2022

Experimental And Computational Analyses Of Locomotor Rhythm Generation And Modulation In Caenorhabditis Elegans

Hongfei Ji
University of Pennsylvania

Experimental And Computational Analyses Of Locomotor Rhythm Generation And Modulation In Caenorhabditis Elegans

Abstract

Neural circuits coordinate with muscles and sensory feedback to generate motor behaviors appropriate to its natural environment. Studying mechanisms underlying complex organism locomotion has been challenging, partly due to the complexity of their nervous systems. Here, I used the roundworm C. elegans to understand the locomotor circuit. With its well-mapped nervous system, easily-measurable movements, genetic manipulability, and many human homologous genes, C. elegans has been commonly used as a model organism for dissecting the circuit, cellular, and molecular principles of locomotion. My work introduces two separate approaches to probe the mechanisms by which the C. elegans motor circuit generates and modulates undulations. First, I quantified C. elegans movements during free locomotion and during transient muscle inhibition. Undulations were asymmetrical with respect to the duration of bending and unbending per cycle. Phase response curves induced by brief optogenetic head muscle inhibitions showed gradual increases and rapid decreases as a function of phase at which the perturbation was applied. A relaxation oscillator model was developed based on proprioceptive thresholds that switch the active muscle moment. It quantitatively agrees with data from free movement, phase responses, and previous results for gait adaptation to mechanical loads. Next, I characterized a proprioception-mediated compensatory behavior during C. elegans forward locomotion: the anterior body bending amplitude compensates for the change in midbody bending amplitude by an opposing homeostatic response. I demonstrated that curvature compensation requires dopamine signaling driven by PDE neurons. Calcium imaging experiments suggested a proprioceptive functionality for PDE in sensing midbody curvature. Downstream of PDE dopamine signaling, curvature compensation requires D2-like dopamine receptor DOP-3 in the interneurons AVK. FMRFamide-like neuropeptide FLP-1, released by AVK, regulates SMB motor neurons via receptor NPR-6 to modulate anterior bending amplitude. These results revealed a mechanism whereby proprioception works with dopamine and neuropeptide signaling to mediate homeostatic locomotor control. Together, through a consolidation of experimental and computational approaches, I found C. elegans utilizes its circuitry not only to act motor behaviors but to adjust/correct its ongoing behaviors in its natural contexts.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Bioengineering

First Advisor

Christopher Fang-Yen

Keywords

Adaptative behavior, Locomotion, Motor circuit, Motor control, Neuromodulation, Proprioception

Subject Categories

Biophysics | Neuroscience and Neurobiology

EXPERIMENTAL AND COMPUTATIONAL ANALYSES OF LOCOMOTOR RHYTHM GENERATION AND MODULATION IN *CAENORHABDITIS ELEGANS*

Hongfei Ji

A DISSERTATION

in

Bioengineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2022

Supervisor of Dissertation

Christopher Fang-Yen
Associate Professor of Bioengineering

Graduate Group Chairperson

Yale E. Cohen
Professor of Bioengineering and Otorhinolaryngology

Dissertation Committee

Konrad Kording, Nathan Francis Mossell University Professor of Bioengineering
Michael Nusbaum, Professor of Neuroscience
Alexander Proekt, Associate Professor of Neuroscience
Gal Haspel, Assistant Professor of Biology, New Jersey Institute of Technology
Niels Ringstad, Associate Professor of Cell and Molecular Biology, New York University

EXPERIMENTAL AND COMPUTATIONAL ANALYSES OF LOCOMOTOR RHYTHM GENERATION AND MODULATION IN *CAENORHABDITIS ELEGANS*

ABSTRACT

EXPERIMENTAL AND COMPUTATIONAL ANALYSES OF LOCOMOTOR RHYTHM GENERATION AND MODULATION IN *CAENORHABDITIS ELEGANS*

Hongfei Ji

Christopher Fang-Yen

Neural circuits coordinate with muscles and sensory feedback to generate motor behaviors appropriate to its natural environment. Studying mechanisms underlying complex organism locomotion has been challenging, partly due to the complexity of their nervous systems. Here, I used the roundworm *C. elegans* to understand the locomotor circuit. With its well-mapped nervous system, easily-measurable movements, genetic manipulability, and many human homologous genes, *C. elegans* has been commonly used as a model organism for dissecting the circuit, cellular, and molecular principles of locomotion. My work introduces two separate approaches to probe the mechanisms by which the *C. elegans* motor circuit generates and modulates undulations. First, I quantified *C. elegans* movements during free locomotion and during transient muscle inhibition. Undulations were asymmetrical with respect to the duration of bending and unbending per cycle. Phase response curves induced by brief optogenetic head muscle inhibitions showed gradual increases and rapid decreases as a function of phase at which the perturbation was applied. A relaxation oscillator model was developed based on proprioceptive thresholds that switch the active muscle moment. It quantitatively agrees with data from free movement, phase responses, and previous results for gait adaptation to mechanical loads. Next, I characterized a proprioception-mediated compensatory behavior during *C. elegans* forward locomotion: the anterior body bending amplitude compensates for the change in midbody bending amplitude by an opposing homeostatic response. I demonstrated that curvature

compensation requires dopamine signaling driven by PDE neurons. Calcium imaging experiments suggested a proprioceptive functionality for PDE in sensing midbody curvature. Downstream of PDE dopamine signaling, curvature compensation requires D2-like dopamine receptor DOP-3 in the interneurons AVK. FMRFamide-like neuropeptide FLP-1, released by AVK, regulates SMB motor neurons via receptor NPR-6 to modulate anterior bending amplitude. These results revealed a mechanism whereby proprioception works with dopamine and neuropeptide signaling to mediate homeostatic locomotor control. Together, through a consolidation of experimental and computational approaches, I found *C. elegans* utilizes its circuitry not only to act motor behaviors but to adjust/correct its ongoing behaviors in its natural contexts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1: INTRODUCTION

BASIC ELEMENTS OF LOCOMOTOR CIRCUITS

Animals display locomotor behaviors such as crawling, walking, swimming, or flying via rhythmic patterns of muscle contractions and relaxations. In many animals, motor rhythms originate from networks of central pattern generators (CPGs), neuronal circuits capable of generating rhythmic outputs without rhythmic input (Cohen and Wallén, 1980; Grillner, 2003; Kiehn, 2011; Kristan and Calabrese, 1976; Marder and Calabrese, 1996; Pearce and Friesen, 1984; Yu et al., 1999). In vertebrates CPG-generated motor rhythms typically arise from a combined contribution of ipsilateral excitatory drive and reciprocal inhibition in the spinal cord (Brown, 1911; Buchanan and Grillner, 1987; Grillner and El Manira, 2020; Kiehn, 2016; Marder and Calabrese, 1996; Roberts et al., 2010; Wilson, 1961; WILSON and WEIS-FOGH, 1962).

Although isolated CPGs can produce outputs in the absence of sensory input, in the intact animal sensory feedback plays a critical role in coordinating motor rhythms across the body and modulating their characteristics (Friesen, 2009; Grillner and Wallen, 2002; Mullins et al., 2011; Pearson, 2004; Wen et al., 2012). Sensory feedback allows animals to adapt locomotor patterns to their surroundings (Andersson et al., 1981; Bidaye et al., 2018; Brodfuehrer and Friesen, 1986) and adapt to unexpected perturbations (Ekeberg and Grillner, 1999). Sensory inputs induced by electric stimulation of receptor cells (Yu and Friesen, 2004) or by mechanical perturbation of body segments (Grillner, 2021; Grillner et al., 1981) can entrain an animal's motor behavior to imposed patterns, demonstrating the flexibility of motor systems in responding to feedback.

Animal movements are driven not only by active muscle contractions, but also by passive mechanical forces including elastic recoil of muscles and other body structure, internal damping forces, and forces from the interaction with the external environment. Efficient locomotion in

vertebrates depends on storage of elastic energy in tendons and muscles (Roberts and Azizi, 2011). In insects, elasticity in the leg joints plays an important role in generating forces for walking and jumping (Ache and Matheson, 2013). A comprehensive understanding of animal locomotion should therefore encompass not only neural activity, muscle activity, and sensory feedback, but also biomechanical forces within the animal's body and between the animal and its environment (**Fig. 1.1A**; Borgmann et al., 2009; Grillner and Wallen, 2002; Kiehn, 1998).

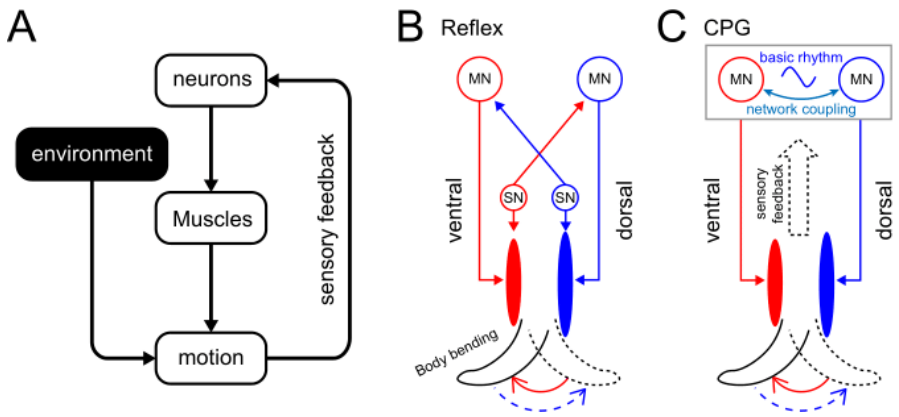

Figure 1.1. Rhythm generation in C. elegans.

(A) Motor neurons generate neuronal signals to control the activation of muscles, which generates movement subject to internal and external environmental constraints. Sensory input provides feedback about body position and the environment.

(B and C) Two possible models for locomotory rhythm generation in *C. elegans*. (B) In a reflex loop model, sensory neurons (SN) detect body postures and excite motor neurons (MN) to activate body wall muscles.

(C) In a central pattern generator (CPG) model, network of motor neurons generates basic rhythmic patterns that are transmitted to body wall muscles (BWM) while sensory feedback modulates the CPG rhythm. Diagrams adapted from Marder and Bucher (Marder and Bucher,

2001).

PROPRIOCEPTIVE CONTROL OF LOCOMOTION

In intact animals, the actual behavioral outputs during locomotion are subjected to proprioceptive signals arising from sensory neurons (Andersson et al., 1981; Brodfuehrer and Friesen, 1986; Friesen, 2009; Grillner and Wallen, 2002; Wen et al., 2012). In leeches (Cang and Friesen, 2000; Cang et al., 2001), lamprey (Bowtell and Williams, 1991), and *Drosophila* (Akitake et al., 2015; Mendes et al., 2013), specialized proprioceptive neurons and sensory receptors in body muscles detect sensory inputs to regulate and coordinate the centrally generated motor patterns. In limbed animals, sensory feedback from stretch receptors in the legs plays a causal role in generating and molding the bursting activity of leg motoneurons during limb movements (Smith et al., 1993; Wisleder et al., 1990; Wolf and Pearson, 1988). In mouse proprioceptive neurons, deletion of *Piezo2*, an excitatory mechanosensitive channel, induces severely uncoordinated body movements (Picton et al., 2021; Woo et al., 2015).

Proprioception is essential for regulating motor output not only during unperturbed locomotion but in a perturbed scenario when a gait perturbation occurs (Pearson, 2000). In humans and cats, the proprioceptive feedback from multisensory inputs, representing different sub-modalities in detecting surrounding changes, is continuously balanced and processed within spinal interneuron circuits to instruct compensatory electromyographic responses to the current locomotor situation (Dietz, 2002). In particular, sensory inputs that mediate monosynaptic spinal reflexes facilitate compensating movements for small ground irregularities (Dietz et al., 1987). Other proprioceptive signals, integrated by the polysynaptic spinal reflex system, produce more complex compensatory responses to ground conditions, involving synergistic coordination of leg muscle activation (Hansen et al., 1988). In limbless animals, both experimental and computational studies demonstrate that they rely critically on proprioceptive feedback to adapt their undulatory

gait to the changing physical environment (Berri et al., 2009; Boyle et al., 2012; Fang-Yen et al., 2010; Fouad et al., 2018; Iwasaki et al., 2014). In *C. elegans*, previous studies reported that optogenetic muscle inhibition of the anterior region could induce simultaneous oscillation at different frequencies in head and tail (Fouad et al., 2018; Xu et al., 2018), and that the motor dynamics displayed a biphasic, sawtooth-shaped phase response curve upon transient perturbations (Ji et al., 2021a), both indicating unique roles of proprioception in an animal's motor system.

Although a variety of proprioceptive components and interactions have been verified to contribute to adaptive locomotion in various organisms, the underlying signaling relationships of premotor neurons, motor neurons, and muscle cells that encode locomotor adaptation to gait perturbations remain primarily unknown (Büschges and Mantziaris, 2021; Dietz, 2002; Zhen and Samuel, 2015).

NEUROMUSCULAR COMPONENTS FOR *C. ELEGANS* MOTOR RHYTHM GENERATION

Fully understanding the molecular and cellular mechanisms of locomotory rhythm generation and coordination requires a model system with easily amenable, yet sophisticated behavioral outputs carried out by circuits that can be thoroughly dissected at the molecular and cellular levels.

Here I study mechanisms of locomotor rhythm generation and its modulation by sensory feedback in the nematode *Caenorhabditis elegans*. With its easily quantifiable behavior (Croll, 1970), well-mapped nervous system (Cook et al., 2019; White et al., 1986), genetic manipulability (Bargmann, 1998; Brenner, 1974; Hobert, 2003), and optical transparency, this worm is a unique model for obtaining an integrative understanding of locomotion.

C. elegans forward locomotion consists of anterior-to-posterior dorsoventral undulations (Croll, 1970). These movements are mediated by a neuromuscular circuit consisting of interneurons, excitatory cholinergic motor neurons, inhibitory GABAergic motor neurons, and

body wall muscles. Laser ablation studies have shown that the cholinergic B-type motor neurons are required for forward locomotion (Chalfie et al., 1985). The GABAergic D-type motor neurons provide dorsoventral cross-inhibition to the body wall muscles and are essential for maintaining high frequency locomotion (Deng et al., 2020; Mclntire et al., 1993). A set of premotor interneurons (AVB, PVC, AVA, AVD, and AVE) regulate forward and reverse movements (Chalfie et al., 1988; Driscoll and Kaplan, 1997; Von Stetina et al., 2006). Ablation of all premotor interneurons does not deprive *C. elegans* of the ability to undulate (Gao et al., 2018; Kawano et al., 2011), suggesting that a network consisting of excitatory motor neurons and muscles may be sufficient to generate rhythmicity.

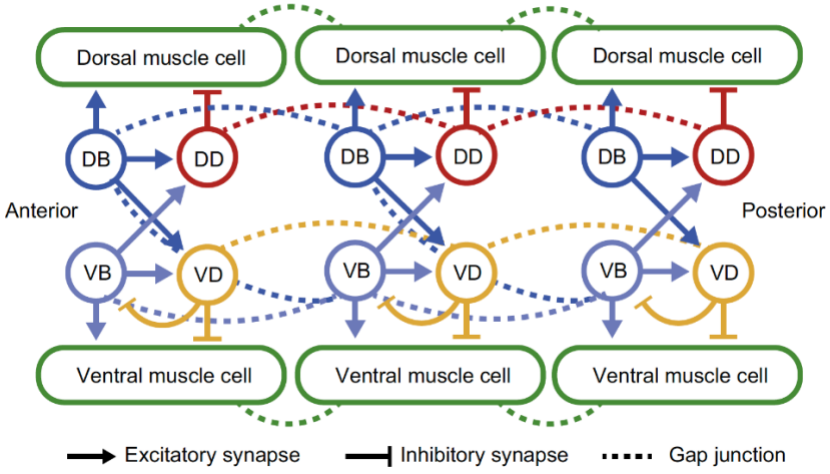

Figure 1.2. A schematic figure of connectivity in the wiring diagram for forward locomotion (adapted from Wen et al., 2012).

In recent years, rapid progress has been made in understanding the motor circuits in *C. elegans* (Boyle et al., 2012; Cohen and Denham, 2019; Gjorgjieva et al., 2014; Haspel et al., 2020; Zhen and Samuel, 2015). In *C. elegans*, locomotory behavior arises from a compact nervous system (**Fig. 1.2**). A principal CPG has been suggested (Karbowski et al., 2008; Niebur and Erdös, 1991; Wen et al., 2012) to be localized in the head to provide bending rhythms, and

the bending waves are propagated along the body through a chain of reflexes connecting adjacent body segments (**Fig. 1.3**). Optogenetic and lesion experiments suggested that multiple oscillators exist in the ventral nerve cord (**Fig. 1.3**; Fouad et al., 2018). However, the mechanisms that give rise to these oscillators are still poorly understood.

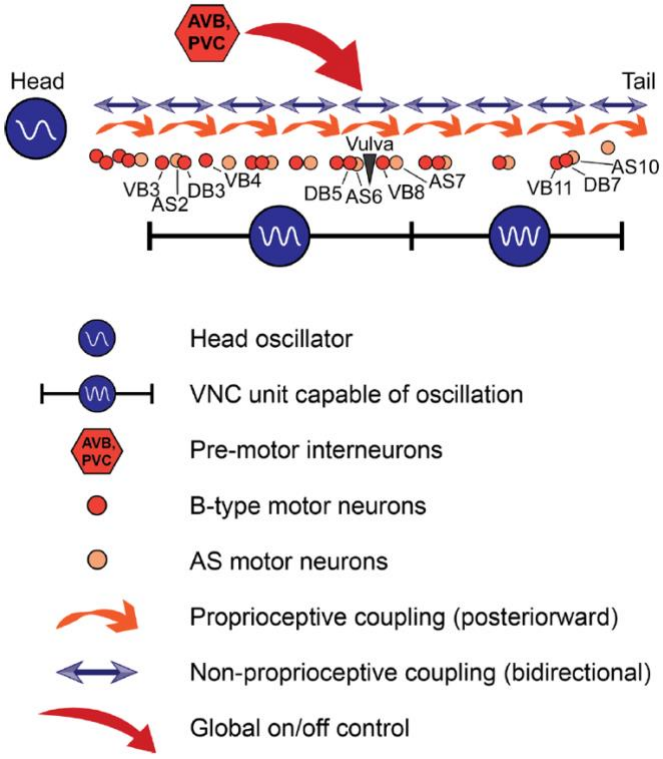

Figure 1.3. Distributed rhythm oscillators underlie *C. elegans* forward locomotion (adapted from Fouad et al., 2018).

Two units of the ventral nerve cord (VNC) motor neurons can independently generate a fictive motor rhythm. All oscillating units are coupled by proprioceptive coupling (Wen et al., 2012) and other unknown, likely non-proprioceptive, bidirectional coupling mechanisms. Premotor interneurons promote or suppress this circuit, out of which AVB may play an additional, unexplained role in rhythm generation.

PROPRIOCEPTIVE CONTROL OF *C. ELEGANS* MOTOR BEHAVIOR

In *C. elegans,* proprioceptive feedback is crucial for generating and modulating locomotor rhythms. Several proprioceptive mechanisms were identified implicating motor neurons, interneurons, sensory neurons, as well as neuromodulation of biogenic amines and neuropeptides.

Some *C. elegans* motor neurons appear to also display proprioceptive functions. For example, the B-type motor neurons mediate proprioceptive coupling from anterior to posterior bending during forward locomotion (Wen et al., 2012). The SMDD motor neurons, localized at the head, have been identified as proprioceptive regulators of head steering during locomotion (Yeon et al., 2018). Both the B-type motor neurons and the SMDD head motor neurons have long asynaptic processes hypothesized to have proprioceptive function (White et al., 1986). In particular, SMDD, B1, and B2 motor neurons have been suggested as candidate locomotor CPG elements (Kaplan et al., 2020). In addition, two types of neurons, the DVA and PVD interneurons, have proprioceptive roles in regulating the worm's body bend movement. DVA exhibits proprioceptive properties that depend on a mechanosensitive channel, TRP-4, which acts as a stretch receptor to regulate the body bend amplitude during locomotion (Li et al., 2006). In another study, body bending was shown to induce local dendritic Ca^{2+} transients in PVD and dendritic release of neuropeptide encoded by *nlp-12*, which appears to regulate the amplitude of body movement (Tao et al., 2019). In vivo Ca^{2+} imaging of dopaminergic ciliated sensory neurons PDE revealed that Ca^{2+} levels oscillates during forward movement, phase-locked to the forward propagating bend, suggesting a proprioceptive capability in PDE of sensing body bends (Cermak et al., 2020).

Despite the recent progress in understanding proprioception and proprioceptive units within this small circuit, how CPGs are formed and how they work with proprioceptive cues in order to

generate locomotion and respond to external environmental modules remain largely unknown.

COMPUTATIONAL MODELS OF *C. ELEGANS* LOCOMOTOR BEHAVIOR

Computational models for *C. elegans* motor behavior have long been an important complement to experimental approaches, since an integrative understanding of locomotion requires consideration of neural, muscular, and mechanical degrees of freedom, and are often tractable only by modeling (Boyle et al., 2012; Bryden and Cohen, 2008; Denham et al., 2018; Izquierdo and Beer, 2018; Johnson et al., 2021; Karbowski et al., 2008; Kunert et al., 2017; Olivares et al., 2021).

An early model (Niebur and Erdös, 1991) assumes that a CPG located in the head initiates dorsoventral bends and that a combination of neuronal and sensory feedback mechanisms propagates the waves in the posterior direction. In this model, sensory feedback plays a modulatory role in producing smoother curvature waves but is not explicitly required for rhythm generation itself. Other computational models have aimed to describe how the motor circuit generates rhythmicity. For example, several neural models have been developed for the forward-moving circuit (Karbowski et al., 2008; Olivares et al., 2021) by incorporating of all major neural components and connectivity (**Fig. 1.4**). In particular, Karbowski's model included a CPG in the head based on effective cross-inhibition between ventral and dorsal groups of interneurons. In contrast, Bryden and Cohen (Bryden and Cohen, 2008) developed a neural model in which each segment along the body is capable of generating proprioception-mediated oscillations. In this model, a circuit of AVB interneurons and B-type motor neurons suffices to generate robust locomotory rhythms without cross-inhibition.

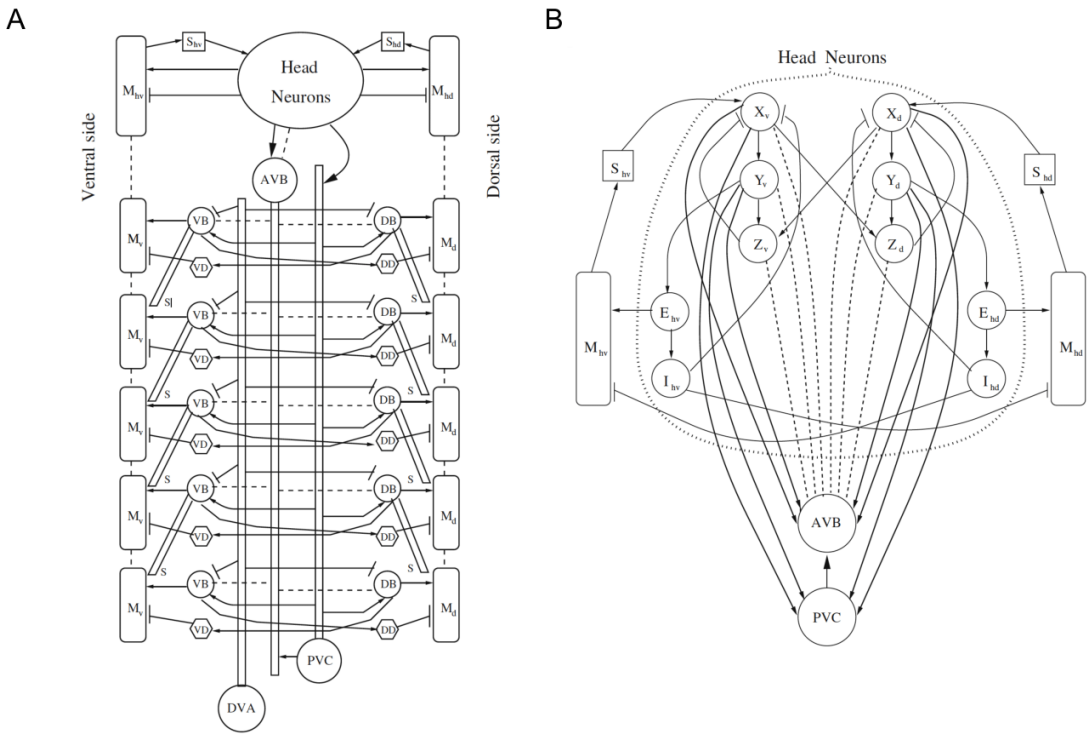

Figure 1.4. System-level neural model of the motor circuit for forward locomotion in *C. elegans* (adapted from Karbowski et al., 2008).

(A) The large-scale view of the circuit.

(B) Schematic diagram of the head CPG originated from cross-inhibitions among interneurons.

Other models have examined how *C. elegans* adapts its undulatory wavelength, frequency, and amplitude as a gait adaptation to external load (Boyle et al., 2012; Denham et al., 2018; Izquierdo and Beer, 2018; Johnson et al., 2021). To account for these changes, these models combined the motor circuit model with additional assumptions of stretch sensitivity in motor neurons, and worm body biomechanical constraints, to create a model that reproduced the changes in undulatory wave patterns under a range of external conditions.

However, these recent computational models were implemented on a cellular level where

assumptions were typically made for detailed cell properties and inter-cellular interactions that are not directly supported by experimental evidence (Bryden and Cohen, 2008; Haspel et al., 2010; Karbowski et al., 2008). Due to the paucity of experimental findings in cellular and synaptic properties of motor circuit elements (Gjorgjieva et al., 2014), this 'bottom-up' strategy of modelling faces potential challenges of experimental verification and computational superfluity.

OBJECTIVES AND OVERVIEW

In this work I sought to explore how the *C. elegans* motor system generates a locomotor rhythm and how it adapts locomotion in response to gait perturbations. In Chapter 2, I introduce a dynamical systems approach to analyze the worm's motor behavior. By integrating quantitative behavioral measurements, optogenetic phase response analyses, and computational modeling I show that the locomotor system acts as a relaxation oscillator (a type of nonlinear oscillator). In Chapter 3, I demonstrate that *C. elegans* uses a posterior-to-anterior proprioceptive feedback loop to adapt its locomotor amplitude to gait perturbation in a homeostatic manner. Using combined experimental analyses, the corresponding mechanisms are described on the behavior, circuit, and molecular levels. In Chapter 4, I present my perspective on future directions for my work. In the Appendix, I post my software for computational modeling and experimental data analysis.

CHAPTER 2: PHASE RESPONSE ANALYSES SUPPORT A RELAXATION OSCILLATOR MODEL OF LOCOMOTOR RHYTHM GENERATION IN *C. ELEGANS*

Hongfei Ji[1,*], Anthony D. Fouad[1,*], Shelly Teng[1], Alice Liu[1], Pilar Alvarez-Illera[1], Bowen Yao[1], Zihao Li[1], and Christopher Fang-Yen[1,2]

[1]Department of Bioengineering, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA 19104

[2]Department of Neuroscience, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104

[*]Equal contributions

This chapter is a slightly edited version of my paper published in the journal *eLife* (Ji et al., 2021b). Anthony Fouad and his students (Shelly Teng, Alice Liu, and Pilar Alvarez-Illera) conducted pioneering work associated with PRC experiments in the early stages including experiment design, data curation, data analysis, and data interpretation. I performed additional experiments including optogenetic inhibition tests with single-side illuminations and under varying viscosities, and viscosity-dependent tests for biomechanical analysis. Christopher Fang-Yen and I together conceived the ideas behind the threshold-switching mechanisms of the model in the early stages. I implemented the primary model and other additional models. Most transgenic lines were generated by my colleague Anthony Fouad (All YX strains in Key resources table 2.1). The optogenetic targeting system was originally designed by Fang-Yen and was later constructed with modifications by Fouad. Custom software to run the laser system was written by Fouad. Custom algorithms for data analysis and modeling were written either by me or Fouad. Fang-Yen also helped with designing, troubleshooting, and interpreting experiments and models.

INTRODUCTION

To experimentally probe mechanisms of rhythmic motor generation, including the role of proprioceptive feedback, we measured the phase response curve (PRC) upon transient optogenetic inhibition of the head muscles. We found that the worms displayed a biphasic, sawtooth-shaped PRC with sharp transitions from phase delay to advance.

