Optimization: MATLAB Optimization Toolbox

MATLAB Optimization Toolboxes

Optimization Toolbox

- linear programming: `linprog`
- quadratic programming: `quadprog`
- unconstrained minimization: `fminunc`, `fminsearch`
- nonlinear least squares: `lsqnonlin`
- constrained minimization: `fmincon`

Global Optimization Toolbox

- genetic algorithm: `ga`
- simulated annealing: `simulannealbnd`

Other toolboxes: NAG, OSL, minpack, MINOS, LANCELOT, ...

See also: Decision Tree for Optimization Software
`http://plato.la.asu.edu/guide.html`

Linear programming

Linear programming: `linprog`

$$\min_x \; c^T x \quad \text{subject to: } Ax \leqslant b,\; A_{\text{eq}}x = b_{\text{eq}}$$

Variant of simplex method

```
x=linprog(c,A,b,Aeq,beq,lb,ub,x0,options)
```

Lower and upper bounds: `lb` $\leqslant$ `x` $\leqslant$ `ub`

Initial guess: `x0`

Options: `options`

- `optimoptions('linprog','Algorithm','dual-simplex')` !!!
- `optimoptions('linprog','Algorithm','interior-point')`
- `optimoptions('linprog','Display','iter')` but ...
- `optimoptions('linprog','MaxIterations',100)` but ...

Brewery scheduling problem

$$\begin{aligned}
\min_{x_1, x_2} \quad & -20\,x_1 - 30\,x_2 \\
\text{s.t.} \quad & x_1 + x_2 \leqslant 100 \\
& 0.1\,x_1 + 0.2\,x_2 \leqslant 14 \\
& x_1,\, x_2 \qquad\qquad \geqslant 0
\end{aligned}$$

```
>> c=[-20 -30];
>> A=[ 1 1;0.1 0.2];
>> b=[100 14]';
>> lb=[0 0]';
>> ub=[Inf Inf]';
>> o=optimoptions('linprog','Algorithm','dual-simplex');
>> x=linprog(c,A,b,[],[],lb,ub,[],o)
Optimization terminated.
x =
   60
   40
```

Linear programming – Output arguments

`[x,fval,flag,output]=linprog(...)`

- `fval`: optimal value of the objective function
- `flag`

  $= 1$: converged to solution

  $= 0$: maximum number of iterations exceeded

  < 0 : algorithm did not find bounded optimal / feasible solution

  $\rightarrow$ use this information and always check value of `flag`!
- `output.iterations`: number of iterations

Quadratic programming

Quadratic programming: `quadprog`

$$\min_x \frac{1}{2}x^T Hx + c^T x \quad \text{subject to: } Ax \leqslant b,\ A_{\text{eq}}x = b_{\text{eq}}$$

Modified Simplex method

```
x=quadprog(H,c,A,b,Aeq,Beq,lb,ub,x0,options)
```

Lower and upper bounds: `lb` $\leqslant$ `x` $\leqslant$ `ub`

Initial guess: `x0`

Options: `options`

```
- optimoptions('quadprog',...
        'Algorithm','interior-point-convex') !!!
- optimoptions('quadprog',...
        'Algorithm','trust-region-reflective)
- optimoptions('quadprog','MaxIterations',100)
```

but ...

Quadratic programming – Output arguments

```
[x,fval,flag,output]=quadprog(...)
```

- `fval`: optimal value of the objective function
- `flag`

  $= 1$: converged to solution

  $= 0$: maximum number of iterations exceeded

  < 0 : algorithm did not converge/did not find an optimal solution

  $\rightarrow$ use this information and always check value of `flag`!
- `output.iterations`: number of iterations

System identification example

ARX model:

$$y(n+1) + ay(n) = bu(n) + e(n)$$

Given: $u(1), \ldots, u(4)$ and $y(1), \ldots, y(5)$

Find estimates $\hat{a}$ and $\hat{b}$ of a and b

$$\underbrace{\begin{bmatrix} \hat{e}(1) \\ \hat{e}(2) \\ \hat{e}(3) \\ \hat{e}(4) \end{bmatrix}}_{E} = \underbrace{\begin{bmatrix} y(2) \\ y(3) \\ y(4) \\ y(5) \end{bmatrix}}_{Y} - \underbrace{\begin{bmatrix} -y(1) & u(1) \\ -y(2) & u(2) \\ -y(3) & u(3) \\ -y(4) & u(4) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}}_{x}$$

