimum TMD for the well trajectory, and the tuned model was iterated two hundred times ($K = 200$) for the swarms to obtain an optimized TMD solution.

Fig. 7 shows the performance of the tuned and non-tuned PSO algorithms in reducing the TMD value of the wellbore trajectory (i.e., equation (7)). This figure illustrates the TMD value in each iteration of the algorithms versus the iteration number. Fig. 7 highlights that the tuned-parameter algorithms lead to better optimum values being found, and those algorithms converge more rapidly to those optima, in comparison with the non-tuned parameter algorithms. For instance, in Fig. 7, the tuned-parameter algorithms all reached a 15,100 solution or better within 25-subswarm iterations, while the non-tuned-param eter algorithm runs did not reach a 15,100 solution until some 75 iterations. Therefore we conclude that the drilling trajectories designed using a PSO algorithm based upon behavioral parameters tuned using Meta optimization (Table 6) are improved and found more quickly.

A summary of the setup for the Meta and tuned optimization steps is listed in Table 7.

Box-and-whisker plots (Fig. 8) confirm that the results shown in Fig. 7 are reproducible. In the tuned PSO box-and-whisker plot (Fig. 8(a)) the behavioral parameters from tuned run 1 (Table 6) were applied to ten separate evaluations of the algorithm, and the values of the 15 variables and the TMD objective function involved in each solution of the wellbore trajectory problem were recorded to form the statistical distributions on which the plots are based. In the non-tuned PSO box-and-whisker plot (Fig. 8(b)) the behavioral parameters from non-tuned run (Table 6) were applied to ten separate evaluations of the algorithm and the values of the 16 variables involved in the wellbore trajectory problem were recorded to form the statistical distributions on which the plot is based. It is clear when comparing Fig. 8(a) and (b) that the tuned-PSO algorithm found a better optimum TMD for the wellbore trajectory with much less dispersion in the values of each variable involved. The results shown in Figs. 7 and 8 together show that the tuned-PSO algorithm finds it optimum value much more rapidly and effectively than the non-tuned PSO algorithm.

Table 8 lists a comparison of the results for the tuned PSO algorithm applied in this study with the results of Shokir et al. (2004), including the WELLDES program (Ahmed, 2000), the conventional wellbore trajectory planning method and the genetic algorithm.

5. Discussion

Fig. 9 shows how the PSO algorithm navigates through parameter space in each dimension. In this figure, positions of individual particles in each iteration are plotted versus the iteration number. This figure has 15 tiles showing the 15 parameters that are being perturbed to obtain an optimum solution. The tiles in Fig. 9 display dimensionless scaled parameters values (0, 1) in the vertical axis and iteration numbers (0, 200) in the horizontal axis. The trends shown for each parameter in Fig. 9 confirm that the PSO algorithm as applied in this study is able to maintain population diversity.

Alteration in variable values and their aggregation to find the optimum answer can be seen for each iteration in Fig. 9, showing that non-tuned PSO runs do converge to a solution, but take a significant number of iterations to do so. Monitoring the Fig. 9 parameter performance charts, makes it possible to determine whether searching power of the PSO algorithm in n-dimensional space seeking an optimum solution has successfully concluded or not. Searching concludes when all points in all performance charts converge to a single point (the optimum answer).

Results from the Meta optimized PSO deviated wellbore trajectory design are compared (Table 8) to the results published by Shokir et al. (2004) for a conventional wellbore design (i.e., based on a trial and error method) the WELLDES program (i.e., based on a sequential unconstrained minimization technique, Ahmed (2000)) and the genetic algorithm method. The optimum solution proposed by the Meta optimized or tuned PSO algorithm (i.e., TMD = 15,023.6 ft.) outperforms the other algorithms. The tuned PSO algorithm optimum result (Table 8) also satisfies all of the constraints and limitations imposed on the generic wellbore trajectory.

The tuning of the behavioral parameters (S, W, C_1, C_2) is the objective of the Meta optimization process. The Meta optimization algorithm involves three runs of the which are averaged to provide a provisional value for the TMD objective function. Fig. 10 illustrates example outputs of behavioral parameters and the objective function for three runs of the Meta optimizer.

6. Conclusions

Detailed deviated wellbore trajectory design can be successfully optimized using recently developed Meta-heuristic particle swarm algorithms. This is demonstrated in this study using a generic deviated well design (i.e. based upon the example published by Shokir et al. (2004) for a deviated well drilled in Egypt).

The specific conclusions drawn from this study are:

- Behavioral-parameter tuning has a significant effect on the convergence of the Meta-heuristic particle swarm optimization algorithm. This is clear from the comparisons between tuned and non-tuned runs shown in Fig. 7. The search capability of the PSO algorithm decreases as the dimensions of a problem increase. In such cases, tuned PSO behavioral parameters tend to achieve better global optimization results faster than non-tuned PSO algorithms. A conclusion that is supported by the results illustrated in Figs. 7 and 8. The results of this study therefore support the more general conclusion that computational benefits are likely to be obta ined by tuning the behavioral parameters of a PSO algorithms app lied to complex problems involving multiple non-linear variables.
- A tuned PSO algorithm proved to be a powerful and rapid optimization method for solving the non-linear deviated wellbore trajectory problem tackled in this study.
- PSO is an evolutionary algorithm based on gaining system knowledge making it a promising method for potentially solving a diverse range of engineering problems, particularly when applied with Meta-optimization routines.

