УДК 62-5:519.6

СИНТЕЗ НЕЧЕТКИХ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ ГЕНЕТИЧЕСКИМИ АЛГОРИТМАМИ ПО ВЕКТОРНЫМ КРИТЕРИЯМ В СРЕДЕ MATLAB

Северин В. П.
Национальный технический университет «Харьковский политехнический институт», Харьков,
e-mail: severinv@mail.ru

Введение

Повышение качества современных систем автоматического управления (САУ) возможно путем развития методов синтеза оптимальных систем [1]. В таких системах управления вместо стандартных ПИД регуляторов предлагается применять интеллектуальные регуляторы, построенные на основе нечеткой логики. В задачах оптимизации систем управления с нечеткими регуляторами характерно наличие большого числа локальных экстремумов [2]. Многоэкстремальные задачи оптимизации успешно решаются генетическими алгоритмами. Однако значительные затраты времени на решение таких задач сдерживают их применение в задачах управления, отличающихся большими объемами вычислений целевых функций интегрального вида [2]. Целесообразно провести развитие методов синтеза систем управления путем использования генетических алгоритмов как методов оптимизации [3-12].

Одной из эффективных математических систем, позволяющих решать задачи оптимизации, является система MATLAB [13-17]. Оптимизационный инструментарий этой системы Optimization Toolbox обладает многими m-функциями для решения типичных задач оптимизации. Однако при большой сложности этих m-функций возможности инструментария для целей изучения, исследования и развития самих методов и задач оптимизации ограничены. В нем недостаточно развиты средства табличного представления процесса оптимизации, отсутствует возможность графического отображения этого процесса, не предусмотрено пакетное исследование методов и задач, затруднено развитие инструментария для решения новых классов оптимизационных задач. В то же время система MATLAB, обладающая обширной библиотекой матричных вычислений и линейной алгебры, развитыми средствами программирования и визуализации, позволила создать компьютерную лабораторию методов оптимизации OPTLAB для реализации широкого набора оптимизационных моделей и методов, с развитыми возможностями табличного и графического представления результатов оптимизации [18-21]. С помощью этой лаборатории разработаны программы множества известных методов оптимизации и решены многие задачи оптимизации реальных систем.

Цель статьи состоит в представлении результатов разработки и исследования методов многокритериального параметрического синтеза систем автоматического управления с линейными и нечеткими ПИД регуляторами в среде MATLAB по улучшенным интегральным квадратичным оценкам и прямым показателям качества с использованием генетических алгоритмов.

Рассматриваются бинарный и непрерывный генетические алгоритмы, проводится анализ их эффективности. Исследуется многокритериальность задач синтеза систем

управления и показывается противоречивость частных критериев качества. Рассматриваются улучшенные интегральные квадратичные оценки в виде свертки частных оценок. На основании условий технической реализуемости и условий устойчивости систем управления формируется векторная штрафная функция. Обосновывается формирование многокритериального критерия в виде векторной целевой функции с прямыми показателями качества и улучшенными интегральными квадратичными оценками. Разрабатываются модифицированные генетические алгоритмы для оптимизации векторных целевых функций. Рассматривается синтез систем управления по улучшенным интегральным квадратичным оценкам и прямым показателям качества, проводится сравнение различных генетических алгоритмов. Разрабатываются методы параметрического синтеза линейных и нечетких ПИД регуляторов с использованием улучшенных интегральных квадратичных оценок, прямых показателей качества, векторных целевых функций и модифицированных генетических алгоритмов.

Основные принципы генетических алгоритмов

Генетические алгоритмы (ГА) основаны на идеях популяционной генетики и представляют методы глобального поиска экстремума целевой функции $f(x)$, $x \in R^p$ в гипербрусе, определяемом двусторонним неравенством $a \le x \le b$.

Введем математические обозначения для формализации действий в ГА. Каждую особь популяции представим хромосомой $\xi = (\xi_1, \xi_2, \ldots, \xi_p)$, состоящей из бинарных или двоичных генов: $\xi_i = (\xi_{i1}, \xi_{i2}, \ldots, \xi_{im})$, $i = \overline{1, p}$, $\xi_{ij} \in \{0; 1\}$. Здесь m — количество битов в гене, то есть длина двоичной последовательности гена. С учетом значений генов хромосома представляется бинарной строкой длины $M = mp$.

Каждой хромосоме ξ, определяющей генотип особи, отвечает вектор переменных параметров $x = (x_1, x_2, \ldots, x_p) \in R^p$, соответствующий фенотипу особи. Пусть $x_i \in [a_i, b_i] \in R$, $i = \overline{1, p}$, и требуется найти решение с q десятичными знаками для каждой переменной x_i. Для этого интервал $[a_i, b_i]$ длины $l_i = b_i - a_i$ разбивается на $10^q l_i$ одинаковых подынтервалов. Это означает применение шага дискретизации $r = 10^{-q}$. Необходимую и достаточную длину двоичной последовательности, требуемой для кодирования десятичного числа x_i из интервала $[a_i, b_i]$ с шагом r, определяет наименьшее натуральное число m, удовлетворяющее системе неравенств $10^q l_i \le 2^m - 1$, $i = \overline{1, p}$. Решая эту систему неравенств, получим длину гена

$$m = \max_i [\log_2 (10^q l_i + 1)] + 1.$$

В табл. 1 представлены значения рассчитанной по этой формуле длины гена m для различных значений параметра точности q и одинаковых длин интервалов $l = l_i$ переменных параметров.

Таблица 1. Зависимость длины гена от параметра точности и длины интервала

q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$l = 1$	4	7	10	14	17	20	24	27	30	34	37	40	44	47	50	54
$l = 10$	7	10	14	17	20	24	27	30	34	37	40	44	47	50	54	57
$l = 100$	10	14	17	20	24	27	30	34	37	40	44	47	50	54	57	60

Так, для получения решения с восьмью десятичными знаками при $q = 8$ необходимо использовать в случае длины интервала $l = 1$ длину гена $m = 27$, а в случае $l = 100$ — $m = 34$. Положим $m = 32$.

Каждому значению гена ξ_i в виде двоичной последовательности с учетом перевода двоичных чисел в десятичную форму соответствует десятичное представление числа

$$d_i = \xi_{i1} 2^{m-1} + \xi_{i2} 2^{m-2} + \ldots + \xi_{im} 2^0 . \quad (1)$$

Значение переменного параметра можно представить выражением

$$x_i = a_i + \frac{l_i}{2^m - 1} d_i . \quad (2)$$

Таким способом задаются фенотипы, соответствующие кодовым последовательностям с длиной m. Выражения (1) и (2) — это следствие линейного отображения интервала $[0, 2^m - 1]$ на интервал $[a_i, b_i]$. Иногда вместо двоично-десятичного кода применяется код Грея.

Для поиска оптимального решения используется популяция особей в виде массива хромосом $\Pi = (\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(N)})$, где N — число особей, определяющее размер популяции. Популяция хромосом отображается по формулам (1) и (2) в популяцию точек $P = (x^{(1)}, x^{(2)}, \ldots, x^{(N)})$ пространства переменных параметров R^p. В отличие от первоначальных ГА, применявшихся для максимизации положительной целевой функции как функции приспособленности, для минимизации целевой функции $f(x)$ произвольного знака используется массив ее значений $\Phi = (f^{(1)}, f^{(2)}, \ldots, f^{(N)})$ для особей популяции. Таким образом, каждая особь с номером $j \in \overline{1, N}$ представляется хромосомой $\Pi_j = \xi^{(j)}$, точкой $P_j = x^{(j)}$ пространства параметров и значением целевой функции $\Phi_j = f^{(j)}$.

Генетические алгоритмы являются итерационными, и каждая итерация в них называется эпохой. На k-той эпохе формируется популяция особей, которой соответствуют массивы хромосом $\Pi^{(k)}$, точек $P^{(k)}$ и значений целевой функции $\Phi^{(k)}$. Массив хромосом начальной популяции $\Pi^{(0)}$ формируется случайным образом. Последующие популяции формируются с применением генетических операторов: селекции, кроссовера и мутации.

Для минимизации целевой функции произвольного знака целесообразно применить комбинацию ранговой селекции с принципом рулетки. Ранговая селекция основана на ранжировании особей популяции по значениям целевой функции. При этом списки особей Π, P и Φ сортируются с использованием операций сравнения значений целевой функции, то есть упорядочиваются от наиболее приспособленных особей с лучшими (меньшими) значениями целевой функции к наименее приспособленным особям с худшими (большими) значениями функции.

Из наиболее приспособленных особей выбираются N_p хромосом родительских особей Π_p, $N_p < N$. Каждой родительской особи по ее месту j в списке приписывается ранг $r_j = N_p + 1 - j$, $j = \overline{1, N_p}$. Это означает нумерацию родительских особей в обратном порядке. Сумма всех рангов равна сумме арифметической прогрессии: $S_r = 0{,}5(N_p + 1)N_p$.

Колесу рулетки сопоставляется интервал $[0; 1]$, а каждой родительской особи — сектор длиной $p_j = r_j / S_r$, соответствующий вероятности попадания особи в сектор. По

принципу рулетки из N_p родительских особей выбирается $n_p = [0{,}5 \cdot (N - N_p)]$ номеров первых N_1 и вторых N_2 родителей для скрещивания. По этим номерам одноточечным кроссовером над массивом родительских особей Π_p формируются массивы хромосом потомков π_1 и π_2. Формируется промежуточная популяция хромосом путем объединения массивов Π_p, π_1 и π_2, после чего она подвергается мутации и выполняется переход к новой популяции.

Бинарный генетический алгоритм

Бинарный генетический алгоритм (ГА) использует бинарные хромосомы особей популяции для глобального поиска экстремума целевой функции $f(x)$, $x \in R^p$ в гипербрусе, определяемом двусторонним неравенством $a \le x \le b$. Простой бинарный ГА можно представить следующим алгоритмом.

Бинарный генетический алгоритм *GAB*.

Входные параметры: a и b — векторы нижних и верхних ограничений переменных параметров, N — размер популяции, m — длина гена, r_p — доля родительских пар в популяции, r_m — доля мутации, N_k — предельное количество итераций.

Выходные параметры: x и f_x — лучшая точка поиска и значение в ней целевой функции.

