Appendix A

1.0 Technical details of the surrogate generation based on the multivariate empirical mode decompositions

The MEMD surrogate generation method has the following four steps (see Fig. A1):

Step 1: Decompose the reconstructed gait dynamics, defined by Eq. 7 of the main text, into intrinsic mode functions by MEMD

Step 2: Consider the sum of high frequency components, \( d_1(t) \) to \( d_4(t) \), as the intra-step details \( y_{\text{high}}(t) \) and the sum of low frequency components, \( d_5(t) \) to \( d_N(t) \) together with the final residual \( r_N(t) \), as the inter-step periodicity \( y_{\text{low}}(t) \) of the gait dynamics (Fig. A1):

\[
\begin{align*}
  y_{\text{high}}(t) &= \sum_{n=1}^{4} d_n(t) \\
  y_{\text{low}}(t) &= \sum_{n=5}^{N} d_n(t) + r_N(t)
\end{align*}
\]

where the original gait dynamics \( x(t) = y_{\text{high}}(t) + y_{\text{low}}(t) \).

Step 3: IAAFT surrogates were generated for the intra-step details, \( y_{\text{high}}(t) \).

Step 4: MEMD surrogate dynamics was obtained by adding the IAAFT surrogates and the inter-step periodicity, \( y_{\text{low}}(t) \).
Fig. A1: Schematic representation of the generation of MEMD surrogates. The original gait dynamics in the reconstructed state space (see upper panel) are decomposed into multiple intrinsic mode functions (IMF) (see middle small panels). The high frequency intra-step details of the gait dynamics are defined as the sum of the first four IMFs (left panels) and low frequency periodicity of the gait dynamics are defined by sum of the remaining IMFs (right panels). A IAAFT surrogate (lower right panel) are generated for the high frequency intra-step details which preserves the distribution and power spectral density, but removes the phase-dependent
changes in these details. The MEMD surrogate is defined as the sum of the IAAFT surrogate and the low frequency periodicity of the gait dynamics (lower left panel).

1.1 Multivariate empirical mode decomposition (MEMD)

The following MEMD algorithm introduced by Rehman and Mandic (2010) was used in the present study (see acknowledgment):

**Step 1:** Generate a Hammersley sequence-based point set on a $3m – 1$ dimensional sphere where $m$ is the number of lags in the state space reconstruction method ($m = 1, 2, \text{ and } 3$ in the present study).

**Step 2:** Compute the projection $p_{\theta_k}(t)$ of the gait dynamics $x(t) = (or \text{ residual } r(t) \text{ or } d(t)$ for iterative steps) along the unit direction vectors $\theta_k$ of the $3m – 1$ dimensional sphere.

**Step 3:** Find the time instant $t_{\theta_k}$ that corresponding to the maxima $p_{\theta_k}^{\text{max}}(t)$ of $p_{\theta_k}(t)$ along all $k = 1, 2, \ldots, 3m – 1$ dimensions.

**Step 4:** Obtain the envelope curves, $e_{\theta_k}(t)$, by component-wise spline interpolations between all time instants $t_{\theta_k}$ of $p_{\theta_k}^{\text{max}}(t)$.

**Step 5:** Compute the mean $m(t)$ of all envelope curves, $e_{\theta_k}(t)$, across all $3m – 1$ directions of the sphere by the following equation:

$$m(t) = \frac{1}{3m-1} \sum_{k=1}^{3m-1} e_{\theta_k}(t) \quad (A6)$$

**Step 6:** The first series of details $d_1(t)$ around the mean $m_1(t)$ is defined as $d_1(t) = x(t) – m_1(t)$. If $d_1(t)$ satisfies the selected stopping criteria, then $d_1(t)$ is defined as an intrinsic mode.
function (IMF) and Step 2 to 5 is performed on first residual, \( r_1(t) = x(t) - d_1(t) \). The second IMF is defined as \( d_2(t) = r_1(t) - m_2(t) \) with residual \( r_2(t) = r_1(t) - d_2(t) \). Consequently, the \( n \)th IMF is defined as \( d_n(t) = r_{n-1}(t) - m_n(t) \) with residual \( r_n(t) = r_{n-1}(t) - d_n(t) \). This iterative shifting procedure (i.e., Step 2 to 5) is continued until two maxima \( p^{\text{max}}(t) \) of the projection \( p_{\theta_k}(t) \) in Step 3 can no longer be found. If \( d_n(t) \) do not satisfy the stopping criteria, then step 2 to 5 is performed as an iterative procedure on \( d_n(t) \) until the stopping criteria is met and an IMF is defined. The stopping criteria used in the present study is similar to the stopping criteria proposed by Rilling et al. (2003) except the criteria of equality between the number of zero crossings and number of maxima. The sum of all IMFs and the final residual,
\[
x(t) = \sum_{n=1}^{N} d_n(t) + r_N(t)
\]
becomes the gait dynamics \( x(t) \).

