SAS® GLOBAL FORUM 2020 #SASGF

Paper 4700-2020

Data-Driven Robotics: Leveraging SAS® and Python to Virtually Build LEGO MINDSTORMS Gear Trains for the EV3 Brick

Troy Martin Hughes

ABSTRACT

LEGO MINDSTORMS Evolution 3 (EV3) represents the third-generation programmable "Brick," a hand-held computer developed by the LEGO Group that intelligently drives and forms the cornerstone of LEGO robotics. Released in 2013, EV3 leverages LEGO Group-built sensors (including haptic, gyroscopic, ultrasonic, infrared, and others) and servomotors—rotary motors that track speed, degrees, and angle of rotation—to interpret, interact with, and respond to environmental and user stimuli. Although EV3 robotics locomotion begins with large and medium LEGO servomotors, gears and gear trains facilitate complex actions, movements, and the increase of speed or torque. To this end, this paper introduces LEGO gears and simple gear trains, and includes SAS® code that programmatically identifies how (and how well) LEGO gears mesh with each other in a two-dimensional (2D) plane. Data-driven software evaluates a table of 41 LEGO gears and programmatically determines where on a virtual 9x9 LEGO stud plane the gears can be placed to mesh. Moreover, by modifying additional tables, the 9x9 stud plane can be replaced with other LEGO Technic beams (or other bricks) to demonstrate where gears can be placed. Additionally, a FUZZ parameter enables the user to specify the number of millimeters of gap or overlap permitted between gears. This data-driven design maximizes software flexibility and configurability, providing dynamic output to meet the needs of different users by modifying tables and parameters only—not code. Finally, the interoperability of data-driven design is showcased in that equivalent SAS and Python code are included, both of which rely on the same parameters and control tables. For more information, please consult the unabridged 69-page text (https://communities.sas.com/t5/SAS-Communities-Library/Data-Driven-Robotics-Leveraging-SAS-and-Python-Software-to/ta-p/641990) and the 30-minute 4K video (https://youtu.be/rvFS0rj6ml4).

INTRODUCTION

Given that gears must mesh well with each other to transfer motion efficiently and with minimal friction, gear size and placement is critical within robotics, and within machinery in general. In addition to transferring power, gear trains are commonly employed to "gear up" (i.e., increase the speed while reducing torque) or "gear down" (i.e., decrease the speed while increasing torque) output gears. For example, automobile transmissions use a low gear (high torque/low speed) to overcome inertia when starting, then use higher gears (lower torque/higher speed) once momentum has been achieved. Although compound gear trains operate in three-dimensional (3D) space, this text evaluates only simple gear trains—those in which each axle contains only one gear. Moreover, only simple gear trains (i.e., having parallel axles) are evaluated.

This software showcases data-driven software design, in which the list of LEGO gears and their respective attributes is maintained within a comma-separated values (CSV) file that promotes interoperability and flexibility. Thus, were the LEGO Group to add (or remove) gears from its inventory, as it has done in both 2019 and 2020, these modifications could be made to the CSV file without necessitating any changes to the underlying code. Software (and control data) interoperability is demonstrated in that both the SAS and Python versions of the program rely on the same control table of gear attributes, and read the same LEGO beam files that contain X-Y positions for axle/shaft holes. This interoperability facilitates master data management (MDM) because only one copy of each control table must be is maintained.

This text demonstrates data-driven software design and its application to robotics design. Moreover, to LEGO builders, this text includes a comprehensive listing (via the REPORT procedure) of the ways in which LEGO gears can interact with each other—even those uncommon, non-orthogonal angles—and how well the gears mesh. The dimensions of several angled "studless" LEGO beams are included as CSV files, and end users are able to create additional CSV files to demonstrate how gears can be placed on additional LEGO beams. Finally, goodness of fit for gear pairings can be modified via the FUZZ parameter, which specifies the sensitivity (in millimeters) of the gap or overlap between gears. This configurability especially enables LEGO builders to discover tighter-fitting gear pairings required for higher-torque power transfer.

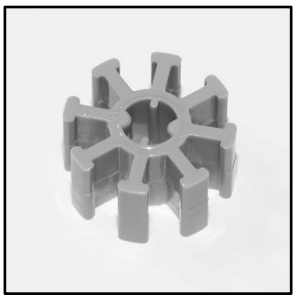
LEGO GEARS

Figure 1 demonstrates the LEGO gears most commonly employed in Technic™ and MINDSTORMS® sets, including gear version (e.g., v1, v2) where gears have been improved over time. All gears are subsequently described in Table 1. Only the final (i.e., color) photo demonstrates gear scale and relative sizes.

8-tooth spur v1

8-tooth spur v2

8-tooth spur v2 (frictionless)



8-tooth timing

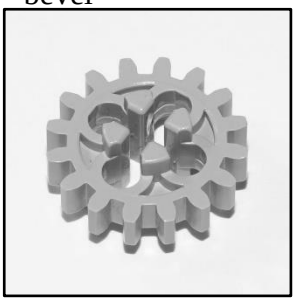
12-tooth bevel

12-tooth double bevel

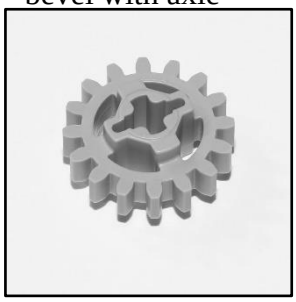
12-tooth double bevel with axle

12-tooth double bevel with arm

14-tooth bevel



16-tooth spur v1



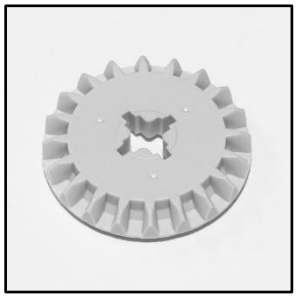
16-tooth spur v2

16-tooth spur with two-sided clutch

16-tooth spur with clutch v1

16-tooth spur with clutch v2

20-tooth bevel with pinhole



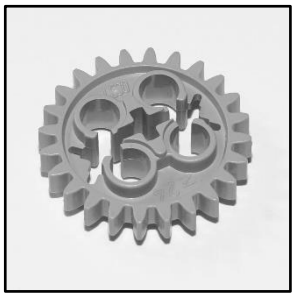
20-tooth bevel

20-tooth double bevel

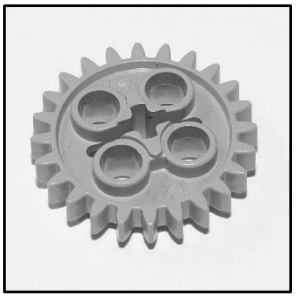
20-tooth double bevel with clutch

24-tooth crown v1

24-tooth crown v2

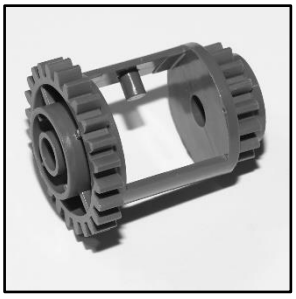


24-tooth spur v1



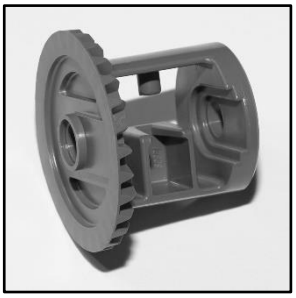
24-tooth spur v2

24-tooth spur (clutch)

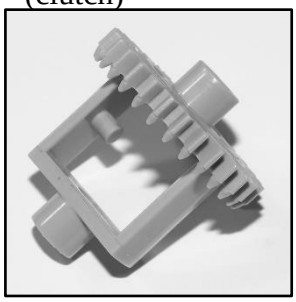


24/16 spur compound differential

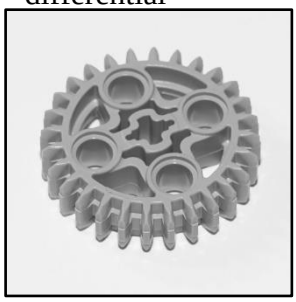
28-tooth bevel differential (closed)



28-tooth bevel differential (open)



28-tooth crown differential

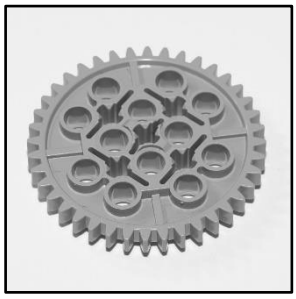


28-tooth double bevel

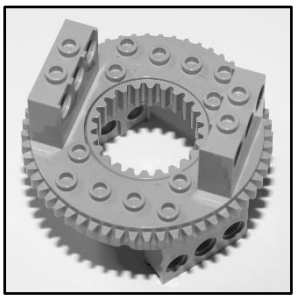
28-tooth double bevel with pinhole

28-tooth turntable

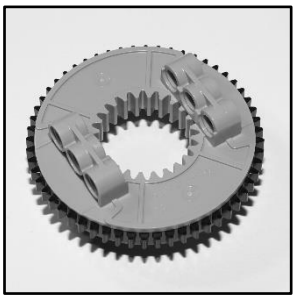
36-tooth double bevel



40-tooth spur



56-tooth turntable (studded)

