MATLAB-based Optimization
Basic Capabilities

Gene Cliff (AOE/ICAM - ecliff@vt.edu)
3:00pm - 4:45pm, Monday, 4 February 2013
..... FDI .....

AOE: Department of Aerospace and Ocean Engineering
ICAM: Interdisciplinary Center for Applied Mathematics

VirginiaTech

MATLAB-based Optimization

- Introduction & *function* functions $\Leftarrow$
- `fminbnd`
- `fminsearch`
- `lsqnonneg`
- `fzero`

VirginiaTech

INTRO:

Basic MATLAB provides several `functions` for elementary minimization and zero-finding problems.

The *Optimization Toolbox* provides techniques for advanced problems, and a *gui* interface.

The OPTIMIZATION TOOLBOX is included with the VT *concurrent license*

(http://www2.ita.vt.edu/software/department/products/mathworks/matlab_annual_concurrent/index.html), and with the VT *student license*

(http://www2.ita.vt.edu/software/student/products/mathworks/matlab/index.html) but not with the VT *annual stand alone license*

(http://www2.ita.vt.edu/software/department/products/mathworks/matlab_annual_standalone/index.html).

Type `ver` ⟨enter⟩ at the MATLAB prompt to see the list of toolboxes available with your license.

In these notes we discuss optimization and zero-finding routines in basic MATLAB. The *Optimization Toolbox* will be discussed in the next lecture.

Basic MATLAB: linear least-squares

- Given an $m \times n$ matrix A, a vector $b \in \mathbb{R}^m$ with $m > n$.
- With more equations than unknowns we do not expect to find an $x \in \mathbb{R}^n$ such that $Ax = b$.
- Instead we seek a *least squares* solution, *i.e.* an x that minimizes $\|Ax - b\|^2$.
- Such an x can be computed by `mldivide` (*matrix left-divide*) $x = A\backslash b$ or $x = mldivide(A, b)$

  If $m = n$ and A is nonsingular, $A\backslash b$ is similar to `inv(A)*b`, though it is not computed that way.
- Other procedures (*e.g.* `minres`, `gmres`) are available if A is sparse or if $A * x$ is the output of a function, *e.g.* `my_Ax(x)`.
- In MATLAB `inv(A)` is rarely a good idea.

VirginiaTech

Basic MATLAB: *function* functions

MATLAB uses this terminology to indicate that the arguments of some `functions` may refer to other *functions*.

A function handle is a MATLAB value that provides a means of calling a function indirectly. You can pass function handles in calls to other functions (often called *function* functions). You can also store function handles in data structures for later use. A function handle is one of the standard MATLAB data types.

`S = functions(funhandle)`
returns, in the MATLAB structure **S**, the function name, type, filename, and other information for the function handle stored in the variable *funhandle*. Note that `functions` is intended for debugging; do not use it your coding.

VirginiaTech

A *named* function is defined in `.m` file; the name on the `function` line should agree with the file name (which takes precedence).

local functions (or *scoped functions*) are included in the same file as a named function and are visible only to it.

```
function x_out = eg_named(x_in)
% Simple example of local (sub-) functions
% We can make use of local_1 and local_2
% which are not visible to other functions
        x_out = local_1(x_in) + local_2(x_in);
%       S     = functions(@local_1)
end
%  Sub functions go here
function out = local_1(in)
% Square the entries
         out = in.^2;
end
function out = local_2(in)
% Multiply by 2
          out = 2*in;
end
```

Note that each *function/subfunction* requires an `end` statement.

nested functions are included in the same file as a named function and are visible only to it. These functions are included before the `end` statement of the host function.

```
function x_out = eg_nested(x_in)
% Simple example of nested (sub-) functions
% We can make use of local_1 and local_2
% which are not visible to other functions
%
% Variables within the scope of this function are available to the nested functions

        temp  = cos(x_in);
        x_out = local_1(x_in) + local_2(x_in);
%       S     = functions(@local_1)
%  nested functions are 'inside' the containing functions 'end'
     function out = local_1(in)
         % Square the entries
         out = in.^2 + temp; % 'temp' is available to nested functions
     end % end for local_1
     function out = local_2(in)
         % Multiply by 2
         out = 2*in;
     end % end for local_2

end  % end for eg_nested
```

MATLAB: a simple named function - my_func

A simple named function

```
function y = my_func(x, param)
%Simple parameterized scalar function
    y = -x.*cos(param*x);
end
```

Examine with functions()

```
   >> S = functions(@my_func)
S =
    function: 'my_func'
        type: 'simple'
        file: [1x70 char]

>> S.file

ans =
/Users/ecliff/Documents/classes/fdi_matlab_2013/optimization/sample_code/my_func.m
```

VirginiaTech

anonymous functions are defined locally and do not exist in a file.

