Software Lab Computational Engineering Science

Group 12, Pusher Mechanism

Aaron Floerke, Arseniy Kholod, Xinyang Song and Yanliang Zhu

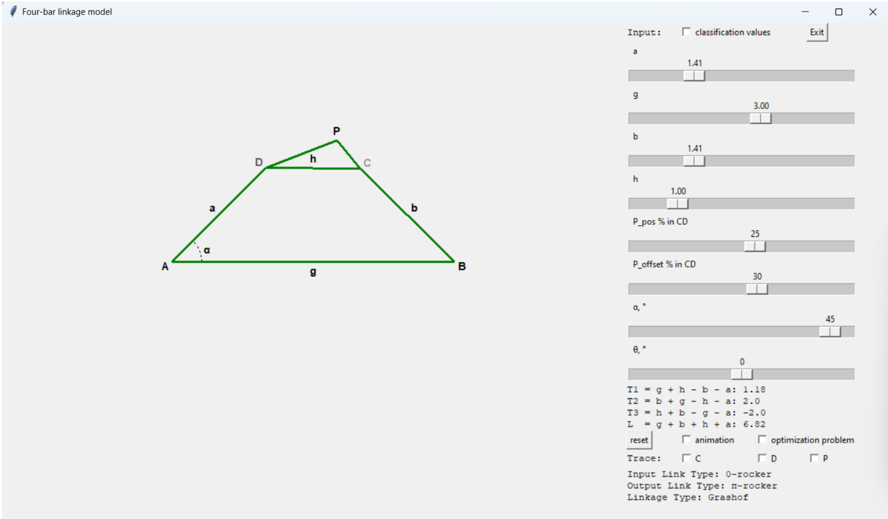
Informatik 12: Software and Tools for Computational Engineering (STCE)
RWTH Aachen University

Contents

- Preface
- Analysis
  - User Requirements
  - System Requirements
- Design
  - System Requirements
  - Class Model(s)
- Implementation
  - Development Infrastructure
  - Four-Bar Linkage Model
  - Software Tests
  - GUI
- Results
  - 27 movement types
  - Optimization problem
- Project Management
- Live Software Demo
- Summary and Conclusion

Preface

Four-bar linkage model



Input: □ classification values

a 1.41

g 3.00

b 1.41

h 1.00

P_pos % in CD 25

P_offset % in CD 30

α, ° 45

θ, ° 0

```
T1 = g + h - b - a: 1.18
T2 = b + g - h - a: 2.0
T3 = h + b - g - a: -2.0
L  = g + b + h + a: 6.82
```

□ animation □ optimization problem

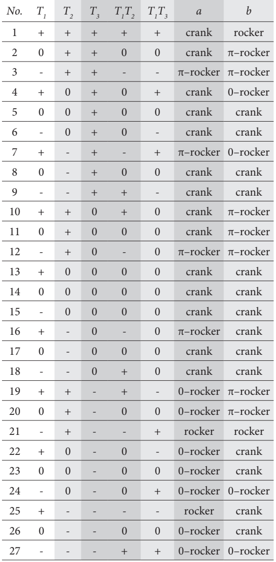
Trace: □ C □ D □ P

```
Input Link Type: 0-rocker
Output Link Type: π-rocker
Linkage Type: Grashof
```

Analysis

User Requirements

- Implement 27 motion types of the four-bar linkage with one bar fixed:
  - Classification values:
    - $T_1 = g + h - b - a$
    - $T_2 = b + g - h - a$
    - $T_3 = h + b - g - a$
- Implement GUI with motion animation and the ability to choose geometrical parameters:
  - Length of the bars
  - Position of the coupler
  - Input angle
  - Angle relative to the horizon
  - Classification values as alternative input

