Development of Optimization Design Software for Bevel Gear Based on Integer Serial Number Encoding Genetic Algorithm

Xiaoqin Zhang
Mechanical and Electrical Engineering College
Hebei Normal University of Science & Technology, Qinhuangdao, China
Email: zxqwlc@163.com

Yu Rong, Jingjing Yu, Liling Zhang and Lina Cui
Mechanical and Electrical Engineering College
Hebei Normal University of Science & Technology, Qinhuangdao, China
Email: lixiangcg@126.com

***Abstract*—Bevel gear drive is widely used, quality of which not only affects its own transmission performance, size and weight, but also has some impact on the machine's performance. This paper introduces optimization design software for bevel gear, in which automatic optimization design is realized. In the paper mathematical model, programming of design data and realization of optimization design based on genetic algorithm are described in detail. The paper proposed integer serial number encoding genetic algorithm, which effectively deals with continuous and discrete variable optimization problem and reduces the code length of the string to improve the encoding and decoding efficiency, no invalid solution or duplicate solutions.**

***Index Terms*—bevel gear, optimization design, genetic algorithm, integer serial number encoding, augmented penalty function**

I. Introduction

Bevel gear drive, characterized by changing direction, high coincidence and smooth transmission, etc., is widely used in the aerospace, automotive and large mechanical transmission system. So its design quality not only affect its own transmission performance, size and weight but also have some impact on the machine's performance. In practical engineering design, involving many parameters, consuming much calculation time, prone to error, and repeated calculation, query and drawing are needed for series of product design, resulting in substantial duplication of effort, so the development of bevel gear design software finding the optimal design is of great significance [1] [2] [3].

Optimization design of bear gear includes the continuous variables and discrete variables. The traditional method is to round the optimal design to the adjacent discrete points. Thus, design point might run out of the feasible region, besides traditional optimization method mostly bases on gradient algorithm, which is likely trapped into local minimum search. Many studies indicate that the genetic algorithm has strong ability of general optimization, which is very effective in treating optimization problem containing continuous and discrete variables[4][5].

This paper proposes the optimization design software for straight tooth bevel gear based on genetic algorithm encoding with integer serial number, no invalid solution or duplicate solutions, the code length of the string reduced, the encoding and decoding efficiency improved.

II. Software Function

The software can accomplish strength-calculation and optimal design of straight tooth bevel gear based on GB/T *Calculation Methods of Load Capacity for Bevel Gear*. Moreover friendly user interface is developed using VB language shown in Fig. 1. Input the known data, click on the button "optimization calculation", the user can get the optimization results of straight tooth bevel gear. Click on the button "gear materials and fatigue limit query", the user can get the corresponding fatigue limit of gear materials, shown in Fig. 2. Click on the button “coefficient calculation”, various coefficients can be calculated, shown in Fig. 3. For example, select Yfsa and click on the button “calculate coefficient”, then the interface for calculating Yfsa appear as shown in Fig. 4. Input the known data and click on the calculating button, the results of recombination tooth coefficient for pinion and gear are gained.

III. Critical Technology

A. Establishing Mathematical Model

Establishing mathematical model is prime step for gear design, in which the design variables, objective function

Manuscript received January 1, 2010; revised December 10, 2010; accepted January 8, 2011
Corresponding author: Zhang Xiaoqin

doi:10.4304/jsw.6.5.915-922

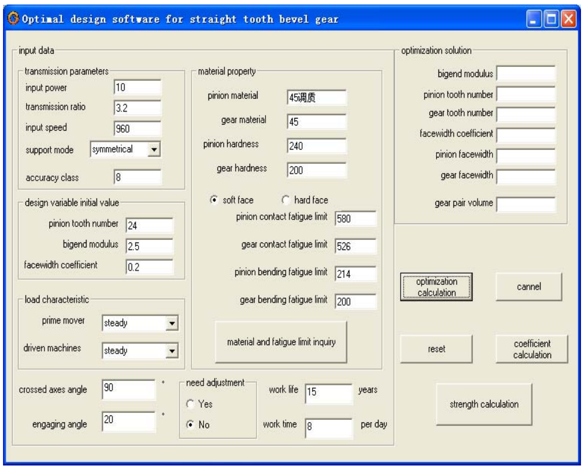
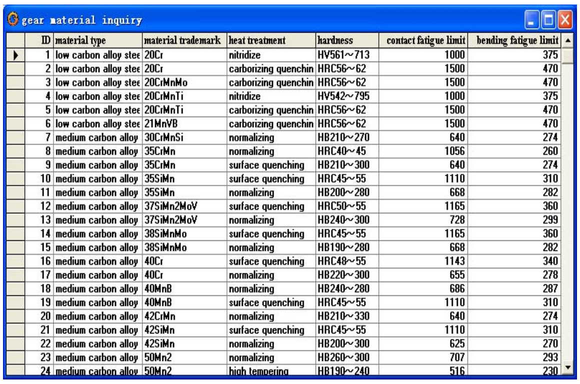