System identification example (2)

$$\min_x E^T E = \min_x (Y - \Phi x)^T (Y - \Phi x)$$

$$= \min_x x^T \Phi^T \Phi x - 2 Y^T \Phi x + Y^T Y \quad = \min_x \frac{1}{2} x^T H x + c^T x$$

subject to

$$-0.99 \leqslant \hat{a} \leqslant 0.99$$

```
>> H=[10.1370 1.7334;1.7334 1.5364];
>> c=[-8.6306 -3.9850]';
>> lb=[-0.99 -Inf]';
>> ub=[ 0.99  Inf]';
>> o=optimoptions('quadprog','Algorithm','active-set');
>> x=quadprog(H,c,[],[],[],[],lb,ub,[],o)
Optimization terminated.
x =
    0.5054
    2.0236
```

Model Predictive Control (MPC) example

Plant:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned}$$

Objectives:

- Steer the output $y(k)$ to zero
- Keep the control effort $u(k)$ small

Assumption: all states are measurable

MPC example (2)

Performance index:

$$J(k) = \sum_{i=1}^{N_\mathrm{p}} y^2(k+i|k) + \lambda\, u^2(k+i-1|k)$$

Optimization problem:

$$\min_{u(k|k),\ldots,u(k+N_\mathrm{p}-1|k)} J(k)$$

$$\text{s.t. } |u(k+i-1|k)| \leqslant 0.25 \quad \text{for } i = 1, 2, \ldots, N_\mathrm{p}$$

Receding Horizon
Every time step, only $u(k|k)$ is applied, model/state is updated, and prediction window is shifted

We need **predictions of output** y at sample steps $k+i$, $i = 1, \ldots, N_\mathrm{p}$

MPC example (3)

$$\begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N_\mathrm{p}|k) \end{bmatrix} = \begin{bmatrix} A \\ \vdots \\ A^{N_\mathrm{p}} \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & & 0 \\ \vdots & & \ddots & \vdots \\ A^{N_\mathrm{p}-1}B & & \cdots & B \end{bmatrix} \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N_\mathrm{p}-1|k) \end{bmatrix}$$

$$\begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+N_\mathrm{p}|k) \end{bmatrix} = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C \end{bmatrix} \begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N_\mathrm{p}|k) \end{bmatrix}$$

$$\begin{aligned} \tilde{x} &= \tilde{A}x(k) + \tilde{B}\tilde{u} \\ \tilde{y} &= \tilde{C}\tilde{x} \;=\; \tilde{C}\tilde{A}x(k) + \tilde{C}\tilde{B}\tilde{u} \end{aligned}$$

$$\left.\begin{array}{l} \tilde{y} \text{ affine function of } \tilde{u} \\ \text{quadratic objective } \tilde{y}^T\tilde{y} + \lambda\tilde{u}^T\tilde{u} \\ \text{linear constraints} \end{array}\right\} \Rightarrow \text{quadratic programming problem}$$

Unconstrained nonlinear optimization

Unconstrained nonlinear optimization: `fminsearch`, `fminunc`

$$\min_{x} f(x)$$

- Nelder-Mead method

```
x=fminsearch(@fun,x0,options)
```

- Direction determination and line search

```
x=fminunc(@fun,x0,options)
```

with `fun.m` m-file that defines f and its gradient g (optional):

```
[f,g]=fun(x)
f=...
% Compute gradient if required.
if ( nargout > 1 )
   g=...
end;
```

fminunc — @ notation for functions

More complex example: specifying, e.g., parameters a and b

```
x=fminunc(@(x)fun(x,a,b),x0,options)
```

```
% Main code
...
a=2;
b=5;
x=fminunc(@(x)fun(x,a,b),x0,options);

[f,g]=fun(x,a,b)
f=...
% Compute gradient if required.
if ( nargout > 1 )
   g=...
end;
```