We used these findings to develop a computational model of rhythm generation in the *C. elegans* motor circuit in which a relaxation-oscillation process, with switching based on proprioceptive feedback, underlies the worm's rhythmic dorsal-ventral alternation. We sought to develop a phenomenological model to describe an overall mechanism of rhythm generation but not the detailed dynamics of specific circuit elements. We aimed to incorporate biomechanical constraints of the worm's body and its environment (Fang-Yen et al., 2010; Gray and Lissmann, 1964; Wallace, 1968), as well as account for how sensory feedback is incorporated. To improve predictive power, we aimed to minimize the number of free parameters used in the model. Finally, we sought to optimize and test this model with new experiments as well as with published findings.

Our model reproduces the observed PRC and describes the locomotory dynamics around optogenetic inhibitions in a manner that closely fits our experimental observations. Our model also agrees with results on gait adaptation to external load and the asymmetry in time-dependent curvature patterns of undulating worms. Our experimental findings and computational model together yield insights into how *C. elegans* generates rhythmic locomotion and modulates them depending on the environment.

RESULTS

C. ELEGANS FORWARD LOCOMOTION EXHIBITS A STABLE AND NONSINUSOIDAL LIMIT CYCLE

To gain insight into wave generation, we first sought to examine the quantitative behavioral characteristics of worms during forward locomotion. First, we measured the undulatory dynamics of body bending by computing the time-varying curvature along the centerline of the body (Fang-Yen et al., 2010; Leifer et al., 2011; Pierce-Shimomura et al., 2008; Wen et al., 2012) from analysis of dark field image sequences of worms exhibiting forward locomotion. In order to quantitatively treat the drag between the body and its environment, we examined locomotion of worms in dextran solutions of known viscosity (see *Methods*; Fang-Yen et al., 2010). The normalized body coordinate is defined by the distance along the body centerline divided by the body length (**Fig. 2.1A**). The curvature κ at each point along the centerline of the body is the reciprocal of local radius of curvature (**Fig. 2.1A**), with a positive (negative) curvature representing ventral (dorsal) bending. We further define the dimensionless curvature $K = \kappa \cdot L$, where L is the length of the worm. We focus on curvature dynamics of worm's head region (0.1-0.3 body coordinate, **Fig. 2.1B**).

We used this behavioral data to generate phase portraits, geometric representations of a dynamical system's trajectories over time (Izhikevich, 2007), in which the time derivative of the curvature is plotted against the curvature. If the curvature were sinusoidal over time, as it is often modeled in slender swimmers (Fang-Yen et al., 2010; Gray, 1933; Guo and Mahadevan, 2008; Niebur and Erdös, 1991; Ranner, 2020), the time derivative of curvature would also be sinusoidal, with a phase shift of $\pi/4$ radians relative to the curvature, and the resulting phase portrait would be symmetric about both the K and dK/dt axes. Instead, we found that the phase portrait of *C. elegans* forward locomotion is in fact non-ellipsoidal and strongly asymmetric with respect to reflection across the K or dK/dt axes (**Figs. 2.1C** and **2.1D**). Plots of both the phase portrait (**Fig.**

2.1D) and the time dependence (**Fig. 2.1C**) show that K and dK/dt are strongly non-sinusoidal.

In addition to the head, other parts of the worm's body also display nonsinusoidal bending movements (**Fig. S2.1**). In this paper, we focus on curvature dynamics of the worm's head region (0.1-0.3 body coordinate) where the bending amplitude is largest and the nonsinusoidal features are most prominent (**Fig. S2.1**).

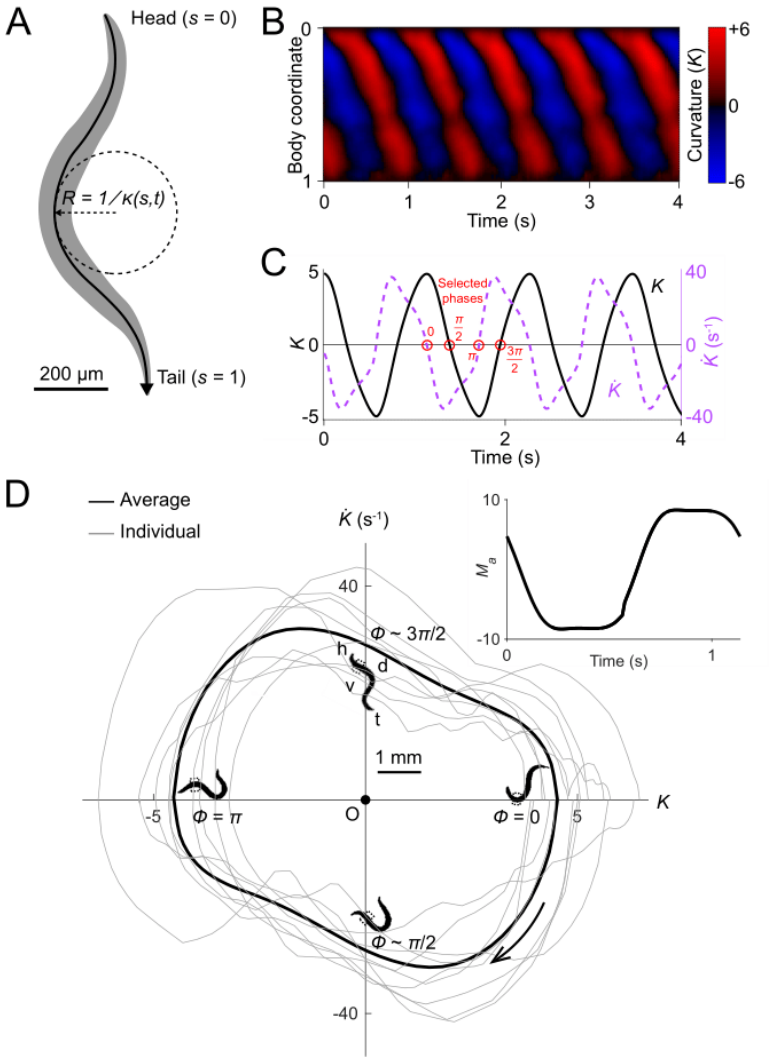

Figure 2.1. Undulatory dynamics of freely moving worms.

(A) Worm undulatory dynamics are quantified by the time-varying curvature along the body. The normalized body coordinate is defined by the fractional distance along the centerline (head = 0, tail = 1). The curvature κ is the reciprocal of the local radius of curvature with positive and negative values representing dorsal and ventral curvature, respectively.

(B) Curvature as a function of time and body coordinate during forward movement in a viscous liquid. Body bending curvature K is represented using the nondimensional product of κ and body length L.

(C) Curvature (black) in the anterior region (average over body coordinate 0.1-0.3) and the time derivative (dashed grey) of this curvature. Red circles mark four representative phases (0, $\pi/2$, π, and $3\pi/2$). The curve is an average of 5041 locomotory cycles from 116 worms.

(D) Phase portrait representation of the oscillatory dynamics, showing the curvature and the time derivative of the curvature parameterized by time. Images of worm correspond to the phases marked in C. Arrow indicates clockwise movement over time. (Inset) waveform of the scaled active muscle moment, estimated by equation $M_a = K + \tau_u \dot{K}$. Both curves were computed from the data used in C.

We asked whether the phase portrait represents a stable cycle, i.e. whether the system tends to return to the cycle after fluctuations or perturbations away from it. To this end, we analyzed the recovery after brief optogenetic muscle inhibition. We used a closed-loop system for optically targeting specific parts of the worm (Fouad et al., 2018; Leifer et al., 2011) to apply brief pulses of laser illumination (0.1 s duration, 532 nm wavelength) to the heads of worms expressing the inhibitory opsin *NpHR* in body wall muscles (*Pmyo-3::NpHR*). Simultaneous muscle inhibition on both sides causes *C. elegans* to straighten due to internal elastic forces (Fang-Yen et al., 2010).

Brief inhibition of the head muscles during forward locomotion was followed by a maximum degree of paralysis approximately 0.3 s after the end of the pulse, then a resumption of undulation (**Figs. 2.2A** and **2.2B**).

To quantify the recovery dynamics, we defined a normalized deviation d describing the state of the system relative to the phase portrait of normal oscillation (see *Methods*), such that d = -1 at the origin, d = 0 at the limit cycle, and d > 0 outside the limit cycle. We found that the deviation following optogenetic perturbation (**Fig. S2.2**) decays toward zero regardless of the initial deviation from the normal cycle, indicating that the worm returns to its normal oscillation after a perturbation. These results show that *C. elegans* head oscillation during forward locomotion is stable under optogenetic perturbation. The dynamics of these perturbed worms also allow us to reconstruct the phase isochrons and vector flow fields (**Fig. S2.3**) of the worm's head oscillation, two other important aspects of an oscillator (see *Methods*).

Taken together, these results show that during forward locomotion, head oscillation of a worm constitutes a stable oscillator containing a nonsinusoidal limit cycle.

TRANSIENT OPTOGENETIC INHIBITION OF HEAD MUSCLES YIELDS A SLOWLY RISING, RAPIDLY FALLING PHASE RESPONSE CURVE

The phase response curve (PRC) describes the change in phase of an oscillation induced by a perturbation as a function of the phase at which the perturbation is applied, and is often used to characterize biological and nonbiological oscillators (Izhikevich, 2007; Pietras and Daffertshofer, 2019; Schultheiss et al., 2011). We performed a phase response analysis of the worm's locomotion upon transient optogenetic inhibitions.

Using data from 991 illuminations (each 0.1 s in duration) in 337 worms, we analyzed the animals' recovery from transient paralysis as a function of the phase at which the illumination occurred. We define the phase such that it equals to zero at the point of maximum ventral

bending (**Fig. 2.2D**). When inhibition occurred with phase in the interval $[0, \pi/6]$, the head typically straightened briefly and then continued the previous bend, resulting in a phase delay for the oscillation (**Figs. 2.2C-E**). When inhibition occurred with phase in the interval $[\pi/3, \pi/2]$, the head usually appeared to discontinue the previous bend movement, which resulted in a small phase advance (**Figs. 2.2F-H**). When inhibition occurred with phase in the interval $[2\pi/3, 5\pi/6]$, the head response was similar to that between the interval $[0, \pi/6]$, and also resulted in a phase delay (**Figs. 2.2I-K**).

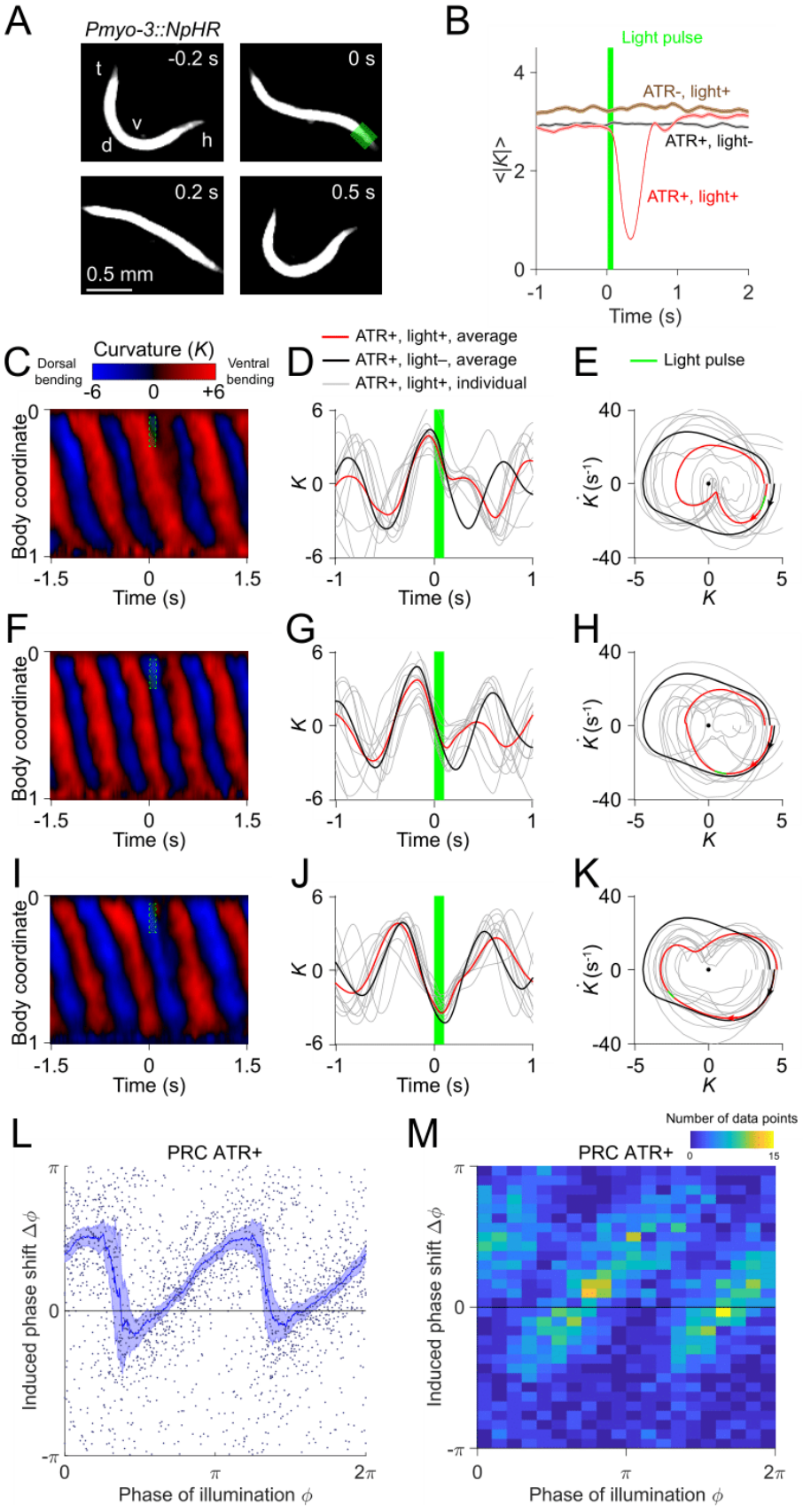
A
Pmyo-3::NpHR
-0.2 s
0 s
0.2 s
0.5 s
t
v
d
h
0.5 mm
B
Light pulse
ATR-, light+
ATR+, light-
ATR+, light+
<|K|>
Time (s)
C
Curvature (K)
Dorsal bending
Ventral bending
-6
0
+6
Body coordinate
Time (s)
D
ATR+, light+, average
ATR+, light–, average
ATR+, light+, individual
K
Time (s)
E
Light pulse
K̇ (s⁻¹)
K
F
G
H
I
J
K
L
PRC ATR+
Induced phase shift Δφ
Phase of illumination φ
M
Number of data points
PRC ATR+
Induced phase shift Δφ
Phase of illumination φ

Figure 2.2. Analysis of phase-dependent inhibitions for head oscillation using transient optogenetic muscle inhibition.

(A) Images of a transgenic worm (Pmyo-3::NpHR) perturbed by a transient optogenetic muscle inhibition in the head during forward locomotion. Green shaded region indicates the 0.1 s laser illumination interval. h: head; t: tail; v: ventral side; d: dorsal side.

(B) Effect of muscle inhibition on mean absolute curvature of the head. Black curve represents control ATR+ (no light) group (3523 measurements using 337 worms). Brown curve represents control ATR- group (2072 measurements using 116 worms). Red curve represents ATR+ group (1910 measurements using 337 worms). Green bar indicates 0.1 s light illumination interval starting at $t = 0$.

(C-E) Perturbed dynamics around light pulses occurring in the phase range $[0, \pi/6]$. (C) Kymogram of time-varying curvature K around a 0.1 s inhibition (green dashed box). (D) Mean curvature dynamics around the inhibitions (green bar, aligned at $t = 0$) from ATR+ group (red curve, 11 trials using 4 worms) and control ATR+ (no light) group (black curve, 8 trials using 3 worms). Grey curves are individual trials from ATR+ group (10 randomly selected trials are shown). (E) Mean phase portrait graphs around the inhibitions (green line) from ATR+ group (same trials as in D) and control group (ATR+, no light, 3998 trials using 337 worms). Grey curves are individual trials from ATR+ group.

(F-H) Similar to C-E, for phase range $[\pi/3, \pi/2]$.

(I-K) Similar to C-E, for phase range $[2\pi/3, 5\pi/6]$.

(L) PRC from optogenetic inhibition experiments (ATR+ group, 991 trials using 337 worms, each point indicating a single illumination of one worm). The curve was obtained via a moving average along the x-axis with 0.16π in bin width and the filled area represents 95% confidence interval

within the bin.

(M) A 2-dimensional histogram representation of the PRC using the same data. The histogram uses 25 bins for both dimensions, and the color indicates the number of data points within each rectangular bin.

Combining the data from all phases of inhibition yielded a sawtooth-shaped PRC with two sharp transitions from phase delay to advance as well as two relatively slow ascending transitions from phase advance to delay (**Figs. 2.2L,M**). In control worms, which do not express *NpHR* in the body wall muscles (see *Methods*), the resulting PRC shows no significant phase shift over any phases of illumination (**Fig. S2.4**). In worms perturbed with shorter pulses (0.055 s duration), we observed a similar sawtooth-shaped PRC (**Fig. S2.5**).

In addition to phase response analyses with perturbations to the worm's anterior, we conducted similar analyses for the dynamics across the body by optogenetically inhibiting body wall muscles of other regions (**Fig. S2.6**). We found that the sawtooth feature of PRC tends to decrease monotonically as the perturbation occurs further away from the head (**Fig. S2.6A,E,I**).

Next, we asked whether the sharp downward transitions in the PRC represent a continuous decrease or instead result from averaging data from a bimodal distribution. When we plotted the distribution of the same data in a 2-D representation we found that the phase shifts display a piecewise, linear increasing dependence on the phase of inhibition with two abrupt jumps occurring at $\phi \approx \pi/3$ and $4\pi/3$, respectively (**Fig. 2.2M**). This result shows that the sharp decreasing transitions in PRC reflect bimodality in the data rather than continuous transitions.

In addition to examining PRCs induced by muscle inhibition, we also calculated PRCs with respect to inhibitions of cholinergic motor neurons. We performed similar experiments on transgenic worms in which the inhibitory opsin NpHR is expressed in either all cholinergic

neurons (*Punc-17::NpHR::ECFP*) or B-type motor neurons (*Pacr-5::Arch-mCherry*). In both strains, we again observed sawtooth-shaped PRCs (**Figs. S2.7** and **S2.8**), with variations only in the magnitudes of phase shifts. These experiments show that the sawtooth-shaped feature of PRC is maintained for motor neuron inhibition, suggesting that the transient muscle and neuron inhibition interrupt the motor circuit dynamics in a similar manner.

The GABAergic D-type motor neurons provide a dorsoventral reciprocal inhibition of opposing muscles during locomotion. We asked whether the D-type motor neurons are required for the observed sawtooth shape of the PRC. We examined transgenic worms that express *NpHR* in the body wall muscles but have mutations *unc-49(e407)*, a loss-of-function mutant of $GABA_A$ receptor that is required by the D-type motor neurons (Bamber et al., 1999). After performing optogenetic inhibition experiments we found that the PRC also displays a sawtooth feature (**Fig. S2.9**). This result shows that D-type motor neurons are not necessary for the motor rhythm generator to show the sawtooth-shaped PRC.

Sawtooth-shaped PRCs are observed in a number of systems with oscillatory dynamics, including the van der Pol oscillator (Rosenblum, 2018), and may reflect a phase resetting property of an oscillator with respect to a perturbation (Izhikevich, 2007; Schultheiss et al., 2011). Further interpretation of the PRC results is given below.

WORM MUSCLES DISPLAY A RAPID SWITCH-LIKE ALTERNATION DURING LOCOMOTION

As a first step in interpreting and modeling our findings, we estimated the patterns of muscle activity in freely moving worms, in part by drawing on previous biomechanical analyses of nematode movement (Fang-Yen et al., 2010; Gray and Lissmann, 1964; Ranner, 2020; Wallace, 1968).

In mechanics, a moment is a measure of the ability of forces to produce bending about an axis. Body wall muscles create local dorsal or ventral bending by generating active moments

across the body. In addition to the active moments from muscles, there are also passive moments generated by the worm's internal viscoelasticity and by the forces due to the interaction of the worm with its external environment.

We estimated the output patterns of the active muscle moment that drives the head oscillations of freely moving worms immersed in viscous solutions. Following previous analyses of *C. elegans* locomotor biomechanics under similar external conditions (Fang-Yen et al., 2010), the scaled active muscle moment can be described as a linear combination of the curvature and the time derivative of the curvature (**Eqn. 2.1**; also see *Methods*). We observed that in the phase portrait graph (**Fig. 2.1D**), there are two nearly linear portions of the curve. We hypothesized that these linear portions correspond to two bouts during which the active muscle moment is nearly constant.

Using fits to the phase plot trajectory (see *Methods*) we estimated the waveform of the active muscle moment as a function of time (**Fig. 2.1D *Inset***). We found that the net active muscle moment alternates between two plateau regions during forward locomotion. From the slope of the steep portions on this curve, we estimated the time constant for transitions between active moments to be $\tau_m \approx 100\ ms$. This time constant is much smaller than the duration of each muscle moment plateau period ($\approx\ 0.5\ s$), suggesting that the system undergoes rapid switches of muscle contractions between two saturation states.

A RELAXATION OSCILLATION MODEL EXPLAINS NONSINUSOIDAL DYNAMICS

We reasoned that the rapid transitions of the active muscle moment might reflect a switching mechanism in the locomotory rhythm generation system. We hypothesized that the motor system generates locomotory rhythms by switching the active moment of the muscles based on proprioceptive thresholds.

To expand further upon these ideas, we developed a quantitative model of locomotory rhythm

generation. We consider the worm as a viscoelastic rod where the scaled curvature *K(t)* varies according to:

$$K(t) + \tau_u \frac{dK(t)}{dt} = M_a(t), \qquad [\mathbf{2.1}]$$

where τ_u describes the time scale of bending relaxation and $M_a(t)$ is the time-varying active muscle moment scaled by the bending modulus and the body length (see detailed derivations in *Methods*). We note that in a stationary state ($dK/dt = 0$), the curvature would be equal to the scaled active muscle moment. That is, the scaled active moment represents the static curvature that would result from a constant muscle moment.

We define a proprioceptive feedback variable P as a linear combination of the current curvature value and the rate of change of curvature. In our model, once this variable reaches either of two thresholds P_{th} and $-P_{th}$ (**Fig. 2.3D**), the active muscle moment undergoes a change of sign (**Fig. 2.3E**), causing the head to bend toward the opposite direction (**Fig. 2.3B**).

Our model has 5 parameters: (1) τ_u, the bending relaxation time scale, (2) τ_m, the muscle switching time scale, (3) M_0, the amplitude of the scaled active muscle moment, (4-5) b and P_{th}, which determine the switch threshold. The first 3 parameters were directly estimated from our experimental results from freely moving worms (see *Methods*). Parameters b and P_{th} were obtained using a two-round fitting procedure by fitting the model first to the freely moving dynamics (first round) and then to the experimental phase response curve (second round) (see *Methods*).

With this set of parameters, we calculated the model dynamics as represented by the phase portrait (**Fig. 2.3C**) as well as curvature waveform in one cycle period (**Fig. 2.3F**). We found that in both cases the model result agreed with our experimental observations. Our model captures

the asymmetric phase portrait trajectory shape found from our experiments (**Fig. 2.1D**). It also describes the asymmetry of head bending during locomotion: bending toward the ventral or dorsal directions occurs slower than straightening toward a straight posture during the locomotory cycle (**Fig. 2.3F *Inset***).

Considering the hypothesized mechanism under the biomechanical background (**Eqn. 2.1**), our model provides a simple explanation for the observed bending asymmetry during locomotion. According to the model, the active muscle moment is nearly constant during each period between transitions of the muscle moment. Biomechanical analysis under this condition predicts an approximately exponential decay in curvature, which gives rise to an asymmetric feature during each half period (**Fig. 2.3F**).

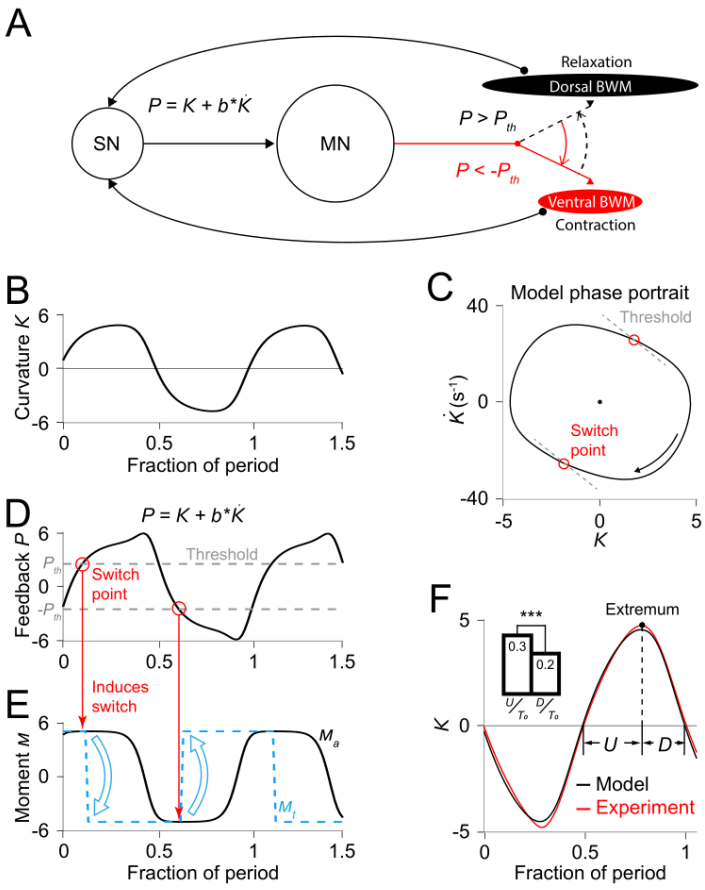

Figure 2.3. Free-running dynamics of a bidirectional relaxation oscillator model.

(A) Schematic diagram of the relaxation oscillator model. In this model, sensory neurons (SN) detect the total curvature of the body segment as well as the time derivative of the curvature. The linear combination of the two values, $P = K + b\dot{K}$, is modeled as the proprioceptive signal which is transmitted to motor neurons (MN). The motor neurons alternatingly activate dorsal or ventral body wall muscles (BWM) based on a thresholding rule: (1) if $P < -P_{th}$, the ventral body wall muscles get activated and contract while the dorsal side of muscles relax; (2) if $P > P_{th}$, vice versa. Hence, locomotion rhythms are generated from this threshold-switch process.

(B) Time-varying curvature K of the model oscillator. The time axis is normalized with respect to oscillatory period (same for D, E, and F).

(C) Phase portrait graph of the model oscillator. Proprioceptive threshold lines (grey dashed lines) intersect with the phase portrait graph at two switch points (red circles) at which the active moment of body wall muscles is switched.

(D) Time-varying proprioceptive feedback P received by the motor neurons. Horizontal lines denote the proprioceptive thresholds (grey dashed lines) that switch the active muscle moment at switch points (red circles, intersections between the proprioceptive feedback curve and the threshold lines).

(E) Time-varying active muscle moment. Blue-dashed square wave denotes target moment (M_t) that instantly switches directions at switch points. Black curve denotes the active muscle moment (M_a) which follows the target moment in a delayed manner.

(F) Time varying curvature in the worm's head region from experiments (red, 5047 cycles using 116 worms) and model (black). Model curvature matches experimental curvature with an MSE ≈ 0.18. (*Inset*) Bar graph of U (time period of bending toward the ventral or dorsal directions) and D

(time period of straightening toward a straight posture). Vertical bars are averages of fractions with respect to undulatory period T_0 of U and T (*** indicates p<0.0005 using Student's *t* test).

RELXATION OSCILLATOR MODEL REPRODUCES RESPONSES TO TRANSIENT OPTOGENETIC INHIBITION

We performed simulations of optogenetic inhibitions in our model. To model the transient muscle paralysis, the muscle moment is modulated by a bell-shaped function of time (**Fig. S2.10**; also see *Methods*) such that, upon inhibition, it decays toward zero and then recovers to its normal value, consistent with our behavioral observations (**Fig. 2.2B**).

From simulations with different sets of model parameters, we found that the model PRCs consistently exhibited the sawtooth shape found in experiments, though differing in height and timing of the downward transitions. In addition to the model parameters τ_u, M_0, and τ_m that had been explicitly estimated from free-moving experiments, we performed a two-round fitting procedure (see *Methods*) to determine the other parameters (including b, P_{th}, and parameters for describing the optogenetically induced muscle inhibitions (see **Fig. S2.10**) to best fit the freely moving dynamics and the experimental PRC, respectively, with a minimum mean squared error (MSE) (**Figs. 2.3F** and **2.4A**; also see *Methods*). For the parameters b and P_{th}, the optimization estimated their values to be $b = 0.046\ s$ and $P_{th} = 2.33$, as shown on the phase portraits (grey dashed lines in **Figs. 2.3C, 2.4B** and **2.4D**).