Figure 1. User interface



ID	material type	material trademark	heat treatment	hardness	contact fatigue limit	bending fatigue limit
1	low carbon alloy stee	20Cr	nitridize	HV561~713	1000	375
2	low carbon alloy stee	20Cr	carborizing quenchin	HRC56~62	1500	470
3	low carbon alloy stee	20CrMnMo	carborizing quenchin	HRC56~62	1500	470
4	low carbon alloy stee	20CrMnTi	nitridize	HV542~795	1000	375
5	low carbon alloy stee	20CrMnTi	carborizing quenchin	HRC56~62	1500	470
6	low carbon alloy stee	21MnVB	carborizing quenchin	HRC56~62	1500	470
7	medium carbon alloy	30CrMnSi	normalizing	HB210~270	640	274
8	medium carbon alloy	35CrMn	normalizing	HRC40~45	1056	260
9	medium carbon alloy	35CrMn	surface quenching	HB210~300	640	274
10	medium carbon alloy	35SiMn	surface quenching	HRC45~55	1110	310
11	medium carbon alloy	35SiMn	normalizing	HB200~280	668	282
12	medium carbon alloy	37SiMn2MoV	surface quenching	HRC50~55	1165	360
13	medium carbon alloy	37SiMn2MoV	normalizing	HB240~300	728	299
14	medium carbon alloy	38SiMnMo	surface quenching	HRC45~55	1165	360
15	medium carbon alloy	38SiMnMo	normalizing	HB190~280	668	282
16	medium carbon alloy	40Cr	surface quenching	HRC48~55	1143	340
17	medium carbon alloy	40Cr	normalizing	HB220~300	655	278
18	medium carbon alloy	40MnB	normalizing	HB240~280	686	287
19	medium carbon alloy	40MnB	surface quenching	HRC45~55	1110	310
20	medium carbon alloy	42CrMn	normalizing	HB210~330	640	274
21	medium carbon alloy	42SiMn	surface quenching	HRC45~55	1110	310
22	medium carbon alloy	42SiMn	normalizing	HB200~300	625	270
23	medium carbon alloy	50Mn2	normalizing	HB260~300	707	293
24	medium carbon alloy	50Mn2	high tempering	HB190~240	516	230

Figure 2. Gear material

and constraints are determined through studying gear design theory.

B. Programming of A Large Number of Graph and Table

In gear design process, involving a large number of graph and table data, how to invert artificial seeking into automatically calculation is a basic problem in the process of CAD operations.

C. Genetic Algorithm Realizing

The augmented penalty function and integer serial number encoding is used in genetic algorithm.

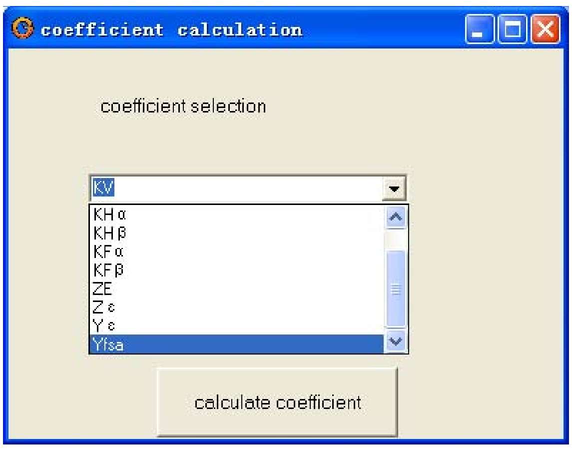


Figure 3. Coefficient calculation

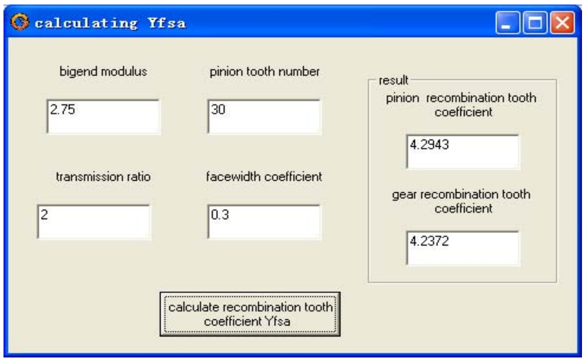


Figure 4. Yfsa calculation

Software design flow chart is shown as Fig. 5

IV. MATHEMATICAL MODEL FOR OPTIMIZATION DESIGN

Optimization objects for straight tooth bevel gear are various. If user requirements needed, the consumption of material is the measurement of design. Therefore the optimization object is the volume of frustum of cone of the bevel gear pair [2][6][7].

