UNIVERSITY OF CAMBRIDGE DEPARTMENT OF ENGINEERING

Matlab vectorisation tricks

Some basic tips on speeding up matlab code and on exploiting vectorisation are mentioned in the Optimisation section of our matlab page. Some of the tricks below come from the comp.soft-sys.matlab newsgroup and the Mathworks site. They are low-level and capable of delivering order-of-magnitude improvements.

I've added the author names (where known). If you have contributions, mail them to tpl@eng.cam.ac.uk.

Indexing using vectors

Many of these tricks use the fact that there are two ways of accessing matrix elements using a vector as an 'index'.

- If X and V are vectors, then X(V) is [X(V(1)), X(V(2)), ..., X(V(n))].
- If X and V are the same size and V only consists of 1s and 0s then MATLAB interprets V as a mask, and returns only the elements of X whose position corresponds to the location of a 1 in X. For example, if X is an array, then X>6 is an array the same size as X with 1s where the corresponding element in X is >6 and 0s elsewhere. This array can be used as a "mask" to select only the elements in X which are >6. Try

```
X=1:10
V=X>6
X(V)
```

or, more succinctly,

```
X=1:10
X(X>6)
```

There can be a problem if MATLAB decides to use the masking scheme when you want index addressing but such situations are rare.

Creating and manipulating Matrixes

To use the indexing ideas effectively you need to be able to create 'mask' matrices efficiently, and manipulate arrays. This requires the use of functions that you may not have used before. Some are listed here.

- Array Manipulation - look at `flipud`, `fliplr`, `rot90`, `triu` (extracts upper triangle), `tril` (extracts lower triangle), `filter`.
- Array Creation - look at `hankel` (try `hankel(1:5)`, for instance), `kron` `toeplitz`, `diag`, `sparse`, `repmat` (replicates and tiles matrices) and `find`. There's a trick for duplicating a vector of size M by 1, n times, to create a matrix of size M by N . In this method, known as Tony's Trick, the first column of the vector is indexed (or referenced, or selected) n times.

```
v = (1:5)';
X = v(:,ones(3,1))
```

Examples

These examples are short, so by reading about the functions used and testing with small matrices, you should be able to discover why they work. More than one way is shown to solve some of these questions so that you can compare methods. When big matrices are used, you might find that some methods are hundreds of times faster than others. Remember that the most elegant-looking way may not be the fastest. Use `flops` or `tic` and `toc` to assess performance.

- *Create a 1000 by 1000 vector full of 7s*
  1. X=7*ones(1000,1000);
  2. X=repmat(7,1000,1000);

  Note that the first method is more common, but the second method is 10 times faster for me.
- *How can you reverse a vector?*
  1. If X has n components, X(n:-1:1)
  2. fliplr(X)
- *How can you reverse one column of a vector*
  1. To reverse column c of a matrix M, try `M(:,c)=flipud(M(:,c))`.
- *How can you swap rows?*
  1. M([1,5],:) = M([5,1],:) interchanges rows 1 and 5 of M.
  2. More generally, if V has m components and W has n components, then M(V,W) is the m-by-n matrix formed from the elements of M whose subscripts are the elements of V and W.
- *Obtain the mean of each column. Ignore elements <=0.*

```
keepers = (x>0);
colSums = sum(x .* keepers);
counts = sum(keepers);
means = colSums ./ counts;
```

- *Vectorise*

```
for i = 1:100
  for j = 1:100
    r(i,j) = sqrt(i^2+j^2);
  end
end

[i,j]=meshgrid(1:100,1:100);
r = sqrt(i.^2+j.^2);
```

- *Vectorise the following threshold function*

```
function A = threshold(Matrix, max_value, min_value)
    [a, b] = size(Matrix);
    for i = 1:a
       for j = 1:b
              c =  Matrix(i,j);
              if ( c > max_value )
                 Matrix(i,j) = max_value;
              elseif ( c < min_value )
                 Matrix(i,j) = min_value;
              end
```

```
    end
  end
```