To assign a handle to an anonymous function use the '@ 'symbol

```
% This can be a function or a script
%
% Variables defined in this code can be used
% as arguments in defining an anonymous function

   parameter  = stuff;

%% Define the (handle to the) anonymous function
   fun_hndl = @(x) my_func(x, parameter);

%% Use the anonymous function
   temp = feval(fun_hndl, 3.2);
```

`feval` is a *built-in function* whose first argument is a *function*

MATLAB: using functions() again

```
param = 2;
fun_hndl = @(x) my_func(x, param);
param = 4;
S = functions(fun_hndl)
S =
     function: '@(x)my_func(x,param)'
         type: 'anonymous'
         file: ''
    workspace: {[1x1 struct]}
fields(S.workspace{1})
ans =
    'param'
S.workspace{1}.param
ans =
    2
```

VirginiaTech

MATLAB-based Optimization

- Introduction & *function* functions
- `fminbnd` $\Leftarrow$
- `fminsearch`
- `lsqnonneg`
- `fzero`

VirginiaTech

Basic MATLAB: fminbnd

`fminbnd` computes the minimum of a scalar-valued function of a scalar variable within a given interval.

```
>> param = 2;
>> fun_hndl = @(x) my_func(x, param);
>> opt = optimset('fminbnd');
>> opt = optimset(opt, 'Display', 'iter');
>> xs = fminbnd(fun_hndl, 0, 1, opt);
 Func-count     x          f(x)          Procedure
     1       0.381966    -0.275826        initial
     2       0.618034    -0.203032        golden
     .... 3 lines omitted
     6       0.430208    -0.280548        parabolic
     7       0.430161    -0.280548        parabolic
     8       0.430127    -0.280548        parabolic
Optimization terminated:
 the current x satisfies the termination
 criteria using OPTIONS.TolX of 1.000000e-04
```

VirginiaTech

Basic MATLAB: available arguments for `fminbnd`

`fminbnd` can be invoked with additional output arguments
doc fminbnd
for details

Syntax

```
x = fminbnd(fun,x1,x2)
x = fminbnd(fun,x1,x2,options)
[x,fval] = fminbnd(...)
[x,fval,exitflag] = fminbnd(...)
[x,fval,exitflag,output] = fminbnd(...)
```

VirginiaTech

MATLAB-based Optimization

- Introduction & *function* functions
- `fminbnd`
- `fminsearch` $\Leftarrow$
- `lsqnonneg`
- `fzero`

VirginiaTech

Basic MATLAB: fminsearch

- `fminsearch` computes a candidate for the minimum of a scalar-valued function of a vector variable from a given starting point .
- With $x \in \mathbb{R}^n$ the procedure is based on a *simplex* of $(n+1)$ points, and proceeds to produce a test point to replace the *worst* point in the simplex.
- The MATLAB algorithm begins by evaluating the function at x_0 and the n perturbed points $x_i = x_0 + \delta * e_i$, with $\delta = 0.05\|x_0\|$. The points are re-labeled to $x_j, \quad j = 1, ..., n+1$ so that $f(x_j) \leq f(x_{j+1})$, *i.e.* in ascending order.
- Compute $\overline{x}$, the average over the set $\{x_k |\ k = 1, .., n\}$.

VirginiaTech

The algorithm then proceeds as described in the following figures:

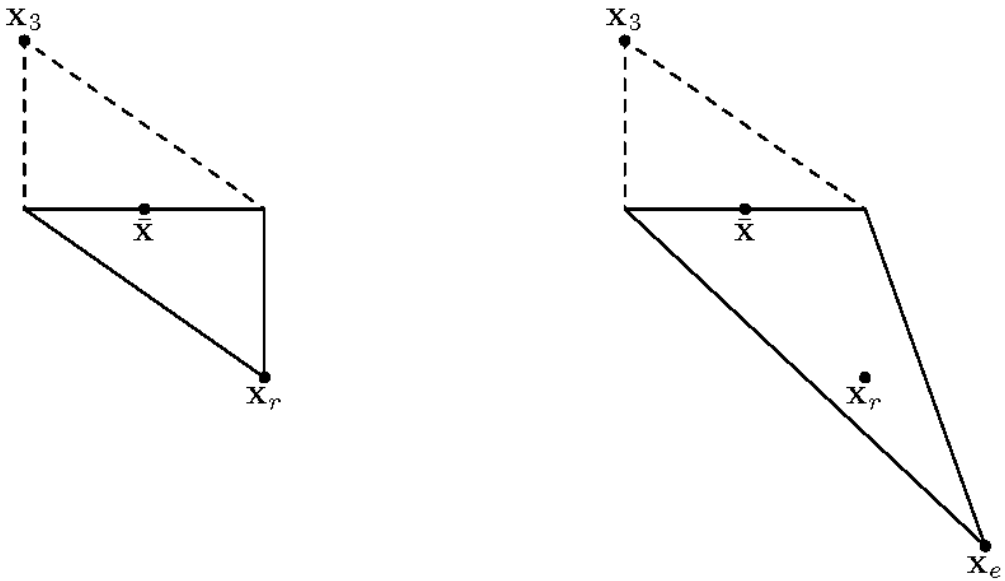


FIG. 1. *Nelder–Mead simplices after a reflection and an expansion step. The original simplex is shown with a dashed line.*

Basic MATLAB: `fminsearch` (cont'd)

or:

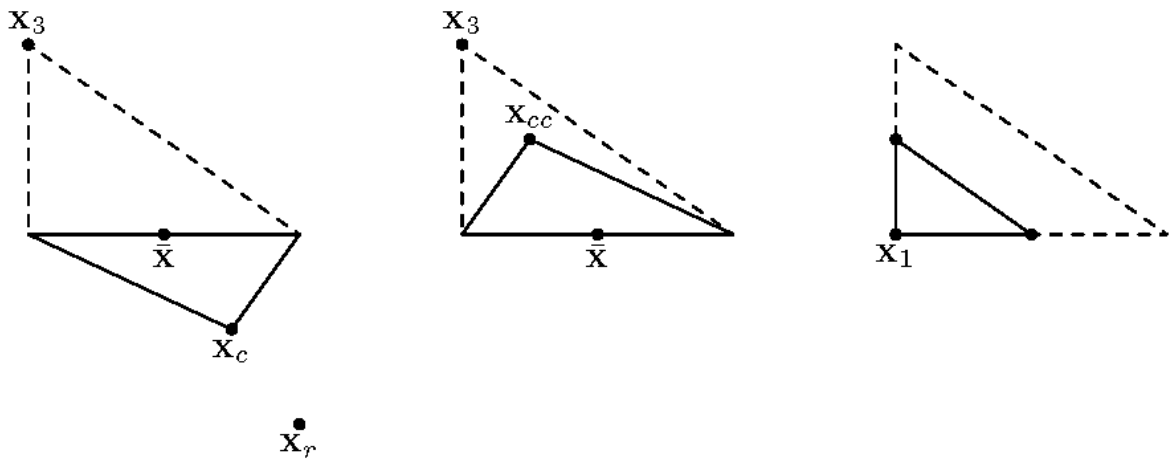


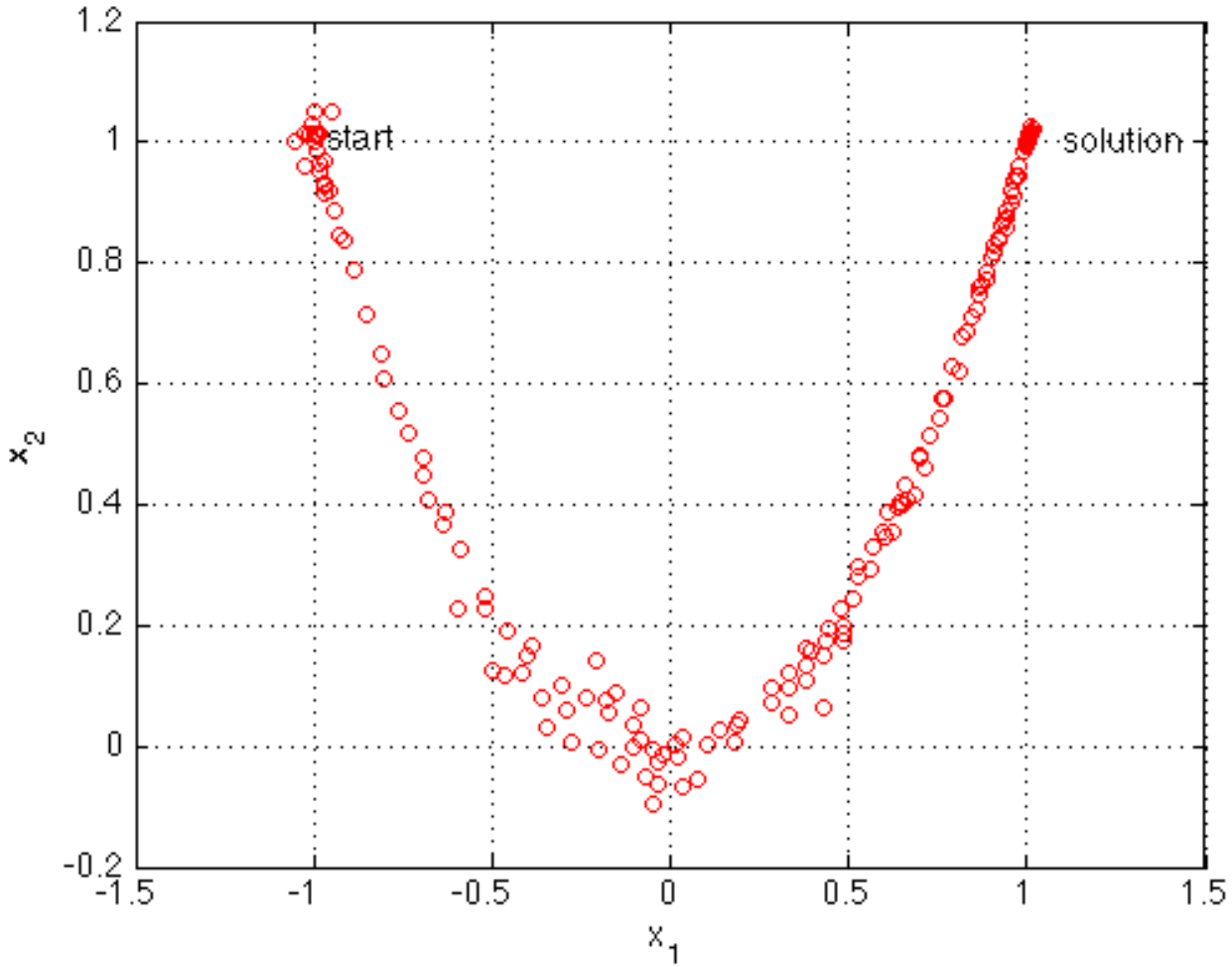
FIG. 2. *Nelder–Mead simplices after an outside contraction, an inside contraction, and a shrink. The original simplex is shown with a dashed line.*

fminsearch for Rosenbrock's example

```
% Script to display use of fminsearch on the Rosenbrock valley function

%
% E.M. Cliff
% Interdisciplinary Center for Applied Mathematics
% Virginia Tech, Blacksburg, VA 24061
% ecliff@vt.edu
%
% FDI Short course on Matlab-based Optimization - basic capabilities

% Example using the Rosenbrock valley function
% The first term in the sum is zero on the parabola x(2)=x(1).^2, and
% increases rapidly as x(:) departs from the 'valley'.

f_hndl = @(x) 100*(x(2) - x(1)*x(1))^2 + (1 - x(1))^2; % anonymous function

x0     = [-1; 1];                % starting point

opt    = optimset('fminsearch');
opt    = optimset(opt, 'Display', 'iter');

xs = fminsearch(f_hndl, x0, opt);  % invoke Nelder Meade
```

```
 Iteration   Func-count     min f(x)         Procedure
     0           1                4
     1           3                4         initial simplex
     2           5                4         contract inside
     3           7                4         contract inside
     4           9                4         contract outside
     5          11                4         contract inside
     6          13          3.93698         expand

    99         185      2.64787e-09         contract outside
   100         187      4.05759e-10         contract inside
```

