```matlab
% Motor Unit Based Muscle Fatigue Model by Jim Potvin & Andrew Fuglevand
% front end (rested size-principle) based on Fuglevand, Winter & Patla (1993)
% last updated 2017-05-21 by Jim Potvin

%clear all
clc
clf('reset')   %clears all graphics

%% Model input parameters
    nu = 120;           % number of neurons (ie. motor units) in the modeled pool ("n")

    samprate = 10;      % sample rate (10 Hz is suggested)
    res = 100;          % resolution of activations (set = 10 for 0.1 activation resolution, 100 for 0.01)
    hop = 20;           % allows for hopping through activations to more rapidly find that
                        % which meets the threshold (10 means every 1/10th of maxact)
    r = 50;             % range of recruitment thresholds (30 or 50)

    fat = 180;          % range of fatigue rates across the motor units (300 best)
    FatFac = 0.0225;    % fatigue factor (FF/S) percent of peak force of MU per second

    tau = 22;           % 22 based on Revill & Fuglevand (2011)
    adaptSF = 0.67;     % 0.67 from Revill & Fuglevand (2011)
    ctSF = 0.379;       % 0.379 based on Shields et al (1997)

    mthr = 1;           % minimum recruitment threshold
    a = 1;              % recruitment gain paramter (1)
    minfr = 8;          % minimum firing rate (8)
    pfr1 = 35;          % peak firing rate of first recruited motor unit (35)
    pfrL = 25;          % peak firing rate of last recruited motor unit (25)
    mir = 1;            % slope of the firing rate increase vs excitation (1)
    rp = 100;           % range of twitch tensions (100)
    rt = 3;             % range of contraction times (3 fold)
    tL = 90;            % longest contraction time (90)


%% Various methods to create, or read in, force (%MVC)time-histories

%     % Create isotonic data -----------------------------------

        fthscale = 0.50             % sets %MVC level for the trial duration (100% MVC is 1.00)
        con = '0.50';               % for output file names
        fthtime = 100;              % duration to run trial (seconds)

        fthsamp = fthtime * samprate;
        fth = zeros(1, fthsamp);
        for z = 1:fthsamp
            fth(z) = fthscale;
        end

    %% Create Ramp Plateau data -----------------------------------

%         con = 'Plateaus'
%         yMAXforce = 35
%         ondur = 32;
%         mag = 0.20
%         frame = 0;
```

```matlab
%         cyc = ondur * samprate          % duration of each plateau
%         transition = 5 * samprate       % duration of transition between plateaus
%         for n = 1:cyc
%             frame = frame + 1;
%             fth(frame) = mag * 1;
%         end
%         for n = 1:transition
%             frame = frame + 1;
%             fth(frame) = (mag * 1) + (mag * n / transition);
%         end
%         for n = 1:cyc
%             frame = frame + 1;
%             fth(frame) = mag * 2;
%         end
%         for n = 1:transition
%             frame = frame + 1;
%             fth(frame) = (mag * 2) + (mag * n / transition);
%         end
%         for n = 1:cyc
%             frame = frame + 1;
%             fth(frame) = mag * 3;
%         end
%         fthsamp = frame

%% Calculations from the Fuglevand, Winter & Patla (1993) Model

    ns = 1:1:fthsamp;   % array of samples for fth
    fth = fth(ns);

    % Recruitment Threshold Excitation (thr)
        thr = zeros(1, nu);
        n = 1:1:nu;
        b = log(r + (1-mthr)) / (nu-1);             % this was modified from Fuglevand et al (1993) RTE(i) equation (1)
        for i = 1:nu                                % as that did not create the exact range of RTEs (ie. 'r') entered
            thr(i) = a * exp((i-1) * b) - (1 - mthr);
        end

    % Peak Firing Rate (frp)
        % modified from Fuglevand et al (1993) PFR equation (5) to remove thr(1) before ratio
        frdiff = pfr1-pfrL;
        frp = pfr1 - frdiff*((thr(n) - thr(1))/(r - thr(1)));
        maxex = thr(nu) + (pfrL - minfr)/mir;       % maximum excitation
        maxact = round(maxex * res);                % max excitation x resolution
