



MARMARA UNIVERSITY
FACULTY OF ENGINEERING



Design Optimization of Gas Turbine Rotor Using Genetic Algorithm

Furkan Kuru

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Doç. Dr. İbrahim Sina Kuseyri

ISTANBUL, 2025



MARMARA UNIVERSITY
FACULTY OF ENGINEERING



Design Optimization of Gas Turbine Rotor Using Genetic Algorithm

by Furkan Kuru

June 23, 2025, Istanbul

**SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE**

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in anyway include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.

Signature of Author(s).....

A photograph of a handwritten signature in black ink.

Department of Mechanical Engineering

Certified By.....

Project Supervisor, Department of Mechanical Engineering

Accepted By

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Doç. Dr. İbrahim Sina Kuseyri, for the valuable guidance and advice on preparing this thesis and giving me moral and material support.

June, 2025

Furkan Kuru

Table of Contents

ACKNOWLEDGEMENTS.....	i
ABSTRACT.....	iii
SYMBOLS.....	iv
ABBREVIATIONS.....	vii
1. INTRODUCTION.....	2
2. THEORETICAL BACKGROUND.....	5
2.1. Rotor Dynamics.....	5
2.2. Finite Element Method.....	7
2.2.1. Beam Elements.....	7
2.2.1.1. Coordinate Systems.....	7
2.2.1.2. Shaft Line Model.....	8
2.2.1.3. Euler-Bernoulli Beam Theory.....	9
2.2.1.4. Mass Matrix.....	12
2.2.1.5. Gyroscopic Matrix.....	12
2.2.1.6. Stiffness Matrix.....	12
2.2.2. Disk Elements.....	13
2.2.3. Bearing Elements.....	14
2.2.3.1. Stiffness Matrix.....	15
2.2.3.2. Damping Matrix.....	15
2.3. Strength of Materials.....	16
2.3.1. Centrifugal Stresses.....	17
2.3.1.1. Thin Disk.....	17
2.3.1.2. Cylindrical Body.....	17
2.3.2. Bending Stresses.....	18
2.3.3. Von Mises Failure Criterion.....	18
2.3.4. Fatigue Failure with Soderberg Criterion.....	19
2.3.4.1. Endurance Limit with Modifying Factors.....	19
2.3.4.2. Stress Concentration.....	20
2.4. PW4000 Engine.....	20
2.5. Optimization Formulation.....	20
2.6. Genetic Algorithm.....	21
3. SOFTWARE IMPLEMENTATION.....	24
4. RESULTS AND DISCUSSION.....	24
5. CONCLUSION.....	27
6. REFERENCES.....	27
7. APPENDICES.....	28

ABSTRACT

Design Optimization of Gas Turbine Rotor Using Genetic Algorithm

Rotor Dynamics is a subject such that combination of the Classical Mechanics , the Finite Element Method and the Theory of Elasticity. It is easier to model a given rotor geometry along with disks it carries on than a rotor with given some constraints consist of deflection , stress and fatigue limits. This one is an inverse problem and this is also an optimization problem. The classical optimization algorithms like Linear or Quadratic Programming will fail because of the model complexity. We showed that the genetic algorithm works well on the rotor design problem. We took PW4000-94 jet engine as a base model and we made assumptions on values of rotor length, rotor speed, disk thicknesses ,outer diameters, disk locations, bearing oil and bearing length. As a result, we get diameters of the beam elements and bearing clearances . In the optimization algorithm, we used the potential energy of the rotor as the objective function ; deflections and centrifugal stresses at disks, fatigue at the beam as constraints. In the optimization problem, we model the rotor using Euler-Bernoulli Beam Theory .

SYMBOLS

- T: Kinetic energies of each degree of freedom
- U: Strain Energies of each degree of freedom
- Q_k : Generalized Forces acting on the kth coordinate
- q_k : Generalized coordinates
- j: Square root of -1
- M: Total mass matrix
- Ω : Angular speed (rad/s)
- G: Gyroscopic matrix
- K: Stiffness matrix
- q: Generalized coordinates (Lateral deflections and slopes of element nodes)
- \Re : Real part of the vector
- b_{0k} : kth node part of b_0 vector
- m_k : Unbalance mass (kg)
- ϵ : Eccentricity of the unbalance mass (m)
- δ : Angular location of unbalance
- I_{dk} : Diametral mass moment of inertia of the kth beam element
- I_{pk} : Polar mass moment of inertia of the kth beam element
- β : Misalignment angle
- γ : Misalignment angular location
- Ox , Oy , and Oz : x,y and z-axes respectively
- u , v , and w : Translation about x,y and z-axes respectively
- θ , ψ and φ : Rotations about x,y and z-axes respectively
- ρ_e : The density of the material (kg/m³)
- A_e :The cross-sectional area of the beam.

m_d : The disk mass (kg)

I_d : The diametral mass moment of inertia of the disk.

I_p : The polar moment of inertia of the disk.

f : Magnitude of the resultant force(N)

D : Diameter of the bushing (m)

L : Length of the bearing(m)

c : Clearance (m)

ϵ : Eccentricity , which is the ratio between the journal displacement and the radial clearance

Γ : Density of the material (kg/m^3) (7800 kg/m^3 for steels)

ν : Poisson ratio (0.3 for steel)

r_i : Inner radius (m) (also outer radius of the beam)

r_e : Outer radius (m)

σ_r : Radial stresses

σ_t : Tangential stresses

σ_a : Alternating Stress , in our case it is bending

σ_m : Mean Stress , in our case it is centrifugal stresses

S_e : Endurance limit for the material.

S_y : Yield strength

S_e' : Endurance limit without modifying factors.

k_a : surface factor (is assumed to be 0.75)

k_b : size factor (is assumed to be 0.63)

k_c : loading factor (for bending it is 1)

σ_{nom} : Nominal Stress

K_t : Stress concentration factor (1.38)

$f(x)$: Objective function ,in our case it is potential energy.

It is given in the “obj_func.m” file

$g_i(x)$: Inequality constraints

$h_j(x)$: Equality constraints

These are calculated by “nonlin_cons.m” file and includes deflection, stress and fatigue limits.

$x^{(L)}$ and $x^{(U)}$: Lower and upper bounds.

ABBREVIATIONS

GA : Genetic Algorithm

E-BBT : Euler-Bernoulli Beam Theory

List of Figures

Figure 1.1 Pratt & Whitney PW4000-112.....	3
Figure 2.1.1: General basic rotor structure with bearings and disks.....	5
Figure 2.2.1: Right-handed coordinate set used in the analysis of rotor.....	8
Figure 2.2.2: The local coordinates for the beam element , for x-axis or yz-plane.....	8
Figure 2.2.3: The local coordinates in the two bending planes.....	11
Figure 4.1: Minimum of the objective function in each generation vs. Generation.....	26
Figure 4.2: Rotor geometry.....	26

List of Tables

1. INTRODUCTION

Rotor Dynamics is a subject such that combination of the Classical Mechanics , the Finite Element Method and the Theory of Elasticity. The Theory of Elasticity and the Finite Element Method deals with mostly stationary elastic systems, Classical Mechanics deals with mostly moving rigid bodies. Rotor Dynamics merges this subjects into a specific scientific area.

Modeling a given rotor with to find deflections and stresses of the beam is a relatively easier than modeling a given deflections and stresses of the beam with to find which rotor satisfies this conditions. Because the first one is a direct problem and when we solve the corresponding equation, we find the result. Satisfying given deflection and stress constraints are relatively easy but it may not be optimal, so we have to find an optimal of the multiple convenient solutions. To provide an optimal solution , we have to have an optimization algorithm and a function to be optimized called ‘objective function’. Optimization algorithms may split into two groups:

1. Analytic / Deterministic Algorithms : For example ;Gradient Descent, Linear/Quadratic Programming, Sequential Quadratic Programming etc.
2. Metaheuristic Algorithms : Genetic Algorithm, Particle Swarm Optimization , Simulated Annealing etc.

First group (Analytic / Deterministic Algorithms) addresses only analytic-continuous problems and our constraint functions should be differentiable. Especially Linear and Quadratic Programming case, we should have full mathematical expressions of the objective function and constraints.

Genetic algorithm is a search and improvement method inspired by natural evolution processes and developed for solving optimization problems. First introduced by John Holland in 1975, this algorithm effectively navigates the solution space by imitating genetic diversity, natural selection, mutation and crossover mechanisms in biological evolution.

GA is used to approach the global optimum in complex and multidimensional problems and is especially preferred in cases where classical optimization methods are insufficient. Its application areas are quite wide and it is widely used in disciplines such as engineering design, artificial intelligence, robotics, machine learning, finance, bioinformatics, route planning and resource allocation.

The basic functioning of the algorithm is based on the processes of evaluating solution candidates in populations, selecting them according to fitness values, and producing new individuals using crossover and mutation operators. This evolutionary cycle continues until sufficient recovery is achieved or the predetermined stopping criterion is reached.

Although the success of genetic algorithms varies depending on the problem structure and parameter settings, it occupies an important place in modern optimization studies thanks to its flexible structure and capacity to produce parallel solutions.

Second group is a more convenient to integrate rotor design problem, because too complex matrix inverses and constraint functions require to be calculated. We have chosen Genetic Algorithm (GA) for this project to apply rotor design problem.



Figure 1.1 Pratt & Whitney PW4000-112

We have chosen PW4000 jet engine to optimize and to demonstrate how GA works . The PW4000 series engines, developed by the American manufacturer Pratt & Whitney, are high-bypass turbofan jet engines designed primarily for wide-body commercial aircraft. First introduced in 1984, the PW4000 family powers various aircraft models including the Boeing 747, 767, 777, and the Airbus A300, A310, and A330. The series includes three major variants with fan diameters of 94, 100, and 112 inches, delivering thrust in the range of approximately 52,000 to 98,000 pounds.

The design of the PW4000 emphasizes fuel efficiency, ease of maintenance, and operational reliability. Its modular architecture allows for simplified on-wing maintenance, reducing downtime and operational costs. Furthermore, the incorporation of Full Authority Digital Engine Control (FADEC) enables precise engine management, enhancing both performance and safety.

Renowned for its durability and adaptability, the PW4000 series represents a significant advancement in modern turbofan engine design, meeting the rigorous demands of long-haul, high-capacity flight operations.