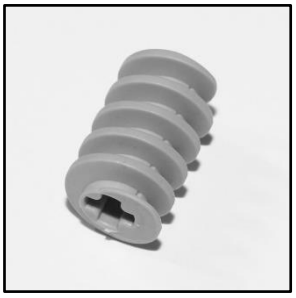


56-tooth turntable (studless)

60-tooth turntable

knob wheel

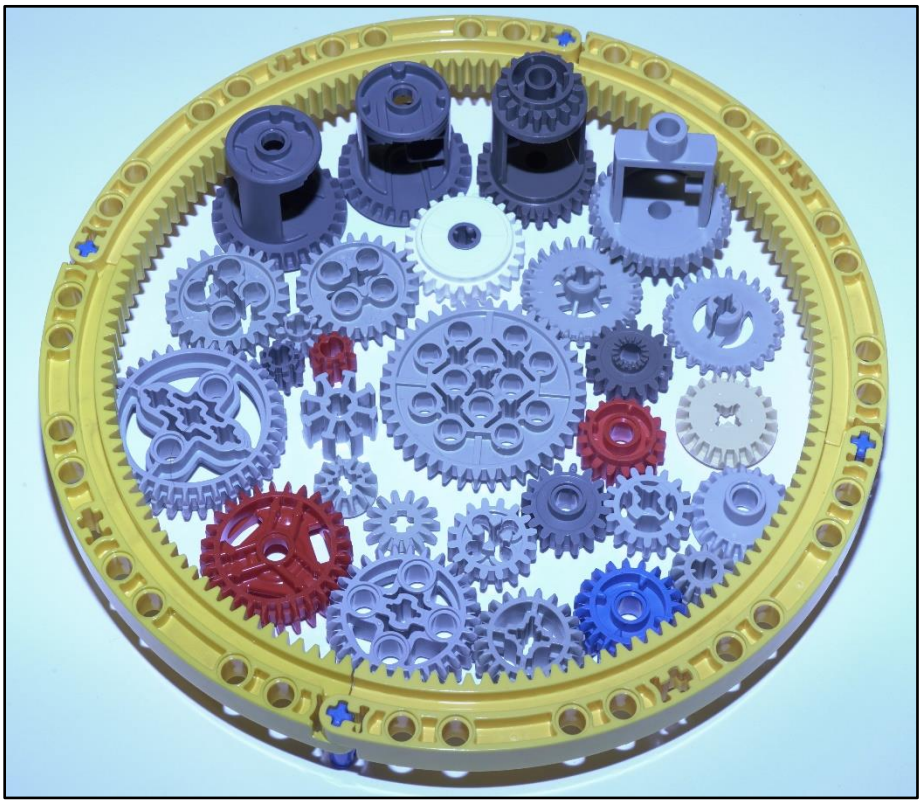
worm screw (short)



worm screw (long) v1

worm screw (long) v2

worm screw with two bushes



140-tooth ring (aka banana) gear with all spur, bevel, double bevel, crown, clutch, and differential gears

Figure 1. Common LEGO Gears

Table 1 enumerates LEGO gears and their attributes, including gear type (i.e., orientation of teeth), number of teeth, gear diameter (i.e., pitch circle), thickness, and whether the gear connects with other gears at 180 degrees (i.e., inline) or 90 degrees (i.e., perpendicular to the primary axle). Spur gears connect to each other with parallel axles, whereas bevel, double bevel, crown gears, knob wheels, and worm gears each facilitate perpendicular transfer of power. Double bevel, crown gears, and knob wheels can flexibly connect either perpendicularly or inline; however, knob wheels can connect only with other knob wheels, so their flexibility is diminished and they are not included in the programmatic gear train evaluation.

Note that the compound differential gear has both 16- and 24-tooth spur gears, so the gear is included twice in the table to facilitate programmatic evaluation of both gears. **For all gears, clicking on the Part Number URL opens the gear's webpage on Bricklink.com, whose historical catalog of LEGO attributes was essential in the creation of this text, and whence many of the pictured gears were purchased.**

#	Gear Type	Teeth	Shaft Type	Part No	Release Years	Diameter (mm)	Thickness (mm)	Inline Mesh	90-Degree Mesh
1	spur (v1)	8	axle	3647	1977-2016	8	8	yes	no
2	spur (v2)	8	axle	10928	2008-2020	8	8	yes	no
3	spur	8	frictionless	11955	2013-2019	8	8	yes	no
4	timing	8	axle	32060	1997-2006	16	8	no	no
5	bevel	12	axle	6589	1993-2020	12	4	no	yes
6	double bevel	12	axle	32270	1999-2020	12	8	yes	yes
7	double bevel arm with axle	12	pinhole	24014	2016-2019	12	8	yes	yes
8	double bevel arm	12	pinhole	41666	2002-2003	12	8	yes	yes
9	bevel	14	axle	4143	1980-2002	14	3	no	yes
10	spur (v1)	16	axle	4019	1979-2016	16	8	yes	no
11	spur (v2)	16	axle	94925	2006-2020	16	8	yes	no
12	spur with clutch (v1)	16	axle	6542	1993-2014	16	8	yes	no
13	spur with clutch (v2)	16	axle	6542b	2009-2016	16	8	yes	no
14	spur with two-sided clutch	16	pinhole	18946	2015-2020	16	8	yes	no
15	spur with compound differential	16	axle	6573	1994-2019	16	32	yes	no

#	Gear Type	Teeth	Shaft Type	Part No	Release Years	Diameter (mm)	Thickness (mm)	Inline Mesh	90-Degree Mesh
16	bevel	20	axle	32198	1999-2020	20	8	no	yes
17	bevel	20	pinhole	87407	2010-2020	20	8	no	yes
18	double bevel	20	axle	32269	1999-2020	20	8	yes	yes
19	double bevel with two-sided clutch	20	pinhole	35185	2018-2020	20	8	yes	yes
20	spur (v1)	24	axle	x187	1977-1998	24	8	yes	no
21	spur (v2	24	axle	3648	1993-2020	24	8	yes	no
22	spur (clutch)	24	axle	60c01	1997-2018	24	8	yes	no
23	crown (v1)	24	axle	3650a	1977-1997	24	8	yes	yes
24	crown (v2)	24	axle	3650b	1985-2016	24	8	yes	yes
25	spur with compound differential	24	pinhole	6573	1994-2019	24	32	yes	no
26	double bevel	28	axle	46372	2019-2019	28	8	yes	yes
27	double bevel	28	pinhole	65413	2020-2020	28	8	yes	yes
28	crown differential	28	pinhole	73071	1980-1994	28	32	yes	yes
29	bevel differential with open center	28	pinhole	62821a	2008-2016	28	32	no	yes
30	bevel differential with closed center	28	pinhole	62821b	2009-2019	28	32	no	yes
31	turntable	28	8x8 mm square	99009c01	2012-2019	28	24	yes	no
32	double bevel	36	axle	32498	2002-2020	36	8	yes	yes
33	spur	40	axle	3649	1977-2019	40	8	yes	no
34	turntable	56	24 mm circle	48452cx1	2004-2016	56	24	yes	no
35	turntable with studs	56	24 mm circle	2856c01	1990-2003	56	44	yes	no
36	turntable	60	24 mm circle	18939c01	2015-2020	60	24	yes	no

#	Gear Type	Teeth	Shaft Type	Part No	Release Years	Diameter (mm)	Thickness (mm)	Inline Mesh	90-Degree Mesh
37	ring gear 11x11 curved rack	140	136 mm circle	24121	2016-2019	168	8	no	no
38	knob wheel	1	axle	32072	1998-2020	16	8	yes	yes
39	worm screw (v1)	1	frictionless	4716	1985-2017	32	.	no	yes
40	worm screw (v2)	1	frictionless	32905	2009-2019	32	.	no	yes
41	worm screw (short)	1	frictionless	27938	2017-2020	16	.	no	yes
42	worm screw with bushes	1	bushes	15457	2014-2014	24	.	no	yes