        ulr = 100 * thr(nu)/maxex;                  % recruitment threshold (%) of last motor unit

    % Calculation of the rested motor unit twitch properties (these will change with fatigue)

        % Firing Rates for each MU with increased excitation (act)
            mufr = zeros(nu, maxact);
            for mu = 1:nu
                for act = 1:maxact
                    acti = act/res;
                    if acti >= thr(mu)
                        mufr(mu, act) = mir * (acti - thr(mu)) + minfr;
                        if mufr(mu, act) > frp(mu)
                            mufr(mu, act) = frp(mu);
```

```
            end
          elseif acti < thr(mu)
            mufr(mu, act) = 0;
          end
        end
    end

    k = 1:1:maxact;  %range of excitation levels

% Twitch peak force (P)
    b = log(rp)/(nu-1);                  % this was modified from Fuglevand et al (1993) P(i) equation (13)
    P = exp(b * (n-1));                  % as that didn't create the exact range of twitch tensions (ie. 'rp') entered

% Twitch contraction time (ct)
    c = log(rp)/log(rt);                 % scale factor
    ct = zeros(1,nu);
    for mu = 1:nu                        % assigns contraction times to each motor unit (moved into loop)
       ct(mu) = tL * (1/P(mu))^(1/c);
    end

% Normalized motor unit firing rates (nmufr) with increased excitation (act)
    nmufr = zeros(nu, maxact);
    for mu = 1:nu
        for act = 1:maxact
            nmufr(mu, act) = ct(mu) * (mufr(mu, act) / 1000);  % same as CT / ISI
        end
    end

% Motor unit force, relative to full fusion (Pr) with increasing excitation
    % based on Figure 2 of Fuglevand et al (1993)
    sPr = 1 - exp(-2 * (0.4^3));
    Pr = zeros(nu, maxact);
    for mu = 1:nu
        for act = 1:maxact
            if nmufr(mu,act) <=0.4                        %linear portion of curve
                Pr(mu,act) = nmufr(mu,act)/0.4 * sPr;   %Pr = MU force relative to rest 100% max excitation of 67
            end
            if nmufr(mu,act) > 0.4
                Pr(mu,act) = 1 - exp(-2 * (nmufr(mu,act)^3));
            end
        end
    end

% Motor unit force (muP) with increased excitation
    muP = zeros(nu, maxact);
    for mu = 1:nu
        for act = 1:maxact
           muP(mu,act) = Pr(mu,act) * P(mu);
        end
    end
    totalP = sum(muP,1);                                  % sum of forces across MUs for each excitation (dim 1)
    maxP = totalP(maxact);

% Total Force across all motor units when rested
    Pnow = zeros(nu, fthsamp);
    Pnow(:,1) = P(:);
```

```
%% Calculation of Fatigue Parameters (recovery currently set to zero in this version)

    % note, if rp = 100 & fat = 180, there will be a 100 x 180 = 1800-fold difference in
    % the absolute fatigue of the highest threshold vs the lowest threshold.
    % The highest threshold MU will only achieve ~57% of its maximum (at 25 Hz), so the actual range of fatigue
    % rates is 1800 x 0.57 = 1026

% fatigue rate for each motor unit  (note: "log" means "ln" in Matlab)
    b2 = log(fat)/(nu-1);
    mufatrate = exp(b2 * (n-1));

    b3 = log(rec)/(nu-1);
    murecrate = exp(b3 * (n-1));

    fatigue = zeros(1,nu);
    recovery = zeros(1,nu);
    for mu = 1:nu
         fatigue(mu) = mufatrate(mu) * (FatFac / fat) * P(mu);
            % the full fatigue rate is mufatrate(mu) * [FatFac / fat] * Pr(mu,act) * P(mu)
            % the only variable is the relative force: Pr(mu,act), so this part is calculated once here
         recovery(mu) = 0;   %set to zero for now
    end

% Establishing the rested excitation required for each target load level (if 0.1% resolution, then 0.1% to 100%)
    startact = zeros(1, 100);
    for force = 1:100
        startact(force) = 0;    % excitation will never be lower than that needed at rest for a given force
        for act = 1:maxact      % so it speeds the search up by starting at this excitation
            if (totalP(act)/maxP * 100) < force
                startact(force) = act - 1;
            end
        end
    end

    Pchangecurves = zeros(nu, maxact);
    for act = 1:maxact
        for mu = 1:nu