The threshold-switch mechanism model provides an explanation for the observed sawtooth-shaped PRC. By comparing model phase portrait graphs around inhibitions occurring at different phases (**Figs. 2.4B-E**), we found that the phase shift depends on the relative position of the inhibition with respect to the switch points on the phase plane. (1) If the effect of the inhibition occurs before the system reaches its switch point (**Fig. 2.4B**), the system will recover by continuing the previous bend and the next switch in the muscle moment will be postponed,

thereby leading to a phase delay (**Fig. 2.4C**). (2) As the inhibition progressively approaches the switch point, one would expect that the next switch in the muscle moment will also be progressively postponed; this explains the increasing portions of the PRC. (3) If the inhibition coincides with the switch point (**Fig. 2.4D**), the muscle moment will be switched at this point and the system will recover by aborting the previous bend tendency, resulting in a small phase advance (**Fig. 2.4E**). This switching behavior explains the two sharp downward transitions in the PRC.

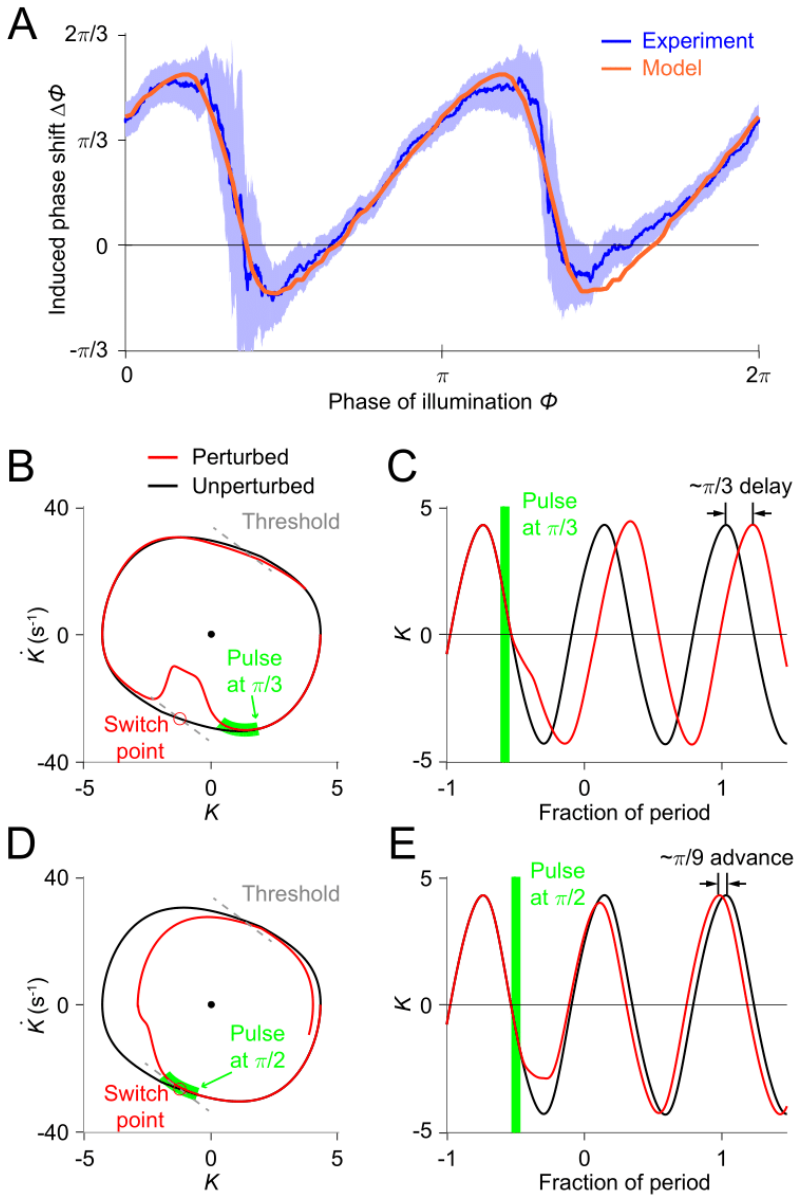

Figure 2.4. Simulations of optogenetic inhibitions in the relaxation oscillator model.

(A) Phase response curves measured from experiments (blue, same as in **Fig. 3L**) and model (orange). Model PRC matches experimental PRC with an MSE ≈ 0.12.

(B,C) Simulated dynamics of locomotion showing inhibition-induced phase delays in the model

oscillator. (B) Simulated phase portrait graphs around inhibition occurring at $\pi/6$ phase of cycle for perturbed (red) and unperturbed (black) dynamics. Green bar indicates the phase during which the inhibition occurs. (C) Same dynamics as in B, represented by time-varying curvatures. The time axis is normalized with respect to oscillatory period (same for E).

(D,E) Simulated dynamics of locomotion showing inhibition-induced phase advances in the model oscillator. (D) Simulated phase portrait graphs around inhibition occurring at $\pi/2$ phase of cycle for perturbed (red) and unperturbed (black) dynamics. (E) Same dynamics as in D, represented by time-varying curvatures.

RELAXATION OSCILLATOR MODEL PREDICTS PHASE RESPONSE CURVES FOR SINGLE-SIDE MUSCLE INHIBITION

As a further test of the model, we asked what PRCs would be produced with only the ventral or dorsal head muscles being transiently inhibited. In the model, the muscle activity is represented using the scaled active moment of muscles. We conducted model simulations (see *Methods*) to predict the PRCs for transient inhibitions of muscles on the dorsal side (**Fig. 2.5A, *Upper***) and ventral side (**Fig. 2.5B, *Upper***), respectively.

To experimentally perform phase response analysis of single-side muscle inhibitions, we visually distinguished each worm's dorsoventral orientation (via vulval location) and targeted light to either the ventral or dorsal side of the animal. Transiently illuminating (0.1 s duration) dorsal or ventral muscles in the head region of the transgenic worms *(Pmyo-3::NpHR*) induced a brief paralyzing effect when the segment was bending toward the illuminated side but did not induce a significant paralyzing effect when the segment was bending away from the illuminated side (**Fig. S2.11**).

Combining the experimental data from all phases of dorsal-side or ventral-side inhibition yielded the corresponding PRCs (**Figs. 2.5A** and **2.5B**, respectively), from which we found that

both PRCs show a peak in the phase range during which the bending side is illuminated but shows no significant phase shift in the other phase range. The experimental observations are qualitatively consistent with model predictions.

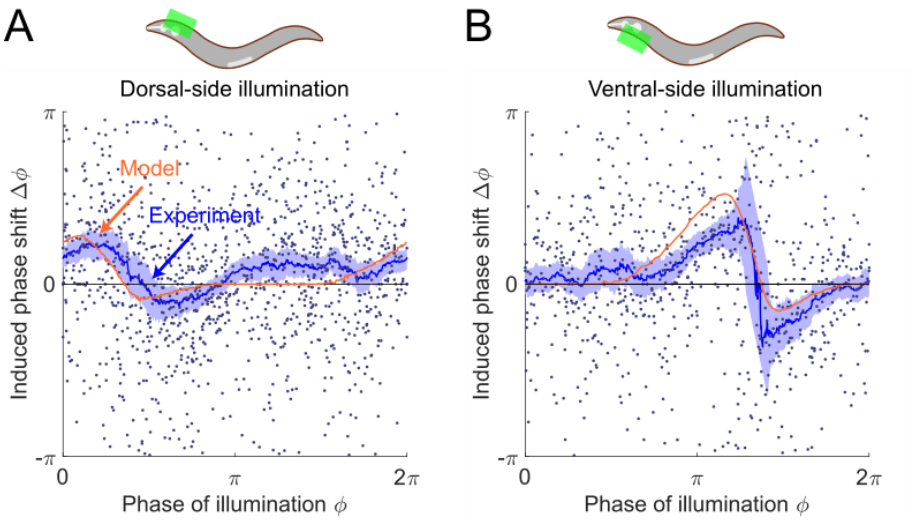

Figure 2.5. The model predicts phase response curves with respect to single-side muscle inhibitions.

(A) (*Upper*) a schematic indicating a transient inhibition of body wall muscles of the head on the dorsal side. (*Lower*) the corresponding PRC measured from experiments (blue, 576 trials using 242 worms) and model (orange).

(B) (*Upper*) a schematic indicating a transient inhibition of body wall muscles of the head on the ventral side. (*Lower*) the corresponding PRC measured from experiments (blue, 373 trials using 176 worms) and model (orange). For the two experiments, each point indicates a single illumination (0.1 s duration, 532 nm wavelength) of one worm. Experimental curves were obtained using a moving average along the *x*-axis with 0.16π in bin width. Filled area of each experimental curve represents 95% confidence interval with respect to each bin of data points.

We found that the PRC of dorsal-side illumination shows a smaller paralytic response than that of ventral-side illumination. This discrepancy may be due to different degrees of paralysis achieved during ventral vs. dorsal illumination (**Fig. S2.11**), possibly due to differences in levels of opsin expression and/or membrane localization. We therefore modulated the parameter for describing degree of paralysis when simulating the PRC of the dorsal-side illumination to qualitatively account for this discrepancy (see *Methods*).

OUR MODEL IS CONSISTENT WITH THE DEPENDENCE OF WAVE AMPLITUDE AND FREQUENCY ON EXTERNAL LOAD

C. elegans can swim in water and crawl on moist surfaces, exhibiting different undulatory gaits characterized by different frequency, amplitude, and wavelength (**Fig. 2.6A**). Previous studies (Berri et al., 2009; Fang-Yen et al., 2010) have shown that increasing viscosity of the medium induces a continuous transition from a swimming gait to a crawling gait, characterized by a decreasing undulatory frequency (**Fig. 2.6C**) and an increasing curvature amplitude (**Fig. 2.6D**). We asked whether our model is consistent with this load-dependent gait adaptation.

We incorporated the effect of external viscosity into our model through the bending relaxation time constant τ_u (see *Methods*). We ran our model to determine the dependence of model output on viscosity with varying viscosity η. We found that model results for frequency and amplitude dependence on viscosity of the external medium are in quantitative agreement with previous experimental results (Fang-Yen et al., 2010) (**Figs. 2.6C,D**).

We sought to develop an intuitive understanding of how the model output changes with increasing viscosity. We recall that the model generates a proprioceptive feedback variable in the form $P = K + b\dot{K}$ (**Fig. 2.3A**), and that the active muscle moment in our model undergoes a change of sign upon the proprioceptive feedback reaching either of two thresholds, P_{th} and $-P_{th}$. As the viscosity increases, one expects that a worm will perform a slower undulation due to the

increase in external load. That is, the term $b\dot{K}$ becomes smaller. To compensate for this effect, the worm needs to undulate with a larger curvature amplitude to maintain the same level of proprioceptive feedback.

Next, we asked how the PRC depends on external viscosity. Model simulations with three different viscosities produced PRCs with similar sawtooth shape but with sharp transitions delayed in phase as the external viscosity increases (**Fig. 2.6F**). We also measured PRCs from optogenetic inhibition experiments in solutions of three different viscosities (**Fig. 2.6G**). Comparing the relative locations of the transitions in PRCs between the model and the data, our prediction also quantitatively agrees with the experimental results.

These results further support the model's description of how undulatory dynamics are modulated by the external environment.

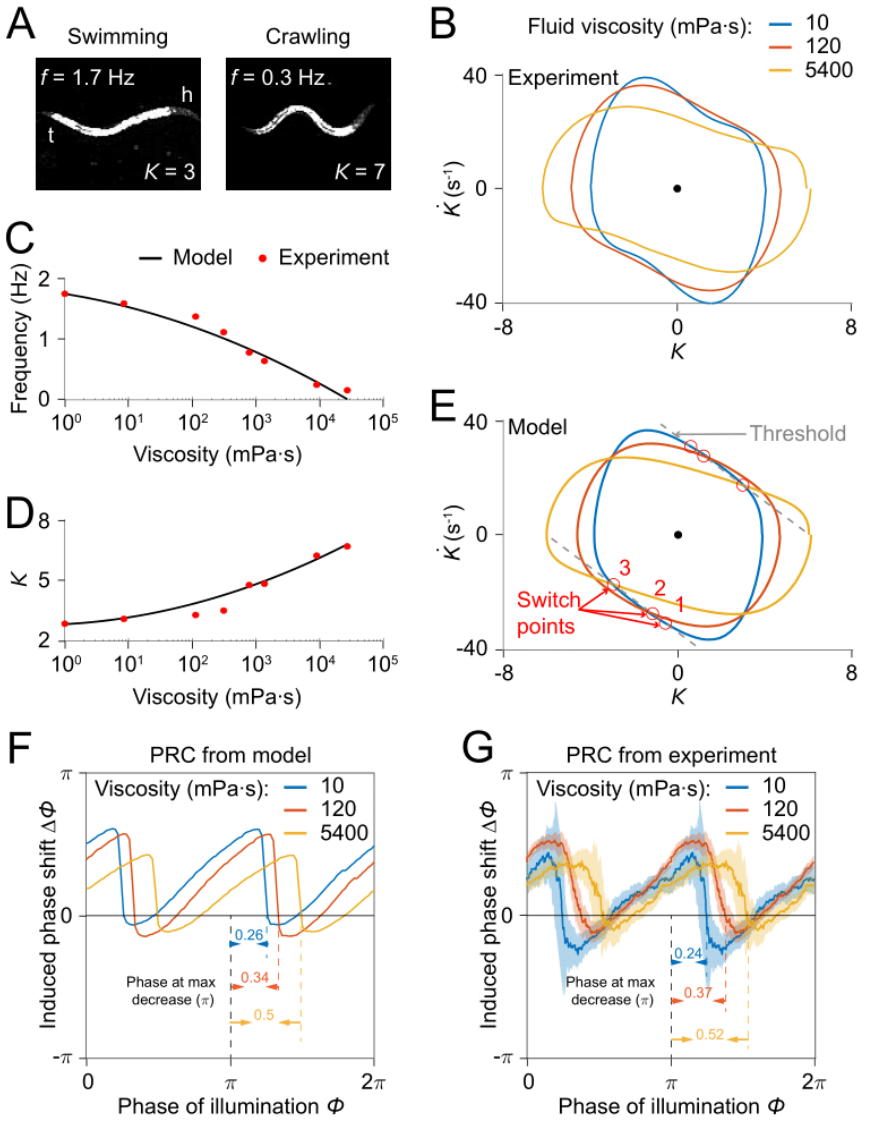

Figure 2.6. Model reproduces *C. elegans* gait adaptation to external viscosity.

(A) Dark field images and the corresponding undulatory frequencies and amplitudes of adult worms (*left*) swimming in NGM buffer of viscosity 1 mPa·s, (*right*) crawling on agar gel surface. The worm head is to the right in both images.

(B) Phase portrait graphs measured from worm forward movements in fluids of viscosity 10

mPa·s (blue, 3528 cycles using 50 worms), 120 mPa·s (red, 5050 cycles using 116 worms), and 5400 mPa·s (yellow, 1364 cycles using 70 worms).

(C,D) The model predicts the dependence of undulatory frequency (C) and curvature amplitude (D) on external viscosity (black) that closely fit the corresponding experimental observations (red).

(E) Phase portrait graphs predicted from the model in three different viscosities (same values as in B). Grey dashed lines indicate threshold lines for dorsoventral bending. The intersections (red circles 1, 2, 3) between the threshold line and phase portrait graphs are switch points for undulations in low, medium, high viscosity, respectively.

(F) Theoretically predicted PRCs in fluids of the three different viscosities show that PRC will be shifted to the right as the viscosity of environment increases.

(G) PRCs measured from optogenetic inhibition experiments in the three viscosities. Experimental PRCs were obtained using a moving average along the *x*-axis with 0.16π in bin width and filled areas are 95% confidence interval. The tendency of shift observed in experimental PRCs verified the model prediction.

EVALUATION OF ALTERNATIVE OSCILLATOR MODELS

Although our computational model agrees well with our experimental results, we asked whether other models could also explain our findings. We examined three alternative models based on well-known mathematical descriptions of oscillators (van der Pol, Rayleigh, and Stuart-Landau oscillators) and compared them with our original threshold-switch model and with our experimental data.

First, we tested the van der Pol oscillator, the first relaxation oscillator model (Van der Pol, 1926) which has long been applied in modeling neuronal dynamics (FitzHugh, 1961; Nagumo et al., 1962). It is based on a second-order differential equation for a harmonic oscillator with a

nonlinear, displacement-dependent damping term (see *Methods*). By choosing a set of appropriate parameters, we found that the free-running waveform and phase plot of the van der Pol oscillator are highly asymmetric, but in an inverted manner (**Fig. S2.12B,F**), compared with the experimental observations (**Figs. 2.1C,D**). Transiently perturbing the system with the bell-shaped modulatory function over all phases within a cycle produced a similar sawtooth-shaped PRC as that observed experimentally (**Fig. S2.12N**). However, the perturbed system was found to recover toward its limit cycle with a much slower rate than that of the experiments (**Fig. S2.12J**). Simulations of single-side muscle inhibitions to the system produced single-sawtooth-shaped PRCs similar to those found experimentally (**Fig. S2.13B,F**).

Next, we examined the Rayleigh oscillator, another relaxation oscillator model which was originally proposed to describe self-sustained acoustic vibrations such as vibrating clarinet reeds (Rayleigh, 1896). It is based on a second-order differential equation with a nonlinear, velocity-dependent damping term and it can be obtained from the van der Pol oscillator via a variable differentiation and substitution (see *Methods*). From its free-running dynamics, we observed that the system exhibits a highly asymmetric waveform and phase plot that are similar to the experimental observations (**Fig. S2.12C,G**). Additionally, the Rayleigh oscillator also produces similar sawtooth-shaped PRCs with respect to transient muscle inhibitions of both sides (**Fig. S2.12O**), dorsal side (**Fig. S2.13C**), and ventral side (**Fig. S2.13G**), respectively, and system's recovery rate after the perturbation was shown to be similar to that of the experiments (**Fig. S2.12K**).

Finally, we considered the Stuart-Landau oscillator, a commonly used model for the analysis of neuronal synchrony (Acebrón et al., 2005). Its nonlinearity is based on a negative damping term which depends on the magnitude of the state variable defined in a complex domain (see *Methods*). The negative damping of the system constantly neutralizes the positive damping on a

limit cycle, making its free-running dynamics a harmonic oscillation which shows a sinusoidal waveform (**Fig. S2.12D,H**). Moreover, PRCs with respect to transient muscle inhibitions are constant with respect to phase (**Fig. S2.12P**), contrary to the experiments.

We compared the results of our models with the experimental results. In the van der Pol oscillator, the free-running waveform displays a different asymmetry (**Fig. S2.12B,F**) compared with the experimental observations and the perturbed system was shown to recover toward its limit cycle with a much slower rate than that of the experiments (**Fig. S2.12J**). The Rayleigh oscillator reproduces a free-running waveform similar to experimental ones (**Fig. S2.12C,G**) and its recovery rate toward limit cycle upon perturbation was close to that of the experiments (**Fig. S2.12K**). However, its PRC (**Fig. S2.12O**) showed weaker agreement with the experimental PRC compared with the threshold-switch model (**Fig. S2.12M**) or the van der Pol model (**Fig. S2.12N**). Of all the models tested, the threshold-switch model showed the least mean-square error with the PRC data (**Fig. S2.12M-P**). We conclude that of these models, our threshold-switch model produced the best overall agreement with experiments.

We also found that two important experimental findings, the nonsinusoidal free-moving dynamics and the sawtooth-shaped PRCs can be achieved in our original model, the van der Pol and Rayleigh oscillators, which are all relaxation oscillators, but not in the Stuart-Landau oscillator, which is not a relaxation oscillator. Taken together, these results are consistent with the idea that a relaxation oscillation mechanism may underlie *C. elegans* motor rhythm generation.

DISCUSSION

In this study, we used a combination of experimental and modeling approaches to probe the mechanisms underlying the *C. elegans* motor rhythm generation.

Our model can be compared to those previously described for *C. elegans* locomotion. Previous detailed models of *C. elegans* locomotion have employed a relatively large number of

free parameters (up to 40 (Boyle et al., 2012; Karbowski et al., 2008)). In our work, we sought to develop a compact phenomenological model to describe an overall mechanism of rhythm generation but not the detailed dynamics of specific circuit elements. To improve predictive power, we aimed to minimize the number of free parameters used in the model. Our model has only 5 free parameters, yet accurately describes a wide range of experimental findings including the nonsinusoidal dynamics of free locomotion, phase response curves to transient paralysis, and dependence of frequency and amplitude on external viscosity.

Our phase portrait analysis of worm's free locomotory dynamics has described a new method for measuring the bending relaxation time scale τ_u and the muscle moment transition time scale τ_m (see *Methods* for details), which may be compared with previous studies of worm biomechanics (Berri et al., 2009; Fang-Yen et al., 2010) and neurophysiology (Milligan et al., 1997). Fang-Yen et al. (Fang-Yen et al., 2010) measured a linear relationship between the bending relaxation time scale and the external viscosity by deforming the worm body in Newtonian fluids with varied viscosities in the range 1 to 25 mPa·s. Through an extrapolation based on that linear relationship, the relaxation time scale in 17% dextran NGM fluid (approximately 120 mPa·s in viscosity) is estimated to be $\approx 282\ ms$, which is quite close to our measured result, $\tau_u \approx 260\ ms$. Furthermore, our measurement of the muscle moment transition time scale ($\tau_m \approx 100$) is consistent with previously measured value for muscle time scale (Milligan et al., 1997) that has also been widely adopted for other detailed models of nematode locomotion (Boyle et al., 2012; Bryden and Cohen, 2008; Butler et al., 2015; Chen et al., 2011; Denham et al., 2018; Izquierdo and Beer, 2018; Johnson et al., 2021; Karbowski et al., 2008; Olivares et al., 2021; Wen et al., 2012).

In our model the mechanism for generating rhythmic patterns can be characterized by a 'relaxation oscillation' process which contains two alternating sub-processes on different time

scales: a long relaxation process during which the motor system varies toward an intended state due to its biomechanics under a constant active muscle moment, alternating with a rapid period during which the active muscle moment switches to an opposite state due to a proprioceptive thresholding mechanism.

The term ‘relaxation oscillation’, as first employed by van der Pol, describes a general form of self-sustained oscillatory system with intrinsic periodic relaxation/decay features (Van der Pol, 1926). The Fitzhugh-Nagumo model (FitzHugh, 1961; Nagumo et al., 1962), a prototypical model of excitable neural systems, was originally derived by modifying the van der Pol relaxation oscillator equations. These and similar relaxation oscillators have been characterized in various dynamical systems in biology and neuroscience (Izhikevich, 2007). For example, the dynamics exhibited from the action potentials of barnacle muscles in their oscillatory modes were found to yield ‘push-pull’ relaxation oscillation characteristics (Morris and Lecar, 1981). The beating human heart was found to behave as a relaxation oscillator (VAN DER POL, 1940). Several studies of walking behavior in stick insects (Bässler, 1977; Cruse, 1976; Graham, 1985; Wendler, 1968) proposed that the control system for rhythmic step movements constitutes a relaxation oscillator in which the transitions between leg movements is determined by proprioceptive thresholds.

Key properties shared by these relaxation oscillators are that their oscillations greatly differ from sinusoidal oscillations and that they all consist of a certain feedback loop with a ‘discharging property’. They contain a switch component that charges an integrating component until it reaches a threshold, then discharges it again (Nave, 1995), then repeats. Many relaxation oscillators, including the van der Pol and Rayleigh models, exhibit sawtooth-shaped phase response curves (VAN DER POL, 1940). As shown in our experimental and model results (**Fig. S2.12**), all the above properties have been revealed in the dynamics of *C. elegans* locomotive behavior, consistent with the idea that the worm’s rhythmic locomotion also results from a type of

relaxation oscillator.

In our computational model, a proprioceptive component sensing the organism's changes in posture is required to generate adaptive locomotory rhythms. What elements in the motor system could be providing this feedback? Previous studies have suggested that head and body motor neurons, including the SMDD head motor neurons and the B-type motor neurons, have proprioceptive capabilities (Wen et al., 2012; Yeon et al., 2018) and may also be involved in locomotory rhythm generation (Fouad et al., 2018; Gao et al., 2018; Kaplan et al., 2020; Xu et al., 2018). This possibility is consistent with an earlier hypothesis that the long undifferentiated processes of these cholinergic neurons may function as proprioceptive sensors (White et al., 1986). In particular, recent findings (Yeon et al., 2018) have revealed that SMDD neurons directly sense head muscle stretch and regulate muscle contractions during oscillatory head bending movements.

In our model, the proprioceptive feedback variable depends on both the curvature and the rate of change of curvature. Many mechanoreceptors are sensitive primarily to time derivatives of mechanical strain rather than strain itself; for example, the *C. elegans* touch receptor cells exhibit such a dependence (Eastwood et al., 2015; O'Hagan et al., 2005). The ability of mechanosensors to sense the rate of change in *C. elegans* curvature has been proposed in an earlier study (Butler et al., 2015) in which it was hypothesized that the B-type motor neurons might function as a proprioceptive component in this manner. Mechanosensors encoding a simultaneous combination of deformation and velocity have been observed in mammalian systems including rapidly-adapting (RA) and intermediate-adapting (IA) sensors in the rat dorsal root ganglia (Rugiero et al., 2010). Proprioceptive feedback that involves a linear combination of muscle length and velocity was also suggested by a study of *C. elegans* muscle dynamics during swimming, crawling, and intermediate forms of locomotion (Butler et al., 2015). In our

phenomenological model, the motor neuron constituent may represent a collection of neurons involved in motor rhythm generation. Therefore, the proprioceptive function posited by our model might also arise as a collective behavior of curvature-sensing and curvature-rate-sensing neurons.

Further identification of the neuronal substrates for proprioceptive feedback may be possible through physiological studies of neuron and muscle activity using Ca^{2+} or voltage indicators. Studies of the effect of targeted lesions and genetic mutations on the phase response curves will also help elucidate roles of specific neuromuscular components within locomotor rhythm generation.

In summary, our work describes the dynamics of the *C. elegans* locomotor system as a relaxation oscillation mechanism. Our model of rhythm generation mechanism followed from a quantitative characterization of free behavior and response to external disturbance, information closely linked to the structure of the animal's motor system (Gutkin et al., 2005; Nadim et al., 2012; Schultheiss et al., 2011; Smeal et al., 2010). Our findings represent an important step toward an integrative understanding of how neural and muscle activity, sensory feedback control, and biomechanical constraints generate locomotion.

METHODS

Table 2.1. Key resources table for Chapter 2

Reagent type (species) or resource	Designation	Source or reference	Identifiers	Additional information
Strain, strain background (*E. coli*)	OP50	CGC	Fang-Yen Lab Strain Collection: OP50 RRID:WB-STRAIN:WBStrain00041971	OP50
Strain,	YX148	Fouad et	Fang-Yen Lab Strain	*qhIs1[Pmyo-*

strain background (*C. elegans*)		al., 2018	Collection: YX148	*3::NpHR::eCFP; lin-15(+)]; qhIs4[Pacr-2::wCherry]*
Strain, strain background (*C. elegans*)	YX119	Fouad et al., 2018	Fang-Yen Lab Strain Collection: YX119	*qhIs1[Pmyo-3::NpHR::eCFP; lin-15(+)]; unc-49(e407)*
Strain, strain background (*C. elegans*)	YX205	Leifer et al., 2011	Fang-Yen Lab Strain Collection: YX205	*hpIs178[Punc-17::NpHR::eCFP ; lin-15(+)]*
Strain, strain background (*C. elegans*)	WEN001	Fouad et al., 2018	Fang-Yen Lab Strain Collection: WEN001	*wenIs001[Pacr-5::Arch::mCherry ; lin-15(+)]*

WORM STRAINS AND CULTIVATION

All worms used in this study were cultivated on NGM plates with *Escherichia coli* strain OP50 at 20°C using standard methods (Sulston and Hodgkin, 1988). Strains used and the procedures for optogenetic experiments are described in the Key resources table for Chapter 2. All experiments were performed with young adult (< 1 day) hermaphrodites synchronized by hypochlorite bleaching.

For optogenetic experiments, worms were cultivated in darkness on plates with OP50 containing the cofactor all-*trans* retinal (ATR). For control experiments and free-moving experiments, worms were cultivated on regular OP50 NGM plates without ATR. To make OP50-ATR plates, we added 2 µL of a 100 mM solution of ATR in ethanol to an overnight culture of 250 µL OP50 in LB medium and used this mixture to seed 6 cm NGM plates.