A. Object Function

Bevel Gear volume are the sum of pinion volume and gear volume, while each bevel gear volume is similar to the volume of frustum of cone between the big end and small end of the pitch circle. Therefore, according to the volume formula of frustum of cone, volume calculation formula of straight tooth bevel gear pair can be expressed as:

$$
\begin{aligned}
V &= V1+V2 \\
&= \frac{\pi}{3} b\cos\delta_1\left[(\frac{mz_1}{2})^2 + \frac{mz_1}{2}(\frac{R-b}{R}\times\frac{mz_1}{2}) + (\frac{R-b}{R}\times\frac{mz_1}{2})^2\right] + \\
&\quad \frac{\pi}{3} b\cos\delta_2\left[(\frac{mz_2}{2})^2 + \frac{mz_2}{2}(\frac{R-b}{R}\times\frac{mz_2}{2}) + (\frac{R-b}{R}\times\frac{mz_2}{2})^2\right]
\end{aligned}
\tag{1}
$$

Where, b is the face width, $\delta 1$, $\delta 2$ are respectively cone angle of pinion and gear, m is modulus, $z1$ is tooth number of pinion, R is cone pitch, $z2$ is tooth number of gear.

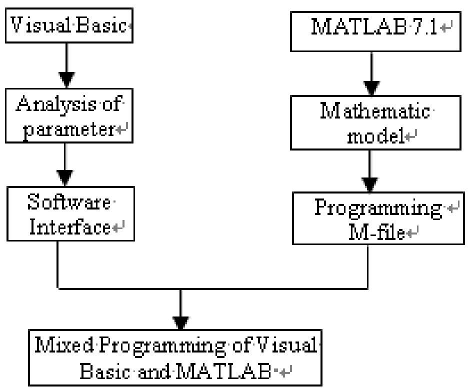


Figure 5. Software design flow chart

B. Design Variable

The independent design parameters of volume of straight bevel gear drive include big end module *m*, tooth number of pinion *z*1, face width coefficient φ_R, so the design variables are.

$X = [m, z_1, \varphi_R]^T = [x_1, x_2, x_3]^T$ (2)

C. Constraint Conditions

1) Contact strength conditions

The bending stress σ_H and contact stress σ_F of gears should be not more than the allowable value, i.e.

$$\sigma_H = Z_E Z_H Z_\varepsilon \sqrt{\frac{4.7KT_1}{\varphi_R(1-0.5\varphi_R)^2 m^3 z_1^3 u}} \le [\sigma_H] \quad (3)$$

Where, ZE is the elastic coefficient; ZH is regional coefficient of pitch point, for straight gear ZH = 2.5; Zε is the coincidence coefficient; K is load coefficient; T1 is input torque; [σH] is the allowable contact stress.

2) Tooth root bending strength conditions

$$\sigma_{F1} = \frac{4.7KT_1 Y_{FSa1} Y_\varepsilon}{\varphi_R(1-0.5\varphi_R)^2 z_1^{\,2} m^3 \sqrt{u^2+1}} \le [\sigma_{F1}] \quad (4)$$

$$\sigma_{F2} = \sigma_{F1}\frac{Y_{FSa2}}{Y_{FSa1}} \le [\sigma_{F1}] \quad (5)$$

σ_{F1},σ_{F2} are respectively tooth root bending stress of small and large bevel gear; $[\sigma]_{F1}$, $[\sigma]_{F2}$ are respectively allowable bending stress of small and large bevel gear; Y_{FSa1}, Y_{FSa2} are respectively tooth recombination coefficient of small and large bevel gear; Yε is the contact ratio coefficient.

3) The maximum peripheral speed conditions.

For straight bevel gear, the peripheral speed of the average diameter *v*m should meet:

$v_m < v_{max}$ (6)

Where, *v*max is the maximum peripheral speed.

4) Modulus constraints

For power transmission gear, the minimum modulus is not less than 1.5, i.e.

$m_{max} \geqslant m \geqslant 1.5$ (7)

Where, m_{max} is the maximum value of *m*.

