



المدرسة الوطنية العليا للفنون والمهن بالرباط
ⵜⴰⴳⴷⴰⵢⵜ ⵜⴰⵎⴻⵔⴰⵏⵜ ⵜⴰⵎⴰⵏⵏⴰⵢⵜ ⵜⴰⵖⴻⵔⴰⵏⵜ ⵜⴰⵎⴰⵏⵏⴰⵢⵜ
Ecole Nationale Supérieure d'Arts et Métiers de Rabat

DOCUMENT TECHNIQUE

Conception des profils de cames

Document 1 - Tome 1

Profil obtenu avec une loi parabolique

Partie 1 : Cas d'un poussoir à galet axe-centré

Département de la Mécanique spécialisée

Filière Ingénierie Aéronautique et Automobile

Date : Décembre 2025



Auteur-QR

Auteur : Koudaya Kossi Boris

Titulaire du Module : Prof. Razouki

Table des matières

Introduction des cames	2
Objectifs	3
Définitions des termes techniques	3
Projet de conception	3
Tracé de la spline de la loi parabolique	3
Composants de base	14
Liaisons du mécanisme	15
Méthode de traçage	16
Construction CATIA	18
Procédure d’obtention	19
Conclusion	22
Annexe	23

Introduction des cames

Les **comes**¹ sont des organes mécaniques essentiels dans la transformation du mouvement. Elles permettent de convertir un mouvement de rotation continu en un mouvement de translation ou d'oscillation selon un profil déterminé. Ce document technique présente la conception spécifique d'un profil de came utilisant une **loi parabolique**² pour le cas particulier d'un **poussoir à galet axe-centré**³.

NB : L'usage de ce document nécessite des connaissances de bases en conception assistée par ordinateur sous CATIA V5.

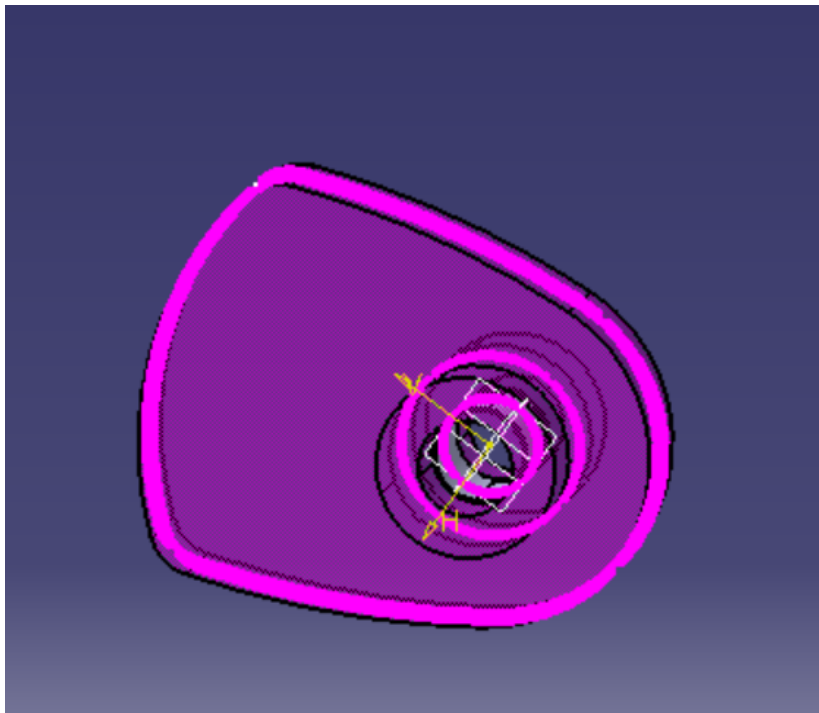


FIGURE 1 – Photo introductive

Caractéristiques principales du système étudié :

- Type de poussoir : **Poussoir à galet axe-centré**
- Loi de mouvement : **Loi parabolique**
- Levée maximale : 0A
- Angles caractéristiques : 0-120° (montée), 120-160° (palier haut), 160-280° (redescente), 280-360° (palier bas)

1. *Terme technique : comes*

2. *Terme technique : loi parabolique*

3. *Terme technique : poussoir à galet axe-centré*

Objectifs

1. Développer une méthodologie de conception pour un profil de came parabolique
2. Implémenter la génération du profil via différents outils (CATIA, Python, Excel)
3. Valider le profil par simulation cinématique
4. Documenter le processus d'itération et d'amélioration
5. Fournir des ressources réutilisables pour des conceptions similaires

Définitions des termes techniques

Terme	Définition
Came ⁴	Pièce mécanique dont le profil permet de transformer un mouvement rotatif en mouvement linéaire selon une loi déterminée.
Poussoir à galet ⁵	Élément suiveur équipé d'un galet qui roule sur le profil de la came, réduisant ainsi les frottements.
Loi parabolique ⁶	Loi de mouvement où l'accélération est constante, caractérisée par des arcs de parabole.
Spline ⁷	Courbe polynomiale par morceaux utilisée en CAO pour définir des profils complexes.
Courbe roulante ⁸	Liaison spécifique entre le galet et la came, permettant un roulement sans glissement.

Projet de conception

1. Tracé de la spline de la loi parabolique

Méthodes disponibles :

- Macro CATIA V5
- Excel avec connecteur GSD (Generative Shape Design)
- Script Python avec export CSV

-
4. *Terme technique : Came*
 5. *Terme technique : Poussoir à galet*
 6. *Terme technique : Loi parabolique*
 7. *Terme technique : Spline*
 8. *Terme technique : Courbe roulante*

— Manuellement dans l'esquisse CATIA

Description de la loi parabolique : La figure suivante illustre la loi parabolique utilisée :

