CMPSC 200 Exam II Cheat Sheet

by xsgirl99 via cheatography.com/26903/cs/7655/

3 Code Categories

Sequences – lines of code are executed one after another.

Selection Structures – executes some piece of code if some known condition is true, otherwise executes some sort of alternative code.

Repetition Structures (loops) – causes a group of statements to be executed multiple times (either a fixed number, or until some stated condition is met).

Relational Operators

`<`	less than
`<=`	Less than or equal to
`>`	greater than
`>=`	greater than or equal to
`==`	equality check
`~=`	not equal to

Logical Operators

`&`	and (used with vectors)
`|`	or (used with vectors)
`&&`	short cut and (used with scalars)
`xor`	exclusive or
`~`	not (used with vectors or scalars)
`||`	short cut or (used with scalars)

Remember 0 is false, 1 is true

Also called short circuit evaluation. Makes sure you don't evaluate additional terms if you don't need to do so.

Definitions

Pseudocode- Verbal description of code. Often is language independent. Intermediate step between everyday language and a programming language.

Flowchart- Exactly what it sounds like – a graphical representation of how code flows or progresses.

Counter- A variable that keeps track of some parameter of interest

Array- Holds “stuff”. It can hold numeric information, character data, symbolic data, etc. Is “an orderly grouping of information”, has no special properties by virtue of its existence.

Matrix-A 2D numeric array used in linear algebra. Is used extensively in STEM fields. Has special mathematical properties!

Code Vectorization- A programming technique that uses vector operations instead of element by element loop-bases operations.

Sentinel Loop- A loop that terminates only when a specific condition(the sentinel condition) is met

Find Command

The command `find` searches a matrix, finding what elements meet the search criteria. The command returns the index/indices of the valid results.

`index = find(x)`

`index = find(x,k)` (first k elements)

Find Command (cont)

`index = find(x,k,'last')` (last k elements)

if Selection Structure

```
if comparison
[commands to do something]
end
```

The simplest “classical” selection structure is the `if` statement.

If the comparison evaluates to be true, then the “do something” statements are executed. If the comparison evaluates to be false, then the “do something” statements are ignored.

Matrix Functions

Raising a matrix to a power can be thought of as multiplying the matrix by itself however many times in the exponent

Matrix must be square. Example: $A^4 = (A)(A)(A)(A)$

Syntax: A^n (requires A to be square; compare against the element-by-element operation `A.^n`, where A doesn’t have to be square)

Matrix inverse: $(A)(A^{-1})=1$ (identity matrix)

A matrix inverse does not exist if the determinant of the matrix is equal to 0; a matrix like this is known as a singular matrix

If a matrix is **square** and **singular**, the operation $M^{-1}M$ would be hard to know *a priori* but the result will be wrong.

Bits & Bytes

The basic unit of information in computers is the bit(“binary digit”). Can store exactly 1 logical variable

Bits can only have one of two values: 0 or 1

There are 8 bits/byte

Power of 2: allows values between 0 and 255 for 1 byte

Solving Matricies

```
A = [a matrix] A's #of rows=b's #of columns
b = [another matrix]
solution = inv(A)*b
A_augmented = [A b]
RREF_result = rref(A_augmented);
solution = RREF_result(:,end)
solution = A\b (ideal way)
```

Interpolation/Extrapolation

Interpolation consists of “method[s] of constructing new data points within the range of a discrete set of known data points”

Extrapolation consists of “the process of estimating, beyond the original observation range, the value of a variable on the basis of its relationship with another variable”

Interpolation/Extrapolation (cont)

Curve fitting is “the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints.” Goal is to minimize the residuals: the difference between the actual and predicted values at a given point

`yi = interp1(x,Y,xi)`. Interpolates to find `yi`, the interpolated function values at the points in the vector or array `xi`.`x` contains your known data points (functions values are `Y`), which must be a vector, though xi can be a scalar, vector, or multidimensional array. `yi` will always be the same size as `xi`