2. THEORETICAL BACKGROUND

2.1. Rotor Dynamics

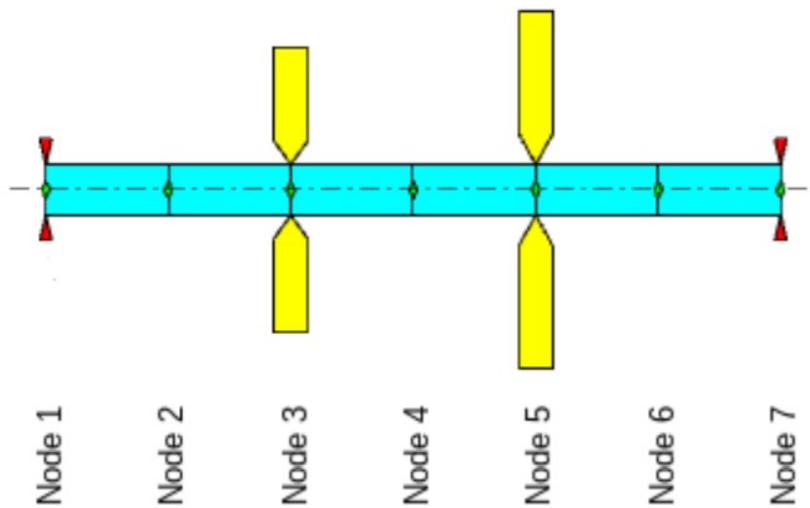


Figure 2.1.1: General basic rotor structure with bearings and disks

A rotor is composed of rotating shaft and disks to carry turbine and compressor blades . Rotor Dynamics is composed of dynamics of merely rotor and auxiliary elements such as bearings , seals , blades etc. We only took into account rotor and hydrodynamic bearing. A general rotor structure for dynamic equations is shown Figure 2.1.1 . All elements such as bearings ,disks , beam elements shown in this figure will contribute the general equation as mass , damping , stiffness and damping matrices.

The equations of motion are obtained by summing the contributions to the strain and kinetic energies from all elements , and by applying Lagrange's equations:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_k}\right) - \left(\frac{\partial T}{\partial q_k}\right) + \left(\frac{\partial U}{\partial q_k}\right) = Q_k \quad , \text{for } k=1,\dots,n \quad (2.1)$$

T: Kinetic energies of each element

U: Strain Energies of each element

Q_k : Generalized Forces acting on the kth coordinate (non-conservative forces and unbalance forces)

q_k : Generalized coordinates

j: Square root of -1

And finally we find the general equation structure for rotor dynamics:

$$M \ddot{q} + \Omega G \dot{q} + C \dot{q} + K q = \Re(\Omega^2 b_0 e^{j\Omega t}) \quad (2.2)$$

M: Total mass matrix

Ω : Angular speed (rad/s) .In our case ,we assumed it is equal to 3000rpm =50 Hz=100π rad/s

G: Gyroscopic matrix

K: Stiffness matrix

q: Generalized coordinates (Lateral deflections and slopes of element nodes)

\Re : Real part of the vector

b_0 : Unbalance forces and moments vector

For each node , b_0 vector is as follows:

$$\mathbf{b}_{0k} = \begin{pmatrix} m_k \epsilon e^{j\delta} \\ -j m_k \epsilon e^{j\delta} \\ j(I_{dk} - I_{pk}) \beta e^{j\gamma} \\ (I_{dk} - I_{pk}) \beta e^{j\gamma} \end{pmatrix} \quad (2.3)$$

b_{0k} : kth node part of \mathbf{b}_0 vector

m_k : Unbalance mass (kg)

ϵ : Eccentricity of the unbalance mass (m)

δ : Angular location of unbalance

I_{dk} : Diametral mass moment of inertia of the kth beam element

I_{pk} : Polar mass moment of inertia of the kth beam element

β : Misalignment angle

γ : Misalignment angular location

We can solve Equation (2.2) to determine the steady-state response to the unbalance forces. Letting $\mathbf{q}(t)=\Re(\mathbf{q}_0 e^{j\Omega t})$ (where \mathbf{q}_0 is complex) gives:

$$\mathbf{q}_0 = [(\mathbf{K} - \Omega^2 \mathbf{M}) + j\Omega(\Omega \mathbf{G} + \mathbf{C})]^{-1} \Omega^2 \mathbf{b}_0 \quad (2.4)$$

Element matrices will be obtained in the following sections.

2.2. Finite Element Method

2.2.1. Beam Elements

We first introduce coordinate systems to apply equations of motion. When we set our coordinate conventions, we continue with deriving element matrices.

2.2.1.1. Coordinate Systems

We assumed the axis of rotation is z-axis , y-axis is the normal to the ground and x-axis was found by right-hand-rule. Figure 2.2.1 illustrate our convention.

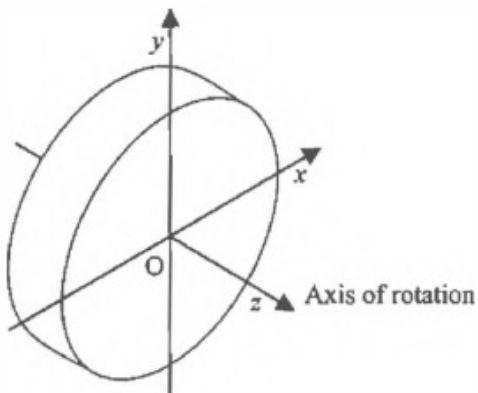


Figure 2.2.1: Right-handed coordinate set used in the analysis of rotor

The center of mass of the rotor is allowed to translate along axes Ox , Oy , and Oz by u , v , and w , respectively. The rotor also may rotate (by small amounts) about axes Ox and Oy by θ and ψ , respectively. Positive values of θ and ψ are in clockwise rotations. Rotation about axis Oz is represented by an angular displacement φ and an angular speed Ω .

2.2.1.2. *Shaft Line Model*

Every single node has 4 degrees of freedom ; translation and rotation about x and y-axes ; u , v , θ and ψ . Every beam elements are represented by two nodes, the beginning and the end, so the beam has 8 degrees of freedom .

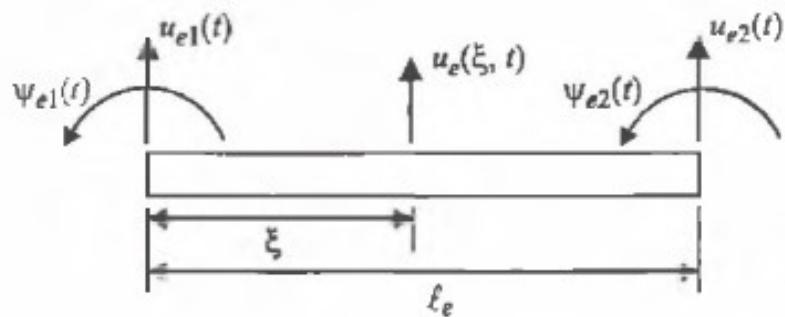


Figure 2.2.2: The local coordinates for the beam element , for x-axis or yz-plane

2.2.1.3. Euler-Bernoulli Beam Theory

Initially we will derive one plane vibration and then we will then give the total element matrices without derivation . The deflection of the element is given by the following interpolation functions:

$$u_e(\xi, t) = [N_{e1}(\xi) \quad N_{e2}(\xi) \quad N_{e3}(\xi) \quad N_{e4}(\xi)] \begin{pmatrix} u_{e1}(t) \\ \psi_{e1}(t) \\ u_{e2}(t) \\ \psi_{e2}(t) \end{pmatrix} \quad (2.5)$$

Where the shape functions $N_{ei}(\xi)$ are:

$$\begin{aligned} N_{e1}(\xi) &= \left(1 - 3\frac{\xi^2}{l_e^2} + 2\frac{\xi^3}{l_e^3}\right) & ; \quad N_{e2}(\xi) &= l_e \left(\frac{\xi}{l_e} - 2\frac{\xi^2}{l_e^2} + \frac{\xi^3}{l_e^3}\right) \\ N_{e3}(\xi) &= \left(3\frac{\xi^2}{l_e^2} - 2\frac{\xi^3}{l_e^3}\right) & ; \quad N_{e4}(\xi) &= l_e \left(\frac{-\xi^2}{l_e^2} + \frac{\xi^3}{l_e^3}\right) \end{aligned} \quad (2.6)$$

This interpolation function is a cubic polynomial in ξ and is the simplest polynomial that satisfies the conditions at the nodes $u_e(0)=u_{e1}$, $\frac{\partial u_e}{\partial \xi}(0)=\psi_{e1}$; $u_e(l_e)=u_{e2}$, $\frac{\partial u_e}{\partial \xi}=\psi_{e2}$. For $v_e(\xi, t)$ we use the same interpolation functions but the equation is little different:

$$v_e(\xi, t) = [N_{e1}(\xi) \quad -N_{e2}(\xi) \quad N_{e3}(\xi) \quad -N_{e4}(\xi)] \begin{pmatrix} u_{e1}(t) \\ \psi_{e1}(t) \\ u_{e2}(t) \\ \psi_{e2}(t) \end{pmatrix} \quad (2.7)$$

The strain energy U_e of a linear elastic material with elastic modulus E_e and the second moment of area I_e ($I_e=\pi r^4/4$, which is for circular shaft with radius r) is given by :

$$U_e = \frac{1}{2} E_e I_e \int_0^{l_e} \left(\frac{\partial^2 u_e(\xi, t)}{\partial \xi^2} \right)^2 d\xi \quad (2.8)$$

This is the case when the cross section does not vary with the beam length. If we put the $u_e(\xi, t)$ into the integral, the strain energy will form as it follows:

$$U_e = \frac{1}{2} \begin{pmatrix} u_{e1}(t) \\ \psi_{e1}(t) \\ u_{e2}(t) \\ \psi_{e2}(t) \end{pmatrix}^T \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{pmatrix} u_{e1}(t) \\ \psi_{e1}(t) \\ u_{e2}(t) \\ \psi_{e2}(t) \end{pmatrix} \quad (2.9)$$

Where :

$$k_{ij} = E_e I_e \int_0^{l_e} N_{ei}''(\xi) N_{ej}''(\xi) d\xi \quad (2.10)$$

So , the resulting stiffness matrix as it follows:

$$\mathbf{K}_e = \frac{E_e I_e}{l_e^3} \begin{bmatrix} 12 & 6l_e & -12 & 6l_e \\ 6l_e & 4l_e^2 & -6l_e & 2l_e^2 \\ -12 & -6l_e & 12 & -6l_e \\ 6l_e & 2l_e^2 & -6l_e & 4l_e^2 \end{bmatrix} \quad (2.11)$$

The mass matrix is calculated using the kinetic energy of the beam elements. Neglecting the rotational effects, the kinetic energy of the beam is given by:

$$T_e = \frac{1}{2} \int_0^{l_e} \rho_e A_e(\xi) \dot{u}_e^2(\xi, t) d\xi \quad (2.12)$$

Where ρ_e is the density of the material, A_e is the cross-sectional area of the beam.

Substituting the interpolations functions:

$$T_e = \frac{1}{2} \begin{pmatrix} \dot{u}_{e1}(t) \\ \dot{\psi}_{e1}(t) \\ \dot{u}_{e2}(t) \\ \dot{\psi}_{e2}(t) \end{pmatrix}^T \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{pmatrix} \dot{u}_{e1}(t) \\ \dot{\psi}_{e1}(t) \\ \dot{u}_{e2}(t) \\ \dot{\psi}_{e2}(t) \end{pmatrix} \quad (2.13)$$

Where :

$$m_{ij} = \rho_e A_e \int_0^{l_e} N_{ei}(\xi) N_{ej}(\xi) d\xi \quad (2.14)$$

As a result , the mass matrix is as follows:

$$\mathbf{M}_e = \frac{\rho_e A_e l_e}{420} \begin{bmatrix} 156 & 22l_e & 54 & -13l_e \\ 22l_e & 4l_e^2 & 13l_e & -3l_e^2 \\ 54 & 13l_e & 156 & -22l_e \\ -13l_e & -3l_e^2 & -22l_e & 4l_e^2 \end{bmatrix} \quad (2.15)$$

When we learn the algorithm, it is easier to obtain two-plane bending matrices. Next sub-chapters will give the total element matrices. Now, the local coordinate vector is $q_e = [u_1, v_1, \theta_1, \psi_1, u_2, v_2, \theta_2, \psi_2]^T$. So ,the element matrices are 8×8 square matrices.