5) Face width coefficient constraints.

Usually the range of face width coefficient φ_R is 0.25~0.3

6) The condition of not root cut for small bevel gear

$z_{1max} \geqslant z_1 \geqslant 17\cos\delta_1$ (8)

V. PROGRAMMING OF DESIGN DATA

In gear design manuals, the relationships between many coefficients and other parameters are expressed by table or line graph. In the traditional design, we can determine the values of relevant coefficients by consulting manual, while in optimization design software, all designs and calculations are fulfilled through the computer automatically. Therefore programming of chart data is needed.

A. Table Programming

Tables used in the mechanical design are divided into simple lists and list functions according to having function relations or not between data. In simple lists various data are independent, having no clear relationship, which can be stored in one-dimensional array, two-dimensional array or three-dimensional array, retrieved by using look-up, interpolation method, and so on. Function relation is existed between function data and variables, but cannot be expressed by clear expression in list function, which is usually treated by interpolation or curve fitting methods.

Table I shows using coefficient KA [6]. It is a simple table. When programming, the data are restored into two-dimensional array. MATLAB program is as follows:

TABLE I.
USING COEFFICIENT KA

Prime mover conditions	**Work Machine Conditions**			
	Steady	***Slight impact***	***Moderate impact***	***Severe impact***
Steady	1.00	1.25	1.50	1.75
Slight impact	1.10	1.35	1.60	1.85
Moderate impact	1.25	1.50	1.75	2.0
Severe impact	1.5	1.75	2.0	2.25

```
function kA=gearKA(i,j)
%i is prime mover condition; j is work machine condition
k=[1.00 1.25 1.50 1.75;1.10 1.35 1.60 1.85;1.25 1.50 1.75 2.0;1.5 1.75 2.0 2.25];
kA=k(i,j);
```

When KA is used, function gearKA() is loaded.

B. Graph Programming

According to sources, line graph can be divided into two kinds: one kind is represented by the parameters of the graph between which have calculation formula, just because the formula is so complex that drawn into graph to manually search; one is that the parameters of the graph can't be found calculation formula. For the first type, it's needed to find the original formula, and incorporated into the program; For the second, it's needed to discrete the graph into table, and treat it as table or seek fitting formulae by curve fitting method and incorporated into the program.

Fig. 6 shows the dynamic load coefficient *KV* [6] [7]. In the paper, each curve is respectively programmed by curve fitting. The programs are respectively named as KV5, KV6, KV7, KV8, KV9, KV10, KV11 and KV12.

The program of KV8 is as follows:

```
function kv=KV8(x)
v=[2 5 8 10 12 15 18 20 25 30];
y=[1.128 1.196 1.244 1.269 1.292 1.323  1.35 1.367 1.404  1.437];
p=polyfit(x,y,2)
```

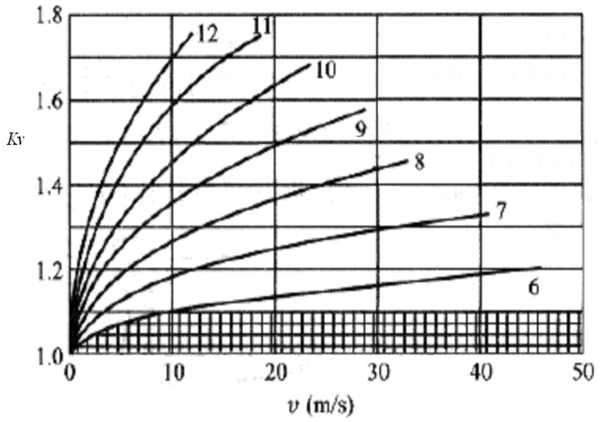


Figure 6. Dynamic load coefficient

```
kv=polyval(p,x)
```

C. Programming of Gear Recombination Coefficients

In gear design, the tooth shape coefficient Y_{Fa} and stress correction coefficient Y_{Sa} are the most complicated to determine, affected by the gear module, helix angle, equivalent number of teeth, variable coefficient, pressure angle, tooth root fillet radius and other factors. Ref. [6] gives the calculating formula in detail, in which a transcendental equation is needed to solve.

1) Solving of transcendental equation.

When calculating Y_{Fa} or Y_{Sa}, it's needed to solve equation:

θ=2.*G.*tan(θ)./zv+H (9)

Where x is needed to calculate, G and H are middle variables, and zv is equivalent tooth number.

This equation can be solved by MATLAB function fsolve( ). For example,

x = fsolve(fun,x0) (10)

Equation (10) starts at x0 and tries to solve the equations described in fun.