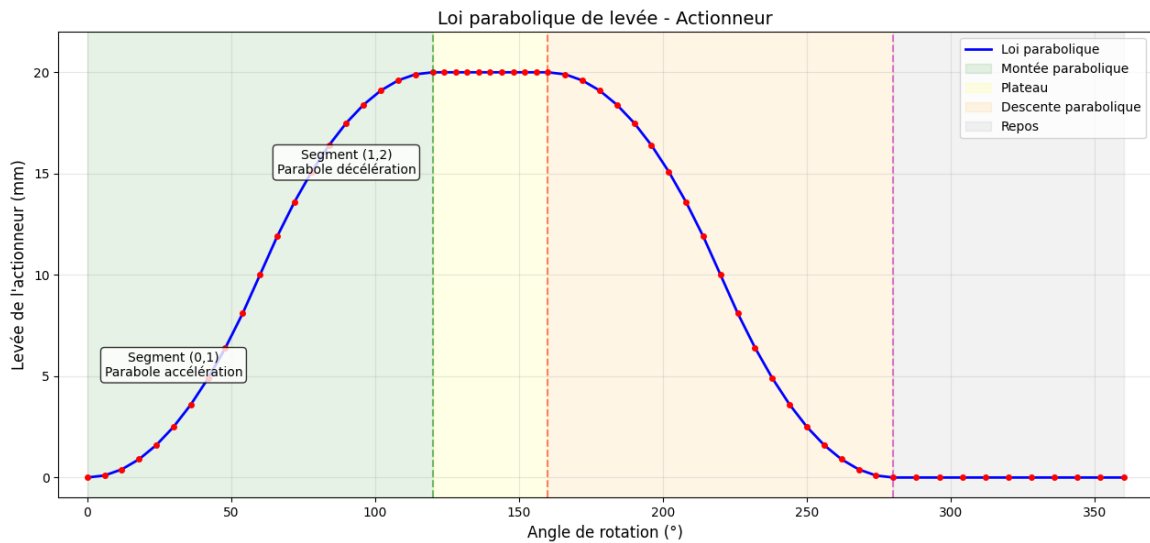


FIGURE 2 – Diagramme de levée - Loi parabolique

Caractéristiques :

- Levée de l'actionneur
- Levée maximale obtenue à 120°
- Palier maintenu pendant 40° de rotation
- Redescente symétrique de 160° à 280°
- Position basse maintenue de 280° à 360°

La grandeur représente la **levée de l'actionneur**. La levée maximale est atteinte pour un angle de rotation de 120° et se maintient sur une plage angulaire de 40° . La phase de descente, comprise entre 160° et 280° , est, dans cet exemple, **symétrique** de la phase de montée ; cette symétrie n'est toutefois pas une condition obligatoire. L'actionneur est ensuite maintenu en **position basse** entre 280° et 360° .

Les arcs de parabole avec coordonnées (*angle, levee*) sont tracés selon la méthode exposée précédemment. Le premier arc de parabole (0,1) est inscrit dans le premier rectangle, et la même procédure est appliquée pour les arcs suivants.

Dans le cas étudié, pour la phase de levée comprise entre 0° et 120° , chaque segment MM' est subdivisé en un nombre de sous-segments égal à celui utilisé pour le découpage du segment angulaire ($0^\circ, 120^\circ$), assurant ainsi une correspondance rigoureuse entre la loi angulaire et la loi de levée.

Exemple de feuille excel Macro-GSD-EXCEL-CATIA pour le spline

	A	B	C	D
1	StartLoft			
2	StartCurve			
3	0	0	0	
4	6	0.1	0	
5	12	0.4	0	
6	18	0.9	0	
7	2.40E+01	1.6	0	
8	30	2.5	0	
9	36	3.6	0	
10	42	4.9	0	
11	48	6.4	0	
12	54	8.1	0	
13	6.00E+01	10	0	
14	66	11.9	0	
15	72	13.6	0	
16	78	15.1	0	
17	84	16.4	0	
18	90	17.5	0	
19	96	18.4	0	
20	102	19.1	0	
21	108	19.6	0	
22	114	19.9	0	
23	120	20	0	
24	124	20	0	
25	128	20	0	
26	132	20	0	
27	136	20	0	
28	140	20	0	
29	144	20	0	
30	148	20	0	
31	152	20	0	
32	156	20	0	
33	160	20	0	
34	166	19.9	0	
35	172	19.6	0	
36	178	19.1	0	
37	184	18.4	0	
38	190	17.5	0	
39	196	16.4	0	
40	202	15.1	0	
41	208	13.6	0	
42	214	11.9	0	
43	220	10	0	

Feuil1 Feuil2 F

Prêt

Accessibilité : non

FIGURE 3 – GSD-PointSplineLoftFromExcel.xls dans le pack de CATIA V5

[Cliquez & Télécharger le fichier Excel Ici](#)

Exemple de script Python pour le spline

```

1  """
2  Script Python pour générer les points de la loi parabolique
3  Export en CSV pour utilisation dans CATIA V5
4  """
5  # -*- coding: utf-8 -*-
6  import win32com.client
7  import math
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  # =====
12  # PARAMÈTRES DE LA LOI PARABOLIQUE
13  # =====
14  h_max = 20.0 # Levée maximale en mm
15  theta_montée = 120.0 # Angle de montée en degrés
16  theta_plateau = 40.0 # Angle de plateau en degrés
17  theta_descente = 120.0 # Angle de descente en degrés
18  theta_repos = 80.0 # Angle de repos en degrés
19
20  # Vérification
21  if abs(theta_montée + theta_plateau + theta_descente + theta_repos -
22        360) > 0.1:
23      print("ATTENTION: La somme des angles doit être égale à 360 !")
24
25  # =====
26  # FONCTIONS DE LA LOI PARABOLIQUE
27  # =====
28  def loi_parabolique_montée(theta, theta_max, h_max):
29      """Loi parabolique pour la montée (2 paraboles)"""
30      if theta <= theta_max / 2:
31          # Première parabole (accélération) - segment (0,1)
32          return (2 * h_max / theta_max ** 2) * theta ** 2
33      else:
34          # Deuxième parabole (décélération) - segment (1,2)
35          return h_max - (2 * h_max / theta_max ** 2) * (theta_max -
36                theta) ** 2
37
38  def loi_parabolique_descente(theta, theta_max, h_max):
39      """Loi parabolique pour la descente (2 paraboles) - symétrique"""
40      if theta <= theta_max / 2:
41          # Première parabole (accélération négative) - segment
42          # (3,4)
43          return h_max - (2 * h_max / theta_max ** 2) * theta ** 2