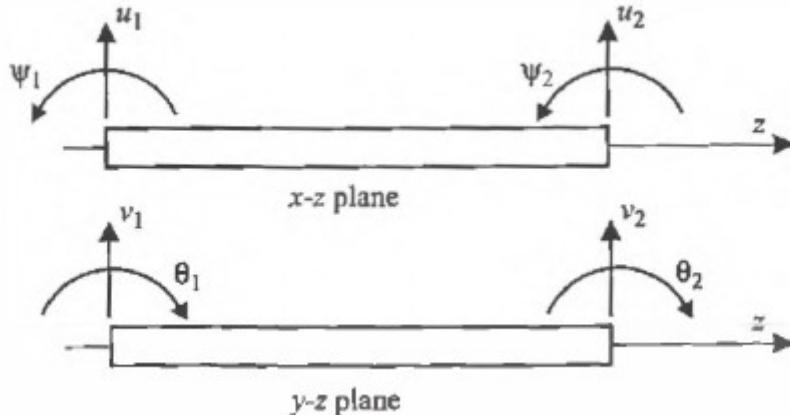


Figure 2.2.3: The local coordinates in the two bending planes

2.2.1.4. Mass Matrix

$$\mathbf{M}_e = \frac{\rho_e A_e l_e}{420} \begin{bmatrix} 156 & 0 & 0 & 22l_e & 54 & 0 & 0 & -13l_e \\ 0 & 156 & -22l_e & 0 & 0 & 54 & 13l_e & 0 \\ 0 & -22l_e & 4l_e^2 & 0 & 0 & -13l_e & -3l_e^2 & 0 \\ 22l_e & 0 & 0 & 4l_e^2 & 13l_e & 0 & 0 & -3l_e^2 \\ 54 & 0 & 0 & -13l_e & 156 & 0 & 0 & -22l_e \\ 0 & 54 & -13l_e & 0 & 0 & 156 & 22l_e & 0 \\ 0 & 13l_e & -3l_e^2 & 0 & 0 & 22l_e & 4l_e^2 & 0 \\ -13l_e & 0 & 0 & -3l_e^2 & -22l_e & 0 & 0 & 4l_e^2 \end{bmatrix} \quad (2.16)$$

2.2.1.5. Gyroscopic Matrix

$$\mathbf{G}_e = \frac{\rho_e I_e}{15l_e} \begin{bmatrix} 0 & 36 & -3l_e & 0 & 0 & -36 & -3l_e & 0 \\ -36 & 0 & 0 & -3l_e & 36 & 0 & 0 & -3l_e \\ 3l_e & 0 & 0 & 4l_e^2 & -3l_e & 0 & 0 & -l_e^2 \\ 0 & 3l_e & -4l_e^2 & 0 & 0 & -3l_e & l_e^2 & 0 \\ 0 & -36 & 3l_e & 0 & 0 & 36 & 3l_e & 0 \\ 36 & 0 & 0 & 3l_e & -36 & 0 & 0 & 3l_e \\ 3l_e & 0 & 0 & -l_e^2 & -3l_e & 0 & 0 & 4l_e^2 \\ 0 & 3l_e & l_e^2 & 0 & 0 & -3l_e & -4l_e^2 & 0 \end{bmatrix} \quad (2.17)$$

2.2.1.6. Stiffness Matrix

$$\mathbf{K}_e = \frac{E_e I_e}{l_e^3} \begin{bmatrix} 12 & 0 & 0 & 6l_e & -12 & 0 & 0 & 6l_e \\ 0 & 12 & -6l_e & 0 & 0 & -12 & -6l_e & 0 \\ 0 & -6l_e & 4l_e^2 & 0 & 0 & -6l_e & 2l_e^2 & 0 \\ 6l_e & 0 & 0 & 4l_e^2 & -6l_e & 0 & 0 & 2l_e^2 \\ -12 & 0 & 0 & -6l_e & 12 & 0 & 0 & -6l_e \\ 0 & -12 & -6l_e & 0 & 0 & 12 & 6l_e & 0 \\ 0 & -6l_e & 2l_e^2 & 0 & 0 & 6l_e & 4l_e^2 & 0 \\ 6l_e & 0 & 0 & 2l_e^2 & -6l_e & 0 & 0 & 4l_e^2 \end{bmatrix} \quad (2.18)$$

2.2.2. Disk Elements

The total kinetic energy of the disk element can be approximated by the following expression:

$$T_d = \frac{1}{2} m_d (\dot{u}^2 + \dot{v}^2) + \frac{1}{2} I_d (\dot{\theta}^2 + \dot{\psi}^2) + \frac{1}{2} I_p (\Omega^2 - 2\Omega\dot{\psi}\dot{\theta}) \quad (2.19)$$

When we substitute into Lagrange's equations:

$$\begin{Bmatrix} \frac{d}{dt} \left(\frac{\partial T_d}{\partial \dot{u}} \right) - \frac{\partial T_d}{\partial u} \\ \dots \\ \frac{d}{dt} \left(\frac{\partial T_d}{\partial \dot{\psi}} \right) - \frac{\partial T_d}{\partial \psi} \end{Bmatrix} = M_e \begin{Bmatrix} \ddot{u} \\ \ddot{v} \\ \ddot{\theta} \\ \ddot{\psi} \end{Bmatrix} + \Omega G_e \begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} \quad (2.20)$$

Where M_e is the disk mass matrix and G_e is the disk gyroscopic matrix and these are as shown below:

$$M_e = \begin{bmatrix} m_d & 0 & 0 & 0 \\ 0 & m_d & 0 & 0 \\ 0 & 0 & I_d & 0 \\ 0 & 0 & 0 & I_d \end{bmatrix} \quad (2.21)$$

Where m_d is the disk mass and I_d is the diametral mass moment of inertia of the disk.

$$\mathbf{G}_e = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_p \\ 0 & 0 & -I_p & 0 \end{bmatrix} \quad (2.22)$$

Where I_p is the polar moment of inertia of the disk.

2.2.3. Bearing Elements

The theory of the hydrodynamic bearing is based on the Reynolds' Equation. Analytical solutions can be determined for the short bearing which length to bush diameter ratio much lower than one ($L/D \ll 1$). The remaining assumptions other assumptions are listed below:

- The lubricant pressure is zero at the edges of the bearing
- The bearing is operating under steady conditions
- The lubricant properties do not vary substantially throughout the oil film
- The shaft does not tilt in the bearing

We will use short bearing in the project, so, following theory is based on short bearing assumption.

The total force applied to the bearing by the rotor is given by:

$$f = \frac{\pi D \Omega \eta L^3 \epsilon}{8 c^2 (1 - \epsilon^2)^2} \left(\left(\frac{16}{\pi^2} - 1 \right) + 1 \right)^{1/2} \quad (2.23)$$

Where:

f : Magnitude of the resultant force(N)

D : Diameter of the bushing (m)

L : Length of the bearing (m)

c : Clearance (m)

ϵ : Eccentricity , which is the ratio between the journal displacement and the radial clearance . It is assumed to be equal to 0.6 .

We assume that bearings model as node. So the matrices are 4×4 but ,according to short bearing assumption, shaft tilts are unconstrained . These matrices drop 2×2 , only u and v are constrained by bearings.

2.2.3.1. *Stiffness Matrix*

$$K_e = \frac{f}{c} \begin{bmatrix} a_{uu} & a_{uv} \\ a_{vu} & a_{vv} \end{bmatrix}$$

where;

$$\begin{aligned} a_{uu} &= h_0 \times 4(\pi^2(2-\epsilon^2) + 16\epsilon^2); \\ a_{uv} &= h_0 \times \frac{\pi(\pi^2(1-\epsilon^2)^2 - 16\epsilon^4)}{\epsilon\sqrt{1-\epsilon^2}}; \\ a_{vu} &= -h_0 \times \frac{\pi(\pi^2(1-\epsilon^2)(1+2\epsilon^2) + 32\epsilon^2(1+\epsilon^2))}{\epsilon\sqrt{1-\epsilon^2}}; \\ a_{vv} &= h_0 \times 4\left(\pi^2(1+2\epsilon^2) + \frac{32\epsilon^2(1+\epsilon^2)}{1-\epsilon^2}\right); \\ h_0 &= (\pi^2(1-\epsilon^2) + 16\epsilon^2)^{-3/2}; \end{aligned} \quad (2.24)$$

2.2.3.2. *Damping Matrix*

$$C_e = \frac{f}{c\Omega} \begin{bmatrix} b_{uu} & b_{uv} \\ b_{vu} & b_{vv} \end{bmatrix}$$

where;

$$\begin{aligned} b_{uu} &= h_0 \times \frac{2\pi\sqrt{1-\epsilon^2}(\pi^2(1+2\epsilon^2)-16\epsilon^2)}{\epsilon}, \\ b_{uv} &= h_0 \times 8(\pi^2(1+2\epsilon^2)-16\epsilon^2); b_{vu} = b_{uv}; \\ b_{vv} &= h_0 \times \frac{2\pi(\pi^2(1-\epsilon^2)^2-16\epsilon^2)}{\epsilon\sqrt{1-\epsilon^2}}; \\ h_0 &= (\pi^2(1-\epsilon^2)+16\epsilon^2)^{-3/2}; \end{aligned} \quad (2.25)$$

2.3. Strength of Materials

Strength considerations are one of the most important part of the mechanical design. We have to ensure safety of a mechanical design.

Major stresses related to the rotor dynamics are centrifugal stresses due to the rotation and bending stresses due to the unbalance forces and moments.

Formulation of the centrifugal stress is different for the thin disk and long cylinder. Also centrifugal stresses are critical mostly inner portion and outmost portion of the beam and disk. We will explain in more detail in following chapters.

Rotor design isn't different from other mechanical structures except rotation effects. Due to the nature of the rotor vibration, static strength considerations are not enough to finish design , we have to check also for fatigue.