`yi = interp1(Y,xi)`. Same as above, except the function will assume that `x = 1:N`, where N is `length(Y)` (for a vector) or `size(Y,1)` (for a matrix)

`yi = interp1(x,Y,xi,method)`. Same as above, except now using a method(ie cubic spline)

`yi = interp1(x,Y,xi,method,'extrap')`. Same as above, except used to extrapolate beyond the data set

Flowcharts & Meaning

Circle/Oval - Indicates the beginning or end of a section of code

Parallelogram - Indicates input or output processes

Diamond - Indicates a decision point

Flowcharts & Meaning (cont)

Rectangle/Square - Indicates calculations

if/else/elseif

```
if comparison % do something
elseif comparison % do something else
else % do something else
end
```

If the first statement (the if statement) does not evaluate to true, it checks the elseif statement(s) If nothing is true by the time one gets to else, the else commands are executed

There can only be one `if`, there must be an `end`, and there can be no more than 1 `else`.

You can have an `else` without an `elseif`, and an `elseif` without an `else`. However, both `else` and `elseif` are dependent on having an `if`.

Switches

Switches have a similar purpose to `if` statements.

Anything you can do with a switch can be done using `if/elseif/else`.

Often personal preference, though you will often see switches when checking strings.

Important: In MATLAB, once a “true” case has been found MATLAB will NOT check the other cases – make sure you plan accordingly.

Switch Example

```
location = 'lion shrine';
switch location
    case 'lion shrine'
        disp('I''m at the lion shrine')
    otherwise
        disp('I''m lost')
end
Output:
I'm at the lion shrine
```

Menus

Instead of requesting input from the Command Window, you can have MATLAB collect input from a menu box.

Syntax: `var = menu('title','option 1','option 2','...')`

Use in conjunction with Switches.

`x=menu('S','x','y'...) if S==1->fprintfx elseif S==2->fprintfy`

numel vs find

`numel` counts the number of elements. `find` only returns the indices where the element that meets that criteria is located, so we count the number of elements.

Matrix Terminology

Zero matrix: matrix of zeros

Identity matrix: a matrix of zeros, except it has 1’s along the main diagonal

Sparse Matrix: most of the elements of the matrix have zero elements

Matrix Terminology (cont)

Dense Matrix: most of the elements of the matrix have non-zero elements

Banded Matrix: non-zero elements are confined to a diagonal band comprising the main diagonal and zeros or more diagonals on either size

Bidiagonal matrix: zero matrix except: for non- zero entries along the main diagonal and either the diagonal above or below the main diagonal

Tridigonal matrix: zero matrix except: for non-zero entries along the main diagonal and on the first diagonal above and below the main diagonal

Cell Arrays

Unlike numeric, character or symbolic arrays, cell arrays can store different data types within the same array. Each element of the array is an array.

Syntax: `mycell = {A, B, C, ...}` (the curly braces are cell array constructors)

`reshape` command reshapes the array Syntax: `reshape(A, r,c ...), reshape(A,r,[])`

`horzcat`command concatenates horizontally (left-right) Syntax: `horzcat(A1, A2, ... )`

Cell Arrays (cont)

`vertcat` command concatenates vertically (up-down) Syntax: `vertcat(A1, A2, A3, ...)`

A = 'We are!'
B = [1 4; 3 2]
C = 'Penn State!'
D = single([1 2; 3 4])
E = {A,B,C,D} % default printing just shows sizes
celldisp(E) % needed to generate display

E{end}=[ ] %will delete the last cell of the cell array E

Character Arrays

Character arrays are arrays of characters

Key idea: The number of elements in each row has to be the same, or MATLAB will throw a warning.

`char` will also accept as input the ASCII representation of a number, letter or symbol.

