Optimización práctica usando Algoritmo Genético

Práctica en MATLAB

Prof. D.Sc. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

7 de diciembre de 2025

Agenda

La Función 'ga' de MATLAB

Global Optimization Toolbox

MATLAB, a través de su Global Optimization Toolbox, proporciona una función potente y flexible llamada ga para resolver problemas de optimización utilizando algoritmos genéticos.

Sintaxis General

Su diseño es altamente flexible, permitiendo abordar una amplia gama de problemas mediante una sintaxis que se adapta a la complejidad del problema, incorporando restricciones, límites y opciones personalizadas.

Agenda

Sintaxis ga()

```
[x,f]=ga(fun)
```

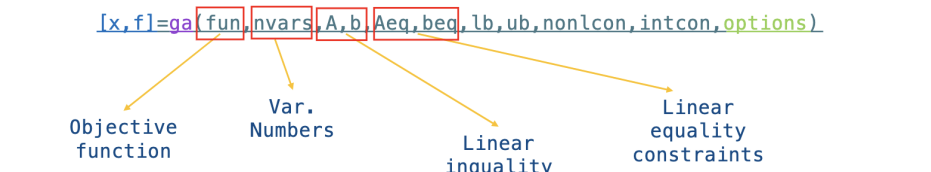


```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

x: vector of solutions

f: fitness (function value at x)

Sintaxis ga()


[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
Objective function
Var. Numbers
Linear inquality constraints
Linear equality constraints
Example: fun = @(x)(x-[4,2]).^2
A*x <= b,
$x_1 + 2x_2 \le 10$
$3x_1 + 4x_2 \le 20$
$5x_1 + 6x_2 \le 30,$
A = [1,2;3,4;5,6];
b = [10;20;30];
$x_1 + 2x_2 + 3x_3 = 10$
$2x_1 + 4x_2 + x_3 = 20,$
Aeq = [1,2,3;2,4,1];
beq = [10;20];

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

Lower and upper bounds

lb represents the lower bounds element-wis, **ub** represents the upper bounds element-wise in lb ≤ x ≤ ub.

Nonlinear constraints

nonlcon is a function that accepts a vector or array x and returns two arrays, c(x) and ceq(x)

```
x = ga(@myfun,4,A,b,Aeq,beq,lb,ub,@mycon)
```

where mycon is a MATLAB® function such as

```
function [c,ceq] = mycon(x)
c = ...     % Compute nonlinear inequalities at x.
ceq = ...   % Compute nonlinear equalities at x.
```

Indices of integer variables

Example: To specify that the even entries in x are integer-valued, set intcon to 2:2:nvars

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

`ConstraintTolerance`	Determines the feasibility with respect to nonlinear constraints. Type: Double, no negative numbers.
`CreationFcn`	Function that creates the initial population. Specify as a name of a built-in creation function or a function handle
`CrossoverFcn`	Function that the algorithm uses to create crossover children. Specify as a name of a built-in crossover function or a function handle.

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

`CrossoverFraction`	The fraction of the population at the next generation, not including elite children, that the crossover function creates. Default: 0.8
`EliteCount`	Positive integer specifying how many individuals in the current generation are guaranteed to survive to the next generation.
`FitnessLimit`	If the fitness function attains the value of `FitnessLimit`, the algorithm halts.

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

FunctionTolerance	The algorithm stops if the average relative change in the best fitness function value over MaxStallGenerations generations is less than or equal to FunctionTolerance.
MaxStallGenerations	The algorithm stops if the average relative change in the best fitness function value over MaxStallGenerations generations is less than or equal to FunctionTolerance.

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

`HybridFcn`	Function that continues the optimization after GA terminates. Specify as a name or a function handle.
`MaxGenerations`	Maximum number of iterations before the algorithm halts.
`MaxStallGenerations`	The algorithm stops if the average relative change in the best fitness function value over `MaxStallGenerations` generations is less than or equal to `FunctionTolerance`.

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

`MutationFcn`	Function that produces mutation children. Specify as a name of a built-in mutation function or a function handle.
`PopulationSize`	Size of the population.
`SelectionFcn`	Function that selects parents of crossover and mutation children. Specify as a name of a built-in selection function or a function handle.

Sintaxis ga()

```
[x,f]=ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

`UseParallel`	Compute fitness and nonlinear constraint functions in parallel.

```
options =
optimoptions("ga","ConstraintTolerance",0.001,"CrossoverFcn","crossov
erheuristic","UseParallel",true)
```

Sintaxis y Argumentos de la Función ga

Argumentos de Entrada (Inputs)

La estructura de la llamada a la función ga se expande para incorporar diferentes tipos de restricciones y configuraciones:

- `ga(fun, nvars)`: La forma más simple.
- `..., A, b`: Añade restricciones de desigualdad lineal ($A \cdot x \leq b$).
- `..., Aeq, beq`: Añade restricciones de igualdad lineal ($A_{eq} \cdot x = b_{eq}$).
- `..., lb, ub`: Define límites (cotas) inferiores y superiores para las variables.
- `..., nonlcon`: Añade restricciones no lineales.
- `..., intcon`: Especifica variables enteras.
- `..., options`: Permite personalizar el comportamiento del algoritmo.

