Drillstring Dynamics (DSD)
Torque and Drag Module
User’s manual

Contents

1

Introduction

1.1 What is DSD?

DSD is an application for drillstring statics and dynamics. DSD simulates the interaction of the drill-string with the borehole under a variety of conditions.

DSD can place the drillstring in a given borehole, and calculate its equilibrium configuration (deflections, internal forces, and contact forces with the borehole).

DSD can assemble the drillstring from components and perform the analysis of the tripping in and tripping out analysis of the drag forces and drag torques.

DSD can report results in graphical form and as spreadsheets to allow for further postprocessing. A variety of outputs can be requested: plots of internal forces, plots of contact forces, deflection curves, torque and drag graphs.

1.2 Methods and Software

DSD is based on the finite element method. The drillstring is represented by a fully nonlinear finite element model consisting of very accurate corotational beams that allow for arbitrary motions and large rotations. The interaction of the drillstring and the borehole is represented by a variety of specialized contact models.

The finite element model is implemented in Matlab. The general machinery of finite element (FE) calculations and the finite element formulation of geometrically nonlinear 3-D beams is not part of this manual. The formulation of the mechanics of drillstrings interacting with a wellbore under a variety of loads is implemented in the program DSD, which built upon the foundation of the above FE methods. The present document is the user's manual for the operation of the program DSD.

1.3 Building process and targets

The DSD understands the process of simulating the various stages in the mechanical deformation of the drillstring as a sequence of targets. Under the term ***target*** we understand here a well-defined computational result. The target may be a *simulation database* representing a particular stage (for example `torque_and_drag`) or a *visualization* of the results of a particular stage (for example `show_deflection_curve`).

The target T we are interested in could depend on another target. This information gives DSD the intelligence to build all intermediate targets needed to obtain T, and only those targets. For instance, let us say we are interested in the target `show_drag_forces`, and let us assume that we are starting from scratch, i.e. no simulation results exist yet. DSD will be started to deliver the target `show_drag_forces`, and it will realize that this target depends on the target `torque_and_drag` which at this point has not been built yet. Therefore DSD will start automatically to build the target torque_and_drag, which, as DSD promptly finds out, doesn't exist either and DSD consequently starts to build this target. The target torque_and_drag in its turn depends on the target setup, and since this target doesn't exist either, DSD starts to build setup. The target setup does not exist, but it does not depend on any other target and DSD can proceed to build it. Once setup is built, DSD can backtrack to build the target torque_and_drag, which once it is available allows DSD to build target show_drag_forces.

Importantly, this approach allows the DSD program to build or re-build only the targets that are actually needed and that are either missing or are out of date. The following rules are used to decide whether a target needs to be rebuilt:

- If a target database exists, and if its timestamp is fresher than the current timestamp in the build context, and if the target data has not changed, then the target is deemed not to be in need of rebuilding.
- If a target database does not exist at this point, or if the target database exists and if the current target data does not match the target data in the database, the target needs to be rebuilt.

As an example, consider the following situation. DSD needs to build the target torque_and_drag. It checks the target selfweight, which in turn depends on the target setup. Let us assume that the target setup exists and for some reason was in the meantime rebuilt so that its timestamp is fresher than the timestamp of the target selfweight. Therefore DSD first rebuilds the target selfweight, and only then starts building the target torque_and_drag.

A similar sequence of events would be triggered if we changed the target data of the target setup: When the dependency of the target selfweight is checked, DSD looks at the target setup and sees that the target data specified for this target is changed with respect to the target data for which the target setup was originally built. Therefore the target setup is rebuilt, and consequently the target selfweight is also rebuilt as the target on which it depends (setup) has changed. Finally the target torque_and_drag can be built.

1.4 Target data and the INI file

The targets are described to the DSD program in the so-called INI file. The control parameters are referred to as target data. An example of the INI file is shown here:

```
[setup]
cased_length_fraction = 3150/4000
cased_radius = (9+5/8)/2 [in]
drillstring_file = Well_1.csv
max_element_length = 90 [ft]
model_name = T&D Verification 1 ex
survey_listing = Survey_Listing_Well_1.csv
uncased_radius = 0.26/2 [m]

[selfweight]
mass_density_of_drilling_fluid = 1380 [KG/M^3]
WOB = 0 [lbf]

[torque_and_drag]
drilling_friction_coefficient = 0
mass_density_of_drilling_fluid = 1380 [KG/M^3]
wall_friction_coefficient_cased = 0.24
wall_friction_coefficient_uncased = 0.24

[show_drag_forces]
```

```
force_units = 1000*kg*g
image_name = torque_and_drag_drag_force.jpg
length_units = m
target = torque_and_drag
travelling_block_mass = 40000[kg]
```

The INI file consists of blocks, one block per target. The targets are named in the square brackets, such as the target `setup` in the first line [`setup`]. Underneath the line that names a target the target data is listed in arbitrary order. For instance the line

```
cased_radius = (9+5/8)/2 [in]
```

supplies value $(9+5/8)/2 \approx 4.8125$ to the target data `cased_radius` in the measurement units of inches.