Nomenclature

Wellbore trajectory system variable definitions (modified after Shokir et al., 2004). Vertical plane variables are illustrated in Fig. 2

ϕ_1, ϕ_2, ϕ_3	first, second, and third hold angles, degrees
θ_1	azimuth angle at kick off point, degrees
θ_2	azimuth angle at end of first build portion, degrees
θ_3	azimuth angle at end of first hold section, degrees
θ_4	azimuth angle at end of second build or drop portion, degrees
θ_5	azimuth angle at end of second hold section, degrees
θ_6	azimuth angle at end of third build portion, degrees
T_1	dogleg severity of first build portion, °/100 feet
T_2	dogleg severity of first hold portion, °/100 feet
T_3	dogleg severity of second build or drop portion, °/100 feet
T_4	dogleg severity of second hold section, °/100 feet
T_5	dogleg severity of third build portion, °/100 feet
TVD	true vertical depth of the well at total depth (TD), feet
D_{KOP}	true vertical depth of kick off point, feet

D_D true vertical depth of the well at the top of drop-off section (top of second build section), feet
D_B true vertical depth of the well at the end of drop-off section (top of third build section), feet
HD lateral length (horizontal length), feet

Appendix A. MATLAB code listing for meta-optimized particle swarm algorithm to solve deviated wellbore trajectories using the radius of curvature method