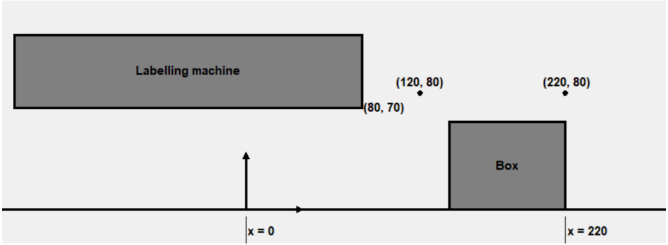


No.	T_1	T_2	T_3	T_1T_2	T_1T_3	a	b
1	+	+	+	+	+	crank	rocker
2	0	+	+	0	0	crank	π–rocker
3	-	+	+	-	-	π–rocker	π–rocker
4	+	0	+	0	+	crank	0–rocker
5	0	0	+	0	0	crank	crank
6	-	0	+	0	-	crank	crank
7	+	-	+	-	+	π–rocker	0–rocker
8	0	-	+	0	0	crank	crank
9	-	-	+	+	-	crank	crank
10	+	+	0	+	0	crank	π–rocker
11	0	+	0	0	0	crank	π–rocker
12	-	+	0	-	0	π–rocker	π–rocker
13	+	0	0	0	0	crank	crank
14	0	0	0	0	0	crank	crank
15	-	0	0	0	0	crank	crank
16	+	-	0	-	0	π–rocker	crank
17	0	-	0	0	0	crank	crank
18	-	-	0	+	0	crank	crank
19	+	+	-	+	-	0–rocker	π–rocker
20	0	+	-	0	0	0–rocker	π–rocker
21	-	+	-	-	+	rocker	rocker
22	+	0	-	0	-	0–rocker	crank
23	0	0	-	0	0	0–rocker	crank
24	-	0	-	0	+	0–rocker	0–rocker
25	+	-	-	-	-	rocker	crank
26	0	-	-	0	0	0–rocker	crank
27	-	-	-	+	+	0–rocker	0–rocker

Figure from "Classification, geometrical and kinematic analysis of four-bar linkages" 10.15308/Sinteza-2018-261-266 by Ivana Cvetkovic et al.

Analysis

User Requirements



- Solve an optimization problem:
  - Push box with size 80×60 from $x = 220$ to $x = 0$
  - Do not cross the area of the labelling machine (Area with $x < 80$ and $y > 70$).
  - Pass above points $(120, 80)$ and $(220, 80)$

System Requirements

Functional

- **Four-bar linkage model**:
  - System simulates all the motion types of the four-bar linkage.
  - System does not crash with any input of geometrical configuration.
- **Tests**:
  - Implement test cases for geometry.
  - Implement test cases with bad input to test system stability.
- **Graphical User Interface**:
  - GUI provides the four-bar linkage visualization and motion animation.
  - User can input geometrical data by moving a point on a slide bar.
  - GUI is coupled with the four-bar linkage model to use implemented motion cases for animation.
  - GUI provides tracing for trajectories of the points.
  - GUI classifies of the linkage.
- **Optimization problem**:
  - It should be possible to find a solution (manually) for the optimization problem using the four-bar linkage model.
  - GUI visualizes the solution.

System Requirements

Non-Functional

- **Performance**:
  - The four-bar linkage model is fast enough to provide smooth GUI animations.
  - GUI animations are not slower than 30 frames per second.
- **Usability**:
  - Every essential part of the four-bar linkage model is well documented.
  - GUI is easy to operate and all functionalities are self-explanatory.
  - GUI source code is well documented.

Design

Principal Components and Third-Party Software

Design

Class Model(s)

Implementation

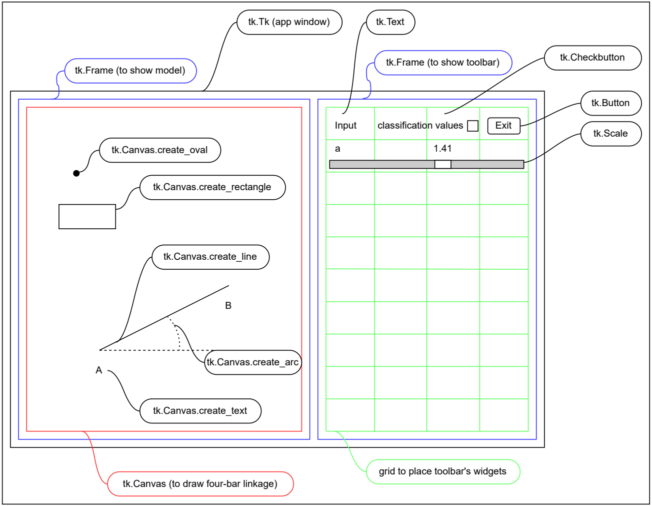
Development Infrastructure

Implementation

Four-Bar Linkage Model

Implementation

Software Tests



- Initiate all tkinter objects inside GUI class and generate app window:

  GUI().tk.mainloop()

Implementation

GUI, Animation

- Update objects in tk.Canvas every animation step using coords and/or itemconfigure for optimization

```
class GUI:
  def __init__(self):
    ...
    self.init_toolbar()
    ...
  def init_toolbar(self):
    ...
    self.enable_animation = tk.IntVar()
    self.animation_button = tk.Checkbutton(self.toolbar_frame, text="animation",
                                           variable=self.enable_animation,
                                           onvalue=1, offvalue=0, command=self.animation)
    self.animation_button.grid(sticky="W", row=10, column=2)
    ...
  def refresh(self):
    ...
    self.linkage.run()
    ...
    self.update_linkage_display()
  def animation(self):
    self.run_animation()
  def run_animation(self):
    if self.enable_animation.get():
      self.linkage.animation_alpha() # alpha = alpha + d_alpha
      self.refresh()
      self.tk.after(25, self.run_animation)
  def update_linkage_display(self):
    ...
    self.model_animation.coords(self.model_animation.AB_line, [A_x, A_y, B_x, B_y])
    ...
```

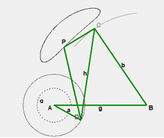
Implementation

GUI, Show and hide objects

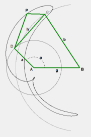
- To display different modes, some objects have to be hidden or shown.
- For objects in tk.Canvas use itemconfigure:
  - Hide:
    self.model_animation.itemconfigure(self.model_animation.AB_line, state=’hidden’)
  - Show:
    self.model_animation.itemconfigure(self.model_animation.AB_line, state=’normal’)
- For widgets like tk.Scale or tk.Text:
  - Hide: self.slider_T1.grid_remove()
  - Show: self.slider_T1.grid()

