function [x,f,exitflag,output] = minFunc(funObj,x0,options,varargin)

% [x,f,exitflag,output] = minFunc(funObj,x0,options,varargin)

%

% Unconstrained optimizer using a line search strategy

%

% Uses an interface very similar to fminunc

% (it doesn't support all of the optimization toolbox options,

% but supports many other options).

%

% It computes descent directions using one of ('Method'):

% - 'lbfgs': Quasi-Newton with Limited-Memory BFGS Updating

% (default: uses a predetermined nunber of previous steps to form a

% low-rank Hessian approximation)

%

% Several line search strategies are available for finding a step length satisfying

% the termination criteria ('LS\_type')

% - 0 : A backtracking line-search based on the Armijo condition (default for 'bb')

% - 1 : A bracekting line-search based on the strong Wolfe conditions (default for all other methods)

% - 2 : The line-search from the Matlab Optimization Toolbox (requires Matlab's linesearch.m to be added to the path)

%

% For the Armijo line-search, several interpolation strategies are available ('LS\_interp'):

% - 0 : Step size halving

% - 1 : Polynomial interpolation using new function values

% - 2 : Polynomial interpolation using new function and gradient values (default)

%

% When (LS\_interp = 1), the default setting of (LS\_multi = 0) uses quadratic interpolation,

% while if (LS\_multi = 1) it uses cubic interpolation if more than one point are available.

%

% When (LS\_interp = 2), the default setting of (LS\_multi = 0) uses cubic interpolation,

% while if (LS\_multi = 1) it uses quartic or quintic interpolation if more than one point are available

%

% To use the non-monotonic Armijo condition, set the 'Fref' value to the number of previous function values to store

%

% For the Wolfe line-search, these interpolation strategies are available ('LS\_interp'):

% - 0 : Step Size Doubling and Bisection

% - 1 : Cubic interpolation/extrapolation using new function and gradient values (default)

% - 2 : Mixed quadratic/cubic interpolation/extrapolation

%

% Several strategies for choosing the initial step size are avaiable ('LS\_init'):

% - 0: Always try an initial step length of 1 (default for all except 'sd' and 'cg')

% (t = 1)

% - 1: Use a step similar to the previous step

% (t = t\_old\*min(2,g'd/g\_old'd\_old))

% - 2: Quadratic Initialization using previous function value and new

% function value/gradient (use this if steps tend to be very long, default for 'sd' and 'cg')

% (t = min(1,2\*(f-f\_old)/g))

% - 3: The minimum between 1 and twice the previous step length

% (t = min(1,2\*t)

% - 4: The scaled conjugate gradient step length (may accelerate

% conjugate gradient methods, but requires a Hessian-vector product, default for 'scg')

% (t = g'd/d'Hd)

%

% Inputs:

% funObj - is a function handle

% x0 - is a starting vector;

% options - is a struct containing parameters (defaults are used for non-existent or blank fields)

% varargin{:} - all other arguments are passed as additional arguments to funObj

%

% Outputs:

% x is the minimum value found

% f is the function value at the minimum found

% exitflag returns an exit condition

% output returns a structure with other information

%

% Supported Input Options

% Display - Level of display [ off | final | (iter) | full | excessive ]

% MaxFunEvals - Maximum number of function evaluations allowed (1000)

% MaxIter - Maximum number of iterations allowed (500)

% optTol - Termination tolerance on the first-order optimality (1e-5)

% progTol - Termination tolerance on progress in terms of function/parameter changes (1e-9)

% Method - [ sd | csd | bb | cg | scg | pcg | {lbfgs} | newton0 | pnewton0 |

% qnewton | mnewton | newton | tensor ]

% c1 - Sufficient Decrease for Armijo condition (1e-4)

% c2 - Curvature Decrease for Wolfe conditions (.2 for cg methods, .9 otherwise)

% LS\_init - Line Search Initialization - see above (2 for cg/sd, 4 for scg, 0 otherwise)

% LS - Line Search type - see above (2 for bb, 4 otherwise)

% Fref - Setting this to a positive integer greater than 1

% will use non-monotone Armijo objective in the line search.