Note that a target to be built does not necessarily need to be described in the INI file: if the defaults for the target data meet the user's needs, there's no need to include the target in the file.

1.5 Model folder

All the files that describe the state of any of the targets for a particular model are stored in the model folder. The model folder is a subfolder of the `output_folder` as specified for the `setup` target.

The model folder name is the parameter `model_name` of the `setup` target. Only model names with characters `a-z`, `A-Z`, underscore (`_`), `0-9`, space ( ), period (`.`), hyphen (`-`), plus (`+`), equal sign (`=`), and comma (`,`) are admissible.

Here is an example of model folders `Example1` and `Example2`. They are subfolders of the output folder `output_folder` specified as `C:\Users\TheUser\out`.

```
C:\Users\TheUser\out
                 |-- Example1
                 |-- Example2
```

1.6 Measurement units

The following list includes all the predefined measurement unit symbols that can be used to supply measurement units to the `DSD` program. The capitalization of the letters of the symbol does not matter, so that the following are equivalent `mm`, `Mm`, `MM`.

Time measurement units:

```
SEC, S = second
MIN = Minute
HR = hour
DAY = Day
WK = Week
MONTH = Month
YR = Year
```

Measurements of angles:

```
RAD=Radian (angle)
DEG=Degree (angle)
REV=Measure of angles in terms of revolutions
```

Length measurement units:

```
M=Meter
IN=Inch
FT=Foot
NMI=Nautical mile
CM=Centimeter
YD=Yard
MM=Millimeter
MILE=Mile
```

Temperature measurement units:

```
K= Degrees Kelvin
RAN= Degrees Rankine
```

Mass measurement units:

```
SLUG=Slug
KG=Kilogram
GM=Gram
```

Force measurement units:

```
N=Newton
OZ=ounce
LBF= Pound of force
KIPS = Thousands of pounds force
```

Measurement units of speed:

```
KT=knot
MPH=mile per hour
```

Measurement units of stress and pressure:

```
PSI=Pounds per square inch
KPSI=Thousands of pounds per square inch
Pa= Pascal
MPa= Millions of Pascal (mega Pascal)
BAR=Bar
ATM=Atmosphere
TORR=torr
mmHG=mm Hg
BA=CGS unit of pressure, dyne/cm^2
```

Measurement units of work and power:

```
J= Joule
CAL = Calorie
MEV = Mega electron volt
ERG = erg
BTU = BTU
```

```
W=watt
MW = Megawatt
HP=Horsepower
```

Measurement units of electrical quantities:

```
COUL=Coulomb (charge);
A = ampere
V = volt
```

Measurement units of frequency and revolutions:

```
HZ = Frequency (Hertz)
RPS = Revolutions per second
RPM =  Revolutions per minute
```

Measurement units of volume:

```
L = Liter
GAL = Gallon
GPM = Gallons per minute
```

Quantitative modifiers:

```
NANO=10^(-9)
MICRO=10^(-6);
MILLI=10^(-3);
KILO=10^3;
MEGA=10^6;
GIGA=10^9;
TERA=10^12;
```

Predefined constants:

```
G=Gravity acceleration
```

Expressions may be formed from the predefined measurement unit symbols using common arithmetic operators. In the INI files the units are always supplied in square brackets. Therefore the following shows a few well-formed measurement unit specifications in square brackets.

```
[ft]
[m/s]
[lbf]
[KG/M^3]
[Kilo*N*m]
[lbf*ft]
```

1.7 How to run DSD

DSD is a compiled application which needs to be run from the command line. The first argument is the name of the INI file, the second argument is the name of the target to build. As an example, this line will invoke DSD to build the target `selfweight`, with the description of the targets in the file `Example_1.ini`.

```
> DSD.exe Example_1.ini selfweight
```

The executable returns the status of the calculation as either of two values: zero (0) for failure, and one (1) for success.

If the last argument (name of the target) is omitted, all targets described in the INI file are built. As an example, this line will invoke `DSD` to build the targets `setup`, `selfweight`, and `show_deflection_curve` whose description was included in the file `Example_1.ini`.

```
> DSD.exe Example_1.ini
```