Table 1. SAS Report Created by GEAR_REPORT Macro and Derived from Gears.csv

The report in Table 1 is derived from Gears.csv, which should be saved as:

```
Gear Type,Teeth,Hole,Part Number,Years Active,Diameter (mm),Thickness (mm),180
Degrees / Inline,90 Degrees / Perpendicular
spur (v1),8,axle,3647,1977-2016,8,8,yes,no
spur (v2),8,axle,10928,2008-2020,8,8,yes,no
spur,8,frictionless,11955,2013-2019,8,8,yes,no
timing,8,axle,32060,1997-2006,16,8,no,no
bevel,12,axle,6589,1993-2020,12,4,no,yes
double bevel,12,axle,32270,1999-2020,12,8,yes,yes
double bevel arm with axle,12,pinhole,24014,2016-2019,12,8,yes,yes
double bevel arm,12,pinhole,41666,2002-2003,12,8,yes,yes
bevel,14,axle,4143,1980-2002,14,3,no,yes
spur (v1),16,axle,4019,1979-2016,16,8,yes,no
spur (v2),16,axle,94925,2006-2020,16,8,yes,no
spur with clutch (v1),16,axle,6542,1993-2014,16,8,yes,no
spur with clutch (v2),16,axle,6542b,2009-2016,16,8,yes,no
spur with two-sided clutch,16,pinhole,18946,2015-2020,16,8,yes,no
spur with compound differential,16,axle,6573,1994-2019,16,32,yes,no
bevel,20,axle,32198,1999-2020,20,8,no,yes
bevel,20,pinhole,87407,2010-2020,20,8,no,yes
double bevel,20,axle,32269,1999-2020,20,8,yes,yes
double bevel with two-sided clutch,20,pinhole,35185,2018-2020,20,8,yes,yes
spur (v1),24,axle,x187,1977-1998,24,8,yes,no
spur (v2,24,axle,3648,1993-2020,24,8,yes,no
spur (clutch),24,axle,60c01,1997-2018,24,8,yes,no
crown (v1),24,axle,3650a,1977-1997,24,8,yes,yes
crown (v2),24,axle,3650b,1985-2016,24,8,yes,yes
spur with compound differential,24,pinhole,6573,1994-2019,24,32,yes,no
double bevel,28,axle,46372,2019-2019,28,8,yes,yes
double bevel,28,pinhole,65413,2020-2020,28,8,yes,yes
crown differential,28,pinhole,73071,1980-1994,28,32,yes,yes
bevel differential with open center,28,pinhole,62821a,2008-2016,28,32,no,yes
bevel differential with closed center,28,pinhole,62821b,2009-2019,28,32,no,yes
turntable,28,8x8 mm square,99009c01,2012-2019,28,24,yes,no
double bevel,36,axle,32498,2002-2020,36,8,yes,yes
spur,40,axle,3649,1977-2019,40,8,yes,no
turntable,56,24 mm circle,48452cx1,2004-2016,56,24,yes,no
turntable with studs,56,24 mm circle,2856c01,1990-2003,56,44,yes,no
turntable,60,24 mm circle,18939c01,2015-2020,60,24,yes,no
ring gear 11x11 curved rack,140,136 mm circle,24121,2016-2019,168,8,no,no
knob wheel,1,axle,32072,1998-2020,16,8,yes,yes
```

```
worm screw (v1),1,frictionless,4716,1985-2017,32,N/A,no,yes
worm screw (v2),1,frictionless,32905,2009-2019,32,N/A,no,yes
worm screw (short),1,frictionless,27938,2017-2020,16,N/A,no,yes
worm screw with bushes,1,bushes,15457,2014-2014,24,N/A,no,yes
```

The following code creates Table 1 by invoking the GEAR_REPORT macro (saved within the Lego_gear_combinations.sas file, included in Appendix A), and must be modified to reference the user's path:

```
%let loc=/folders/myfolders/legos/; * USER MUST CHANGE LOCATION;
%let gearfile=&loc.gears.csv;
%include "&loc.lego_gear_combinations.sas";

%gear_report(csvfile=&loc.gears.csv, rptpath=&loc, rptfile=gear_report.html);
```

LEGO MODULARITY AND STANDARD UNIT OF MEASUREMENT

The standard unit of measurement (i.e., "stud") for LEGO bricks is 8mm, which forms the basis of interaction among not only gears, axles, pins, and bushes, but also beams and other LEGO bricks. For example, a five-stud beam is 40mm in length—five studs at 8mm a piece—and contains five holes through which axles can pass. Figure 2 demonstrates beams ranging from five studs (i.e., 40mm) to 15 studs (i.e., 120mm).

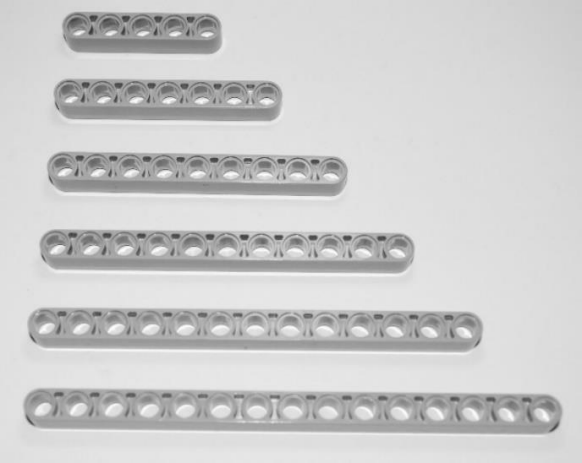


Figure 2. 5-, 7-, 9-, 11-, 13-, and 15-Stud Studless Beams

Thus, six eight-tooth gears can occupy six consecutive (i.e., inline) holes within a studless beam. This same distance (48mm) can be traversed with two 24-tooth spur (or crown or clutch) gears. Figure 3 demonstrates these two gear trains. Note that in both cases, because the first and last gears in each gear train have the same number of teeth, the gear trains have a 1:1 gear ratio in which both torque and speed remain constant.

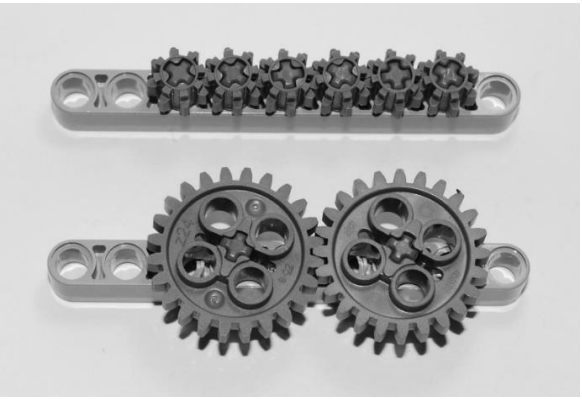


Figure 3. Six 8-Tooth Spur Gears Versus Two 24-Tooth Spur Gears

Gear trains that lie inline with LEGO beams (as depicted in Figure 3) can be conceptualized and designed more readily because gear placement can be determined by “counting the holes” that lie along the beam. Gear train design becomes more complex, however, when gears interact at angles that are non-orthogonal to their underlying beams. For example, Figure 4 demonstrates the placement of a 12-tooth double bevel gear in the upper-left hole of a 9x9 stud matrix of beams, and a 24-tooth spur gear in the third column and second row. This non-orthogonal pairing of two gears is less apparent than orthogonal pairings, which are either horizontal or vertical. Too often, non-orthogonal gear pairings are discovered by LEGO builders through trial and error in a tedious and inefficient fashion.

Figure 4. Non-Orthogonal Pairing of 12-Tooth Double Bevel and 24-Tooth Spur Gears

Where turntables are utilized, and because they cannot be attached with an axle (like other gears), non-orthogonal pairings are even more complex and less likely to be discovered or implemented. For example, Figure 5 demonstrates the pairing of a 28-tooth turntable and a 40-tooth spur gear in the fourth column and fourth row. Note that the 9x9 matrix of beams must be expanded in the upper-left corner to accommodate attachment of the turntable.

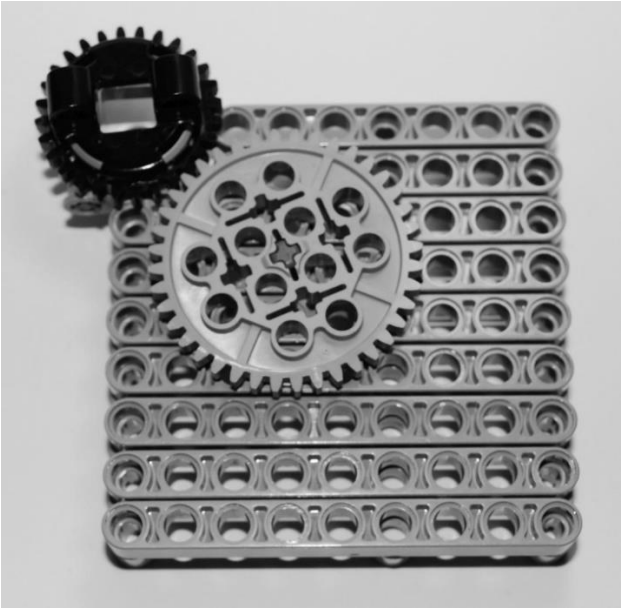


Figure 5. Non-Orthogonal Pairing of 28-Tooth Turntable and 40-Tooth Spur Gear

These and other non-orthogonal gear pairings were the rationale for this text; their discovery should be comprehensive and efficient rather than haphazard, and their catalog should be complete and persistent. This text provides not only a comprehensive listing of all (inline) LEGO simple gear trains but also the methodology and code to create this listing programmatically. In seconds, the software analyzes millions of potential gear-pairing permutations to evaluate which will produce viable gear trains, irrespective of the angle of gear alignment, and with respect to the goodness of fit (i.e., gap or overlap) of the two gears.