% (20 for bb, 10 for csd, 1 for all others)

% numDiff - [ 0 | 1 | 2] compute derivatives using user-supplied function (0),

% numerically user forward-differencing (1), or numerically using central-differencing (2)

% (default: 0)

% (this option has a different effect for 'newton', see below)

% useComplex - if 1, use complex differentials if computing numerical derivatives

% to get very accurate values (default: 0)

% DerivativeCheck - if 'on', computes derivatives numerically at initial

% point and compares to user-supplied derivative (default: 'off')

% outputFcn - function to run after each iteration (default: []). It

% should have the following interface:

% outputFcn(x,iterationType,i,funEvals,f,t,gtd,g,d,optCond,varargin{:});

% useMex - where applicable, use mex files to speed things up (default: 1)

%

% Method-specific input options:

% newton:

% HessianModify - type of Hessian modification for direct solvers to

% use if the Hessian is not positive definite (default: 0)

% 0: Minimum Euclidean norm s.t. eigenvalues sufficiently large

% (requires eigenvalues on iterations where matrix is not pd)

% 1: Start with (1/2)\*||A||\_F and increment until Cholesky succeeds

% (an approximation to method 0, does not require eigenvalues)

% 2: Modified LDL factorization

% (only 1 generalized Cholesky factorization done and no eigenvalues required)

% 3: Modified Spectral Decomposition

% (requires eigenvalues)

% 4: Modified Symmetric Indefinite Factorization

% 5: Uses the eigenvector of the smallest eigenvalue as negative

% curvature direction

% cgSolve - use conjugate gradient instead of direct solver (default: 0)

% 0: Direct Solver

% 1: Conjugate Gradient

% 2: Conjugate Gradient with Diagonal Preconditioner

% 3: Conjugate Gradient with LBFGS Preconditioner

% x: Conjugate Graident with Symmetric Successive Over Relaxation

% Preconditioner with parameter x

% (where x is a real number in the range [0,2])

% x: Conjugate Gradient with Incomplete Cholesky Preconditioner

% with drop tolerance -x

% (where x is a real negative number)

% numDiff - compute Hessian numerically

% (default: 0, done with complex differentials if useComplex = 1)

% LS\_saveHessiancomp - when on, only computes the Hessian at the

% first and last iteration of the line search (default: 1)

% mnewton:

% HessianIter - number of iterations to use same Hessian (default: 5)

% qnewton:

% initialHessType - scale initial Hessian approximation (default: 1)

% qnUpdate - type of quasi-Newton update (default: 3):

% 0: BFGS

% 1: SR1 (when it is positive-definite, otherwise BFGS)

% 2: Hoshino

% 3: Self-Scaling BFGS

% 4: Oren's Self-Scaling Variable Metric method

% 5: McCormick-Huang asymmetric update

% Damped - use damped BFGS update (default: 1)

% newton0/pnewton0:

% HvFunc - user-supplied function that returns Hessian-vector products

% (by default, these are computed numerically using autoHv)

% HvFunc should have the following interface: HvFunc(v,x,varargin{:})

% useComplex - use a complex perturbation to get high accuracy

% Hessian-vector products (default: 0)

% (the increased accuracy can make the method much more efficient,

% but gradient code must properly support complex inputs)

% useNegCurv - a negative curvature direction is used as the descent

% direction if one is encountered during the cg iterations

% (default: 1)

% precFunc (for pnewton0 only) - user-supplied preconditioner

% (by default, an L-BFGS preconditioner is used)

% precFunc should have the following interfact:

% precFunc(v,x,varargin{:})

% lbfgs:

% Corr - number of corrections to store in memory (default: 100)

% (higher numbers converge faster but use more memory)

% Damped - use damped update (default: 0)

% cg/scg/pcg:

% cgUpdate - type of update (default for cg/scg: 2, default for pcg: 1)

% 0: Fletcher Reeves

% 1: Polak-Ribiere

% 2: Hestenes-Stiefel (not supported for pcg)

% 3: Gilbert-Nocedal

% HvFunc (for scg only)- user-supplied function that returns Hessian-vector

% products

% (by default, these are computed numerically using autoHv)

% HvFunc should have the following interface:

% HvFunc(v,x,varargin{:})

% precFunc (for pcg only) - user-supplied preconditioner

% (by default, an L-BFGS preconditioner is used)

% precFunc should have the following interface:

% precFunc(v,x,varargin{:})

% bb:

% bbType - type of bb step (default: 0)

% 0: min\_alpha ||delta\_x - alpha delta\_g||\_2

% 1: min\_alpha ||alpha delta\_x - delta\_g||\_2

% 2: Conic BB

% 3: Gradient method with retards

% csd:

% cycle - length of cycle (default: 3)

%

% Supported Output Options

% iterations - number of iterations taken

% funcCount - number of function evaluations

% algorithm - algorithm used

% firstorderopt - first-order optimality

% message - exit message

% trace.funccount - function evaluations after each iteration

% trace.fval - function value after each iteration

%

% Author: Mark Schmidt (2005)

% Web: http://www.di.ens.fr/~mschmidt/Software/minFunc.html

%

% Sources (in order of how much the source material contributes):

% J. Nocedal and S.J. Wright. 1999. "Numerical Optimization". Springer Verlag.