Results

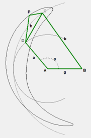
27 movement types



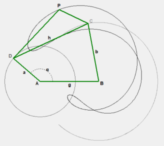
$T_{1,2,3} = 1.0, 1.0, 1.0$



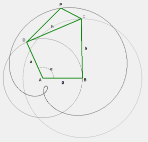
$T_{1,2,3} = 0.0, 1.0, 1.0$



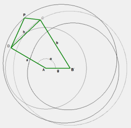
$T_{1,2,3} = -1.0, 1.0, 1.0$



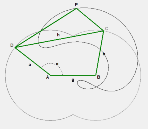
$T_{1,2,3} = 1.0, 0.0, 1.0$



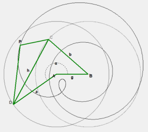
$T_{1,2,3} = 0.0, 0.0, 1.0$



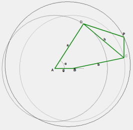
$T_{1,2,3} = -1.0, 0.0, 1.0$



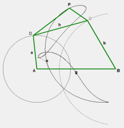
$T_{1,2,3} = 1.0, -1.0, 1.0$



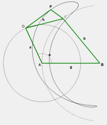
$T_{1,2,3} = 0.0, -1.0, 1.0$



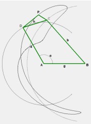
$T_{1,2,3} = -1.0, -1.0, 1.0$



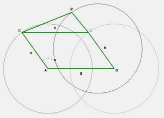
$T_{1,2,3} = 1.0, 1.0, 0.0$



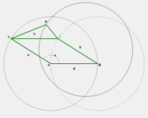
$T_{1,2,3} = 0.0, 1.0, 0.0$



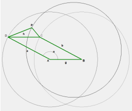
$T_{1,2,3} = -1.0, 1.0, 0.0$



$T_{1,2,3} = 1.0, 0.0, 0.0$



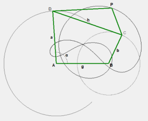
$T_{1,2,3} = 0.0, 0.0, 0.0$



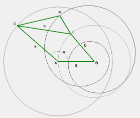
$T_{1,2,3} = -1.0, 0.0, 0.0$

Results

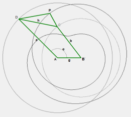
27 movement types



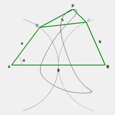
$T_{1,2,3} = 1.0, -1.0, 0.0$



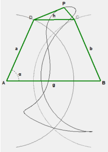
$T_{1,2,3} = 0.0, -1.0, 0.0$



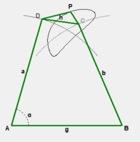
$T_{1,2,3} = -1.0, -1.0, 0.0$



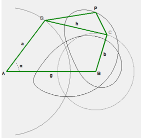
$T_{1,2,3} = 1.0, 1.0, -1.0$



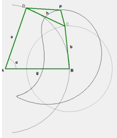
$T_{1,2,3} = 0.0, 1.0, -1.0$



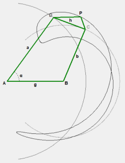
$T_{1,2,3} = -1.0, 1.0, -1.0$



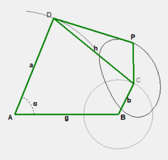
$T_{1,2,3} = 1.0, 0.0, -1.0$



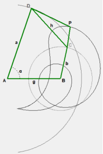
$T_{1,2,3} = 0.0, 0.0, -1.0$



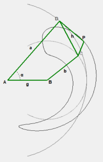
$T_{1,2,3} = -1.0, 0.0, -1.0$



$T_{1,2,3} = 1.0, -1.0, -1.0$



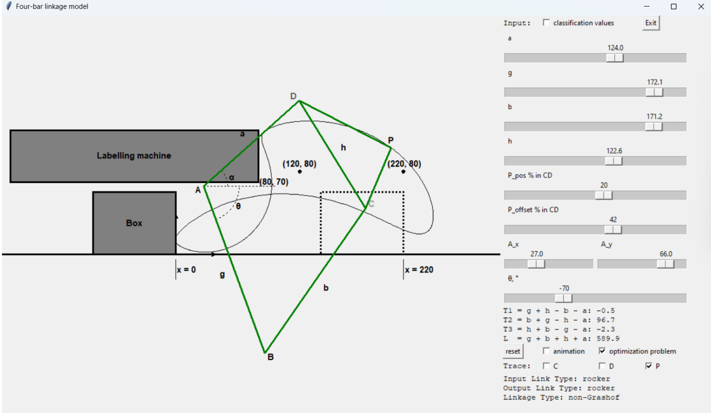
$T_{1,2,3} = 0.0, -1.0, -1.0$



$T_{1,2,3} = -1.0, -1.0, -1.0$

Results

Optimization problem



Input: □ classification values

a: 124.0

g: 172.1

b: 171.2

h: 122.6

P_pos % in CD: 20

P_offset % in CD: 42

A_x: 27.0

A_y: 66.0

θ, °: -70

```
T1 = g + h - b - a: -0.5
T2 = b + g - h - a: 96.7
T3 = h + b - g - a: -2.3
L  = g + b + h + a: 589.9
```

□ animation ☑ optimization problem

Trace: □ C □ D ☑ P

```
Input Link Type: rocker
Output Link Type: rocker
Linkage Type: non-Grashof
```

- 9 degrees of freedom (all lengths in cm):
  - Length of four bars: $a = 124.0$, $b = 171.2$, $g = 172.1$, $h = 122.6$.
  - Coupler position: $P_{pos} = 20.0\%$, $P_{offset} = 42.0\%$ of h.
  - Position of point A: $A_x = 27.0$, $A_y = 66.0$.
  - Angle of ground bar relative to horizon: $\theta = -70.0°$

Project Management

Live Software Demo

Summary and Conclusion

Literature

- Cvetkovic, Ivana and Stojicevic, Misa and Popkonstantinović, Branislav and Cvetković, Dragan. (2018). Classification, geometrical and kinematic analysis of four-bar linkages. 261-266. 10.15308/Sinteza-2018-261-266.