SETUP AND INVOCATION

The SAS program Lego_gear_combinations.sas (see Appendix A) contains four macros:

1. **GEAR_REPORT** – generates an HTML report of gears and attributes, as previously described
2. **CREATE_RECTANGLE** – generates a CSV file that includes LEGO stud positions (into which gear axles can be placed) within a virtual matrix of LEGO axle holes
3. **GEAR_TRAIN_REPORT** – executed by EVAL_COORDS to create the HTML gear train report
4. **EVAL_COORDS** – evaluates goodness of fit for all gears for all stud positions

CREATE_RECTANGLE can be used to generate a 9x9 stud grid that models nine 9-stud beams. In this demonstration, SAS University Edition is utilized so the program is saved to the Legos folder (..\SASUniversityEdition\myfolders\legos\) and is invoked with the following code:

```
%let loc=/folders/myfolders/legos/; * USER MUST CHANGE LOCATION;
%let gearfile=&loc.gears.csv;
%include "&loc.lego_gear_combinations.sas";

%create_rectangle(csvfile=&loc.coords_rectangle_9x9.csv,width=9, height=9);
```

The EVAL_COORDS macro (as well as the subsequent EVAL_COORDS Python function) has the following parameters:

- **GEARFILE** – The path and file name of Gears.csv.
- **BEAMFILE** – The path and file name of the stud positions (i.e., axle hole X and Y coordinates in millimeters) being analyzed. CREATE_RECTANGLE generates rectangular matrix files, although additional CSV files are included in subsequent subsections for several LEGO L beams and angled beams. Each row contains X and Y stud positions (in millimeters) delimited by a comma, so “16,8” (without quotes) references the hole two studs to the right and one stud down (of the origin, which may or may not be explicitly included as one of the X-Y values in the CSV file).
- **FUZZ** – The number of millimeters (integer or decimal) of gear overlap or gear gap that will be discovered. At 1mm overlap, gear axles can be bent sufficiently to increase friction beyond a useable level, and at a 1mm gap, axles can be sufficiently loose such that they slip in higher torque.
- **HOLE** – **FIRST** directs that only the first hole encountered in the BEAMFILE file will be used for the position for the primary gears in each gear train analysis. Thus, when FIRST is indicated, the secondary gear will be (virtually) attempted in every position listed in the BEAMFILE file, whereas the primary gear will remain stable in the first position. For example, if the 9x9 matrix is utilized, the first position generated will be 0,0, thus all gear trains analyzed will have the primary gear at position 0,0 and only the secondary gear position will vary.

  ALL directs that both the primary and secondary gears will be virtually fitted in all X-Y positions found in the BEAMFILE file, thus ALL effectively directs a cartesian join of positions be performed. FIRST is more useful when a matrix (like 9x9) is being analyzed, whereas ALL is more useful when specific LEGO beams or frames are being analyzed.
- **VERBOSITY** – **VERBOSE** (any Zork fans?) indicates that all specific gears will be evaluated and thereby listed in the gear train analysis report. For example, although the 28-tooth double bevel, 28-tooth double bevel with pinhole, and 28-tooth crown differential will each connect identically with

other gears (when forming an inline simple gear train), each of these gears will be listed separately in the VERBOSE HTML report.

BRIEF, conversely, lists gears uniquely by number of teeth only. Thus, these three distinct 28-tooth gears will be represented only once as a generic "28-tooth" entry in the HTML report, rather than multiple entries that additionally describe other attributes. BRIEF should almost always be selected.

- **RPTPATH** – Path of the HTML gear train analysis report.
- **RPTFILE** – File name (and extension) of the HTML gear train analysis report.

With the 9x9 stud grid created, the EVAL_COORDS macro ingests and analyzes the Gears.csv control table with respect to the underlying rectangular grid of beams:

```
%eval_coords(gearfile=&loc.gears.csv,
   beamfile=&loc.coords_rectangle_9x9.csv,
   fuzz=.5,
   hole=first,
   verbosity=verbose,
   rptpath=&loc,
   rptfile=Lego_gear_trains_9x9_1stpos_brief.html);
```

This invocation creates a 786-line HTML report, the first few lines of which are demonstrated in Table 2.

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
1	8-tooth spur (frictionless)	0	0	8-tooth spur (frictionless)	1	0	0.0	0.000	1:1
2		0	0	8-tooth spur (v1)	1	0	0.0	0.000	1:1
3		0	0	8-tooth spur (v2)	1	0	0.0	0.000	1:1
4		0	0	24-tooth crown (v1)	2	0	0.0	0.000	3:1
5		0	0	24-tooth crown (v2)	2	0	0.0	0.000	3:1
6		0	0	24-tooth spur (clutch)	2	0	0.0	0.000	3:1
7		0	0	24-tooth spur (v1)	2	0	0.0	0.000	3:1
8		0	0	24-tooth spur (v2	2	0	0.0	0.000	3:1
9		0	0	24-tooth spur with compound differential (pinhole)	2	0	0.0	0.000	3:1
10		0	0	40-tooth spur	3	0	0.0	0.000	5:1

Table 2. Demonstration of VERBOSE mode for VERBOSITY Parameter

Note that in VERBOSE mode, each type of 8-tooth gear is matched with every other type of 8-tooth gear, thus resulting in three different lines in which the frictionless 8-tooth gear is paired with 8-tooth gears. In

BRIEF mode, which is demonstrated in all subsequent examples, only one generic “8-tooth” gear is listed so the number of gears being tested dramatically decreases and output readability is improved.

In this sample invocation, the FUZZ parameter specifies that only gear pairings that are within 0.5mm from a theoretical perfect mesh will be identified. For example, two eight-tooth gears (each having a radius of 4mm for a combined 8mm) placed in adjoining holes (8mm apart) form a perfect mesh with neither gap nor overlap. FUZZ enables less-than-perfect gear pairings to be detected and utilized, as a slight amount of mesh disparity is often tolerable when gears are utilized in low-torque conditions. As torque is increased, however, gear pairings in which a gap exists may cause the gears to slip and, in some cases, this slip can damage or destroy one or both gears.

The disparity between two gears is calculated by subtracting the sum of the radii for each of the two gears from the distance between the gear axle centers (i.e., the hypotenuse from the coordinates of the primary gear to the secondary gear). For example, the two gear trains in Figure 3 each have no disparity because they mesh perfectly. However, the non-orthogonal gears demonstrated in Figure 4 do not mesh perfectly, as the sum of their radii is 18mm whereas the distance between their axle holes is 17.889mm. For most if not all purposes, this negligible overlap is unnoticeable, and absent analysis, many LEGO builders might be unaware of this overlap.

Similarly, the non-orthogonal pairing of gears in Figure 5 may appear to be a perfect mesh, and for all purposes may be, but mathematically this pairing produces an infinitesimal overlap. The sum of the radii of the gears is 34mm whereas the distance between their axle holes is 33.941mm. To put this 0.059mm overlap in perspective, this snugness is less than one percent of the thickness of the 8mm LEGO stud.

EVAL_COORDS creates the Matching_coords data set that includes all gear pairings that match the minimum FUZZ requirement, and which includes the following variables:

- **Gear1** – primary gear (number of teeth, gear type, and shaft type if not “axle”) being analyzed
- **Xmm_start** – horizontal position (measured in millimeters) of primary gear
- **Ymm_start** – vertical position (measured in millimeters) of primary gear
- **Xstud_start** – horizontal position (measured in 8mm studs) of primary gear
- **Ystud_start** – vertical position (measured in 8mm studs) of primary gear
- **Gear2** – secondary gear (number of teeth, gear type, and shaft type if not “axle”) being analyzed
- **Xmm** – horizontal position (measured in millimeters) of secondary gear
- **Ymm** – vertical position (measured in millimeters) of secondary gear
- **Xstud** – horizontal position (measured in 8mm studs) of secondary gear
- **Ystud** – vertical position (measured in 8mm studs) of secondary gear
- **Gear_distance** – sum of center distance radii (in millimeters) of both gears
- **Stud_distance** – distance (in millimeters) between the two axle holes, from the primary gear center to the secondary gear center
- **Theta** – the angle (in degrees) between the primary gear and secondary gear
- **Gap_overlap** – when positive, the gap between two gears, and when negative, the overlap between two gears
- **Ratio** – the gear ratio of number of teeth on the secondary gear (i.e., driven gear) to the number of teeth on the primary gear (i.e., drive gear), in which ratios greater than one increase the torque of the secondary gear and ratios smaller than one increase the speed of the secondary gear

When FUZZ is set to 0, only perfect gear pairings are included—those that are both the strongest and most efficient—in which the Gear_distance equals the Stud_distance. A positive Gap_overlap value indicates that a gap exists between the gears, and gear pairings with small gaps can be useful. Negative Gap_overlap

values represent gears that overlap subtly, which are less useful because they tend to cause bunching and friction. As demonstrated in both Figures 4 and 5, however, non-orthogonal, imperfect pairings are often utilized, and many can perform well under high-speed and high-torque circumstances.