The program of solving (9) is as followings:

```
sita0=pi./6.*ones(1,length(zv))  %starting point
sitax=@(x)bevelsita(x,G,H,zv) % function handle
sita=fsolve(sitax,sita0)
```

Where bevelsita is a M-file function, the content of which is as followings:

```
function s=bevelyfasita(x,G,H,zv)
s=x-2.*G.*tan(x)./zv+H;
```

2) Recombination gear tooth shape coefficient

In the paper, the tooth shape coefficient Y_{Fa} and stress correction coefficient Y_{Sa} are integrated into one parameter, that is, Y_{Fa}* Y_{Sa}, which are called recombination gear tooth shape coefficient, and programmed by one M-file function, BevelGearYfsa.m, the content of which is as followings:

```
function [yfs1,yfs2]=bevelgearYfsa(m,z1,u,fr)
%generating method, m: module; z1: tooth number of %pinion; u: rotating speed ratio; fr: tooth face %coefficient
if app==1    % applied to  industry;
d1=m*z1     %big end diameter
…
else app==2  % applied to automobile
d2=m*z1    % big end diameter
…
end
…
sf1=(zv1*sin(pi/3-sita1)+sqrt(3)*(G/cos(sita1)-pa0/mm))*mm
sf2=(zv2*sin(pi/3-sita2)+sqrt(3)*(G/cos(sita2)-pa0/mm))*mm
pf1=(pa0/mm+2*G^2/(cos(sita1)*(zv1*cos(sita1)^2-2*G)))*mm
pf2=(pa0/mm+2*G^2/(cos(sita2)*(zv2*cos(sita2)^2-2*G)))*mm
hfa1=(0.5*zv1*(cos(alf)/cos(alffa1)-cos(pi/3-sita1))+0.5*(pa0/mn-G/cos(sita1)))*mn
hfa2=(0.5*zv2*(cos(alf)/cos(alffa2)-cos(pi/3-sita2))+0.5*(pa0/mn-G/cos(sita2)))*mn
%calculating tooth shape factor of pinion and gear
yfa1=(6*hfa1/mm)/((sf1/mm)^2)*cos(alffa1)/cos(alf)
yfa2=(6*hfa2/mm)/((sf2/mm)^2)*cos(alffa2)/cos(alf)
La1=sf1./hfa1
La2=sf2./hfa2
qs1=sf1./2./pf1
qs2=sf2./2./pf2
%calculating stress correction coefficients of pinion %and gear
ysa1=(1.2+0.13.*La1).*qs1.^(1./(1.21+2.3./La1))
ysa2=(1.2+0.13.*La2).*qs2.^(1./(1.21+2.3./La2))
% recombination gear tooth shape coefficients of %pinion and gear
yfs1=yfa1.*ysa1
yfs2=yfa2.*ysa2
```

VI. Realization of Optimizaition Design

A. Principle of Genetic Algorithm

Genetic Algorithm (GA) is referred to as a search method of optimal solution to simulating Darwin's genetic selection and biological evolution process. Genetic algorithm is a series of random iterations and evolutionary computations simulating the process of selection, crossover and mutation occurred in natural selection and population genetic, in according to the survival of the fittest, through crossover and mutation, good quality gradually maintained and combined, while continually producing better individuals and out of bad individuals. Through the generational produce and optimizing the individual, the whole group evolves forward and constantly approaches to the optimal solution.

Genetic algorithm, not requiring gradient information and continuous function, optimization results being global, applied to mechanical design optimization problems, can effectively avoid local optimal solutions, and get the global optimal solution [2] [7].

B. Outline of the Genetic Algorithm

The following outline summarizes how the genetic algorithm works [8]:

1) The algorithm begins by creating a random initial population.

2) The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:

a) Scores each member of the current population by computing its fitness value.

b) Scales the raw fitness scores to convert them into a more usable range of values.

c) Selects members, called parents, based on their fitness.

d) Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.

e) Produces children from the parents.

Children are produced either by making random changes to a single parent - mutation - or by combining the vector entries of a pair of parents - crossover.

f) Replaces the current population with the children to form the next generation.

3) The algorithm stops when one of the stopping criteria is met.

C. Integer Serial Number Encoding

In the paper, integer serial number encoding method is used, in which each chromosome represents the number of a variable value in discrete collection. The advantages of integer serial number encoding relative to the binary are smaller seek space and simple, intuitive operation. For example, a design variable has 10 optional discrete values, 4-bit binary code is needed (16 kinds of solutions can be expressed), then will result in 6 invalid or duplicate combination solutions. However, integer serial number encoding can fully express combination solutions with only numbers from 0 to 9, so the genetic algorithm does not produce an invalid solution or repeated solution, but also reduces the code string length and improves the encoding and decoding efficiency [9].