1.7.1 Logs

`DSD` writes a log in the working folder. Below is an example of the log file written out in the working folder for a build of the target `setup`, followed by the build of the target `show_borehole_profile`.

```
--- 2013-10-11@14:55:54.342
  DSD_build: started for target setup
--- 2013-10-11@14:55:54.345
  DSD_build_target: working on setup
--- 2013-10-11@14:56:33.421
  DSD_build_target: success for setup
--- 2013-10-11@14:56:33.423
  DSD_build: succeeded for setup

--- 2013-10-11@14:56:33.435
  DSD_build: started for target show_borehole_profile
--- 2013-10-11@14:56:33.437
  DSD_build_target: working on show_borehole_profile
--- 2013-10-11@14:56:33.464
  DSD_build_target: working on setup
--- 2013-10-11@14:56:33.633
  DSD_build_target: success for setup
--- 2013-10-11@14:56:35.005
  DSD_build_target: success for show_borehole_profile
--- 2013-10-11@14:56:35.008
  DSD_build: succeeded for show_borehole_profile
```

`DSD` also writes a log file in the model folder. This log file has a more detailed information, and only for the particular model in question. Here is an example of the model log corresponding to the `DSD` log above:

```
2013-10-11@14:55:54.42
  Model initialized
2013-10-11@14:55:54.423
  Reading borehole profile
  SurveyListing_sample_model_3.csv
2013-10-11@14:55:54.478
  Done
2013-10-11@14:55:54.483
  Making drillstring
  DrillstringModel1.csv
2013-10-11@14:55:54.643
  Done
```

```
2013-10-11@14:55:54.647
  Making wellbore
2013-10-11@14:55:54.805
  Done
2013-10-11@14:55:54.809
  Making finite element models
2013-10-11@14:55:54.892
  Done
2013-10-11@14:55:54.895
  Positioning drillstring in the wellbore
2013-10-11@14:56:33.296
  Done
2013-10-11@14:56:33.698
  Showing borehole profile
2013-10-11@14:56:34.203
  Done
2013-10-11@14:56:34.208
  Saving image
  ./out\Example_1\borehole_profile.jpg
```

Here is an example of an INI file with a mistake introduced on purpose in its last line: the specification for the measurement units uses an invalid symbol of `feet` (the correct symbol is `ft`).

```
[setup]
model_name = Example_1_w_mistake
bit_distance_from_top = 2000 [ft]
cased_length_fraction = 0
cased_radius = 0.15557+0.007 [m]
drillstring_file = DrillstringModel1.csv
input_folder = ./in
output_folder = ./out
survey_listing = SurveyListing_sample_model_3.csv
uncased_radius = 0.15557+0.007 [m]
max_element_length = 90 [feet]
```

When DSD is run, it fails (returns the status of 0). The log in the working folder then can be inspected:

```
--- 2013-10-11@15:46:39.333
  DSD_build: started for target setup
--- 2013-10-11@15:46:39.343
  DSD_build_target: working on setup
--- 2013-10-11@15:46:39.568

  Invalid units specification: feet
--- 2013-10-11@15:46:39.571
  DSD_build_target: failure for setup
--- 2013-10-11@15:46:39.573
  DSD_build: failed for setup
```

and the offending line can be tracked in the INI file.

1.7.2 Progress file

DSD also writes a progress file in the model folder. For each simulation that takes more than a few seconds this file records the name of the operation currently executing, and the progress in percent. The name of the file is composed of the name of the model and the extension .pgs. For example, here are the contents of the progress file for an executing selfweight target which is 49% finished:

```
Selfweight
49
```

2

Simulation targets

2.1 Target `setup`

This target establishes initial static equilibrium of the drill string inserted into the borehole. This is the first target to build for any simulation.