fminunc — Main options

Set `options` with `options=optimoptions('fminunc',...,...)`

- `'HessUpdate'`: sets search direction

  `'bfgs'` = BFGS direction

  `'dfp'` = DFP direction

  `'steepdesc'` = steepest-descent direction

Direction determination and line search

$$x_{k+1} = x_k + d_k s_k$$

d_k search direction

- BFGS: $d_k = -\hat{H}^{-1}(x_k)\nabla f(x_k)$
  $\hat{H}(x_k)$ approximation of the Hessian
- DFP: $d_k = -D(x_k)\nabla f(x_k)$
  $D(x_k)$ approximation of the inverse Hessian
- Steepest descent: $d_k = -\nabla f(x_k)$

s_k step length

$$s_k = \arg\min_s f(x_k + d_k s)$$

fminunc — Other options

- `'Algorithm'`: selects algorithm, e.g., `quasi-newton`
- `'Display'`: controls display of intermediate values
- `'GradObj'`: indicates whether gradient is defined by user
- `'MaxFunEvals'`: maximum number of function evaluations
- `'MaxIter'`: maximum number of iterations
- `'TolFun'`: termination tolerance on f
- `'TolX'`: termination tolerance on x
- `'DerivativeCheck'`: compare user-supplied gradient with finite-difference derivatives but ...

fminunc — Output arguments

```
[x,fval,flag,output]=fminunc(...)
```

- `flag`

  > 0 : converged to solution

  $= 0$: maximum number of function evaluations or iterations exceeded

  < 0 : algorithm did not converge

  $\rightarrow$ use this information and always check value of `flag`!

- `output.funcCount`: number of function evaluations

  `output.iterations`: number of iterations

Nonlinear least squares

$$f(x) = e^T(x)e(x)$$

$$\nabla f(x) = 2\nabla e(x)e^T(x)$$

$$H(x) = 2\nabla e(x)\nabla^T e(x) + \sum_{i=1}^{N} 2\nabla^2 e_i(x)e_i(x)$$

Approximation of the Hessian:

$$\hat{H}(x) := 2\nabla e(x)\nabla^T e(x)$$

Levenberg-Marquardt method:

$$x_{i+1} = x_i - \left(\lambda I + \hat{H}(x_i)\right)^{-1} \nabla f(x_i) s_i$$

Unconstrained nonlinear least squares

Unconstrained nonlinear least squares: `lsqnonlin`

$$\min_x \; e^T(x)e(x)$$

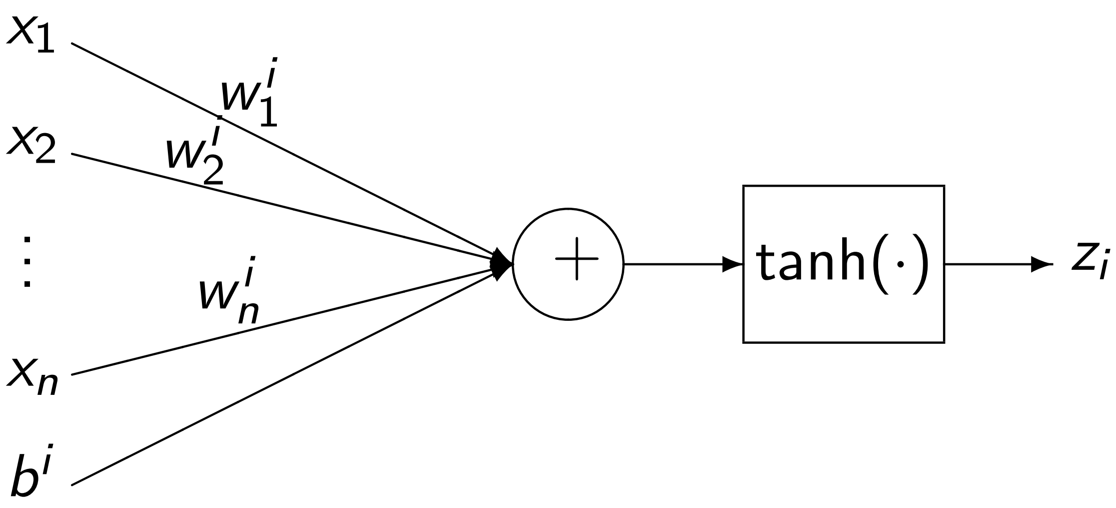
```
x=lsqnonlin(@fun,x0,lb,ub,options)
```

`fun` should return full vector $e(x)$, not just $e^T(x)e(x)$!