% R. Fletcher. 1987. "Practical Methods of Optimization". Wiley.

% J. Demmel. 1997. "Applied Linear Algebra. SIAM.

% R. Barret, M. Berry, T. Chan, J. Demmel, J. Dongarra, V. Eijkhout, R.

% Pozo, C. Romine, and H. Van der Vost. 1994. "Templates for the Solution of

% Linear Systems: Building Blocks for Iterative Methods". SIAM.

% J. More and D. Thuente. "Line search algorithms with guaranteed

% sufficient decrease". ACM Trans. Math. Softw. vol 20, 286-307, 1994.

% M. Raydan. "The Barzilai and Borwein gradient method for the large

% scale unconstrained minimization problem". SIAM J. Optim., 7, 26-33,

% (1997).

% "Mathematical Optimization". The Computational Science Education

% Project. 1995.

% C. Kelley. 1999. "Iterative Methods for Optimization". Frontiers in

% Applied Mathematics. SIAM.

高亮部分是LBFGS方法自给的，其余均是默认值

function [verbose,verboseI,debug,doPlot,maxFunEvals,maxIter,optTol,progTol,method,... corrections,c1,c2,LS\_init,cgSolve,qnUpdate,cgUpdate,initialHessType,...

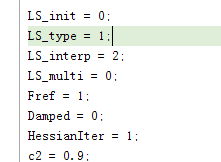
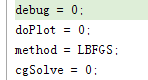
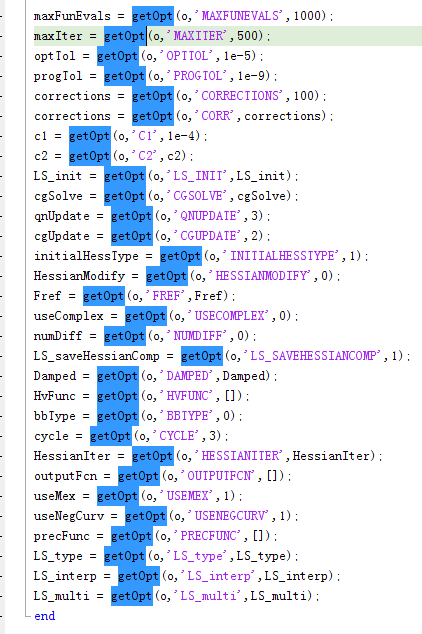
HessianModify,Fref,useComplex,numDiff,LS\_saveHessianComp,...

Damped,HvFunc,bbType,cycle,...

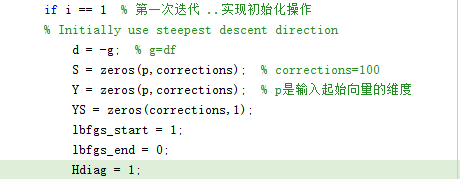
HessianIter,outputFcn,useMex,useNegCurv,precFunc,...

LS\_type,LS\_interp,LS\_multi,DerivativeCheck]

= minFunc\_processInputOptions(o)

 verbose,verboseI 均为0，maxFunEvals 为25，method 为5

第一次迭代:





function [t,f\_new,g\_new,funEvals,H] = WolfeLineSearch(...

x,t,d,f,g,gtd,c1,c2,LS\_interp,LS\_multi,maxLS,progTol,debug,doPlot,saveHessianComp,funObj,varargin)

%

% Bracketing Line Search to Satisfy Wolfe Conditions

%

% Inputs:

% x: starting location

% t: initial step size

% d: descent direction

% f: function value at starting location

% g: gradient at starting location

% gtd: directional derivative at starting location

% c1: sufficient decrease parameter

% c2: curvature parameter

% debug: display debugging information

% LS\_interp: type of interpolation

% maxLS: maximum number of iterations

% progTol: minimum allowable step length

% doPlot: do a graphical display of interpolation

% funObj: objective function

% varargin: parameters of objective function

%

% Outputs:

% t: step length

% f\_new: function value at x+t\*d

% g\_new: gradient value at x+t\*d

% funEvals: number function evaluations performed by line search

% H: Hessian at initial guess (only computed if requested

WolfeLineSearch

第一次迭代参数