Московский Государственный Технический Университет
им. Н. Э. Баумана

Курсовая работа по курсу:

**“Нелинейные системы автоматического управления”**

“Метод Попова”

Студент: Серебряков И. В.

Факультет: Аэрокосмический

Группа: АК 4-81

Преподаватель:

профессор Хабаров В. С.

Москва

1997

Список литературы:

1. Боднер В. А. “ Системы управления летательными аппаратами”,

М.: “Машиностроение”, 1973.

2. Воронов А. А. “Устойчивость, управляемосчть, наблюдаемость”

Главная редакция физико-математической литературы

издательства “ Наука”, М., 1979, 336 стр.

3. Цыпкин Я. З. “ Релейные автоматические системы “

Издательство “ Наука ”, главная редакция

физико-математической литературы, М., 1974, 576 стр.

Дополнительно использованные средства.

1. Для построения АЧХ и ФЧХ системы был использован программный пакет по системам управления “ Classic Control ”.

2. Оформление курсовой работы производилось в текстово-графическом редакторе “ Microsoft Word ” ver. 6.0, под оболочкой Windows 95.

3. Приложенная программа была написана на языке С++, под оболочкой Borland C++ ver. 3.0 .

# Устойчивость “в малом” и “ в большом”. Связь критерия Попова с методами Ляпунова.

Пусть линейная система устойчива в секторе (0, К)-см рис. 5.9; начальная часть нелинейной характеристики, соответствующая­ , лежит внутри этого сектора, а при выходе х за указанные пределы выходит за пределы сектора. Очевидно, что в данном случае нельзя утверждать, что равновесие системы будет абсолютно устойчиво, т.е. устойчиво в целом при любых f(l), но мы можем утверждать, что при таких , которые вызывают отклонение х, не выходящее за пределы (-х2, х1), будет имеет место устойчивость положения равновесия в большом и, конечно, устойчивость в малом.

С помощью критерия Попова легко можно пояснить, когда применим первый метод Ляпунова. Заменим нелинейную характеристику в точке равновесия касательной (рис. 5-10). Если линейная система устойчива (а не находится на границе устойчивости), то небольшой подъем луча 0К в положение 0К1 не нарушит устойчивости, то при этом начальная часть нелинейной характеристики попадает внутрь сектора (0, К1), и равновесие нелинейной системы будет устойчивым в малом.

рис. 5-9. рис. 5-10.

Если же мы имеем критический случай, то касательная является границей сектора, внутри которого линейная система устойчива, и мы не можем судить об устойчивости равновесия нелинейной системы.

Функция Ляпунова может быт построена различными способами для одной и той же системы. Для каждой такой частной функции Ляпунова можно построить свою область устойчивости в пространстве параметров, но каждая такая область не будет истинной областью устойчивости, поскольку второй метод Ляпунова дает лишь достаточное условие устойчивости.

Р. Калман показал, что область устойчивости, даваемая критерием Попова, будет огибающей для всех областей устойчивости, определяемых функциями Ляпунова вида “квадратичная форма плюс нелинейность”, т.е. будет шире и ближе к истинной области устойчивости, чем любая из областей устойчивости, определяемая по функции Ляпунова заданной формы.

Большим преимуществом метода Попова является то, что он без особых затруднений распространяется на системы с запаздыванием и распределенными параметрами, а также на некоторые классы импульсных систем управления.

Рассмотренные критерии - квадратичный, вытекающий и него круговой и критерий Попова - различаются степенью подробности учета специфических особенностей нелинейных характеристик, что отражается на ширине области устойчивости, даваемой тем или иным критерием, т.е. лучшим критерием является тот, который дает более широкую область устойчивости.

Если сравнивать круговой критерий с методом Попова, то первый дает более узкую область устойчивости, если исследуется класс стационарных нелинейностей, но зато охватывает более широкий класс нелинейностей.

#include <graphics.h>

#include <stdio.h>

#include <iostream.h>

#include <conio.h>

#include <dos.h>

#include <math.h>

typedef float (\*Otv)(float w);

unsigned char GraphMode(void);

void BackGround(void);

float Formula1(float w);

float Formula2(float w);

void ShowGrafic(void);

void LinePopov(void);

 float Xmas=30; // Масштаб

 float Ymas=30;

main()

{

 clrscr();

 if (GraphMode()==0){ // Инициализация графики

 BackGround(); // Построение СКО

 ShowGrafic(); } // Построение графиков

 return 0;

}

unsigned char GraphMode(void)