2.1.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[setup]
bit_distance_from_top = [] [ft]
cased_length_fraction = 0
cased_radius = [] [in]
drillstring_file = drillstring.CSV
input_folder = ./in
max_element_length = 45 [ft]
model_name = No-name
num_components_to_include= []
output_folder = ./out
remark =
survey_listing = survey_listing.CSV
uncased_radius = [] [in]
```

2.1.2 Description of target data

- `bit_distance_from_top`
  The bit distance from top (i.e. the measured depth of the bit) can be supplied as empty ([]), in which case the distance from the topis implied which would put the top of the drill string at the surface.
- `cased_length_fraction`
  Fraction of the length of the borehole that is cased, non-dimensional. The value must be between zero and one, inclusive.
- `cased_radius`
  Radius of the cased part of the borehole. If the `cased_length_fraction` is supplied as greater than zero (i. e. if part of the whole is actually cased), the `cased_radius` parameter must be supplied.

- `drillstring_file`
  Comma-Separated Value (CSV) spreadsheet that describes the drillstring composition. All dimensions (radii and arc lengths) are assumed to be in meters. The mass density is assumed to be in kilograms per meter cubed. The Young's (elasticity) modulus is assumed to be in Pascals. Here is an example of a 10-component drillstring, each component consisting of a single cylinder. Note that the optional outer radius of the tool joints is specified.

Component	Cylinder	Youngs	Poisson	Mass Dens	Eff External Radius	Eff Internal Radius	Arclength	TJOR
Tri-Cone Bit	Cylinder 1	2.05E+11	0.3	7800	0.108	0	0.3	
Drill Collar	Cylinder 2	2.05E+11	0.3	7800	0.07	0.029070552	40	
Heavy Weight	Cylinder 3	2.05E+11	0.3	7800	0.0635	0.031500622	150	0.08255
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	200	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	200	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	200	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	400	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	1000	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	1000	0.0889
Drill Pipe	Cylinder 4	2.05E+11	0.3	7800	0.0635	0.051315772	809.7	0.0889

  Here the abbreviations in the first line of the file stand for: `Comp.`= component, `Cyl.`= cylinder, `Youngs`= Young's modulus, `Poiss.`= Poisson ratio, `Mass D.`= mass density, `Eff. Ext. Rad.`= effective external radius, `Eff. Int. Rad.`= effective internal radius, `Arclength`= arc length, `TJOR` = tool joint outer radius.
  Note that the value `TJOR` (the tool joint outer radius) is optional. If it is not supplied, the effective external radius is considered instead of the tool joint radius. The radius is used in any calculation that involves contact with the borehole surface. Using the correct radius is crucial especially in calculations of the drag torques. If the tool joint radius is known (and different from the effective external radius), it should be supplied as input.
  Here is an example of a 8-component drillstring, each component consisting of one or more cylinders. Note that the elastic properties may be specified for each cylinder or only for each component.

Component	Cylinder	Youngs	Poisson	Mass Density	Eff External Radius	Eff Internal Radius	Arclength
Component 1	Cylinder 1	2.05E+11	0.3	7850	0.13018	0	0.0508
	Cylinder 2				0.15557	0	0.2794
	Cylinder 3				0.10477	0	0.127
Component 2	Cylinder 4	2.05E+11	0.3	7850	0.10477	0.0381	0.1524
	Cylinder 5				0.15557	0.0381	0.6096
	Cylinder 6				0.10477	0.0381	0.1524
Component 3	Cylinder 7	2.05E+11	0.3	7850	0.10477	0.0381	1.524
Component 4	Cylinder 8	2.05E+11	0.3	7850	0.10477	0.066421	8.5344
Component 5	Cylinder 9	2.05E+11	0.3	7850	0.10477	0.03937	0.3048
	Cylinder 10				0.15319	0.03937	1.2192
	Cylinder 11				0.10477	0.03937	0.3048
Component 6	Cylinder 12	2.05E+11	0.3	7850	0.10477	0.065659	6.7056
Component 7	Cylinder 13	2.05E+11	0.3	7850	0.10477	0.0381	0.6096
Component 8	Cylinder 14	2.05E+11	0.3	7850	0.10351	0.03556	0.6096
	Cylinder 15				0.15557	0.03556	1.8288
	Cylinder 16				0.10351	0.03556	0.6096

- `input_folder` Name of the folder that holds the input files.

- `max_element_length` Maximum allowed length of a finite element in the mesh of the drillstring tubulars. Except for very curved boreholes the default length should be acceptable for good accuracy. The user may verify the accuracy of the simulations by reducing this parameter, rerunning the simulations, and comparing the results. Alternatively, this convergence behavior may be studied by increasing `max_element_length` by some factor and comparing the results for the two different element lengths.
- `model_name` Name of the model.
- `num_components_to_include`
  Number of components to include in the model of the drill string from the file. If supplied as empty (`[]`), taking all components specified in the file to be part of the drill string is implied. Otherwise, components 1 (the bit), 2, ..., up to `num_components_to_include` are taken from the drillbit file to comprise the drillstring for the simulation.
- `output_folder`
  Name of the folder to hold the output folder for the model.
- `remark`
  A one line comment to describe the target (optional).
- `survey_listing`
  The data are read from the survey listing file with the following structure:
  The SurveyListing.csv file
  A survey listing file is a 3-column, comma-delimited text file which specifies the length and orientation of the borehole within which the BHA will be placed and analyzed. Survey listings can be in one for three formats; Cartesian, Polar and single-point survey. Both the format and units are automatically detected. Units of length can in either feet (specified by the abbreviation ft) or meters (specified by the abbreviation m). The units of angular measure are assumed to be in degrees. A description of each of the file formats are as follows:
  - Cartesian. In this format, the three columns represent True Vertical Depth (TVD), N(+)/S and E(+)/W. The TVD corresponds to the negative z-axis, the N(+)/S to the +y/-y axis and the E(+)/W to the +x/-x. An example of a file in Cartesian format is shown below.

```
TVD,N(+)/S,E(+)/W
ft,ft,ft
0.00,0.00,0.00
0.00,2788.71,0.00
8.20,2962.55,0.00
```

  - Polar. In this format, the three columns represent Measured Depth (MD), Inclination and Azimuth. Inclination is defined as; 0 degrees is vertical (downward pointing) and 90 degrees is horizontal. An angle greater than 90 degrees coincides with the term "drilling up". Azimuth is defined as; measuring clockwise 0 for a heading of North, 90 for a heading of East, 180 for a heading of South and 270 for a heading of West. An example of a file in polar format is shown below.