The following invocation now demonstrates all gear trains in which the FUZZ is 1mm and VERBOSITY is BRIEF:

```
%eval_coords(gearfile=&loc.gears.csv,
   beamfile=&loc.coords_rectangle_9x9.csv,
   fuzz=1,
   hole=first,
   verbosity=brief,
   rptpath=&loc,
   rptfile=Lego_gear_trains_9x9_1stpos_brief.html);
```

Table 3 demonstrates the results of the REPORT procedure within EVAL_COORDS, showing 184 gear trains. This analysis is recreated utilizing Python in a subsequent section that demonstrates interoperability.

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
1	8-tooth	0	0	8-tooth	1	0	0.0	0.000	1:1
2		0	0	24-tooth	2	0	0.0	0.000	3:1
3		0	0	40-tooth	3	0	0.0	0.000	5:1
4		0	0	56-tooth	4	0	0.0	0.000	7:1
5		0	0	8-tooth	0	1	90.0	0.000	1:1
6		0	0	16-tooth	1	1	45.0	-0.686	2:1
7		0	0	28-tooth	2	1	26.6	-0.111	3.5:1
8		0	0	56-tooth	4	1	14.0	0.985	7:1
9		0	0	24-tooth	0	2	90.0	0.000	3:1
10		0	0	28-tooth	1	2	63.4	-0.111	3.5:1
11		0	0	36-tooth	2	2	45.0	0.627	4.5:1
12		0	0	40-tooth	0	3	90.0	0.000	5:1
13		0	0	60-tooth	3	3	45.0	-0.059	7.5:1
14		0	0	56-tooth	0	4	90.0	0.000	7:1
15		0	0	56-tooth	1	4	76.0	0.985	7:1

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
16	12-tooth	0	0	20-tooth	2	0	0.0	0.000	1.666:1
17		0	0	36-tooth	3	0	0.0	0.000	3:1
18		0	0	12-tooth	1	1	45.0	-0.686	1:1
19		0	0	24-tooth	2	1	26.6	-0.111	2:1
20		0	0	40-tooth	3	1	18.4	-0.702	3.333:1
21		0	0	20-tooth	0	2	90.0	0.000	1.666:1
22		0	0	24-tooth	1	2	63.4	-0.111	2:1
23		0	0	60-tooth	4	2	26.6	-0.223	5:1
24		0	0	36-tooth	0	3	90.0	0.000	3:1
25		0	0	40-tooth	1	3	71.6	-0.702	3.333:1
26		0	0	56-tooth	3	3	45.0	-0.059	4.666:1
27		0	0	60-tooth	2	4	63.4	-0.223	5:1
28	16-tooth	0	0	16-tooth	2	0	0.0	0.000	1:1
29		0	0	8-tooth	1	1	45.0	-0.686	1:2
30		0	0	20-tooth	2	1	26.6	-0.111	1.25:1
31		0	0	36-tooth	3	1	18.4	-0.702	2.25:1
32		0	0	16-tooth	0	2	90.0	0.000	1:1
33		0	0	20-tooth	1	2	63.4	-0.111	1.25:1
34		0	0	28-tooth	2	2	45.0	0.627	1.75:1
35		0	0	40-tooth	3	2	33.7	0.844	2.5:1
36		0	0	56-tooth	4	2	26.6	-0.223	3.5:1
37		0	0	36-tooth	1	3	71.6	-0.702	2.25:1

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
38		0	0	40-tooth	2	3	56.3	0.844	2.5:1
39		0	0	56-tooth	2	4	63.4	-0.223	3.5:1
40	20-tooth	0	0	12-tooth	2	0	0.0	0.000	1:1.666
41		0	0	28-tooth	3	0	0.0	0.000	1.4:1
42		0	0	60-tooth	5	0	0.0	0.000	3:1
43		0	0	16-tooth	2	1	26.6	-0.111	1:1.25
44		0	0	60-tooth	5	1	11.3	0.792	3:1
45		0	0	12-tooth	0	2	90.0	0.000	1:1.666
46		0	0	16-tooth	1	2	63.4	-0.111	1:1.25
47		0	0	24-tooth	2	2	45.0	0.627	1.2:1
48		0	0	36-tooth	3	2	33.7	0.844	1.8:1
49		0	0	28-tooth	0	3	90.0	0.000	1.4:1
50		0	0	36-tooth	2	3	56.3	0.844	1.8:1
51		0	0	60-tooth	4	3	36.9	0.000	3:1
52		0	0	60-tooth	3	4	53.1	0.000	3:1
53		0	0	60-tooth	0	5	90.0	0.000	3:1
54		0	0	60-tooth	1	5	78.7	0.792	3:1
55	24-tooth	0	0	8-tooth	2	0	0.0	0.000	1:3
56		0	0	24-tooth	3	0	0.0	0.000	1:1
57		0	0	40-tooth	4	0	0.0	0.000	1.666:1
58		0	0	56-tooth	5	0	0.0	0.000	2.333:1
59		0	0	12-tooth	2	1	26.6	-0.111	1:2

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
60		0	0	28-tooth	3	1	18.4	-0.702	1.166:1
61		0	0	40-tooth	4	1	14.0	0.985	1.666:1
62		0	0	56-tooth	5	1	11.3	0.792	2.333:1
63		0	0	8-tooth	0	2	90.0	0.000	1:3
64		0	0	12-tooth	1	2	63.4	-0.111	1:2
65		0	0	20-tooth	2	2	45.0	0.627	1:1.2
66		0	0	24-tooth	0	3	90.0	0.000	1:1
67		0	0	28-tooth	1	3	71.6	-0.702	1.166:1
68		0	0	56-tooth	4	3	36.9	0.000	2.333:1
69		0	0	40-tooth	0	4	90.0	0.000	1.666:1
70		0	0	40-tooth	1	4	76.0	0.985	1.666:1
71		0	0	56-tooth	3	4	53.1	0.000	2.333:1
72		0	0	56-tooth	0	5	90.0	0.000	2.333:1
73		0	0	56-tooth	1	5	78.7	0.792	2.333:1
74	28-tooth	0	0	20-tooth	3	0	0.0	0.000	1:1.4
75		0	0	36-tooth	4	0	0.0	0.000	1.285:1
76		0	0	8-tooth	2	1	26.6	-0.111	1:3.5
77		0	0	24-tooth	3	1	18.4	-0.702	1:1.166
78		0	0	36-tooth	4	1	14.0	0.985	1.285:1
79		0	0	8-tooth	1	2	63.4	-0.111	1:3.5
80		0	0	16-tooth	2	2	45.0	0.627	1:1.75
81		0	0	28-tooth	3	2	33.7	0.844	1:1

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
82		0	0	60-tooth	5	2	21.8	-0.919	2.142:1
83		0	0	20-tooth	0	3	90.0	0.000	1:1.4
84		0	0	24-tooth	1	3	71.6	-0.702	1:1.166
85		0	0	28-tooth	2	3	56.3	0.844	1:1
86		0	0	40-tooth	3	3	45.0	-0.059	1.428:1
87		0	0	36-tooth	0	4	90.0	0.000	1.285:1
88		0	0	36-tooth	1	4	76.0	0.985	1.285:1
89		0	0	60-tooth	2	5	68.2	-0.919	2.142:1
90	36-tooth	0	0	12-tooth	3	0	0.0	0.000	1:3
91		0	0	28-tooth	4	0	0.0	0.000	1:1.285
92		0	0	60-tooth	6	0	0.0	0.000	1.666:1
93		0	0	16-tooth	3	1	18.4	-0.702	1:2.25
94		0	0	28-tooth	4	1	14.0	0.985	1:1.285
95		0	0	60-tooth	6	1	9.5	0.662	1.666:1
96		0	0	8-tooth	2	2	45.0	0.627	1:4.5
97		0	0	20-tooth	3	2	33.7	0.844	1:1.8
98		0	0	36-tooth	4	2	26.6	-0.223	1:1
99		0	0	12-tooth	0	3	90.0	0.000	1:3
100		0	0	16-tooth	1	3	71.6	-0.702	1:2.25
101		0	0	20-tooth	2	3	56.3	0.844	1:1.8
102		0	0	56-tooth	5	3	31.0	0.648	1.555:1
103		0	0	28-tooth	0	4	90.0	0.000	1:1.285