The following MATLAB code evaluates the non-tuned PSO algorithms developed and applied in this study. It is annotated with comments to clarify the function of each section of the code.

```
clc
clear
%Place the final optimized result (i.e.16 variables plus TMD objective function into the IU matrix, which is used to draw boxplots (see Figure 8)
IU=zeros(17,100);
COUNTER=1;
%Run the whole algorithm RWTPth times. For one execution of the algorithm RWTPh equals 1. To find an average of multiple runs RWTPh should be greater than 1 (e.g. 10 to provide sufficient data for boxplots)
RWTPth=1;
while COUNTER<RWTPth+1
k=0;h=0;t=0;a1=0;E1=size(1,400);mat2=size(60,60);v = size(60,60);
mat = size(60, 60);vi=size(60,60);pv=size(60);E=size(60);n = 0;
G=size(200);
%For the initial meta-optimization step, values for PSO behavioral parameters c1, c2 and w from Pederson 2010 are applied. Subsequently the values for the behavioral parameters c1, c2 and w from the best TMD solution in Table 6 (i.e. Run1) are applied.
c1=-0.414658484;% Pedersen(2010)=0.2699;
c2=2.354873241;% Pedersen(2010)=4.2373;
w=0;% Pedersen(2010)=-0.67;
%Number of variables in problem.
x = 16;
%For the initial meta-optimization step the number of particle swarms (ns) applied is 45 (Pedersen, 2010). Subsequently the number of swarms applied is 65 from the best TMD solution in Table 6 (i.e. Run1).
ns=65;
iterations=200;
format long
disp('Random TMD');
%%
%Variable constraints: lower limits cl; upper limits cu
cl(1) = 2500;cl(2) = 10;cl(3) = 40;cl(4) = 90;cl(5) = 270;
cl(6) = 270;cl(7) = 270;cl(8) = 330;cl(9) = 330;cl(10) = 355;
cl(11) = 0;cl(12) = 0;cl(13) = 0;cl(14) = 6000;cl(15) = 10000;
cl(16) = 600;cu(1) = 2500;cu(2) = 20;cu(3) = 70;cu(4) = 95;
cu(5) = 280;cu(6) = 280;cu(7) = 280;cu(8) = 340;cu(9) = 340;
cu(10) = 360;cu(11) = 5;cu(12) = 5;cu(13) = 5;cu(14) = 7000;
cu(15) = 10200;cu(16) = 1000;
%Initialize the particle positions are generated with a uniformly distributed random vector between the lower and upper limits of each variable.
while n<ns;
for i = 1 : x
v(i,n+1) = cl(i) + (cu(i)-cl(i)).*rand;
end
%Wellbore trajectory radius of curvature calculations
rad = (3.14159 / 180);
R1 = 100 / (v(11, n + 1)* rad);
R3 = 100 / (v(12, n + 1)* rad);
R5 = 100 / (v(13, n + 1)* rad);
D1 = R1 * (((v(6, n + 1) - v(5, n + 1)) * rad) ^ 2 * (sin(v(2, n + 1) * rad / 2)) ^ 4 ...
   + ((v(2, n + 1) * rad) ^ 2)) ^ (0.5);
D2 = (v(14, n + 1) - v(16, n + 1) - D1 * (sin(v(2, n + 1) * rad) - sin(0)) / (v(2, n + 1) * rad - 0)) / cos(v(2, n + 1) * rad);
D3 = R3 * (((v(8, n + 1) - v(7, n + 1)) * rad) ^ 2 * (sin((v(3, n + 1)...
   + v(2, n + 1)) * rad / 2)) ^ 4 + ((v(3, n + 1) - v(2, n + 1)) * rad) ^ 2) ^ (0.5);
D4 = (v(15, n + 1) - v(14, n + 1) - D3 * (sin(v(3, n + 1) * rad) - sin(v(2, n + 1) * rad)) / (v(3, n + 1) * rad - v(2, n + 1) * rad)) / cos(v(3, n + 1) * rad);
D5 = R5 * (((v(10, n + 1) - v(9, n + 1)) * rad) ^ 2 * (sin((v(4, n + 1) + v(3, n + 1)) * rad / 2)) ^ 4 ...
   + ((v(4, n + 1) - v(3, n + 1)) * rad) ^ 2) ^ (0.5);
% z calculates the TMD of the entire wellbore
z = v(1, n + 1) + D1 + D2 + D3 + D4 + D5 + v(16, n + 1);
cas1 = v(16, n + 1) + D1 * (sin(v(2, n + 1) * rad) / (v(2, n + 1) * rad));
cas2 = cas1 + D2 * cos(v(2, n + 1) * rad) + D3 * ((sin(v(3, n + 1) * rad) - sin(v(2, n + 1) * rad)) / ((v(3, n + 1) - v(2, n + 1)) * rad));
cas3 = cas2 + D4 * cos(v(3, n + 1) * rad) + D5 * ((sin(v(4, n + 1) * rad) - sin(v(3, n + 1) * rad)) / ((v(4, n + 1) - v(3, n + 1)) * rad));
```

```
phi1=v(2,n+1);
phi2=v(3,n+1);
phi3=v(4,n+1);
Hold1=D2*cos(phi1*rad);
Hold2=D4*cos(phi2*rad) ;
tvdbuild1=(D1*(sin(rad*(phi1))-sin(rad*0)))/(phi1*rad);
tvddrop=D3*(sin(rad*phi2)-sin(rad*phi1))/((phi2-phi1)*rad);
tvdbuild2=D5*(sin(rad*phi3)-sin(rad*phi2))/((phi3-phi2)*rad);
TVD=tvdbuild1+tvdbuild2+tvddrop+Hold1+Hold2+v(16,n+1);
%The following wellbore constraints are also applied.
if D1 > 0 && D2 > 0 && D3 > 0 && D4 > 0 && D5 > 0 && cas1 > 1800 &&...
cas1 < 2200 && cas2 > 7200 && cas2 < 8700 && cas3 > 10835 &&...
cas3 < 10885 && TVD>10850 && TVD<10900
%Values of matrix v are only passed to matrix mat if they satisfy all the wellbore constraints (i.e. provide a valid solution).
for i = 1 : x
mat(i, n+1) = v(i, n + 1);
end
mat(i+1, 1 + n) = z;
n = n + 1;
%Display the initial values which satisfy all constraints
disp(z);
end
end
disp('Search for better answers');
%Start timer
tic
%%
% Input first column of mat as best global values into G. If better solutions are found in subsequent runs then the values in matrix G column 1 are replaced with those better values.
for i=1:x
G(i,1)=mat(i,1);
end
% The best global value for the TMD objective function is placed in GV and replaced with better values as they are found.
GV=mat(17,1);
%Initialize the matrix holding each particles best known position (i.e. pi) to each particles initial position
pi=size(60,60);
for n=1:ns
for i = 1 : x+1
pi( i,  n) = mat(i, n );
end
end
%Update each particle swarm's best known position.
s=ns;
for i=1:s
if pi(17,i)<GV
for j=1:x
G(j,1)=pi(j,i);
end
GV=pi(17,i);
end
end