```
MD,INC,AZ
ft,deg,deg
0.00,0.00,0.00
266,0.14,187.83
327,0.31,257.33
```

  - Single-point survey. This is a condensed version of the survey file consisting of just one line with five entries. The first three entries specify the location of the bit using measured depth, inclination and azimuth. The next two entries specify constant build and walk rates between the between the bit and the top of the BHA. The wellbore profile along the length of the BHA is calculated by DSD. A sample entry is shown below.

```
MD,INC,AZ,Build Rate,Walk Rate
ft,deg,deg,deg/100 ft,deg/100 ft
16500,19.9,202.42,0.1,-0.76
```

- `uncased_radius` Radius of the open (uncased) hole. The `uncased_radius` parameter must always be supplied.

2.1.3 Procedure

The model is first initialized.

Initialize the model with basic quantities: system of measurement units, and input and output folder. The input folder must exist and be accessible. If the output folder does not exist it is created.

The borehole profile is read and initialized.

The list of control points is refined internally to maintain the shape of the curve. The transitions between segments are first refined with tangency-enforcing points, and the segments of strong curvature have then additional points inserted.

The drill string is read and initialized.

The drillstring is localized by placing the bit at the indicated distance from the surface (measured depth). Note that the length of the borehole profile must be longer than the total length of the drill string.

The wellbore data structure is created.

The finite element method machine (FEMM) data structures are created.

The FEMMs represent the mechanical response of the drillstring and the interaction of the drill string and the wellbore.

The model of the drill string is positioned in the borehole.

The drill string finite element model is positioned within the confines of the borehole surface by nonlinear iterations of equilibrium. This means that the drill string is placed in equilibrium inside an arbitrarily curved borehole. The bending of the drill string and the contact of the drill string with the borehole surface are correctly represented: in general the drill string is not going to be stress-free in a curved borehole, and the contact forces between the drill string and the borehole are not going to be zero.

2.1.4 Example of specification

The following section describes a `setup` target for a partially cased hole.

```
[setup]
model_name = T+D Verification 2
input_folder = ./in
output_folder = ./out
drillstring_file = Well_2.csv
```

```
survey_listing = Survey_Listing_Well_2.csv
cased_length_fraction = 3150/4290
cased_radius = 0.22 [m]
uncased_radius = 0.28 [m]
max_element_length = 90 [ft]
```

2.1.5 Target depends on

The targets on which the present target depends:

- None.

2.1.6 Dependent targets

The targets that depend on this target:

- `selfweight`

2.1.7 Visualization targets

The applicable visualization targets:

- `show_borehole_profile`,
- `show_deflection_curve`,
- `show_drillstring_wellbore`,
- `show_internal_force`,
- `show_shape`,
- `show_shape_fly_through`.

2.2 Target `selfweight`

This target establishes static equilibrium of the drill string under self-weight and drilling loads.

2.2.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[selfweight]
mass_density_of_drilling_fluid = 1500 [KG/M^3]
remark =
WOB = 0 [lbf]
```

2.2.2 Description of target data

- `mass_density_of_drilling_fluid` Mass density of the drilling fluid.
- `remark`
  A one line comment to describe the target (optional).
- `WOB` Weight-on-bit (WOB) force.

2.2.3 Procedure

Equilibrium

The equilibrium of the drill string is obtained by dynamic relaxation. The drill string is allowed to deform dynamically and the kinetic energy is filtered out by numerical dissipation of the Newmark time integration algorithm and by ad hoc mass-and stiffness-proportional Rayleigh damping. The nonlinear incremental problem in each time step is solved by iteration. In addition, the hook force is updated at selected time instants in order to obtain the desired weight on bit. The hook force is calculated with tolerance

`Hook_force_tol=max([1e-3*WOB,1e-5*Initial_guess_of_Hook_force]).`

Here `WOB` is the weight on bit, and `Initial_guess_of_Hook_force` is the initial estimate of the hook force. The contact with the borehole is in this computation considered to be frictionless.

Boundary conditions

The boundary conditions are as follows: At the kelly we apply spring restraints against lateral displacements, effectively enforcing a pinned condition at the rotary table, and all rotations are also penalized by spring constants. At the bit we apply axial restraint to mimic rock-bit contact. Finally, at the kelly a hook force is applied in the direction tangential to the borehole curve of a magnitude that will produce weight on bit (WOB) as given in the target data. Since the hook force to produce the desired WOB is unknown in general, we need to calculate it iteratively from the condition of equilibrium.

2.2.4 Example of specification

This target specification is for a bit rotating off-bottom (WOB is zero).

```
[selfweight]
mass_density_of_drilling_fluid = 1380 [KG/M^3]
WOB = 0 [lbf]
```

2.2.5 Target depends on

The targets on which the present target depends:

- `setup`.

