Tutorial 49: Generation of non circular gear pairs by Yannick Krieger/Sebastian Baumgartner

2020-09-30: Yannick Krieger/Sebastian Baumgartner, at Pro. Lueth's Lab Technische Universität München, Germany (URL: http://www.SG-Lib.org) - Last Change: 2020-09-30

Contents

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 5.0 required)

This tutorial explains the functions for the design of non-circular gear pairs. The functions are based on the semester thesis of Sebastian Baumgartner from 2020. Starting with several simple examples explaining the use of the four functions for the design of non-circular gears, the speed control of a four-bar linkage is shown at the end.This tutorial explains the functions for the design of non-circular gear pairs.

2. List of functions introduced in this tutorial

- PLNonCircPitchCurve - Design of two pitch curves that roll without slipping and hava a constant center distance
- PLNonCircGear - Generates the 2D- geometry of one non circular gear
- SGNonCircGear - Generates a solid geometry of two matching non circular gears
- SpeedControlFourBar_NonCircGear - Calculates non circular pitch curves to control a four bar mechanism

3. Process of generating non circular gears

1. Calculation of non circular pitch curves with the function *PLNonCircPitchCurve* or *SpeedControlFourBar_NonCircGear*
2. Generating SG of two matching non circular gears by plugging in the pitch curves to *SGNonCircGear(PLPitchCurve1,PLPitchCurve2,...)*

Remark: The function *PLNonCircGear* is used in the function *SGNonCircGear* and is only called manually if the geometry of one single non circular gear is to be generated by hand (not recommended).

4. Design of two matching non circular pitch curves

The function *PLNonCircPitchCurve()* offers four options to define non circular pitch curves, that roll without slipping and have a constant center distance. With the first input argument the option is selected.

PLNonCircPitchCurve(Option,...)

1. Option = 'v+v+t': Definition by angular velocities over time
2. Option = 'a+a+t': Definition by angular positions over time
3. Option = 'tr+a': Definition by transmission ratio over the angular position of the drive gear
4. Option = 'r+a': Definition by polar coordinates of one non circular pitch curve

Output results of *[PLPitchCurve1, PLPitchCurve2, E, phi1, phi2, r1, r2, w1, w2, t] = PLNonCircPitchCurve(...)*

- **PLPitchCurve1**: Point list of pitch curve 1 in cartesian coordinates.
- **PLPitchCurve2**: Point list of pitch curve 2 in cartesian coordinates.
- **E**: Center distance
- **phi1**: Angular position of first pitch curve (drive)
- **phi2**: Angular position of second pitch curve (driven)
- **r1**: Radius of the first pitch curve (drive)
- **r2**: Radius of the second pitch curve (driven)
- **w1**: Angular velocity of the first pitch curve (drive)
- **w2**: Angular velocity of the second pitch curve (driven)
- **t**: Time at which the position or velocity values are given

4.1 Option = 'v+v+t': Two matching non circular pitch curves by definition of the angular velocities over time

Inputs of *PLNonCircPitchCurve('v+v+t',w1,w2,t,E)*:

- **'v+v+t'**: Selecting angular velocities over time as input

- **w1**: ANGULAR VELOCITY of first pitch curve (drive) as a collumn vector with values at given timesteps. The last entry must be the same as the first. If the velocitiy does not result in one complete revolution of 2*pi it is changed during calculation.
- **w2**: ANGULAR VELOCITY of second pitch curve (driven) as a collumn vector with values at given timesteps. The last entry must be the same as the first. If the velocitiy does not result in one complete revolution of 2*pi it is changed during calculation.
- **t**: TIME (collumn vector) at which the velocity values are given (default: t = linspace(0,2*pi/mean(w2),length(w2))')
- **E**: CENTER DISTANCE between pitch curve 1 and 2 as a scalar value. (default: E = 50)

Remark: The cycle time of the system is given by T = t(end)-t(1);

Example 1: The angular velocities *w1* and *w2* of the gears are defined by a function of time *t* . Default for center distance is used.