            Pchangecurves(mu,act) = fatigue(mu) * Pr(mu, act) * P(mu);  % just used for graphical display
        end
    end

    mes = 'start of fatigue analysis'

%% Moving through force time-history and determing the excitation required to meet the target force at each time

    TmuPinstant = zeros(nu, fthsamp);
    m = zeros(1, fthsamp);
    mufrFAT = zeros(nu, fthsamp);
    ctFAT = zeros(nu, fthsamp);
    ctREL = zeros(nu, fthsamp);
    nmufrFAT = zeros(nu, fthsamp);
    PrFAT = zeros(nu, fthsamp);
    muPt = zeros(nu, fthsamp);
    TPt = zeros(nu, fthsamp);
```

```
Ptarget = zeros(1, fthsamp);
Tact = zeros(1, fthsamp);
Pchange = zeros(nu, fthsamp);
TPtMAX = zeros(1, fthsamp);
muPtMAX = zeros(nu, fthsamp);
muON = zeros(nu);
adaptFR = zeros(nu,fthsamp);
Rdur = zeros(1,nu);
acttemp = zeros(fthsamp, maxact);
muPna = zeros(nu, fthsamp);
muForceCapacityRel = zeros(nu,fthsamp);
timer = 0;


for i = 1:fthsamp
    if i == (timer + 1) * samprate * 60         % shows a timer value every 15 seconds
        timer = timer + 1;
        current = i / samprate
    end

    force = round(fth(i) * 100) + 1;            % used to start at the minimum possible excitation (lowest it can be is 1)
    if force > 100                                % so start with excitation needed for fth(i) when rested (won't be lower than this)
        force = 100;
    end
    s = startact(force) - (5 * res);            % starts a little below the minimum
    if s < 1
        s = 1;
    end

    acthop = round(maxact / hop);               % resets 'acthop' to larger value for new sample
    act = s;                                    % start at lowest value then start jumping by 'acthop'
    for a = 1:maxact                            % this starts at the mimimum (s) then searches for excitation required to meet the target
        acttemp(i,a) = act;
        for mu = 1:nu

            % MU firing rate adaptation modified from Revill & Fuglevand (2011)
            % this was modified to directly calculate the firing rate adaption, as 1 unit change in excitation
            % causes 1 unit change in firing rate
            % scaled to the mu threshold (highest adaptation for hightest threshold mu)

            if muON(mu) > 0
                Rdur(mu) = (i - muON(mu) + 1)/samprate;                             % duration since mu was recruited at muON
            end
            if Rdur(mu) < 0
                Rdur(mu) = 0;
            end

            adaptFR(mu,i) = (thr(mu)-1)/(thr(nu)-1) * adaptSF * (mufr(mu,act) - minfr + 2) * (1 - exp(-1 * Rdur(mu) / tau ));
                if adaptFR(mu,i) < 0                                    % firing rate adaptation
                    adaptFR(mu,i) = 0;
                end

            mufrFAT(mu,i) = mufr(mu,act) - adaptFR(mu,i);               % adapted motor unit firing rate: based on time since recruitment
                  mufrMAX = mufr(mu,maxact) - adaptFR(mu,i);            % adapted FR at max excitation

            ctFAT(mu,i) = ct(mu) * (1 + ctSF * (1 - Pnow(mu,i)/P(mu)));  % corrected contraction time: based on MU fatigue
```

```
            ctREL(mu,i) = ctFAT(mu,i)/ct(mu);

        nmufrFAT(mu,i) = ctFAT(mu,i) * (mufrFAT(mu, i) / 1000);                % adapted normalized Stimulus Rate (CT * FR)
            nmufrMAX = ctFAT(mu,i) * (mufrMAX / 1000);                         % normalized FR at max excitation

        if nmufrFAT(mu,i) <=0.4                                                % fusion level at adapted firing rate
            PrFAT(mu,i) = nmufrFAT(mu,i) / 0.4 * sPr;                          %   linear portion of curve
        end
        if nmufrFAT(mu,i) > 0.4                                                %   non-linear portion of curve
            PrFAT(mu,i) = 1 - exp(-2 * (nmufrFAT(mu,i)^3));
        end
        muPt(mu, i) = PrFAT(mu, i) * Pnow(mu, i);                              % MU force at the current time (muPt):
                                                                               % based on adapted postion on fusion curve

            if nmufrMAX <=0.4                                                  % fusion force at 100% maximum excitation