```

```

44     # Deuxi me parabole (d c l ration n gative) - segment
        (4,5)
45     return (2 * h_max / theta_max ** 2) * (theta_max - theta) ** 2
46
47
48 def g n rer_points_loi():
49     """G n re les points de la loi parabolique selon votre
        description"""
50     points = []
51
52     # Nombre de points pour chaque segment
53     nb_pts = 10 # Points par sous-segment
54
55     # 1. Phase de mont e parabolique (0 120 ) - Segments (0,1) et
        (1,2)
56     print("\nPHASE DE MONT E (0-120 ):")
57     print("Segment (0,1): Parabole d'acc l ration")
58     print("Segment (1,2): Parabole de d c l ration")
59
60     for i in range(nb_pts * 2 + 1):
61         theta = i * (theta_mont e / (nb_pts * 2))
62         h = loi_parabolique_mont e(theta, theta_mont e, h_max)
63         points.append((theta, h))
64
65     # 2. Phase de plateau (120 160 )
66     print("\nPHASE DE PLATEAU (120-160 ):")
67     for i in range(1, nb_pts + 1):
68         theta = theta_mont e + i * (theta_plateau / nb_pts)
69         points.append((theta, h_max))
70
71     # 3. Phase de descente parabolique (160 280 ) - Segments (3,4)
        et (4,5)
72     print("\nPHASE DE DESCENTE (160-280 ):")
73     print("Segment (3,4): Parabole d'acc l ration n gative")
74     print("Segment (4,5): Parabole de d c l ration n gative")
75
76     for i in range(1, nb_pts * 2 + 1):
77         theta = theta_mont e + theta_plateau + i * (theta_descente /
            (nb_pts * 2))
78         h = loi_parabolique_descente(i * (theta_descente / (nb_pts *
            2)), theta_descente, h_max)
79         points.append((theta, h))
80
81     # 4. Phase de repos (280 360 )
82     print("\nPHASE DE REPOS (280-360 ):")
83     for i in range(1, nb_pts + 1):
84         theta = theta_mont e + theta_plateau + theta_descente + i * (
            theta_repos / nb_pts)

```



```

85         points.append((theta, 0))
86
87     return points
88
89
90     # =====
91     # G N RATION DES POINTS
92     # =====
93     points_loi = g n rer_points_loi()
94
95     # Afficher les coordonn es
96     print("\n" + "=" * 80)
97     print("COORDONN ES DE LA LOI PARABOLIQUE")
98     print("=" * 80)
99     print("Angle( )\tLev e(mm)\tX(mm)\t\tY(mm)")
100    print("-" * 80)
101
102    coords_x = []
103    coords_y = []
104
105    for angle_deg, lev e in points_loi:
106        # Pour afficher comme courbe 2D dans CATIA (X=angle, Y=lev e)
107        x_catia = angle_deg
108        y_catia = lev e
109
110        coords_x.append(x_catia)
111        coords_y.append(y_catia)
112
113        print(f"{angle_deg:8.2f}\t{lev e:10.3f}\t{x_catia:10.3f}\t{
114            y_catia:10.3f}")
115
116    print("=" * 80)
117
118    # =====
119    # TRAC DANS PYTHON (MATPLOTLIB)
120    # =====
121    print("\nCr ation du graphique Python...")
122
123    plt.figure(figsize=(12, 6))
124
125    # Courbe principale
126    plt.plot(coords_x, coords_y, 'b-', linewidth=2, label='Loi parabolique
127        ')
128    plt.plot(coords_x, coords_y, 'ro', markersize=4)
129
130    # Ajouter les zones
131    plt.axvspan(0, theta_mont e, alpha=0.1, color='green', label='Mont e
132        parabolique')

```

```

130 plt.axvspan(theta_mont e , theta_mont e + theta_plateau , alpha=0.1,
    color='yellow', label='Plateau')
131 plt.axvspan(theta_mont e + theta_plateau, theta_mont e +
    theta_plateau + theta_descente , alpha=0.1, color='orange',
132 label='Descente_parabolique')
133 plt.axvspan(theta_mont e + theta_plateau + theta_descente, 360, alpha
    =0.1, color='gray', label='Repos')
134
135 # Ajouter les lignes verticales pour les transitions
136 plt.axvline(x=theta_mont e , color='g', linestyle='--', alpha=0.5)
137 plt.axvline(x=theta_mont e + theta_plateau, color='r', linestyle='--'
    , alpha=0.5)
138 plt.axvline(x=theta_mont e + theta_plateau + theta_descente, color='m
    ', linestyle='--', alpha=0.5)
139
140 # Ajouter les annotations pour les segments de paraboles
141 plt.text(theta_mont e / 4, h_max * 0.25, 'Segment_(0,1)\nParabole_
    acc l l r a t i o n',
142 ha='center', bbox=dict(boxstyle="round,pad=0.3", facecolor="
    white", alpha=0.8))
143 plt.text(theta_mont e * 3 / 4, h_max * 0.75, 'Segment_(1,2)\nParabole
    _ d c l r a t i o n',
144 ha='center', bbox=dict(boxstyle="round,pad=0.3", facecolor="
    white", alpha=0.8))
145
146 # Configuration du graphique
147 plt.xlabel('Angle_de_rotation_( )', fontsize=12)
148 plt.ylabel('Lev e _de_l\'actionneur_(mm)', fontsize=12)
149 plt.title('Loi_parabolique_de_lev e _-Actionneur', fontsize=14)
150 plt.grid(True, alpha=0.3)
151 plt.legend(loc='upper_right')
152 plt.xlim(-10, 370)
153 plt.ylim(-1, h_max + 2)
154
155 plt.tight_layout()
156 plt.savefig('loi_parabolique_actionneur.png', dpi=150)
157 plt.show()
158
159 print("Graphique_Python_sauvegard _sous_'loi_parabolique_actionneur.
    png'")
160
161 # =====
162 # TRAC DANS CATIA V5
163 # =====
164 print("\nConnexion _CATIA_V5...")
165
166 try:
167     # -----

```

```

168 # Connexion CATIA
169 # -----
170 catia = win32com.client.Dispatch("CATIA.Application")
171 catia.Visible = True
172
173 # Cr er un nouveau document Part
174 documents = catia.Documents
175 part_doc = documents.Add("Part")
176 part = part_doc.Part
177
178 # Factory pour cr er les formes
179 hsf = part.HybridShapeFactory
180
181 # Cr er un corps hybride pour organiser les lments
182 hybrid_bodies = part.HybridBodies
183 hybrid_body = hybrid_bodies.Add()
184 hybrid_body.Name = "Loi_Parabolique_Actionneur"
185
186 # -----
187 # Cr ation des points dans CATIA
188 # -----
189 print("Cr ation des points dans CATIA...")
190 points_catia = []
191
192 for i, (angle_deg, lev e) in enumerate(points_loi):
193     # Cr er un point avec coordonn es (X=angle, Y=lev e , Z=0)
194     pt = hsf.AddNewPointCoord(angle_deg, lev e , 0.0)
195     pt.Name = f"Point_{i+1:03d}"
196     hybrid_body.AppendHybridShape(pt)
197     points_catia.append(pt)
198
199     if i % 10 == 0: # Afficher progression
200         print(f" Point_{i+1}/{len(points_loi)} cr ")
201
202 # -----
203 # Cr ation de la spline dans CATIA
204 # -----
205 print("Cr ation de la spline dans CATIA...")
206 spline = hsf.AddNewSpline()
207 spline.SetSplineType(0) # 0 pour spline par points
208
209 for pt in points_catia:
210     spline.AddPoint(pt)
211
212 spline.Name = "Courbe_Loi_Parabolique"
213 hybrid_body.AppendHybridShape(spline)
214
215 # -----