Time vector:

```
t = linspace(0,10,361)';
```

Angular velocitiy of the drive gear (constant and results in a revolution of 2π)

```
w1 = 2*pi/10*ones(1,361)';
```

Angular velocity of the driven gear (sine function and results in a revolution of 2π)

```
w2 = 1/2*sin(t*2*pi/10)+2*pi/10;
```

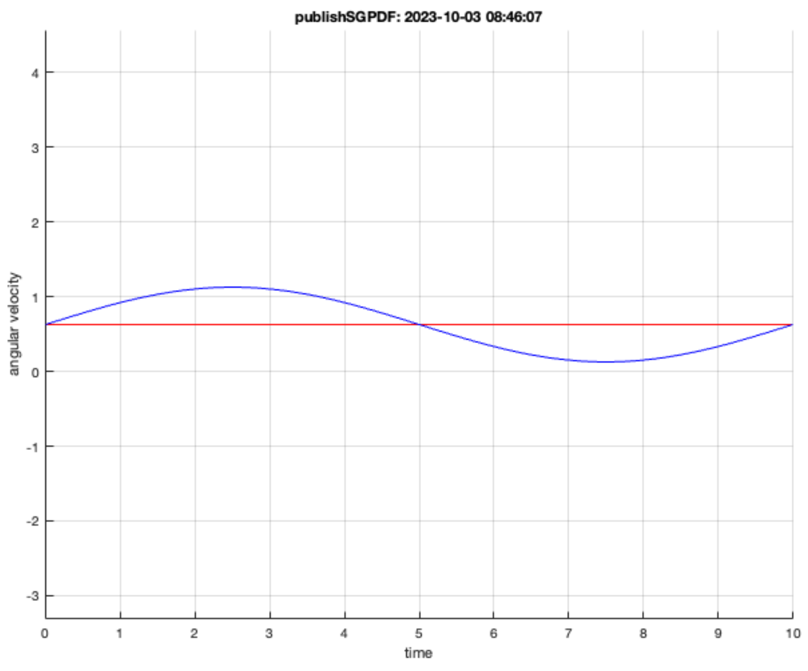
Calculation of the pitch curves that transform the angular velocity of the drive (w1) to the angular velocity of the output (w2) when the curves roll against each other:

```
[PLPitchCurve1,PLPitchCurve2,E] = PLNonCircPitchCurve('v+v+t',w1,w2,t);
```

```
Warning: PLNonCircPitchCurve(): No input for center distance. Choosen default
value E = 50
```

Remark: The generated pitch curves are both orientated in a way that the center of rotation is at [0,0] and the contact point for rolling against each other lies on the positive x-axis. Therefor the second pitch curve is mirrored on the y-axis and shifted in positive x-direction by the center distance.

```
SGfigure; PLplot(PLPitchCurve1,'-r'); PLplot(PLPitchCurve2.*[-1,1]+[E,0],'-b'); PLplot([0 0; E 0;],'oblack'); view([0 0 1]); axis equal; drawnowvid(30)
SGfigure; plot(t,w1,'r'); hold on; plot(t,w2,'b'); xlabel('time'); ylabel('angular velocity'); grid on; drawnowvid(30);
```



Example 2: The angular velocities *w1* and *w2* of the gears are given at few discrete time steps. Default for center distance is used.

Time vector:

```
t_in = [0;1;2;3;4];
```

Angular velocitiy of the drive gear (constant and results in a revolution of 2π)

```
w1_in = 2*pi/t_in(end)*[1;1;1;1;1];
```

Angular velocity of the driven gear (piecewise constant and results in a revolution of 2π)

```
w2_in = 2*pi/t_in(end)*[4/3; 4/3; 2/3; 2/3; 4/3];
```

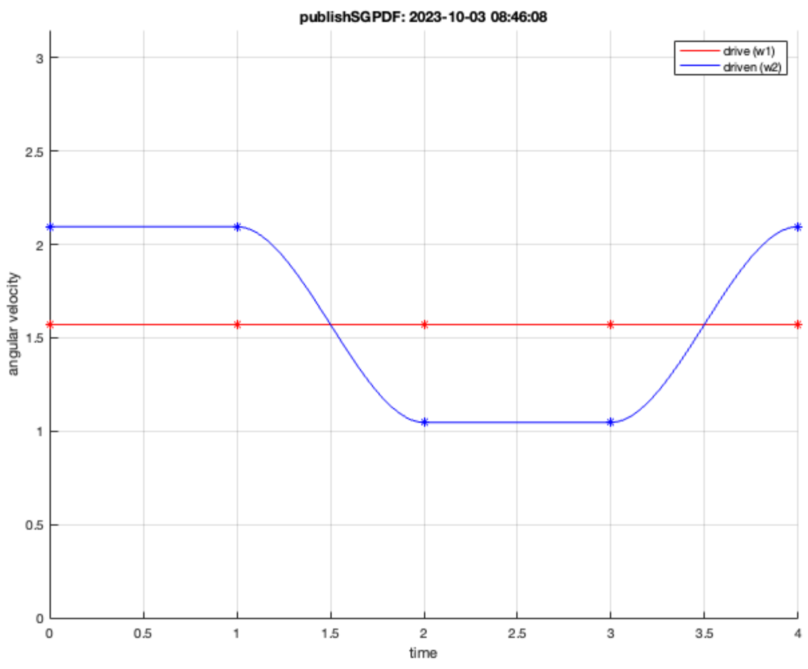
Calculation of the pitch curves that transform the angular velocity of the drive (w1) to the angular velocity of the output (w2) when the curves roll against each other:

```
[PLPitchCurve1,PLPitchCurve2,E,phi1,phi2,r1,r2,w1_out,w2_out,t_out] = PLNonCircPitchCurve('v+v+t',w1_in,w2_in,t_in);
```

```
Warning: PLNonCircPitchCurve(): No input for center distance. Choosen default
value E = 50
Warning: PLNonCircPitchCurve(): Too few discretization points of the input
vectors. The vectors are interpolated with 361 steps
```