%Initialize each particle's velocity, vi, with a uniformly distributed random vector between the lower and upper limits of each variable.
for n=1:s
for i=1:x
pv(i)=cu(i)-cl(i);
vi( i,  n)=-pv(i) + (2*pv(i)).*rand;
end
end
%The algorithm is repeated for the specified number of iterations.
ite=0;
tic
while ite<iterations
%For each particle i
for i=1:s;
%For each dimension (variable) d
for d=1:x;
%Select uniformly distributed random numbers with which to adjust behavioral parameters applied to each particle's velocity.
rp=rand;
rg=rand;
%Update each particle's velocity.
vi( d, i)=w* vi(d,i) + c1 *rp* (pi(d,i)-mat(d,i)) + c2* rg* (G(d)-mat(d,i));
end
%Update each particle's position with the newly adjusted velocity
```

```
%Test that each adjusted variable position is within its upper and lower limits.
for d=1:x;
if mat(d,i)+vi(d,i)<cl(d)
mat2(d,i)=cl(d);
elseif mat(d,i)+vi(d,i)>cu(d)
mat2(d,i)=cu(d);
else
mat2(d,i)=mat(d,i)+vi(d,i);
end
end
%Evaluate wellbore trajectory for newly generated particle positions.
rad = (3.14159 / 180);
R1 = 100 / (mat2(11, i) * rad);
R3 = 100 / (mat2(12, i) * rad);
R5 = 100 / (mat2(13, i) * rad);
D1 = R1 * (((mat2(6, i) - mat2(5, i)) * rad) ^ 2 * (sin(mat2(2, i) * rad / 2)) ^ 4 + ((mat2(2, i ) * rad) ^ 2)) ^ (0.5);
D2 = (mat2(14, i) - mat2(16, i) - D1 * (sin(mat2(2, i) * rad) - sin(0)) / (mat2(2, i) * rad - 0)) / cos(mat2(2, i) * rad);
D3 = R3 * (((mat2(8, i) - mat2(7, i)) * rad) ^ 2 * (sin((mat2(3, i) + mat2(2, i)) * rad / 2)) ^ 4 + ((mat2(3, i) - mat2(2,
i)) * rad) ^ 2) ^ (0.5);
D4 = (mat2(15, i) - mat2(14, i) - D3 * (sin(mat2(3, i) * rad) - sin(mat2(2, i) * rad)) / (mat2(3, i) * rad - mat2(2, i) *
rad)) / cos(mat2(3, i) * rad);
D5 = R5 * (((mat2(10, i) - mat2(9, i)) * rad) ^ 2 * (sin((mat2(4, i) + mat2(3, i)) * rad / 2)) ^ 4 + ((mat2(4, i) -
mat2(3, i)) * rad) ^ 2) ^ (0.5);
z = mat2(1, i) + D1 + D2 + D3 + D4 + D5 + mat2(16, i);
mat2(17,i)=z;
cas1 = mat2(16, i) + D1 * (sin(mat2(2, i) * rad) / (mat2(2, i) * rad));
cas2 = cas1 + D2 * cos(mat2(2, i) * rad) + D3 * ((sin(mat2(3, i) * rad) - sin(mat2(2, i) * rad)) / ((mat2(3, i) -
mat2(2, i)) * rad));
cas3 = cas2 + D4 * cos(mat2(3, i) * rad) + D5 * ((sin(mat2(4, i) * rad) - sin(mat2(3, i) * rad)) / ((mat2(4, i) -
mat2(3, i)) * rad));
phi1=mat2(2,i);
phi2=mat2(3,i);
phi3=mat2(4,i);
Hold1=D2*cos(phi1*rad);
Hold2=D4*cos(phi2*rad) ;
tvdbuild1=(D1*(sin(rad*(phi1))-sin(rad*0)))/(phi1*rad);
tvddrop=D3*(sin(rad*phi2)-sin(rad*phi1))/((phi2-phi1)*rad);
tvdbuild2=D5*(sin(rad*phi3)-sin(rad*phi2))/((phi3-phi2)*rad);
TVD=tvdbuild1+tvdbuild2+tvddrop+Hold1+Hold2+mat2(16,i);
%The following wellbore constraints are also applied.
if D1 > 0 && D2 > 0 && D3 > 0 && D4 > 0 &&...
D5 > 0 && cas1 > 1800 && cas1 < 2200 && cas2 > 7200 &&...
cas2 < 8700 && cas3 > 10835 && cas3 < 10885  &&...
TVD > 10850 && TVD < 10900
%If constraints are all satisfied the variable values from matrix mat2 are transferred into matrix mat.
for d=1:x
mat(d,i)=mat2(d,i);
end
mat(17,i)=mat2(17,i);
%Update the particles best known position
if z<pi(17,i)
for d=1:x
pi(d,i)=mat(d,i);
end
pi(17,i)=z;
end
%Update the swarm's best known position
if z<GV
for d=1:x
G(d)=mat(d,i);
end
GV=z;
a1=TVD;
end
end
end
ite=ite+1;
figure (1)
E1(ite)=GV;
plot(1:ite,E1(1:ite),'b')
hold on
xlabel('Number of Iterations')
ylabel('True Measured Depth TMD (ft)')
end
%Place G matrix into the IU matrix for each run of the algorithm.
IU(1:16,COUNTER)=G(1:16,1);
IU(17,COUNTER)=GV;
```

```
COUNTER=COUNTER+1;
end
disp('----------------------------------')
disp('          The Optimum values found by PSO are:          ')
disp('----------------------------------')
disp('HP='); disp('phi1=');disp('phi2='); disp('phi3=');
disp('theta1=');disp('theta2='); disp('theta3=');disp('theta4=');
disp('theta5='); disp('theta6=');disp('T1='); disp('T3=');
disp('T5='); disp('Dd=');disp('Db='); disp('Dkop=');
G(1:16,1)
disp('TMD=');
disp(GV);
disp('TVD=');
disp(a1);
toc
%If more than one run of the algorithm has been calculated (i.e.RWTPth>1) draw the boxplots.
if COUNTER>2
%Draw Boxplots
subplot(4,4,1);boxplot(IU(2,1:RWTPth));
xlabel('Phi1 (degree)')
subplot(4,4,2);boxplot(IU(3,1:RWTPth));
xlabel('Phi2 (degree)')
subplot(4,4,3);boxplot(IU(4,1:RWTPth));
xlabel('Phi3 (degree)')
subplot(4,4,4);boxplot(IU(5,1:RWTPth));
xlabel('Theta1 (degree)')
subplot(4,4,5);boxplot(IU(6,1:RWTPth));
xlabel('Theta2 (degree)')
subplot(4,4,6);boxplot(IU(7,1:RWTPth));
xlabel('Theta3 (degree)')
subplot(4,4,7);boxplot(IU(8,1:RWTPth));
xlabel('Theta4 (degree)')
subplot(4,4,8);boxplot(IU(9,1:RWTPth));
xlabel('Theta5 (degree)')
subplot(4,4,9);boxplot(IU(10,1:RWTPth));
xlabel('Theta6 (degree)')
subplot(4,4,10);boxplot(IU(11,1:RWTPth));
xlabel('T1 (degree/100ft)')
subplot(4,4,11);boxplot(IU(12,1:RWTPth));
xlabel('T3 (degree/100ft)')
subplot(4,4,12);boxplot(IU(13,1:RWTPth));
xlabel('T5 (degree/100ft)')
subplot(4,4,13);boxplot(IU(14,1:RWTPth));
xlabel('Dd (ft)')
subplot(4,4,14);boxplot(IU(15,1:RWTPth));
xlabel('Db (ft)')
subplot(4,4,15);boxplot(IU(16,1:RWTPth));
xlabel('Dkop (ft)')
subplot(4,4,16);boxplot(IU(17,1:RWTPth));
xlabel('True Measured Depth (ft)')
end
```