2.2.6 Dependent targets

The targets that depend on this target:

- `modal`.

2.2.7 Visualization targets

The applicable visualization targets:

- `show_deflection_curve`,
- `show_internal_force`,
- `show_shape`,
- `show_shape_fly_through`.

2.3 Target torque_and_drag

This target solves the torque-and-drag problem.

2.3.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[torque_and_drag]
axial_speed = 0 [ft/min]
drilling_friction_coefficient=0
mass_density_of_drilling_fluid = 1500 [KG/M^3]
num_components= []
remark =
rotation_speed = 0 [RPM]
wall_friction_coefficient_cased= 0.2
wall_friction_coefficient_uncased= 0.2
```

2.3.2 Description of target data

- `axial_speed` Axial speed of drillstring motion. Must be nonnegative. Used only in conjunction with nonzero `rotation_speed`. Positive axial speed reduces drag torque. The larger the axial speed, the smaller the drag torque.
- `drilling_friction_coefficient` The dimensionless coefficient to calculate the drilling torque from the weight on bit. The drilling torque is produced as a product of the WOB, the radius of the bit, and this coefficient.

  ```
  Frictional Torque = drilling_friction_coefficient * WOB * (2/3) * bit_radius
  ```

- `mass_density_of_drilling_fluid` Mass density of the drilling fluid.
- `num_components` List of the numbers of components to use. If supplied as empty ([]), constructing the drillstring successively out of all its components is implied. Let us say the drillstring has been defined to have six components. The torque-and-drag simulation will successively considered the drillstring consisting of one, two, three, four, five, and six components.
- `remark` A one line comment to describe the target (optional).
- `rotation_speed` Rotation speed of drillstring motion. Must be nonnegative. Used only in conjunction with nonzero `axial_speed`. Positive rotation speed reduces axial drag force. The larger the rotation speed, the smaller the drag axial drag force.
- `wall_friction_coefficient_cased` Friction coefficient for the cased part of the borehole.
- `wall_friction_coefficient_uncased` Friction coefficient for the uncased (open) part of the borehole.
- `WOB` Weight on bit.

2.3.3 Procedure

The static equilibrium of the drill string under the given set of self-weight, WOB, and hook force is found. The result is a set of normal contact forces between the drill string and the borehole. These forces are converted to friction forces, which when added up in a vectorial fashion will allow for the determination of three hook forces: tripping out, tripping in, and static. The friction forces are also converted to friction torques which are added up to estimate the torque at the rotary table needed to turn the drill string against friction.

These hook forces are calculated for a variable composition of the drill string given by the `num_components` target data. The drill string is repeatedly built up of the desired number of components, equilibrium is iterated, and the hook forces are calculated.

Boundary conditions

The boundary conditions are as follows: At the kelly we apply spring restraints against lateral displacements and all rotations are also penalized by spring constants. At the bit we apply axial restraint to mimic rock-bit contact. Finally, at the top a hook force is applied in the direction tangential to the borehole curve of a magnitude that will produce weight on bit (WOB) as given in the target data. Since the hook force is unknown initially, we need to calculate it by iteration from the condition of equilibrium.

The function will calculate the friction forces due to the interaction of the drill bit with the borehole walls. If WOB=0, the hook force balances out the drill string without friction. The pickup weight would be this value of the hook force plus all the friction forces between the drillstring and the wall.

2.3.4 Example of specification

This target specification is for a torque-and-drag analysis for zero weight on bit, considering 2, 6, 10, 14, and 18 components.

```
[torque_and_drag]
  mass_density_of_drilling_fluid = 1500 [KG/M^3]
  num_components= 2:4:18
  wall_friction_coefficient_cased= 0.2
  wall_friction_coefficient_uncased= 0.2
```

2.3.5 Target depends on

The targets on which the present target depends:

- `setup`.

2.3.6 Dependent targets

The targets that depend on this target:

- None.

2.3.7 Visualization targets

The applicable visualization targets:

- `show_drag_torques`,
- `show_drag_forces`.

3

Visualization targets

3.1 Target show_borehole_profile

3.1.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_borehole_profile]
image_dpi = 96
image_height = 936
image_name = borehole_profile.jpg
image_width = 615
interactive = false
length_units = ft
remark =
```

3.1.2 Description of target data

- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).
- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.
- `length_units` Measurement units for the horizontal axis.
- `remark` A one line comment to describe the target (optional).

3.1.3 Procedure

The curve representing the borehole profile is plotted in 3-D Cartesian coordinates. The control points are shown as markers.

Note that the interactive plot can be manipulated using the 3-D visualization tool described in the appendix.

3.1.4 Example of specification

```
[show_borehole_profile]
image_name = borehole_profile.jpg
length_units = ft
remark = Note the two straight sections: the vertical and the curve
```

3.1.5 Target depends on

The targets on which the present target depends:

- `setup`.