{

 int driver, mode;

 unsigned char err;

 driver = DETECT;

 initgraph(&driver,&mode,"");

 err = graphresult();

 if (err!=grOk) {cout<<"\n\t"<<grapherrormsg(err);

 getch();

 return 1;}

 return 0;

}

void BackGround(void)

{

 int xmax = getmaxx();

 int ymax = getmaxy();

 setcolor(15);

 rectangle(0,0,xmax,ymax);

 line((int)(xmax/2),10,(int)(xmax/2),ymax-10);

 line(10,(int)(ymax/2),xmax-10,(int)(ymax/2));

 line((int)(xmax/2)-3,15,(int)(xmax/2),10);

 line((int)(xmax/2),10,(int)(xmax/2)+3,15);

 line(xmax-15,(int)(ymax/2)-3,xmax-10,(int)(ymax/2));

 line(xmax-15,(int)(ymax/2)+3,xmax-10,(int)(ymax/2));

 settextstyle(2,0,5);

 outtextxy((int)(xmax/2)+7,10,"jwQ(w)");

 outtextxy(xmax-35,(int)(ymax/2)+7,"P(w)");

 settextstyle(2,0,4);

 outtextxy((int)(xmax/2)-8,(int)(ymax/2)+1,"0");

 setcolor(15);

}

void ShowGrafic(void)

{

 Otv Re=Formula1;

 Otv Im=Formula2;

 int xmax = getmaxx();

 int ymax = getmaxy();

 for(float Step=0; Step<=40; Step=Step+0.0001){ //procedure Grafic

 putpixel(xmax/2+Xmas\*(\*Re)(Step),ymax/2+Ymas\*(\*Im)(Step),15);

 }

 setcolor(11);

 line((int)xmax/2+Xmas,(int)(ymax/2)+4,(int)(xmax/2)+Xmas,(int)(ymax/2)-4);

 line((int)xmax/2-Xmas,(int)(ymax/2)+4,(int)(xmax/2)-Xmas,(int)(ymax/2)-4);

 LinePopov();

 while(!kbhit()); // Ожидание нажатия

 closegraph();

}

float Formula1(float w)

{

 return (0.0044\*w\*w\*w\*w-0.223\*w\*w-1)/

 (-(0.63\*w-0.009\*w\*w\*w)\*(0.63\*w-0.009\*w\*w\*w)-(1-0.087\*w\*w)\*(1-0.087\*w\*w));

}

float Formula2(float w)

{

 return -w\*w\*(0.034\*w\*w+0.14)/

 (-(0.63\*w-0.009\*w\*w\*w)\*(0.63\*w-0.009\*w\*w\*w)-(1-0.087\*w\*w)\*(1-0.087\*w\*w));

}

void LinePopov(void)

{

float PrXo, PXo, QXo, Xo, Yo, h;

float Y;

float pr2= 15;

float kol, time, otv;

Otv Fx1=Formula1;

Otv Fx2=Formula2;

int xmax = getmaxx();

int ymax = getmaxy();

 signed char done=0;

 h=0.00001; // Погрешность производной

 Xo=-0.92; // Задание точки Попова

 Yo=0; // на реальной оси (-1/k)

 time=7;

 while(!done){ // Нахождение точки касания прямой Попова

 // и модифицированного годографа

 time=time+0.0001;

 PXo=(\*Fx1)(time); // Проекция w на Р

 QXo=(\*Fx2)(time); // Проекция w на wQ

 PrXo=((\*Fx2)(time)-(\*Fx2)(time-h))/((\*Fx1)(time)-(\*Fx1)(time-h));

 otv=PrXo\*(Xo-PXo)+QXo-Yo;

 if ((otv>=-0.001)&&(otv<=0.001)) done=1;

 if (time==pr2) done=1;

 putpixel(xmax/2+Xmas\*(\*Fx1)(time),ymax/2+Ymas\*(\*Fx2)(time),13);

 gotoxy(2,2);

 printf("PXo %3.4f",PXo);

 gotoxy(2,3);

 printf("QYo %3.4f",-QXo);

 printf("\nOtvet %f",otv);

 gotoxy(2,5);

 printf("w %2.3f",time);

 }

 gotoxy(2,6);

 printf(" Угол равен %2.2f", -atan(QXo/PXo)\*180/3.14);// Угол касательной

 rectangle(0,0,150,90); // к Р

for (kol=-1.5; kol<=1.5; kol=kol+0.001){

Y=PrXo\*(kol-Xo)+Yo;

putpixel(xmax/2+Xmas\*(kol),ymax/2+Ymas\*Y,10); // Прямая Попова

 }

 }

...