2.3.1. Centrifugal Stresses

2.3.1.1. Thin Disk

$$\left\{ \begin{array}{l} \sigma_r = \frac{3+v}{8} \sigma_0 \left(1 + \beta^2 - \frac{\beta^2}{\rho^2} \right) \\ \sigma_t = \frac{3+v}{8} \sigma_0 \left(1 + \beta^2 + \frac{\beta^2}{\rho^2} - \frac{1+3v}{3+v} \rho^2 \right) \end{array} \right.$$

where;

$$\sigma_0 = \Gamma \omega^2 r_e^2 \quad (2.26)$$

$$\beta = \frac{r_i}{r_e}$$

$$\rho = \frac{r}{r_e}$$

Where;

Γ : Density of the material (kg/m^3) (7800 kg/m^3 for steels)

v : Poisson ratio (0.3 for steel)

r_i : Inner radius (m) (also outer radius of the beam)

r_e : Outer radius (m)

σ_r : Radial stresses

σ_t : Tangential stresses

2.3.1.2. Cylindrical Body

$$\left\{ \begin{array}{l} \sigma_r = \frac{3-2v}{8-8v} \sigma_0 \left(1 + \beta^2 - \frac{\beta^2}{\rho^2} - \rho^2 \right) \\ \sigma_t = \frac{3-2v}{1-v} \sigma_0 \left(1 + \beta^2 + \frac{\beta^2}{\rho^2} - \frac{1+2v}{3-2v} \rho^2 \right) \\ \sigma_z = \frac{v\sigma_0}{4-4v} \left(1 + \beta^2 - 2\rho^2 \right) \end{array} \right. \quad (2.27)$$

2.3.2. Bending Stresses

$$\sigma_z(R, z) = (\sigma_z)_{max} = -E R u''(z) \quad (2.28)$$

Where , R is the radius of the shaft.

According to Equation (2.5):

$$\begin{aligned} u_e(\xi) &= \left(1 - 3\frac{\xi^2}{l_e^2} + 2\frac{\xi^3}{l_e^3}\right)u_1 + l_e\left(\frac{\xi}{l_e} - 2\frac{\xi^2}{l_e^2} + \frac{\xi^3}{l_e^3}\right)\psi_1 \\ &\quad + \left(3\frac{\xi^2}{l_e^2} - 2\frac{\xi^3}{l_e^3}\right)u_2 + l_e\left(-\frac{\xi^2}{l_e^2} + \frac{\xi^3}{l_e^3}\right)\psi_2 \end{aligned} \quad (2.29)$$

To determine the deflections , we have to find the second derivatives:

$$\begin{aligned} u_e''(\xi) &= \left(-\frac{6}{l_e^2} + 12\frac{\xi}{l_e^3}\right)u_1 + l_e\left(\frac{-4}{l_e^2} + 6\frac{\xi}{l_e^3}\right)\psi_1 \\ &\quad + \left(\frac{6}{l_e^2} - 12\frac{\xi}{l_e^3}\right)u_2 + l_e\left(\frac{-2}{l_e^2} + 6\frac{\xi}{l_e^3}\right)\psi_2 \end{aligned} \quad (2.30)$$

Letting $\xi = 0$ in Equation (2.30) to determine the bending stress at the left end of the beam; letting $\xi = l_e$ to determine the bending stress at the right end of the beam. This gives:

$$\begin{aligned} u''_{z1} &= (-6u_1 + 6u_2 - 4l_e\psi_1 - 2l_e\psi_2)/l_e^2 \\ u''_{z2} &= (6u_1 - 6u_2 + 2l_e\psi_1 + 4l_e\psi_2)/l_e^2 \\ v''_{z1} &= (-6v_1 + 6v_2 + 4l_e\theta_1 + 2l_e\theta_2)/l_e^2 \\ v''_{z2} &= (6v_1 - 6v_2 - 2l_e\theta_1 - 4l_e\theta_2)/l_e^2 \end{aligned} \quad (2.31)$$

2.3.3. Von Mises Failure Criterion

Von mises criterion is the most popular criterion to determine failure point of metals at the 3-axis stress state . Formulation is as shown below:

$$\sigma_e = [\sigma_r^2 + \sigma_t^2 + \sigma_z^2 - \sigma_r^2 \sigma_t^2 - \sigma_r^2 \sigma_z^2 - \sigma_z^2 \sigma_t^2]^{1/2} \quad (2.32)$$

σ_e : Von Mises Stress

2.3.4. Fatigue Failure with Soderberg Criterion

We chose Soderberg criterion to check fatigue safety because it does not require any other stress control.

$$\frac{\sigma_a}{S_e} + \frac{\sigma_m}{S_y} < 1 \quad (2.33)$$

σ_a : Alternating Stress , in our case it is bending

σ_m : Mean Stress , in our case it is centrifugal stresses

S_e : Endurance limit for the material.

S_y : Yield strength

2.3.4.1. Endurance Limit with Modifying Factors

We have chosen as a material AISI 4340 Q&T (315 °C) steel . Its ultimate strength is 1720 Mpa , yield strength is 1590 Mpa , endurance limit without modifying factors is 700 Mpa .

$$S_e = k_a k_b k_c S_e' \quad (2.34)$$

S_e' : Endurance limit without modifying factors.

k_a : surface factor (is assumed to be 0.75)

k_b : size factor (is assumed to be 0.63)

k_c : loading factor (for bending it is 1)

2.3.4.2. Stress Concentration

We assumed that disks are attached with a sled runner key and its stress concentration factor is 1.38:

$$\sigma = K_t \sigma_{nom} \quad (2.34)$$

σ_{nom} : Nominal Stress

K_t : Stress concentration factor (1.38)

2.4. PW4000 Engine

This engine has 4 LP, 11 HP compressor ; 2 HP , 4 LP turbine. We assumed that compressor disks thickness are 5 cm , turbine disks are 10 cm. Outer diameters of HP compressor disks are 60 cm, LP compressors' are 50cm, HP turbines' are 40cm , LP turbines' are 100cm .The shaft is 4 m long and divided into 29 elements. Length of each element is represented by the following code sentence:

```
l1=0.025;  
l2=0.1;  
l3=0.175;  
l4=0.2;  
l5=0.15;  
l6=0.05;  
l=[l1 l2*ones(1,14) l3 l4*ones(1,6) l5 l4*ones(1,5) l6];
```

2.5. Optimization Formulation

The optimization variables are 29 beam diameters and 2 bearing clearances .

$$\begin{aligned}
& \min_{x \in \mathbb{R}^n} f(x) \\
\text{subject to } & \begin{aligned} g_i(x) \leq 0, & i=1, \dots, m \\ h_j(x) = 0, & j=1, \dots, p \\ x^L \leq x \leq x^{(U)} \end{aligned} \tag{2.35}
\end{aligned}$$

$f(x)$: Objective function ,in our case it is potential energy.

It is given in the “obj_func.m” file

$g_i(x)$: Inequality constraints

$h_j(x)$: Equality constraints

These are calculated by “nonlin_cons.m” file and includes deflection, stress and fatigue limits.

$x^{(L)}$ and $x^{(U)}$: Lower and upper bounds.

2.6. Genetic Algorithm

Genetic Algorithms (GA) are powerful optimization techniques inspired by the process of natural evolution, as originally proposed by John Holland in the 1970s. They fall under the broader category of evolutionary algorithms and are particularly useful for solving complex, non-linear, and multidimensional problems where traditional gradient-based methods fail or are inefficient.

GA simulate the process of biological evolution using a set of operations that mimic natural selection, reproduction, and mutation. The core idea is to evolve a population of candidate solutions over multiple generations to approach an optimal or near-optimal solution.

1) Initialization

The algorithm begins by generating an initial population of individuals (also called chromosomes). Each individual represents a possible solution to the optimization problem. Typically, individuals are encoded as vectors of variables (binary, integer, or real-valued).

- Population size (e.g., 50–200) is a key parameter.
- Initialization can be fully random or guided (e.g., within known feasible bounds).

2) Fitness Evaluation

Each individual is evaluated using a fitness function. This function assigns a numerical value that indicates how well the solution satisfies the objective of the problem.

For example:

- In structural optimization, the fitness might relate to weight minimization.
- In scheduling, it might represent total time or cost.

A higher fitness value typically implies a better solution.

3) Selection

Based on the fitness scores, a selection process chooses individuals to become parents and contribute to the next generation.

Common selection methods:

- Roulette Wheel Selection: Probability of selection is proportional to fitness.
- Tournament Selection: Random subsets are created and the best among each group is selected.
- Rank Selection: Individuals are ranked, and probabilities assigned accordingly.

The goal is to favor better solutions while preserving genetic diversity.

4) Crossover

In this step, selected parent solutions are paired and recombined to form new offspring.

The crossover mimics the exchange of genetic material in biological reproduction.

- Single-point crossover: One cut point; parts swapped.
- Two-point crossover: Two cut points; middle segments exchanged.

- Uniform crossover: Each gene has an independent chance of swapping.

Crossover allows the algorithm to explore new regions of the solution space by combining partial traits from different parents.

5) Mutation

To avoid premature convergence and loss of diversity, random mutations are applied to some genes in the offspring population.

- Mutation rate is usually low (e.g., 0.5%–2%).
- Mutation types depend on encoding (e.g., bit flip for binary, small perturbation for real values).

Mutation helps the algorithm escape local optima and explore unvisited areas of the search space.

6) Replacement (Survivor Selection)

After crossover and mutation, the new offspring are introduced into the population.

Common strategies:

- Generational Replacement: Entire old population is replaced.
- Elitism: Best individuals from the current generation are preserved.
- Steady-State Replacement: Only a few individuals are replaced each generation.

Maintaining elite individuals ensures that the best solutions are not lost during stochastic operations.

7) Termination

The algorithm continues iterating through evaluation → selection → reproduction until a termination condition is met. Common stopping criteria include:

- A fixed number of generations
- Convergence of fitness values

- Computational budget (time, evaluations)

At the end, the best individual (or set of individuals) is reported as the solution.

3. SOFTWARE IMPLEMENTATION

All softwares were given in the Appendices chapter.

Main program is “rotor_opt_gen_alg.m”. It establishes the general optimization structures, sets GA parameters, defines constraints.

Program of the GA is “gen_alg.m”. This program operate the genetic algorithm. Usage guide of this program was given in Appendices section.

“assembly.m” file assembles all the matrices into single big matrix. It needs all “bearing_damp_mtx.m” , “bearing_stif_mtx.m” , “disk_mass_mtx.m” , “disk_gyr_mtx.m” , “mass_mtx.m” , “stif_mtx.m” . These files calculates element matrices.

“nonlin_cons.m” contains nonlinear constraints.

“cylinder_bending_stress.m” , “cylinder_cent_stress.m” , “disk_cent_stress.m” calculates stresses acting on the beams and disks.

“soderberg_beam.m” calculates compliance with the Soderberg criterion.

“obj_func.m” calculates the objective function.

“picotor.m” draws rotor. This code is given by the book in Ref. 1.

“picotor_set.m” gives the rotor geometry to “picotor.m” .

4. RESULTS AND DISCUSSION

When we set the algorithm as it passes through “rotor_opt_gen_alg.m” :

```
options = struct(...
```

```
'PopulationSize',    80, ...
'Generations',       1000, ...
'CrossoverRate',    0.7, ...
'MutationRate',     0.05, ...
'EliteCount',        4, ...
'Display',           'iter',...
'SelectionMethod',   'tournament',...
);
```

and lower and upper bounds intuitively :

```
lb= [0.1*ones(29,1); 0 ;0];
ub=[0.3*ones(29,1); 0.002 ;0.002 ];
```

We are ready to perform the algorithm.