LOCOMOTION ANALYSIS

To analyze worm locomotion in viscous fluids, we placed worms in dextran solutions in chambers

formed by a glass slide and a coverslip separated by 125-μm-thick polyester shims (McMaster-Carr 9513K42). For viscosity-dependence experiments, we used 5%, 17%, and 35% (by mass) solutions of dextran (Sigma-Aldrich D5376, average molecular weight 1,500,000-2,800,000) in NGMB. These solutions were measured to have viscosities of 10, 120, and 5400 mPa·s (Fang-Yen et al., 2010), respectively. We used a 17% dextran solution for all other experiments. NGMB consists of the same components as NGM media (Stiernagle, 2006), but without agar, peptone, or cholesterol.

We recorded image sequences using a custom-built optogenetic targeting system based on a Leica DMI4000B microscope under 10X magnification with dark field illumination provided by red LEDs. Worm images were recorded at 40 Hz with a sCMOS camera (Photometrics optiMOS). We used custom-written C++ software (Fouad et al., 2017) to perform real-time segmentation of the worm during image acquisition. The worm was identified in each image by its boundary and centerline calculated from a binary image. Anterior-posterior orientation was noted visually during the recording. Segmentation information, including coordinates of the worm boundary and centerline, was saved to disk along with the corresponding image sequences.

Post-acquisition image analysis was performed using a custom MATLAB (Mathworks) similar to previous reports (Fouad et al., 2017). The worm centerline of each image was smoothed using a cubic spline fit. We calculated curvature κ as the dot product between the unit normal vector to the centerline and derivative of the unit tangent vector to the centerline with respect to the body coordinate. Dimensionless curvature K was calculated as the product of κ and the worm body length L represented by length of the centerline. Since the segmentation was relatively noisy at the tips of the worm, we excluded curvature in the anterior and posterior 5% of the body length. The worm's direction of motion was identified by calculating the gradients in the curvature over time and body coordinate, and image sequences in which the worm performed consistent forward

movement (lasting at least 4 s) were selected for analysis. The anterior curvature $K(t)$ was defined as the average of the dimensionless curvature over body coordinate 0.1-0.3; this range avoided high frequency movements of the anterior tip of the animal.

To quantify oscillatory dynamics during forward locomotion, we identified undulatory cycles from the time sequence of anterior curvature in each worm. Local extrema along each sequence were identified and portions between consecutive local maxima were defined as individual cycles. To minimize the effects of changes in the worm's frequency, we excluded cycles whose period deviated by more than 20% from the average period of all worms' undulations in each experimental session.

For the ease of computing average dynamics, we converted individual cycles from a time-dependent to a phase-dependent curvature by uniformly rescaling each cycle to a phase range of 2π. The averaged curvature within one cycle was then computed by averaging all individual cycles in the phase domain: $\langle K(\phi)\rangle = \sum_{i=1}^{N} K_i(\phi)\,/N$. Similarly, the averaged phase derivative of curvature within one cycle was calculated as $\langle dK/d\phi\rangle = \sum_{i=1}^{N}(dK_i/d\phi)\,/N$.

STABILITY OF THE WORM'S HEAD OSCILLATION

To examine the stability of the worm's head oscillation during forward locomotion, we analyzed head oscillations of worms that were optogenetically perturbed with 0.1 s muscle inhibitions and estimated their recovery dynamics after being deviated from the normal oscillation due to the perturbation.

To illustrate the oscillation dynamics, we use a two-dimensional variable, i.e. $\boldsymbol{x} = (K, \xi\dot{K})$ in the unit of curvature where $\xi = 0.135\ s$ is a scaling factor. In **Fig. S2.3** we depicted the closed trajectory (black) in the plane spanned by the variables K and $\xi\dot{K}$ for the head oscillation of unperturbed moving worms (this coordinate plane is in fact a linearly scaled version of the phase

plane spanned by the variables K and $\dot{K}$), which we call as the normal cycle of the worm's head oscillation.

Next, we defined an amplitude variable d that represents the normalized deviation to the normal cycle. If the oscillator is stable, the closed orbit for the unperturbed dynamics is usually called the stable limit cycle. Here, we stick to the notion of normal cycle instead of using 'limit cycle' to avoid any confusion on the stability of the worm's head oscillation. For any phase state of an individual oscillation, the normalized deviation to the normal cycle is defined as $d(\phi) = (D(\phi) - D^C(\phi))/D^C(\phi)$. Here, $D(\phi)$ is distance to the center of oscillation on the phase plane, which is set to the origin, such that $D(\phi) = \sqrt{K(\phi)^2 + \left(\xi\dot{K}(\phi)\right)^2}$ where ϕ denotes the phase value of the current state estimated by the four-quadrant inverse tangent of the variable pair $(K, \xi\dot{K})$. In this expression $D^C(\phi)$ denotes the distance to the center of oscillation that is evaluated exactly on the normal cycle at phase ϕ.

Using the deviation to the normal cycle to describe the amplitude of the worm's head oscillation, we collected the amplitude dynamics over time for all periods of the worm's head oscillations during which no illumination pulse occurs, that is, all periods of locomotion between two consecutive illumination pulses. We grouped the amplitude dynamics into bins according to their initial amplitudes and then calculated the collective amplitude dynamics for each bin. As shown in **Fig. S2.2**, the collective amplitude variable d converges to zero after roughly 0.5 s regardless of the initial amplitude. This result indicates that the worm's head oscillation returns to its normal oscillation after being perturbed and that the normal cycle may represent a stable limit cycle for the oscillation.

PHASE ISOCHRON MAP AND VECTOR FIELD FOR THE WORM'S HEAD OSCILLATION

On the normal cycle we define the phase of the oscillation as $\phi^C(t) = \omega_0 \cdot t_{mod T_0}$, where $\omega_0 =$

$2\pi/T_0$ is the angular frequency of normal oscillation and we determined the initial phase ($\phi^C = 0$) to be when K reaches local maximum (or $\boldsymbol{x} = (K_{max}, 0)$) and hence $\phi^C = \pi$ to be when K reaches local minimum (or $\boldsymbol{x} = (K_{min}, 0)$). In this way, we parameterized the normal cycle by defining a bijective map between phases and state points $\Phi(\boldsymbol{x}^C) = \phi^C$.

The map $\Phi(\boldsymbol{x}) = \phi$ has been well defined for all the state points on the normal cycle C. We next estimate the phases for points off the normal cycle. By definition (Izhikevich, 2007), if $\boldsymbol{x}_0$ is a point on the normal cycle and $\boldsymbol{y}_0$ is a point off the normal cycle, then $\boldsymbol{y}_0$ will have the same phase as $\boldsymbol{x}_0$ if the trajectory starting at the initial point $\boldsymbol{y}_0$ off the normal cycle converges to the trajectory starting at the initial point $\boldsymbol{x}_0$ on the normal cycle as time goes to infinity. Here, we define the set of all state points off the normal cycle having the same phase as a point $\boldsymbol{x}_0$ on the normal cycle as the isochron (Winfree, 2001) for phase $\phi_0 = \Phi(\boldsymbol{x}_0)$.

In our analysis, it was not possible to define an isochron according to the theoretical definition since data were always recorded in a finite time period during experiments. We used an alternative way to estimate all isochrons on the phase plane for the worm's head oscillation. For each individual trial of illumination, we observed that, due to the optogenetic inhibition, the variable $\dot{K}$ quickly decayed toward zero value immediately after the illumination and then recovered after approximately 0.3 s as the oscillation converged to a normal oscillation. Therefore, by finding the local minimal of $\dot{K}$ immediately after each illumination pulse, we located the point at which the paralyzing effect is just removed and after which the oscillation starts a free resumption to normal oscillation. We call this point the "notch point" $\boldsymbol{x}_N$ as it can be clearly seen from the phase plot (as shown in **Figs. 2.2E, 2.2H**, and **2.2K**). After the notch point $\boldsymbol{x}_N$, the oscillation will proceed to its next phase states $\boldsymbol{x}(\phi = 2\pi)$ and $\boldsymbol{x}(\phi = \pi)$ (or vice-versa), both of which can be easily identified through peak finding from the curvature dynamics K. Hence, we obtained two sub-trajectories from the oscillation: one $\boldsymbol{x}_N \to \boldsymbol{x}(\phi = 2\pi)$, and the other $\boldsymbol{x}_N \to \boldsymbol{x}(\phi =$

π). Next to determining the timing of the notch point $t(\boldsymbol{x}_N)$, we determined the phase of the notch point in the following steps: (1) we computed the phase value of the state at which the illumination occurs, $\phi(\boldsymbol{x}_{illum})$, using the method described in the next subsection; (2) then we computed the phase of the state on the normal cycle at the timing of the notch point $t(\boldsymbol{x}_N)$ as if the perturbation had not been applied, which is $\phi(\boldsymbol{x}_N^C) = \left(\phi(\boldsymbol{x}_{illum}) + \omega_0\big(t(\boldsymbol{x}_N) - t(\boldsymbol{x}_N)\big)\right)_{mod\ 2\pi}$; (3) we calculated the induced phase shift $PRC(t_{illum})$ and the phase of the notch point is $\phi(\boldsymbol{x}_N) = \phi(\boldsymbol{x}_N^C) - PRC(t_{illum})$. After obtaining the sub-trajectories $\boldsymbol{x}_N \to \boldsymbol{x}(\phi = 2\pi)$ and $\boldsymbol{x}_N \to \boldsymbol{x}(\phi = \pi)$ and calculating the phase of $\boldsymbol{x}_N$, we then estimated the phase values for all the points within each of the two sub-trajectories through linear interpolation.

Following the above steps, we calculated the phase values for all the state points on the phase plane that have been recorded from the optogenetic experiments. We then applied a 2-D moving average (using the angular statistics method) for the obtained phase values over the phase plane to smooth out the isochron map. Finally, we used a linear 2-D interpolation to obtain a phase isochron map with a finer resolution as shown in **Fig. S2.3**.

To compute the vector field of the worm's head oscillation, we collected all the sub-trajectories $\boldsymbol{x}_N \to \boldsymbol{x}(\phi = 2\pi)$ and $\boldsymbol{x}_N \to \boldsymbol{x}(\phi = \pi)$ that are defined above and took derivative of the trajectories with respect to time. Thus, by collecting all the phase states $(K, c\dot{K})$ and their corresponding time derivatives $(dK/dt, d(c\dot{K})/dt)$ that describe the tangent vectors of trajectories, we generated the raw form of vector field for the worm's head oscillation. Again, we applied a 2-D moving average for the raw outcome over the phase plane to smooth out the vector field. We used a linear 2-D interpolation to obtain a vector field with an appropriate number of quivers to be displayed (**Fig. S2.3**).

PHASE RESPONSE ANALYSIS

To generate phase response curves (PRCs) from optogenetic inhibition experiments, each trial's illumination phase ϕ, as well as the induced phase shift F, were calculated. To calculate the two variables, the animal's phase of oscillation was estimated based on timings of local extrema identified from the time-varying curvature profiles via a peak finding method. Specifically, (i) the occurrence of illumination of the trial was set to $t = T_{illum}$; $t = 0$ was set at the beginning of each experiment. (ii) Around the illumination, timings of the two local maxima of curvature immediately before and after were identified as the two zero-phase points of the oscillation before and after the illumination, respectively. Here, the timings are denoted as TZ_{-2}, TZ_{-1}, TZ_{+1}, and TZ_{+2}, in the ascending order of time. (iii) Similarly, timings of the two local minima of curvature immediately before and after the illumination were identified as the two half-cycle-phase points before and after the illumination, respectively. Here, the timings are denoted as TH_{-2}, TH_{-1}, TH_{+1}, and TH_{+2}, in the ascending order of time. (iv) With these measurements, cycle period T_0 was computed as $T_0 = (TZ_{+2} - TZ_{+1} + TZ_{-1} - TZ_{-2} + TH_{+2} - TH_{+1} + TH_{-1} - TH_{-2})/4$, so the angular frequency of undulation $\omega_0 = 2\pi/T_0$ (T_0 was computed as the average of differences of adjacent local maxima/minima before and after illumination; multiple cycles were used here to reduce noise). In addition, the illumination phase ϕ of each individual trial was computed as $\phi_u = \omega_0(T_{illum} - TZ_{-1})_{mod\, T_0}$, $\phi_l = \omega_0(T_{illum} - TH_{-1} + T_0/2)_{mod\, T_0}$, and the corresponding phase shift F was computed as $F_u = \omega_0(TZ_{+1} - TZ_{-1})_{mod\, T_0} - \pi$, $F_l = \omega_0(TH_{+1} - TH_{-1} + T_0/2)_{mod\, T_0} - \pi$. (Here, phase of illumination and the corresponding phase shift were computed twice using zero (subscripted with u) and half-cycle (subscripted with l) phase points as references, respectively.)

We generated 2-D scatter plots for all trials with illumination phase as *x* coordinate and the corresponding phase shift as *y* coordinate. To visualize the distribution of the scatter points we generated bivariate histogram plots by grouping the data points into 2-D bins with 25 bins on both dimensions covering the range $[0,2\pi]$ for *x* and range $[-\pi,\pi]$ for *y*. To indicate average tendency

of phase shift depending on phase of illumination, we calculated a mean-curve representation of PRCs via a moving average operation. In this process, each mean was calculated over a sliding window of width 0.16π along the direction of ϕ from 0 to 2π. The 95% confidence interval relative to each window of data points was also computed and an integral number of them were displayed as filled area around the PRC. Through the computation, all statistical calculations followed the rules of directional statistics (Fisher et al., 1993) since ϕ and F are circular variables defined in radians.

PHASE RESPONSE CURVES FROM PERTURBATIONS OF OTHER BODY REGIONS

We asked how phase responses for the other regions of the body would compare to that of the anterior region. We conducted optogenetic experiments that inhibited *Pmyo-3::NpHR* transgenic worms by transiently illuminating 0.1-0.3 (anterior), 0.4-0.6 (middle), and 0.6-0.8 (posterior) of the body length, respectively. We found that the amplitude of the sawtooth feature of PRC tends to decrease as the perturbation occurs further from the head (**Fig. S2.6A,E,I**). We also noticed that, for the same perturbed region, the PRC shape remains unaffected regardless of the region at which the dynamics were analyzed (see **Fig. S2.6A-C, D-F, G-I**, respectively), suggesting that posterior regions of a freely moving worm follow their anterior neighbors with a constant phase offset. Taken together, these results suggest that a main rhythm generator may operate near the head of the worm to produce primary oscillations during forward locomotion. The sawtooth-shape feature of the PRC becomes stronger if the perturbation hits closer to the rhythm generator (**Fig. S2.6A**) or becomes weaker if the perturbation indirectly affects it (**Fig. S2.6E,I**)

THE RELAXATION OSCILLATOR MODEL FOR LOCOMOTOR WAVE GENERATION

We first developed a relaxation oscillator model to simulate the rhythm generation during *C. elegans* forward locomotion. In this model, we incorporated a novel neuromuscular mechanism with a previously described biomechanical framework (Fang-Yen et al., 2010). Here, we only

simulated the bending rhythms generated from the head region; the wave propagation dynamic is out of the scope of our study. Our phenomenological model does not contain detailed activities of individual neurons but focus on describing key neuromuscular mechanisms and their contributions to the rhythm generation.

To produce model variables that can be directly compared with experimental observations of moving animals, a biomechanical framework was first developed to describe worm's behavioral dynamics in its external environments. Following previous derivations for *C. elegans* biomechanics (Fang-Yen et al., 2010), the relationship between animal behavioral outputs and the active muscle moments can be described as follows:

$$C_N \frac{\partial y}{\partial t} + a \frac{\partial^2 \kappa}{\partial s^2} + a_v \frac{\partial}{\partial t}\left(\frac{\partial^2 \kappa}{\partial s^2}\right) = m_a. \qquad [\mathbf{S2.1}]$$

In **Eqn. S2.1**, the first term indicates the external viscous force that is transverse to the body segment where C_N is the coefficient of viscous drag to the transverse movement and y denotes the lateral displacement of body segment; the second term indicates the internal elastic force where a is the bending modulus of the worm body; the third term indicates the internal viscous force where a_v is the coefficient of the body internal viscosity. On the right side of **Eqn. S2.1** is the active muscle moment m_a.

Taking the second partial derivative with respect to body coordinate s on both sides of **Eqn. S2.1** and, using the linear relation under the small-amplitude approximation, $\kappa \approx y_{ss}$, we arrive at:

$$C_N \frac{\partial \kappa}{\partial t} + a \frac{\partial^4 \kappa}{\partial s^4} + a_v \frac{\partial}{\partial t}\left(\frac{\partial^4 \kappa}{\partial s^4}\right) = \frac{\partial^2 m_a}{\partial s^2}. \qquad [\mathbf{S2.2}]$$

Under the assumptions of small-amplitude undulations and a fixed wavelength λ down the worm body, κ can be considered as a travelling sinusoidal wave with a small deviation, $\kappa(s,t) =$

$\kappa_0 \sin(2\pi s/\lambda - \omega t) + \delta$, which leads to an approximation, $\kappa_{ssss} \approx (2\pi/\lambda)^4 \kappa$. Plugging this approximation into **Eqn. S2.2** while keeping s fixed, after some rearrangement, one gets:

$$\kappa + \frac{C_N \left(\frac{\lambda}{2\pi}\right)^4 + a_v}{a} \dot{\kappa} = \frac{\lambda^4}{(2\pi)^4 a} \frac{\partial^2 m_a}{\partial s^2}. \quad [\mathbf{S2.3}]$$

In terms of the dimensionless curvature $K = \kappa \cdot L$ and dimensionless muscle moment

$$M_a = \frac{\lambda^4 L}{(2\pi)^4 a} \frac{\partial^2 m_a}{\partial s^2}, \quad [\mathbf{S2.4}]$$

we can rewrite **Eqn. S2.3** as:

$$K + \tau_u \dot{K} = M_a\,, \quad [\mathbf{S2.5}]$$

where

$$\tau_u = \frac{C_N \left(\frac{\lambda}{2\pi}\right)^4 + a_v}{a}, \quad [\mathbf{S2.6}]$$

and we note that **Eqns. S2.5** and **S2.6** yield **Eqn. 2.1**. In **Eqn. S2.6**, both the wavelength λ and the normal viscous drag coefficient C_N vary with the fluid viscosity η (Berri et al., 2009; Fang-Yen et al., 2010).

The above biomechanical framework in our model treats the worm's body segment as a viscoelastic rod and describes how the body segment bends under the forces provided by the active muscle moment. However, the simulated oscillation in K comes from the rhythmicity of the active muscle moment which originates from the hypothesized neuromuscular mechanism described by the following relaxation-oscillation process:

i. Proprioceptive feedback is sensed as a linear combination of the current curvature value

and the current rate of change of curvature, $P = K + b\dot{K}$ (black curve in **Fig. 2.3D**).

ii. During movement of bending, this proprioceptive feedback is constantly compared with two threshold values P_{th} and $-P_{th}$ (grey dashed bars in **Fig. 2.3D**).

iii. Once the feedback reaches either of the thresholds (the switch points as indicated by red circles in **Fig. 2.3D**), a switch command is initiated (blue square wave in **Fig. 2.3E**).

iv. The switch command triggers the active muscle moment to change toward the opposite saturation value (black curve in **Fig. 2.3E**).

To simulate the switch-triggered muscle transition we used a modified logistic function: $M_a(t) = \pm M_0 \cdot \tanh(t/2\tau_m)$. Here, the plus sign indicates the dorsal-to-ventral muscle moment transition while the minus sign indicates the opposite direction.

To initiate the oscillation in our model we set the system to bend toward the ventral side by setting $M_a|_{t=0} = M_0$ and $K|_{t=0} = 0$. During forward locomotion, the active muscle moment oscillates by undergoing a relaxation oscillation process: a relaxation subperiod during which M_a stays at a saturated bending state (M_0 for ventral bending, $-M_0$ for dorsal bending), alternating between a shorter subperiod during which M_a quickly transits toward the opposite state due to effects described in iii and iv. The bending curvature $K(t)$ which is driven by M_a in an exponential decaying manner (**Eqn. S2.5**) follows the rhythmic activity of M_a, thereby also exhibiting an oscillatory dynamic (**Fig. 2.3B**).

This relaxation oscillator model reproduces two key features of free locomotion that we observed from experiments. First, freely moving worms exhibit nonsinusoidal curvature waveform with an intrinsic asymmetry: bending toward the ventral or dorsal directions occurs slower than straightening toward a straight posture during each locomotory cycle (**Fig. 2.3F**). Second, dynamic of the active muscle moment shows a trapezoidal waveform during forward locomotion (**Fig. 2.1D *Inset*** and **Fig. 2.3E**). These results are independent of external conditions but reflect

intrinsic properties of the neuromuscular mechanisms underlying locomotion rhythm generation.

Note that parameters M_0, τ_u, and τ_m were estimated from data of free locomotion using phase portrait techniques described the following subsection. Parameters b and P_{th} were yet degenerate in this model of free locomotion. Here, we temporarily set $b = 0$ and then set P_{th} such that the oscillatory period predicted by model matched the average period measured from experiments with a minimum squared error:

$$P_{th} = \underset{P_{th}>0}{\operatorname{argmin}} \left(T_{model}(P_{th}) - T_{experiment}\right)^2. \qquad [\mathbf{S2.7}]$$

The nondegeneracy of b and P_{th} was determined by fitting the model to the experimental PRC as described in the later subsection so that all the parameters for the model are provided as $M_0 = 8.45$, $\tau_u = 260\,ms$, $\tau_m = 100\,ms$, $b = 46\,ms$, and $P_{th} = 2.33$.

MEASURING BENDING RELAXATION TIME SCALE AND AMPLITUDE OF ACTIVE MUSCLE MOMENT

To estimate these two parameters, we applied a heuristic method that uses the shape properties of *C. elegans* free-running phase plot (**Fig. 2.1D**). From the curve in the figure, we noticed two 'flat' portions symmetrically distributed at quadrant *I* and *III* on the phase plane. Recalling **Eqn. 2.1** (or **Eqn. S2.5**): $K + \tau_u \cdot \dot{K} = M_a(t)$, the two flat regions indicate that the scaled active muscle moment, $M_a(t)$, is nearly constant during the corresponding time bouts.

We then computed the linear correlation between variables K and $\dot{K}$ to identify the two 'flat' regions and, through linear fits, obtained two linear relations respectively: $\langle K\rangle + \tau_1 \cdot \langle \dot{K}\rangle = M_1$ (region 1) and $\langle K\rangle + \tau_2 \cdot \langle \dot{K}\rangle = M_2$ (region 2). Thus, the bending relaxation time scale τ_u and the amplitude of the scaled active muscle moment are estimated as $\widehat{\tau_u} = (\tau_1 + \tau_2)/2$ and $\widehat{M_0} = (|M_1| + |M_2|)/2$, respectively.

The above method used the phase plot measured from locomotion of worms swimming in a 17% dextran solution (120 mPa·s viscosity) as an example. However, it is also valid for estimating parameters of locomotion in other viscosities.

MEASURING ACTIVE MOMENT TRANSITION TIME SCALE

With τ_u (estimated from the above method), $\langle K \rangle$ and $\langle \dot{K} \rangle$ (measured from locomotion) plugged to the left side of **Eqn. 2.1**, we were able to compute the waveform of the scaled active muscle moment $M_a(t)$ on the right side of **Eqn. 2.1**. As expected and shown in **Fig. 2.1D *Inset***, the curve of $M_a(t)$ is roughly centrally symmetric around point $(T_0/2, 0)$ on the plane, with two plateau portions indicating two saturated states for dorsal and ventral muscle contractions, respectively.

Between the two plateau portions represents a period during which the active muscle moment is undergoing a ventral-to-dorsal (or vice-versa) transition. We used a modified logistic function to model the ventral-to-dorsal muscle moment transition (substituting t with $-t$ for transition in the other direction):

$$M_a(t) = M_0 \cdot \tanh\left(\frac{t}{\tau_m}\right). \qquad [\mathbf{S2.8}]$$

To estimate τ_m, the exponential time constant for the transition of active muscle moment, we took the time derivative of **Eqn. S2.8** and took the absolute value of the resultant:

$$\left|\frac{dM_a}{dt}\right| = \frac{M_0}{\tau_m} \cdot \frac{\exp(2t/\tau_m)}{(1+\exp(2t/\tau_m))^2}. \qquad [\mathbf{S2.9}]$$

We notice that when $t = 0$, the maximum of $|dM_a/dt|$ is achieved and the value is M_0/τ_m. On the other hand, the maximum of $|dM_a/dt|$ can be obtained from the experimental observation by simply finding the peak of $|dM_a/dt|$ curve where $M_a = \langle K(t) \rangle + \widehat{\tau_u} \cdot \langle dK(t)/dt \rangle$. Thus, τ_m can be estimated as:

$$\tau_m = \widehat{M_0} \cdot \left|\frac{dM_a}{dt}\right|_{max}^{-1}. \qquad [\mathbf{S2.10}]$$

PARAMETER ESTIMATION

For our original threshold-switch model, parameters τ_u, τ_m, and M_0 were estimated from free locomotion experiments as described above. These three parameters nearly fully determine the biomechanical framework of *C. elegans* bending movements (governed by **Eqns. S2.5** and **S2.8**). On the other hand, parameters b and P_{th} describe the proprioceptive feedback and the threshold-switch features in our model. Specifically, they characterize two threshold lines, $K + b\dot{K} = \pm P_{th}$ (as shown in **Fig. 2.3C**). The two switch points—defined by the intersection between the phase trajectory and the threshold lines on the phase plane—determine the timing of switches for the active muscle moment (see **Figs. 2.3C-E**). We noted that the model behavioral output of free locomotion is degenerate with respect to these two parameters; the same outcome would be produced if the threshold lines cross the same pair of switch points. To first determine the free-moving dynamic as well as the switch points, we temporarily set $b = 0$ and then set P_{th} such that the oscillatory period defined by model matched the average period measured from the experiments.

To obtain the nondegeneracy of P_{th} and b, we fit our model to the experimental phase response curve using a global optimization procedure. Full procedure for the determination of b and P_{th} is given below.

MODELING WORM OSCILLATION IN VARIED ENVIRONMENTS

Differences in various environments will change only those parameters that are related to contact with external forces whereas parameters related to oscillator's internal properties will not be affected. In terms of the internal parameters of our model, we used values that were previously determined, which are $\tau_m = 100\ ms$, $M_0 = 8.45$, $b = 46\ ms$, $P_{th} = 2.33$. For the exogenous

parameters, only the time constant of undulation, τ_u, varies according to external conditions. According to **Eqn. S2.6**, τ_u is explicitly determined in terms of other physical parameters, including biomechanical parameters measured in previous work (Fang-Yen et al., 2010): the internal viscosity of worm body is measured as $a_v = 5 \cdot 10^{-16} Nm^3 s$; the bending modulus of worm body is measured as $a = 9.5 \cdot 10^{-14} Nm^3$; $C_N = 31\eta$ is the coefficient of viscous drag for movement normal to the body (Katz et al., 1975), where η is the fluid viscosity. According to previous measurements of undulatory wavelengths in different viscous solutions (Fang-Yen et al., 2010), we applied a logarithmic fit to the data points, yielding $\lambda/L = -0.158 \log_{10}(\eta/\eta_0) + 1.5$ for a continuous model realization in undulatory frequency and amplitude. Here, λ is the wavelength and $\eta_0 = 1\ mPa \cdot s$.

ALTERNATIVE MODELS FOR LOCOMTOR WAVE GENERATION

To further evaluate the performance of our original model, we explored three alternative models for simulating locomotory rhythm generation to make comparisons across these models and the experimental observations. Alternative models are based on three previously studied self-oscillator models described by 2-D nonlinear systems: the van der Pol, Rayleigh, and Stuart-Landau oscillators.

First, we developed a model oscillator in the form taken from the van der Pol Oscillator:

$$\ddot{K} + a_V(b_V K^2 - 1)\dot{K} + \omega_V^2 K = 0, \qquad [\mathbf{S2.11}]$$

where K indicates the nondimensional bending curvature. This model has a nonlinear damping term with a coefficient depending on K. Simulated oscillation is a limit cycle of the model (**Fig. S2.12B,F**), given parameters $a_V = -0.026\ s^{-1}$, $b_V = -2.04$, $\omega_V = 5.51\ s^{-1}$.