1. Положить $p = \dim(a)$, $l = b - a$, $M = m \cdot p$, $N_p = [r_p \cdot N]$, $n_p = [0{,}5 \cdot (N - N_p)]$, $n_m = [(N-1) \cdot r_m \cdot M]$, $k = 1$.
2. Вычислить $\Pi = Round(Rand(N, M))$.
3. Вычислить $P = Decod(\Pi, a, l, m)$.
4. Для $j = \overline{1, N}$ вычислить $\Phi_j = f(P_j)$.
5. Положить $(\Phi, J) = Sort(\Phi)$, $\Pi = \Pi(J)$, $P = P(J)$, $\Pi_p = \Pi(1 : N_p)$.
6. Вычислить $(N_1, N_2) = Roulette(N_p, n_p)$, $(\pi_1, \pi_2) = Crossover(\Pi_p, N_1, N_2)$.
7. Положить $\Pi = (\Pi_p, \pi_1, \pi_2)$.
8. Вычислить $\Pi = Mutate(\Pi, n_m)$.
9. Вычислить $P = Decod(\Pi, a, l, m)$.
10. Для $j = \overline{1, N}$ вычислить $\Phi_j = f(P_j)$.
11. Положить $(\Phi, J) = Sort(\Phi)$, $\Pi = \Pi(J)$, $P = P(J)$, $\Pi_p = \Pi(1 : N_p)$.
12. Положить $x = P_1$, $F_x = \Phi_1$.
13. Если $k < N_k$, положить $k = k + 1$ и перейти к п. 6.
14. Выход.

В этом алгоритме на шаге 1 инициализируются основные параметры ГА: равное размерности вектора a число переменных параметров p, вектор длин интервалов переменных параметров l, длина хромосомы M, число родительских особей N_p, число пар родителей n_p, количество мутаций n_m, счетчик числа итераций k.

На шаге 2 создается начальная популяция хромосом Π путем формирования

процедурой *Rand* матрицы размерности $N \times M$ случайных вещественных чисел из интервала $[0;1]$ и округления этих чисел до ближайших целых значений 0 или 1 процедурой *Round*. На шаге 3 популяция хромосом декодируется по формулам (2) и (3) процедурой *Decod* в множество точек P пространства переменных параметров R^p. На шаге 4 вычисляются значения векторной целевой функции для всех особей популяции.

На шаге 5 выполняется сортировка особей по значениям проекций вектора целевой функции Φ в порядке ухудшении значений. Применяется процедура пузырьковой сортировки *Sort*. В результате выполнения этой процедуры получен отсортированный вектор значений функции Φ и соответствующий ему массив исходных индексов проекций J, который используется для сортировки массивов хромосом Π и соответствующих им точек P. Из массива хромосом Π выбирается N_p первых хромосом, составляющие «родительский пул» Π_p.

На шаге 6 по принципу рулетки с ранговой селекцией процедурой *Roulette* из N_p родительских особей выбирается n_p номеров первых N_1 и вторых N_2 родителей для скрещивания. С хромосомами выбранных родителей процедурой *Crossover* выполняется одноточечный кроссовер и формируются массивы хромосом потомков π_1 и π_2. На шаге 7 из родительских особей и потомков формируется промежуточная популяция хромосом. На шаге 8 выполняется мутация этой популяции процедурой *Mutate*. На шагах 9–11 опять выполняется декодирование хромосом, вычисление значений целевой функции и сортировка популяции. На шаге 12 определяется лучшая точка текущей популяции. На шаге 13 проверяется критерий завершения ГА по достижению счетчиком итераций k предельного количества итераций N_k.

Итерацию ГА составляют шаги 6–13. По умолчанию задаются значения входных параметров алгоритма $r_p = 0{,}5$, $r_m = 0{,}1$. Размер популяции обычно определяется числом переменных — $N = 8p$.

Программа бинарного ГА *GAB* составлена в лаборатории методов оптимизации OPTLAB среды MATLAB в виде m-функции `metGAbin`.

```
function metGAbin    % Бинарный генетический алгоритм (ГА)
global STATE         % структура данных процесса оптимизации
npar = optget('n');  % число переменных параметров
T = optget('Task');  % данные задачи оптимизации
varlo = (T.a)'; varhi = (T.b)'; % массивы ограничений переменных
c = varhi - varlo;   % вектор длин интервалов переменных
popsize = 8*npar;    % размер популяции
nbits = 32;          % длина гена - число бит для каждого параметра
selection = 0.5;     % доля родительских пар в популяции
mutrate = .1;        % доля мутации
Nt = nbits*npar;     % длина бинарной хромосомы
keep = floor(selection*popsize); % число родительских особей
M = ceil((popsize-keep)/2);      % число пар родителей
nmut = ceil((popsize-1)*Nt*mutrate); % общее количество мутаций
%              Создание начальной популяции
pop = round(rand(popsize,Nt));     % массив хромосом начальной популяции
par = gadecode(pop,varlo,c,nbits); % декодирование массива хромосом
how = 'нач попул';
for j = 1:popsize % вычисление значений целевой функции
  cost(j) = newPoint(par(j,:)',how);
  if stateget('exit')
    return
```

```
  end
end
[cost,ind] = sort(cost); % сортировка значений целевой функции
par = par(ind,:); pop = pop(ind,:); % сортировка популяции
STATE.mNx(1) = STATE.Nx;  % номер лучшей особи популяции
fi = double(cost(1));     % лучшее значение целевой функции
stateset('fi',fi)
STATE.Fi(:,1) = fi;
fm = mean(cost);
stateset('fm',fm)    % среднее значение целевой функции в популяции
STATE.Fm(:,1) = fm;
onesp = ones(popsize-1,1);
H = max(max(abs(par(2:popsize,:)-par(onesp,:)))); % Погрешность
stateset('H',H)
while 1  %       Итерационный цикл
  iterCount  %  счетчик числа итераций
  %                Селекция родителей
  prob = flipud([1:keep]'/sum([1:keep])); % веса хромосом
  odds = [0 cumsum(prob(1:keep))'];   % колесо рулетки
  pick1 = rand(1,M);                  % массив первых родителей
  pick2 = rand(1,M);                  % массив вторых родителей
  % ma и pa содержат индексы хромосом, которые будут скрещиваться
  ic = 1;
  while ic<=M
    for id = 2:keep+1
      if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
         ma(ic) = id - 1;
      end
      if pick2(ic)<=odds(id) & pick2(ic)>odds(id-1)
         pa(ic) = id - 1;
      end
    end % id
    ic = ic + 1;
  end % while ic
  %   Скрещивание с использованием одноточечного кроссовера
  ix = 1:2:keep;                                  % индекс первого родителя
  xp = ceil(rand(1,M)*(Nt-1));                    % точки кроссовера
  pop(keep+ix,:)=[pop(ma,1:xp) pop(pa,xp+1:Nt)];  % первый потомок
  pop(keep+ix+1,:)=[pop(pa,1:xp) pop(ma,xp+1:Nt)];% второй потомок
  %                       Мутация
  mrow=ceil(rand(1,nmut)*(popsize-1))+1;    % номера строк мутаций
  mcol=ceil(rand(1,nmut)*Nt);               % номера столбцов мутаций
  for ii = 1:nmut
      pop(mrow(ii),mcol(ii)) = abs(pop(mrow(ii),mcol(ii)) - 1); % мутация
  end
  %   Перевычисление популяции
  par(2:popsize,:) = gadecode(pop(2:popsize,:),varlo,c,nbits);% декодиров
  how = 'новая поп';
  for j = 2:popsize
    cost(j) = newPoint((par(j,:))',how);
    if stateget('exit')
       return
    end
  end
  [cost,ind] = sort(cost); % сортировка особей популяции
  par = par(ind,:); pop=pop(ind,:);
  fi = double(cost(1)); % лучшее значение целевой функции
  stateset('fi',fi)
  STATE.Fi(:,STATE.Ni) = fi;
  stateset('fm',mean(cost)) % среднее значение функции в популяции
  H = max(max(abs(par(2:popsize,:)-par(onesp,:)))); % погрешность
```

```
  stateset('H',H)
  if stateiter return, end % критерий останова
end % while 1
```

В этой программе используется m-функция декодирования двоичных хромосом gadecode.

```
function par = gadecode(chrom,lo,c,bits)
[M,N] = size(chrom);                % размерность массива хромосом
npar = N/bits;                      % число переменных
quant = (0.5.^[1:bits]');           % уровни дискретизации (степени 2)
quant = quant/sum(quant);           % нормализация уровней
ct = reshape(chrom',bits,npar*M)'; % нормализация двоичного кода переменной
par = ct*quant;                     % нормализованный десятичный код
par = reshape(par,npar,M)';         % массив десятичных кодов переменных
for j = 1:M
  par(j,:) = par(j,:).*c + lo;      % формирование массива точек
end
```

Следует отметить, что здесь представлен один из простейших бинарных ГА, для которых существует множество различных модификаций.

Непрерывный генетический алгоритм

Наряду с бинарными ГА применяются и непрерывные ГА, в которых исходной является не популяция двоичных хромосом Π, а популяция точек P пространства переменных параметров задачи R^p. В этом случае ГА упрощается, поскольку исчезает необходимость декодирования хромосом. Массив точек начальной популяции $P^{(0)}$ формируется случайным образом. Последующие популяции формируются с применением генетических операторов: ранговой селекции с принципом рулетки, а также непрерывных кроссовера и мутации [198]. Непрерывный ГА можно представить следующим алгоритмом.

Непрерывный генетический алгоритм *GAC*.

Входные параметры: a и b — векторы нижних и верхних ограничений переменных, N — размер популяции, r_p — доля родительских пар, r_m — доля мутации, N_k — предельное количество итераций.

Выходные параметры: x и f_x — лучшая точка поиска и значение в ней целевой функции.

1. Положить $k=1$, $p=\dim(a)$, $l=b-a$, $N_p=[r_p\cdot N]$, $n_p=[0{,}5\cdot(N-N_p)]$, $n_m=[(N-1)\cdot r_m\cdot p]$.
2. Для $j=\overline{1,N}$ вычислить $P_j=a+l\cdot Rand(1,p)$, $\Phi_j=f(P_j)$.
3. Положить $(\Phi,J)=Sort(\Phi)$, $P=P(J)$, $P_p=P(1:N_p)$.
4. Вычислить $(N_1,N_2)=Roulette(N_p,n_p)$, $(p_1,p_2)=CrossoverC(P_p,N_1,N_2)$.
5. Положить $P=(P_p,\ p_1,\ p_2)$.
6. Вычислить $P=MutateC(P,n_m)$.
7. Для $j=\overline{1,N}$ вычислить $\Phi_j=f(P_j)$.
8. Положить $(\Phi,J)=Sort(\Phi)$, $P=P(J)$, $P_p=P(1:N_p)$.

9. Положить $x = P_1$, $F_x = \Phi_1$.

10. Если $k < N_k$, положить $k = k + 1$ и перейти к п. 4.

11. Выход.

Здесь на шаге 1 инициализируются основные параметры ГА: счетчик числа итераций k, число переменных параметров p, вектор длин интервалов переменных параметров l, число родительских особей N_p, число пар родителей n_p, количество мутаций n_m.