Miltivariate Interpolation

Accomplished using the commands `interp2` (for 2D data) or `interp3` (for 3D data)

`ZI = interp2(X,Y,Z,XI,YI)`. ZI is a matrix containing elements corresponding to the elements of XI and YI, as determined by interpolation within the 2D function specified by the matrices X, Y and Z. X and Y must be monotonic and have the same format ("plaid") as if they were produced by meshgrid. Matrices X and Y specify the points at which the data Z is given. Out of range values are returned as `NaNs`

Curve Fitting

`y = polyval(p,x)`. Returns the value of the polynomial y of degree n evaluated at x. The input argument p is a vector of length n + 1 whose elements are the coefficients in descending powers of the polynomial y. This function accepts matrix or vector x.

`p = polyfit(x,y,n)`. This finds the coefficients of the polynomial p(x) of degree n that fits the data. p is a row vector of length n + 1 that contains the coefficients of the polynomial

`[p,~,mu] = polyfit(x,y,n)`. Same as above except that it also returns mu, a two element vector mu=[u1, u2] where u1=mean(x) and u2=std(x)

`x = fzero(fun,x0).fzero` tries to find a zero of the function fun near x0 if x0 is scalar

Loops

Loops should not be your first choice. Low performance (bad “clock times”)

Alternatives: Array operations, `find` command, Code vectorization

1. `for` loop - Primarily used if you know *a priori*(Before the fact) how many times the loop will need to run (or can calculate it)

Loops (cont)

2. `while` loop - A pre-test loop: it checks the condition before completing the iteration.The first time MATLAB sees the while loop, it checks to see if it should go into the while loop. If the condition is false, MATLAB will never go into the while loop. If the condition is true, MATLAB proceeds into the `while` loop.

3. `do while loop` – unavailable in MATLAB. Guarantees one pass through the loop.

Every `for` loop can be made into a `while` loop, but not every `while` loop can be made into a `for` loop.

Valid `for` loop indexes include scalars, vectors, and matrices.

For Loop

```
for some_index_variable = some_matrix
    some commands to be executed
end
Input: for k = [1 3 5 7]
     k
end
Output: k=1 k=3 k=5 k=7
```

Break & Continue Functions

Can use a break statement to cause the termination of the smallest enclosing `for` or `while` loop

Break & Continue Functions (cont)

Often considered bad form to use break without a good reason. It is much better to write “better” loop termination conditions.

Can use a `continue` statement to skip the rest of the loop, advancing to the next loop pass.

They should not be “go- to” techniques.

Timing Functions

`clock/etime` performs comparison between a start time and an end time

`cputime` returns CPU time (in seconds) since you started MATLAB; can use differences to do timing

`tic/toc` can be used as stopwatches; the time difference is in seconds (best way)

Issues with trying to time runs: Run times can vary from run to run depending on available RAM. Also, the OS can make adjustments to the system clock, using it for timing purposes can cause errors.

Transpose Command

Swaps rows/columns: `A(i,j)` becomes `A(j,i)`

Command (2 versions): `transpose(x)` or `x'`

Dot Product

`sum(A.*B)` (syntactically valid)

`dot(A,B)` (preferred and easier to implement)

In general, $AB \neq BA$

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

Dot Product Example

`A=[a11 a12; a21 a22]` (a^{11}, a^{12}, a^{21}, a^{22})

`B=[b11 b12; b21 b22]` (b^{11}, b^{12}, b^{21}, b^{22})

A*B=[$(a^{11}b^{11} + a^{12}b^{21})$, $(a^{11}b^{12} + a^{12}b^{22})$;... $(a^{21}b^{11} + a^{22}b^{21})$, $(a^{21}b^{12} + a^{22}b^{22})$

Cross Product

Result is a vector, always at a right angle (normal, orthogonal) to the plane defined by the two input vectors. Mathematically is a special case of a determinant whose first row comprises unit vectors. Must contain three elements

Numeric Data Types

Double-Precision Floating-Point Numbers (doubles)- MATLAB stores numeric data as doubles. Each value requires 8 bytes of space(64 bits).

Single-Precision Floating-Point Numbers (singles)- Uses half the storage space of a double, implies that they have half the storage. Each value requires 4 bytes of space(32 bits).

Complex Numbers- Can be doubles, singles, or integers. Requires twice the space of the base data types because one needs space for both the real-valued and complex- valued components.