Second, we developed a model oscillator by modifying the Rayleigh Oscillator:

$$\ddot{K} + a_R\left(b_R\dot{K}^2 - 1\right)\dot{K} + \omega_R^2 K = 0, \qquad [\mathbf{S2.12}]$$

where K again indicates the nondimensional bending curvature. This model has a nonlinear damping term with a coefficient depending on $\dot{K}$. Simulated oscillation is a limit cycle of the model (**Fig. S2.12C,G**), given parameters $a_R = 2.73\ s^{-1}$, $b_R = 0.0023\ s^2$, $\omega_R = 5.6\ s^{-1}$.

Third, we developed a model oscillator by modifying the Stuart-Landau Oscillator:

$$\dot{Z} + \left(\frac{l}{2}|Z|^2 - \sigma\right)Z = 0. \qquad [\mathbf{S2.13}]$$

Here, the system is described in the complex domain where $Z = Z_r + iZ_i$, $l = l_r + il_i$ are complex variables, and σ is real. We let Z_r, the real part of Z, denote the nondimensional curvature K. This model has a nonlinear damping term with coefficient depending on $|Z|$. Simulated oscillation is a limit cycle of the model (**Fig. S2.12D,H**), given parameters $l_r = 0.54\ s^{-1}$, $l_i = 0.52\ s^{-1}$, $\sigma = 5.54\ s^{-1}$.

The three alternative models produce free-running oscillatory dynamics with similar frequency and amplitude as measured from worms swimming in fluids with viscosity $120\ mPa \cdot s$.

SIMULATION OF OPTOGENETIC INHIBITION

According to our experimental observations on the effect of the optogenetic muscle inhibition (**Figs. 2.2A,B**), paralysis of muscles of the illuminated region initiated upon illumination (defined as $t = 0$ for **Fig. 2.2B**) and reached maximal effect approximately at $t = 0.3\ s$. Here, we modeled the process of muscle inhibition by multiplying the scaled active muscle moment, M_a, with a factor, $1 - Q(\Delta t)$, as a function of the time interval Δt in a bell-shaped form (**Fig. S2.10**, **Eqn. S2.14**).

As described in our model, the dorsoventrally alternating feature of the active muscle moment

during locomotion are described by the dynamics of $M_a(t)$. Specifically, $M_a(t)$ is positive when ventral muscles contract and dorsal muscles relax, and negative for the other half of the cycle. Therefore, in our threshold-switch model, specifically inhibiting dorsal- or ventral- or both-side muscles was computationally equivalent to conditionally modulating $M_a(t)$ with the bell-shaped modulating function depending on the sign of $M_a(t)$.

For simulating inhibition process in the three alternative models, we factored out a specific term from individual model equations as a generalized active muscle moment. We applied the bell-shaped modulating function to this term conditionally for each individual model. Detailed descriptions of implementing modeled inhibitions in alternative models are available from below.

To get a deeper understanding of how phase response curves are related to systems dynamics during wave generation, we systematically simulated transient muscle inhibitions on individual model oscillators at different times within a cycle period to generate model PRCs. To do that, we theoretically simulated the process of muscle inhibition by multiplying model active muscle moment with a modulatory factor, $1 - Q(\Delta t)$, which has a bell-shaped profile (**Fig. S2.10**):

$$Q(\Delta t) = \frac{H}{\left(1 + \left|\frac{\Delta t - r}{p}\right|^{2q}\right)}, \qquad [\mathbf{S2.14}]$$

where $r = 0.3\ s$ is the timing of the occurrence of maximal paralysis according to our experimental observations on the effect of muscle inhibition (**Figs. 2.2A,B**), H indicated the maximal degree of paralysis, and p, q measure the paralyzing rate and duration, respectively. To ensure sufficient smoothness during computation, we let $p = 0.3 \cdot 10^{-1/q}$ so that $Q|_{\Delta t=0} > 0.99$. Note that when modeling the dorsal-side-only muscle inhibition, the parameter H for describing max degree of optogenetic muscle inhibition was modulated to $H = 0.5 * H_{optimal}$ to qualitatively agree with experimental observations (**Fig. 2.5**). This factor accounts for unequal degrees of

paralysis during ventral vs. dorsal illumination (**Fig. S2.11**), causing the PRC of dorsal-side illumination to show a relatively moderate response compared to ventral-side illumination.

To simulate the muscle inhibition on our threshold-switch model, we multiplied M_a with $(1-Q)$ any time the model was to be inhibited during its oscillatory period. To apply this operation to the alternative models, we factored out a term as a generalized active muscle moment for each individual model and then multiplied it with the bell-shaped function described above. The generalized forms of active muscle moment for the alternative models are implemented by modifying their original forms as follows:

a. For the van der Pol Oscillator, it is modified as:

$$\begin{cases} \ddot{K} + \left(-\widetilde{M_V} + P_V\right)\dot{K} + \omega_V^2 K = 0 \\ M_V = a_V(1 - b_V K^2) + P_V \end{cases}; \qquad [\mathbf{S2.15}]$$

b. For the Rayleigh Oscillator, it is modified as:

$$\begin{cases} \ddot{K} + \left(-\widetilde{M_R} + P_R\right)\dot{K} + \omega_R^2 K = 0 \\ M_R = a_R\left(1 - b_R \dot{K}^2\right) + P_R \end{cases}; \qquad [\mathbf{S2.16}]$$

c. For the Stuart-Landau Oscillator, it is modified as:

$$\begin{cases} \dot{Z} + \left(-\widetilde{M_S} + P_S\right)Z = 0 \\ M_S = \sigma - \dfrac{l}{2}|Z|^2 + P_S \end{cases}. \qquad [\mathbf{S2.17}]$$

For each individual model listed above, $\widetilde{M_\iota}$ (subscript i represents V, R, and S, respectively) is the generalized muscle moment which is to be multiplied by the bell-shaped factor $(1-Q)$ upon perturbation, and P_i is the additional damping coefficient. Note that the minus sign prior to M_i in the first equation of each set indicates that M_i is a negative damping term that provides power to the system, while P_i is set positive for modeling the effect of bending toward the straight posture due to internal and external viscosity. Also note that **Eqns. S2.15-2.17** would be equivalent to their original form (**Eqns. S2.11-2.13**) when inhibition is absent (in this case, $\widetilde{M_\iota} =$

M_i).

By modeling the muscle inhibition process during locomotion, we were able to perform simulations of phase response experiments on individual models to produce perturbed systems dynamics (**Fig. S2.12J-L**) and the corresponding PRCs (**Fig. S2.12N-P** and **Fig. S2.13**).

OPTIMIZATION OF MODELS

For each individual model we developed, the parameters were determined via a two-round fitting process. First, a subset of parameters was determined by fitting the model to observations of free-moving dynamics; the model could generate free-moving dynamics close to observations at this point. Second, the rest of the parameters were settled by fitting it to experimental phase response curves; a model would be fully determined at this point. Detailed descriptions of the two-step optimization procedure for individual models are provided as follows:

For the original threshold-switch model, parameters τ_u, M_0, and τ_m were explicitly estimated from the experiments of free locomotion using phase portrait techniques described above. To simulate free locomotion, we further determined the position of switch points in the model (as indicated in **Fig. 2.3C** red circle), which we did using method described by **Eqn. S2.7**. Next, we plugged the determined parameters into the model and conducted the second round of optimization by fitting the model with undetermined parameters P_{th}, b, as well as the parameters for simulating muscle inhibition—H and q. We generated model PRC by perturbing the model oscillator at different times within a cycle period and settled the parameters such that the model PRC matched the experimental one with a minimum mean squared error (MSE) (During the computation of MSE, values of both model and experimental PRCs were sampled across the entire range of ϕ with 100 evenly distributed samples. In this case, $\Delta\phi = 2\pi/100$):

$$(P_{th}, b, H, q) = \underset{P_{th},b,H,q}{\operatorname{argmin}} \sum_0^{2\pi} \left(PRC_{model}(P_{th}, b, H, q; \phi) - PRC_{experiment}(\phi)\right)^2 \Delta\phi \qquad [\mathbf{S2.18}]$$

To find the parameters that minimize the difference, a global minimum search was performed using the MATLAB function 'GlobalSearch' (Ugray et al., 2007). When run, the function repeatedly uses a local minimum solver with different batches of parameter range and attempts to locate a solution that produces the lowest MSE value.

Similarly, the two-step optimization procedures for individual alternative models are summarized in Table S2.1.

Table S2.1. Objective functions used in the optimization procedures for alternative models

Type	Free locomotion model	Complete model
van der Pol	$\underset{a_V,b_V,\omega_V}{\operatorname{argmin}}\left(\left(\frac{T_{vdP}}{T_{expt}}-1\right)^2+\left(\frac{A_{vdP}}{A_{exp}}-1\right)^2\right)$	$\underset{p_V,H,q}{\operatorname{argmin}}\sum_0^{2\pi}\left(PRC_{vdP}(\phi)-PRC_{exp}(\phi)\right)^2\Delta\phi$
Rayleigh	$\underset{a_R,b_R,\omega_R}{\operatorname{argmin}}\left(\left(\frac{T_{Rayleigh}}{T_{expt}}-1\right)^2+\left(\frac{A_{Rayleigh}}{A_{exp}}-1\right)^2\right)$	$\underset{p_R,H,q}{\operatorname{argmin}}\sum_0^{2\pi}\left(PRC_{Rayleigh}(\phi)-PRC_{exp}(\phi)\right)^2\Delta\phi$
Stuart-Landau	$\underset{a_S,b_S,\omega_S}{\operatorname{argmin}}\left(\left(\frac{T_{SL}}{T_{expt}}-1\right)^2+\left(\frac{A_{SL}}{A_{exp}}-1\right)^2\right)$	$\underset{p_S,H,q}{\operatorname{argmin}}\sum_0^{2\pi}\left(PRC_{SL}(\phi)-PRC_{exp}(\phi)\right)^2\Delta\phi$

Two-step optimization procedure for van der Pol, Rayleigh, and Stuart-Landau oscillators. The first-step optimization determines part of parameters such that individual models generate free locomotion dynamics. The second-step optimization leads to complete models such that models' perturbed dynamics and phase response curves are produced.

ACKNOWLEDGEMENTS

We thank Mei Zhen and Quan Wen for providing strains. Some strains were provided by the

CGC, funded by NIH Office of Research Infrastructure Programs (P40 OD010440). We thank Gal Haspel, Michael Carchidi, and Patrick McClanahan for helpful discussions. H.J., A.D.F, and C.F.-Y. were supported by the National Institutes of Health (1R01NS084835). S.T. was supported by an Abraham Noordergraaf Research Fellowship and a Littlejohn Fellowship.

CHAPTER 3: A PROPRIOCEPTIVE FEEDBACK CIRCUIT CONTROLS LOCOMOTOR AMPLITUDE THROUGH DOPAMINE AND NEUROPEPTIDE SIGNALING IN *C. ELEGANS*

Hongfei Ji[1], Anthony D. Fouad[1], Zihao Li[1], Andrew Ruba[1], and Christopher Fang-Yen[1,2]

[1]Department of Bioengineering, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA 19104

[2]Department of Neuroscience, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104

This chapter is a slightly modified manuscript in preparation.

Author contributions: Hongfei Ji: Data curation, Software, Formal analysis, Investigation, Visualization, Methodology, Writing - original draft, review, and editing; Anthony D Fouad: Software, Methodology; Zihao Li, Andrew Ruba: Investigation; Christopher Fang-Yen: Conceptualization, Resources, Software, Supervision, Funding acquisition, Methodology, Writing - original draft, review, and editing, Project administration.

INTRODUCTION

Navigation in natural environments requires flexible and adaptable locomotor behavior (Alexander, 2013; Dickinson et al., 2000). During locomotion in a natural context, an animal often encounters obstacles and irregularities, and needs to modulate the activity in order to adapt its body posture and motor patterns to the inevitable perturbations (Grillner and El Manira, 2020; Pearson, 2000). In lamprey and other undulatory animals, while generating a spatiotemporal pattern for propulsion, the motor circuits also precisely and contextually tune the body's biomechanical responses to the external conditions (Berri et al., 2009; Blight, 1977; Fang-Yen et al., 2010; Ji et al., 2021a; LONG, 1998; Tytell et al., 2018). Kinematic and electromyographic

studies in legged animals characterized a phasic compensatory reaction in a perturbed animal, which induces rapid corrective movements adapted to the perturbation (Dietz et al., 1987; Forssberg, 1979; Forssberg et al., 1975, 1977; Mayer and Akay, 2018; McVea and Pearson, 2007; Potocanac et al., 2016; Prochazka et al., 1978).

Well-adapted locomotion against unexpected perturbations relies on intricate interactions between a dedicated neural circuitry capable of generating basic locomotor rhythm (central pattern generator; Cohen and Wallén, 1980; Delcomyn, 1980; Grillner, 2003; Kiehn, 2011; Kristan and Calabrese, 1976; Marder and Calabrese, 1996; Pearce and Friesen, 1984; Yu et al., 1999) and various feedback pathways that modulate locomotion (Grillner, 2006; Grillner and El Manira, 2020; Kiehn, 2016; Rossignol, 2006; Windhorst, 2007). In particular, proprioception plays an essential role in providing rapid feedback on body position for locomotor control during natural movements (Andersson et al., 1981; Brodfuehrer and Friesen, 1986; Friesen, 2009; Grillner and Wallen, 2002; Pearson, 2000; Wen et al., 2012). In mammals, proprioceptive inputs from multiple sensory organs are continuously weighted and processed within spinal cord circuits to instruct compensatory electromyographic responses to the current locomotor situation (Dietz, 2002). In lamprey and other undulators alike, adaptive locomotion relies critically on proprioceptive feedback to adjust or correct its undulatory gait to the changing physical space (Berri et al., 2009; Boyle et al., 2012; Fang-Yen et al., 2010; Fouad et al., 2017; Iwasaki et al., 2014; Ji et al., 2021a; Picton et al., 2021; Susoy et al., 2021).

The neural mechanisms underlying adaptive locomotion are complex. In vertebrates, the corrective locomotor control for handling perturbations during movements involves delicate computations within circuits of spinal cord, brain stem, and forebrain (Grillner, 2003; Grillner et al., 2005; Roseberry et al., 2016; Svoboda and Li, 2018). Recent genetic expression and manipulation techniques have greatly facilitated the analyses of the function of CPGs in fine

locomotor control as well as the identification of relevant interneurons (Goulding, 2009; Grillner and Jessell, 2009; Kiehn, 2016). In the mouse, many spinal cord interneurons with their membrane properties and synaptic connectivity have been identified and found to be important for corrective locomotor control (Alvarez et al., 2005; Bourane et al., 2015a, 2015b; Bui et al., 2013; Hilde et al., 2016; Koch et al., 2017; Zagoraiou et al., 2009; Zhang et al., 2008). However, it is unclear how the identified interneurons are implicated in the underlying neuronal pathways to contribute to the motor control, in part because of the lack of *in vivo* methods for acutely interfering neuronal activity to assess their role in locomotor movements (Grillner and El Manira, 2020; Mayer and Akay, 2018). Moreover, our knowledge of how proprioception modality is relayed to control movement and posture, and which interneurons are responsible for gating proprioceptive signal transmission is still limited (Büschges and Mantziaris, 2021; Dietz, 2002; Pearson, 2004; Zhen and Samuel, 2015). Here, we use the locomotor behavior of *C. elegans* to ask how the corrective locomotor control is embedded in nervous system.

C. elegans has a relatively small and well-identified nervous system (Bargmann, 2012; Cook et al., 2019; White et al., 1986). Advances in physiology and neurogenetics establish a variety of genetic and physical methods (Bargmann, 1998; Brenner, 1974; Chronis et al., 2007; Dong et al., 2021; Hobert, 2003; Leifer et al., 2011; Lockery et al., 2008), offering an opportunity for a systems level dissection of locomotor control.

Undulatory movements of *C. elegans* during locomotion are generated by a set of body wall muscles arranged in two dorsal and two ventral rows running along the length of worm's body (Von Stetina et al., 2006), with adjacent muscle cells within each row coupled by gap junctions (Liu et al., 2006). Alternating activity of antagonist muscles are driven by motor neurons located in head ganglia and the ventral nerve cord (Zhen and Samuel, 2015), of which the proprioception coupling between the ventral cord motor neurons is required for propagating rhythmic activity

along the body (Wen et al., 2012). Motion direction is controlled by a set of premotor interneurons (previously named as “command interneurons”) that directly instruct motor neurons to coordinate forward and backward locomotion (Chalfie et al., 1988; Kawano et al., 2011).

While basic sinusoidal locomotion can arise from a circuitry with only motor neurons and command interneurons (Chalfie et al., 1985; Deng et al., 2020; Gao et al., 2018; Kawano et al., 2011), adaptable locomotion in natural contexts involves motor control of the head and sublateral motor neurons with various functional roles in modulating postures (Gray et al., 2005; Kaplan et al., 2020; Kato et al., 2015; Kratsios et al., 2012; Schwarz and Bringmann, 2017; Yeon et al., 2018). Moreover, context-dependent, optimal motor control is subject to the feedback input from a large number of interneurons and sensory neurons (Cermak et al., 2020; Cook et al., 2019; Hums et al., 2016; Li et al., 2006; López-Cruz et al., 2019; Oranth et al., 2018; Shen et al., 2016; Tao et al., 2019; Xu et al., 2018), as well as neuromodulation of biogenic amines and neuropeptides which reconfigures circuit properties for driving various long-term or short-term locomotor states (Bargmann, 2012; Chase et al., 2004; Churgin et al., 2017; Donnelly et al., 2013; Flavell et al., 2013; Hills et al., 2004; Hu et al., 2011; Sawin et al., 2000; Susoy et al., 2021; Vidal-Gadea et al., 2011).

In the wild, natural *C. elegans* populations can be found from soil, compost, leaf, and various other environments where they inevitably need to adjust their movements against gait perturbations such as terrain obstructions. Previous optical physiological studies reported several cases where *C. elegans* exhibits several spatiotemporally distinctive locomotory dynamics upon different gait perturbations. When physically immobilized in the midbody, *C. elegans* anterior unrestrained region continues undulating while the posterior free end stays at a static curvature that follows the shape of the immobilized region (Wen et al., 2012). When the anterior bending activity is optogenetically inhibited, *C. elegans* head region oscillates at a lower frequency while

the posterior undulation frequency doubles (Fouad et al., 2018; Xu et al., 2018). Upon a transient inhibition on the anterior bending activity, the local bending dynamic exhibits a phase-dependent perturbation response curve (Ji et al., 2021a). Despite the few descriptions under perturbative conditions, corrective locomotor control to gait perturbations in *C. elegans* is still not well characterized, nor are the underlying neuronal control mechanisms.

Here, we sought to understand how the *C. elegans* motor system adapts locomotion in response to gait perturbations on the behavioral, circuit, and molecular levels. The present study demonstrates that *C. elegans* uses a posterior-to-anterior proprioceptive coupling to adapt its locomotor amplitude to gait perturbations in a homeostatic manner. Through microfluidic manipulation of behavioral outputs, *in vivo* optical neurophysiology, and molecular genetics, we dissect the underlying neuronal pathways: dopaminergic PDE neurons, functioning as proprioceptors by sensing the midbody curvature, transduce local proprioceptive input into dopamine signaling. The dopamine activity of PDE then drives AVK interneurons activity via D2-like dopamine receptor DOP-3. FMRFamide-like neuropeptide FLP-1, released by AVK, regulates SMB motor neurons via receptor NPR-6, hence modulating anterior bending amplitude. Our findings identify a behavioral circuit for a type of corrective movement control in *C. elegans* locomotion from sensory input to motor output.

RESULTS

C. ELEGANS MODULATES ANTERIOR AMPLITUDE RETROGRADELY IN RESPONSE TO THE OPTOGENETICALLY PERTURBED MIDBODY CURVATURE

During forward locomotion, *C. elegans* propagates sinusoidal bending waves posteriorly from the head region by alternating contralateral muscle contraction and relaxation along the worm's body (Croll, 1970; Wen et al., 2012). To describe the undulatory behavior, we quantified the kinematics of worm undulation by calculating the time-varying curvature along the body centerline (Fang-Yen

et al., 2010; Leifer et al., 2011). We defined the normalized curvature as the nondimensional product of the body length and the reciprocal of local radius of curvature along the centerline of the body (**Fig. 3.1A**). With this metric, we quantified the undulatory behavior using the time-varying normalized curvature from head to tail as shown in a kymograph (**Fig. 3.1B**).

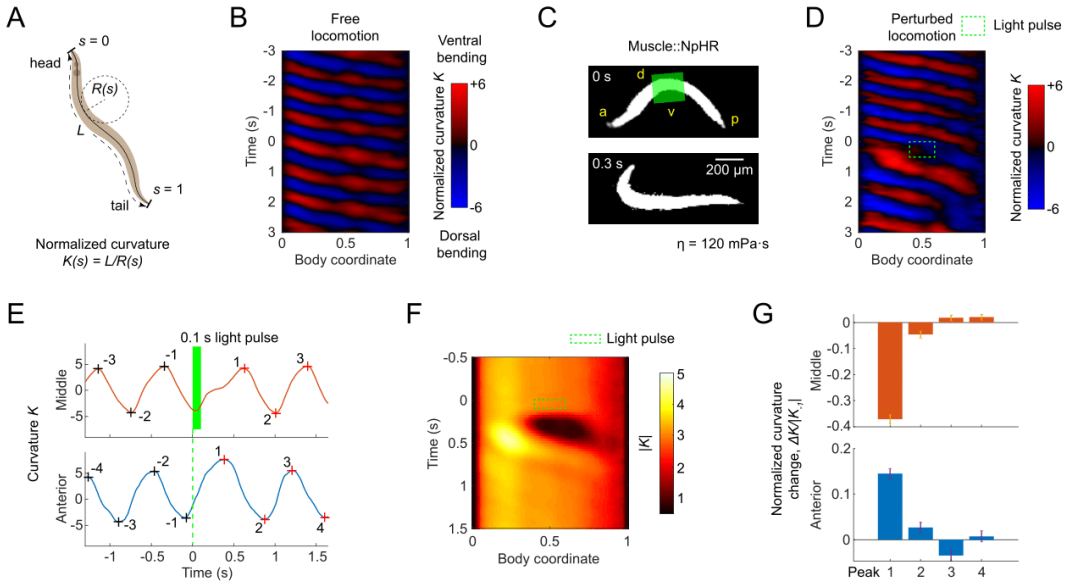

Figure 3.1. Optogenetic inhibition of midbody muscles causes increase in anterior bending amplitude.

(A) Worm locomotory dynamics can be represented by time-varying curvature along the body. Body coordinate s is denoted by the distance along the centerline normalized by the body length L (head = 0, tail = 1). The normalized curvature K is the nondimensional product of the body length and the reciprocal of the local radius of curvature with positive and negative values representing ventral and dorsal bending, respectively.

(B) Curvature kymograph (curvature as a function of time and body coordinate) of a freely moving worm during forward locomotion.

(C) Images of a transgenic worm (Muscle::NpHR) perturbed by an optogenetic muscle inhibition in the midbody during forward locomotion in a viscous liquid (viscosity = 120 mPa·s). Green shaded region indicates the laser illumination. a: anterior, p: posterior, d: dorsal, v: ventral. Scale bar, 200 μm.

(D) Curvature kymograph of the worm locomotion shown in (C). Green dashed box indicates the 0.5 s laser illumination interval starting at $t = 0$ applied to the midbody.

(E) A representative trial of curvature dynamics of a worm's middle (*upper*) and anterior (*lower*) regions around a 0.1 s muscle inhibition in the midbody (green bar, aligned at $t = 0$). Black and red crosses mark the last four pre-illumination and the first four post-illumination curvature peaks, respectively.

(F) Kymograph of mean absolute curvature around the 0.1 s inhibitions (green dashed box) from 1160 trials using 206 worms.

(G) Undulatory amplitude change upon transient midbody muscle inhibitions, measured as mean ± SEM of the normalized curvature change of the first four post-illumination curvature peaks of the middle (*upper*) and anterior (*lower*) body regions, respectively. Same data as used in (F). For each trial of illumination, the normalized curvature change is defined by $\Delta K/|K_{-1}| = (|K_{+n}| - |K_{-1}|)/|K_{-1}|$, where $|K_{+n}|$ denotes the absolute value of the n^{th} post-illumination curvature peak and $|K_{-i}|$ ($i = 1$ or 2) denotes the absolute value of the last pre-illumination curvature peak that has the same bending direction of K_{+n}. Regarding worm body regions, middle = 0.4-0.6, anterior = 0.1-0.3 body coordinate.

To examine how *C. elegans* modulates locomotion in reaction to gait perturbation, we systematically perturbed the motor activity in different body regions and analyzed the corresponding undulatory dynamics during forward locomotion. With an optogenetic laser

targeting system (Fouad et al., 2018), we applied laser illumination (532 nm wavelength) to selected body regions of animals expressing the inhibitory opsin *NpHR* in body wall muscles (via the transgene *Pmyo-3::NpHR*).

Transient optogenetic inhibitions (0.1 or 0.5 s duration) of muscles at the head region (0.05-0.25 body length) and neck region (0.2-0.4 body length) both caused a rapid straightening of the anterior region followed with a mild amplitude decline in the subsequent body bends propagating from head to tail (**Fig. S3.1A, B, E**, and **F**), consistent with previous findings (Fouad et al., 2018; Ji et al., 2021c; Xu et al., 2018).

When we inhibited muscles at midbody (0.4-0.6 body length), however, besides the paralytic effect propagating from midbody to tail, we observed exaggerated undulatory oscillations at the anterior region (0.1-0.3 body length; **Figs. 3.1C-F**; **Fig. S3.1C** and **G**). Quantitatively, midbody amplitude decreased by ~35% immediately after a transient midbody muscle inhibition (0.1 s duration) whereas the anterior amplitude increased by ~15%; within about one undulatory cycle after the inhibition, worms recovered to baseline undulatory amplitude (**Fig. 3.1G**).

Next, we asked how a worm would modulate its undulation in response to midbody amplitude exaggeration instead of reduction. To do this, we stimulated midbody muscles only on the dorsal side by illuminating (473 nm wavelength) the corresponding region of animals expressing the excitatory opsin *ChR2* in body wall muscles (via the transgene *Pmyo-3::ChR2*). We found that while stimulating dorsal muscles in the midbody led to exaggerated midbody bending, the anterior bending decreased (**Figs. 3.2A-D**). Quantitatively, midbody amplitude increased by ~12% immediately after a transient midbody dorsal muscle stimulation (0.1 s duration) whereas the anterior amplitude decreased by ~12% (**Fig. 3.2E**).

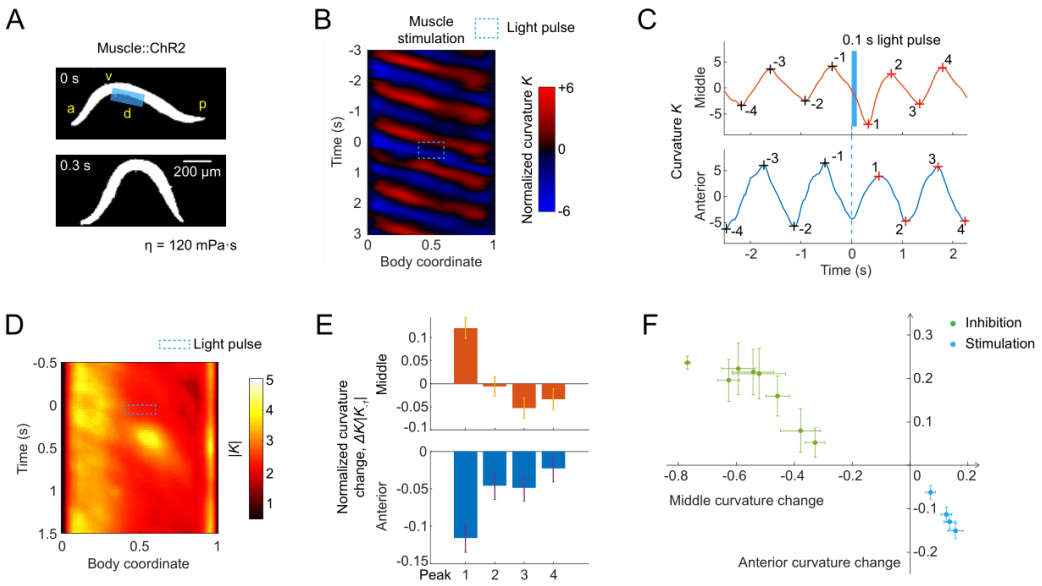

Figure 3.2. *C. elegans* modulates anterior amplitude retrogradely in response to the optogenetically perturbed midbody curvature.