На шаге 2 случайным образом создается начальная популяция точек P с использованием процедуры *Rand* и вычисляется массив значений векторной целевой функции Φ. На шаге 3 выполняется сортировка особей по массиву Φ процедурой *Sort*, и по отсортированному множеству индексов J сортируется массив точек P. Из массива P выбирается N_p первых точек, составляющие «родительский пул» P_p.

На шаге 4 процедурой *Roulette* из N_p родительских особей выбирается n_p номеров первых N_1 и вторых N_2 родителей для скрещивания. С координатами точек выбранных родителей процедурой *CrossoverC* выполняется одноточечный кроссовер и формируются массивы точек потомков p_1 и p_2. На шаге 5 из родительских особей и потомков формируется промежуточная популяция точек. На шаге 6 выполняется непрерывная мутация этой популяции процедурой *MutateC*.

На шагах 7–8 опять вычисляются значения целевой функции, и популяция сортируется. На шаге 9 определяется лучшая точка популяции. На шаге 10 проверяется критерий завершения ГА. Итерационный цикл ГА составляют шаги 4–10. По умолчанию задаются значения входных параметров алгоритма $r_p = 0{,}5$, $r_m = 0{,}2$.

Программа непрерывного ГА составлена в лаборатории методов оптимизации OPTLAB среды MATLAB в виде m-функции `metGAcon`.

```
function metGAcon    % Непрерывный генетический алгоритм (ГА)
global STATE         % структура данных процесса оптимизации
npar = optget('n');  % число переменных параметров
T = optget('Task');  % данные задачи оптимизации
varlo = (T.a)'; varhi = (T.b)'; % массивы ограничений переменных
c = varhi - varlo;   % вектор длин интервалов переменных
popsize = 8*npar;    % размер популяции
selection = 0.5;     % доля родительских пар в популяции
mutrate = .1;        % доля мутации
Nt = npar;           % длина хромосомы
keep = floor(selection*popsize); % число родительских особей
M = ceil((popsize-keep)/2);      % число пар родителей
nmut = ceil((popsize-1)*Nt*mutrate); % общее количество мутаций
%               Создание начальной популяции
[fx,x] = stateget('fx','x');
par = zeros(popsize,npar); % предразмещение популяции
par(1,:) = x; cost(1) = fx;
how = 'нач попул';
for j = 2:popsize % Формирование начальной популяции
   x = varlo + c.*rand(1,npar); par(j,:) = x;
   cost(j) = newPoint(x',how);
   if stateget('exit')
      return
   end
end
[cost,ind] = sort(cost); % сортировка значений целевой функции
```

```
par = par(ind,:);         % сортировка популяции
STATE.mNx(1) = STATE.Nx; % номер лучшей особи популяции
fi = double(cost(1));     % лучшее значение целевой функции
stateset('fi',fi)
STATE.Fi(:,1) = fi;
fm = mean(cost);
stateset('fm',fm)         % среднее значение целевой функции в популяции
STATE.Fm(:,1) = fm;
onesp = ones(popsize-1,1);
H = max(max(abs(par(2:popsize,:)-par(onesp,:)))); % погрешность
stateset('H',H)
while 1   %      Итерационный цикл
  iterCount  %  счетчик числа итераций
  %                Селекция родителей
  prob = flipud([1:keep]'/sum([1:keep])); % веса хромосом
  odds = [0 cumsum(prob(1:keep))'];   % колесо рулетки
  pick1 = rand(1,M);                  % массив первых родителей
  pick2 = rand(1,M);                  % массив вторых родителей
  % ma и pa содержат индексы хромосом, которые будут скрещиваться
  ic = 1;
  while ic<=M
    for id = 2:keep+1
      if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
        ma(ic) = id-1;
      end
      if pick2(ic)<=odds(id) & pick2(ic)>odds(id-1)
        pa(ic) = id-1;
      end
    end
    ic = ic+1;
  end % while ic
  %    Скрещивание с использованием одноточечного кроссовера
  ix = 1:2:keep;           % индекс первого родителя
  xp = ceil(rand(1,M)*Nt); % точки кроссовера
  r = rand(1,M);           % параметр смешения
  for ic = 1:M
    xy = par(ma(ic),xp(ic))-par(pa(ic),xp(ic));  % ma и pa скрещиваются
    par(keep+ix(ic),:) = par(ma(ic),:);          % первый потомок
    par(keep+ix(ic)+1,:) = par(pa(ic),:);        % второй потомок
    par(keep+ix(ic),xp(ic)) = par(ma(ic),xp(ic))-r(ic).*xy; % 1-й
    par(keep+ix(ic)+1,xp(ic)) = par(pa(ic),xp(ic))+r(ic).*xy; % 2-й
    if xp(ic)<npar % кроссовер, когда последняя переменная не выбрана
       par(keep+ix(ic),:) = [par(keep+ix(ic),1:xp(ic))
       par(keep+ix(ic)+1,xp(ic)+1:npar)];
       par(keep+ix(ic)+1,:) = [par(keep+ix(ic)+1,1:xp(ic))
       par(keep+ix(ic),xp(ic)+1:npar)];
    end % if
  end % ic
  %                       Мутация
  mrow = sort(ceil(rand(1,nmut)*(popsize-1))+1);
  mcol = ceil(rand(1,nmut)*Nt);
  for ii = 1:nmut
      par(mrow(ii),mcol(ii)) = c(mcol(ii))*rand+varlo(mcol(ii));  % мутация
  end
  %   Перевычисление популяции
  how = 'новая поп';
  for j = 1:popsize
    cost(j) = newPoint((par(j,:))',how);
    if stateget('exit')
      return
    end
```

```
  end
  [cost,ind] = sort(cost); % сортировка особей популяции
  par = par(ind,:);        % сортировка точек
  fi = double(cost(1));    % лучшее значение целевой функции
  stateset('fi',fi)
  STATE.Fi(:,STATE.Ni) = fi;
  stateset('fm',mean(cost))  % среднее значение функции в популяции
  H = max(max(abs(par(2:popsize,:)-par(onesp,:)))); % погрешность
  stateset('H',H)
  if stateiter return, end % критерий останова
end % while 1
```

Это программа одного из простейших непрерывных ГА.

Тестирование генетических алгоритмов

Для проверки эффективности методов глобальной оптимизации используются различные многоэкстремальные целевые функции:
синусоидальная функция

$$f(x) = x_1 \sin 4x_1 + 1{,}1x_2 \sin 2x_2,$$

модифицированная функция Растригина

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos 2\pi x_i),$$

функция Швефеля

$$f(x) = 418{,}9829n - \sum_{i=1}^{n} x_i \sin\sqrt{|x_i|},$$

функция Эккли

$$f(x) = 20 + e - 20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right),$$

функция Михалевича

$$f(x) = -\sum_{i=1}^{n} \sin x_i \sin\frac{ix_i^2}{\pi},$$

функция Негневицкого

$$f(x) = (x_1 - x_1^3 - x_2^3)\exp(-x_1^2 - x_2^2) - (1 - x_1)^2 \exp[-x_1^2 - (x_2 + 1)^2].$$

Ниже представлены подпрограммы этих функций в системе MATLAB.

```
function f = fungasin(x)
% Тестовая функция с синусами
% Глобальный минимум находится в точке
% x = [9.038991604817884 8.6681889577870734]' с fx = [-18.554721077382709]'
% 0 <= x <= 10
% Входной параметр: x - двумерный вектор переменных
% Выходной параметр: f - значение функции
f = x(1).*sin(4*x(1)) + 1.1*x(2).*sin(2*x(2));

function f = fungaRas(x)
% Тестовая функция Растригина
% Глобальный минимум находится в точке O(0;0;...;0) с f=0
% -5,12 <= x <= 5,12
% Входной параметр: x - двумерный вектор переменных
% Выходной параметр: f - значение функции
p2 = 2*pi;
```

```
f = 10*size(x,1) + sum(x.^2 - 10*cos(p2.*x),1);

function f = funSchwefel(x)
% Тестовая функция Швефеля
% Глобальный минимум находится в точке с координатами
% [4.2096874619338882e2 4.2096874622355651e2]' с f = 2.5455132345086895e-5
% -50 <= x <= 500
% Входной параметр: x - двумерный вектор переменных
% Выходной параметр: f - значение функции
f = 418.9829*size(x,1) - sum(x.*sin(sqrt(abs(x))),1);

function f = funAckley(x)
% Тестовая функция Эккли
% Глобальный минимум находится в точке О с координатами 0 и с f = 0
% -30 <= x <= 30
% Входной параметр: x - двумерный вектор переменных
% Выходной параметр: f - значение функции
n = size(x,1); r = 1/n; e = exp(1); p2 = 2*pi;
e1 = exp(-0.2*sqrt(r*sum(x.^2,1)));
e2 = exp(r*sum(cos(p2*x),1));
f = 20 + e - 20*e1 - e2;

function f = funMichal(x)
% Тестовая функция Михалевича
% Глобальный минимум находится в точке с координатами
% [1.9256295731179141 1.5521170532724]' с f = -1.8662535885812193
% -pi <= x <= pi
% Входной параметр: x - двумерный вектор переменных
% Выходной параметр: f - значение функции
n = size(x,1); r = (1:n)'/pi;
f = - sum(sin(x).*sin(r.*x.^2),1);

function f = funNegnev(v)
% Тестовая функция Негневицкого
% Глобальный минимум находится в точке с координатами
% [4.2096874619338882e2 4.2096874622355651e2]' с f = 2.5455132345086895e-5
% -3 <= v <= 3
% Входной параметр: v - двумерный вектор переменных
% Выходной параметр: f - значение функции
x = v(1); y = v(2);
f = (x - x^3 - y^3)*exp(-x^2-y^2) - (1 - x)^2*exp(-x^2-(y+1)^2);
```