Results are as shown below:

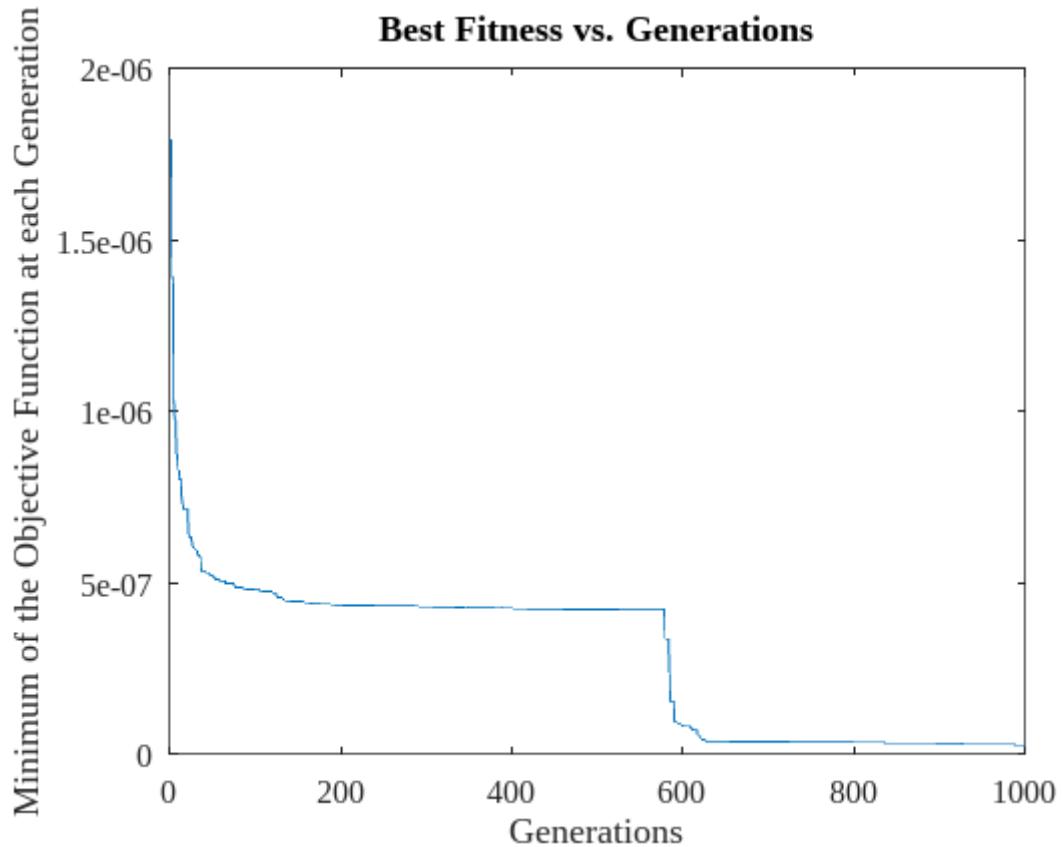


Figure 4.1: Minimum of the objective function in each generation vs. Generation

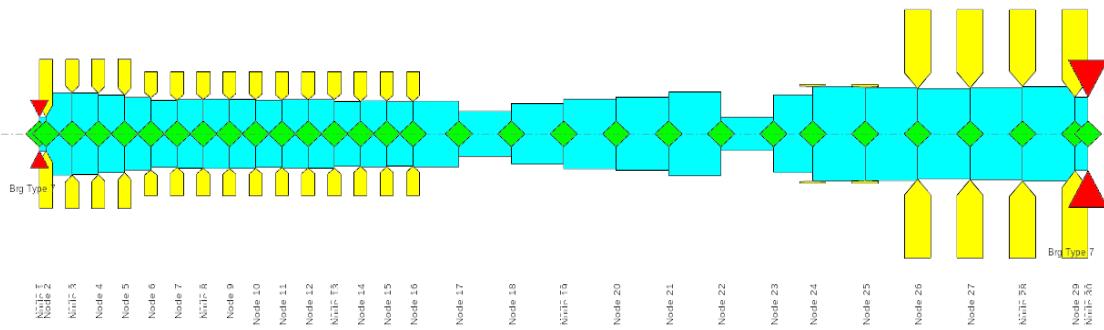


Figure 4.2: Rotor geometry

Diameters of the beam elements and bearing clearances can be found by the algorithm is given by “output.txt” file in Appendices.

As can be seen in Figure 4.1, algorithm first converges an arbitrary value but ,maybe from some mutations, algorithm found another solution and converged to a new solution.

5. CONCLUSION

As it can be seen from results, GA works well for rotor design problem. Algorithm converges after ~ 100 generations, but if the algorithm find a new local minimum area ,algorithm converges about this point,if we want the algorithm to give more accurate results, if we want to minimize the risk of getting stuck in local minimums, we need to increase the number of generations and run the algorithm for longer periods of time.

Obviously ,there are several imperfections at the side of modeling because we used Euler-Bernoulli beam theory, if we had used Timoshenko theory , it would have been more precise calculation. But in the long beams, the effect of shear force and rotary inertia are negligible, so we can use this E-BBT for design purpose. As a good further improvement , we can count this features:

- We can be taken hollow and conical beam elements.
- The program can be extended to perform optimization for any rotor structure entered and any given angular speed.
- Material models can be improved.
- Transient analysis and running through critical speed cases can be investigate.
- Effect of seals, torque and axial forces can be investigated.
- More complex bearing models can be added.
- Algorithm can be run for longer times for avoiding to stuck local minima.

6. REFERENCES

1. Michael I. Friswell , John E.T. Penny ,Seamus D. Garvey , Arthur W. Lees ;Dynamics of Rotating Machines , 2010, Cambridge University Press
2. Vincenzo Vullo ,Francesco Vivio ;Rotors: Stress Analysis and Design ,2013, Springer
3. Richard G. Budynas,J. Keith Nisbett, Shigley's Mechanical Engineering Design,Tenth Edition, McGraw Hill.
4. https://en.wikipedia.org/wiki/Pratt_%26_Whitney_PW4000

7. APPENDICES

%rotor_opt_gen_alg.m

```
clc;clear all;close all;

options = struct(...
    'PopulationSize',     80, ...
    'Generations',       200, ...
    'CrossoverRate',     0.7, ...
    'MutationRate',       0.05, ...
    'EliteCount',         4, ...
    'Display',            'iter',...
    'SelectionMethod',   'tournament',...
);

lb= [0.1*ones(29,1); 0 ;0];
```

```

ub=[0.4*ones(29,1); 0.002 ;0.002 ];

nvars = 31;

[x_opt, fval, exitflag, output,history] = gen_alg
(@obj_func, nvars, [] , [] , [], [], lb, ub,
@nonlin_cons,options);

[cc,~]=nonlin_cons(x_opt);
c=cc';
c_logic=(c<=0);
save("history_output.mat", "history");
save("nonlin_cons_check.mat", "c");
save("nonlin_cons_check_logic.mat", "c_logic");

disp("Optimal solution:");
disp(x_opt);
disp("Minimum value of Objective Function:");
disp(fval);
disp("Nonlinear Constraints:");
disp(c);
disp("Are constraints provided");

```

```

disp(c_logic);

Best_Fitness=[];
Generations=1:options.Generations;
for ii=1:options.Generations
    Best_Fitness(ii)=min(history(ii).fitness);
end

plot(Generations,Best_Fitness);
xlabel("Generations");
ylabel("Minimum of the Objective Function at each Generation");
title("Best Fitness vs. Generations");
print("Best Fitness vs Generations", '-dpng');

%%%%%%%%%%%%%%%
% gen_alg.m

function [best_x, best_fval, exitflag, output, history] =
gen_alg(objfun, nvars, A, b, Aeq, beq, lb, ub, nonlcon,
options)

% === Default Options ===
if nargin < 10, options = struct(); end

```

```

    if ~isfield(options, "PopulationSize"),
options.PopulationSize = 50; end

    if ~isfield(options, "Generations"), options.Generations
= 100; end

    if ~isfield(options, "CrossoverRate"),
options.CrossoverRate = 0.8; end

    if ~isfield(options, "MutationRate"),
options.MutationRate = 0.05; end

    if ~isfield(options, "EliteCount"), options.EliteCount =
2; end

    if ~isfield(options, "Display"), options.Display = "off";
end

    if ~isfield(options, "SelectionMethod"),
options.SelectionMethod = "tournament"; end

% === Initial Population ===

popSize = options.PopulationSize;

pop = repmat(lb', popSize, 1) + rand(popSize, nvars) .* ...
(repmat((ub-lb)', popSize, 1));

fitness = inf(popSize, 1);

history = struct('generation', {}, 'population', {},
'fitness', {});

for gen = 1:options.Generations

```

```

% fitness calculation

for i = 1:popSize

    xi = pop(i, :)';
    if is_feasible(xi, A, b, Aeq, beq, lb, ub, nonlcon)
        fitness(i) = objfun(xi);
    else
        fitness(i) = inf;
    end

end

history(gen).generation = gen;
history(gen).population = pop;
history(gen).fitness = fitness;

if strcmpi(options.Display, "iter")
    [best_gen_fval, best_idx] = min(fitness);
    fprintf("Generation %3d: Best Fitness = %.6f\n", gen,
best_gen_fval);

end

[~, idx] = sort(fitness);
elites = pop(idx(1:options.EliteCount), :);

```

```

newPop = elites;

while size(newPop,1) < popSize

    p1 = select_parent(pop, fitness,
options.SelectionMethod);

    p2 = select_parent(pop, fitness,
options.SelectionMethod);

if rand < options.CrossoverRate

    [c1, c2] = crossover(p1, p2);

else

    c1 = p1; c2 = p2;

end

c1 = mutate(c1, lb, ub, options.MutationRate);
c2 = mutate(c2, lb, ub, options.MutationRate);

newPop = [newPop; c1; c2];

end

pop = newPop(1:popSize, :);

end

final_fitness = inf(popSize, 1);

for i = 1:popSize

```

```

xi = pop(i, :)';
if is_feasible(xi, A, b, Aeq, beq, lb, ub, nonlcon)
    final_fitness(i) = objfun(xi);
end
end

[best_fval, best_idx] = min(final_fitness);
best_x = pop(best_idx, :)';
exitflag = 1;
output = struct();
output.generations = options.Generations;
output.funccount = options.Generations * popSize;
output.message = 'GA terminated successfully after
reaching maximum generations.';
end

function ok = is_feasible(x, A, b, Aeq, beq, lb, ub,
nonlcon)
ok = true;
if ~isempty(A) && any(A*x > b + 1e-6), ok = false;
return; end
if ~isempty(Aeq) && any(abs(Aeq*x - beq) > 1e-6), ok =
false; return; end

```

```

    if any(x < lb - 1e-6) || any(x > ub + 1e-6), ok = false;
return; end

if ~isempty(nonlcon)

    [c, ceq] = nonlcon(x);

    if any(c > 1e-6) || any(abs(ceq) > 1e-6), ok = false;
return; end

end