In the standard straight bevel gear design, the independent design variables are three: m, $z1$, fR, where m and $z1$ are discrete, and fR is continuous. If fR is kept two fractions, it is changed into discrete variable. Consistent with modulus constrains, if m_{max}=50, the standard modulus number is 36, m = [1.5 1.75 2 2.25 2.5 2.75 … 45 50], consistent with tooth number constrains, the number of z1 is 28, z = [13 14 15 16 17 … 38 39 40], consistent with tooth width coefficient constraints, the number of fR is 6, fR = [0.25 0.26 0.27 0.28 0.29 0.3]. So code ranges of $x(1)$, $x(2)$ and $x(3)$ are respectively 1-36, 1-28 and 1-6 [6][7].

Decoding ways of the three parameters are separately as follows:

Modulus m:

```
switch x(1)
    case 1
        m=1.5
    case 2
        m=1.75
    …
    case 35
        m=45
    case 36
        m=50
    otherwise
        disp ('modulus is not in the range')
end
```

Tooth number z:

z=12+ x(2)

Face width coefficient fR:

fR=0.24+x(3)/100

D. Initial Population

The initial population is produced by following program:

```
A=rand(PopulationSize,3)
X1=round(1+(19-1).*A(1:PopulationSize,1))
X2=round(1+(24-1).*A(1:PopulationSize,2))
X3=round(1+(13-1).*A(1:PopulationSize,3))
InitialPopulation =[X1 X2 X3]
```

Where, rand() is random function, round() is rounding function and PopulationSize(Population Size) is the number of individuals.

Population size is an important parameter in genetic algorithm, which specifies how many individuals there are in each generation. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing the chance that the algorithm will return a local minimum that is not a global minimum. However, a large population size also causes the algorithm to run more slowly. So it can not be too large or too small. The range of population size is 10-160. It is determined to be 50 by trying.

The program is as follows:

```
options = gaoptimset(' Populationsize ',50)
```

E. Fitness Function

Gear optimization problem is nonlinear constrained optimization problems, commonly used penalty function method. Penalty function method is simple in principle, easy in algorithm, wide application and widely used. However, many problems exist in penalty, for example, only when the penalty factor r → ∞ (outside point method) or r → 0 (interior point method), converge the algorithm is, the iterative process is of slow convergence. In addition, when the initial value of the penalty factor r0 is inappropriate, the penalty function may become sick, making optimization difficult [5].

In this paper, augmented penalty function combined Lagrange multiplier method with penalty function method for solving straight bevel gear optimization problems. The form of augmented penalty function is as follows:

$$M(x,\lambda,r) = f(x) + \frac{1}{2r}\sum_{j=1}^{m}\left\{\max[0,\lambda_{1j} + rg_j(x)]^2 - \lambda_{1j}^2\right\} + \sum_{p=1}^{l}\lambda_{2p}h_p(x) + \frac{r}{2}\sum[h_p(x)]^2$$

Where, f(x) is the objective function; g_j(x) is the inequality constraints, m is the number of inequality constraints; h_p(x) is the equality constraints; l is the number of equality constraints; r is penalty factor; λ_{1j} the multiplier vector for inequality constraint functions; λ_{2p} multipliers for the equality constraints vector [10].

F. Solving through MATLAB GA Toolbox

MATLAB is advanced mathematics software launched by the MathWorks Company since the mid 1980s, which faces to science and engineering.

MATLAB genetic toolbox is customized toolbox for genetic algorithm, with which various problems to optimize using genetic algorithm can be easily figured out.

Below is the detailed process to realizing optimization design of straight bevel gear using MATLAB 7.1 genetic toolbox [8].