By Peter J. Acklam

```
A = min( max( Matrix, min_value ), max_value );
```

- *From A = [3,4,3] and B = [1,2,3,4,5,6,7,8,9,10] produce a vector C where C(1) is the sum of the first A(1) elements of B, C(2) is the sum of the next A(2) elements of B, etc.*
  By Ulrich Elsner

```
foo=cumsum(B);
C=diff([0 foo(cumsum(A))])
```

- *Find the maximum of sin(x) * cos(y) where x is 1,2..7 and y is 1,2 .. 5.*

```
x=1:7
[rx,cx]=size(x);

y=1:5
[ry,cy]=size(y);

% Use "Tony's Trick"
X=x(ones(cy,1),:)
Y=y(ones(cx, 1),:)'

% Doing max(max( sin(X).*cos(Y))) would find the max in one
% line. To know the location of the maximum too, use
XY=sin(X).*cos(Y)
[r,c]=max(XY);
[r2,c2]=max(max(XY));
sprintf('The biggest element in XY is %d at XY(%d,%d)',r2,c(c2), ...
        c2)
```

- *Count how many times the components of a matrix are repeated.*

1. ```
   bins = min(min(X)):max(max(X));
   [numTimesInMatrix, Number] = hist(X(:),bins);
   ```

2. From Mathworks - "Hist can only partially vectorize this problem, so it uses a loop as well. Therefore, the algorithm runs with order O(length(bins)). Also, this algorithm reports the number of occurrences of all numbers in bin, including those that appear zero times. To get rid of these, add this":

   ```
   ind=find(~numTimesInMatrix);
   Number(ind)=[];numTimesInMatrix(ind)=[];
   ```

3. ```
   x = sort(X(:));
   difference = diff([x;max(x)+1]);
   count = diff(find([1;difference]));
   y = x(find(difference));
   ```

Note that we use the (:) operation to ensure that x is a vector.

4. `count = sparse(1,X,1);`

(read about `sparse` if, like me, you were surprised by this)

- *Retrieve those elements that are shared in matrices A and B*
  1. intersect(A(:),B(:)) (Sijmen-de.Jong)
  2. For *positive* integers:

```
a=A(:); % make A a row
b=B(:);
m=max([a;b]);
x1=zeros(m,1);
x2=zeros(m,1);
x1(a)=ones(length(a),1);
x2(b)=ones(length(b),1);
find(x1&x2)
```

  3. For equally sized A and B:

```
a = A(:)'+1;
b = B(:)'+1;
simple = [find(sparse(1,a,1)>0);find(sparse(1,b,1)>0)];
values = find(full(sparse(1,simple',1))>1)-1
```

- *Scale all the rows of each column by the data in col 300.*

```
[nr,nc] = size(A);
B = A(:,300);
A = A ./ B( :, ones(nc,1) );
```

- *Write a function d = nearney(x, y) where x, y and d are column vectors of the same length and where d(i) is the smallest distance on a plane from the point [x(i) y(i)] to [x(j) y(j)] for all i ~= j; that is the distance from [x(i) y(i)] to its nearest neighbor.*
  1. From moler@mathworks.com - "This gets good marks for terseness, but has lousy time and space complexity"

```
function d = nearney(x,y)
[X,Y] = meshgrid(x+i*y);
d = min(abs(X-Y) + realmax*eye(length(x)))';
```

  2. From Sijmen de Jong

```
z = [x, y];
d  = sum(z'.^2);
[m,n] = size(z);
d = sqrt(min(d(ones(m,1),:)+d(ones(m,1),:)'-2*z*z'+realmax*eye(m)));
```

- *Write an m-file which will replace each element of a matrix with a 4x4 matrix of that element.*
  1. From pete@electrosystems.com

```
x=[1 2 3; 4 5 6];
```

```
y=kron(x,ones(4))
```

2.
```
A=randn(100);
A(ceil((1:(size(A,1)*4))/4),ceil((1:(size(A,2)*4))/4));
```

- *Another 'find the nearest' question Write the code to do this*

```
function out = nearest(y,x1,x2);
% function out = nearest(y,x1,x2);
% y,x1,x2, and out are row vectors of the same length
% out(i) = x1(i) or x2(i) depending on which one is closer to y(i)
%
% for example if:
%       y = [3    5    7  9    11]
%      x1 = [3.1  6    7  8.9  12]
%      x2 = [3.2  5.9  0  0    10.9]
% then out = [3.1  5.9  7  8.9  10.9]
```