```

```

function sel = select_parent(pop, fitness, method)

switch lower(method)

case 'tournament'

    idx = randperm(size(pop,1), 2);

    if fitness(idx(1)) < fitness(idx(2))

        sel = pop(idx(1), :);

    else

        sel = pop(idx(2), :);

    end

case 'roulette'

    fit = 1./(fitness + 1e-12);

    probs = fit / sum(fit);

    cumprobs = cumsum(probs);

    r = rand;

    idx = find(cumprobs >= r, 1);

```

```

sel = pop(idx, :);

case 'rank'

[~, sortIdx] = sort(fitness);

ranks = 1:length(fitness);

ranks = ranks(sortIdx);

probs = ranks / sum(ranks);

cumprobs = cumsum(probs);

r = rand;

idx = find(cumprobs >= r, 1);

sel = pop(idx, :);

case 'random'

idx = randi(size(pop,1));

sel = pop(idx, :);

otherwise

error('Unknown selection method: %s', method);

end

end

function [c1, c2] = crossover(p1, p2)

alpha = rand;

c1 = alpha*p1 + (1-alpha)*p2;

c2 = alpha*p2 + (1-alpha)*p1;

end

```

```

function mutant = mutate(ind, lb, ub, rate)
    mutant = ind;
    for i = 1:length(ind)
        if rand < rate
            mutant(i) = lb(i) + rand * (ub(i) - lb(i));
        end
    end
end

%%%%%%%%%%%%%%%
%assembly.m

```

```

function [M,G,C,K,q]=assembly(x)

d=x(1:29);
clearence1=x(30);
clearence2=x(31);

b0=zeros(120,1); %unbalance vector
% 0.011141 kg.m unbalance occur according to standarts
%divide to 21 disk, 5.3052e-4 kg.m for one disk
%assume all of them in x direciton
for a=[2:16 24:29]

```

```

b0(4*a-3) += 5.3052e-04;
end

omg=100*pi;
angular_vel=omg;
dynamic_visc=0.4; %Pa.s SAE40 oil 25 C
angular_vel=100*pi;
eccentricity=0.6;
E=210*10^9; %GPa
rho=7800; %kg/m^3
rho1=8190;%"Inconel 718" for disk
l1=0.025;
l2=0.1;
l3=0.175;
l4=0.2;
l5=0.15;
l6=0.05;
l=[l1 l2*ones(1,14) l3 l4*ones(1,6) l5 l4*ones(1,5) l6];
%beam element lengths
t=[0.05*ones(1,15) 0.1*ones(1,6)];%disk thicknesses
d_o=[0.60*ones(1,4) 0.50*ones(1,11) 0.40 0.40 1*ones(1,4)];
%disk outer diameters
M=zeros(120);
C=zeros(120);

```

```

G=zeros(120);

K=zeros(120);

%bearing matrices

C(1:2,1:2)=bearing_damp_mtx(d(1),dynamic_visc,clearence1,angular_vel,eccentricity);

C(117:118,117:118)=bearing_damp_mtx(d(29),dynamic_visc,clearence2,angular_vel,eccentricity);

K(1:2,1:2)=bearing_stif_mtx(d(1),dynamic_visc,clearence1,angular_vel,eccentricity);

K(117:118,117:118)=bearing_stif_mtx(d(29),dynamic_visc,clearence2,angular_vel,eccentricity);

%beam elements

for i=1:29

M((4*i-3):(4*i+4),(4*i-3):(4*i+4))=mass_mtx(rho,d(i),l(i));

K((4*i-3):(4*i+4),(4*i-3):(4*i+4))=stif_mtx(E,d(i),l(i));

end

%disk elements

jj=1;%disk counting

for i=[2:16 24:29]

M((4*i-3):4*i,
(4*i-3):4*i)=disk_mass_mtx(rho1,d_o(jj),d(i),t(jj));

```

```

G((4*i-3):4*i,
(4*i-3):4*i)=disk_gyr_mtx(rho1,d_o(jj),d(i),t(jj));

jj+=1;

end

% DEFLECTION

q=abs(inv([(K-omg^2*M)+j*omg*(omg*G+C)])*(omg^2*b0));

end

%%%%%%%%%%%%%%%
%mass_mtx.m

function M=mass_mtx(rho,d,l)

M=[156      0      0      22*l     54      0      0      -13*l ;
    0      156   -22*l     0      0      54      13*l     0      ;
    0     -22*l   4*l^2     0      0     -13*l   -3*l^2     0      ;
    22*l     0      0      4*l^2     13*l     0      0      -3*l^2;
    54      0      0      13*l     156      0      0      -22*l ;
    0      54     -13*l     0      0      156     22*l     0      ;
    0     13*l   -3*l^2     0      0     22*l     4*l^2     0      ;
    -13*l     0      0     -3*l^2   -22*l     0      0      4*l^2];;

A=pi*d^2/4;

```

```

M=rho*A*l/420*M;

end

%%%%%%%%%%%%%%%
%stif_mtx.m

function K=stif_mtx(E,d,l)

K = [12     0      0      6*l      -12      0      0      6*l;
      0     12     -6*l      0      0      -12     -6*l      0;
      0    -6*l     4*l^2      0      0      6*l     2*l^2      0;
      6*l      0      0      4*l^2     -6*l      0      0      2*l^2;
     -12      0      0     -6*l      12      0      0     -6*l;
      0    -12     6*l      0      0      12      6*l      0;
      0    -6*l     2*l^2      0      0      6*l     4*l^2      0;
      6*l      0      0      2*l^2     -6*l      0      0      4*l^2];

inertia=pi*d^4/64;
K = E*inertia/(l^3)*K;

end

%%%%%%%%%%%%%%%
%disk_gyr_mtx.m

```

```

function G=disk_gyr_mtx(rho,d_o,d_i,t)

md=rho * pi * t * (d_o^2 - d_i^2) / 4;
Ip=md * (d_o^2 + d_i^2) / 8;

G=[ 0      0      0      0;
    0      0      0      0;
    0      0      0      -Ip;
    0      0      -Ip     0];

end
%%%%%%%
%disk_mass_mtx.m

function M=disk_mass_mtx(rho,d_o,d_i,t)

md=rho * pi * t * (d_o^2 - d_i^2) / 4;
Ip=md * (d_o^2 + d_i^2) / 8;
Id=1/2 * Ip + 1/12 * md * t^2;

M=[md      0      0      0;
   0      0      0      0;
   0      0      0      -Ip;
   0      0      -Ip     0];

```

```

0      md      0      0;
0      0      Id      0;
0      0      0      Id];
end

%%%%%%%%%%%%%%%
%bearing_stif_mtx.m

function K= bearing_stif_mtx( d_bush , dynamic_visc ,
clearence,angular_vel ,eccentricity )

d=d_bush;
%l=bearing_length= d_bush/10 => short bearing assumption
eta=dynamic_visc;
c=clearence;
ep=eccentricity;
omg=angular_vel;

f=(pi*10^-3*d^4*omg*eta)/(8*c^2)*(ep/(1-ep^2)^2)*[(16/pi^2-
1)*ep^2+1]^(1/2);

h0=[pi^2*(1-ep^2)+16*ep^2]^-3/2;

auu=h0*4*(pi^2*(2-ep^2)+16*ep^2);

```

```

auv=h0*(pi*(pi^2*(1-ep^2)^2)-16*ep^4)/(ep*sqrt(1-ep^2));
avu=-h0*(pi*(pi^2*(1-ep^2)*(1+2*ep^2)+32*ep^2*(1+ep^2)))/
(ep*sqrt(1-ep^2));
avv=h0*4*(pi^2*(1+2*ep^2)+(32*ep^2*(1+ep^2))/(1-ep^2));

K=(f/(c))*[auu auv;avu avv];

end

%%%%%%%%%%%%%%%
%bearing_damp_mtx.m

function
C=bearing_damp_mtx(d_bush,dynamic_visc,clearence,angular_ve
l,eccentricity)

d=d_bush;
%l=bearing_length= d_bush/10 => short bearing assumption
eta=dynamic_visc;
c=clearence;
ep=eccentricity;
%ep=0.6;%eccentricity;
omg=angular_vel;
%omg=100*pi;%=3000rpm angular_vel;

```

```

f=(pi^10^-3*d^4*omg*eta)/(8*c^2)*(ep/(1-ep^2)^2)*[(16/pi^2-
1)*ep^2+1]^(1/2);

h0=[pi^2*(1-ep^2)+16*ep^2]^-3/2;
buu=h0*(2*pi*sqrt(1-ep^2)*(pi^2*(1+2*ep^2)-16*ep^2))/ep;
buv=-h0*8*(pi^2*(1+2*ep^2)-16*ep^2);
bvv=h0*(2*pi*(pi^2*(1-ep^2)^2+48*ep^2))/(ep*sqrt(1-ep^2));

C=(f/(c*omg))*[buu buv; buv bv];

```

end

%%%%%%%%%%%%%%

%disk_cent_stress.m

```
function [rho_sqrt_beta,rho_1]=disk_cent_stress(x)
```

%rho_sqrt_beta means centrifugal stress at rho=sqrt(beta)

%rho_1 means stress at rho=1 , r=re

%both of them are 1*21 vector

%ri=inner radius disk

%re=outer radius of disk

%d_i is disk ,inner diameter vector 1*21

```

d_i=x([1:15 24:29])';
d_o=[0.60*ones(1,4) 0.50*ones(1,11) 0.40 0.40 1*ones(1,4)];
%disk outer diameters

re=d_o/2;%disk outer radius

nu=0.3;%poisson ratio


ri=d_i/2;
b=ri./re;
rho1=sqrt(ri./re);
rho2=1;

gama=8190;%kg/m^3 density Inconel 718
omg=100*pi;%angular velocity 3000 rpm


sigma_0=gama*omg^2*re.^2;
sigma_r=@(rho) (3+nu)/8*sigma_0.* (1+b.^2-b.^2/rho.^2-
rho.^2);
sigma_t=@(rho) (3+nu)/8*sigma_0.* (1+b.^2+b.^2/rho.^2-
(1+3*nu)/(3+nu)*rho.^2);
von_mises=@(rho) sqrt(sigma_r(rho).^2+sigma_t(rho).^2-
sigma_r(rho).*sigma_t(rho));

rho_sqrt_beta=von_mises(rho1);
rho_1=von_mises(rho2);

```

```

end

%%%%%%%%%%%%%%%
%cylinder_cent_stress.m

function [s_i,s_o]=cylinder_cent_stress(x)

%s_i stress at center
%s_o stress at outside
%Soderberg criterion will be applied
%both of them are 1*21 vector
%ri=inner radius disk
%re=outer radius of disk
%d is beam diameter vector 1*29

d=x([1:29])';
re=d/2;%beam radius
nu=0.3;%poisson ratio
ri=0;% inside is full
b=0;%b=ri/re
rho1=0;%inside part of beam
rho2=1;%outside part of beam
gama=7800;%kg/m^3 density