1) Convert the optimization mathematical model of IV into the following forms applied to MATLAB:

a) Design variable:

$X=[x_1,x_2,x_3]^{\mathrm{T}}$

b) Object function:

$F(X)=\pi * u * (u+1) * x_3 (x_1 * x_2)^3 * (1-x_3+x_3^2/3)/8 \rightarrow \min$

c) Constraint conditions:

Linear inequality constraints:
A*X<=b,
Linear equality constraints:
Aeq*X=beq
Bound constraints:
LB<=X<=UB
Nonlinear inequality constraints:
C(X)<=0
Non linear equality constraints:
Ceq(X)=0

2) Establish .m file for objective function:

```
function  obj=bevelobj(x)
 %m: x1; z1:x2; fR:x3
global u %global variable speed ratio
obj=pi*u*(u+1)*x(3)*(x(1)*x(2))^3*(1-x(3)+x(3)^2/3)
/8
```

3) Establish .m file for nonlinear constraints:

```
function [c,ceq]= bevelnonlcon (x)
%define global variables
global u    % global variable speed ratio
global  p   %power
global n1   %rotate speed
global Zk   %bevel gear factor
global kA %use coefficient
global pregrd  %precision grade
global supp %support type
global adjust %if adjust when assemble
global gearface %soft gear face or hard gear face
global T1 %torpue
…
%integer serial number encoding

% calculating load coefficient for contact strength KH
…
kV=bevelgearKV(pregrd,vet)
 [Zls,Zmb,Yzi]=bevelgearZmb(u,x(1),x(2),x(3))
kHb=bevelgearKHbeta(supp)
kHalf=bevelgearKHalfa(kA,x(1),x(2),pregrd,gearface,
Zls,T1,u,x(3))
KH=kA.*kV.*kHb.*kHalf
% calculating contact stress and allowable stress
…
%contact stress
sigmH=ZH.*sqrt(Fmt*KH*(uv+1)/(dv1*b*uv))
 % allowable contact stress
sigmHmin=SigmaHlim.*ZN./SHmin
% calculating load coefficient for bending strength KF
kFb=bevelgearKFbeta(kHb)
kFalf=bevelgearKFalfa(x(1),x(2),u,pregrd,gearface,T1,
x(3),kA)
KF=kA.*kV.*kFb.*kFalf
Yk=1
Yls=Zls^2
% calculating recombination gear tooth coefficient
[Yfs1,Yfs2]=bevelgearYfsa(x(1),x(2),u,x(3))

%bending stress and allowable bending stress
Rm=R-0.5*b
mm=x(1)*Rm/R
sigmF1=Fmt*KF.*Yfs1.*Yzi*Yk*Yls/b/mm
sigmF2=Fmt*KF.*Yfs2.*Yzi*Yk*Yls/b/mm
sigmFmin1=SigmaFlim1.*YNX1./SFmin
sigmFmin2=SigmaFlim2.*YNX2./SFmin
 c=[sigmH-sigmHmin
   sigmF1-sigmFmin1;
   sigmF2-sigmFmin2;
    17*u/sqrt(1+u^2)-x(2);
    0.25-x(3);
    x(3)-0.3;
    1.5-x(1)]
ceq=[ ]
```

4) Main Program

```
%bevelgearopt.m
%define global variables
global u    % global variable speed ratio
global  p   %power
global n1   %rotate speed
global Zk   %bevel gear factor
global Ze   %elastic influence factor
global Zh   %area factor
…
T1=9549*p/n1
kA=gearkA(orgm,wkm)   %load factor
Zh=2*sqrt(1/sin(2*20/180));
Ze=gearZE(m1,m2)   %elasticity factor
Zk=0.8
x0=[2.5;34;0.3];
a=[]
b=[]
aeq=[];
beq=[];
lb=[];
ub=[];
%fitnessfunction
```

```
fitnessFunction = @bevelobj;
%Set optimization options:
options = gaoptimset; %fault options
% change optimization options
options = gaoptimset(options,'PopInitRange' ,[1 1 1 ; 36 28 6 ]);
options = gaoptimset(options,'MigrationDirection' , 'both');
options = gaoptimset(options,'TolFun' ,0.001);
options = gaoptimset(options,'TolCon' ,0.001);
options = gaoptimset(options,'SelectionFcn', {@selectiontournament 4 });
options = gaoptimset(options,'MutationFcn', @mutationadaptfeasible);
options = gaoptimset(options,'HybridFcn' ,{ @fmincon [ ]})
%Call ga( ) function:
[X,FVAL]=ga(@FitnessFcn,nvars,A,b,Aeq,beq,LB, UB, @bevelnonlcon,options)
```