(A) Images of a transgenic worm (Muscle::ChR2) perturbed by an optogenetic muscle stimulation in the dorsal midbody during forward locomotion in a viscous liquid (viscosity = 120 mPa·s). Blue shaded region indicates the laser illumination. a: anterior, p: posterior, d: dorsal, v: ventral. Scale bar, 200 µm.

(B) Curvature kymograph of the worm locomotion shown in (A). Blue dashed box indicates the 0.5 s laser illumination interval starting at $t = 0$ applied to the dorsal midbody.

(C) A representative trial of curvature dynamics of a worm's middle (*upper*) and anterior (*lower*) regions around a 0.1 s muscle stimulation in the dorsal midbody (blue bar, aligned at $t = 0$). Black and red crosses mark the last four pre-illumination and the first four post-illumination curvature peaks, respectively.

(D) Kymograph of mean absolute curvature around the 0.1 s stimulations (blue dashed box) from

693 trials using 122 worms.

(E) Undulatory amplitude change upon transient dorsal midbody muscle stimulations, measured as mean ± SEM of the normalized curvature change of the first four post-illumination curvature peaks (same definition as in **Fig. 3.1G**) of the middle (*upper*) and anterior (*lower*) regions, respectively. Same data as used in (D).

(F) A scatter plot of the mean anterior curvature change plotted against the mean midbody curvature change. Each data point represents mean ± SEM of the corresponding normalized value of the first post-illumination curvature peak. Green and blue data points denote data induced by optogenetic midbody muscle inhibition (both sides) and stimulation (dorsal side), respectively. We varied the laser power and/or pulse duration to induce different degrees of optogenetic perturbation for individual groups. Each group consists of ~120 trials totaling ~15 animals. Regarding worm regions, middle = 0.4-0.6, anterior = 0.1-0.3 body coordinate.

We also tested worm locomotion perturbed by brief muscle inhibition (0.1 s duration) at the posterior region (0.6-0.8 body length). In this case, posterior bending amplitude rapidly reduced upon illumination, but bending amplitude of the anterior half did not increase (**Fig. S3.1D** and **H**).

From these optogenetic experiments, our findings suggest that anterior amplitude might change retrogradely in response to the midbody amplitude change. To get more insight into this locomotor adaptation during midbody perturbation, we conducted dose-response experiments in which the degree of midbody muscle inhibition or stimulation was modulated by applying laser illumination with various pulse durations and irradiances (see *Methods*). We found that treated animals modulated the anterior amplitude in response to the induced midbody amplitude change with a negatively correlated relationship (**Fig. 3.2F**). Our data thus indicate an anteriorward compensatory coupling mechanism underlying locomotor adaption to midbody perturbation. We refer to this anterior amplitude modulation of moving animals under midbody amplitude

perturbation as the “curvature compensatory response”.

MICROFLUIDIC CONSTRAINT OF MIDBODY CAUSES INCREASE IN ANTERIOR BENDING AMPLITUDE

During our optogenetic experiments, curvature change was induced by a direct manipulation of muscle activity, leaving open the possibility of a proprioception-free muscular coupling that solely supports the curvature compensation.

To determine whether proprioception is involved in the curvature compensation, we designed a microfluidic device that constrained the middle region of a worm in a straight channel (**Figs. 3.3A** and **3.3B**). We used a 200-µm-long, 60-µm-wide channel to constrain the bending amplitude in the midbody. By comparing between constrained and free locomotion, we found that tested animals exhibited exaggerated oscillations in the anterior region during midbody constraint (**Figs. 3.3B-E**).

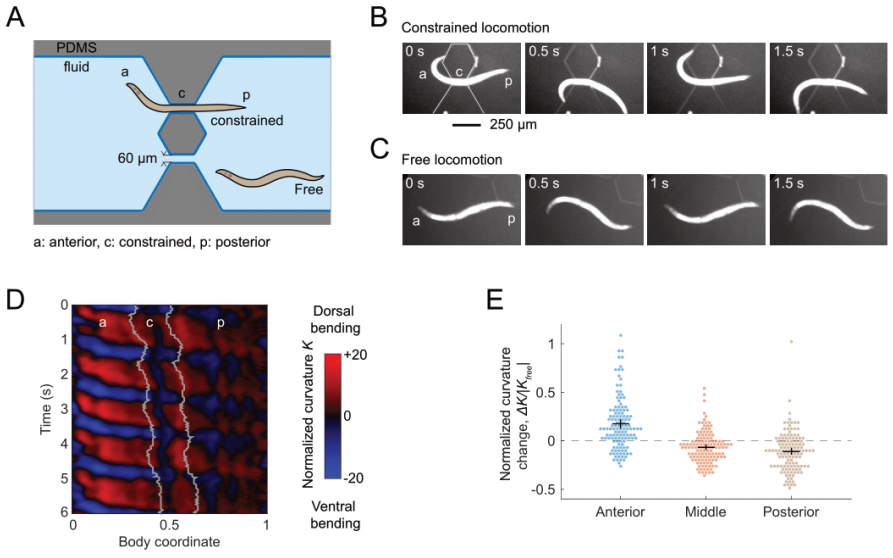

Figure 3.3. Microfluidic constraint of midbody causes increase in anterior bending amplitude.

(A) Schematic of the microfluidic device for constraining body curvature. By manipulating its relative position within the chamber through a fine flow control (see *Methods*), we were able to make a worm alternate between free locomotion and constrained locomotion. a: anterior region, p: posterior region, c: constrained middle region of a worm. PDMS: Poly-dimethyl siloxane.

(B and C) Video images of a wild-type worm doing constrained locomotion with its midbody confined by the narrow channel (B) and doing free locomotion in the open area of the chamber (C). Scale bar, 250 μm.

(D) Curvature kymograph of the constrained locomotion shown in (B). Gray lines indicate the anterior and posterior limits of the narrow channel with the worm body as a frame of reference.

(E) Effects of midbody constraint during forward locomotion on the undulatory bending amplitude of the anterior, middle, and posterior regions, measured as mean ± SEM of the normalized curvature change of corresponding regions. Each data point is the mean of a 3 s period of constrained locomotion pooled across 19 wild-type animals.

Adjacent body wall muscle cells are connected through electrical coupling regulated by an innexin UNC-9 (Liu et al., 2006). We thus further asked whether the body muscle cells themselves might transduce the proprioceptive signals from midbody to anterior regions. To verify this, we used the microfluidic channel to test transgenic animals that lacked these gap junctions in the muscle cells (via a *unc-9* pan-neuronal rescued strain with gap junction deficiency only in muscle) (Wen et al., 2012). We found that these transgenic animals again showed exaggerated bending movements in the anterior region during midbody constraint (**Fig. S3.2B**).

Taken together, these results indicate that curvature compensation is mediated by an anteriorward proprioceptive coupling mechanism while intermuscular coupling is insufficient to transduce the proprioceptive signals.

CURVATURE COMPENSATION REQUIRES FUNCTIONAL DOPAMINE SIGNALING BY PDE NEURONS

We next sought to explore the mechanisms underlying the curvature compensation. First, we asked whether neurotransmission is essential for this behavioral response. To do that, we examined several mutant strains bearing defects in biogenic amine synthesis including dopamine (DA), serotonin (5-HT), tyramine (TA), and octopamine (OA). For each mutant, we analyzed the anterior bending amplitude of animals being constraint by the microfluidic channel in the midbody and compared with the amplitude during free locomotion. An indicator of curvature compensation was defined across all tested animals as the difference between anterior bending amplitudes during constrained and free locomotion normalized by the amplitude during free locomotion (see *Methods*).

Mutants *tph-1(n4622)* (defective in serotonin synthesis) and *tdc-1(n3421)* (defective in both tyramine and octopamine syntheses) displayed nearly normal locomotion and largely intact curvature compensation during midbody constraint from microfluidic channel (**Fig. S3.2B**), suggesting that serotonin, tyramine, and octopamine are unrequired for curvature compensation. In contrast, dopamine-deficient *cat-2(e1112)* mutants displayed normal locomotion but impaired curvature compensation, as they showed reduced anterior curvature change in response to the microfluidic constraint in the midbody (**Fig. 3.4C**). Addition of exogenous dopamine fully restored the curvature compensatory response (**Fig. 3.4C**), demonstrating that the defect in curvature compensation induced by microfluidic constraint is due to the lack of dopamine.

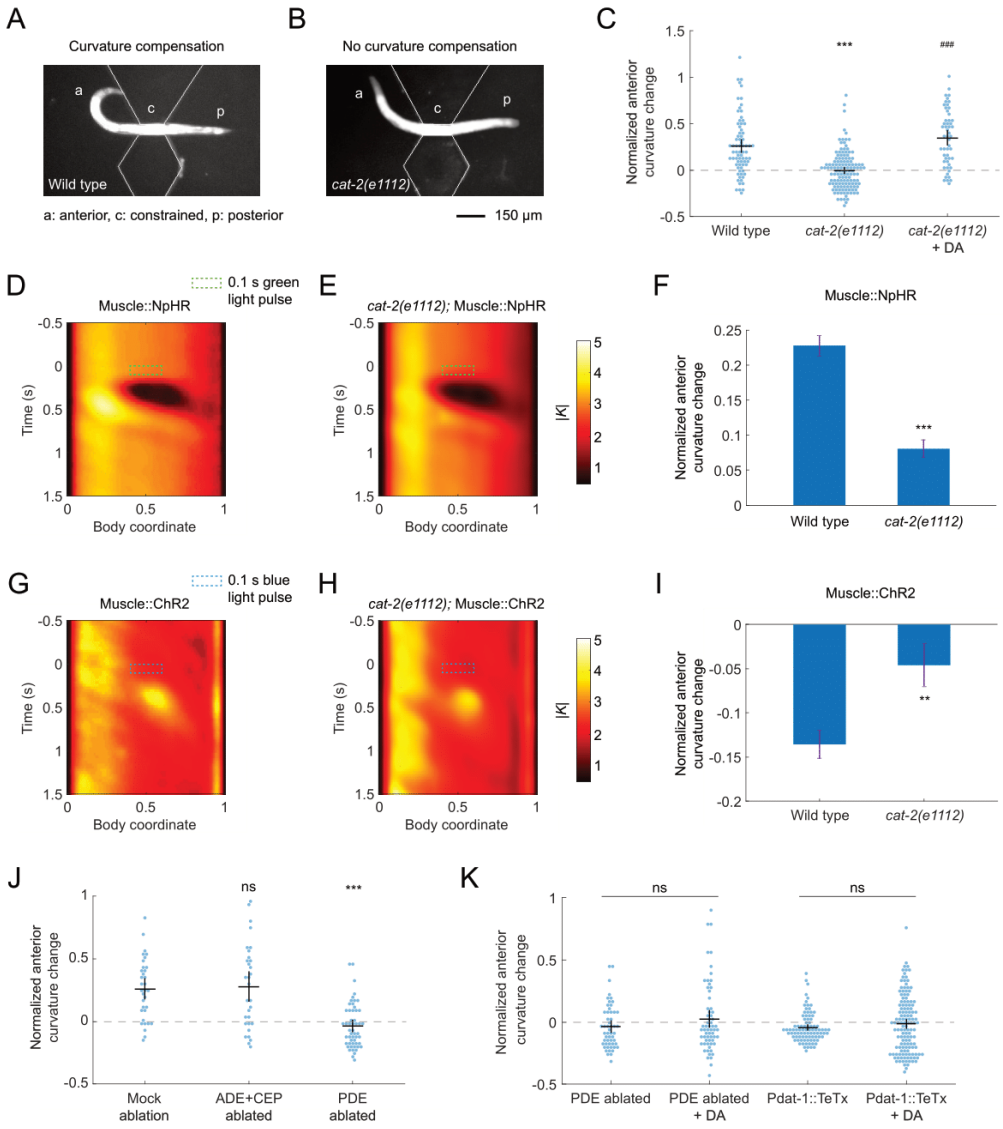

Figure 3.4. Curvature compensation requires functional dopamine signaling by PDE neurons.

(A and B) Video images of a wild-type animal (A) and a *cat-2(e1112)* mutant (B) with midbody confined by the narrow channel, exhibiting curvature compensation and no curvature compensation, respectively. a: anterior region, p: posterior region, c: constrained middle region of a worm. Scale bar, 150 µm.

(C) *cat-2* mutants showed impaired curvature compensatory response to midbody constraint, which was rescued by exogenous dopamine. Data represent mean ± SEM of the normalized anterior curvature change in response to midbody constraint for wild type and *cat-2(e1112)* mutants in either the absence or presence of exogenous 50 mM dopamine. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each indicated condition. ***p<0.001 when compared with wild type, $^{###}$p<0.001 when compared with *cat-2* mutants without exogenous dopamine, Tukey-Kramer multiple comparison tests.

(D-F) *cat-2* mutants showed impaired curvature compensatory response to midbody curvature decrease induced by transient optogenetic muscle inhibition. (D and E) Kymographs of mean absolute curvature around 0.1 s illuminations (green dashed box) for animals expressing Muscle::NpHR in wild type (D, same data as used in **Fig. 3.1F**) and *cat*-2 mutants (E, n = 133 trials using 33 worms). (F) Normalized anterior curvature change of the first post-illumination curvature peak for animals expressing Muscle::NpHR in wild type and *cat-2* mutants (same data as used in D and E, respectively), mean ± SEM. ***p<0.001, Student's *t* test.

(G-I) *cat-2* mutants showed impaired curvature compensatory response to midbody curvature increase induced by transient optogenetic dorsal muscle stimulation. (G and H) Kymographs of mean absolute curvature around 0.1 s illuminations (blue dashed box) for animals expressing Muscle::ChR2 in wild type (G, same data as used in **Fig. 3.2D**) and *cat-2* mutants (H, n = 112 trials using 24 worms). (I) Normalized anterior curvature change of the first post-illumination curvature peak for animals expressing Muscle::ChR2 in wild type and *cat-2* mutants (same data as used in G and H, respectively), mean ± SEM. ***p<0.001, Student's *t* test.

(J) Out of all dopaminergic neurons, ablating PDE eliminated curvature compensation. Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for animals with genetic ablation of ADE and CEP (*Pdat-1::ICE*, PDE survival

confirmed by using co-expression of *Pdat-1::RFP*) and laser ablation of PDE, compared with mock-ablated control group. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each condition. ***p<0.001, ns: not significant, Dunnett's multiple comparison tests.

(K) Curvature compensation requires dopamine signaling specifically by PDE neurons. Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for PDE-ablated worms and transgenic animals expressing tetanus toxin light chain in all dopaminergic neurons (*Pdat-1::TeTx*), in the absence and presence of exogenous 50 mM dopamine, respectively. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each indicated condition. ns: not significant, Student's *t* test.

In our previous experiments with wild-type animals, a worm's midbody curvature change was induced by either optogenetic manipulation or microfluidic constraint. Since the two methods manipulated curvature physiologically differently, it is unclear whether *cat-2* mutants are also defective in curvature compensation in response to optogenetic perturbation in the midbody. To address this issue, we integrated transgenic expression Muscle::NpHR and Muscle::ChR2 respectively into *cat-2* mutants and performed optogenetic muscle inhibition and stimulation experiments with these strains by following the same procedures as described in the earlier section. As opposed to wild-type animals, we found *cat-2* mutants again showed impaired curvature compensatory response to midbody curvature decrease or exaggeration triggered by optogenetic muscle inhibition (**Figs. 3.4D-F**) or stimulation (**Figs. 3.4G-I**).

The above data suggest that dopamine signaling is required for the curvature compensation in response to both midbody curvature decrease and increase. In *C. elegans*, dopamine plays an essential role in a variety of behaviors including locomotion, food sensation, touch sensation, egg

laying, spatial pattern selectivity, gait transition (Calhoun et al., 2015; Chase et al., 2004; Han et al., 2017; Hills et al., 2004; Kindt et al., 2007; Sawin et al., 2000; Vidal-Gadea et al., 2011). The *C. elegans* hermaphrodite has eight dopaminergic neurons including four CEPs, two ADEs, and two PDEs (Sulston et al., 1975). To test which of these dopaminergic neurons were required for the curvature compensatory response, we ablated specific subsets of dopaminergic neurons of young larvae at their L3 stage, and examined the resulting adults' compensatory response to microfluidic constraint in the midbody.

First, we ablated the ADEs and CEPs using transgenic animals expressing the human caspase interleukin-1β-converting enzyme (ICE) in the dopaminergic neurons under the *dat-1* promoter (Hills et al., 2004). By integrating the *Pdat-1::ICE* strain with a transgene *Pdat-1::mCherry* that expresses RFP in all dopaminergic neurons, we verified the cell death of ADEs and CEPs as well as the survival of PDEs through the resulting RFP expression (see details in *Methods*). Second, we ablated only PDE neurons using a thermal laser beam (Fouad et al., 2021; also see *Methods*). Compared with the mock-ablated group, transgenic worms in which ADEs and CEPs were killed still exhibited curvature compensatory response to the microfluidic constraint in the midbody, while worms lacking only PDEs did not exhibit curvature compensation (**Fig. 3.4J**) and failed to be restored by the addition of exogenous dopamine (**Fig. 3.4K**). These results suggest that, out of all dopaminergic neurons, only PDE neurons are necessary for curvature compensation.

To further explore the dopamine signaling of PDE for curvature compensation, we examined transgenic animals with dopaminergic neurons expressing tetanus toxin light chain (*Pdat-1::TeTx*) that blocks their synaptic transmission. Inhibiting neurotransmitter release from dopaminergic neurons eliminates curvature compensation (**Fig. 3.4K**) and adding exogenous dopamine failed to restore this phenotype (**Fig. 3.4K**). Note that, in exogenous dopamine environments, only the

dopamine synthesis-deficient mutant *cat-2(e1112)* got rescued for curvature compensation whereas animals with PDE eliminated or with TeTx-expressing dopaminergic neurons still exhibited impaired curvature compensation. We argue the discrepancy in the rescue results may be because *cat-2* mutants still have the vesicles for the release of dopamine, but the other groups do not. Thus, the results above further suggest the necessity of functional vesicle release of dopamine from PDE for curvature compensation.

Taken together, these experiments indicate that synaptic release of dopamine from PDEs, but not from ADEs or CEPs, is required for curvature compensation.

CALCIUM IMAGING SHOWS THAT PDE NEURONS RESPONSE TO MIDBODY CURVATURE

Ultrastructurally, PDE is the only dopaminergic neuron whose neuronal processes extend across the midbody, the region where the curvature perturbation was applied. Thus, the interpretation of our experiments further implies that PDEs might function as a proprioceptor that transduces the midbody proprioceptive input for curvature compensation. Previous studies have demonstrated that PDE Ca^{2+} activity in a wild-type animal is phase-locked to its bending waves during roaming (Cermak et al., 2020), but whether PDEs are proprioceptive to the body bends has not been characterized.

As a first step in investigating the neuronal activity of PDE in response to bending curvatures, we monitored the spontaneous Ca^{2+} transients of PDE in freely crawling animals expressing genetically encoded Ca^{2+} indicators GCaMP in the PDE neurons (under the *Pdat-1* promoter). We prepared the transgenic animals on an agarose pad covered with a microscope slide and mounted the pad onto our fluorescence microscope (see *Methods* for preparation details). As an animal performed free locomotion on the setup, we observed robust oscillating Ca^{2+} dynamics in the PDE soma (**Fig. 3.5B**) during its forward movement (**Fig. 3.5A**). We also noticed that the Ca^{2+} activity in PDE soma was correlated with the animal's body curvature (**Fig. 3.5B** shows an

example; yellow curve in **Fig. 3.5E** shows correlations between PDE activity and curvatures of different body regions). These correlations between PDE fluorescence and body posture were not observed in the control group of transgenic animals expressing GFP in PDE (**Fig. S3.4**). Our Ca^{2+} imaging experiments indicate that the native neuronal activity of PDE correlates with body posture during free locomotion of an intact wild-type animal, as has been previously reported (Cermak et al., 2020).

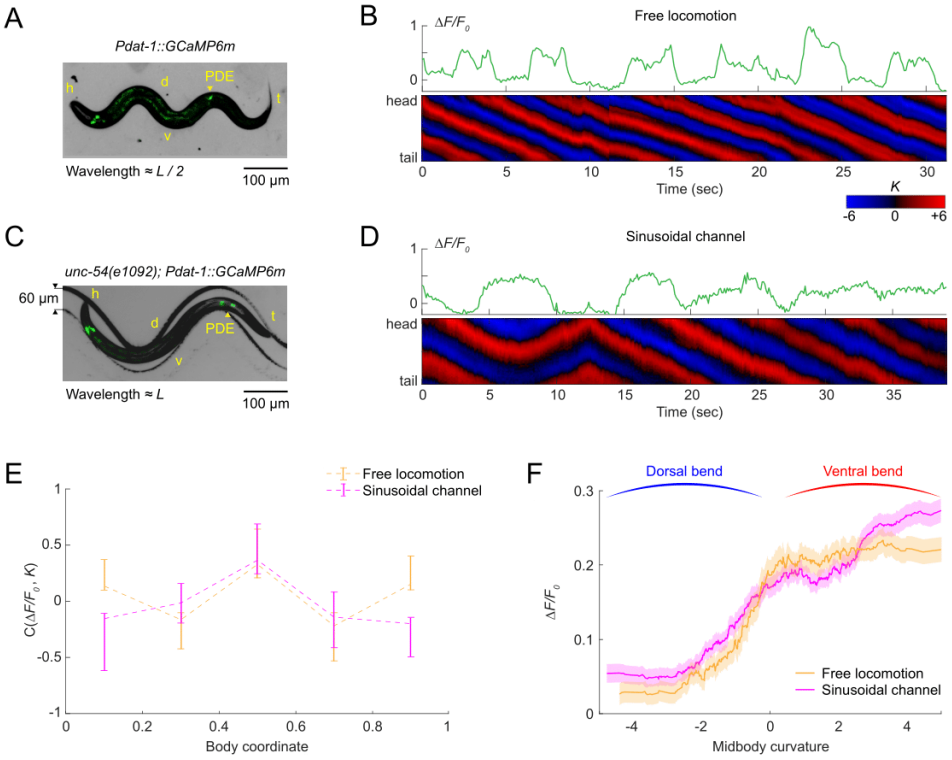

Figure 3.5. Ca^{2+} imaging shows that PDE neurons respond to midbody curvature.

(A and C) Fluorescent video images of PDE neuron (via transgenic expression *Pdat-1::GCaMP6m*) in a freely moving wild-type animal on an agar surface (A) and a muscularly paralyzed mutant *unc-54(e1092)* restrained within a 60-µm-wide sinusoidal channel (C). Undulatory wavelength of the unrestrained and channel-restrained locomotion is roughly $L/2$ and

L, respectively. h: head, t: tail, d: dorsal, v: ventral. L, worm body length. Scale bar, 100 μm.

(B and D) Intracellular Ca^{2+} dynamics of PDE (*upper*) and the corresponding curvature dynamics from head to tail (*lower*) for freely moving worms (B) and muscularly paralyzed worms restrained within sinusoidal channels (D), respectively. The intracellular Ca^{2+} activity is inferred from $\Delta F/F_0$, the relative deviation of *GCaMP6m* fluorescence intensity from the baseline.

(E) Cross-correlation between intracellular Ca^{2+} dynamics of PDE and curvatures of different body regions from head to tail, for freely moving worms (yellow) and channel-restrained worms (purple).

(F) Average PDE Ca^{2+} activity at different values of midbody curvature, for freely moving worms (yellow) and channel-restrained worms (purple). Curves are obtained via a moving average along the *x*-axis with 2 in bin width.

For (E) and (F), n = 12 and 20 animals for free locomotion group and sinusoidal channel group, respectively. Data are shown as means ± 95% confidence interval.

To provide physiological evidence that body posture-correlated PDE Ca^{2+} activity was attributable to a proprioceptive response to body bending, we monitored the PDE Ca^{2+} dynamics in the *unc-54(e1092)* mutants, which were defective in muscle contraction due to a lack of a major myosin heavy chain protein. To manually bend the worm body, we first restrained the posture of worms within a microfluidic sinusoidal channel (**Fig. 3.5C**) filled with viscous solutions (120 mPa·s in viscosity). We then manipulated the worm position within the sinusoidal channel to force the body segments at different curvature values by controlling the direction and rate of the fluidic flow with a syringe pump connected to the microfluidic device. Under this experimental setup, we again observed fluctuating PDE Ca^{2+} dynamics in response to the varying induced body posture as we moved the paralyzed worm through the channel (**Fig. 3.5D**). Despite the

mutant animals' incapability of moving due to muscle paralysis, we still observed significant correlations between PDE fluorescence and bending curvature of various body segments (**Fig. 3.5E**, purple curve). These data suggest that body bending is sufficient to induce the neuronal activity in PDE.

We further reasoned that the proprioceptive response in PDE Ca^{2+} dynamics was caused by the midbody curvature. First, by analyzing worm postures, we quantified the body wavelengths of freely moving worms (as shown by **Fig. 3.5A**) and paralyzed worms restrained within sinusoidal channels (as shown by **Fig. 3.5C**) to be approximately $L/2$ and L, respectively. Second, we noticed that similar periodicities were exhibited from the profiles of the curvature-PDE Ca^{2+} activity correlations under the two corresponding experimental conditions (by comparing yellow and purple curves in **Fig. 3.5E** with body postures in **Figs. 3.5A** and **3.5B** respectively). This was because curvature dynamics at any two body regions one half of wavelength apart are very highly anticorrelated and the curvature-neuronal activity correlations can be transitive. Third, the two independent curvature-neuronal activity correlation profiles coincided only at midbody region (**Fig. 3.5E**). This observation indicated that midbody might be the spatial receptive field of the proprioceptive response in PDE neurons. Furthermore, by quantifying the dependence of PDE activity on midbody curvature, we found that PDE Ca^{2+} levels increased as the midbody curvature varied from a dorsal bend to a ventral bend (**Fig. 3.5F**), whether the movement was due to muscle contractions of freely moving worms or external forces from sinusoidal channels.

Together with previous work (Cermak et al., 2020), our experiments suggest a proprioceptive functionality in PDE neurons in sensing the midbody curvature. Based on the *C. elegans* neuronal morphology (White et al., 1986), PDE neurons have short ciliated dendrites along the dorsal side of the posterior body and long axons travelling across the entire body along the ventral side. Since the proprioception receptive field of PDE seems to be the midbody region, we thus suggest

that it is the axons rather than the dendrites that play the proprioceptive role in PDE neurons.

CURVATURE COMPENSATION REQUIRES D2-LIKE DOPAMINE RECEPTOR DOP-3 IN AVK NEURONS

Our results so far demonstrated that PDE neurons sense proprioceptive inputs from midbody and regulate a dopaminergic pathway that is required for curvature compensation. To better understand this pathway, we next set to determine what other cellular and molecular components were responsible for curvature compensation downstream of the dopamine signaling from PDE neurons.

First, we determined that the D2-like dopamine receptor DOP-3 is required for mediating dopamine effects for curvature compensation. By constraining worms' midbody in the microfluidic channel, we examined curvature compensation in animals each lacking a single type of dopamine receptor (DOP-1 through DOP-4; **Fig. 3.6A**) and all combinations of the DOP-1, DOP-2, and DOP-3 receptors (**Fig. 3.6B**). We found that the *dop-3* mutation had a significant defective effect on curvature compensation in any genetic background, and adding exogenous dopamine did not restore compensatory behavior in *dop-3* mutants. Our single- and double-mutant analysis also showed that mutants that did not contain *dop-3* mutation did show normal curvature compensation like wild-type animals. Furthermore, we examined the effect of *dop-3* mutation on the curvature compensatory response to optogenetic perturbation in the midbody. By performing the earlier described optogenetic muscle perturbation experiments on the *dop-3* mutants expressing Muscle::NpHR and Muscle::ChR2, we found *dop-3* mutants again displayed significant defects in curvature compensation triggered by midbody muscle inhibition (**Figs. 3.6C-E**) or stimulation (**Figs. 3.6F-H**).