Remark: The generated pitch curves are both orientated in a way that the center of rotation is at [0,0] and the contact point for rolling against each other lies on the positive x-axis. Therefor the second pitch curve is mirrored on the y-axis and shifted in positive x-direction by the center distance.

```
SGfigure; PLplot(PLPitchCurve1,'-r'); PLplot(PLPitchCurve2.*[-1,1]+[E,0],'-b'); PLplot([0 0; E 0;],'oblack'); view([0 0 1]); axis equal; drawnowvid(30)
SGfigure; hold on; plot(t_out,w1_out,'r'); plot(t_out,w2_out,'b'); plot(t_in,w1_in,'r*'); plot(t_in,w2_in,'b*'); xlabel('time'); ylabel('angular veloci
```



4.2 Option = 'a+a+t': Two matching non circular pitch curves by definition of the angular positions over time

Inputs of *PLNonCircPitchCurve('a+a+t',phi1,phi2,t,E)*:

- **'a+a+t'**: Selecting angular postions over time as input
- **phi1**: ANGULAR POSITION of first pitch curve (drive) as a collumn vector with values at given timesteps. It should increase steadily from 0 to 2π.
- **phi2**: ANGULAR POSITION of second pitch curve (driven) as a collumn vector with values at given timesteps. It should increase steadily from 0 to 2π.
- **t**: TIME (collumn vector) at which the velocity values are given (default: t = linspace(0,length(phi1)-1,length(phi1))' --> results in time steps with distance 1)
- **E**: CENTER DISTANCE between pitch curve 1 and 2 as a scalar value. (default: E = 50)

Remark: The cycle time of the system is given by T = t(end)-t(1);

Example 3: The angular positions of the driven gear (*phi2*) is defined by a function of the angular positions of the drive (*phi1*). The default value for the time vector and center distance is used.

Angular position of the drive gear (discrete points between 0 and 2π increasing steadily)

```
phi1_in = [0;2;4.7;2*pi];
```

Angular position of the driven gear (discrete points between 0 and 2π increasing steadily)

```
phi2_in = [0;3;4;2*pi];
```