`fminsearch` for Rosenbrock's example



VirginiaTech

Setting algorithmic parameters: `optimset`

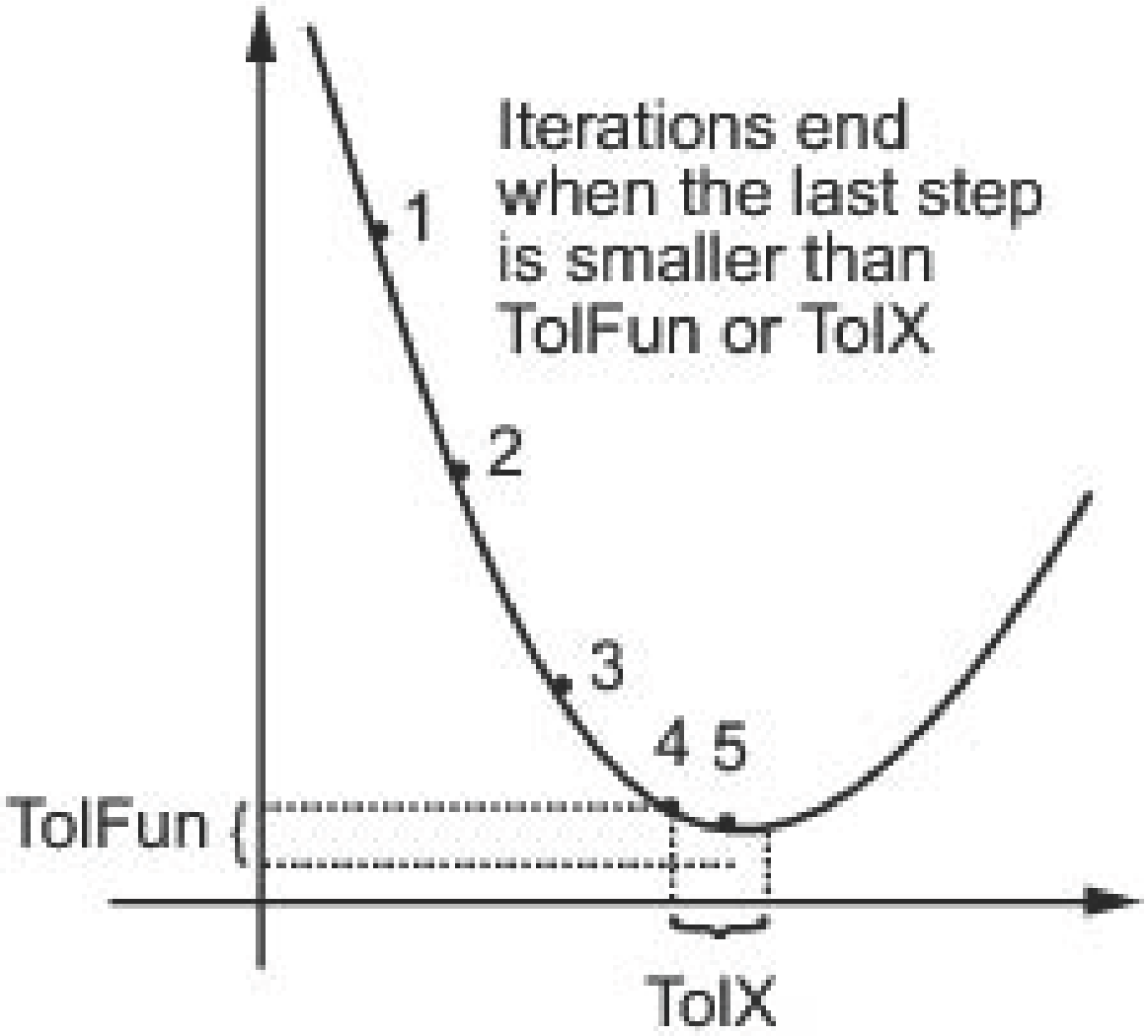
Syntax `options =`

```
optimset('param1',value1,'param2',value2,...)
options = optimset(optimfun)
options = optimset(oldopts,'param1',value1,...)
options = optimset(oldopts,newopts)
```

Which parameters are meaningful depends on the `function` and the *algorithm*.

VirginiaTech

Interpreting TolFun and TolX



virginiaTech

MATLAB-based Optimization

- Introduction & *function* functions
- `fminbnd`
- `fminsearch`
- `lsqnonneg`⇐
- `fzero`