Appendix B. MATLAB code listing for meta-optimized (tuned) particle swarm algorithm to solve deviated wellbore trajectories using the radius of curvature method

The following MATLAB code evaluates the Meta-optimized (tuned) PSO algorithms developed and applied in this study. It is annotated with comments to clarify the function of each section of the code.

Section 1

```
tic
clc
clear
%WMeta, C1Meta and C2Meta were picked up form pederson paper for problems with 5 dimensions
WMeta=-0.3593;C1Meta=-0.7238;C2Meta=2.0289;
PI=size(60,60);Global=size(200,1);ITE=0;ITERATIONS=200;VI=size(60,60);PV=size(60);MAT=size(60,60);Goo
dValues=zeros(200,200);
%Constraints
%C1
CL(1)=-0.5;CU(1)=0;
%C2
CL(2)=0;CU(2)=4;
%w
CL(3)=-0.5;CU(3)=0;
%ns
CL(4)=20;
CU(4)=70;
%number of Super-swarms %
NS=5; % Pederson (2010) suggested NS=47.  That value of NS was tried but took too long in this problem for
the benefits it provided.
%%
%Initialize the particle's position with a uniformly distributed random vector between the lower and upper
boundaries of the search-space.
for j=1:NS
for i=1:4
MAT(i,j)=CL(i) + (CU(i)-CL(i)).*rand;
end
end
%round the number of swarms to a whole number
for j=1:NS
MAT(4,j)=round(MAT(4,j));
end
c1=MAT(1,1);
c2=MAT(2,1);
w=MAT(3,1);
ns=MAT(4,1);
%Calculate TMD using proposed input parameters
%GVmeta=TMDCalculation(c1,c2,w,ns);
for n=1:NS
c1=MAT(1,n);
c2=MAT(2,n);
w=MAT(3,n);
ns=MAT(4,n);
MAT(5, n )=TMDCalculation(c1,c2,w,ns);
end
%%
%SUPERSWARM
%Initialize the particle's best known position to its initial position.
for n=1:NS
for i = 1 : 5
PI( i,  n) = MAT(i, n );
end
end
% Select first particle as best global value.
% Then compare that particle with other particles to find best global position.
for i=1:4
Global(i,1)=PI(i,1);
end
GVmeta=PI(5,1);
%Test to find best global value
for i=1:NS
if PI(5,i)<GVmeta
for j=1:4
```

```
Global(j,1)=PI(j,i);
end
GVmeta=PI(5,i);
end
end
%Initialize the particle's velocity.
for n=1:NS
for i=1:4
PV(i)=CU(i)-CL(i);
VI( i,  n)=-PV(i) + (2*PV(i)).*rand;
end
end
%Repeat for the specified number of iterations.
while ITE<ITERATIONS
%For each superswarm particle.
for i=1:NS;
%For each dimension / variable.
for d=1:4;
%Select uniform random numbers with which to adjust behavioral parameters.
rp=rand;
rg=rand;
%Update each particle's velocity.
VI( d, i)=WMeta* VI(d,i) + C1Meta *rp* (PI(d,i)-MAT(d,i)) + C2Meta* rg* (Global(d)-MAT(d,i));
end
%Update each particle's position ensuring it is within the specified upper and lower constraints.
for d=1:4;
if MAT(d,i)+VI(d,i)<CL(d)
MAT(d,i)=CL(d);
elseif MAT(d,i)+VI(d,i)>CU(d)
MAT(d,i)=CU(d);
else
MAT(d,i)=MAT(d,i)+VI(d,i);
end
end
%Number of swarms could be input as a fixed integer. Here the algorithm searches for number for the optimum
number of swarms too. Hence, it is necessary to round that number.
MAT(d,i)=round(MAT(d,i));
%%
%SUBSWARM
c1=MAT(1,i);
c2=MAT(2,i);
w=MAT(3,i);
ns=MAT(4,i);
%Calls to wellbore trajectory calculation function (see section 2 of this code listing).
MAT(5, i )=TMDCalculation(c1,c2,w,ns);
%GVmeta=TMDCalculation(c1,c2,w,ns)
%Update best known position of each particle.
if MAT(5,i)<PI(5,i)
for d=1:4
PI(d,i)=MAT(d,i);
end
PI(d+1,i)=MAT(5,i);
end
end
%Update best global value by comparing updated best position(so far) of each particle relative to best global
value recorded (so far).
for i=1:NS
if PI(5,i)<GVmeta
for j=1:4
Global(j,1)=PI(j,i);
end
GVmeta=PI(5,i);
end
end
Global(5,1)=GVmeta;
%Best positions (so far) are saved into the GoodValues matrix
for j=1:5
GoodValues(j,ITE+1)=Global(j,1);
end
%Plot particle movements for each iteration
figure (1)
%C1
subplot(2,3,1)
for i=1:1:NS
plot(ITE,MAT(1,i));
hold on
```

```
xlabel 'Number of Iterations'
ylabel 'Cognitive (C1)'
end
%C2
subplot(2,3,2)
for i=1:1:NS
plot(ITE,MAT(2,i));
hold on
xlabel 'Number of Iterations'
ylabel 'Social (C2)'
end
%W
subplot(2,3,3)
for i=1:1:NS
plot(ITE,MAT(3,i));
hold on
xlabel 'Number of Iterations'
ylabel 'Inertia (W)'
end
%Number of swarms.
subplot(2,3,4)
for i=1:1:NS
plot(ITE,MAT(4,i));
hold on
xlabel 'Number of Iterations'
ylabel 'Number of Swarms (S)'
end
%Good Values
subplot(2,3,5)
for i=1:1:NS
plot(1:ITE,GoodValues(5,1:ITE));
hold on
xlabel 'Number of Iterations'
ylabel 'TMD (ft)'
end
ITE=ITE+1;
end
toc
```