3.1.6 Dependent targets

The targets that depend on this target:

- None.

3.2 Target show_internal_force

3.2.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_internal_force]
image_dpi = 96
image_height = 936
image_name = internal_force.jpg
image_width = 615
interactive = false
length_units = ft
remark =
resultant = N
resultant_units = lbf
style = r.-
target = selfweight
xlim = 0,1
```

3.2.2 Description of target data

- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).
- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.

- `length_units` Measurement units for the horizontal axis.
- `remark` A one line comment to describe the target (optional).
- `resultant` A list of codes of the resultants to plot. The individual resultants are `N` (axial force), `S2` (shear force along the local x2 direction), `S3` (shear force along the local x3 direction), `M1` (torsional moment), `M2` (bending moment about the local x2 axis), `M3` (bending moment about the local x3 axis), `Mmax` (maximum bending moment). The input `resultant` can be a combination of the symbols above. For instance, `resultant =N,S2,S3` or `resultant=M2,M3,Mmax`.
- `resultant_units` units in which to present the resultant(s). Note that resultants of different meanings (such as forces with torques) should not be mixed together in a single graph.
- `style` List of style strings to be used for the colors and markers of the locations on the list `distance_from_bit`.
- `target` Name of target for which the internal force should be displayed. The default is `selfweight`.
- `xlim` This is a pair of numbers between zero and one, which is the normalized interval on the mode number axis to be shown in the plot.

3.2.3 Procedure

The curves representing the internal force resultants are plotted in a single graph. Resultants of different meanings, such as torques and forces, should not be mixed together in a single graph.

3.2.4 Example of specification

Display the internal axial force in the drillstring due to the deformation of the drillstring under self-weight loads.

```
[show_internal_force]
image_name = selfweight_internal_force_Axial.jpg
length_units = ft
resultant = N
resultant_units = kilo*lbf
target = selfweight
style = k-. k-- r.-
```

3.2.5 Target depends on

The targets on which the present target depends:

- `setup`, `selfweight`.

3.2.6 Dependent targets

The targets that depend on this target:

- None.

3.3 Target `show_contact_forces`

Produce a plot of the contact forces between the drill string and the borehole.

3.3.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_contact_forces]
force_scale=100 [ft/lbf]
image_dpi = 96
image_height = 936
image_name = shape.jpg
image_width = 615
interactive = false
length_units = ft
remark =
target = selfweight
```

3.3.2 Description of target data

- `force_scale` Numerical factor to scale the length of forces compared to the dimensions of the drill string.
- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).
- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.
- `length_units` Measurement units for the lengths. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `target` Name of the target for which the contact forces should be calculated and displayed. Any target that computes the deformed shape of the drill string is allowed (i.e. targets `setup`, `selfweight`).
- `remark` A one line comment to describe the target (optional).

3.3.3 Procedure

The plot shows the contact forces between the drill string and the borehole. These are only the forces of the frictionless normal contact.

The forces are calculated and displayed for the target specified as input. Any target that computes the deformed shape of the drill string is allowed.

Note that the interactive plot can be manipulated using the 3-D visualization tool described in the appendix.

3.3.4 Example of specification

The following specification is for a plot of the contact forces that maintain the drill string within the borehole, without any loads, due purely to the curvature of the borehole (i. e. for the target `setup`).

```
[show_contact_forces]
force_scale=1000 [ft/lbf]
image_name = setup_shape.jpg
target = setup
```

3.3.5 Target depends on

The targets on which the present target depends:

- Any target that computes the deformed shape of the drill string.

3.3.6 Dependent targets

The targets that depend on this target:

- None.

3.4 Target show_deflection_curve

Produce a plot of the schematic deflection curve of the drillstring.

3.4.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_deflection_curve]
deflection_units = in
image_dpi = 96
image_height = 936
image_name = deflection_curve.jpg
image_width = 615
interactive = false
length_units = ft
remark =
target = setup
xlim = 0,1
```

3.4.2 Description of target data

- `deflection_units` Measurement units for the deflection. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).

- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.
- `length_units` Measurement units for the lengths. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `xlim` This is a pair of numbers between zero and one, which is the normalized interval on the mode number axis to be shown in the plot.
- `target` Name of the target for which deflection curve should be plotted. Any target which computes the deformed shape of the drillstring can be specified.
- `remark` A one line comment to describe the target (optional).

3.4.3 Procedure

The plot shows the deflection curve of the drillstring. The deflection is the lateral displacement of the drill string in the plane fitted to the borehole curve which includes the largest lateral displacement of the drill string. Strictly speaking the curve is going to be a faithful representation of the deformation of the drill string only for loads that result in a planar deformation of the string. If the drill string deforms into a spatial curve, the present plot is only a crude approximation.

In addition to an image of the graph, the data is also saved as an Excel spreadsheet in the CSV format.

3.4.4 Example of specification

The following specification is for a plot of the deflection curve of the drillstring under self-weight loads.