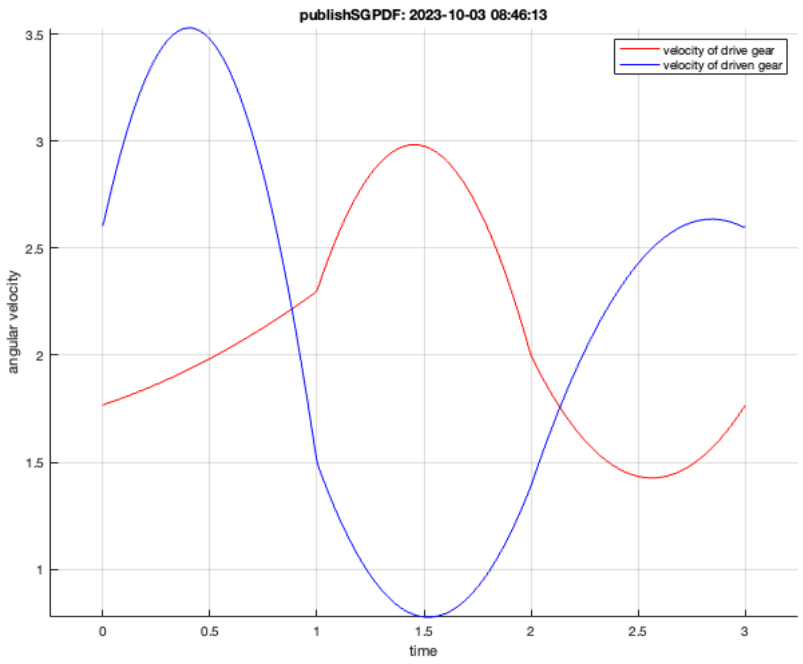
Calculation of the pitch curves which provide that the desired angle positions at the output (*phi2*) are reached at the given drive angles (*phi1*) when the curves roll against each other:

```
[PLPitchCurve1,PLPitchCurve2,E,phi1_out,phi2_out,r1,r2,w1,w2,t] = PLNonCircPitchCurve('a+a+t',phi1_in,phi2_in);
```

```
Warning: PLNonCircPitchCurve(): No input for time. Choosen default value: t =
linspace(0,3,4)'
Warning: PLNonCircPitchCurve(): No input for center distance. Choosen default
value E = 50
Warning: PLNonCircPitchCurve(): Too few discretization points of the input
vectors. The vectors are interpolated with 361 steps
```

Remark: The generated pitch curves are both orientated in a way that the center of rotation is at [0,0] and the contact point for rolling against each other lies on the positive x-axis. Therefor the second pitch curve is mirrored on the y-axis and shifted in positive x-direction by the center distance.

```
SGfigure; PLplot(PLPitchCurve1,'-r'); PLplot(PLPitchCurve2.*[-1,1]+[E,0],'-b'); PLplot([0 0; E 0;],'oblack'); view([0 0 1]); axis equal; ; drawnowvid(3
SGfigure; plot(phi1_in,phi2_in,'r*'); hold on; plot(phi1_out,phi2_out,'r'); xlabel('angular position of the drive gear (phi1)'); ylabel('angular positi
SGfigure; plot(t,w1,'r'); hold on; plot(t,w2,'b'); xlabel('time'); ylabel('angular velocity'); legend('velocity of drive gear','velocity of driven gear
```



4.3 Option = 'tr+a': Two matching non circular pitch curves by definition of the transmission ratio over the angular position of the drive gear

Inputs of *PLNonCircPitchCurve('tr+a',tr,phi1,E)*:

- **'tr+a'**: Selecting transmission ratio over drive angle as input
- **tr**: TRANSMISSION RATIO from angular velocity 1 to 2. tr = w1/w2 = r2/r1 as a collumn vector at given drive angel positions. If the transmission ratio does not result in one complete revolution of the second angle (phi2) it is scaled so the mean inverse transmission ratio is mean(1/tr)=1.
- **phi1**: ANGULAR POSITION of first pitch curve (drive) as a collumn vector. It should increase steadily from 0 to 2π. (default: phi1 = linspace(0,2*pi,length(tr))')
- **E**: CENTER DISTANCE between pitch curve 1 and 2 as a scalar value. (default: E = 50)

Example 4: The transmission ratio *tr* is defined by a sine function. Default value of the drive angle *phi1* is used. Center distance *E = 30* ;

Transmission ratio (sine function):

```
tr = sin(4*linspace(0,2*pi,500)')+2;
```

Calculation of the pitch curves that transform the motion of the drive through the transmission ratio tr to the driven output when the curves roll against each other:

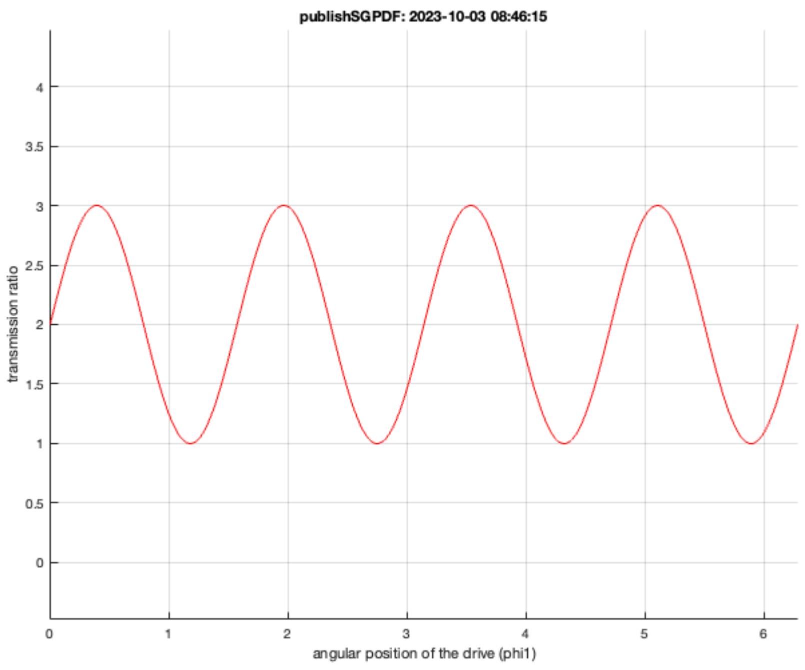
```
E = 30;
[PLPitchCurve1,PLPitchCurve2,E,phi1,phi2] = PLNonCircPitchCurve('tr+a',tr,'',E);
```

```
Warning: PLNonCircPitchCurve(): No input for first angle. Choosen default
value: phi1 = linspace(0,2*pi,500)'
```

```
Warning: PLNonCircPitchCurve(): Input of transmission ratio does not result in
one complete revolution (2*pi) of the second angle. It is multiplied by the
factor 5.773503e-01, so the mean inverse transmission ratio is mean(1/tr)=1
```

Remark: The generated pitch curves are both orientated in a way that the center of rotation is at [0,0] and the contact point for rolling against each other lies on the positive x-axis. Therefor the second pitch curve is mirrored on the y-axis and shifted in positive x-direction by the center distance.

```
SGfigure; PLplot(PLPitchCurve1,'-r'); PLplot(PLPitchCurve2.*[-1,1]+[E,0],'-b'); PLplot([0 0; E 0;],'oblack'); view([0 0 1]); axis equal; ; drawnowvid(3
SGfigure; plot(phi1,tr,'r'); xlabel('angular position of the drive (phi1)'); ylabel('transmission ratio'); grid on; ; drawnowvid(30);
```



4.4 Option = 'r+a': Design of two matching non circular pitch curves by giving the polar coordinates of one non circular pitch curve

Inputs of *PLNonCircPitchCurve('r+a',r1,phi1)*:

- **'r+a'**: Selecting polar coordinates of one non circular pitch curve as input
- **r1**: RADIUS of the first pitch curve (drive) at given angles *phi1* as a collumn vector.
- **phi1**: ANGULAR POSITION of first pitch curve (drive) as a collumn vector with values at given radii *r1* . It should increase steadily from 0 to 2π.

Remark: The center distance can not be defined explicitly. It is already defined by giving the shape of the first ptich curve.

Example 5: The radius r1 of the first pitch curve is defined by a sine function of the polar angle of this curve. The default value of the center distance is used.

Angular position of the first pitch curve (drive) (linear from 0 to 2π)

```
phi1 = linspace(0,2*pi,361)';
```

Radius of the first pitch curve (drive) (sine function shifted to the positive)

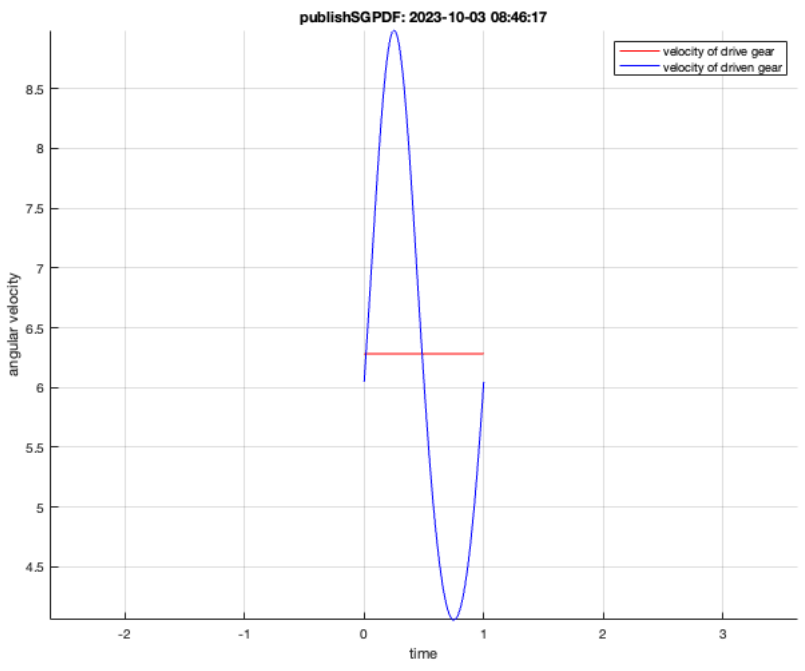
```
r1 = 50+10*sin(phi1);
```

Calculation of the second pitch curve that rolls on the given curve (*phi1,r1*) with out slipping at a constant center distance:

```
[PLPitchCurve1,PLPitchCurve2,E,phi1,phi2,r1,r2,w1,w2,t] = PLNonCircPitchCurve('r+a',r1,phi1);
```