```

```

216 # Ajouter des axes et rep res
217 # -----
218 # Axe X (angle)
219 pt_origine = hsf.AddNewPointCoord(0, 0, 0)
220 pt_x = hsf.AddNewPointCoord(380, 0, 0)
221 line_x = hsf.AddNewLinePtPt(pt_origine, pt_x)
222 line_x.Name = "Axe_Angle"
223 hybrid_body.AppendHybridShape(pt_origine)
224 hybrid_body.AppendHybridShape(pt_x)
225 hybrid_body.AppendHybridShape(line_x)
226
227 # Axe Y (lev e)
228 pt_y = hsf.AddNewPointCoord(0, h_max + 5, 0)
229 line_y = hsf.AddNewLinePtPt(pt_origine, pt_y)
230 line_y.Name = "Axe_Levee"
231 hybrid_body.AppendHybridShape(pt_y)
232 hybrid_body.AppendHybridShape(line_y)
233
234 # -----
235 # Ajouter du texte pour les zones
236 # -----
237 # Zone mont e
238 pt_text_mont e = hsf.AddNewPointCoord(theta_mont e / 2, h_max /
239 2, 0)
240 text_mont e = part.MainBody.DrawingTexts.Add("MONT E┐PARABOLIQUE
241 ", pt_text_mont e)
242
243 # Zone plateau
244 pt_text_plateau = hsf.AddNewPointCoord(theta_mont e +
245 theta_plateau / 2, h_max, 0)
246 text_plateau = part.MainBody.DrawingTexts.Add("PLATEAU",
247 pt_text_plateau)
248
249 # Zone descente
250 pt_text_descente = hsf.AddNewPointCoord(theta_mont e +
251 theta_plateau + theta_descente / 2, h_max / 2, 0)
252 text_descente = part.MainBody.DrawingTexts.Add("DESCENTE┐
253 PARABOLIQUE", pt_text_descente)
254
255 # Zone repos
256 pt_text_repos = hsf.AddNewPointCoord(theta_mont e + theta_plateau
257 + theta_descente + theta_repos / 2, 2, 0)
258 text_repos = part.MainBody.DrawingTexts.Add("REPOS", pt_text_repos
259 )
260
261 # -----
262 # Mise      jour et affichage
263 # -----

```

```

256     part.Update()
257
258     # Ajuster la vue
259     viewer = part_doc.GetViewer3D(1)
260     viewer.FitAllIn()
261
262     print("\n" + "=" * 80)
263     print("TRAC  CATIA_V5_R  USSI!")
264     print("=" * 80)
265     print(f"Document: {part_doc.Name}")
266     print(f"Corps_hybride: {hybrid_body.Name}")
267     print(f"Nombre_de_points: {len(points_catia)}")
268     print(f"Spline: {spline.Name}")
269     print("\n l  ments  cr   s:")
270     print("  - Courbe de la loi parabolique")
271     print("  - Axes X (angle) et Y (lev e)")
272     print("  - Annotations des zones")
273     print("  - Tous les points de discr tisation")
274
275     # -----
276     # Cr er un rapport texte dans CATIA
277     # -----
278     param tres_text = f"""
279     PARAM TRES_DE_LA_LOI_PARABOLIQUE:
280     =====
281     Lev e_maximale: {h_max}mm
282     Phase_mont e: 0-{theta_mont e}  (parabolique)
283     Phase_plateau: {theta_mont e}-{theta_mont e+theta_plateau}
284     Phase_descente: {theta_mont e+theta_plateau}-{theta_mont e+
285                     theta_plateau+theta_descente}  (parabolique)
286     Phase_repos: {theta_mont e+theta_plateau+theta_descente}-360
287     Nombre_de_points: {len(points_loi)}
288     Segments_paraboliques:
289     - (0,1): Acc l ration
290     - (1,2): D c l ration
291     - (3,4): Acc l ration n gative
292     - (4,5): D c l ration n gative
293     """
294
295     pt_param = hsf.AddNewPointCoord(-50, h_max + 10, 0)
296     text_param = part.MainBody.DrawingTexts.Add(param tres_text,
297                                                  pt_param)
298
299     print("\nParam tres ajout s au document CATIA")
300
301 except Exception as e:
302     print(f"\nERREUR lors de la connexion  CATIA: {str(e)}")
303     print("\nAssurez-vous que:")

```

```

302     print("1. CATIA V5 est installé ")
303     print("2. CATIA est ouvert")
304     print("3. La bibliothèque que win32com est installée (pip install pywin32)")
305
306     # Sauvegarder les points dans un fichier pour import manuel
307     print("\nSauvegarde des points dans un fichier CSV...")
308     with open('points_loi_parabolique.csv', 'w', encoding='utf-8') as f:
309         f.write("Index;Angle(deg);Levee(mm);X;Y\n")
310         for i, (angle_deg, levee) in enumerate(points_loi):
311             f.write(f"{i+1};{angle_deg:.2f};{levee:.3f};{angle_deg:.3f};{levee:.3f}\n")
312
313     print("Points sauvegardés dans 'points_loi_parabolique.csv'")
314
315     # =====
316     # SAUVEGARDE DES DONNÉES
317     # =====
318     print("\n" + "=" * 80)
319     print("SAUVEGARDE DES DONNÉES")
320     print("=" * 80)
321
322     # Sauvegarder dans un fichier texte
323     with open('loi_parabolique_detailed.txt', 'w', encoding='utf-8') as f:
324         f.write("=" * 80 + "\n")
325         f.write("LOI PARABOLIQUE D'ACTIONNEUR - RAPPORT TAILLÉ\n")
326         f.write("=" * 80 + "\n\n")
327
328         f.write("PARAMÈRES:\n")
329         f.write("-" * 40 + "\n")
330         f.write(f"Levee maximale (h_max): {h_max} mm\n")
331         f.write(f"Angle monté: 0-{theta_monté} (theta_monté/360 * 100: .1f) % du cycle)\n")
332         f.write(
333             f"Angle plateau: {theta_monté}-{theta_monté+theta_plateau} (theta_plateau/360 * 100: .1f) % du cycle)\n")
334         f.write(
335             f"Angle descente: {theta_monté+theta_plateau}-{theta_monté+theta_plateau+theta_descente} (theta_descente/360 * 100: .1f) % du cycle)\n")
336         f.write(
337             f"Angle repos: {theta_monté+theta_plateau+theta_descente}-360 (theta_repos/360 * 100: .1f) % du cycle)\n\n")
338
339         f.write("SEGMENTS PARABOLIQUES:\n")
340         f.write("-" * 40 + "\n")