В лаборатории OPTLAB создана база данных этих тестовых функций для количества переменных $n = 2$ в виде m-функции `bdfunga`.

```
function F = bdfunga(n)
% База данных тестовых функций для генетических алгоритмов
% Входной параметр: n - номер тестовой функции
% Выходной параметр: F - структура данных тестовых функций
F(1).title = 'Функция с синусами'; F(1).short = 'Fsin';
F(1).mfun = @fungasin; F(1).n = 2; F(1).m = 1; F(1).par = {};
F(1).x0 = [0.1;0.1]; F(1).Goal = 1;
F(1).fo = -18.554721077382709;
F(1).xo = [9.038991604817884 8.6681889577870734]';
F(1).a = [0;0]; F(1).b = [10;10];
F(2).title = 'Функция Растригина'; F(2).short = 'FRas';
F(2).mfun = @fungaRas; F(2).n = 2; F(2).m = 1; F(2).par = {};
F(2).x0 = [0.5;0.5]; F(2).Goal = 1; F(2).fo = 0; F(2).xo = [0;0];
F(2).a = [-5.12;-5.12]; F(2).b = [5.12;5.12];
```

```
F(3).title = 'Функция Швефеля'; F(3).short = 'Shw';
F(3).mfun = @funSchwefel; F(3).n = 2; F(3).m = 1; F(3).par = {};
F(3).x0 = [0 0]'; F(3).Goal = 1;
F(3).fo = 2.5455132345086895e-5;
F(3).xo = [4.2096874619338882e2 4.2096874622355651e2]';
F(3).a = [-50;-50]; F(3).b = [500;500];
F(4).title = 'Функция Эккли'; F(4).short = 'Ack';
F(4).mfun = @funAckley; F(4).n = 2; F(4).m = 1; F(4).par = {};
F(4).x0 = [25 25]'; F(4).Goal = 1; F(4).fo = 0; F(4).xo = [0 0]';
F(4).a = [-30;-30]; F(4).b = [30;30];
F(5).title = 'Функция Михалевича'; F(5).short = 'Mic';
F(5).mfun = @funMichal; F(5).n = 2; F(5).m = 1; F(5).par = {};
F(5).x0 = [0.01*pi 0.01*pi]';
F(5).Goal = 1; F(5).fo = -1.8662535885812193;
F(5).xo = [1.9256295731179141 1.5521170532724]';
F(5).a = [0;0]; F(5).b = [pi;pi];
F(6).title = 'Функция Негневицкого'; F(6).short = 'Neg';
F(6).mfun = @funNegnev; F(6).n = 2; F(6).m = 1; F(6).par = {};
F(6).x0 = [2.5 2.5]';
F(6).Goal = 1; F(6).fo = -1.6709859058883743;
F(6).xo = [-0.62286612513846851 -0.82773347143844012]';
F(6).a = [-3;-3]; F(6).b = [3;3];
if nargin F = F(n); end
```

В результате вызова этой m-функции создается структура `F` с полями: `title` – полное название функции, `short` – краткое обозначение функции, `mfun` – ссылка на m-функцию, `n` – количество переменных, `m` – размерность функции, `par` – дополнительные постоянные параметры функции, `x0` – начальная точка поиска, `Goal` – флаг наличия решения задачи оптимизации, `fo` – оптимальное значение целевой функции, `xo` – оптимальное значение вектора переменных, `a` – вектор нижних ограничений переменных, `b` – вектор верхних ограничений переменных.

В табл. 2 представлено количество вычислений значений тестовых функций, выполненное бинарным и непрерывным ГА до вычисления оптимального значения функции с абсолютной погрешностью менее 10^{-2}. Бинарный ГА *GAB* оказался эффективнее непрерывного ГА *GAC* по количеству вычислений.

Таблица 2. Количество вычислений значений целевых функций генетическими алгоритмами

ГА	Функция					
	синусоидальная	Растригина	Швефеля	Эккли	Михалевича	Негневицкого
GAB	285	1592	808	2605	135	1197
GAC	1187	8372	1338	–	137	1351

На рис. 1-3 представлен процесс минимизации многоэкстремальной синусоидальной целевой функции бинарным ГА.

На рис. 1 и 2 на графике линий уровня и трехмерном графике начальная точка обозначена зеленым кругом, конечная – красным ромбом. Лучшие точки поиска соединены. Также показаны все остальные точки поиска, которые сгущаются в окрестностях точек локальных минимумов.

На рис. 3 показана соответствующая рис. 1 и 2 зависимость лучшего значения целевой функции (best value) и среднего по популяции значения функции (mean value) от номера итерации *Ni*.

Приведенные ГА не позволяют решать задачи параметрического синтеза САУ ввиду особенностей этих задач.