Section 2(subcodes):

```
function [GVmeta]=TMDCalculation(C1,C2,W,NS)
%Input parameters.
ns=NS;iterations=70;c1=C1;c2=C2;w=W;
x = 16;n = 0;GVmeta=0;
mat2=size(60,60);v = size(60,60);mat = size(60, 60);vi=size(60,60);
pv=size(60);G=size(200);
format long
% Repeat calculations three times and then calculate an average.
for COUNTER=1:3
%Constraints
cl(1) = 2500;cl(2) = 10;cl(3) = 40;cl(4) = 90;cl(5) = 270;
cl(6) = 270;cl(7) = 270;cl(8) = 330;cl(9) = 330;cl(10) = 355;
cl(11) = 0;cl(12) = 0;cl(13) = 0;cl(14) = 6000;cl(15) = 10000;
cl(16) = 600;cu(1) = 2500;cu(2) = 20;cu(3) = 70;cu(4) = 95;
cu(5) = 280;cu(6) = 280;cu(7) = 280;cu(8) = 340;cu(9) = 340;
cu(10) = 360;cu(11) = 5;cu(12) = 5;cu(13) = 5;cu(14) = 7000;
cu(15) = 10200;cu(16) = 1000;
while n<ns;
for i = 1 : x
v(i,n+1) = cl(i) + (cu(i)-cl(i)).*rand;
end
rad = (3.14159 / 180);
R1 = 100 / (v(11, n + 1)* rad);
R3 = 100 / (v(12, n + 1)* rad);
R5 = 100 / (v(13, n + 1)* rad);
D1 = R1 * (((v(6, n + 1) - v(5, n + 1)) * rad) ^ 2 * (sin(v(2, n + 1) * rad / 2)) ^ 4 ...
 + ((v(2, n + 1) * rad) ^ 2)) ^ (0.5);
D2 = (v(14, n + 1) - v(16, n + 1) - D1 * (sin(v(2, n + 1) * rad) - sin(0)) / (v(2, n + 1) * rad - 0)) / cos(v(2, n + 1) *
rad);
D3 = R3 * (((v(8, n + 1) - v(7, n + 1)) * rad) ^ 2 * (sin((v(3, n + 1)...
 + v(2, n + 1)) * rad / 2)) ^ 4 + ((v(3, n + 1) - v(2, n + 1)) * rad) ^ 2) ^ (0.5);
D4 = (v(15, n + 1) - v(14, n + 1) - D3 * (sin(v(3, n + 1) * rad) - sin(v(2, n + 1) * rad)) / (v(3, n + 1) * rad - v(2, n +
1) * rad)) / cos(v(3, n + 1) * rad);
D5 = R5 * (((v(10, n + 1) - v(9, n + 1)) * rad) ^ 2 * (sin((v(4, n + 1) + v(3, n + 1)) * rad / 2)) ^ 4 ...
 + ((v(4, n + 1) - v(3, n + 1)) * rad) ^ 2) ^ (0.5);
z = v(1, n + 1) + D1 + D2 + D3 + D4 + D5 + v(16, n + 1);
cas1 = v(16, n + 1) + D1 * (sin(v(2, n + 1) * rad) / (v(2, n + 1) * rad));
```

```
cas2 = cas1 + D2 * cos(v(2, n + 1) * rad) + D3 * ((sin(v(3, n + 1) * rad) - sin(v(2, n + 1) * rad)) / ((v(3, n + 1) - v(2,
n + 1)) * rad));
cas3 = cas2 + D4 * cos(v(3, n + 1) * rad) + D5 * ((sin(v(4, n + 1) * rad) - sin(v(3, n + 1) * rad)) / ((v(4, n + 1) - v(3,
n + 1)) * rad));
phi1=v(2,n+1);
phi2=v(3,n+1);
phi3=v(4,n+1);
Hold1=D2*cos(phi1*rad);
Hold2=D4*cos(phi2*rad) ;
tvdbuild1=(D1*(sin(rad*(phi1))-sin(rad*0)))/(phi1*rad);
tvddrop=D3*(sin(rad*phi2)-sin(rad*phi1))/((phi2-phi1)*rad);
tvdbuild2=D5*(sin(rad*phi3)-sin(rad*phi2))/((phi3-phi2)*rad);
TVD=tvdbuild1+tvdbuild2+tvddrop+Hold1+Hold2+v(16,n+1);
if D1 > 0 && D2 > 0 && D3 > 0 && D4 > 0 && D5 > 0 && cas1 > 1800 &&...
cas1 < 2200 && cas2 > 7200 && cas2 < 8700 && cas3 > 10835 &&...
cas3 < 10885 && TVD>10850 && TVD<10900
for i = 1 : x
mat(i, n+1) = v(i, n + 1);
end
mat(i+1, 1 + n) = z;
n = n + 1;
end
end
%%
for i=1:x
G(i,1)=mat(i,1);
end
GV=mat(17,1);
%Initialize each particle's best known position to its initial position.
pi=size(60,60);
for n=1:ns
for i = 1 : x+1
pi( i,  n) = mat(i, n );
end
end
%Update the swarm's best known position
s=ns;
for i=1:s