```

```

341     f.write("MONT E:\n")
342     f.write("  Segment (0,1): Parabole d'acc l ration (0-60 )\n")
343     f.write("  Segment (1,2): Parabole de d c l ration (60-120 )\n")
344     f.write("DESCENTE:\n")
345     f.write("  Segment (3,4): Parabole d'acc l ration n gative (160-220 )\n")
346     f.write("  Segment (4,5): Parabole de d c l ration n gative (220-280 )\n\n")
347
348     f.write("POINTS DE LA COURBE:\n")
349     f.write("-" * 40 + "\n")
350     f.write("Index\tAngle( )\tLev e (mm)\tX\tY\n")
351     f.write("-" * 80 + "\n")
352
353     for i, (angle_deg, lev e) in enumerate(points_loi):
354         f.write(f"{i+1:03d}\t{angle_deg:8.2f}\t{lev e:10.3f}\t{angle_deg:10.3f}\t{lev e:10.3f}\n")
355
356     print("Donn es sauvegard es dans:")
357     print("  loi_parabolique_detailed.txt (rapport complet)")
358     print("  loi_parabolique_actionneur.png (graphique)")
359     if 'points_loi_parabolique.csv' in locals():
360         print("  points_loi_parabolique.csv (points pour import)")
361
362     print("\n" + "=" * 80)
363     print("PROCESSUS TERMIN AVEC SUCC S!")
364     print("=" * 80)
365     print("\nR sum :")
366     print(f"  Graphique Python cr et affich ")
367     print(f"  Courbe trac e dans CATIA V5 (X=angle, Y=lev e)")
368     print(f"  {len(points_loi)} points g n r s")
369     print(f"  4 segments paraboliques conformes votre description")
370     print(f"  Lev e maximale: {h_max} mm atteinte {theta_mont e} ")
371     print("\n" + "=" * 80)

```

Listing 1 – Génération de spline avec la méthode win23 pour loi parabolique (loi-parabolique.py)

 [Télécharger le script complet](#)

2. Les composants de base du produit à simuler

Le mécanisme complet comprend les fichiers CATPart suivants : (Vous pouvez télécharger les différentes pièces sur not github)

Fichier	Description	Lien Git Hub
suiveur.part	Tige suiveuse avec guide prismatique	suiveur.part
came.part	Profil de came parabolique	came.part
galet.part	Galet du poussoir	galet.part
S0.part	Bâti fixe de référence	S0.part

TABLE 1 – Fichiers CATPart du mécanisme

Visualisation des composants :

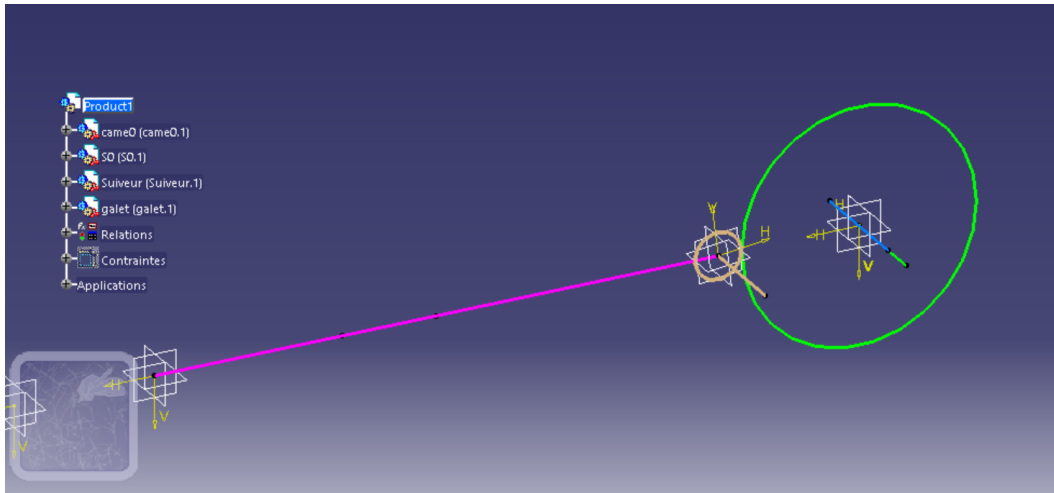


FIGURE 4 – Vue en mode filaire des composants principaux

3. Les liaisons du mécanisme

les liaisons à définir sur Catia V5 :

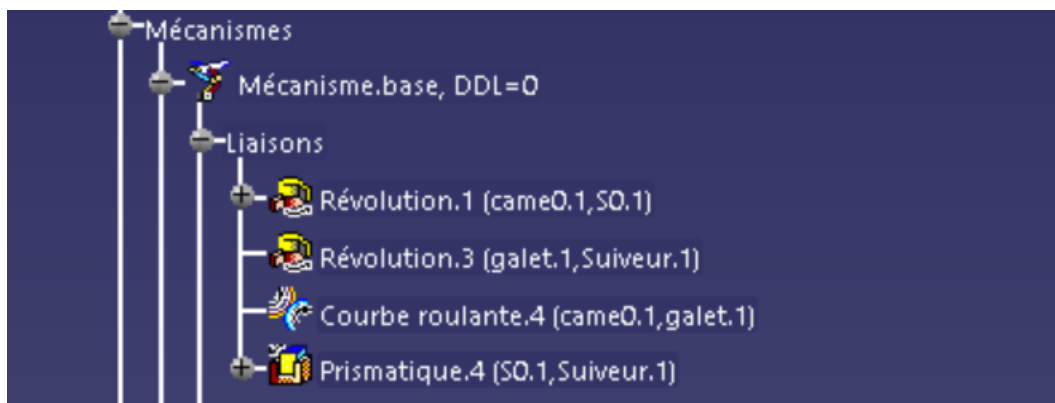


FIGURE 5 – Atelier-kinematics

1. **Révolution** entre **came**⁹ et **S0**¹⁰ (centre de rotation)