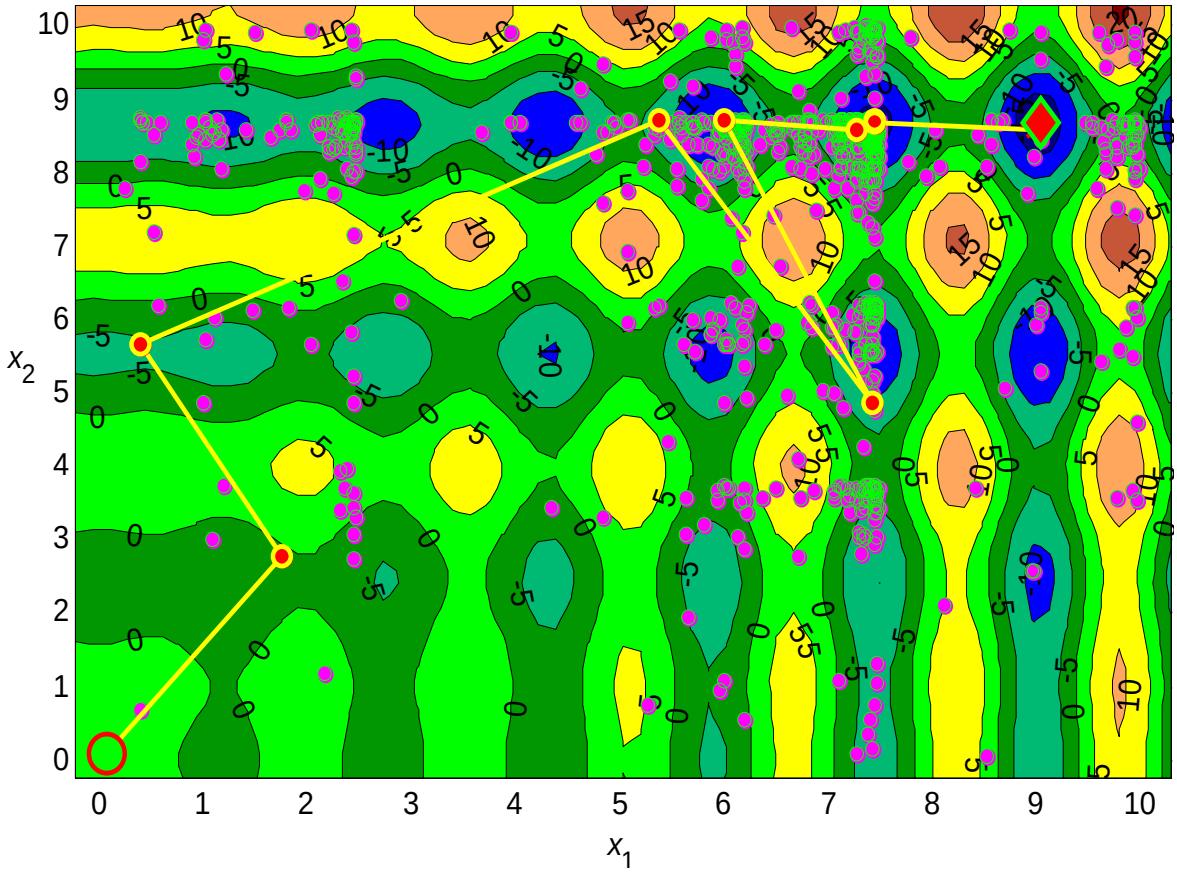

Рис. 1. Минимизация синусоидальной функции бинарным ГА на графике линий уровня

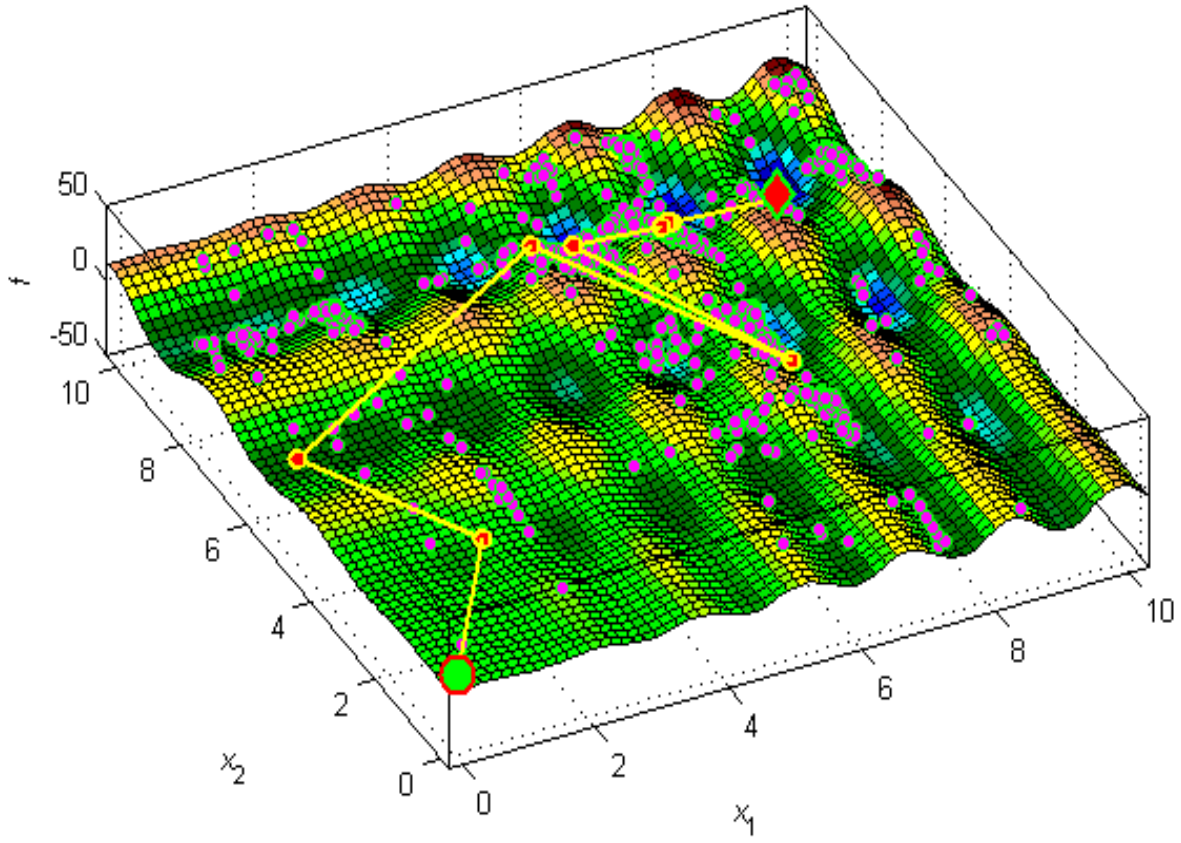

Рис. 2. Минимизация синусоидальной функции бинарным ГА на трехмерном графике

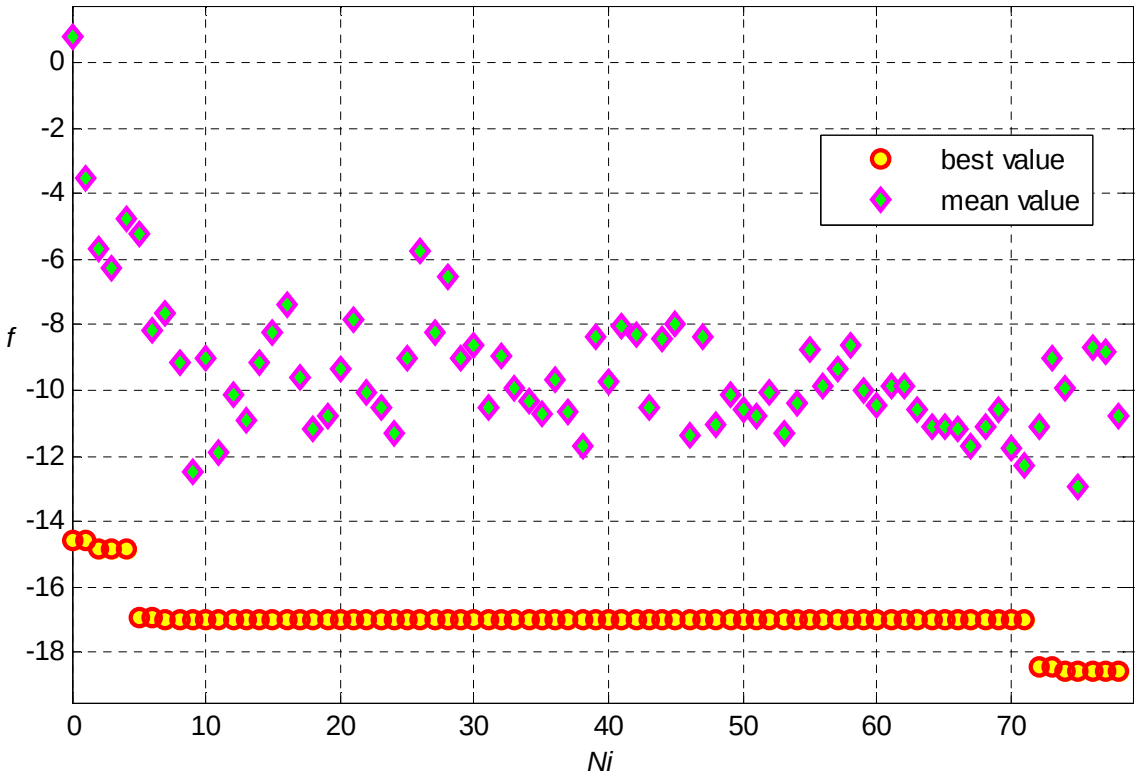

Рис. 3. Зависимость лучшего и среднего значений целевой функции от номера итерации бинарного ГА

Многокритериальность задач синтеза систем управления

Рассмотрим особенности задач синтеза САУ для модификации ГА. Критерии качества систем автоматического управления определим по математическим моделям систем [8, 9], которые можно записать в пространстве состояний или в виде передаточной функции с учетом вектора переменных параметров САУ x:

$$dX/dt = A(x)X + B(x)u\,, \quad y = C(x)X\,, \tag{3}$$

$$dX/dt = f(x, X, u)\,, \tag{4}$$

$$W(x,s) = \beta(x,s)/\alpha(x,s)\,, \tag{5}$$

$$\beta(x,s) = \sum_{i=0}^{m} \beta_i(x) s^{m-i}\,, \quad \alpha(x,s) = \sum_{i=0}^{n} \alpha_i(x) s^{n-i}\,.$$

Для исследования критериев качества выбраны тестовые передаточные функции САУ: второго порядка с одним переменным параметром x и третьего порядка с двумя переменными параметрами x_1, x_2:

$$W(x,s) = \frac{1}{s^2 + xs + 1}\,, \qquad W(x,s) = \frac{1}{s^3 + x_1 s^2 + x_2 s + 1}\,. \tag{6}$$

Приведенные в табл. 3 результаты минимизации частных прямых показателей качества (ППК) – перерегулирования $\sigma(x)$ и времени регулирования $t_c(x)$ или его относительного значения $\tau(x) = t_c(x)/T_f$, где T_f – время наблюдения переходного процесса, для тестовых систем (6) различных порядков n показывают, что ППК противоречивы. Таким образом, задача оптимизации ППК, вычисляемых в общем случае интегрированием систем дифференциальных уравнений (3), (4), является многокритериальной задачей.

Таблица 3. Минимизация частных ППК для тестовых САУ

n	min	x_1^*	x_2^*	σ^*	τ^*	t_c^*, с
2	$\sigma(x)$	1,98	—	0,00	0,233	4,66
	$\tau(x)$	1,38	—	0,05	0,143	2,86
3	$\sigma(x)$	1,92	2,38	0,00	0,216	4,31
	$\tau(x)$	1,45	2,02	0,05	0,161	3,22

Интегральные квадратичные оценки (ИКО) более удобны для синтеза систем. Частные ИКО определяются интегралами квадратов отклонения управляемой величины от ее установившегося значения и производных отклонения:

$$J_0(x) = \int_0^\infty [z(x,t)]^2 dt, \qquad J_k(x) = \int_0^\infty \left[\frac{\partial^k z(x,t)}{\partial t^k} \right]^2 dt .$$

Результаты минимизация частных оценок для тестовых систем (6), приведенные в табл. 4, показывают, что эти критерии также противоречивы. В этой таблице представлены значения ППК: ζ^* – значение размаха колебаний $\zeta(x)$, λ^* – значение показателя затухания колебаний $\lambda(x)$. Эти значения свидетельствуют о значительной колебательности процессов, полученных при минимизации $J_0(x)$. Таким образом, задача минимизации интегральных квадратичных оценок также является многокритериальной задачей.

Таблица 4. Минимизация частных ИКО тестовых САУ

n	k	x_1^*	x_2^*	J_k^*	σ^*	ζ^*	λ^*	t_c^*, с
2	0	1,98	—	1,0	0,16	0,19	0,16	5,29
	1	1,38	—	0,1	0,00	0,00	0,00	14,6
3	0	1,92	2,38	1,5	0,07	0,23	2,15	8,73
	1	1,45	2,02	0,1	0,00	0,00	0,00	12,6

Для решения этой задачи применяются улучшенные интегральные квадратичные оценки:

$$J(x) = \int_0^\infty \sum_{k=0}^{l} w_k [z_t^{(k)}(x,t)]^2 dt,$$

$$I(x) = \int_0^\infty \left[\sum_{k=0}^{l} \tau_k z_t^{(k)}(x,t) \right]^2 dt . \qquad (7)$$

Первая из этих оценок является сверткой частных критериев. Путем ее преобразования перейдем к оценке (7), имеющей тот же минимум, что показано на рис. 4 для $n=2$, $w_1=1$, $\tau_1=1$. Оценка (7) предпочтительнее, поскольку связана с экстремалью y_e для оптимального переходного процесса y, что представлено на рис. 5.

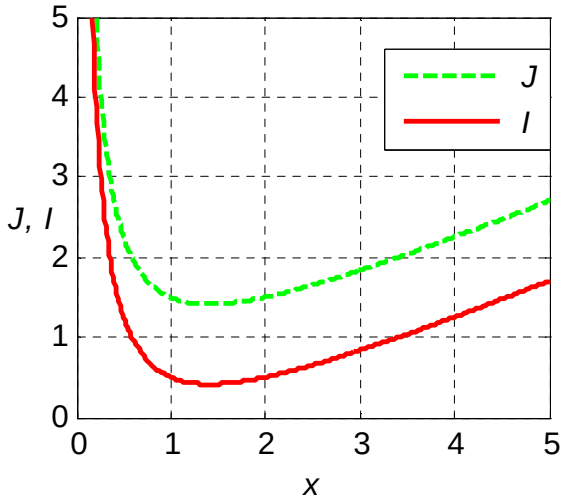

Рис. 4. Графики различных улучшенных ИКО для системы второго порядка

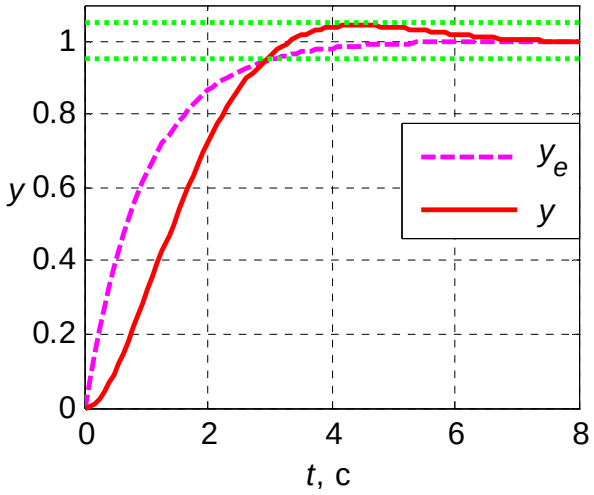

Рис. 5. Графики экстремали и оптимального переходного процесса

Для вычисления оценки (7) применяются методы Каца и Острема [8]. Однако непосредственная минимизация функции $f(x) = I(x)$, которая определяется этими методами, может привести к выходу из области устойчивости. Рис. 6 и 7 демонстрируют несколько шагов процесса минимизации ИКО САУ (6) методами адаптации шага и Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ) с выходом из области устойчивости из-за большого начального шага. Начальная точка показана кругом, конечная – ромбом, отмеченная звездочкой точка минимума ИКО оказалась недостижимой.