if pi(17,i)<GV
for j=1:x
G(j,1)=pi(j,i);
end
GV=pi(17,i);
end
end
%%
%Initialize each particle's velocity with a uniform random number.
for n=1:s
for i=1:x
pv(i)=cu(i)-cl(i);
vi( i,  n)=-pv(i) + (2*pv(i)).*rand;
end
end
%Repeat for the specified number of iterations.
ite=0;
while ite<iterations
%For each particle.
for i=1:s;
%For each dimension.
for d=1:x;
%Select uniform random numbers.
rp=rand;
rg=rand;
%Update each particle's velocity.
vi( d, i)=w* vi(d,i) + c1 *rp* (pi(d,i)-mat(d,i)) + c2* rg* (G(d)-mat(d,i));
end
%Update each particle's position.
for d=1:x;
if mat(d,i)+vi(d,i)<cl(d)
mat2(d,i)=cl(d);
elseif mat(d,i)+vi(d,i)>cu(d)
mat2(d,i)=cu(d);
else
mat2(d,i)=mat(d,i)+vi(d,i);
end
end
```

```
%Evaluate wellbore trajectory components for each new particle.
rad = (3.14159 / 180);
R1 = 100 / (mat2(11, i) * rad);
R3 = 100 / (mat2(12, i) * rad);
R5 = 100 / (mat2(13, i) * rad);
D1 = R1 * (((mat2(6, i) - mat2(5, i)) * rad) ^ 2 * (sin(mat2(2, i) * rad / 2)) ^ 4 + ((mat2(2, i ) * rad) ^ 2)) ^ (0.5);
D2 = (mat2(14, i) - mat2(16, i) - D1 * (sin(mat2(2, i) * rad) - sin(0)) / (mat2(2, i) * rad - 0)) / cos(mat2(2, i) * rad);
D3 = R3 * (((mat2(8, i) - mat2(7, i)) * rad) ^ 2 * (sin((mat2(3, i) + mat2(2, i)) * rad / 2)) ^ 4 + ((mat2(3, i) - mat2(2,
i)) * rad) ^ 2) ^ (0.5);
D4 = (mat2(15, i) - mat2(14, i) - D3 * (sin(mat2(3, i) * rad) - sin(mat2(2, i) * rad)) / (mat2(3, i) * rad - mat2(2, i) *
rad)) / cos(mat2(3, i) * rad);
D5 = R5 * (((mat2(10, i) - mat2(9, i)) * rad) ^ 2 * (sin((mat2(4, i) + mat2(3, i)) * rad / 2)) ^ 4 + ((mat2(4, i) -
mat2(3, i)) * rad) ^ 2) ^ (0.5);
z = mat2(1, i) + D1 + D2 + D3 + D4 + D5 + mat2(16, i);
mat2(17,i)=z;
cas1 = mat2(16, i) + D1 * (sin(mat2(2, i) * rad) / (mat2(2, i) * rad));
cas2 = cas1 + D2 * cos(mat2(2, i) * rad) + D3 * ((sin(mat2(3, i) * rad) - sin(mat2(2, i) * rad)) / ((mat2(3, i) -
mat2(2, i)) * rad));
cas3 = cas2 + D4 * cos(mat2(3, i) * rad) + D5 * ((sin(mat2(4, i) * rad) - sin(mat2(3, i) * rad)) / ((mat2(4, i) -
mat2(3, i)) * rad));
phi1=mat2(2,i);
phi2=mat2(3,i);
phi3=mat2(4,i);
Hold1=D2*cos(phi1*rad);
Hold2=D4*cos(phi2*rad) ;
tvdbuild1=(D1*(sin(rad*(phi1))-sin(rad*0)))/(phi1*rad);
tvddrop=D3*(sin(rad*phi2)-sin(rad*phi1))/((phi2-phi1)*rad);
tvdbuild2=D5*(sin(rad*phi3)-sin(rad*phi2))/((phi3-phi2)*rad);
TVD=tvdbuild1+tvdbuild2+tvddrop+Hold1+Hold2+mat2(16,i);
%If all the following constraints are satisfied transfer mat2 matrix values into matrix mat.
if D1 > 0 && D2 > 0 && D3 > 0 && D4 > 0 &&...
D5 > 0 && cas1 > 1800 && cas1 < 2200 && cas2 > 7200 &&...
cas2 < 8700 && cas3 > 10835 && cas3 < 10885  &&...
TVD > 10850 && TVD < 10900
for d=1:x
mat(d,i)=mat2(d,i);
end
mat(17,i)=mat2(17,i);
%Update each particle's best known position.
if z<pi(17,i)
for d=1:x
pi(d,i)=mat(d,i);
end
pi(17,i)=z;
end
%Update the swarm's best known position.
if z<GV
for d=1:x
G(d)=mat(d,i);
end
GV=z;
end
end
end
ite=ite+1;
end
GVmeta=GV+GVmeta;
end
GVmeta=GVmeta/COUNTER;
end
```