### 1.2 Iterated amplitude adjusted fourier transform (IAAFT)

IAAFT surrogate was generated for the intra-step details of the gait dynamics, \( y_{\text{high}}(t) \), defined by Eq. A1 by following five steps (Schreiber and Schmitz, 1996):

**Step 1:** Store the rank ordering of the amplitudes of \( y_{\text{high}}(t) \).

**Step 2:** Fast fourier transform \( y_{\text{high}}(t) \) and store the obtained spectral amplitudes.

**Step 3:** Randomly shuffle \( y_{\text{high}}(t) \).

**Step 4:** Initiate an iterative procedure where:

1) The spectral amplitudes of shuffled time series are obtained by a fast Fourier transformation.
2) The obtained spectral amplitudes of the shuffled time series are substituted with the stored spectral amplitudes of $y_{\text{high}}(t)$.

3) The surrogate series are obtained by an inverse fast Fourier transformation.

4) The amplitudes of obtained surrogate series are ranked as in $y_{\text{high}}(t)$.

In the present study, 1 to 4 is iterated a maximum of 500 times to obtain a saturation effect as described by Schreiber and Schmitz (1996) where the surrogate series had equal power spectral density and probability density function as $y_{\text{high}}(t)$, but where the phase-dependent changes in $y_{\text{high}}(t)$ are removed.
Appendix B

Matlab code for numerical estimation of fractional stability

The following Matlab function, frac_stab, estimate \( \alpha, \beta, \lambda_f \) and \( C \) of fractional stability defined by Eq. 5. The Matlab code is based on, div_frac_calc, used to calculate reaction size \( d(t) \) in Rispens et al. (2015) and \textit{mlf} developed by Podlubney which computes the generalized Mittag-Leffler function in Eq. 6. \textit{mlf} and is available at

https://www.mathworks.com/matlabcentral/fileexchange/8738-mittag-leffler-function/content/mlf.m

\textit{mlffit2} developed by Podlubney et al. (2012) could be used instead of \textit{gml_stability_fit}. \textit{mlffit2} is available at

http://www.mathworks.com/matlabcentral/fileexchange/32170-fitting-data-using-the-mittag-leffler-function/content/mlffit2.m

```matlab
function [alfa,beta,lambda_f,C,dt_upsamp,dt_fit,t]=frac_stab(Acc_all,mean_step_time,lag_size,lag_numb,singular_numb,fs,Fig)

%Estimation of fractional stability defined by Eq. 5 in Ihlen et al. %{(submitted)

%INPUT PARAMETERS--------------------------------------------------------
%Acc_all_________[n,3] matrix where Acc_all(:,1) are acceleration %
in AP direction, Acc_all(:,2) the same in ML %
direction and Acc_all(:,3) the same in V direction. %
mean_step_time____Mean step time for Acc_all or the mean periodicity %
of the signal. %
lag_size___________Chosen lag size in number of samples for the state %
space reconstruction %
lag_numb___________Number of lag used in the delayed coordinate %
embedding %
singular_numb________Number of singularities in d(t) to be used in the %
estimation of fractional stability in Eq. 5 %
fs__________Sampling frequency %
Fig________________0/1-flag where 1 displays the fit of the fractional %
stability (Eq. 5) %

%OUTPUT PARAMETERS-------------------------------------------------------
%alfa________________Vector with m indices where m = singular_numb %
beta________________Vector with m indices where m = singular_numb %
lambda_f___________Vector with m indices where m = singular_numb %
C___________________Vector with m indices where m = singular_numb %
dt_upsamp__________Upsampled version of d(t) between singularities.
```
% [n,m] matrix where n is sample number and m is the number of singularities.
% The fitted fractional stability (Eq. 5) to d(t).
% Same [n,m] matrix as a above
% Time indexes for dt_upsamp and dt_fit. Same [n,m] matrix as above.