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
104		0	0	28-tooth	1	4	76.0	0.985	1:1.285
105		0	0	36-tooth	2	4	63.4	-0.223	1:1
106		0	0	56-tooth	4	4	45.0	-0.745	1.555:1
107		0	0	56-tooth	3	5	59.0	0.648	1.555:1
108		0	0	60-tooth	0	6	90.0	0.000	1.666:1
109		0	0	60-tooth	1	6	80.5	0.662	1.666:1
110	40-tooth	0	0	8-tooth	3	0	0.0	0.000	1:5
111		0	0	24-tooth	4	0	0.0	0.000	1:1.666
112		0	0	40-tooth	5	0	0.0	0.000	1:1
113		0	0	56-tooth	6	0	0.0	0.000	1.4:1
114		0	0	12-tooth	3	1	18.4	-0.702	1:3.333
115		0	0	24-tooth	4	1	14.0	0.985	1:1.666
116		0	0	40-tooth	5	1	11.3	0.792	1:1
117		0	0	56-tooth	6	1	9.5	0.662	1.4:1
118		0	0	16-tooth	3	2	33.7	0.844	1:2.5
119		0	0	60-tooth	6	2	18.4	0.596	1.5:1
120		0	0	8-tooth	0	3	90.0	0.000	1:5
121		0	0	12-tooth	1	3	71.6	-0.702	1:3.333
122		0	0	16-tooth	2	3	56.3	0.844	1:2.5
123		0	0	28-tooth	3	3	45.0	-0.059	1:1.428
124		0	0	40-tooth	4	3	36.9	0.000	1:1
125		0	0	24-tooth	0	4	90.0	0.000	1:1.666

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
126		0	0	24-tooth	1	4	76.0	0.985	1:1.666
127		0	0	40-tooth	3	4	53.1	0.000	1:1
128		0	0	40-tooth	0	5	90.0	0.000	1:1
129		0	0	40-tooth	1	5	78.7	0.792	1:1
130		0	0	56-tooth	0	6	90.0	0.000	1.4:1
131		0	0	56-tooth	1	6	80.5	0.662	1.4:1
132		0	0	60-tooth	2	6	71.6	0.596	1.5:1
133	56-tooth	0	0	8-tooth	4	0	0.0	0.000	1:7
134		0	0	24-tooth	5	0	0.0	0.000	1:2.333
135		0	0	40-tooth	6	0	0.0	0.000	1:1.4
136		0	0	56-tooth	7	0	0.0	0.000	1:1
137		0	0	8-tooth	4	1	14.0	0.985	1:7
138		0	0	24-tooth	5	1	11.3	0.792	1:2.333
139		0	0	40-tooth	6	1	9.5	0.662	1:1.4
140		0	0	56-tooth	7	1	8.1	0.569	1:1
141		0	0	16-tooth	4	2	26.6	-0.223	1:3.5
142		0	0	60-tooth	7	2	15.9	0.241	1.071:1
143		0	0	12-tooth	3	3	45.0	-0.059	1:4.666
144		0	0	24-tooth	4	3	36.9	0.000	1:2.333
145		0	0	36-tooth	5	3	31.0	0.648	1:1.555
146		0	0	8-tooth	0	4	90.0	0.000	1:7
147		0	0	8-tooth	1	4	76.0	0.985	1:7

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
148		0	0	16-tooth	2	4	63.4	-0.223	1:3.5
149		0	0	24-tooth	3	4	53.1	0.000	1:2.333
150		0	0	36-tooth	4	4	45.0	-0.745	1:1.555
151		0	0	60-tooth	6	4	33.7	-0.311	1.071:1
152		0	0	24-tooth	0	5	90.0	0.000	1:2.333
153		0	0	24-tooth	1	5	78.7	0.792	1:2.333
154		0	0	36-tooth	3	5	59.0	0.648	1:1.555
155		0	0	56-tooth	5	5	45.0	0.569	1:1
156		0	0	40-tooth	0	6	90.0	0.000	1:1.4
157		0	0	40-tooth	1	6	80.5	0.662	1:1.4
158		0	0	60-tooth	4	6	56.3	-0.311	1.071:1
159		0	0	56-tooth	0	7	90.0	0.000	1:1
160		0	0	56-tooth	1	7	81.9	0.569	1:1
161		0	0	60-tooth	2	7	74.1	0.241	1.071:1
162	60-tooth	0	0	20-tooth	5	0	0.0	0.000	1:3
163		0	0	36-tooth	6	0	0.0	0.000	1:1.666
164		0	0	20-tooth	5	1	11.3	0.792	1:3
165		0	0	36-tooth	6	1	9.5	0.662	1:1.666
166		0	0	12-tooth	4	2	26.6	-0.223	1:5
167		0	0	28-tooth	5	2	21.8	-0.919	1:2.142
168		0	0	40-tooth	6	2	18.4	0.596	1:1.5
169		0	0	56-tooth	7	2	15.9	0.241	1:1.071

#	Primary Gear	Gear 1 (X)	Gear 1 (Y)	Secondary Gear	Gear 2 (X)	Gear 2 (Y)	Angle (degrees) between gears	Gap (+) or Overlap (-) in mm	Gear Ratio
170		0	0	8-tooth	3	3	45.0	-0.059	1:7.5
171		0	0	20-tooth	4	3	36.9	0.000	1:3
172		0	0	60-tooth	7	3	23.2	0.926	1:1
173		0	0	12-tooth	2	4	63.4	-0.223	1:5
174		0	0	20-tooth	3	4	53.1	0.000	1:3
175		0	0	56-tooth	6	4	33.7	-0.311	1:1.071
176		0	0	20-tooth	0	5	90.0	0.000	1:3
177		0	0	20-tooth	1	5	78.7	0.792	1:3
178		0	0	28-tooth	2	5	68.2	-0.919	1:2.142
179		0	0	36-tooth	0	6	90.0	0.000	1:1.666
180		0	0	36-tooth	1	6	80.5	0.662	1:1.666
181		0	0	40-tooth	2	6	71.6	0.596	1:1.5
182		0	0	56-tooth	4	6	56.3	-0.311	1:1.071
183		0	0	56-tooth	2	7	74.1	0.241	1:1.071
184		0	0	60-tooth	3	7	66.8	0.926	1:1

Table 3. Simple Gear Trains with 1mm FUZZ Parameter Specified

For comparison, had this analysis been run with VERBOSITY=VERBOSE, it would have produced 1,285 distinct gear trains rather than 184, demonstrating the advantages of specifying VERBOSITY=BRIEF.

Utilizing this table, LEGO builders can quickly look up combinations of gears to determine how they can mesh, in what stud positions the gears should be placed, and what resultant angle and gear ratio will be produced. Beyond 1mm FUZZ, gears are either too loose to mesh or too tight to turn so these extremes are omitted. In the next section, the analysis of gear train placement on additional LEGO beams is demonstrated.

ANALYZING GEAR TRAIN PLACEMENT ON SPECIFIC LEGO BEAMS

The benefits of data-driven design have already been demonstrated, in that as the LEGO Group adds or modifies its inventory of gears, these changes can be made to the Gears.csv file; thereafter, the software can be rerun without the necessity to modify any code. Data-driven design further benefits the analysis of LEGO beams and frames because these dimensions can be encoded in CSV files and referenced with the

BEAMFILE parameter. The following examples enumerate specific beams and their dimensions, and subsequently include analysis reports. All analyses specify HOLE=ALL and VERBOSITY=BRIEF.

Figure 6 demonstrates the LEGO beams and frames that will be analyzed. The stud positions (i.e., X-Y coordinates listed in the beam CSV files) correspond to the placement and directionality of the beams as they appear in Figure 6. All fixed axle holes (which do not allow an axle to rotate) appear in red circles in Figure 6 and are not included in the CSV files.

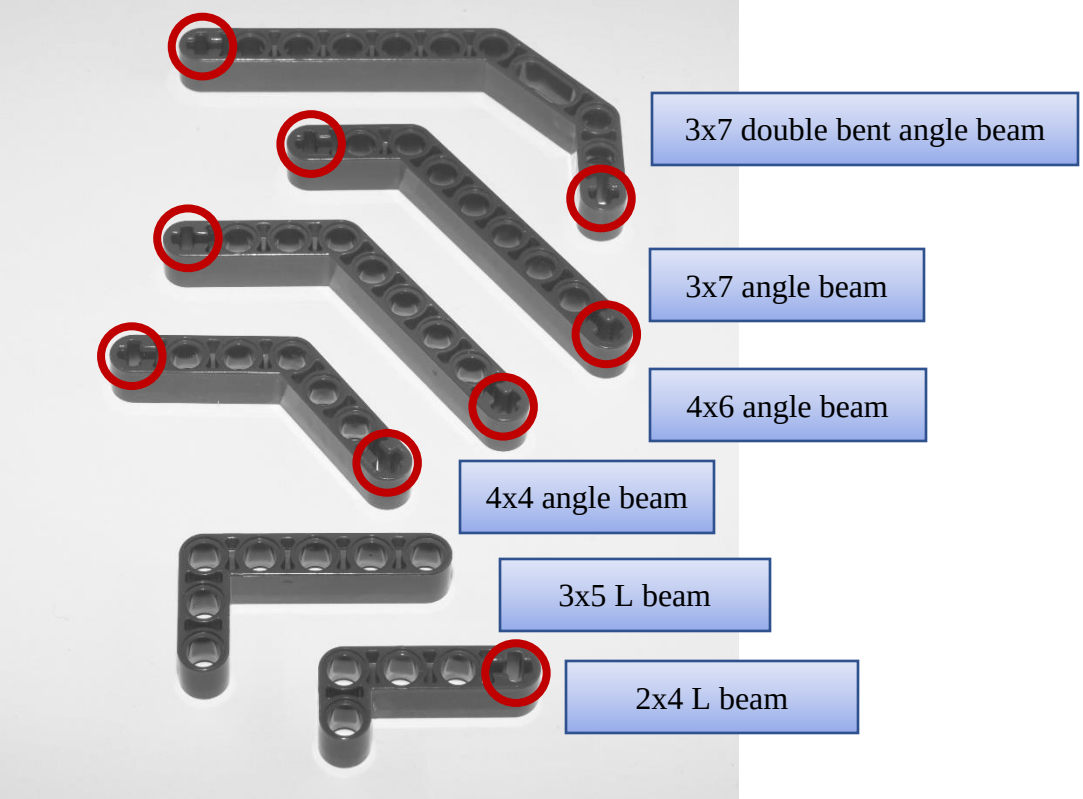


Figure 6. LEGO Studless L Beams and Angle Beams (with Axle Holes Circled)