References

Adams, J.N., Charrier, T., 1985. Drilling Engineering: a Complete Well Planning Approach. PennWell Publishing Company, Tulsa, Oklahoma, pp. 342–345.

Ahmadi, M.A., Zendehboudi, S., Lohi, A., Elkamel, A., Chatzis, I., 2013. Application of hybrid genetic algorithm with particle swarm optimization and neural network for reservoir permeability prediction. Geophys. Prospect. 61 (3), 582–598.

Ahmed, K., 2000. A 3-D Optimization Method for Planning Horizontal Wells (M.Sc. thesis). Cairo University.

Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K., 2002. A racing algorithm for configuring metaheuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 11–18.

Craig, J.T., Randall, B.V., March 1976. Directional survey calculations. Pet. Eng., 38–45.

Dalzell, J., 2013. Conventional Natural Gas Supply Costs in Western Canada – an Update. Canadian Energy Research Institute (December) study no. 136, 31 pp.

Eberhart, R.C., Hu, Xiaohui, 1999. Human tremor analysis using particle swarm optimization. In: Evolutionary Computation. CEC 99. Proceedings of the 1999 Congress.

Engelbrecht, A.P., 2005. Fundamentals of Computational Swarm Intelligence. Wiley, West Sussex, England.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. 16, 122–128.

Han, P., Huang, Y., Jia, Z.-Z., Wang, D.-F., Li, Y.-L., 2005. Optimal control for superheated steam temperature system based on PSO optimization. In: Machine Learning and Cybernetics, 2005. Proceedings International Conference, vol. 2, pp. 960–964.

Hashim, A-A., 1995. Simplified PC program calculates modified S-type well trajectory. Oil Gas J. 93 (8), 54–57.

Kashikar, S., 2005. New frontiers in directional drilling. Schlumberger Middle East Asia Reserv. Rev. 6, 25–43. http://www.slb.com/resources/publications/industry_articles/mearr/num6_directional_drilling.aspx.

Keane, A.J., 1995. Genetic algorithm optimization in multi-peak problems: studies in convergence and robustness. Artif. Intell. Eng. 9 (2), 75–83.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948.

Kennedy, J., 1997. The particle swarm: social adaptation of knowledge. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 303–308.

Mercer, R.E., Sampson, J.R., 1978. Adaptive search using a reproductive metaplan. Kybern. Int. J. Syst. Cybern. 7 (3), 215–228.

Meissner, M., Schmuker, M., Schneider, G., 2006. Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. BMC Bioinform. 7 (125), 1–11.

Onwunalu, J.E., 2010. Optimization of Field Development Using Particle Swarm Optimization and New Well Pattern Descriptions (PhD thesis). Stanford University.

Onwunalu, J.E., Durlofsky, L.J., 2010. A New Well Pattern Optimization Procedure for Large-Scale Field Development. Stanford University.

Pedersen, M.E.H., Chipperfield, A.J., 2010. Simplifying particle swarm optimization. Appl. Soft Comput. 10 (2), 618–628.

Pedersen, M.E.H., 2010a. Tuning and Simplifying Heuristical Optimization (PhD thesis). Computational Engineering and Design Group School of Engineering Sciences, University of Southampton, 192 pp.

Pedersen, M.E.H., 2010b. Good Parameters for Particle Swarm Optimization. Hvass Laboratories technical report no. HL1001, pp. 1–12.

Prata, D.M., Schwaab, M., Lima, E.L., Pinto, J.C., 2009. Nonlinear dynamic data reconciliation and parameter estimation through particle swarm optimization: application for an industrial polypropylene reactor. Chem. Eng. Sci. 64 (18), 3953–3967.

Pratt, S., 2004. A fresh angle on oil drilling. Geotimes (March edition). http://www.geotimes.org/mar04/feature_horizdrill.html.

Rui, X., Wunsch II, C., Frank, L., 2007. Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. In: IEEE/ACM Transactions on Computational Biology and Bioinformatics.

Sayyad, H., Manshad, A.K., Rostami, H., 2014. Application of hybrid neural particle swarm optimization algorithm for prediction of minimum miscibility pressure (MMP). Fuel 116, 625–633.

Sharma, P., Khurana, N., 2013. Study of optimal path finding techniques. Int. J. Adv. Technol. 4 (2).

Shi, Y., Eberhart, R.C., 1998a. A modified particle swarm optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73.

Shi, Y., Eberhart, R.C., 1998b. Parameter selection in particle swarm optimization. In: Proceedings of Evolutionary Programming VII (EP98), pp. 591–600.

Shokir, E.M., Emera, M.K., Eid, S.M., Wally, A.W., 2004. A new optimization model for 3-D well design. Emir. J. Eng. Res. 9 (1), 67–74.

Short, J., 1993. Introduction to Directional and Horizontal Drilling. PennWell Books, Tulsa, Oklahoma, pp. 1–222.

Smit, S.K., Eiben, A.E., 2009. Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 399–406.

Suryanarayana, P.V.R., McCann, R.C., Rudolf, R.L., Rupani, R.A., 1998. Mathematical technique improves directional well-path planning. Oil Gas J. 96 (34), 57–63.

Xiao, X., Dow, E.R., Eberhart, R., Miled, Z.B., Oppelt, R.J., 2003. Gene clustering using self-organizing maps and particle swarm optimization. In: Parallel and Distributed Processing Symposium, 2003. Proceedings. International, 10 pp.

Xiushan, L., Ziahong, S., Sen, F., 1997. Natural parameter method accurately calculates well bore trajectory. Oil Gas J. 95 (4), 90.