Dingle<complex # of singles=double

Structure Arrays

Similar idea to cell arrays. Instead of using content indexing, however, each matrix is stored is assigned a location called a field (each field can be thought of like a property).

Field names are stored in order of their creation.

`L.myphrase1 = 'We are!'
L.nums = [1 4; 3 2]
L.myphrase2 = 'Penn State!'`
L= We Are! [2x2] Penn State!

Differential Equations

When specifying a derivative, use the symbol D (`Dy`). n^{th} order derivative: specify n after the symbol D (4th order derivative for y: `D4y`)

`dsolve(equation)` will result in the family of solutions to the DE with respect to the default variable

`dsolve(equation,symvar)` will result in the family of solutions to the DE with respect to the symbolic variable `symvar`

`dsolve(equation,condition1,condition2, …, conditionN, symvar)` will result in the family of solutions to the DE equation using the initial or boundary conditions `condition1, condition2, … conditionN` (conditions are written as equations), with respect to the `symvar` (if you just want the default, `t`, then omit the `symvar`)

Differential Equations (cont)

`dsolve(equation1,equation2, … equationN, condition1, condition2, … conditionN, symvar)` will result in the family of solutions to the DEs `equation1, equation2, …, equationN` using initial or boundary conditions `condition1, condition2, …, conditionN` with respect to the symbolic variable

`ode45` solves ordinary differential equations

Numerical Integration

`q=quad(function,a,b)`. Takes a function between limits a and b and numerical integrates it to within a default error of 1e-6 using a recursive adaptive Simpson quadrature

`q=quad(function,a,b,tol)`. Same as above, except you can specify the accuracy needed with `tol`.

`[q,NFE]=quad(...)`. Same choices available as above, except it also returns the number of function evaluations

`Z = trapz(Y)`. Y is the vector representing the function whose integral you want to approximate.

`Z = trapz(X,Y)`. Same as above, except that the integration will be done with respect a variable `X`

`cumtrapz` operates virtually the same as `trapz`, except that it will return the cumulative sums

Short Response(Practice Exam)

If the availability of memory is a concern, using the smallest necessary storage type is advantageous, enabling you to store more things in memory. An example discussed in class related to “classic” video games vs. modern phone apps (where it seems that apps and downloads are getting bigger(50 MB+) everyday.

A matrix is always 2D and has special mathematical properties. An array need not be 2D, has no special mathematical properties, and is merely a “holder” for data.

Character arrays must have the same number of rows in every column, and the same number of columns in every row. Cell arrays of `chars`, however, have no such restriction.

`elseif` is a case inside of an `if` selection structure; `else if` is a nested `if` selection structure inside of an `else` case of another `if` selection structure. MATLAB generally ignores white space, so `else if` is the same thing to the interpreter as the programmer had properly indented the code.

Creating a flowchart and pseudocode before attempting to create a computer program is a good idea because it gives you an opportunity to think your way through the program. A builder wouldn’t start building a house without a blueprint; it is advisable to think through your programs as well.

True/False(Practice Exam)

In general, for performance reasons it is preferable to use built-in MATLAB features such as the `find` command instead of using MATLAB loops.

If MATLAB finds a true case in a switch, it will **NOT** continue checking the other cases.

An exclusive or(`xor`) evaluates as TRUE when either A or B (but not both) are non-zero.

The ' operator and the transpose command both compute transposes, but these two techniques do not behave identically under all circumstances.

For personal computers (PCs) or laptops, chars in MATLAB are represented by their ASCII value when stored in memory.

In MATLAB, what would be result of this expression: `FALSE || (TRUE && FALSE)` Answer: False

. The default numeric data type in MATLAB is the double.

Matrix multiplication is **NOT** commutative for any square matrix.

A matrix A is invertible if its determinant is not equal to **0**.

Long Response Hints

`det(A)` takes the determinant of matrix A.