Remark: The generated pitch curves are both orientated in a way that the center of rotation is at [0,0] and the contact point for rolling against each other lies on the positive x-axis. Therefor the second pitch curve is mirrored on the y-axis and shifted in positive x-direction by the center distance.

```
SGfigure; PLplot(PLPitchCurve1,'-r'); PLplot(PLPitchCurve2.*[-1,1]+[E,0],'-b'); PLplot([0 0; E 0;],'oblack'); view([0 0 1]); axis equal; ; drawnowvid(3
SGfigure; plot(t,w1,'r'); hold on; plot(t,w2,'b'); xlabel('time'); ylabel('angular velocity'); legend('velocity of drive gear','velocity of driven gear
```



5 Generating solid geometry (SG) of two matching non circular gears

To generate the SG of two matching non circular gears the pitch curves calculated before (section 4 or 6) are inserted in the function *SGNonCircGear* .

Input parameters of *SGNonCircGear(PLPitchCurve1, PLPitchCurve2, thick, rh, z, arrow, chco, thrf, stcol, alt, visual, alpha, cc, hc)* :

- **PLPitchCurve1**: Point list of pitch curve 1 in cartesian coordinates.
- **PLPitchCurve2**: Point list of pitch curve 2 in cartesian coordinates.
- **thick**: Gear thickness as a scalar (default is integer of rounded 4*m)
- **rh**: Hole radius: A scalar value for both holes or [Bore 1 radius, Bore 2 radius] (default value is based on minimum rolling curve radius and module)
- **zm**: Number of teeth or module (default is zm = [1,2]):
- - as scalar: zm > 12: zm = Number of teeth / zm <=12: zm = Module
- - or vector: zm = [z,1]: z = Number of teeth / zm = [m,2]: m = Module
- - If the number of teeth is specified, the module depends on the circumference of the pitch curves. If the module is specified, it will be adapted so that an integer number of teeth is obtained.
- **arrow**: Adding an arrow to the 3D geometry of each gear, pointing to the starting point of the rolling curves, to simplify assembly.
- - Display arrow: 'arrow', 'on', 1 (deafault)
- - Do not display: Any character
- **chco**: Switching on/off the collision check for the rolling of the gears:
- - Switch on: 'Check Collision', 'on', 1 (default)
- - Switch off: Any character
- **thrf**: Tooth height reduction factor: Scalar > 0 (no tooth height reduction for thrf = 0) (default thrf = 1.8)
- **stcol**: Collision subtraction:
- - Switch on: 'Subtract Collision', 'on', 1 (default)
- - Switch off: Any character
- **alt** : Alternating the teeth in different planes:
- - Switch on: 'Alternate Teeth', 'on', 1
- - Switch off: Any character (default)
- **visual**:
- - 0 no visualization of calculation progress (deafault, if output is requested)
- - 1 visualization of gear at end of calculation
- - 2 visualization of gear after each cahnage of curvature (deafault, if no output is requested)
- - 3 visualization of whole calculation progress
- **alpha**: Pressure angle of the teeth as a scalar in degrees (default alpha = 20)
- **cc**: Head clearance factor of the tooth geometry as a scalar (default cc = 0.25)
- **hc**: Tooth height factor of the tooth geometry as a scalar (default hc = 1)

Output results of *[SGncgear1,SGncgear2] = SGNonCircGear(...)* :

- **SGncgear1.CPL/VL/FL/PL/EL**
- **SGncgear1.param**: Cell array with fields: {'center distance', 'hole radius', 'thickness', 'module', 'tooth number', 'alpha', 'tip clearance', 'addendum factor', 'collision'}
- **SGncgear2.CPL/VL/FL/PL/EL**
- **SGncgear2.param**: Cell array with fields: {'center distance', 'hole radius', 'thickness', 'module', 'tooth number', 'alpha', 'tip clearance', 'addendum factor', 'collision'}