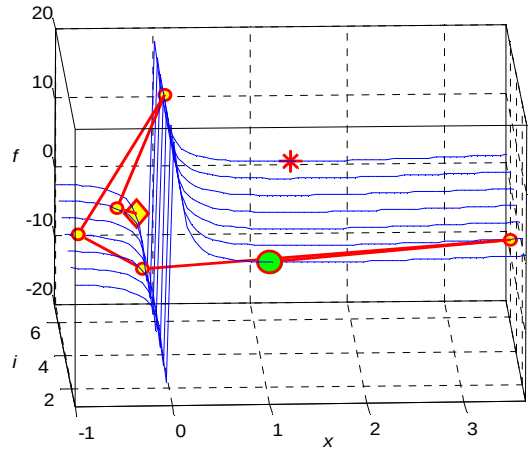

Рис. 6. Минимизация ИКО САУ второго порядка

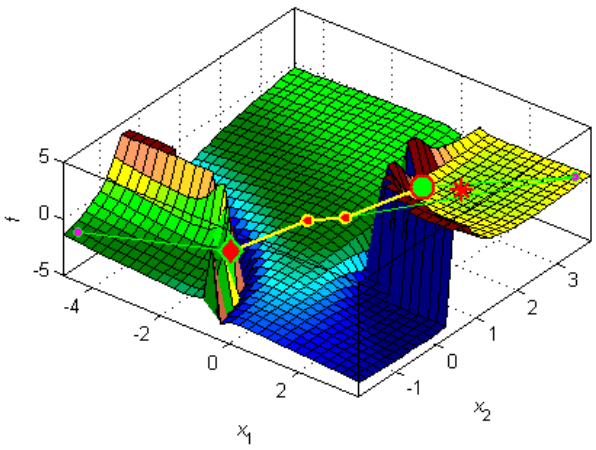

Рис. 7. Минимизация ИКО САУ третьего порядка

Эти результаты позволяют сделать вывод, что частные критерии качества САУ противоречивы, а процесс синтеза может выйти из области устойчивости.

Векторные критерии качества

Для перехода в область устойчивости системы автоматического управления и удержания в ней процесса параметрического синтеза развиты векторные критерии качества [8-12].

В задаче параметрического синтеза учитываются условия технической реализуемости системы управления и условия устойчивости:

$$a_i \le x_i \le b_i, \quad i=\overline{1,\,p}, \tag{8}$$

$$\alpha_i(x)>0, \quad i=\overline{0,n}; \qquad \rho_k(x)>0, \quad k=\overline{2,n-1}, \tag{9}$$

которые заключаются в положительности коэффициентов характеристического многочлена модели (5) и элементов первого столбца таблицы Рауса $\rho_k(x)$. На основании этих условий формируется допустимая область D путем пересечения и вычитания множеств:

$$G_1=\{x \mid a_i \le x_i \le b_i, i=\overline{1,\,p}\}; \tag{10}$$

$$G_2=\{x \mid \alpha_i(x)>0, i=\overline{0,n}\}; \tag{11}$$

$$G_k=\{x \mid \rho_{k-1}(x)>0\}, \quad k=\overline{3,n}; \tag{12}$$

$$D_1=G_1; \qquad D_k=D_{k-1}\cap G_k, \quad k=\overline{2,n}; \tag{13}$$

$$H_0=R^p \setminus D_1; \qquad H_k=D_k \setminus D_{k+1}, \quad k=\overline{1,n-1}; \qquad D=D_n. \tag{14}$$

Условиям технической реализуемости и устойчивости (8), (9) соответствуют скалярные штрафные функции

$$P(x)=\sum_{i=1}^{p}\left[(a_i-x_i)_+ +(x_i-b_i)_+\right],$$

$$S(x)=\sum_{k=1}^{n}\left[-\alpha_k(x)\right]_+,$$

для формирования которых применяется операция срезки $v_+=\max\{v,0\}$.

Для удержания процесса синтеза в области устойчивости использована векторная штрафная функция, первая проекция которой $F_1(x)$ – функция уровня соответствует количеству выполненных ограничений, а вторая проекция $F_2(x)$ – функция штрафа представляет штраф нарушенного ограничения:

$$F_S(x)=\begin{cases}(0;P(x)), & x\in H_0,\\ (1;S(x)), & x\in H_1,\\ (k;-\rho_k(x)), & x\in H_k, \quad k=\overline{2,n-1}.\end{cases} \tag{15}$$

Переход в допустимую область системы управления сведен к оптимизации векторной функции на основании сравнения двух ее произвольных значений $U=(U_1;U_2)$ и $V=(V_1;V_2)$ операцией «лучше»:

$$U<V=\begin{cases}1, & (U_1>V_1)\vee[(U_1=V_1)\wedge(U_2<V_2)],\\ 0, & (U_1<V_1)\vee[(U_1=V_1)\wedge(U_2\ge V_2)].\end{cases} \tag{16}$$

Задачу синтеза САУ по ППК можно сформулировать в виде задачи условной оптимизации:

$$x\in D, \quad \sigma(x)\le\sigma_m, \quad \zeta(x)\le\zeta_m, \quad \min\tau(x). \tag{17}$$

Однако такая постановка не учитывает приоритет критериев. Воспользуемся пошаговым подходом, справедливым для перехода в допустимую область параметров САУ [8-11]. Для этого расширим области (10)–(14) областями для учета неравенств (17):

$$G_{n+1}=\{x \mid \sigma(x)\le\sigma_m\}, \quad G_{n+2}=\{x \mid \zeta(x)\le\zeta_m\};$$

$$D_k=D_{k-1}\cap G_k, \quad k=\overline{n+1,n+2},$$

$$H_k=D_k \setminus D_{k+1}, \quad k=\overline{n,n+1}; \qquad H_{n+2}=D_{n+2}.$$

Для синтеза САУ по ППК на основании функции (15) сформирована векторная целевая функция

$$F(x)=\begin{cases} F_S(x), & x\notin D; \\ (n;\sigma(x)-\sigma_m), & x\in H_n; \\ (n+1;\zeta(x)-\zeta_m), & x\in H_{n+1}; \\ (n+2;\tau(x)), & x\in H_{n+2}. \end{cases} \quad (18)$$

Аналогично формируется векторная функция для учета в задаче (17) ограничения $\lambda(x)\le\lambda_m$.

Для синтеза систем по улучшенным ИКО предложена векторная целевая функция

$$F_I(x)=\begin{cases} F_S(x), & x\notin D; \\ (n;I(x)), & x\in D. \end{cases} \quad (19)$$

Разработан класс векторных функций в лаборатории OPTLAB. Конструктор этого класса представлен m-функцией `stepfun`.

```
function F = stepfun(V)
% Конструктор класса stepfun векторной целевой функции пошагового подхода
% STEP FUNction - пошаговая функция
% Входной параметр: V - значение вектор-функции
% Выходной параметр: F - объект класса stepfun
if nargin == 0 % при вызове конструктора нет входных аргументов
   F.c = [];   % создается структура объекта
   F = class(F,'stepfun'); % создается объект по заданной структуре
elseif isa(V,'stepfun') % проверка входного параметра как объекта класса
   F = V;   % копирование входного параметра
else        % входной параметр - вектор
   F.c = V; % преобразование входного вектора
   F = class(F,'stepfun');  % создание объекта по заданной структуре
end
```

В этом классе операция «лучше» (16) представлена m-функцией `lt`, которая переопределяет операцию сравнения «меньше» < для скалярных целевых функций.

```
function B = lt(U,V)
% Операция "лучше" пошагового подхода
% Входные параметры: U, V - значения векторной функции
% Выходной параметр: B - булевский результат сравнения:
%   1 - U лучше V, 0 - U не лучше V
U = stepfun(U); V = stepfun(V); B = 0;
if U.c(1) > V.c(1)
   B = 1; return
elseif U.c(1) < V.c(1)
   return
elseif U.c(2) < V.c(2)
   B = 1;
end
```

На рис. 8, 9 на графиках проекций векторной функции (19) показан процесс ее оптимизации методом БФГШ для тестовой системы третьего порядка (6). Одна из точек поиска вышла из допустимой области, но процесс оптимизации продолжился в этой области, и была найдена точка минимума оценки.

Этот и другие вычислительные эксперименты позволяют сделать вывод, что применение векторной целевой функции позволяет удержать процесс синтеза системы управления в допустимой области [8-12].

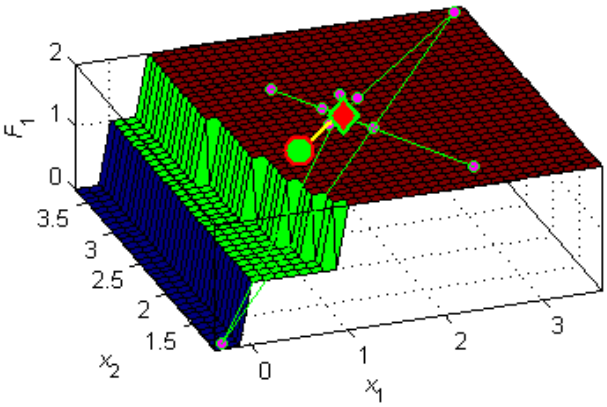

Рис. 8. Минимизация ИКО на графике функции уровня

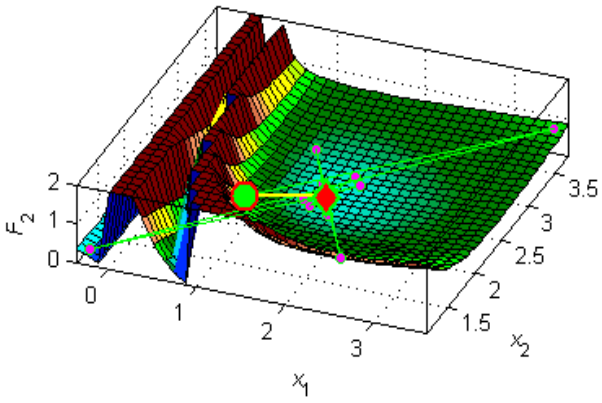

Рис. 9. Минимизация ИКО на графике функции штрафа

Векторные целевые функции (18), (19) являются разрывными и многоэкстремальными, поэтому для повышения надежности синтеза интеллектуальных САУ применим генетические алгоритмы [3-7].

Модификация генетических алгоритмов

Модификация генетических алгоритмов выполнена по основным принципам этих алгоритмов [3-7]. Каждая особь популяции представлена хромосомой $\xi = (\xi_1, \xi_2, \ldots, \xi_p)$ длины $M = mp$, состоящей из бинарных генов $\xi_i = (\xi_{i1}, \xi_{i2}, \ldots, \xi_{im})$ длины $m = 32$, $i = \overline{1, p}$, $\xi_{ij} \in \{0; 1\}$, и определяющей вектор переменных $x = (x_1, x_2, \ldots, x_p)$.

Хромосомы $\Pi = (\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(N)})$, точки $P = (x^{(1)}, x^{(2)}, \ldots, x^{(N)})$, значения векторной целевой функции (18) или (19) $\Phi = (F^{(1)}, F^{(2)}, \ldots, F^{(N)})$ образуют популяцию размера N. Начальная популяции $\Pi^{(0)}$ формируется случайно, для следующих популяций применяются генетические операторы: селекция, кроссовер, мутация. На k-той итерации формируется популяция особей, которой соответствуют массивы хромосом $\Pi^{(k)}$, точек $P^{(k)}$ и значений целевой функции $\Phi^{(k)}$. Применяется ранговая селекция с принципом рулетки (*Roulette*): популяция сортируется сравнением значений векторной целевой функции (*Sort*); родительской особи сопоставляется сектор колеса рулетки, пропорциональный ее рангу; по принципу рулетки выбираются родители для скрещивания. Кроссовером (*Crossover*) с последующей мутацией (*Mutate*) формируется новая популяция.