See Linear Algebra Section

`if rem(k,2)==0` Checks to see if k is divisible by 2(even)

Exam 1 Material

`logspace(start,end,interval)` Allocates numbers from start to end in evenly logarithmically spaced intervals.

`linspace(start,end,interval)` Allocates numbers from start to end in evenly linearly spaced intervals.

Potentially Useful Code

```
num_rows = 3;
num_cols = 4;
num_pages = 2;
value = 46;
A = zeros(num_rows, num_cols, num_pages); % Optional
for k = 1:num_pages % loop over # of pages
   for i = 1:num_rows % loop over # of rows
     for j = 1:num_cols % loop over # of cols
         A(i,j,k) = value;
         value = value - 2;
       end % cols
   end % rows
end % pages
disp(A)
```

More Potentially Useful Code

```
grades = load('P50.csv');
A_find = numel(find(grades>=90));
B_find = numel(find(grades >= 80 & grades < 90));
C_find = numel(find(grades >= 70 & grades < 80));
```

More Potentially Useful Code (cont)

```
failing_find = numel(find(grades < 70));
[num_rows, ~] = size(grades);
count = 0;
A_loop = 0; B_loop = 0;
C_loop = 0; failing_loop = 0;
while count < num_rows
 count = count + 1;
 if grades(count) >= 90
 A_loop = A_loop + 1;
 elseif grades(count) >= 80
 B_loop = B_loop + 1;
 elseif grades(count) >= 70
 C_loop = C_loop + 1;
 else
 failing_loop = failing_loop + 1;
 end
end
fprintf('%i A''s\n', A_find)
fprintf('%i B''s\n', B_find)
fprintf('%i C''s\n', C_find)
fprintf('%i D''s\n', failing_find)
```

Differentials

`diff(f)` calculates the symbolic first derivative of a symbolic function *F* with respect to the default independent variable

`diff(f,symvar)` calculates the symbolic first derivative of a symbolic function *F* with respect to the symbolic variable symvar (symvar has to be in single quotes if the variable does not already exist as a symbolic variable)

`diff(f,n)` calculates the symbolic nth derivative of the symbolic function *F* with respect to the default independent variable

`diff(f,symvar,n)` or `diff(f,n,symvar)` calculates the symbolic nth derivative of the symbolic function *F* with respect to the symvar

Integration

`int(f)` calculates the symbolic single integral of a symbolic function *F* with respect to the default independent variable

`int(f,symvar)` calculates the symbolic single integral of the symbolic function *F* with respect to the symbolic variable symvar

`int(f,a,b)` evaluates the results of the integral over the symbolic or numeric range [a, b] of the independent variable

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

Integration (cont)

`int(f,symvar,a,b)` calculates the symbolic single integral of a symbolic function *F* with respect to the symbolic variable symvar; evaluates the results of the integral over the symbolic or numeric range [a, b] of the independent variable

symvar has to be in single quotes if the variable does not already exist as a symbolic variable(same for differentials)

Advanced Graphics

`pcolor` command creates a pseudocolor checkerboard plot

MATLAB generally recognizes three different techniques for storing and representing images: 1. Intensity Images (“gray scale”) 2. Indexed Images 3. RGB (“true color”) images

intensity image can be created with the `imagesc` command

Can adjust the colormap of an image with the `colormap()` command

Can check image properties with the `imfinfo('image.jpg')` command

Can read in image data using `imread` and `imagesc`. Code: `X = imread('lighthouse.jpg') imagesc(X)`

Advanced Graphics (cont)

`imwrite(arrayname, colormap, 'filename.format')` manually saves an image. Four possible fields: `arrayname`: name of the MATLAB array in which the data is stored. `colormap`: the name of your colormap, if applicable. `filename`: the name you want to use to store the data. `format` is the file extension

`set(h,'PropertyName',PropertyValue,...)` where `h=plot(x,y)`

`drawnow` causes figure windows and their children to update, and flushes the system event queue

`h = animatedline()` creates an animated line without data, adding it to the current axis. Can use a loop to later add data points. Can also use the `addpoints` command.