Below is an incomplete grammar for Matlab, adapted from a grammar posted at Stackoverflow. (Octave has a complete parser.) Symbols in single quotes and words in UPPERCASE are tokens. In Matlab try `help punct`, `help ':'`, `help relop`, `help if`, `help while`, `help clear`, etc.

```
statement
                  ->  global_statement
                  ->  clear_statement
                  ->  assignment_statement
                  ->  expression_statement
                  ->  selection_statement
                  ->  iteration_statement
                  ->  jump_statement
statement_list
                  ->  statement
                  ->  statement_list  statement
eostmt
                  ->   ','
                  ->   ';'
                  ->   CR
global_statement
                  ->  GLOBAL  identifier_list  eostmt
clear_statement
                  ->  CLEAR  identifier_list  eostmt
identifier_list
                  ->  IDENTIFIER
                  ->  identifier_list  IDENTIFIER
assignment_statement
                  ->  assignment_expression  eostmt
assignment_expression
                  ->  postfix_expression  '='  expression
expression_statement
                  ->  eostmt
                  ->  expression  eostmt
selection_statement
                  ->  IF  expression  statement_list  END   eostmt
                  ->  IF  expression  statement_list  ELSE  statement_list  END  eostmt
                  ->  IF  expression  statement_list  elseif_clause  END  eostmt
                 ->  IF  expression  statement_list  elseif_clause  ELSE  statement_list  END  eostmt
elseif_clause
                  ->  ELSEIF  expression  statement_list
                  ->  elseif_clause  ELSEIF  expression  statement_list
iteration_statement
                  ->  WHILE  expression  statement_list END  eostmt
                  ->  FOR  IDENTIFIER  '='  expression  statement_list END  eostmt
                  ->  FOR  '('  IDENTIFIER  '='  expression  ')'  statement_list END  eostmt
jump_statement
                  ->  BREAK   eostmt
                  ->  RETURN  eostmt
translation_unit
                  ->  statement_list
                  ->  FUNCTION  function_declare  eostmt  statement_list
function_identifier_list
                  ->  IDENTIFIER
                  ->  function_identifier_list  ','  IDENTIFIER
function_return_list
                  ->  IDENTIFIER
                  ->  '['  function_identifier_list  ']'
function_declare_lhs
                  ->  IDENTIFIER
                  ->  IDENTIFIER  '('  ')'
                  ->  IDENTIFIER  '('  function_identifier_list  ')'
function_declare
                  ->  function_declare_lhs
                  ->  function_return_list  '='  function_declare_lhs
```

```
expression
                ->  or_expression
                ->  expression  ':'  or_expression
or_expression
                ->  and_expression
                ->  or_expression  '|'  and_expression
and_expression
                ->  equality_expression
                ->  and_expression  '&'  equality_expression
equality_expression
                ->  relational_expression
                ->  equality_expression    '=='   relational_expression
                ->  equality_expression    '~='   relational_expression
relational_expression
                ->  additive_expression
                ->  relational_expression  '<'    additive_expression
                ->  relational_expression  '>'    additive_expression
                ->  relational_expression  '<='   additive_expression
                ->  relational_expression  '>='   additive_expression
additive_expression
                ->  multiplicative_expression
                ->  additive_expression   '+'   multiplicative_expression
                ->  additive_expression   '-'   multiplicative_expression
multiplicative_expression
                ->  unary_expression
                ->  multiplicative_expression  '*'   unary_expression
                ->  multiplicative_expression  '/'   unary_expression
                ->  multiplicative_expression  '\'   unary_expression
                ->  multiplicative_expression  '^'   unary_expression
                ->  multiplicative_expression  '.*'  unary_expression
                ->  multiplicative_expression  './'  unary_expression
                ->  multiplicative_expression  '.\'  unary_expression
                ->  multiplicative_expression  '.^'  unary_expression
unary_expression
                ->  postfix_expression
                ->  unary_operator  postfix_expression
unary_operator
                ->  '+'
                ->  '-'
                ->  '~'
postfix_expression
                ->  primary_expression
                ->  IDENTIFIER  '('  index_expression_list  ')'
                ->  postfix_expression  HTRANSPOSE                  %%%  HTRANSPOSE = '
                ->  postfix_expression  TRANSPOSE                   %%%  TRANSPOSE  = .'
primary_expression
                ->  IDENTIFIER
                ->  CONSTANT
                ->  STRING                                          %%%  STRINGs are quoted with  '
                ->  '('  expression  ')'
                ->  '['  ']'
                ->  '['  array_list  ']'
array_list
                ->  array_element
                ->  array_list  array_element
array_element
                ->  expression
                ->  expression_statement
index_expression
                ->  ':'
                ->  expression
index_expression_list
                ->  index_expression
                ->  index_expression_list  ','  index_expression
```