Table of Contents


```
function [column_space, row_space, left_null_space, right_null_space] =
 fundamental_subspaces(A)

%FUNDAMENTAL_SUBSPACES: Given an mxn matrix, calculate the basis for all 4 of
 its
%fundamental subspaces
%   Column Space
%   Right Null Space
%   Row Space
%   Left Null Space
m = size(A, 1);
n = size(A, 2);
```

ROW REDUCTION AND PROCESSING

compute the reduced row echelon form of A as E_r, also get the pivot columns locations

```
[E_r, pivots] = rref(A);
E_r

% track the indices of the zero rows of E_r
zero_rows = [];

% select the nonzero rows of E_r as the basis of the row space
non_zero_rows = [];
for row_index = 1:m % loop over the rows
    if E_r(row_index, :) == zeros(1,n);
        zero_rows(end+1) = row_index;
    else
        non_zero_rows(end+1) = row_index;
    end
end
```

ROW SPACE

the row_space is spanned by the nonzero rows of E_r

```
row_space = E_r(non_zero_rows,:)'
```

COLUMN SPACE

the columns space is spanned by the equivalent columns of A that hold the pivots when reduced to E_r

```
column_space = A(:, pivots)
```

LEFT NULL SPACE

to find the basis of the left null space, we can compute the reduced row echelon form of [A I], where the dimension of I is consistent with the rows of A

```
E_r_M = rref([A eye(m)]);
left_null_space = E_r_M(zero_rows, n+1:end)'
```

NUMERIC RIGHT NULL SPACE

MATLAB has a built in way to calculate the right null space of a matrix

```
right_null_space_numerical = null(A)

% unfortunately, this doesn't seem super exact, and also seems subject to
% numerical approximation problems
% instead, lets try to get an algebraic (and therefore exact) determination
% of the null space:
```

ANALYTIC RIGHT NULL SPACE

locate the non-pivot columns of A

```
A_non_pivots = A;
% strip the pivots from A to leave only non-pivots
A_non_pivots(:, pivots) = [];

% locate the columns indexes of the non-pivot columns of A
non_pivots = [];
for col_index = 1:n
    for non_pivot_index = 1:size(A_non_pivots,2)
        if A(:, col_index) == A_non_pivots(:, non_pivot_index)
            non_pivots(end+1) = col_index;
        end
    end
end

% strip the zero rows from E_r
E_r_nonzero = E_r;
E_r_nonzero(all(A == 0,2), :) = [];

% construct an empty matrix of the appropriate dimensions (n x n)
E_r_aug = zeros(n);

% construct an empty matrix of the appropriate dimensions (n x (r-n))
I_aug = zeros(n,size(non_pivots, 2));
```

```
% define some counters we will use to iterate over pivots/non-pivots
row_counter = 1;
I_column_counter = 1;

% loop over columns to select appropriate rows...
while row_counter <= n
    % columns are either pivots, in which case we add the appropriate row
    if find(pivots == row_counter)
        E_r_aug(row_counter, :) = E_r_nonzero(1, :);
        E_r_nonzero(1, :) = []; % remove added row, simplifies indexing
    end
    % or non-pivots, in which case we add an "identity" equation (x_y =
    % x_y)
    if find(non_pivots == row_counter)
        E_r_aug(row_counter, row_counter) = 1;
        I_aug(row_counter, I_column_counter) = 1;
        I_column_counter = I_column_counter + 1;
    end
    row_counter = row_counter + 1;
end

% concat the augmented E_r matrix with the augmented I matrix, row reduce
E_r_rns = rref([E_r_aug I_aug]);

% take the last r - n columns of the concatenated matrix
right_null_space = E_r_rns(:, end - size(non_pivots, 2) + 1:end)
```

CONFIRMATION VIA DIMENSIONS

do some final checking to ensure all dimensions agree

```
if m ~= size(row_space, 2) + size(left_null_space, 2)
    fprintf('something is off, the row subspace dimensions do not agree with
 the matrix dimensions')
end

if n ~= size(column_space, 2) + size(right_null_space, 2)
    printf('something is off, the column subspace dimensions do not agree with
 the matrix dimensions')
end

end
```

Published with MATLAB® R2023a