9. *Terme technique : came*

10. *Terme technique : S0*

2. **Révolution** entre **galet**¹¹ et **tige suiveuse**¹²
3. **Prismatique** entre **S0**¹³ et **tige suiveuse**¹⁴
4. **Courbe roulante** entre **poussoir**¹⁵ et **galet**¹⁶ (liaison principale critique)
5. **Fixe** pour **S0**¹⁷ (bâti)

4. Méthode de traçage du profil de la came

Profil de la came

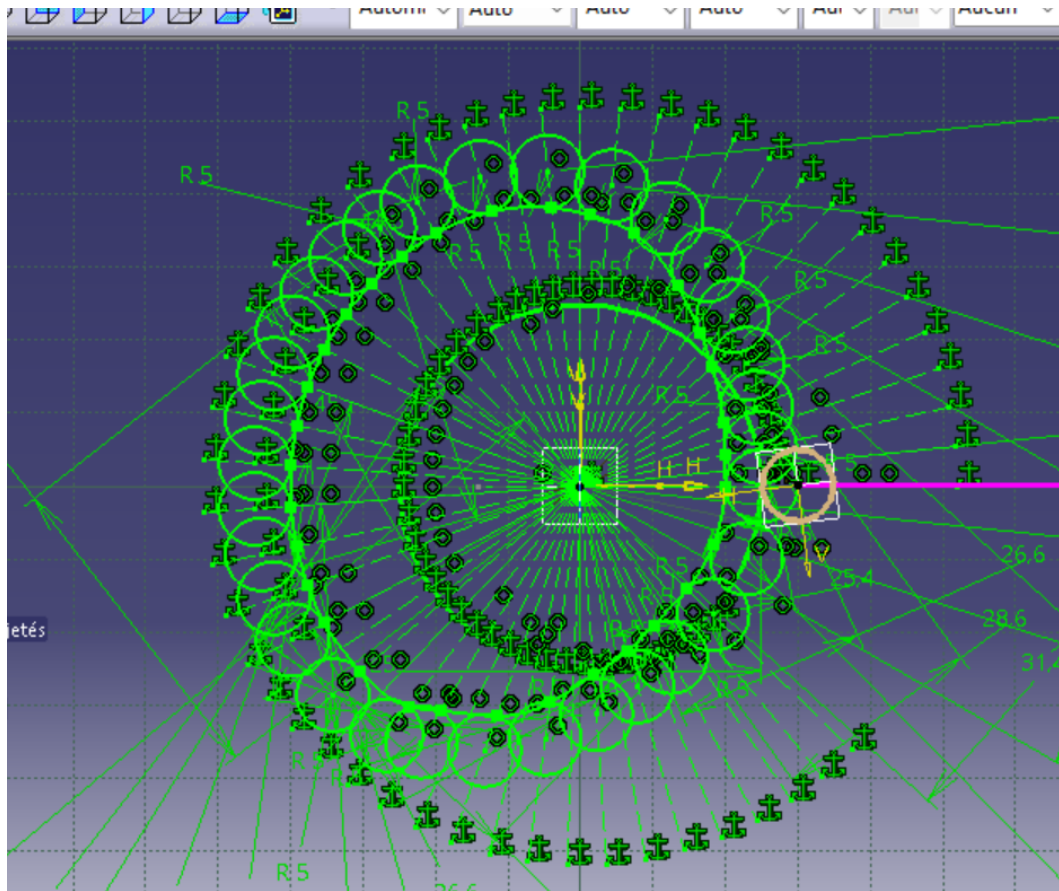


FIGURE 6 – Sketch du profil parabolique

1. Tracer le cercle de base :

— Centre : centre de rotation de la came

11. *Terme technique : galet*
12. *Terme technique : tige suiveuse*
13. *Terme technique : S0*
14. *Terme technique : tige suiveuse*
15. *Terme technique : poussoir*
16. *Terme technique : galet*
17. *Terme technique : S0*

- Rayon : distance minimale centre-came → point de contact avec le poussoir
- 2. **Diviser le cercle de base** en secteurs angulaires correspondant au diagramme de levée
- 3. **Mesurer la hauteur de levée** sur le diagramme et la reporter à l'extérieur du cercle de base
- 4. **Tracer la courbe enveloppe** des points de contact avec corrections selon :
 - Type de poussoir : sabot, plateau tangent, galet
 - Position de l'axe : centré ou excentré

Méthode des arcs de parabole : Les arcs de parabole sont tracés comme suit : La construction du profil de la came sous **CATIA** repose sur la loi de levée décrite précédemment et peut être réalisée selon plusieurs méthodes, en fonction du niveau de maîtrise et des habitudes du concepteur. Elle peut faire appel à différents outils avancés du logiciel, tels que la **diversification optimisée**, les **Power Copy**, les opérations de **rotation**, les **miroirs**, ainsi que la création d'**esquisses entièrement projetées**, ensuite isolées et rigoureusement définies par des **contraintes géométriques**. Des opérations de **translation** peuvent également être utilisées pour ajuster le positionnement des éléments.

Dans le cas étudié, le profil final de la came est obtenu par la **combinaison cohérente de l'ensemble de ces outils**, permettant d'aboutir à une géométrie conforme à la loi parabolique imposée. Pour la phase de levée (0, 120°), diviser MM' en sous-segments égaux au nombre de divisions angulaires.

Le tracé d'une came consiste à déterminer, dans un premier temps, son **profil théorique** à partir du **diagramme espace-temps**, en s'appuyant sur un **cercle minimal**. À partir de ce profil théorique, on obtient ensuite le **profil pratique** de la came en traçant une série de **cercles concentriques** de rayon égal à celui du galet, puis en déterminant la **courbe enveloppe** tangente à ces cercles. Afin de simplifier l'analyse géométrique, on adopte généralement l'hypothèse selon laquelle la **came est considérée fixe**, tandis que le **suiveur** est supposé tourner en **sens inverse** du sens réel de rotation de la came.

La méthode générale de tracé se décompose en plusieurs étapes fondamentales. Elle débute par la définition des données géométriques et cinématiques, telles que le **diamètre du galet**, le **rayon minimal de la came** (correspondant à la levée nulle), le **sens de rotation**, ainsi que le **diagramme espace-temps** représentant l'évolution de la levée. On procède ensuite au tracé du **cercle minimal**, qui est divisé en un nombre de secteurs angulaires égaux, identiques à ceux du diagramme espace-temps. Les **courses** relevées sur ce diagramme sont alors reportées radialement à partir du cercle minimal afin de déterminer les positions successives du centre du galet.