Основное отличие модифицированного бинарного ГА *GAB* от ГА для оптимизации скалярных целевых функций состоит в использовании значений векторной целевой функции для точек популяции и в сортировке этих значений на шагах 6 и 12 с применением операции сравнения (16). С аналогичным отличием разработан модифицированный непрерывный генетический алгоритм *GAC*, в котором исходной является не популяция двоичных хромосом, а популяция точек пространства переменных параметров, что исключает декодирование.

Процедура сортировки значений скалярных целевых функций переопределяется в классе для векторных целевых функций аналогичной процедурой в виде m-функции пузырьковой сортировки `sort`.

```
function [F,I] = sort(F)
% Пузырьковая сортировка объектов класса stepfun
```

```
% в порядке возрастания (ухудшения)
% Входной параметр: F - массив объектов класса stepfun
% Выходные параметры:
%   G - упорядоченный массив объектов класса stepfun
%   I - массив индексов объектов класса stepfun
[m n] = size(F); I = 1:n; G = []; T = [];
% Сортировка значений в порядке ухудшения
for i = 1:n
   for j = n:-1:i+1
      if F(j) < F(j-1)
         k = j - 1; l = I(k); f = F(k);
         I(k) = I(j); F(k) = F(j);
         I(j) = l; F(j) = f;
      end
   end
end
```

Итак, в отличие от большинства ГА, в модифицированных векторных ГА используются значения векторной целевой функции и разработанная для их сортировки процедура. Рассмотрим применение ГА к минимизации ИКО и оптимизации ППК.

Синтез систем модифицированными генетическими алгоритмами

Синтез систем генетическими алгоритмами проводился на тестовых примерах систем управления.

Для минимизации улучшенных интегральных квадратичных оценок применялись тестовые передаточные функции порядков от 3 до 105 с переменными коэффициентами знаменателя:

$$W(x,s) = 1/\alpha(x,s), \qquad \alpha(x,s) = s^n + \sum_{i=1}^{n-1} \alpha_i(x) s^{n-i} + 1.$$

Наряду с простыми ГА – бинарным *GAB* и непрерывным *GAC* разработаны комбинированные – бинарный и непрерывный ГА с методами Хука-Дживса и Нелдера-Мида на завершающем этапе поиска.

Результаты вычислительных экспериментов позволяют сделать вывод, что для синтеза систем управления комбинированный бинарный ГА с методом Нелдера-Мида *GABNM* наиболее эффективен.

На рис. 10 и 11 представлены графики проекций векторной функции (17), все точки поиска и траектория лучших точек минимизации улучшенной ИКО бинарным ГА для тестовой передаточной функции третьего порядка на графиках $F_1(x)$ и $F_2(x)$ при $n = 3$, $p = 2$. На рис. 12, 13 представлена траектория лучших точек минимизации ИКО для тестовой передаточной функции при $n = 10$, $p = 2$.

Оптимизация ППК проводилась для тестовой модели системы управления третьего порядка (6), $n = 3$. Результаты трех экспериментов оптимизации функции (18) представлены в табл. 5. В ней для номера эксперимента N_e даны предельные значения показателей качества, координаты конечной точки (x_1^*, x_2^*), значения ППК в конечной точке: σ^*, ζ^*, λ^*, t_c^*. На рис. 14, 15 представлены оптимальные точки и им соответствующие переходные процессы. Для второго эксперимента (см. табл. 5, $N_e = 2$) на рис. 16, 17 даны графики проекций векторной функции (18) и траектории поиска, которые подтверждают, что оптимальная точка достигнута.

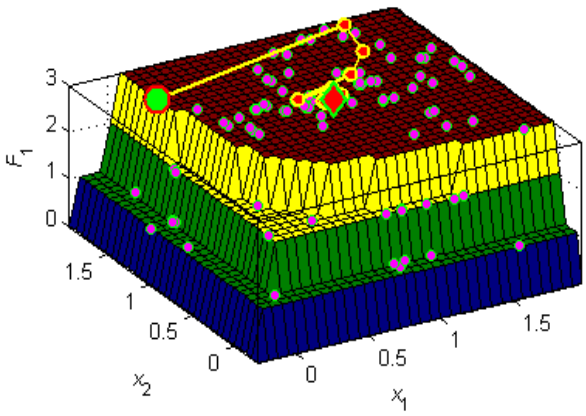

Рис. 10. Функция уровня для системы третьего порядка

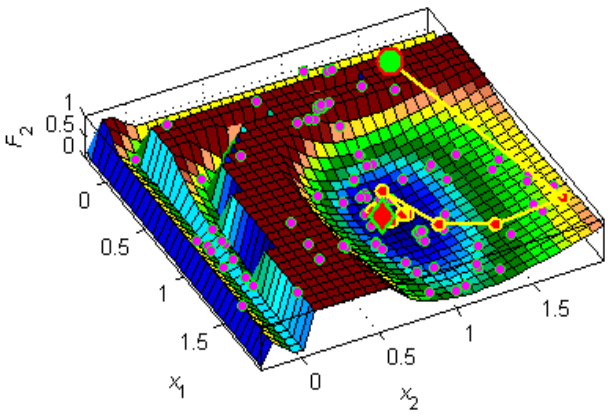

Рис. 11. Функция штрафа для системы третьего порядка

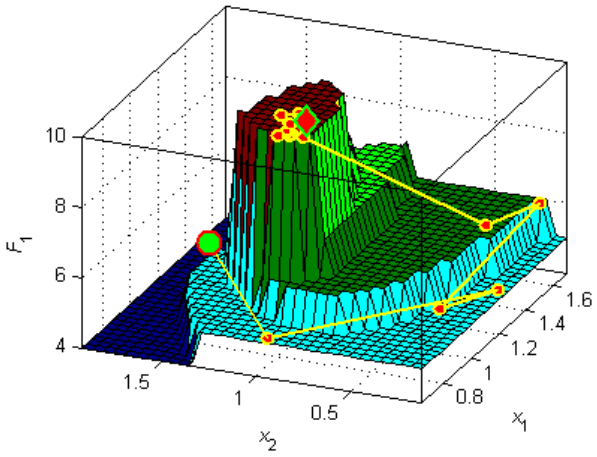

Рис. 12. Функция уровня для системы десятого порядка

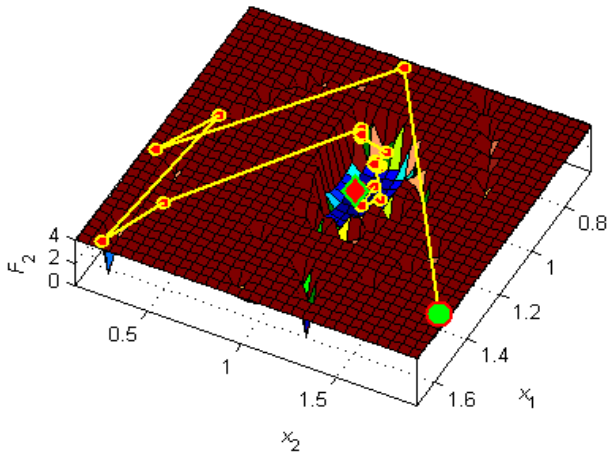

Рис. 13. Функция штрафа для системы десятого порядка

Таблица 5. Оптимизация прямых показателей качества

Параметры	Значения параметров		
	$N_e = 1$	$N_e = 2$	$N_e = 3$
σ_m	0,05	0,05	0,05
ζ_m	0,065	0,065	0
λ_m	—	0,3	—
x_1^*	1,504	1,885	1,942
x_2^*	2,126	2,079	2,383
σ^*	0,015	0,050	0
ζ^*	0,065	0,065	0
λ^*	3,334	0,300	0
t_c^*, с	3,377	3,552	4,316
Маркер	∇		*

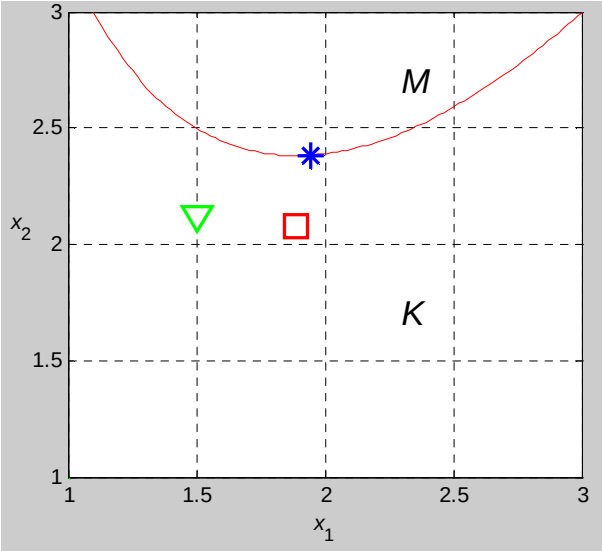

Рис. 14. Оптимальные точки

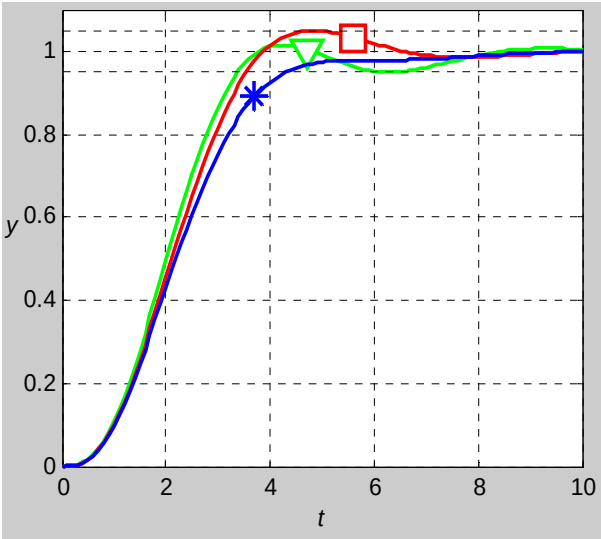

Рис. 15. Оптимальные процессы

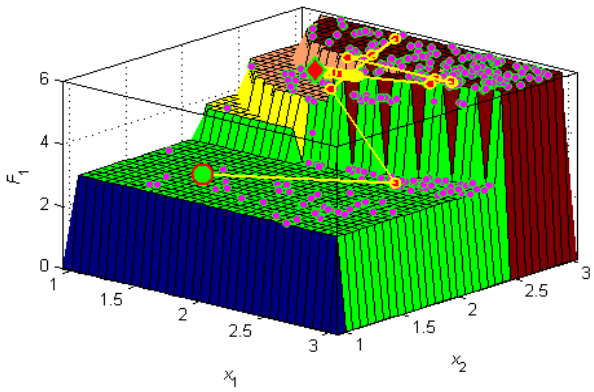

Рис. 16. Оптимизация ППК на графике функции уровня

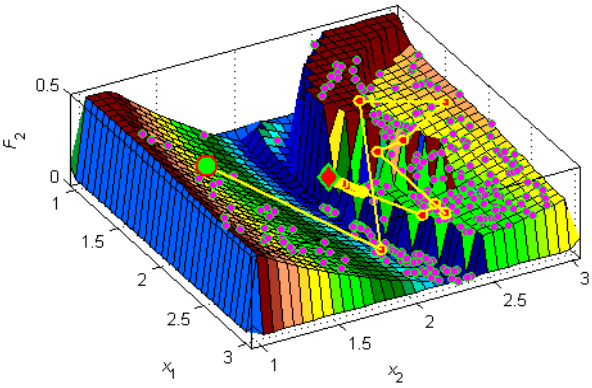

Рис. 17. Оптимизация ППК на графике функции штрафа

Проведенные эксперименты по оптимизации ППК САУ с передаточными функциями порядков от трех до семи подтвердили эффективность работы модифицированного генетического алгоритма *GABNM* .

