1. **move\_me.m**

function w = move\_me(v,a)

if nargin<2,

a = 0;

end

w = [v(v ~= a) v(v == a)];

end

OR

function w = move\_me(v,a)

if nargin < 2

a = 0;

end

w = [];

vv = [];

for ii = 1:length(v)

if v(ii) ~= a

w(end+1) = v(ii);

else

vv(end+1) = a;

end

end

w = [w vv];

end

1. **half\_sum.m**

function h = halfsum(A)

% there is a built-in MATLAB function for almost anything...

h = sum(sum(triu(flipud(A))))

end

OR

function s = halfsum(A)

[r c] = size(A);

for ii = 1:r

for jj = 1:c

if ii < r-jj+1

A(ii,jj) = 0;

end

end

end

s = sum(A(:));

end

1. **small\_elements.m**

function I = small\_elements(X)

% The matrix multiplication of a column vector of the row numbers and

% a row vector of the column numbers gives the indices of the matrix.

[r,c] = size(X);

[x,y] = find(X < ((1:r)' \* (1:c)));

I = [x(:), y(:)];

end

OR

function found = small\_elements(A)

[row col] = size(A);

found = [];

for jj = 1:col

for ii = 1:row

if A(ii,jj) < ii \* jj

found = [found; ii jj];

end

end

end

end

1. **approximate\_e.m**

function [a,k] = approximate\_e (delta)

e = exp(1); % reference value for e

f = 1; % first factorial term

a = 1; % first series term

k = 0; % term subscript

while abs(a-e)>delta

k = k+1;

f = f\*k;

a = a+1/f;

end

end

1. **spiral\_diag\_sum.m**

function s = spiral\_diag\_sum(n)

% The last element is always n square and it is in the corner.

% A simple arithmetic expression gives the sum of the four corners.

% Do this for a series of odd numbers from 3 to n.

% Add 1 for the starting value in the center

v = 3:2:n;

s = 1 + sum(4\*v.^2 - 6\*(v-1));

end

1. **triangle\_wave.m**

function tri = sub\_triangle\_wave(n)

t = 0:pi/250:4\*pi;

tri = zeros(1,length(t));

sign = 1;

for ii = 1:2:2\*n+1

tri = tri + sign \* sin(ii\*t)/ii^2;

sign = -sign;

end

end

1. **max\_product.m**

function [mx ind] = max\_product(v,n)

ind = -1;

if isempty(v) || n > length(v)

mx = 0;

return;

end

mx = -Inf;

for ii = 1:length(v)-n+1

mul = prod(v(ii:ii+n-1));

if mul > mx

mx = mul;

ind = ii;

end

end

end

OR

function [lp,li] = max\_product (V,n)

e = length(V);

if n>e

lp = 0;

li = -1;

else

P = -Inf(1,e-n+1); % preallocate product vector

for i = 1:e-n+1

P(i) = prod(V(i:i+n-1)); % populate product vector

end

[lp,li] = max(P); % return maximum product and its location

end

end

1. **pendulum.m**

function T = pendulum(L,angle0)

T = 0;

if L > 0

dt = 1e-6;

g = 9.8;

angle = abs(angle0);

omega = 0;

T = 0;

while angle > 0

a = g\*sin(angle)/L;

omega = omega + dt \* a;

angle = angle - dt \* omega;

T = T + dt;

end

T = T \* 4;

end

end