```

```

omg=100*pi;%angular velocity 3000 rpm

sigma_0=gama*omg^2*re.^2;
sigma_r=@(rho)
(3-2*nu)/(8*(1-nu))*sigma_0.* (1+b^2-b^2/rho^2-rho^2);
sigma_t=@(rho) (3-2*nu)/(1-nu)*sigma_0*(1+b^2+b^2/rho^2-
(1+2*nu)/(3-2*nu)*rho^2);
sigma_z=@(rho) nu/(4*(1-nu))*sigma_0*(1+b^2-2*rho^2);
von_mises=@(rho)
sqrt(sigma_r(rho).^2+sigma_t(rho).^2+sigma_z(rho).^2-...
sigma_r(rho).*sigma_t(rho)-
sigma_r(rho).*sigma_z(rho)-...
sigma_z(rho).*sigma_t(rho));

s_i=von_mises(rho1);
s_o=von_mises(rho2);

end
%%%%%%%%%%%%%%%
%cylinder_bending_stress.m

function sigma_z=cylinder_bending_stress(x)

%d=x(1:29);

```

```

%clearance1=x(30);
%clearance2=x(31);

E=210*10^9;%elastic modulus

l1=0.025;
l2=0.1;
l3=0.175;
l4=0.2;
l5=0.15;
l6=0.05;

l=[l1 l2*ones(1,14) l3 l4*ones(1,6) l5 l4*ones(1,5) l6];
%beam element lengths

R=x(1:29)/2;%element radii column vector

Kt=1.38;%sled runner key stress concentration factor

[~,~,~,~,q]=assembly(x);

uz1pp=zeros(29,1);
vz1pp=zeros(29,1);
uz1p=zeros(29,1);
vz1p=zeros(29,1);
uz1=zeros(29,1);
vz1=zeros(29,1);
r1pp=zeros(29,1);

```

```

r1p=zeros(29,1);

r1=zeros(29,1);

for ii=1:29

uz1pp(ii)=(-6*q(4*ii-3)+6*q(4*ii+1)-4*l(ii)*q(4*ii-1)-
2*l(ii)*q(4*ii+2))/l(ii)^2;

vz1pp(ii)=(-6*q(4*ii-2)+6*q(4*ii+2)+4*l(ii)*q(4*ii)
+2*l(ii)*q(4*ii+4))/l(ii)^2;

uz1p(ii)=q(4*ii-1);%psi slope

vz1p(ii)=-q(4*ii);%-theta

uz1(ii)=q(4*ii-3);

vz1(ii)=q(4*ii-2);

r1(ii)=sqrt(uz1(ii)^2+vz1(ii)^2);

r1p(ii)=(uz1(ii)*uz1p(ii)+vz1(ii)*vz1p(ii))/r1(ii);

r1pp(ii)=uz1(ii)*uz1pp(ii)+uz1p(ii)^2+vz1(ii)*vz1pp(ii)
+vz1p(ii)^2;

r1pp(ii)=r1p(ii)^2;

r1pp(ii)/=r1(ii);

end

sigma_z=Kt*E*R.*r1pp;

```

```

end

%%%%%%%%%%%%%%%
%soderberg_beam.m

function S=soderberg_beam(x)

S_y1=1590*10^6;%N/m^2 AISI 4340 Q&T 315C
S_y2=850*10^6;%Inconel 718 for all disks @700C celcius
Se1=700*10^6;%Se' endurance limit of AISI4340 Q&T @315C
ka=0.75;
kb=0.633;
Se=ka*kb*Se1;%endurance limit with modifying factors
Sut=1720;%ultimate strength of AISI4340 Q&T @315C

n=1;%safety factor

sigma_a=cylinder_bending_stress(x);%fluctuating part
[~,sigma_m]=cylinder_cent_stress(x);%mean stress part

S=sigma_m/S_y1+sigma_a/Se-1/n;%=<0

```

```

end

%%%%%%%%%%%%%%%
%nonlin_cons.m

function [c,ceq] = nonlin_cons(x)
%d=x(1:29);
%clearance1=x(30);
%clearance2=x(31);

[~,~,~,~,q]=assembly(x);
q0=0.002;%toleransi 1 mm alalim

qx=[];qy=[];
jj=1;
for ii=[2:16 24:29]
    qx(jj)=q(4*ii-3);
    qy(jj)=q(4*ii-2);
    jj+=1;
end
q1=sqrt(qx.^2+qy.^2);
% Compute nonlinear inequalities at x. c(x) <= 0

```

```

for ii=1:21

c(ii) =q1(ii)-q0;

end


%stress and fatigue control

S_y1=1590*10^6;%N/m^2 AISI 4340 Q&T 315C

S_y2=850*10^6;%Inconel 718 for all disks @700C celcius

Se1=700*10^6;%Se' endurance limit of AISI4340 Q&T @315C

Sut=1720;%ultimate strength of AISI4340 Q&T @315C

ka=0.75;

kb=0.633;

Se=ka*kb*Se1;%endurance limit with modifying factors


%disk stress control

[s_disk_i,s_disk_o]=disk_cent_stress(x);

for ii=1:42

if ii<22

c(21+ii)=s_disk_i(ii)-S_y2;

else

c(21+ii)=s_disk_o(ii-21)-S_y2;

end

end

```

```

%beam stress & fatigue control

S=soderberg_beam(x);

for ii=1:29

c(63+ii)=S(ii);

end

ceq = [] ; ... % Compute nonlinear equalities at x. ceq(x)
= 0

end

%%%%%%%%%%%%%%%
%obj_func.m

function y = obj_func(x)

d=x(1:29);

clearence1=x(30);

clearence2=x(31);

xx=[d ;clearence1; clearence2];

[~,~,~,K,q]=assembly(xx);

```

```
%potential energy  
y =q'*K*q;  
  
end  
  
%%%%%%%%%%%%%  
%gen_alg_usage_guide.txt
```

gen_alg Function User's Manual

Function Definition

```
[best_x, best_fval,exitflag,output,history] =  
gen_alg(objfun,nvars,A,b,Aeq,beq,lb,ub,nonlcon, options)
```

Descriptions

- gen_alg performs limited optimization with Genetic Algorithm (GA).
- Supports both linear and nonlinear equality/inequality constraints.
- The search area can be restricted by specifying the lower and upper limits.Dec.
- It gets a user-defined purpose function and a constraint function.
- parameters can be set with the options structure.

Input Parameters

Parameter Description Type/Format

objfun is the handle of the objective (fitness) function. A single vector receives input. @function_name or anonymous function

the number of nvars Decision variables is a positive integer

A Linear inequality matrix ($A^*x \leq b$) Matrix ($m \times n$) or empty []

b Linear inequality boundary vector Vector ($m \times 1$) or empty []

Aeq Linear equality matrix ($A_{eq}^*x = beq$) Matrix ($p \times n$) or empty []

beq Linear equality boundary vector Vector ($p \times 1$) or empty []

the lower bounds of lb Decision variables are Vector ($n \times 1$)

the upper bounds of ub Decision variables are Vector ($n \times 1$)

handle the nonlcon Nonlinear constraint function. in the form c, ceq = nonlcon(x). @function_name or anonymous function, if empty [] there is none

options Struct struct containing the operating parameters (optional)

options Fields and Default Values

Field Description Default Value

PopulationSize Population size 50

```
ations The maximum number of generations is 100
Crossover Crossover ratio (0-1) 0.8
Mutation Rate Mutation rate (0-1) 0.05
Number of EliteCount Elite individuals 2
Display Console output: 'off' or 'pushes' 'off'
SelectionMethod Selection algorithm: 'tournament',
'roulette', 'rank', 'random' 'tournament'
```

Function Outputs

Output Description

the best solution found in `best_x` (decision variables)

`best_fval` is the value of the objective function of the best solution

`exitflag` Solution status (1 = successful termination)

`output` Working information struct

- ◆ Population and fitness data for each generation

Example of Use

```
% Objective function (example)
```

```
objfun= @(x)(x(1)-3)^2+(x(2)+1)^2;
```

```
% Nonlinear constraint function (example)
```

```
nonlcon= @(x) deal([x(1)^2+x(2)^2 - 4], []);
```

```

% Linear constraints

A= [1, 2];
b= [3];
Aeq= [];
beq= [];

% Limits

lb = [-10; -10];
ub = [10; 10];

% Settings

options = struct(...

    'PopulationSize', 100, ...
    'Generations', 150, ...
    'Crossover', 0.7, ...
    'mutationRate', 0.1, ...
    'EliteCount', 3, ...
    'Display', 'pushes', ...
    'SelectionMethod', 'roulette' ...
);

% Optimization

```

```
[best_x, best_fval, exitflag, output, history] =  
ga_gpt(objfun, 2, A, b, Aeq, beq, lb, ub, nonlcon, options);
```

```
fprintf('Best solution: [%f, %f]  
, best_x(1), best_x(2));  
fprintf('Value of the objective function: %f  
, best_fval);
```

Notes

- the nonlcon function must be in the format c, ceq = nonlcon(x);
 - c: inequality constraints (≤ 0)
 - ceq : equality constraints ($=0$)
- the history structure contains as many elements as the number of generations, each element contains population and fitness information.
- Display = 'pushes' option writes the best fitness value in each generation to the console.
- The SelectionMethod option allows you to select one of four different selection algorithms.

```
%%%%%%%%%%%%%%
```

%picrotor.m

```
function [] = picrotor(model)  
%
```

```

% function to plot a schematic of the rotor system
%
% [] = picrotor(model)
%
% This function takes the definitions for the nodes,
% shafts, bearings
%
% and discs and plots the rotor cross-section on a 2D plot
%
%
% This function is part of a MATLAB Toolbox to accompany
% the book
%
% 'Dynamics of Rotating Machinery' by MI Friswell, JET
% Penny, SD Garvey
%
% & AW Lees, published by Cambridge University Press, 2010
%
Node_Def = model.node;
Shaft_Def = model.shaft;
Disc_Def = model.disc;
Bearing_Def = model.bearing;

FontName = 'Arial';
FontSize = 10;

plot_nodes = 1; % determines whether to plot marker and
numbers for nodes

```

```

% Set up plotting functions

% The first column is the z position and the 2nd column the
radius

[nnode,ncol_node] = size(Node_Def);
ndof = 4*nnode;
if ncol_node == 1
    Node_Def = [(1:nnode)' zeros(nnode,2) Node_Def];
end
if ncol_node == 2
    Node_Def = [Node_Def(:,1) zeros(nnode,2)
Node_Def(:,2)];
end

[nshaft,ncol_shaft] = size(Shaft_Def);
[ndisc,ncol_disc] = size(Disc_Def);
[nbearing,ncol_bearing] = size(Bearing_Def);

shaft_length = max(Node_Def(:,4)) - min(Node_Def(:,4));

hold on

% Plot shaft sections

max_shaft_radius = 0.0;
shaft_radii_all = zeros(nshaft,2);

```

```

for i = 1:nshaft

    n1 = Shaft_Def(i,2);
    n2 = Shaft_Def(i,3);

    z1 = Node_Def(n1,4);
    z2 = Node_Def(n2,4);

    % symmetric shaft
    if Shaft_Def(i,1) < 10
        shaft_or1 = 0.5*Shaft_Def(i,4);
        shaft_ir1 = 0.5*Shaft_Def(i,5);
        shaft_or2 = shaft_or1;
        shaft_ir2 = shaft_ir1;
    end

    % asymmetric shaft - estimate radii from stiffness
    values
    if Shaft_Def(i,1) > 10 && Shaft_Def(i,1) < 20
        shaft_ir1 = 0.0; % as good as any!
        shaft_ir2 = 0.0;
        E = 211e9; % assume steel
        EI = 0.5*(Shaft_Def(i,4)+Shaft_Def(i,5));
        shaft_or1 = sqrt(sqrt(4*EI/(E*pi)));
        shaft_or2 = shaft_or1;
    end

    % tapered shaft
    if Shaft_Def(i,1) > 20