Синтез оптимальных регуляторов

Линейные пропорциональный (П), интегральный (И) и дифференциальный (Д) законы управления формируют управляющее действие u регуляторов на объект управления по ошибке ε:

$$u_P = K_P\varepsilon, \quad u_I = \frac{1}{T_I}\int \varepsilon dt, \quad u_D = \tau_D \frac{d\varepsilon}{dt}.$$

Передаточные функции линейных П, И и реализуемого Д регуляторов имеют вид:

$$W_P = K_P, \quad W_I(s) = \frac{1}{T_I s}, \quad W_D(s) = \frac{K_D T_D s}{T_D s + 1}.$$

Обозначим параметры этих регуляторов:

$$\lambda_I = 1/T_I, \quad \lambda_D = 1/T_D, \quad K_D = 10.$$

Тогда их уравнения примут вид:

$$u_P = K_P\varepsilon; \qquad \frac{du_I}{dt} = \lambda_I\varepsilon; \qquad \frac{dv_D}{dt} = -\lambda_D(v_D + 10\varepsilon), \quad u_D = v_D + 10\varepsilon.$$

В результате синтеза САУ переменные параметры регуляторов должны принадлежать заданному интервалу: K_P, λ_I, $\lambda_D \in [0;100]$.

Нечеткий ПИ регулятор на рис. 18 использует систему нечеткого вывода (СНВ), которая показана на рис. 19 и включает базу правил, блоки фаззификации, нечеткого вывода и дефаззификации. На рис. 20 показаны симметричные треугольные и трапецеидальные функции принадлежности входной ξ и выходной η переменных СНВ: $\xi_n = -\xi_p$, $\eta_n = -\eta_p$.

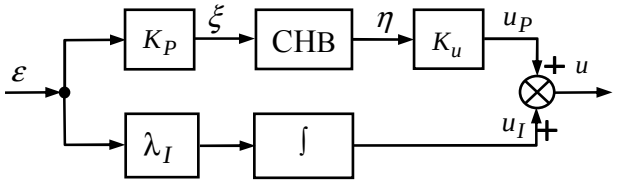

Рис. 18. Схема нечеткого ПИ регулятора

Рис. 19. Система нечеткого вывода

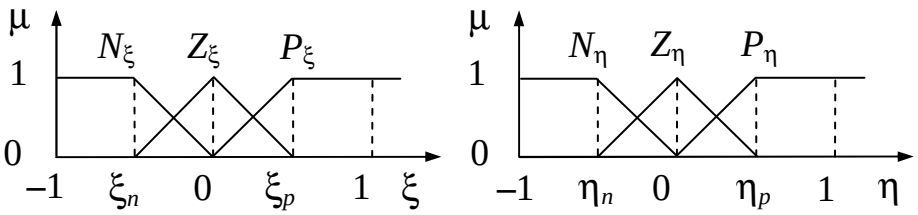

Рис. 20. Функции принадлежности

Простейшая база правил имеет вид:

1) $\xi = N_\xi \Rightarrow \eta = N_\eta$; 2) $\xi = Z_\xi \Rightarrow \eta = Z_\eta$; 3) $\xi = P_\xi \Rightarrow \eta = P_\eta$.

Управляющее действие u нечеткого ПИ регулятора формируется выражениями:

$$u = u_P + u_I,\ u_P = f_F(\xi_p, \eta_p, K_P\varepsilon)K_u,\ du_I/dt = \lambda_I\varepsilon.$$

Вектор переменных параметров нечеткого ПИ регулятора включает пять параметров:

$$x = (K_P, \lambda_I, K_u, \xi_p, \eta_p); \qquad K_P, \lambda_I, K_u \in [0;100]; \qquad \xi_p, \eta_p \in [0;1].$$

Таким образом, модель САУ с нечетким ПИ регулятором является нелинейной моделью вида (4). Для синтеза такой САУ целесообразно оптимизировать целевые функции (18) и (19) предложенными модифицированными генетическими алгоритмами.

Выводы

Результаты проведенных исследований позволяют сформулировать следующие выводы.

1. Рассмотрены бинарный и непрерывный генетические алгоритмы. Путем минимизации многоэкстремальных целевых функций показана большая эффективность

бинарного генетического алгоритма.

2. Проведен анализ многокритериальности прямых показателей качества и интегральных квадратичных оценок систем автоматического управления, который показал противоречивость этих частных критериев качества. Сформированы многокритериальные улучшенные оценки качества. Показано, что применение методов оптимизации для синтеза систем может вывести процесс оптимизации из области устойчивости.

3. Задачи многокритериального параметрического синтеза систем управления сведены к задачам оптимизации векторных целевых функций, решение которых позволяет удержать процесс синтеза систем в допустимой области.

4. Для оптимизации векторных целевых функций систем автоматического управления модифицированы бинарный и непрерывный генетические алгоритмы.

5. Показана эффективность применения модифицированных генетических алгоритмов для синтеза систем управления путем оптимизации векторных целевых функций.

6. Рассмотрение задач синтеза линейных и нечетких ПИД регуляторов показало, что в задаче синтеза нечеткого регулятора определяется вектор переменных параметров большей размерности, а в модели системы управления вместо линейных уравнений применяются нелинейные уравнения с использованием системы нечеткого вывода.

Литература

1. *Бесекерский В. А., Попов Е. П.* Теория систем автоматического управления. СПб.: Профессия. 2004. 752 с.
2. *Гостев В. И.* Синтез нечетких регуляторов систем автоматического управления. К.: Радіоаматор. 2005. 708 с.
3. *Goldberg D. E.* Genetic Algorithms in Search Optimizations and Machine Learning. Addison: Wesly. 1989.
4. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / *Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергеев.* Харьков: Основа. 1997. 112 с.
5. *Панченко Т. В.* Генетические алгоритмы. Учебное пособие. Астрахань: Издательский дом «Астраханский университет». 2007. 88 с.
6. *Weise T.* Global Optimization Algorithms Theory and Application. 2008. 818 p.
7. *Alireza M.* Genetic algorithms. Tehran: Naghoos. 2008. 144 p.
8. *Северин В. П.* Параметрический синтез систем автоматического управления методами векторной оптимизации // Техническая электродинамика. Силовая электроника и энергоэффективность. 2008. Ч. 4. С. 47-52.
9. *Северин В. П.* Векторная оптимизация систем автоматического управления генетическими алгоритмами // Техническая электродинамика. Силовая электроника и энергоэффективность. 2009. Ч. 5. С. 80–85.
10. *Severin V. P.* Application of Genetic Algorithms to Vector Optimization of the Automatic Control Systems // Intelligent Information and Engineering Systems. Information Science and Computing. Information Technologies and Knowledge. 2009. N. 13. Vol. 3. P. 90–97.
11. *Северин В. П.* Методы анализа и синтеза систем автоматического управления энергоблоков атомных электростанций // Проблемы обеспечения безопасности информационных и управляющих систем АЭС. Одесса: Астропринт. 2010. С. 137-152.
12. *Джафари Хенджани Сейед Моджтаба.* Многокритериальный синтез интеллектуальных систем управления энергоблоков АЭС генетическими алгоритмами: Автореф. дис... канд. техн. наук: 05.13.07. НТУ «ХПИ». Харьков. 2010. 20 с.

13. *Потемкин В. Г., Рудаков П. И.* Система MATLAB 5 для студентов. М.: ДИАЛОГ-МИФИ, 1999. 448 с.
14. *Потемкин В. Г.* Система инженерных и научных расчетов MATLAB 5.x. В 2-х томах. М.: ДИАЛОГ-МИФИ, 1999. Том 1, 366 с. Том 2, 304 с.
15. *Дьяконов В.* MATLAB 6: учебный курс. СПб.: Питер, 2001. 592 с.
16. *Мэтьюз Д. Г., Финк К. Д.* Численные методы. Использование MATLAB, 3-е издание. Пер. с англ. М.: Издательский дом «Вильямс», 2001. 720 с.
17. *Медведев В. С., Потемкин В. Г.* Control System Toolbox. MATLAB 5 для студентов. М.: ДИАЛОГ-МИФИ, 1999. 287 с.
18. *Северин В. П.* Лаборатория моделей и методов оптимизации систем автоматического управления // Труды III Всероссийской научной конференции «Проектирование научных и инженерных приложений в среде MATLAB». Санкт-Петербург: СПбГУ. 2007. С. 1189–1200.
19. *Северин В. П.* Структура лаборатории методов оптимизации OPTLAB в системе MATLAB // Труды IV Всероссийской научной конференции «Проектирование научных и инженерных приложений в среде MATLAB». Астрахань. 2009. Издательский дом «Астраханский университет». С. 235–267
20. *Никулина Е. Н., Северин В. П.* Программирование и изучение методов оптимизации в компьютерной математической системе MATLAB // Труды IV Всероссийской научной конференции «Проектирование инженерных и научных приложений в среде MATLAB». Астрахань: Издательский дом «Астраханский университет». 2009. С. 612–634.
21 *Джафари Хенджани Сейед Моджтаба, Северин В. П.* Многокритериальный параметрический синтез систем автоматического управления минимизацией интегральных квадратичных оценок в среде MATLAB // Труды IV Всероссийской научной конференции «Проектирование инженерных и научных приложений в среде MATLAB». Астрахань: Издательский дом «Астраханский университет». 2009. С. 444–456.