                PrMAX = nmufrMAX / 0.4 * sPr;
            end
            if nmufrMAX > 0.4
                PrMAX = 1 - exp(-2 * (nmufrMAX^3));
            end
            muPtMAX(mu, i) = PrMAX * Pnow(mu, i);                              % Max MU force capacity at the current time

    end % next motor unit (mu)

    TPt(i) = sum(muPt(:,i))/maxP;                                              % total sum of MU forces at the current time (TPt)
    TPtMAX(i) = sum(muPtMAX(:,i))/maxP;

    % used to speed up the search for the right excitation to meet the current target
    if TPt(i) < fth(i) && act == maxact
        break;
    end
    if TPt(i) < fth(i)
        act = act + acthop;
        if act > maxact
            act = maxact;
        end
    end
    if TPt(i) >= fth(i) && acthop == 1
        break;  % stop searching as the correct excitation is found
    end
    if TPt(i) >= fth(i) && acthop > 1
        act = act - (acthop - 1);  % if the last large jump was too much, it goes back and starts increasing by 1
        if act < 1
            act = 1;
        end
        acthop = 1;
    end

end % next excitation (act)

for mu = 1:nu
    if muON(mu) == 0 && (act/res) >= thr(mu)         % can be modified to reset if the MU turns off
        muON(mu) = i;                                       % time of onset of mu recruitment (s)
    end
end
```

```
        Ptarget(i) = TPt(i);        % modeled force level ?? do I need to do this, or can I just use TPt(i)
        Tact(i) = act;              % descending (not adapted) excitation required to meet the target force at the current time

        % Calculating the fatigue (force loss) for each motor unit

        for mu = 1:nu
            if  mufrFAT(mu, i) >= recminfr                                          % Force loss of each MU for each interval
                Pchange(mu,i) = -1 * (fatigue(mu) / samprate) * PrFAT(mu, i);       % based on % of MU fusion force
            elseif mufrFAT(mu, i) < recminfr
                Pchange(mu,i) = recovery(mu) / samprate;
            end

            if i < 2
                Pnow(mu, i+1) = P(mu);
                   % Pnow(mu, i+1) = 0;  % Use this to start the muscle fully exhausted
            elseif i >=2
                Pnow(mu, i+1) = Pnow(mu, i) + Pchange(mu,i);                            % instantaneous strength of MU
            end                                                                         % right now without adaptation

            if Pnow(mu, i+1) >= P(mu)
               Pnow(mu, i+1) = P(mu);                                                   % does not let it increase past rested strength
            end

            if Pnow(mu, i+1) < 0
               Pnow(mu, i+1) = 0;                                                       % does not let it fatigue below zero

            end

        end % next motor unit

    end % next fthsamp

    Tstrength = zeros(1, fthsamp);
    for i = 1:fthsamp
        for mu = 1:nu
            muPna(mu,i) = Pnow(mu,i) * muP(mu,maxact) / P(mu);                          % non-adapted MU max force at 100% excitation
(muPna)
        end
        Tstrength(i) = sum(muPna(:,i)) / maxP;                                          % Current total strength without adaptation relative to
max rested capacity
    end
    for i = 1:fthsamp
        endurtime = i / samprate;
        if TPtMAX(i) < fth(i)
            break;
        end
    end

clf('reset')   %clears all graphics

%% Output

EndStrength = (TPtMAX(fthsamp) * 100);

endurtime
```

```
for mu = 0:mujump:nu
    if mu == 0
        mu = 1;
    end
    muForceCapacityRel(mu,ns) = Pnow(mu,ns)*100/P(mu);  % for outputs below
end

hold off;
   combo = [ns(:)/samprate, fth(:), Tact(:)/res/maxex * 100,Tstrength(:) * 100 ,Ptarget(:) * 100,TPtMAX(:)* 100];
   dlmwrite(strcat(con,' A - Target - Act - Strength (no adapt) - Force - Strength (w adapt).csv'), combo)

   dlmwrite(strcat(con,' B - Firing Rate.csv'),transpose(mufrFAT(:,:)))
   dlmwrite(strcat(con,' C - Individual MU Force Time-History.csv'), transpose(muPt(:,:)))
   dlmwrite(strcat(con,' D - MU Capacity - relative.csv'),transpose(muForceCapacityRel(:,:)))

beep;
```