VII. INTERFACE BETWEEN VISUAL BASIC LANGUAGE AND MATLAB SOFTWARE

Interface between Visual Basic language and MATLAB software can be realized by three methods, DLL(dynamic linked library), DDE(dynamic data exchange) or ActiveX technology. In the paper, the third method is adopted. Partial programs are as follows:

```
Private Sub OptiCommand_Click()
'Announce ActiveX loading MATLAB
Dim matlab As Object

'Loading MATLAB application program
Set matlab = CreateObject("Matlab.Application")

'Define variables for input parameters
Dim p As Double, n1 As Double, u As Double
Dim z1 As Double
Dim fR As Double
Dim mi As Double, mj As Double   'gear material
…
'Define variables for output results
Dim outm As Variant
Dim outz1 As Variant, outz2 As Variant
…
'The known data in UI are transmitted to input parameters
p = Val(Textp.Text)
u = Val(Textu.Text)
n1 = Val(Textn1.Text)
z1 = Val(Textz1.Text)
fr= Val(TextfR.Text)
…
'Visual Basic variables are transmitted to MATLAB workspace
Call MATLAB.PutWorkspaceData("orgm", "base", orgm) 'original machine working conditions
Call MATLAB.PutWorkspaceData("wkm", "base", wkm) 'work machine working conditions
Call MATLAB.PutWorkspaceData("pregrd", "base", accgrd) 'precision grade
Call MATLAB.PutWorkspaceData("p", "base", p)
Call MATLAB.PutWorkspaceData("n1", "base", n1)
Call MATLAB.PutWorkspaceData("u", "base", u)
Call MATLAB.PutWorkspaceData("z1", "base", z1)
Call MATLAB.PutWorkspaceData("m", "base", min)
Call MATLAB.PutWorkspaceData("fr", "base", fr)
Call MATLAB.PutWorkspaceData("supp", "base", supp)
'Executing the MATLAB program of bevel gear optimization design
matlab.execute ("bevelgearopt")
'Results from MATLAB are transmitted to Visual Basic
Call MATLAB.GetWorkspaceData("M", "base", mout)
Call MATLAB.GetWorkspaceData("Z1","base", outz1)

…
'Optimization results are transmitted to UI
Textoutm.Text = mout
Textoutz1.Text =outz1
…
```

VIII. APPLICATION EXAMPLE

It is known that input power P=5kw, small gear speed n1=960r/min, gear ratio u=4.8; load stability, expected life is 15 years, 300 days a year, accounting for 30% of working time. Work condition is steady, pinion is cantilever, big gear is two-support.

Result compared with the results of traditional optimization design and conventional design is shown in Table II.

TABLE II.
RESULTS COMPARED WITH THE TRADITIONAL OPTIMIZATION DESIGN AND CONVENTIONAL DESIGN

Design method	Module	Tooth number	Face width coefficient	Volume
Genetic algrithm	2.75	24	0.3	6.8834e+005
Traditional optimization	3	33	0.27	2.1604e+006
Conventional design	3	34	0.3	2.5408e+006

IX. CONCLUSION

Optimization design software for straight bevel gear is developed in the paper to achieve the automatic optimization by using VB and MATLAB, in which the genetic algorithm is selected and the augmented penalty function and integer serial number encoding is used, so the global optimal solution can be obtained, design efficiency greatly improved, weight of bevel gear reduced, the production cycle shortened.

REFERENCES

[1] Zeng Qiang, "The R & D of Gear CAD system based on artificial intelligence," Xi'an Architecture and Technology university, 2003

[2] Liang Shangming, Yin Guofu, “Modern mechanical optimization Methods,” Beijing,: Chemical Industry Press, 2005, pp.231-234
[3] Ang Xueye, Ding Jianmei, “3-Dimensional parametrical modeling method of straight bevel gear,” engineering graphics transaction, vol. 6, 2007, pp.22~25
[4] Bi Changchun, Ding Yuzhan, “Application of real number encoding genetic algorithm in helical gear drive optimization design,” mechanical science and technology, vol. 11, 2000, pp.82-84
[5] Chen Xiuning, “Mechanical optimization Design,” Zhejiang University Press, Hangzhou, 1997, pp. 252-260
[6] Cheng Daxian, *Mechanical Design Handbook*. Machine drive, Beijing: Chemical Industry Press, 2004.
[7] Zhu Xiaolu, *Gear drive design manual*, Beijing: Chemical Industry Press, 2005.
[8] The MathWorks, Inc, http://www.mathsworks.com
[9] Laumanns M, Thiele L, Zitzler E, Welzl E, Deb K, “Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem,” Parallel Problem Solving from Nature—PPSN vol. VII, Spain: Granada, 2002, pp.44-53
[10] Schwefel H. *Evolution and optimum seeking*, New York: John Wiely &Sons, 1995.

Xiaoqin Zhang: born in Tangshan, China in March, 1971. A PHD Candidate, Mechanical Engineering College, Yanshan University, Qinhuangdao, China, Major in CAD/CAM.