```

```

    shaft_or1 = 0.5*Shaft_Def(i,4);
    shaft_or2 = 0.5*Shaft_Def(i,5);
    shaft_ir1 = 0.5*Shaft_Def(i,6);;
    shaft_ir2 = 0.5*Shaft_Def(i,7);;

end

max_shaft_radius = max([max_shaft_radius shaft_or1
shaft_or2]);

shaft_radii_all(i,:) = [shaft_or1 shaft_or2];

zfill = [z1 z1 z2 z2];

if shaft_ir1 == 0 && shaft_ir2 == 0
    xfill = [-shaft_or1 shaft_or1 shaft_or2
-shaft_or2];
    fill(zfill,xfill,'c')
else
    xfill = [shaft_ir1 shaft_or1 shaft_or2 shaft_ir2];
    fill(zfill,xfill,'c')
    xfill = [-shaft_or1 -shaft_ir1 -shaft_ir2
-shaft_or2];
    fill(zfill,xfill,'c')
    plot([z1 z1],[shaft_or1 -shaft_or1],'k',[z2 z2],
[shaft_or2 -shaft_or2],'k')
end

end

% Plot discs

```

```

max_disc_radius = 0;
max_disc_thick = 0.5*max_shaft_radius;
for i = 1:ndisc
    Disk_Type = round(Disc_Def(i,1));
    disk_node = Disc_Def(i,2);
    if Disk_Type == 1 || Disk_Type == 3
        thickness = Disc_Def(i,4);
        disk_radius = 0.5*Disc_Def(i,5);
        dsk_iradius = 0.5*Disc_Def(i,6);
    end
    if Disk_Type == 2 || Disk_Type == 4
        % approximate radius from Id (neglecting thickness)
        % and mass
        disk_radius = sqrt(4*Disc_Def(i,4)/Disc_Def(i,3));
        % calculate convenient thickness as it is difficult
        % to estimate
        thickness = 0.2*disk_radius;
        dsk_iradius = [];
        for ii = 1:nshaft % find shaft elements at node
            if Shaft_Def(ii,2)== disk_node
                dsk_iradius = [dsk_iradius
shaft_radii_all(ii,1)];
            end
            if Shaft_Def(ii,3)== disk_node
                dsk_iradius = [dsk_iradius
shaft_radii_all(ii,2)];
            end
        end
    end
end

```

```

dsk_iradius = mean(dsk_iradius);
end

max_disc_thick = max([max_disc_thick thickness]);
max_disc_radius = max([max_disc_radius disk_radius]);
% disk_node

zc = Node_Def(disk_node,4);
z1 = zc - thickness/2;
z2 = zc + thickness/2;
alpha=0.2;
knee_radius = alpha*disk_radius+(1-alpha)*dsk_iradius;
xfill = [dsk_iradius knee_radius disk_radius
disk_radius knee_radius dsk_iradius];
zfill = [zc z1 z1 z2 z2 zc];
fill(zfill,xfill,'y')
xfill = -[dsk_iradius knee_radius disk_radius
disk_radius knee_radius dsk_iradius];
zfill = [zc z1 z1 z2 z2 zc];
fill(zfill,xfill,'y')

end

% Plot bearings - not to scale

max_shaft_radius_bearing = 0;
for i = 1:nbearing
    bearing_type = Bearing_Def(i,1);
    bearing_node = Bearing_Def(i,2);
    if bearing_node==1
        shaft_radius = shaft_radii_all(1,1);

```

```

elseif bearing_node==nnode
    shaft_radius = shaft_radii_all(nnnode-1,2);
else
    shaft_radius =
0.5*(shaft_radii_all(bearing_node-1,2)+shaft_radii_all(bear
ing_node,1));
end

max_shaft_radius_bearing =
max([max_shaft_radius_bearing shaft_radius]);

bear_radius = 2*shaft_radius;
bear_width = 0.5*shaft_radius;
zc = Node_Def(bearing_node,4);
z1 = zc - bear_width;
z2 = zc + bear_width;
xfill = [shaft_radius bear_radius bear_radius];
zfill = [zc z1 z2];
fill(zfill,xfill,'r',zfill,-xfill,'r')
text(z1, -1.6*bear_radius,[' Brg Type ',
num2str(bearing_type)],...
'HorizontalAlignment','center','FontName',FontName, ...
'FontSize',FontSize);

end

% Plot markers for nodes

if plot_nodes==1
%     marker_size = 0.35*max_shaft_radius;
marker_size = 0.35*max_shaft_radius_bearing;

```

```

for i = 1:nnode
    zc = Node_Def(i,4);
    z1 = zc - marker_size;
    z2 = zc + marker_size;
    xfill = [0 marker_size 0 -marker_size];
    zfill = [z1 zc z2 zc];
    fill(zfill,xfill,'g')
end

% add node numbers
for i = 1:nnode
    text(Node_Def(i,4),-1.2*max([max_disc_radius
max_shaft_radius]),...
        ['Node ' num2str(Node_Def(i,1))],...
        'HorizontalAlignment','right','FontName',FontName,....
        'FontSize',FontSize,'Rotation',90)
end

end

% Plot dot-dashed centreline
plot([-0.05*shaft_length 1.05*shaft_length], [0 0],'k-.')

axis('off')
hold off
set(gca,'position',[0.05 0.5 0.9 0.4]);

```

```

drawnow

%%%%%%%%%%%%%%%
%picrorotor_set.m

clear

format short e

close all


set(0,'defaultaxesfontsize',12)
set(0,'defaultaxesfontname','Times New Roman')
set(0,'defaulttextfontsize',12)
set(0,'defaulttextfontname','Times New Roman')

% set the material parameters

E = 211e9;
G = 81.2e9;
rho = 7810;
rho_disk=8190;
damping_factor = 0;    % no damping in shaft

l1=0.025;l2=0.1;l3=0.175;
l4=0.2;l5=0.15;l6=0.05;

```

```

l=[0 l1 l2*ones(1,14) l3 l4*ones(1,6) l5 l4*ones(1,5) l6];

for ii=1:30

model.node(ii,:) = [ii sum(l(1:ii))];

end


shaft_od = [ 1.0583e-01;2.6969e-01; 2.7304e-01;
2.5530e-01; 2.4132e-01;...
2.3596e-01; 2.3024e-01; 2.2333e-01; 2.2194e-01;
2.2182e-01;...
2.2246e-01; 2.2557e-01; 2.2302e-01; 2.1146e-01;
1.9402e-01;...
1.6670e-01; 1.0237e-01; 1.1749e-01; 2.8113e-01;
2.9279e-01;...
2.7511e-01; 1.9064e-01; 1.3162e-01; 1.5384e-01;
1.3953e-01;...
1.2137e-01; 1.2169e-01; 1.0011e-01; 1.2650e-01];

shaft_id = 0.0;

model.shaft=zeros(29,9);

for ii=1:29

model.shaft(ii,:) = [2 ii ii+1 shaft_od(ii) shaft_id rho E
G damping_factor];

end


d_o=[0.60*ones(1,4) 0.50*ones(1,11) 0.40 0.40 1*ones(1,4)];
%disk outer diameters

```

```

disk_thick=[0.05*ones(1,15) 0.1*ones(1,6)];%disk
thicknesses;

disk_node=[2:16 24:29];

shaft_od_disk=shaft_od([1:15 24:29]);

for ii=1:21

model.disc(ii,:) = [1 disk_node(ii) rho_disk disk_thick(ii)
d_o(ii) shaft_od_disk(ii)];

end

omg=100*pi;ep=0.6;

eta=0.4; %Pa.s SAE40 ya   25 derecede

f=@(d,c)(pi^10^-3*d^4*omg*eta)/(8*c^2)*(ep/(1-
ep^2)^2)*[(16/pi^2-1)*ep^2+1]^(1/2);

d1=shaft_od(1);d2=shaft_od(29);

c1=1.6655e-03;c2=6.2884e-05;

model.bearing = [7 1 f(d1,c1) shaft_od(1) shaft_od(1)/10
c1 eta; ...

7 30 f(d2,c2) shaft_od(29) shaft_od(29)/10
c2 eta];

% draw the rotor

figure(1), clf

picotor(model)

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%output.txt

```
options = struct(...  
    'PopulationSize',     80, ...  
    'Generations',        1000, ...  
    'CrossoverRate',      0.7, ...  
    'MutationRate',       0.05, ...  
    'EliteCount',         4, ...  
    'Display',            'iter', ...  
    'SelectionMethod',    'tournament' ...  
);
```

```
lb= [0.1*ones(29,1); 0 ;0];  
ub=[0.4*ones(29,1); 0.002 ;0.002 ];  
nvars = 31;
```

```
Generation 1000: Best Fitness = 0.000000
```

```
Optimal solution:
```

```
1.3686e-01  
3.3257e-01
```

3.2867e-01
3.1246e-01
2.9235e-01
2.7047e-01
2.7686e-01
2.8007e-01
2.7845e-01
2.7118e-01
2.7545e-01
2.7705e-01
2.6189e-01
2.6592e-01
2.6014e-01
2.6494e-01
1.8109e-01
2.4378e-01
2.8109e-01
2.9472e-01
3.3756e-01
1.3545e-01
3.1414e-01
3.8213e-01
3.7543e-01

3.6544e-01

3.6934e-01

3.8129e-01

2.9621e-01

1.4008e-03

1.4006e-05

Minimum value of Objective Function:

2.7929e-08

Nonlinear Constraints:

-2.0000e-03

-2.0000e-03
-2.0000e-03
-2.0000e-03
-2.0000e-03
-2.0000e-03
-2.0000e-03
-2.0000e-03
-8.0567e+08
-8.0275e+08
-8.0284e+08
-8.0322e+08
-8.1687e+08
-8.1732e+08
-8.1719e+08
-8.1713e+08
-8.1716e+08
-8.1731e+08
-8.1722e+08
-8.1719e+08
-8.1748e+08
-8.1741e+08
-8.1752e+08
-8.2514e+08

-8.2535e+08
-7.2471e+08
-7.2462e+08
-7.2434e+08
-7.2604e+08
-8.3415e+08
-8.1883e+08
-8.1926e+08
-8.2099e+08
-8.2691e+08
-8.2896e+08
-8.2838e+08
-8.2808e+08
-8.2823e+08
-8.2890e+08
-8.2851e+08
-8.2836e+08
-8.2972e+08
-8.2937e+08
-8.2988e+08
-8.2000e+08
-8.2084e+08
-7.9237e+08

- 7 . 9189e+08
- 7 . 9040e+08
-8 . 0001e+08
-9 . 9728e-01
-9 . 9728e-01

-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9728e-01
-9.9726e-01

Are constraints provided

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1