Pour chaque position obtenue, on trace un cercle de rayon égal à celui du galet. La **courbe enveloppe** de l'ensemble de ces cercles constitue le **profil pratique de la came**. Dans le cas particulier d'une **came à contact direct (poussoir à sabot)**, le profil pratique est confondu avec la courbe obtenue par le report direct des levées depuis le cercle minimal.

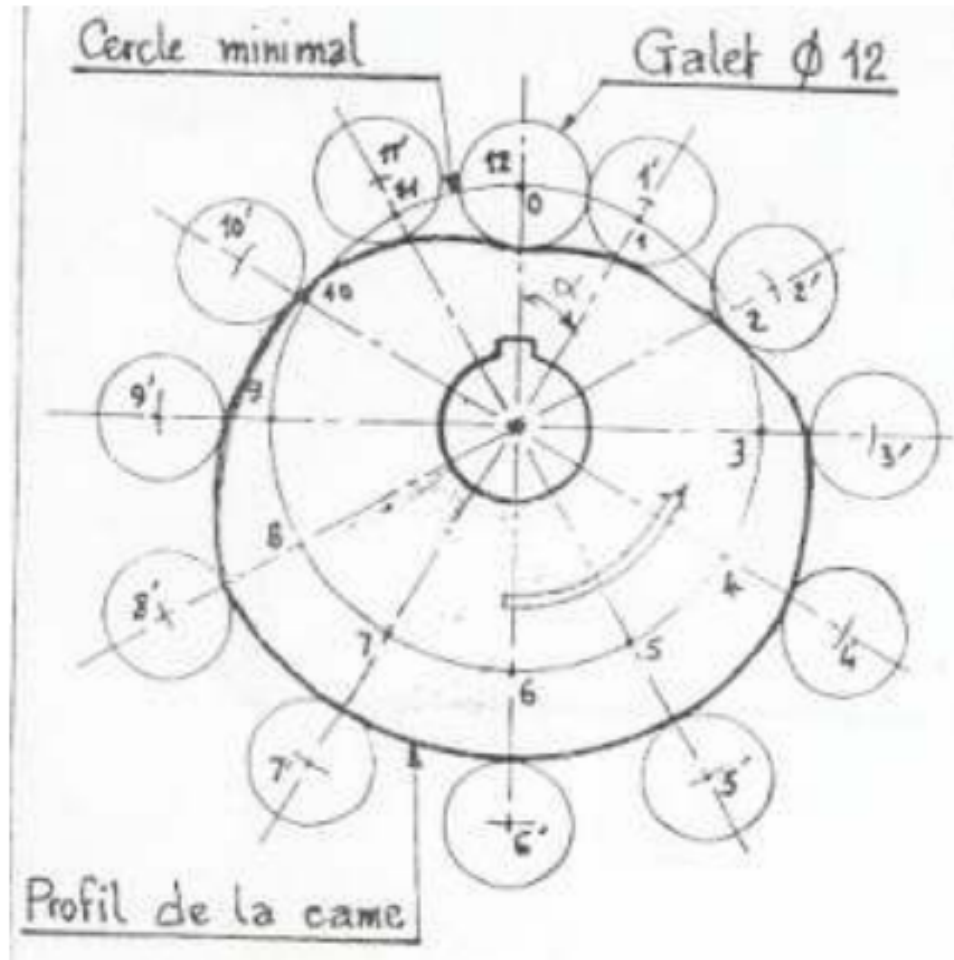


FIGURE 7 – Dessin explicatif du tracer

5. Construction du profil sur CATIA

Outils CATIA utilisés :

- **PowerCopy**¹⁸ : Pour réplication de motifs
- **Rotations**¹⁹ et **Miroirs**²⁰ : Pour symétries
- **Esquisse_tout projetée**²¹ : Projection complète

18. Terme technique : PowerCopy

19. Terme technique : Rotations

20. Terme technique : Miroirs

21. Terme technique : Esquisse_tout projetée

- **Contraintes**²² : Fixation géométrique
- **Translations**²³ : Déplacement de profils

Procédure détaillée :

1. Import des points CSV générés par Python ou Macro Excel
2. Création d'une **spline**²⁴ passant par tous les points
3. Projection dans l'esquisse de la came
4. Application des rotations nécessaires pour fermer le profil
5. Vérification de la continuité tangente (G1) et courbure (G2)

6. Procédure d'obtention du profil parfait

Premier mécanisme de test

1. Création d'un mécanisme de base avec profil circulaire
2. Commande de la liaison prismatique avec la loi parabolique importée
3. Comparaison avec commande simple de révolution

Résultat initial : Le système n'évolue pas correctement comparé à la commande simple de révolution. Cause identifiée : profil mal construit suivant la loi parabolique.

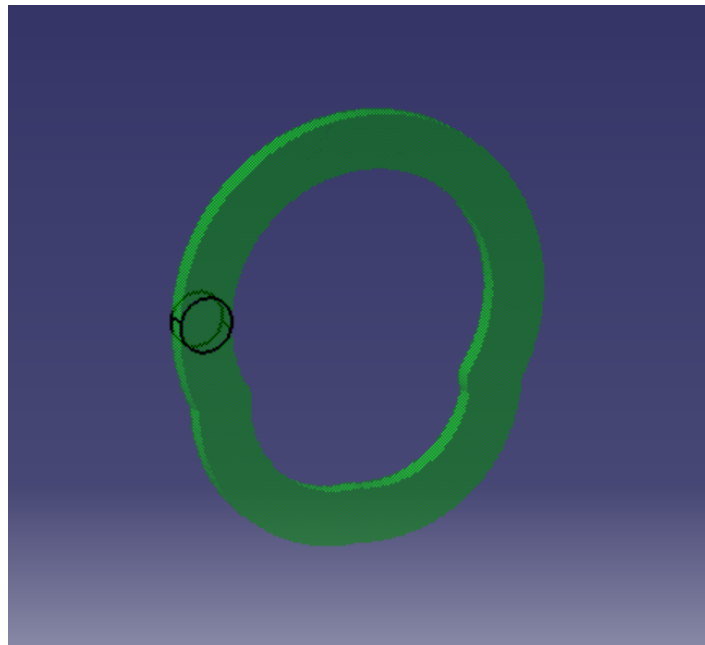


FIGURE 8 – Image montrant une enveloppe générée par l'outil de balayage

22. *Terme technique : Contraintes*
23. *Terme technique : Translations*
24. *Terme technique : spline*