VirginiaTech

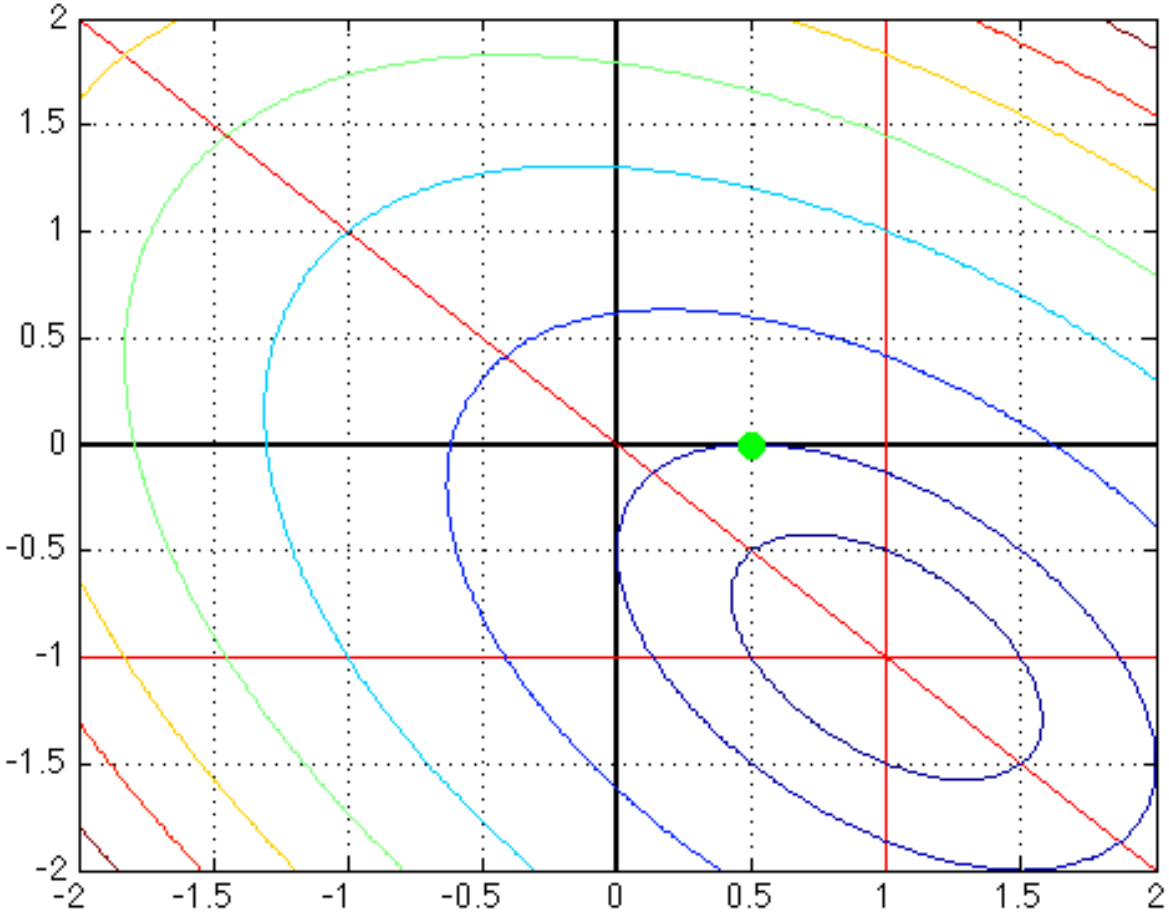
lsqnonneg

`x = lsqnonneg( A, b)`
minimize $\|Ax - b\|^2$ subject to $x \geq 0$

```
>> A = [ 1 1; 1 0; 0 1];
>> b = [0; 1; -1];
>> x1 = lsqnonneg(A, b)

x1 =

    0.5000
         0
```

VirginiaTech

lsqnonneg



VirginiaTech

MATLAB-based Optimization

- Introduction & *function* functions
- `fminbnd`
- `fminsearch`
- `lsqnonneg`
- `fzero`⇐

VirginiaTech

fzero

- `xs = fzero(fun, x0)` tries to find a zero of the scalar-valued function `fun` near the scalar `x0`.
- if `x0` is a vector of length 2, then `fzero` assumes that `fun(x0(1))` and `fun(x0(2))` have opposite signs. Assuming that `fun` is continuous, `fzero` will return a scalar near where `fun` changes sign.
- If unsuccessful, `fzero` will return a `NaN`.
- `fzero(fun, x0, optim)` allows one to set OPTIONS.

VirginiaTech

fzero: an example

```
>> fh = @(x) sin(x)

fh =

    @(x)sin(x)

>> fzero(fh, 1)

ans =

   1.5485e-24

>> fzero(fh, 3)

ans =

    3.1416
```

VirginiaTech

THE END

- In next week's class we'll cover the

  Optimization Toolbox

  particularly `fmincon`.

- We shall use today's material on `nested functions`

Please complete the evaluation form
`http://www.fdi.vt.edu/training/evals/`

Thanks

VirginiaTech