`options=optimoptions('lsqnonlin',...,...)`

- `'Algorithm'`:
  `'levenberg-marquardt'`: Levenberg-Marquardt
- `'Jacobian'`: use user-defined gradient/Jacobian

Neural network example



Neural network equations:

$$v_j(k) = \sum_{i=1}^{N^i} w_{ij}^h \phi_i(k) + b_j^h$$

$$\hat{y}(k) = \sum_{j=1}^{N^h} w_j^o \tanh\Big(v_j(k)\Big) + b^o \qquad \text{with } \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Matrix notation: $V = \Phi W^h + U_N B^h$

$$\hat{Y} = \tanh(V) W^o + U_N b^o$$

Neural network example (2)

Identification of a nonlinear plant:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$$

Neural network model: $\hat{y}(k) = f\Big(\phi(k)\Big)$

$$\phi(k) = \left[\begin{array}{cc} u(k-1) & y(k-1) \end{array}\right]$$

Model error:

$$e(k) := y(k) - \hat{y}(k)$$

Nonlinear least squares problem

$$\min_{W^h, W^o, B^h, b^o} \sum_{k=1}^{N} |e(k)|^2$$

$\rightarrow$ `lsqnonlin`

Constrained nonlinear optimization

Constrained nonlinear optimization: `fmincon`

$$\min_x f(x), \quad \text{subject to } h(x) = 0 \text{ and } g(x) \leqslant 0$$

Sequential Quadratic Programming

```
x=fmincon(@fun,x0,A,b,Aeq,beq,lb,ub,@nonlcon,options)
```

Lower and upper bounds: `lb` $\leqslant$ `x` $\leqslant$ `ub`

Linear constraints: `A x <= b, Aeq x = beq`

Nonlinear constraints: `c(x) <= 0, ceq(x) = 0`

`[c,ceq,Jc,Jceq]=nonlcon(x)`

`options=optimoptions('fmincon',...,...)`

- `'Algorithm': 'sqp'` $\rightarrow$ SQP

fmincon — Gradient and Jacobian

→ make sure to put the options 'GradObj' and 'GradConstr' to 'on'!

```
[f,g]=fun(x)
f=...
% Compute gradient if required.
if ( nargout > 1 )
    g=...
end;

[c,ceq,Jc,Jceq]=nonlcon(x)
c=...
ceq=...
% Compute TRANSPOSED Jacobians if required.
if ( nargout > 2 )
    Jc=...
    Jceq=...
end;
```

Genetic algorithm

Genetic algorithm: `ga`

$$\min_x f(x), \quad \text{subject to } h(x) = 0 \text{ and } g(x) \leqslant 0$$

Genetic algorithm:

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

Lower and upper bounds: `lb` $\leqslant$ `x` $\leqslant$ `ub`

Linear constraints: `A x <= b, Aeq x = beq`

Nonlinear constraints: `c(x) <= 0, ceq(x) = 0`

`[c,ceq]=nonlcon(x)`

Set `options` using `gaoptimset`!

e.g., `options=gaoptimset('Generations',50)`

Simulated annealing

Simulated annealing: `simulannealbnd`

$$\min_{x} f(x), \quad \text{subject to } x_{\text{lb}} \leqslant x \leqslant x_{\text{ub}}$$

Simulated annealing:

```
x=simulannealbnd(@fun,x0,lb,ub,options)
```

Lower and upper bounds: `lb` $\leqslant$ `x` $\leqslant$ `ub`

Set `options` using `saoptimset`!

e.g., `options=saoptimset('InitialTemperature',50)`

Summary

- Overview of main functions of Matlab Optimization Toolbox and of the Global Optimization Toolbox
- Main options and caveats