This section has been redacted from this abridged version of the text, but is available at https://communities.sas.com/t5/SAS-Communities-Library/Data-Driven-Robotics-Leveraging-SAS-and-Python-Software-to/ta-p/641990.

CONTROL DATA INTEROPERABILITY WITH PYTHON

Each of the control tables referenced previously (including the Gears.csv file, 9x9 matrix, and various LEGO beam files) is saved in a canonical file format—the CSV file—that promotes interoperability across systems, applications, and programming languages. This data-driven design promotes interoperability, which facilitates MDM principles because only one version of each control table must be maintained. Thus, both the SAS and Python instances of software can rely on the same underlying tables, and if a control table must be modified, both instances of software remain synchronized with these updated control data. Moreover, both the SAS and Python programs rely identical parameters (and arguments), as previously described.

The Python program does not generate the gears or gear train analysis reports, but rather creates a list of lists (reports_list) that developers can further exploit for analysis or reporting purposes. Rather, the output produced lists the total number of theoretical gear train permutations evaluated and the number of acceptable gear trains that were discovered. The program follows:

```
import csv
import math
```

```
from operator import itemgetter

path='C:/Users/Juan Carlos/cosas/SAS/SASUniversityEdition/myfolders/legos/'
gears='gears.csv'


def create_rectangle(csvfil, width, height):
    with open(csvfil, 'w', newline='') as f:
        fnames=['x','y']
        writer=csv.DictWriter(f,fieldnames=fnames)
        for y in range(0,height):
            for x in range(0,width):
                writer.writerow({'x':x*8, 'y':y*8})


create_rectangle(csvfil=path+'coords_rectangle_9x9.csv',width=9, height=9)


def eval_coords(gearfile, beamfile, fuzz, hole, verbosity, rptpath, rptfile):
    # 1) type, 2) teeth, 3) shaft, 4) partNO, 5) years,
    # 6) diam, 7) thick, 8) 180, 9) 90
    counter=0
    results_list=[]
    results_counter=0
    with open(gearfile) as gears:
        reader=csv.reader(gears)
        temp_list=list(reader)
        gear_list=[]
        gear_set=set()
        for gear in temp_list:
            if gear[0].lower().split()[0] in ('spur','double','crown','turntable'):
                if verbosity.lower() == 'verbose':
                    gear_list.append(gear)
                else:
                    if gear[1] not in gear_set:
                        gear_set.add(gear[1])
                        gear_list.append(gear)
    with open(beamfile) as holes:
        reader=csv.reader(holes)
        hole_list=list(reader)
    for hole_1st in hole_list:
        for hole_2nd in hole_list:
            for gear_1st in gear_list:
                gear1=gear_1st[1] + '-tooth'
                if verbosity.lower() == 'verbose':
                    gear1=gear1 + ' ' + gear_1st[0]
                    if gear_1st[2].lower() != 'axle':
                        gear1=gear1 + ' (' + gear_1st[2] + ')'
                for gear_2nd in gear_list:
                    gear2=gear_2nd[1] + '-tooth'
                    if verbosity.lower() == 'verbose':
                        gear2=gear2 + ' ' + gear_2nd[0]
                        if gear_2nd[2].lower() != 'axle':
                            gear2=gear2 + ' (' + gear_2nd[2] + ')'
                    gear_distance=(int(gear_1st[5])/2) + (int(gear_2nd[5])/2)
                    stud_distance=math.sqrt(((float(hole_2nd[0])-
                                              float(hole_1st[0]))**2)
                                            + ((float(hole_2nd[1])-
                                                float(hole_1st[1]))**2))
                    gap_overlap=round(stud_distance-gear_distance,3)
                    if not(hole_1st[0] == hole_2nd[0] \
                           and hole_1st[1] == hole_2nd[1]):
                        counter += 1
```

```
                    if gap_overlap >= (0 - fuzz) and gap_overlap <= fuzz:
                        if hole_1st[0] == hole_2nd[0]:
                            theta=90.0
                        else:
                            theta=round(math.degrees(math.atan((
                                float(hole_2nd[1])-float(hole_1st[1]))
                                /(float(hole_2nd[0])-float(hole_1st[0])))),1)
                        if gear_1st[1] == gear_2nd[1]:
                            ratio='1:1'
                        elif int(gear_1st[1]) < int(gear_2nd[1]):
                            ratio=str(round(float(gear_2nd[1]) /
                                            float(gear_1st[1]),3)) + ':1'
                        else:
                            ratio='1:' + str(round(float(gear_1st[1])
                                                   / float(gear_2nd[1]),3))
                        results_list.append([gear1,float(hole_1st[0])/8,
                                             float(hole_1st[1])/8,
                                             gear2,float(hole_2nd[0])/8,
                                             float(hole_2nd[1])/8,
                                             theta,gap_overlap,ratio])
                        results_counter+=1
            if hole.lower() == 'first':
                break
    print('Permutations: %s   Gear Matches: %s' %(counter, results_counter))
```

A sample invocation of the program follows, which recreates the 9x9 matrix demonstrated in Table 3:

```
eval_coords(gearfile=path+gears,
            beamfile=path+'coords_rectangle_9x9.csv',
            fuzz=1,
            hole='FIRST',
            verbosity='BRIEF',
            rptpath=path,
            rptfile='FILE.html')
```

Note that as no reports are generated by the Python program, the RPTPATH and RPTFILE parameters are functionless placeholders only. Notwithstanding, the Results_list list of lists contains the data represented in Table 3 and can be utilized for subsequent analysis or reporting.

CONCLUSION

Gear trains are essential within robotics and other machinery and facilitate the transfer of power as well as the increase or decrease of speed or torque. This text examined LEGO gear trains that can be constructed virtually by meshing two LEGO gears inline within a 9x9 matrix of axle holes. A comprehensive list of all LEGO gear trains, including gear type, number of teeth, position, angle, and gear ratio is included and should be used as a reference by LEGO builders. Moreover, the dimensions of additional LEGO beams are provided in separate CSV files that can be evaluated to assess where gear trains should be positioned, and builders can evaluate other LEGO beams by creating CSV files with beam dimensions. This data-driven software design maximizes software flexibility and configurability because only control tables and parameters—rather than code—must be modified by end users to produce these dynamic results. Finally, data-driven design also facilitates control data interoperability (and master data management) because equivalent SAS and Python instances of the software can rely on the same underlying control tables and parameters.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Troy Martin Hughes

E-mail: troymartinhughes@gmail.com

REFERENCES

Hughes, T. M. (2019). *SAS® Data-Driven Development: From Abstract Design to Dynamic Functionality.* San Diego, California: CreateSpace.