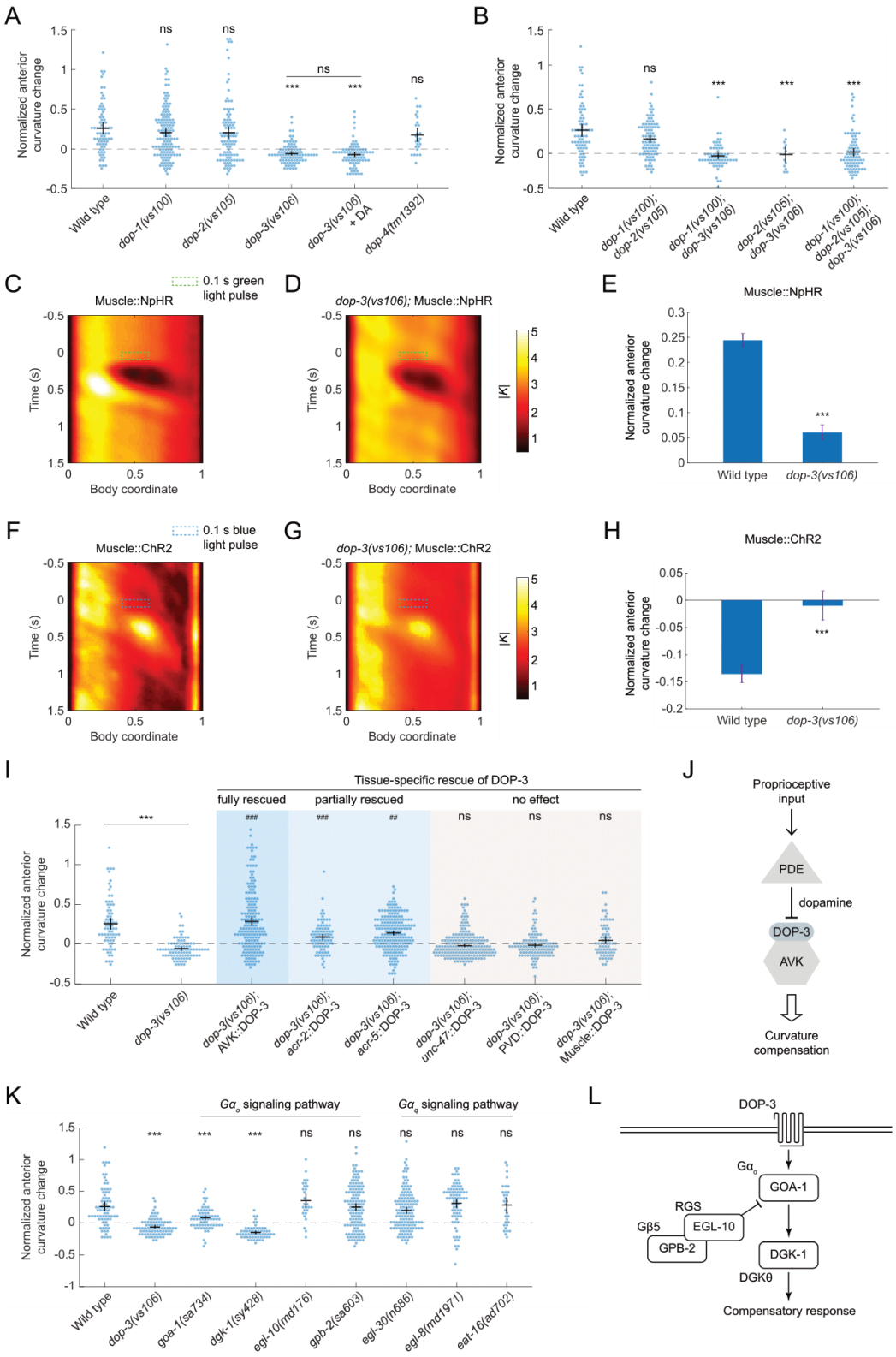
A
Normalized anterior curvature change
ns
ns
ns
***
***
ns
Wild type
dop-1(vs100)
dop-2(vs105)
dop-3(vs106)
dop-3(vs106) + DA
dop-4(tm1392)
B
Normalized anterior curvature change
ns
***
***
***
Wild type
dop-1(vs100); dop-2(vs105)
dop-1(vs100); dop-3(vs106)
dop-2(vs105); dop-3(vs106)
dop-1(vs100); dop-2(vs105); dop-3(vs106)
C
0.1 s green light pulse
Muscle::NpHR
Time (s)
Body coordinate
D
dop-3(vs106); Muscle::NpHR
Time (s)
Body coordinate
|K|
E
Muscle::NpHR
Normalized anterior curvature change
***
Wild type
dop-3(vs106)
F
0.1 s blue light pulse
Muscle::ChR2
Time (s)
Body coordinate
G
dop-3(vs106); Muscle::ChR2
Time (s)
Body coordinate
|K|
H
Muscle::ChR2
Normalized anterior curvature change
***
Wild type
dop-3(vs106)
I
Tissue-specific rescue of DOP-3
fully rescued
partially rescued
no effect
***
###
###
##
ns
ns
ns
Normalized anterior curvature change
Wild type
dop-3(vs106)
dop-3(vs106); AVK::DOP-3
dop-3(vs106); acr-2::DOP-3
dop-3(vs106); acr-5::DOP-3
dop-3(vs106); unc-47::DOP-3
dop-3(vs106); PVD::DOP-3
dop-3(vs106); Muscle::DOP-3
J
Proprioceptive input
PDE
dopamine
DOP-3
AVK
Curvature compensation
K
Gα_o signaling pathway
Gα_q signaling pathway
***
***
***
ns
ns
ns
ns
ns
Normalized anterior curvature change
Wild type
dop-3(vs106)
goa-1(sa734)
dgk-1(sy428)
egl-10(md176)
gpb-2(ss603)
egl-30(n686)
egl-8(md1971)
eat-16(ad702)
L
DOP-3
Gα_o
GOA-1
RGS
Gβ5
EGL-10
GPB-2
DGK-1
DGKθ
Compensatory response

Figure 3.6. Curvature compensation requires D2-like dopamine receptor DOP-3 in AVK neurons.

(A and B) Analysis of curvature compensatory response to midbody constraint for dopamine receptor knockout single (A) and double/triple (B) mutants. (A) D-2 like receptor DOP-3 is required for curvature compensation. Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for wild type and dopamine receptor knockout single mutants *dop-1(vs101)*, *dop-2(vs105)*, *dop-3(vs106)*, and *dop-4(tm1392)* under indicated conditions. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each indicated condition. ***p<0.001 when compared with wild type, Dunnett's multiple comparison tests; ns: not significant when comparing *dop-3* mutants in the absence and presence of exogenous 50 mM dopamine, Student's *t* test. (B) Double/triple mutants with DOP-3 receptor knockout showed impaired curvature compensatory response to midbody constraint. Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for *dop-1 dop-2*, *dop-1 dop-3*, *dop-2 dop-3* double mutants and *dop-1 dop-2 dop-3* triple mutants, compared with wild-type animals. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each indicated condition. ***p<0.001, Dunnett's multiple comparison tests.

(C-E) *dop-3* mutants showed impaired curvature compensatory response to midbody curvature decrease induced by transient optogenetic muscle inhibition. (C and D) Kymographs of mean absolute curvature around 0.1 s illuminations (green dashed box) for animals expressing Muscle::NpHR in wild type (C, same data as used in **Fig. 3.1F**) and *dop-3* mutants (D, n = 183 trials using 31 worms). (E) Normalized anterior curvature change of the first post-illumination curvature peak for animals expressing Muscle::NpHR in wild type and *dop-3* mutants (same data as used in C and D, respectively), mean ± SEM. ***p<0.001, Student's *t* test. (F-H) *dop-3* mutants

showed impaired curvature compensatory response to midbody increase induced by transient optogenetic dorsal muscle stimulation.

(F and G) Kymographs of mean absolute curvature around 0.1 s illuminations (blue dashed box) for animals expressing Muscle::ChR2 in wild type (F, same data as used in **Fig. 3.2D**) and *dop-3* mutants (G, n = 213 trials using 33 worms).

(H) Normalized anterior curvature change of the first post-illumination curvature peak for animals expressing Muscle::ChR2 in wild type and *dop-3* mutants (same data as used in F and G, respectively), mean ± SEM. ***p<0.001, Student's *t* test.

(I) Analysis of curvature compensatory response to midbody constraint for *dop-3* mutants with rescue of *dop-3* function by transgenic expression in different tissues. The impaired curvature compensation of *dop-3* mutants was fully rescued by transgenic expression of *dop-3* function in AVK neurons (via promoter *flp-1*), and partially rescued by transgenic expression of *dop-3* function in cholinergic neurons (via promoter *acr-2*) and B-type motor neurons (via promoter *acr-5*). Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for *dop-3* mutants with *dop-3* function rescued by transgenic expression in AVK neurons (*Pflp-1(trc)::DOP-3*), cholinergic neurons (*Pacr-2::DOP-3*), B-type motor neurons (*Pacr-5::DOP-3*), GABAergic neurons (*Punc-47::DOP-3*), PVD neurons (*Pser-2-prom3::DOP-3*), and body wall muscle cells (*Pmyo-3::DOP-3*), compared with wild type and *dop-3(vs106)* mutants. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each condition. ***p<0.001 when compared with wild type, ns: not significant and $^{\#\#\#}$p<0.001 when compared with *dop-3* mutants, Tukey-Kramer multiple comparison tests.

(J) A schematic model circuit showing how midbody proprioceptive input gets transduced to regulate anterior curvature compensation through DOP-3 dependent dopamine signaling from PDE to AVK neurons.

(K and L) Dopamine regulates compensatory response by binding to the receptor DOP-3 and activating GOA-1 $G\alpha_o$ signaling. (K) Curvature compensatory response analysis of mutants that disrupt the $G\alpha_o$ and $G\alpha_q$ signaling. Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for *dop-3(vs106)* mutants, $G\alpha_o$ signaling mutants *goa-1(sa734)*, *dgk-1(sy428)*, *egl-10(md176)*, *gpb-2(sa603)*, $G\alpha_q$ signaling mutants *egl-30(n686)*, *egl-8(md1971)*, *eat-16(ad702)*, compared with wild-type animals. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each condition. ***p<0.001, ns: not significant, Dunnett's multiple comparison tests. (L) Schematic representation of the $G\alpha_o$ protein signaling pathways that regulate the curvature compensatory response in *C. elegans* (adapted from Chase et al. 2004).

Second, we asked which specific cell types expressing DOP-3 are responsible for mediating the dopamine effect for curvature compensation. According to the *dop-3* gene expression in wild-type animals, DOP-3 receptors are expressed in various cell types, including GABAergic neurons, cholinergic motor neurons, mechanosensory neurons PVD, interneurons AVK, and body wall muscle cells (Chase et al., 2004; Oranth et al., 2018). Thus, using promoters active in the above cells, we expressed DOP-3 individually in those types of cells and tested the ability of such transgenes to rescue the impaired curvature compensation of *dop-3* mutants (**Fig. 3.6l**). Restoring DOP-3 expression in GABAergic neurons (promoter *Punc-47*), PVDs (promoter *Pser-2prom3*), or body wall muscles (promoter *Pmyo-3*) failed to rescue the *dop-3* defect. However, when DOP-3 was expressed in cholinergic (promoter *Pacr-2*), or B-type motor neurons (promoter *Pacr-5*), *dop-3* animals exhibited partial rescue, and only when we restored DOP-3 expression specifically to AVK in *dop-3* mutants (via promoter *Pflp-1*), the animal's curvature compensation was fully restored to the wild-type level (**Fig. 3.6l**). Thus our rescue experiments suggest that DOP-3 receptors in AVK (and potentially some cholinergic motor neurons) mediate the proprioception-triggered dopaminergic signals from upstream PDE neurons to regulate curvature

compensatory behavior (**Fig. 3.6J**).

Third, we examined curvature compensation in mutants that disrupted the downstream G protein signaling of DOP-3 and DOP-1, the $G\alpha_o$ and $G\alpha_q$ pathways (**Fig. 3.6K**). Mutants with deficiency in GOA-1, the *C. elegans* ortholog of the G protein $G\alpha_o$ (coupled to DOP-3), exhibited impaired curvature compensation. The similar defect was found in the mutants with deficiency in the RGS protein DGK-1, a putative downstream effector of GOA-1 $G\alpha_o$. In contrast, mutants with deficiencies in EGL-10, GTPase activating protein that inhibits GOA-1 $G\alpha_o$, and GPB-2, an obligate subunit of EGL-10 RGS, exhibited wild-type level of curvature compensation. Mutants with deficiencies in proteins that are associated with $G\alpha_q$ (coupled to DOP-1) all exhibited normal curvature compensation. Earlier studies suggested that DOP-3 and DOP-1 have opposing effects on locomotion by signaling through these two antagonistic G protein pathways, respectively (Chase et al., 2004). Given that the DOP-3 receptors, but not the DOP-1 receptors, were found necessary for curvature compensation behavior (**Figs. 3.6A** and **3.6B**), our results are indeed consistent with the previously proposed model in which DOP-3 affects locomotion by activating the $G\alpha_o$ signaling pathway (**Fig. 3.6L**; Chase et al., 2004).

FMRFAMIDE-LIKE NEUROPEPTIDE FLP-1, RELEASED BY AVK, REGULATES SMB MOTOR NEURONS VIA RECEPTOR NPR-6 TO MODULATE ANTERIOR BENDING AMPLITUDE

So far we have demonstrated that PDE neurons sense midbody curvature and that the dopamine/DOP-3 signaling pathway from PDE to AVK is required for curvature compensation. AVK mediates FLP-1 FMRFamide-like neuropeptide signaling via release of dense core vesicles (DCVs) to modulate locomotion in response to various sensory inputs (Hums et al., 2016; Oranth et al., 2018), and the deletion of *flp-1* gene results in loopy undulation with exaggerated sinusoidal waveform in both agar surface (Nelson et al., 1998) and liquid environments.

Do AVK and FLP-1 neuropeptide signaling also play a role in curvature compensation? To

answer that question, we first examined worms with AVK neurons eliminated by laser ablation (see *Methods*) and transgenic animals with AVK expressing tetanus toxin that blocks synaptic vesicle release (*Pflp-1::TeTx*). We found that either ablating AVK or blocking synaptic transmission from AVK led to superficially wildtype locomotion but strongly compromised curvature compensatory responses to the microfluidic constraint in the midbody (**Fig. 3.7A**). We also tested *flp-1(yn4)* and *flp-1(sy1599)* mutants both lacking FLP-1 neuropeptides, and *unc-31(e169)* mutants lacking CAPS (Ca^{2+}-activated protein for secretion, required for all neuropeptides release). We again found significant defects in curvature compensation in these mutant animals (**Fig. 3.7B**). These results suggest that FLP-1 peptide signaling from AVK neurons is required for animals to exhibit normal curvature compensation.

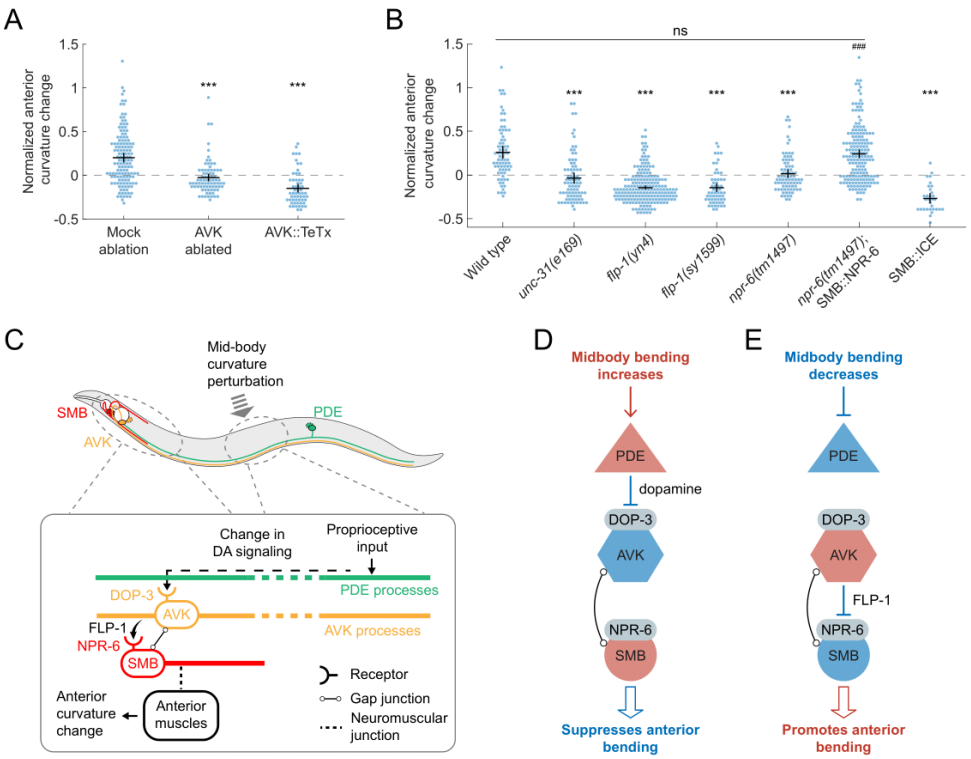

Figure 3.7. FMRFamide-like neuropeptide FLP-1, released by AVK, regulates SMB motor neurons via receptor NPR-6 to modulate anterior bending amplitude.

(A and B) Curvature compensation requires AVK- released FLP-1 regulation of SMB neuronal activity via receptor NRP-6. (A) Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for animals with laser ablation of AVK and transgenic animals expressing tetanus toxin in AVK (*Pflp-1::TeTx*), compared with the mock ablation control group. Each data point is the mean of a 3 s period of constrained locomotion pooled across 11 or more animals for each case. ***p<0.001, Dunnett's multiple comparison tests. (B) Data denote mean ± SEM of the normalized anterior curvature change in response to midbody constraint for wild type, mutants *unc-31(e169)*, *flp-1(yn4)*, *flp-1(sy1599)*, *npr-6(tm1497)* and *npr-6* mutants with *npr-6* function rescued by transgenic expression in SMB neurons, and animals lacking SMB (ablation by caspase). Each data point is the mean of a 3 s period of constrained locomotion pooled across 12 or more animals for each case. ***p<0.001, ns: not significant, when compared with wild type, ###p<0.001 when compared with *npr-6* mutants, Tukey-Kramer multiple comparison tests.

(C-E) Schematic models for the mechanisms underlying the curvature compensatory response. (C) (*Upper*) An anatomical representation showing the relative positions of PDE (green), AVK (yellow), and SMB neurons (red) and their soma/processes within a worm body. (*Lower*) A zoom-in representation proposing the underlying neuronal pathway for curvature compensation. Dopaminergic neurons PDE transduce the proprioceptive input from the midbody curvature perturbation and signal to AVK neurons via dopamine signaling through DOP-3 receptors. In the anterior region, the AVK neurons signal via FLP-1 neuropeptide to negatively regulate the head-bending-suppressing motor neurons SMB via NPR-6 receptors. Since PDE negatively regulates AVK via dopamine, AVK negatively regulates SMB via FLP-1 peptides, and SMB negatively regulates head bending, perturbation to the midbody bending leads to a net negative regulatory effect on the anterior bending, as illustrated schematically in two separated scenarios (D and E). Red and blue colors indicate active and inhibited neuronal states, respectively.

The AVK neurons, however, are interneurons which do not directly innervate muscles to drive body bending. To further probe the circuit underlying curvature compensation, we asked what downstream cells constitute the remaining pathway that directly affect the anterior body bending amplitude while being regulated by the upstream FLP-1 signaling from AVK.

Some clues provided by previous studies prompted us to speculate that SMB, a class of head motor neurons, might be such a candidate residing within the circuit: AVK synapses via both electrical and chemical couplings onto SMB (Hums et al., 2016; White et al., 1986), whose major function is regulating head and neck muscles and thus setting the overall amplitude of sinusoidal forward movement (Gray et al., 2005). Specifically, SMB activity is regulated by AVK-released FLP-1 signaling through the inhibitory receptor NPR-6 (Oranth et al., 2018).

Thus, we sought to determine the role of SMB as well as its peptide-regulated activity in mediating anterior bending amplitude during curvature compensation. First, we ablated SMB neurons by ICE expression, which led to deeply flexed head swings and a resulting significant increase in body bending amplitude (Gray et al., 2005). Despite having loopy sinusoidal movement, worms lacking SMBs showed a dramatically impaired curvature compensation (**Fig. 3.7B**). Next, we examined *npr-6(tm1497)* mutants with and without the transgene that restores NPR-6 receptors in SMB neurons. We found that *npr-6* mutants showed impaired curvature compensation, and that this phenotype was rescued by expressing NPR-6 receptors in SMBs in *npr-6* mutants (**Fig. 3.7B**). These experiments support the instructive role of SMB motor neurons in curvature compensation, which is to modulate anterior bending amplitude under the regulation of AVK-released FLP-1 neuropeptide signaling.

Taken together, we have demonstrated that *C. elegans* uses proprioception to mediate homeostatic control of locomotor amplitude during forward movement. Our results support the following neural network for this motor control (**Fig. 3.7C**): (1) dopaminergic neurons PDE

transduce the proprioceptive input from the midbody curvature perturbation, and regulate AVK activity via dopamine/DOP-3 signaling. (2) Downstream of dopamine signaling, AVK mediates FLP-1 peptide signaling, through which SMB is negatively regulated via NPR-6 receptors. (3) the head motor neurons SMB, affected by FLP-1 signaling, directly modulates bending amplitude of the anterior region.

CURVATURE COMPENSATION MECHANISM IS CONSISTENT WITH GAIT ADAPTATION OF BENDING AMPLITUDE IN RESPONSE TO MECHANICAL LOAD

C. elegans can move through water or across moist substrates like agarose gels. At the scale of *C. elegans* size and speed, forces due to surface tension experienced by a crawling worm on the agar surface are 10,000-fold larger than forces due to viscosity experienced by a swimming worm in water (Sauvage, 2007). As a versatile limbless swimmer/crawler, *C. elegans* performs undulatory movements with appropriate kinematic patterns to propel itself adaptively through contexts with a wide range of mechanical load (Berri et al., 2009; Fang-Yen et al., 2010). Here we asked whether the curvature compensation mechanism contributes to this gait adaptation.

To do this, we measured worm undulatory parameters (frequency, wavelength, and curvature amplitude) in media of varying viscosities, using several core strains that we had examined for curvature compensation.

First, by immersing worms into solutions of five different viscosities ranging from 10 to 27,900 mPa·s, we tested gait adaptions of wildtype animals and *dop-3(vs106)* mutants with and without transgenes expressing DOP-3 in AVK. For all these strains, increasing viscosity of the medium caused gait transitions from a swimming gait to a crawling gait, characterized by decreasing trends in frequency (**Fig. S3.4A**) and wavelength (**Fig. S3.4B**) and increasing trends in curvature amplitude (**Fig. S3.4C**). Despite the overall similarity in gait transition among the three strains, mutants *dop-3(vs106)* displayed a significantly higher curvature amplitude in comparison with wild

type in every tested medium, and this difference got restored in *dop-3* mutants with transgenic expression AVK::*dop-3* (**Fig. S3.4C**).

Next, besides the above three strains, we tested four additional strains by putting them in two intermediate viscous solutions (1390 and 9079 mPa·s), with a focus on their curvature amplitude during locomotion. The additional strains included transgenic worms expressing AVK::TeTx, mutants *flp-1(sy1599)*, and mutants *npr-6(tm1497)* with and without transgenes expressing NPR-6 in SMB. By comparing curvature amplitudes of locomotion in 1390 mPa·s solutions, we found strains defective in curvature compensation had a relatively higher curvature amplitude than strains showing normal curvature compensation (**Fig. S3.4D**).

To further compare gait adaptation between these strains, an index of curvature amplitude adaptation is provided by the difference between the curvature amplitudes in high (9079 mPa·s) and low (1390 mPa·s) viscous solutions divided by the curvature amplitude in the low viscous solution: $(K_{high\ vis} - K_{low\ vis})/K_{low\ vis}$ (**Fig. S3.4E**). A greater index indicates higher plasticity of curvature amplitude modulation adapting to mechanical load. We found that strains defective in curvature compensation had a significantly smaller adaptation index in comparison with those who showed normal curvature compensation (**Fig. S3.4E**).

The above results implied a consistency between curvature compensation mechanism and gait adaptation of bending curvature: animals bearing defects in curvature compensation exhibit higher curvature amplitude but smaller plasticity of curvature amplitude modulation in response to mechanical load.

The proprioception-mediated curvature compensation mechanism provides an explanation for these observations. In low viscosities, a worm tends to have a large bending amplitude due to the relatively small constraint and curvature compensation then leads to a smaller anterior

bending amplitude. Since undulatory waves propagate posteriorly, this flattens out the overall amplitude of sinusoidal movement. In contrast, worms defective in curvature compensation lack this negative regulation of undulatory amplitude and thus tend to have a larger bending amplitude.

When shifting to a more viscous environment, a worm's bending amplitude decreases due to a larger constraint it experiences. Curvature compensation leads to an increase in the anterior bending amplitude which counteracts the decreasing tendency. Worms with impaired curvature compensation, however, fail this feat and show a smaller change in curvature amplitude (i.e., a smaller adaptation index).

METHODS

Table 3.1. Key resources table for Chapter 3

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Bacterial and Virus Strains		
E. coli OP50	CGC	OP50-1
Chemicals, Peptides, and Recombinant Proteins		
All-*trans* retinal (ATR)	Sigma-Aldrich	Cat#R2500
Dextran from *Leuconostoc mesenteroides*	Sigma-Aldrich	Cat#D5376
Poly-dimethyl siloxane (PDMS) + curing agent	Dow Corning	SylGard 184
Bovine Serum Albumin	Sigma-Aldrich	N/A
Gibson Assembly Master Mix	NEB	N/A
Experimental Models: Organisms/Strains		
qhIs1[Pmyo-3::NpHR::eCFP]; qhIs4[Pacr-2::wCherry]	This paper	YX148
hpIs199[Pmyo-3::ChR2::eGFP]	Zhen lab	ZM5398
mec-4(u253)	CGC	TU253
mec-4(e1611)	CGC	CB1611
mec-10(e1515)	CGC	CB1515
del-1(ok150)	CGC	NC279
unc-8(e15lb145)	CGC	MP145
trp-4(sy695)	CGC	TQ296
trpa-1(ok999)	CGC	TQ233
unc-9(fc16); hpEx803[Prgef-1::unc-9cDNA + Podr-1::GFP]	Zhen lab	ZM2509
cat-2(e1112)	CGC	CB1112
tph-1(n4622)	CGC	MT14984
tdc-1(n3421)	CGC	MT10549

cat-2(e1112); qhIs1[Pmyo-3::NpHR::eCFP]	This paper	YX287
cat-2(e1112); hpIs199[Pmyo-3::ChR2::eGFP]	This paper	YX289
akEx387[dat-1::GFP; dat-1::ICE]	Villu Maricq lab	VM6365
otIs181[Pdat-1::mCherry; ttx-3::mCherry]; maIs188[Pmir-288::GFP]	Kang lab	TV23560
akEx387[dat-1::GFP; dat-1::ICE]; otIs181[Pdat-1::mCherry; ttx-3::mCherry]	This paper	YX296
egIs1[Pdat-1::GFP]	CGC	BZ555
kyEx6101[Pdat-1::TeTx::sl2GFP]	Bargmann lab	YX297
flvEx127[Pdat-1::GCaMP6m; Pmyo-3::mCherry]	Flavell lab	SWF331
unc-54(e1092); flvEx127[Pdat-1::GCaMP6m; Pmyo-3::mCherry]	This paper	YX298
dop-1(vs101)	CGC	LX636
dop-2(vs105)	CGC	LX702
dop-3(vs106)	CGC	LX703
dop-4(tm1392)	CGC	FG58
dop-2(vs105); dop-1(vs100)	CGC	LX706
dop-1(vs100); dop-3(vs106)	CGC	LX705
dop-2(vs105); dop-3(vs106)	CGC	LX704
dop-2(vs105); dop-1(vs100); dop-3(vs106)	CGC	LX734
dop-3(vs106); qhIs1[Pmyo-3::NpHR::eCFP]	This paper	YX288
dop-3(vs106); hpIs199[Pmyo-3::ChR2::eGFP]	This paper	YX290
dop-3(vs106); qhEx263[Pser-2-prom3::dop-3(+) + Punc-47::GFP]	This paper	YX291
dop-3(vs106); qhEx264[Punc-47::dop-3(+) + Punc-47::GFP]	This paper	YX292
dop-3(vs106); qhEx265[Pacr-2::dop-3(+) + Punc-47::GFP]	This paper	YX293
dop-3(vs106); qhEx266[Pmyo-3::dop-3(+) + Punc-47::GFP]	This paper	YX294
dop-3(vs106); qhEx267[Pacr-5::dop-3(+) + Punc-47::GFP]	This paper	YX295
goa-1(sa734)	CGC	JT734
dgk-1(sy428)	CGC	JT748
egl-10(md176)	CGC	MT8504
gpb-2(sa603)	CGC	JT603
egl-30(n686)	CGC	MT1434
egl-8(md1971)	CGC	RM2221
eat-16(ad702)	CGC	DA702
dop-3(vs106); zxIs20[Pdat-1::ChR2(H134R)::mCherry; Pmyo-2::mCherry]; zxEx1063[Pflp-1(trc)::DOP-3::SL2::GFP; Pmyo-3::CFP]	Gottschalk lab	ZX2201
flp-1(yn-4)	CGC	NY16
flp-1(sy1599)	Ringstad lab	PS8997
flp-11(tm2706)	CGC	HBR507
ynIs72[Pflp-1::GFP]	CGC	NY2072
wzEx664[Pflp-1::TeTx; Pflp-1::mCherry]	Ringstad lab	FQ2747
npr-6(tm1497)	National	F41E7.3

	Bioresource Project for the Experimental Animal "Nematode *C. elegans*"	
npr-1(ky13)	CGC	CX4148
zxIs29[Pflp-12::Cre; Podr-2(18)::LoxP::ICE; Pmyo-2::mCherry]	Gottschalk lab	ZX3058
npr-6(tm1497); zxEx850[Pflp-12::LoxP::LacZ::STOP::LoxP::NPR-6::SL2::GFP; Podr-2(18)::Cre]	Gottschalk lab	ZX2037
Oligonucleotides		
Forward primer for amplifying *dop-3* sequence (GCCAAAGGACCCAAAGGTATGTTTCG)	Ringstad lab	SAZ86
Reverse primer for amplifying *dop-3* sequence (CCGATCTTTCTTGCATCGTGCTCATC)	Ringstad lab	SAZ87
Forward primer for inserting *dop-3* sequence (GTTTGTCAAGAGTTTCGAGGACGG)	Ringstad lab	SAZ88
Reverse primer for inserting *dop-3* sequence (CAAGGGTCCTCCTGAAAATGTTCTAT)	Ringstad lab	SAZ89
Recombinant DNA		
pDC50*(unc-47::dop-3)* [75 ng/µL]	Koelle lab	N/A
pDC66*(unc-47::GFP)* [75 ng/µL]	Koelle lab	N/A
pYX36*(ser-2-prom3::dop-3)* [75 ng/µL]	This paper	N/A
pYX37*(myo-3::dop-3)* [75 ng/µL]	This paper	N/A
pYX38*(acr-2::dop-3)* [75 ng/µL]	This paper	N/A
pYX39*(acr-5::dop-3)* [75 ng/µL]	This paper	N/A

CONTACT FOR REAGENT AND RESOURCE SHARING

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Dr. Christopher Fang-Yen (chfan@seas.upenn.edu).