1.
```
a = [x1; x2];
[c d] = min(abs([y; y] - a));
out = diag(a(d,:))';
```

2.
```
out = x1;
i2 = find(abs(x1-y) > abs(x2-y));
out(i2) = x2(i2);
```

3. More generally

```
function out = nearest(y,x)
% function out =nearest(y,x)
% y is a row-vector and x is composed of row-vectors of the same length as y.
%
% out(i)=the element of x(:,i) closest to y(i)
%
% For example if:
%       y= [3    5    7  9    11]
%       x=[[3.1  6    7  8.9  12];
%          [3.2  5.9  0  0    10.9]]
% then out=[3.1  5.9  7  8.9  10.9]
%
[c d]=min(abs(y(ones(size(x,1),1),:)-x));
out=x(d+(0:size(x,2)-1)*size(x,1));
```

- *Duplicate each element of a vector.*
  1. By Ulrich Elsner

```
rep=2;
foo=b(ones(1,rep),:);   % Tony's trick
                        % maybe use b([1 1],:)
b=foo(:)'               % reshape into row
```

- *Multiply every column of matrix X of size (1000,500) by another vector V which has dimension (1000).*

1. A non-vectorised way to do this is:

```
X2 = zeros(1000,500)
for k = 1:500
   X2(:,k) = V.*X(:,k);
end
```

2.
```
X2 = diag(V)*X;
```

3.
```
rows=1000;
columns = 500;
Gcolvec = [1:rows]'; % Be sure that G is a column vector
Gmatrix = repmat(Gcolvec,1,columns);
R = ones(rows, columns);
tic, GxR = Gmatrix.*R; toc
```

4. By marco@chinook.physics.utoronto.ca

```
Ra = R .* (G * ones(1,500));
Rb = R .* repmat(G,1,500);
Rc = R .* G(:,ones(1,500));
```

- *If you have two matrices, which together give x and y coordinates into another matrix, eg.*

```
xc = [1 2 1; 3 2 4 ];
yc = [1 1 1; 2 2 1; ];

M = [ 6  7  8  9;
     10 11 12 13];
```

produce the 2x3 matrix R which contains the values of M at the locations given in xc and yc. Thus:

```
R = [ 6  7 6;
     12 11 9];
```

1.
```
R=M(sub2ind(size(M),yc,xc));
```

2.
```
a = [1 4 3 2]
b = [2 3 1 4]
ind = sub2ind([4 4],a,b)
A = rand(4,4)
A(ind)
```

- *Create a matrix X, where each column is a shifted copy of the vector v*

```
v = (1:5)';
X = toeplitz(v, v([1,length(v):-1:2]))
```

- *Unique sort (i.e. duplicates removed)*

  -
```
x = sort(x(:));
```

```
unique(x)
```

- ```
  x = sort(x(:));
  difference = diff([x;NaN]);
  y = x(difference~=0);
  ```

  From Mathworks - "You may be wondering why we didn't use the find function, rather than ~=0 - especially since this function is specifically mentioned above. This is because the find function does not return indices for NaN elements, and the last element of difference as we defined it is a NaN."

- *Create a vector vi which is made by interleaving elements from an array v1 of length N and an array v2 of length N-1*

```
% By Peter J. Acklam
N=10000
v1 = 1:N;
v2 = 1:N-1;

   vi = zeros(1, 2*N-1);
   vi(1:2:end) = v1;
   vi(2:2:end) = v2;
% or
   vi = [ v1 ; v2 0 ];
   vi = vi(1:end-1);
```

- *A vector x contains some 0s. Create y s.t. y[i] is 0 if x[i] is 0, otherwise y[i] is log(x[i])*

```
   y=zeros(size(x));
   y(find(x))=log(x(find(x)))
```

See Also

- How Do I Vectorize My Code? (from Mathworks)

Updated: April 2002
Tim Love, tpl@eng.cam.ac.uk
```