% Reconstruction of state space defined by lag- numb and lag_size.
The dynamics are reconstructed for the AP, ML, and V direction combined---

if singular_numb>lag_numb,
    disp('singular_numb should be equal or lower than lag_numb')
end

X_ap=Acc_all(:,1);
X_ml=Acc_all(:,2);
X_v=Acc_all(:,3);

N_ss = size(X_ap,1)-(lag_numb-1)*lag_size;
state_ap=nan(N_ss,lag_numb*size(X_ap,2));
state_ml=nan(N_ss,lag_numb*size(X_ml,2));
state_v=nan(N_ss,lag_numb*size(X_v,2));

for dim=1:size(X_ap,2),
    for delay=1:lag_numb,
        state_ap(:,(dim-1)*lag_numb+delay)=X_ap((1:N_ss)'+(delay-1)*lag_size,dim);
        state_ml(:,(dim-1)*lag_numb+delay)=X_ml((1:N_ss)'+(delay-1)*lag_size,dim);
        state_v(:,(dim-1)*lag_numb+delay)=X_v((1:N_ss)'+(delay-1)*lag_size,dim);
    end
end

state_space=[state_ap,state_ml,state_v];

% Calculation of reaction distance d(t) by Rosenstein et al. (1993) method---
[dt]=div_frac_calc(state_space,mean_step_time,fs);

% Identification of singularities in d(t)-----------------------------
[pks,locs0]=findpeaks(-dt,'MinPeakDistance',lag_size-2);
locs=[1,locs0];

% Resampling and estimation of fractional stability (Eq. 5) between each singularity
for n=1:singular_numb,
    dt_upsamp(:,n)=interp1(1:length(locs(n):locs(n+1)),
        dt(locs(n):locs(n+1)),linspace(1,length(locs(n):locs(n+1)),50),'pchip');
    t1=linspace(1,length(locs(n):locs(n+1)),50)./fs;
    t(:,n)=linspace(locs(n),locs(n+1),50)./fs;
    par=[2; 2; dt_upsamp(1,n); -1];
    [c0,dt_fit0]=gml_stability_fit(dt_upsamp(:,n)',t1,par);
    dt_fit(:,n)=dt_fit0;
    alfa(n)=c0(1);
    beta(n)=c0(2);
    lambda_f(n)=c0(4);
    C(n)=c0(3);
end
%Figure output-----------------------------------------------
if Fig==1,
    figure;
    for n=1:singular_numb,
        plot(t(:,n),dt_upsamp(:,n),'bo-');hold all
        plot(t(:,n),dt_fit(:,n),'r-','LineWidth',2);
    end
end
%-------------------------------------------------------------------

function [d_t]=div_frac_calc(state_space,mean_step_time,fs)

%computation of reaction size d(t) according to method proposed by
%Rosenstein et al. (1993). The script is a modified version of a script
%div_calc developed by Sietse Rispens used in Rispens et al. (2015).

period_sec = mean_step_time/fs;
WindowLen = floor(mean_step_time);
ws_sec=WindowLen/fs;

[m,n]=size(state_space);
ws=round(ws_sec*fs);
period=round(period_sec*fs);
mcompletewindow = m-ws+1;
statecw = state_space(1:mcompletewindow,:);
divergence_sum=zeros(1,ws);
divergence_count=zeros(1,ws);
diff_state = statecw*0;
diff_state_sqr = diff_state;
diff_total = zeros(size(diff_state,1),1);
for i = 1:mcompletewindow
    if ~isnan(state_space(i,:))
        start=round(max([1,i-period+1]));
        stop=round(min([mcompletewindow,i+period-1]));
        for j=1:n,
            diff_state(:,j) = statecw(:,j)-statecw(i,j);
            diff_state_sqr(:,j)=diff_state(:,j).^2;
        end
        diff_total(:,1)=sum(diff_state_sqr,2);
        diff_total(start:stop,1)=NaN;
        [mini,index]=min(diff_total);
        if i+ws>m || index+ws>m
            % do not use these data
        else
            divergence_sum = divergence_sum + sqrt(sum((state_space(i:i+ws-1,:) - state_space(index:index+ws-1,:)).^2,2))';
            divergence_count = divergence_count + 1;
        end
    end
end

% do not use these data
end

d_t = (divergence_sum./divergence_count);
function [par_fit,d_fit]=gml_stability_fit(d,t,par0)

fun = @(par,t)gml_stability(par,t);
lb=[0,0,0,-100000];
ub=[5,3,1000,100000];
options = optimoptions(@lsqcurvefit,'Algorithm','trust-region-reflective','TolFun',1.0000e-14,'MaxFunEvals',3000,'MaxIter',3000,'Display','none');
par_fit = lsqcurvefit(fun,par0,t,d,lb,ub,options);
d_fit=gml_stability(par_fit,t);

function d=gml_stability(para,t)

d=para(3)* t.^(para(2)-1) .* mlf(para(1), para(2), para(4)*t.^para(1),6);