```
[show_deflection_curve]
deflection_units = in
image_name = deflection_curve.jpg
length_units = ft
target = selfweight
```

3.4.5 Target depends on

The targets on which the present target depends:

- Any target that computes the deformed shape of the drillstring.

3.4.6 Dependent targets

The targets that depend on this target:

- None.

3.5 Target `show_drag_forces`

3.5.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_drag_forces]
force_units =lbf
image_dpi = 96
image_height = 936
image_name = drag_forces.jpg
image_width = 615
interactive = false
length_units = ft
remark =
travelling_block_mass = 0 [KG]
```

3.5.2 Description of target data

- `force_units` Units in which to express the hook force. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).
- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.
- `length_units` Measurement units for the measured-depth axis. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `remark` A one line comment to describe the target (optional).
- `travelling_block_mass` Mass of the traveling block.

3.5.3 Procedure

The plot shows the hook force for the three modes of operation of the drillstring: rotating off-bottom, pick up (tripping out), and drop-in (tripping in), for successively increasing numbers of components of the drillstring and hence for successively increasing measured depth.

In addition to an image of the graph, the data is also saved as an Excel spreadsheet in the CSV format.

3.5.4 Example of specification

The following specification is for the plot of the hook force expressed in tons of force (`1000*kg*g`), and the measured depth expressed in meters. The traveling block mass is 40 tons.

```
[show_drag_forces]
  force_units =1000*kg*g
  image_name = drag_forces.jpg
  length_units = m
  travelling_block_mass = 40 [1000*KG]
```

3.5.5 Target depends on

The targets on which the present target depends:

- `torque_and_drag`.

3.5.6 Dependent targets

The targets that depend on this target:

- None.

3.6 Target show_drag_torques

3.6.1 List of target data

The target is defined by data in the INI file as shown below (default values including the applicable measurement units are indicated):

```
[show_drag_torques]
torque_units =lbf*ft
image_dpi = 96
image_height = 936
image_name = drag_torques.jpg
image_width = 615
interactive = false
length_units = ft
remark =
```

3.6.2 Description of target data

- `torque_units` Units in which to express the torque. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `image_dpi` Dots-per-inch resolution of the image.
- `image_height` Image height in paper units (points).
- `image_name` Image name (with or without .jpg as extension). The image is saved in the model folder.
- `image_width` Image width in paper units (points).
- `interactive` Should the figure be displayed interactively? True or false. If interactive==true, the plots produced by the target are displayed on the screen and are available for interactive manipulation by the user. Otherwise the plots are produced automatically and stored in the model folder.
- `length_units` Measurement units for the measured-depth axis. Note that the units are not enclosed in square brackets as is required when supplying numerical values to target data.
- `remark` A one line comment to describe the target (optional).

3.6.3 Procedure

The plot shows the torque required to rotate the drill string off-bottom for successively increasing numbers of components of the drillstring and hence for successively increasing measured depth.

In addition to an image of the graph, the data is also saved as an Excel spreadsheet in the CSV format.

3.6.4 Example of specification

The following specification is for the plot of the torque expressed in N*m, and the measured depth expressed in meters.

```
[show_drag_torques]
  torque_units =N*m
  image_name = drag_torques.jpg
  length_units = m
```

3.6.5 Target depends on

The targets on which the present target depends:

- `torque_and_drag`.

3.6.6 Dependent targets

The targets that depend on this target:

- None.

4

Appendices

3-D visualization tool

Manipulation of the 3-D view: Switch to an appropriate mode as explained below, and modify the camera settings. Double click with the left mouse button to restore the original view.

In order to switch the camera manipulation mode one can use one of two methods. Either click with the right mouse button anywhere outside of the displayed graphics to get a context menu. Or, the controls are available by pressing keys when the mouse pointer is located within the general bounds of the plot:

- hit ”z” key to switch to ZOOM mode,
- hit ”r” key to switch to ROTATION mode,
- hit ”d” key to switch to DOLLY mode (sideways translation),
- hit ”t” key to switch to TARGET mode(setting of the camera target).

Note that the mode is indicated by the shape of the pointer. ZOOM  , ROTATION  , DOLLY  , TARGET  .

- **In ROTATION mode:** Press and hold left mouse button to rotate about screen xy axis. Press and hold middle mouse button to rotate about screen z axis. Press ’-’ to slow down rotation, press ’+’ to speed up rotation.
- **In ZOOM mode:** Press and hold left mouse button to zoom in and out.
- **In DOLLY mode:** press and hold left mouse button to dolly the camera horizontally and vertically.
- **In TARGET mode:** Click on the point to which the camera should be targeted. For the drillstring they are typically the joints between the components, or the joints between finite elements. For the borehole they are the control points of the midline curve.

Index