Sintaxis y Argumentos de la Función ga

Argumentos de Salida (Outputs)

Además de la solución `x`, la función puede devolver:

- `fval`: El valor de la función objetivo en la solución `x`.
- `exitflag`: Un código que indica por qué terminó el algoritmo.
- `output`: Una estructura con estadísticas detalladas del proceso.
- `population`: La población final completa.
- `scores`: Las puntuaciones de aptitud de la población final.

Agenda

Minimización sin Restricciones

Uso Básico

El uso más simple de ga es encontrar el mínimo de una función sin ninguna restricción. La sintaxis es:

```
x = ga(@nombre_funcion, nvars)
```

- `@nombre_funcion`: Un manejador a la función objetivo.
- `nvars`: El número de variables de la función.

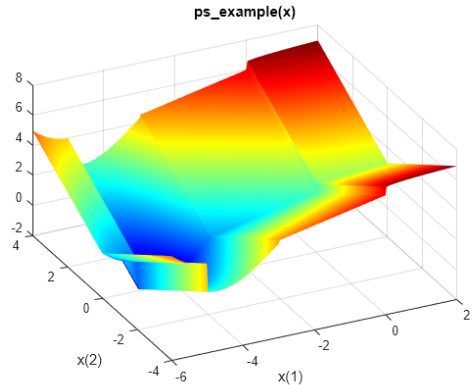


Figura 1: Visualización de la función objetivo `ps_example(x)` a minimizar.

Minimización sin Restricciones

Ejemplo

```
% Para reproducibilidad
rng default

% Llamada a la funcion ga
x = ga(@ps_example, 2)

```

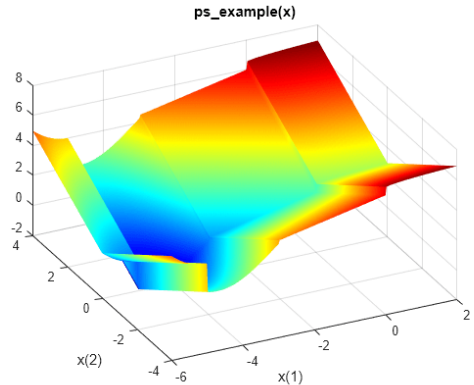


Figura 2: Visualización de la función objetivo `ps_example(x)` a minimizar.

Agenda

Minimización con Restricciones Lineales

Sintaxis

La función ga puede manejar restricciones lineales de desigualdad ($A \cdot x \leq b$) y de igualdad ($A_{eq} \cdot x = b_{eq}$).

Ejemplo

Minimizar `ps_example(x)` sujeto a:

- $-x_1 - x_2 \leq -1$ (desigualdad)
- $-x_1 + x_2 = 5$ (igualdad)

Minimización con Restricciones Lineales

Implementación en MATLAB:

```
% Restriccion de desigualdad: [-1, -1] * [x1; x2] <= -1
A = [-1, -1];
b = -1;

% Restriccion de igualdad: [-1, 1] * [x1; x2] = 5
Aeq = [-1, 1];
beq = 5;

% Llamada a la funcion ga
rng default
x = ga(@ps_example, 2, A, b, Aeq, beq);

```

Agenda

Límites y Restricciones No Lineales

Límites (Cotas) para las Variables

Se pueden definir límites inferiores (`lb`) y superiores (`ub`) para cada variable de decisión.

```
% Definir limites: 1 <= x1 <= 6, -3 <= x2 <= 8
lb = [1, -3];
ub = [6, 8];

% Llamada a ga incluyendo los limites
x = ga(fun, 2, A, b, Aeq, beq, lb, ub);

```

Límites y Restricciones No Lineales

Restricciones No Lineales

Para restricciones complejas no lineales, se debe crear una función aparte.

```
% Funcion que define las restricciones
function [c, ceq] = ellipsecons(x)
    % Desigualdad: c(x) <= 0
    c = 2*x(1)^2 + x(2)^2 - 3;
    % Igualdad: ceq(x) = 0
    ceq = (x(1)+1)^2 - (x(2)/2)^4;
end

% Llamada a ga con el manejador @ellipsecons
nonlcon = @ellipsecons;
x = ga(fun, 2, [],[],[],[],[],[], nonlcon);

```

Ejemplos

Optimización de la función $f(x, y) = x^2 + y^2 + 2\sin(x)\sin(y)$

https://github.com/AboudOnji/Ex_Fund_IC/blob/main/Cap7/Exmple_GA_Opt.m

Optimización de la función $\mathrm{ps}_e xample(x)$

https://github.com/AboudOnji/Ex_Fund_IC/blob/main/Cap7/Example_GA2.m