FIGURE 9 – outil de balayage

Amélioration par balayage d'enveloppe

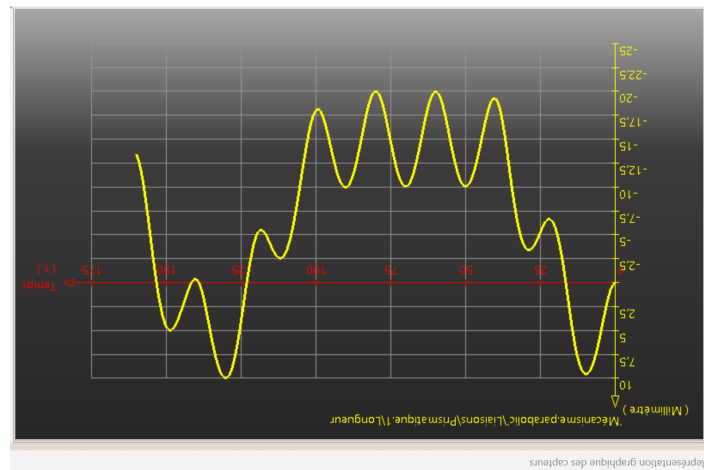


FIGURE 10 – Essai A - Courbe obtenue (défauts visibles)

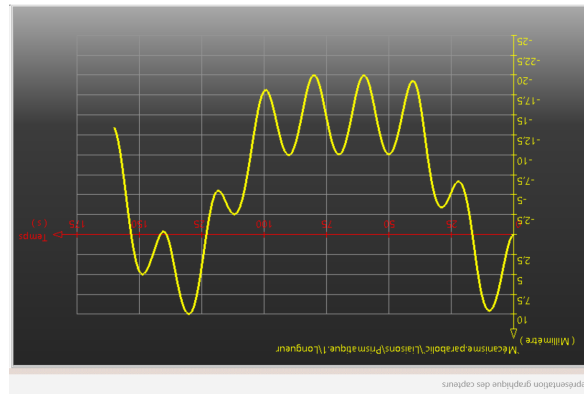
- Génération du volume balayé par le poussoir (enveloppe)
- Visualisation des interférences et zones problématiques
- Correction itérative du profil

Itérations successives :

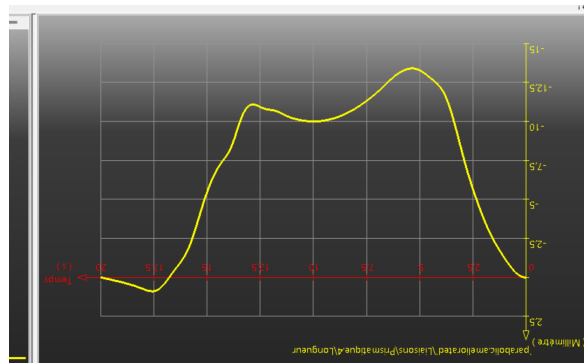
Dans une phase d'amélioration du profil, des **balayages de volume**, appelés **enveloppes**, sont générés afin de visualiser le volume effectivement balayé par le poussoir au cours du mouvement. Cette analyse permet d'identifier les zones du profil de la came présentant des écarts ou des défauts de tracé. Les portions concernées du profil sont alors revues et corrigées, donnant lieu à des améliorations successives de la conception. Après chaque modification, le **graphique de levée obtenu** est de nouveau visualisé et comparé à la **courbe théorique de référence**. Les résultats sont analysés à travers plusieurs essais successifs (**Essai A**, **Essai B**, **Essai C**), mettant en évidence une convergence progressive du comportement réel du mécanisme vers le comportement attendu. Cette démarche itérative est poursuivie jusqu'à l'obtention d'un profil de came présentant une conformité quasi parfaite avec la loi de levée imposée.

Essai	État	Observations
A	Échec	Profil incorrect, discontinuités dans le mouvement
B	Amélioration	Transitions améliorées mais encore des défauts
C	Presque parfait	Graphique s'approche de la forme idéale
Final	Validé	Forme conforme à la loi parabolique

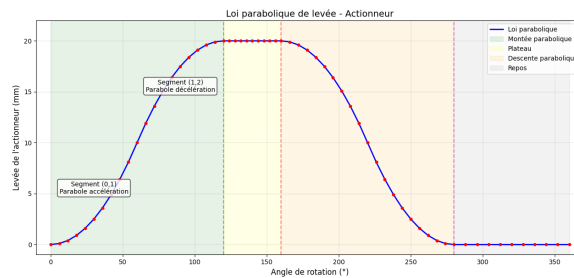
TABLE 2 – Itérations d'amélioration du profil



Essai B - Amélioration



Essai C - Proche du but



La forme attendue

FIGURE 11 – Évolution du profil à travers les itérations

Conclusion

La conception d'un profil de came avec loi parabolique pour un poussoir à galet axe-centré nécessite une approche méthodique et itérative. Les points clés de réussite sont :

- **Précision du tracé** : Utilisation d'outils numériques (Macro GSD-Excel-Catia, Python, CATIA) pour générer les points exacts
- **Vérification itérative** : Comparaison constante entre le mouvement théorique et simulé

- **Enveloppe de balayage** : Outil essentiel pour visualiser les défauts du profil
- **Validation cinématique** : Test complet du mécanisme dans son environnement de simulation

Le profil final obtenu garantit une transition douce entre les phases de mouvement et minimise les chocs dynamiques, conformément aux exigences de la loi parabolique.

Annexe


Ressources complémentaires

- Veuillez contacter l'adresse indiquée ci-dessous pour obtenir les autres documents complets, notamment le **Tome II**, le **Tome III**, et les volumes suivants.
- Bibliothèque de lois de mouvement (parabolique, sinusoïdale, modifiée)
- Scripts Python avancés avec interface graphique
- Macros CATIA pour automatisation complète

Paramètres recommandés pour ce projet

- Rayon de base : 25 mm
- Levée maximale (0A) : 20 mm
- Diamètre du galet : 5X2 mm
- Excentrement : 0 mm (axe-centré) : [ce problème à résoudre avec outil translation]

Contacts et informations

- **Auteur** : Koudaya Kossi Boris
- Mon Manager sur (+212) 719-737036
- **Email institutionnel** : kossiboris_koudaya@um5.ac.ma
- **GitHub** :  [Référentiel complet](#)
- **Version du document** : 1.0 (Décembre 2025)
- **Mots-clés** : Came, profil parabolique, poussoir galet, CATIA, Python, simulation