APPENDIX A. LEGO_GEAR_COMBINATIONS.SAS

```
/* print report of all gears and their attributes */
%macro gear_report(csvfile= /* path+file+ext */,
   rptpath= /* path for report */,
   rptfile= /* file+ext */);
data gear_report;
   infile "&gearfile" truncover dsd firstobs=2 delimiter=',';
   length  type $50 teeth 8 hole $20 partNO $10 years $10
      diameter 8 thickness 8 deg180 $3 deg90 $3;
   input type : $50. teeth : 8. hole : $20. partNO : $10.
      years : $10. diameter : 8. thickness : 8.
      deg180 : $3. deg90 : $3.;
   label type='Gear Type' teeth='Teeth' hole='Shaft Type'
      partNO='Part No' years='Release Years'
      diameter='Diameter (mm)' thickness='Thickness (mm)'
      deg180='Inline Mesh' deg90='90-Degree Mesh';
run;
ods html path="&rptpath" file="&rptfile";
proc report data=gear_report nocenter nowindows nocompletecols
      style(report)=[foreground=black backgroundcolor=white
      background=black] style(header)=[font_size=2 background=black
      backgroundcolor=black foreground=white]
      style(column)=[backgroundcolor=very light grey];
   column obs type teeth hole partNO years diameter thickness deg180 deg90;
   define obs / computed '#';
   compute obs;
      obs_pvt+1;
      obs=obs_pvt;
      call define("_c1_",'style','style=[backgroundcolor=black
         foreground=white]');
      endcomp;
   compute partNO / char length=100;
      partNOurl='https://www.bricklink.com/v2/catalog/catalogitem.page?P='||partNo;
      call define(_col_,'URL',partNOurl);
      endcomp;
run;
ods html close;
%mend;


/* creates a rectangular matrix of stud holes (in mm)
as measured from upper-left stud center in 8mm increments
[24,8] corresponds to a hole 3 that is studs to the right
and 1 stud down from the [0,0] origin */
%macro create_rectangle(csvfile= /* path+file+ext */,
   width= /* (in studs) of simulated beam */,
   height= /* (in studs) of simulated beam */);
data _null_;
   file "&csvfile" dsd dlm=',';
   length x y 3;
   do y=0 to (&height-1)*8 by 8;
      do x=0 to (&width-1)*8 by 8;
         put x y;
         end;
      end;
run;
%mend;


%macro gear_train_report(rptpath= /* report path */,
   rptfile= /* report file name+ext */,
```

```
   perm= /* calculated # of permutations */);
ods html path="&rptpath" file="&rptfile";
title "&rptfile";
title2 "Permutations Tested: &perm";
proc report data=matching_coords nocenter nowindows nocompletecols
      style(report)=[foreground=black backgroundcolor=white
      background=black] style(header)=[font_size=2 background=black
      backgroundcolor=black foreground=white]
      style(column)=[backgroundcolor=very light grey];
   column obs gear1 xstud_start ystud_start gear2 xstud ystud theta gap_overlap
ratio;
   define obs / computed '#';
   define gear1 / group order=data;
   define xstud_start / display 'Gear 1 (X)';
   define ystud_start / display 'Gear 1 (Y)';
   define gear2 / display;
   define xstud / display 'Gear 2 (X)';
   define ystud / display 'Gear 2 (Y)';
   define theta / display;
   define gap_overlap / display;
   define ratio / display;
   compute obs;
      obs_pvt+1;
      obs=obs_pvt;
      call define("_c1_",'style','style=[backgroundcolor=black foreground=white]');
   endcomp;
run;
ods html close;
%mend;


/* Gears.csv has a header row and must have the following columns:
- Gear type (which includes version parenthetically)
- Number teeth (excluded for knob wheels and worm screws)
- Hole Type (e.g., axle, frictionless, pinhole, N/A)
- LEGO Part Number
- Range of part production years (YYYY-YYYY)
- Diameter (in mm)
- Thickness (in mm)
- Inline meshing (yes - if the gear works inline like spur gears)
- Perpendicular meshing (yes - if the gear can round a corner) */
%macro eval_coords(gearfile= /* gears CSV file */,
   beamfile= /* beams CSV file of axle hole coords */,
   rptpath= /* path for HTML gear train analysis */,
   rptfile= /* file name+ extension for analysis */,
   hole=FIRST /* FIRST to test one hole, or ALL for all */,
   fuzz= /* distance (mm) gear radii can differ from hypotenuse */,
   verbosity=verbose /* VERBOSE or BRIEF for gear distinction */);
* evaluate available gears;
%local nobs holes x y;
data gears;
   infile "&gearfile" truncover dsd firstobs=2 delimiter=',';
   length  type $50 teeth 8 hole $20 partNO $10 years $10
      diameter 8 thickness 8 deg180 $3 deg90 $3;
   input type : $50. teeth : 8. hole : $20. partNO : $10.
      years : $10. diameter : 8. thickness : 8.
      deg180 : $3. deg90 : $3.;
   if lowcase(type)=:'spur' or lowcase(type)=:'double bevel' or
      lowcase(type)=:'turntable' or lowcase(type)=:'crown' then output;
run;
%if "%lowcase(&verbosity)"="verbose" %then %do;
   proc sort data=gears;
      by teeth type hole;
```

```
   run;
   %end;
%else %do;
   proc sort data=gears nodupkey;
      by teeth;
   run;
   %end;
data _null_;
   set gears end=eof;
   if eof then call symputx('nobs',strip(put(_n_,8.)),'l');
run;
* ingest beam structure that houses gears in pinholes;
data holes1 (rename=(xmm=xmm_start ymm=ymm_start xstud=xstud_start
ystud=ystud_start)) holes2; * two data sets so join can be performed;
   length  xmm ymm 8;
   infile "&beamfile" truncover dsd firstobs=1 delimiter=',' end=eof;
   input xmm : 8. ymm : 8.;
   xstud=xmm/8;
   ystud=ymm/8;
   format xmm ymm xstud ystud best8.3;
   if eof then call symputx('holes',put(_n_,8.),'l');
run;
%put NOBS: &nobs;
data matching_coords (drop=counter i j pi type teeth diameter thickness deg180
deg90 hole partNO years);
   pi=constant("pi");
   if _n_=1 then do;
      length i counter 8;
      retain counter 0;
      i=0;
      do until(eof1);
         set gears end=eof1;
         i=i+1;
         array arrtype[&nobs] $50 _temporary_;
         arrtype[i]=strip(type);
         array arrteeth[&nobs] 8 _temporary_;
         arrteeth[i]=teeth;
         array arrhole[&nobs] $20 _temporary_;
         arrhole[i]=hole;
         array arrpartNO[&nobs] $10 _temporary_;
         arrpartNO[i]=partNO;
         array arryears[&nobs] $10 _temporary_;
         arryears[i]=years;
         array arrdiam[&nobs] 8 _temporary_;
         arrdiam[i]=diameter;
         array arrthick[&nobs] 8 _temporary_;
         arrthick[i]=thickness;
         array arrdeg180[&nobs] $3 _temporary_;
         arrdeg180[i]=deg180;
         array arrdeg90[&nobs] $3 _temporary_;
         arrdeg90[i]=deg90;
         end;
   end;
   length gear1 $50 gear2 $50 gear_distance stud_distance theta gap_overlap 8
      ratio $8;
   format gear1 $50. gear2 $50. gear_distance 8.3 stud_distance 8.3 theta 8.1
      xstud best8.3 ystud best8.3 xmm best8.3 ymm best8.3 gap_overlap 8.3
      ratio $8.;
   label gear1='Primary Gear' gear2='Secondary Gear'
      gear_distance='Proposed distance (mm) between gear axles'
      stud_distance='Actual distance (mm) between gear axles'
      theta='Angle (degrees) between gears'
      xstud='Studs right' ystud='Studs down' xmm='Studs right' ymm='Studs down'
```

```
      gap_overlap='Gap (+) or Overlap (-) in mm'
      ratio='Gear Ratio';
   set holes2 end=eof2;
%if "&hole"="first" %then %do;
   do p=1 to 1;
   %end;
%else %do;
   do p=1 to &holes;
   %end;
      set holes1 point=p;
      do i=1 to dim(arrtype); * loop through primary gears;
         gear1=put(arrteeth[i],8.)||'-tooth';
         %if "%lowcase(&verbosity)"="verbose" %then %do;
            gear1=catx(' ',gear1,arrtype[i],ifc(lowcase(arrhole[i])^=
               'axle','('||strip(arrhole[i])||')',''));
            %end;
         do j=1 to dim(arrtype); * loop through secondary gears;
            gear2=put(arrteeth[j],8.)||'-tooth';
            %if "%lowcase(&verbosity)"="verbose" %then %do;
               gear2=catx(' ',gear2,arrtype[j],ifc(lowcase(arrhole[j])^=
                  'axle','('||strip(arrhole[j])||')',''));
               %end;
            gear_distance=arrdiam[i]/2 + arrdiam[j]/2; * add radii;
            stud_distance=sqrt((xmm_start-xmm)**2 + (ymm_start-ymm)**2); * hyp;
            gap_overlap=stud_distance-gear_distance;
            if not(xmm=xmm_start and ymm=ymm_start) then counter+1;
            if gap_overlap >= (0-&fuzz) and gap_overlap <= &fuzz then do;
               if xmm=xmm_start then theta=90;
               else theta=atan((ymm-ymm_start)/(xmm-xmm_start))*180/pi;
               if arrteeth[i]=arrteeth[j] then ratio='1:1';
               else if arrteeth[i]<arrteeth[j] then
                  ratio=substr(strip(put(arrteeth[j]/arrteeth[i],best8.3)),1,5)
                     || ':1';
               else ratio='1:' ||
                  substr(strip(put(arrteeth[i]/arrteeth[j],best8.3)),1,5);
               output;
               end;
            end;
         end;
      end;
   if eof2 then do;
      call symputx('counter',counter,'g');
      put counter;
      end;
run;
%put PERMUTATIONS: &counter;
%gear_train_report(rptpath=&rptpath, rptfile=&rptfile, perm=&counter);
%mend;
```