EXPERIMENTAL MODEL AND SUBJECT DETAILS

C. elegans were cultivated at 20°C on nematode growth media (NGM) plates seeded with *Escherichia coli* strain OP50 using standard methods (Sulston and Hodgkin, 1988). For optogenetic experiments, animals were cultivated in darkness on plates with OP50 containing 800 µM all-*trans* retinal (ATR). All experiments were performed with 1-day-old adult hermaphrodites synchronized by hypochlorite bleaching.

Wild-type animals were Bristol strain N2. Transgenic strains for tissue-specific rescue of *dop-*

3 function were generated by microinjection of a transgene of DNA clones and a fluorescent co-injection marker (see Key resources table for Chapter 3 for plasmid concentrations).

METHOD DETAILS

MOLECULAR BIOLOGY

pYX36(*ser-2-prom3::dop-3*), pYX37(*myo-3::dop-3*), pYX38(*acr-2::dop-3*), pYX39(*acr-5::dop-3*): Plasmid constructs are for tissue-specific expression of *dop-3* function (**Fig. 3.6J**). The *Pser-2-prom3* (PVD), *Pmyo-3* (body-wall muscles), *Pacr-2* (cholinergic neurons), and *Pacr-5* (B-type motor neurons) promoters are used for cell-specific expression, which were constructed from donor plasmids *Pser2prom3::GFP* (gift of Kang lab), *Pmyo-3::RCaMP1h* (made by Gottschalk lab), *Pacr-2::GFP* (gift of Koelle lab), and *Pacr-5::Arch::GFP* (gift of Takagi lab), respectively. *dop-3* gene sequence was amplified from pDC50(*unc-47::dop-3*) using primers SAZ86 and SAZ87. Constructs containing promoter sequences were amplified from the corresponding donor plasmids using primers SAZ88 and SAZ89. Reconstruction procedures were conducted using Gibson Assembly method (Gibson Assembly Master Mix, NEB). Resulting plasmids were verified by sequencing (ABI 3730XL sequencer, Penn Genomic Analysis Core).

BEHAVIORAL ASSAYS

Optogenetic Manipulation Experiments

For experiments with optogenetic manipulation (**Figs. 3.1C-G, 3.1S1, 3.2A-F, 3.4D-I**, and **3.6C-H**), worms were prepared in a viscous solution [17% (by mass) dextran in NGM buffer; 120 mPa·s in viscosity] confined within chambers formed by a microscope slide and a coverslip separated by 125-µm-thick polyester shims (9513K42, McMaster-Carr).

Optogenetic experiments were carried out on a Leica DMI4000B microscope coupled with a motorized stage (CTR4000, Leica). Image sequences were recorded at 40 Hz with a sCMOS camera (optiMOS, Photometrics) under 10X magnification (Leica Plan Fluotar; N.A., 0.30) with

dark field illumination provided by red LEDs. We used a custom-built optogenetic targeting system (Fouad et al., 2018) to perform spatially selective optogenetic manipulation on worm's muscle activity during locomotion. To optogenetically inhibit or stimulate muscles, we used a 532-nm solid-state laser (GL532T3-300, SLOC) with irradiance at 16 mW/mm^2 or a 473-nm solid-state laser (BL473T3-150, SLOC) at 3.5 mW/mm^2, respectively.

For optogenetic muscle inhibition (**Figs. 3.1C-G, 3.1S1, 3.2F** green data points, **3.4D-F**, and **3.6C-D**) and stimulation (**Figs. 3.2A-E, 3.2F** blue data points, **3.4G-I**, and **3.6F-H**), we used wild-type and mutant animals with body wall muscles expressing (via *Pmyo-3*) inhibitory opsin *NpHR* and excitatory opsin *ChR2*, respectively. During experiments, each individual animal was illuminated at the middle region (0.4-0.6 body coordinate; illuminating both sides for inhibition, dorsal side for stimulation) by a brief laser pulse (0.1 s duration, unless otherwise stated) repeated 10 times with 6 s interval between successive pulses. We used a custom-written C++ software (Fouad et al., 2018) to perform real-time identification of the worm with its boundary and centerline detected by gray level thresholding during image acquisition. The head-and-tail and dorsoventral orientations of a worm were noted visually during the recording. The calculated information was saved to disk along with the corresponding image sequences. Postprocessing of the behavioral data is discussed in later section.

Microfluidic-Based Experiments

Besides optogenetic experiments, other behavioral assays (**Figs. 3.3B-E, 3.3S1, 3.4A-C, 3.4J-K, 3.6A-B, 3.6I, 3.6K, 3.7A-B**) were performed based on a custom-made microfluidic polydimethylsiloxane (PDMS) device fabricated using soft lithography techniques. Video sequences were recorded at 30 Hz with a 5-megapixel CMOS camera (DMK33GP031, The Imaging Source) and a C-mount lens (Nippon Kogaku NIKKOR-H; effective focal length, 28 mm) using IC Capture software (The Imaging Source). Red LED rings (outer size, 80 mm; Qasim)

surrounding the device provided dark field illumination.

As shown by the schematic in **Fig. 3.3A**, the microfluidic chamber consists of 2000-µm-wide open areas which are connected by two parallel narrow channels (60 µm x 200 µm). The microfluidic chamber was loaded with NGM buffer with 0.1% (by mass) bovine serum albumen (BSA) added to the solution to prevent worms from adhering to chamber surfaces or tubing. By using a 3-way luer valve (Cole-Parmer) and polyethylene tubing (Saint-Gobain), the worm chamber of the microfluidic device was connected in parallel to a 1-mL syringe and a reservoir containing NGM buffer. The tubing between the chamber and the syringe was mildly compressed by a screw-bolt unit where the spacing in between can be finely adjusted.

For microfluidic-based behavioral experiments, young adults were first transferred to food-free NGM buffer for ~5 min to wash carried-over bacteria off the animals. Then animals were pipetted from the buffer into the inlet of the microfluidic chamber. To translate worms to the field of view (approximately 4 mm x 3 mm) for video recording, we used the syringe on the inlet to apply pressure and vacuum. For an individual animal within the field of view, behavior images were recorded for 3 minutes during which the worm alternating between constrained locomotion and free locomotion (as shown in **Fig. 3.3A**) with each mode lasting for ~30 s. The worm position within the chamber was manually controlled by slowly twisting the screw-bolt unit. Post-acquisition behavioral quantification is discussed in the section below.

BEHAVIORAL DATA QUANTIFICATION

General Postprocessing

Postprocessing of the behavioral data from the two experiments described above was performed using MATLAB custom software (MathWorks) similar to previous reports (Fouad et al., 2017; Ji et al., 2021c). With the worm centerline in each image smoothed via a cubic spline fit, the body curvature κ is calculated as the dot product of the unit normal vector to the centerline and the

derivative of the unit tangent vector along the centerline with respect to the body coordinate. The normalized curvature K is the product of κ and the worm body length L derived from the length of worm centerline. We excluded curvature in the anterior and posterior 5% body regions to avoid high frequency movements at the tips of the worm. The moving direction of a worm was determined by the gradients in the curvature over time and body coordinate, and image sequences during which the worm moved forward for at least 4 seconds were selected for analysis. The curvature dynamics of the anterior, middle, and posterior regions were defined as the average of the normalized curvature over 0.1-0.3, 0.4-0.6, and 0.7-0.9 body coordinates, respectively.

Although the method for calculating locomotor dynamics was shared for both experiments, the specific steps for quantifying the effects of optogenetic or microfluidic manipulations on worm undulatory amplitude were different due to the largely different durations of disturbances used in the two types of experiment. Detailed descriptions are presented below.

Quantifying Optogenetic Behavioral Data

To quantify the effect of optogenetic perturbations on worm undulatory amplitude, we calculated the curvature amplitude of the anterior and middle regions, respectively, around each trial of laser illumination. Regarding each body region, we used the MATLAB function *findpeaks* to identify local extrema along the time-varying curvature profiles (as shown in **Figs. 3.1E** and **3.2C**). Around each illumination, $|K_{-1}|$ denotes the absolute value of the last pre-illumination curvature peak which was used to define the baseline curvature amplitude; $|K_{+n}|$ denotes the absolute value of the n^{th} post-illumination peak. The corresponding normalized curvature change, defined by $\Delta K_{+n}/|K_{-1}| = (|K_{+n}| - |K_{-1}|)/|K_{-1}|$, was used to quantify the change in curvature amplitude induced by optogenetic perturbations as shown in **Figs. 3.1G, 3.2E, 3.2F, 3.4F, 3.4I, 3.6E**, and **3.6H**.

Quantifying Microfluidic Behavioral Data

To quantify the effect of microfluidic-channel constraint on worm undulatory amplitude, the whole-body curvature amplitude during constrained locomotion was computed and compared with curvature amplitude during free locomotion.

Regarding free locomotion, we analyzed worm locomotor dynamics to generate an averaged curvature amplitude profile as a function of body coordinate. To do that, we divided the worm body coordinate into 10 even sections from head to tail (starting from 0.05 to 0.95, as movements of the anterior and posterior 5% regions were omitted). For each individual section, we calculated the average of the normalized curvature over the body coordinate of the section for all periods of free locomotion. Local extrema along each time sequence of curvature were identified (via peak finding method), and the mean of the absolute value of these local extrema was defined as the curvature amplitude at the body coordinate defined by the mid-point of the section (e.g., 0.1 for section 0.05-0.15). After computing curvature amplitudes for the ten sections, the whole-body averaged curvature amplitude profile, $A_{free}(s)$, was obtained through a linear 1-D interpolation with 100 sample points of values computed across the worm body.

Regarding constrained locomotion, we first used a 3-second time window to divide video sequences of constrained movement into individual short sequences. Due to the unavoidable disturbances in controlling worm position by syringe pump, the body region being constrained could not consistently maintain in the middle and occasionally varied a lot. To record the relative position of the constraint with respect to the worm body (as shown by the gray lines in **Fig. 3.3D**), we manually marked the channel position in each image sequence by drawing a rectangle with its short sides aligned at the two limits of the channel, respectively.

To calculate normalized curvature change in response to mid-body constraint (**Fig. 3.3E, 3.3S1, 3.4C, 3.4J-K, 3.6A-B, 3.6I, 3.6K**, and **3.7A-B**), we only counted periods during which the

anterior and posterior limits of the narrow channel were consistently within 0.35-0.65 body coordinate, and denoted the corresponding curvature dynamics as K_{const}. We took the maximum value of $|K_{const}(s,t)|$ in the direction of time for all qualified short periods and defined the resulting quantity, $A_{const}(s) = \max_t |K_{const}(s,t)|$, as the curvature amplitude profile of individual periods. The normalized curvature change of an individual period is thus represented by $A_{const}(s)/A_{free}(s) - 1$. Additionally, the normalized anterior, mid-body, and posterior curvature changes of an individual period are $\langle A_{const}(s)/A_{free}(s)\rangle|_{0.1}^{0.3} - 1$, $\langle A_{const}(s)/A_{free}(s)\rangle|_{0.4}^{0.6} - 1$, and $\langle A_{const}(s)/A_{free}(s)\rangle|_{0.6}^{0.8} - 1$, respectively ($\langle X(s)\rangle|_a^b$ denotes the average of X in interval [a b]).

LASER ABLATION OF NEURONS

Cell ablation experiments (ablation of PDE or AVK neurons) were carried out with a custom-built thermal laser ablation system (Fouad et al., 2021) based on an inverted microscope (Nikon TE-2000). Transgenic animals (*Pdat-1::GFP* for PDE ablation; *Pflp-1::GFP* for AVK ablation) at third larva stage were immobilized on 10% agar pads using 50 nm polystyrene beads and mounted on the microscope. The GFP-labeled somas of target neurons (PDE or AVK) were visualized under GFP fluorescence optics and illuminated with 1~2 laser pulses (1.5 ms in duration, 400 mW in power) through a 63X oil-immersion objective. After ablation, animals were transferred to a fresh OP50 plate to recover overnight. On the next day, the illuminated animals were mounted on the system again to run a double-check on the elimination of target neurons. Confirmed animals were transferred back to seeded plates to resume growth for an additional day until they turned young adults during behavioral assays. Mock-ablation groups were mounted on the system but not irradiated with the laser.

PDE CALCIUM IMAGING IN MOVING OR PARALYZED ANIMALS

For recording PDE Ca^{2+} activity in freely behaving animals (**Figs. 3.5A-B**), transgenic worms expressing GCaMP in PDE neurons (*Pdat-1::GCaMP6m*), after getting off carried-over bacteria,

were picked onto a 5% agar pad with a few microliters of NGM buffer, and covered with a #1.5 cover glass, so that worms were between agar and cover glass. Worms moved relatively slower under this condition but still maintained normal body shape and locomotion. For recording PDE Ca^{2+} activity in paralyzed animals (**Figs. 3.5C-D**), *unc-54(e1092)* myosin heavy chain mutants expressing GCaMP in PDE neurons (*Pdat-1::GCaMP6m*) were loaded and restrained in a sinusoidal microfluidic channel which was filled with 17% (by mass) dextran solution. To induce varying body curvature, worm position within the channel was manually controlled by a syringe (1 mL in volume) connected to polyethylene tubing on the inlet.

Ca^{2+} imaging of PDE in freely behaving worms and muscularly paralyzed worms was performed on a Leica DMI3000 B microscope equipped with a motorized stage (CTR3000, Leica). The GCaMP6m protein in the PDE neurons was excited by the broadband excitation light derived from Leica EL6000. Worm body was visualized under a red dark field illumination provided by a built-in halogen lamp (LH107/2, Leica). To support simultaneous recording of Ca^{2+} activity and worm movement, green fluorescence emission and red dark field illumination were collected through a Leica Plan Apo 10X objective (working distance, 1 mm; N.A., 0.40), separated by a dual-view beam splitter (DV2, Photometrics) with a CFP/GFP filter set, projected onto an EMCCD sensor (Cascade 1K, Photometrics). The unbinned image sequences were streamed at 9 frames per second (fps) acquisition rate under 100 ms exposure time operated by MicroManager. Approximately 2 min of data were acquired for each animal.

Image sequences acquired in either of the worm preparations were processed offline using custom analysis routines. Briefly, each image in these sequences was split in half so that signals obtained through red and green channels were separated into individual sub-images. In the red sub-image sequences, each image was background-subtracted and thresholded to produce a binary image. The binary image sequences were used to quantify worm curvature dynamics

using similar methods as described above. With the computed curvature dynamics, the corresponding binary image sequences were computationally deformed from a worm shape into a skinny rectangle which provided a mask to crop out whole-body fluorescence signals from the green sub-image sequences. From the masked-out rectangular fluorescent images, regions of interest (ROIs) were selected on somas of the target neuron PDEs. GCaMP signals were measured as integrated fluorescence over the ROI, subtracted by a background value computed within each recording using a secondary ROI drawn around PDEs but lacking labeled neurons. To obtain normalized signals (**Figs. 3.5B** and **3.5D**), the GCaMP values were subtracted and then divided by a baseline F_0 value calculated per recording as the mean of the lowest 50% of GCaMP values. Image splitting, binarizing, and GCaMP signal extraction were conducted using software ImageJ. Curvature calculation and binary image deformation were performed using custom-written Python scripts (Ruba, Fang-Yen, unpublished).

QUANTIFICATION AND STATISTICAL ANALYSIS

Each transgenic and mutant strains were tested in at least two different experiments done on two different days within a week, and compared to control experiments done in parallel on the same days. All quantification has been explained within relevant sections of Methods. Specification of all statistics analysis is reported in the figure legends.

DATA AND SOFTWARE AVAILABILITY

Raw data for all experiments and behavioral analyzing software will be available upon request.

CHAPTER 4: CONCLUSION AND FUTURE DIRECTIONS

CONCLUSION

I consolidated laboratory experiments and theoretical modelling in pursuit of a system- and circuit-level understanding of how *C. elegans* generates and modulates locomotion.

In Chapter 2, with the help of optogenetic perturbation, behavioral quantification, and computational modeling, I performed different levels of analyses which can be used to constrain the space of hypotheses being evaluated, allowing to construct higher-level principles and structures in a motor circuit model. The quantitative agreement between the model and experiments-importantly including the perturbation experiments-suggests forward locomotion in worms can be understood as being driven by a relaxation oscillator. The proposed model provides a 'top-down' framework for understanding the neural computations underlying the motor circuit, which can potentially be used to guide further experiments to address details.

In Chapter 3, through a combined effort of optogenetic, microfluidic manipulation, and systematic quantification of behavior, I characterized a homeostatic mechanism underlying *C. elegans* locomotion modulation in response to external postural perturbation. Using reverse genetic analysis, Ca^{2+} imaging, and neural ablation, I reveal a complete neural circuit responsible for this curvature compensatory behavior. This circuit involves a dopamine and neuropeptide signaling pathway, orchestrated by a set of sensory neurons, interneurons, and motor neurons. My findings demonstrates a unique mechanism where proprioception can work with dopamine and neuropeptide signaling to mediate homeostatic control of locomotion.

FUTURE DIRECTIONS

Following the current findings, the next steps of the research program could be divided into two parts.

IDENTIFY AND CHARACTERIZE NEURAL ELEMENTS CONVEYING PROPRIOCEPTIVE FUNCTIONS IN LOCOMOTORY RHYTHM GENERATION

Highly quantitative and robust behavioral assays (as described by Methods in Chapter 2) will not only allow us to characterize and model the neuronal mechanisms that generate motor rhythms, but do enable us to identify critical neurons for regulating motor activities.

We will first identify and characterize the neural elements with the proprioceptive roles in locomotory rhythm generation proposed by Chapter 2. To do this task, we will conduct a behavioral screen on existing mutants to identify mutations that seem to be involved in the proprioceptive feedback detection and computation. Should any mutants show a defect in locomotory dynamics during our behavioral screening process, we will characterize their roles in the proprioception by confirming their site-of-action using cell-specific rescue and analyzing their effects on the Ca^{2+} response of specific neurons using Ca^{2+} imaging techniques.

EXPLORE THRESHOLD-BASED SWITCHING MECHANISM PROPOSED IN OUR MODEL

Next, we will explore the threshold-based switching mechanism proposed in our model. As indicated by the model, the proprioceptive feedback is defined as a linear combination of the total curvature of a body segment and its time derivative, and this factor is being compared with a pair of additive-inverse postural thresholds during muscle contractions. Therefore, two aspects concerning this hypothesis remain to be examined experimentally: (a) dependence of proprioceptive feedback on rate of change of curvature, (b) contributions of the threshold-based algorithms to the locomotory outputs.

To test the dependence of proprioceptive feedback on rate of change of curvature, I will examine Ca^{2+} dynamics in transgenic animals expressing the Ca^{2+} sensitive protein GCaMP selectively in motor neurons (head: SMD neurons, body: B-type neurons) of the motor circuit. We will place a worm in a pneumatic microfluidic device described in previous studies (Wen et al.,

2012) where a segment of the worm is trapped in a channel flanked by two chambers. By pressurizing the chamber on one side while depressurizing the one on the other side, we will be able to manipulate the curvature of body segment in a controllable manner. Using this microfluidic setup, we will first systematically measure the Ca^{2+} signal of the neuron at different curvature values with the channel shape fixed at each measurement, and then measure the Ca^{2+} dynamics of the neuron while changing the channel curvature at a certain constant rate. In the experiments, we need to measure these neuronal activities concurrently with the changing curvatures.

For exploring how the threshold-based algorithms contribute to locomotory outputs, we will use some kind of non-depolarizing agents (need to explore further) that bind to the NMJ receptor as antagonists and leave fewer receptors for acetylcholine to bind (Bowman, 2006). Thus, the decrease in binding of acetylcholine will lead motor neuron transmission to the muscle to be less likely to occur and we assume this physiological effect to be conceptually equivalent to directly increasing the threshold value. We will then change the dosage of the drug to accomplish a range of effect from being mild to fully paralyzing the locomotion. Our behavioral analyzing system is highly automated and quantitative, so it is possible to obtain ample data from moving animals that are affected under different levels of paralysis (different proprioceptive thresholds). The choice of paralyzing locations can be various, such as the whole body or a specific side of body, which allows us to explore the threshold-based mechanism in a flexible way.

APPENDIX A: SUPPLEMENTAL FIGURES

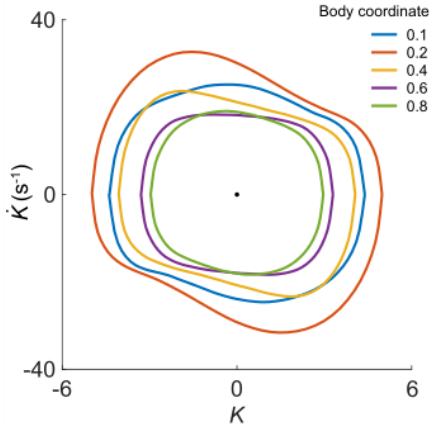

Figure S2.1. Phase portrait representations of the oscillatory bending dynamics for various body coordinates.

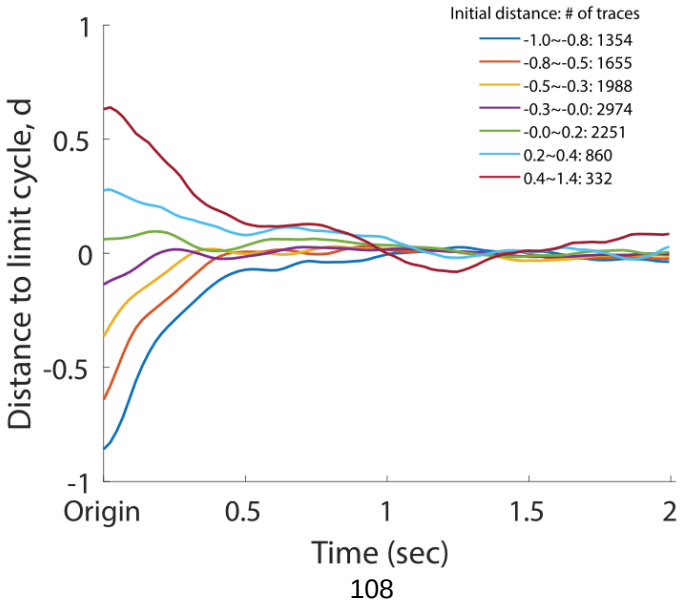

Figure S2.2. Normalized deviation to the normal cycle (the unperturbed oscillation) for the head oscillation of the perturbed worms.

Individual dynamics were grouped into different bins by binning their initial amplitude at $t = 0$. In the figure, each trace represents the collective amplitude dynamics of the corresponding group. Distance d is defined such that $d = -1$ at the origin and $d = 0$ on the limit cycle. The legend indicates initial amplitude range of each bin and the corresponding number of individual traces within the bin.

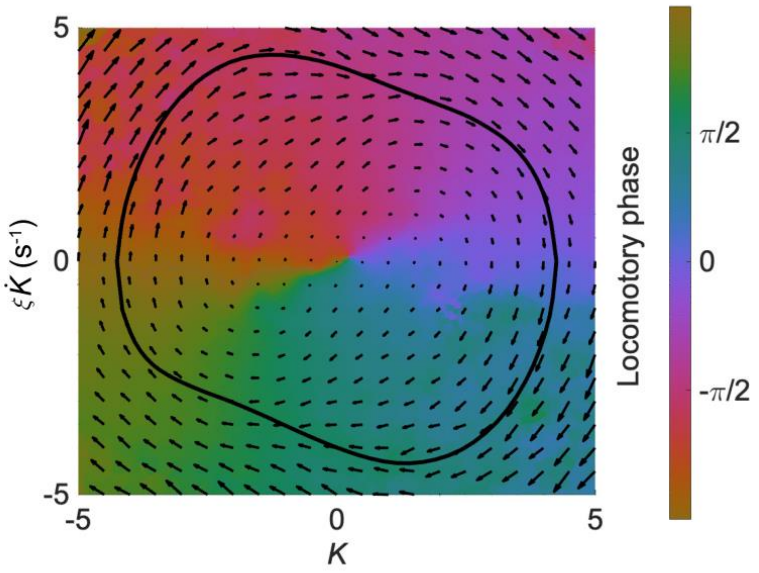

Figure S2.3. The isochron map overlaid with the vector field for the worm's head oscillation.

On the isochron map, a point on the normal cycle (black trajectory) and all other points off the normal cycle that share the same color form a manifold representing states having an equal

phase (indicated by the color bar). On the vector field, an arrow represents the phase state $(dK/dt, d(\xi\dot{K})/dt)$ which determines the time derivative of the state of a trajectory. Both maps were computed from the results of experiments with Pmyo-3::NpHR worms.

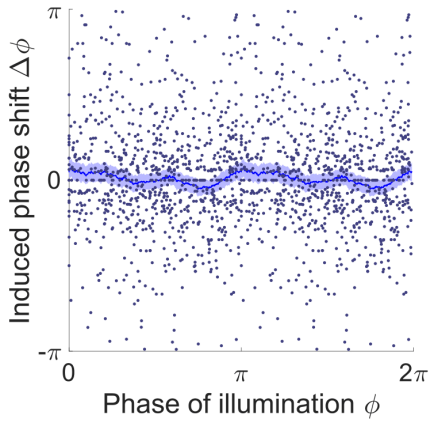

Figure S2.4. Phase response curve of Pmyo-3::NpHR worms (ATR- control group).

Each point represents a single illumination (0.1 s duration, 532 nm wavelength) of one worm. Filled area represents 95% confidence interval. Data collected from 414 trials using 116 worms.

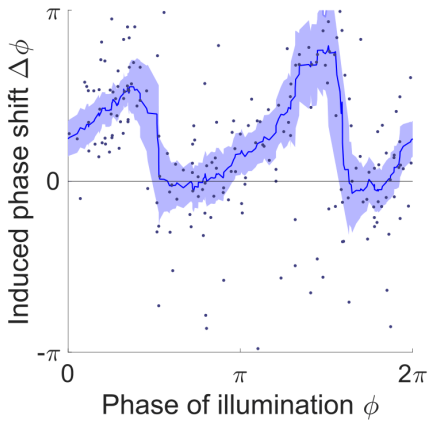

Figure S2.5. Phase response curve of Pmyo-3::NpHR worms perturbed by a 0.055 s optogenetic muscle inhibition.

Curve was obtained from 150 trials of transient inhibitions of head muscles using 115 worms. Each point represents a single illumination (0.055 s duration, 532 nm wavelength) of one worm. Filled area represents 95% confidence interval.

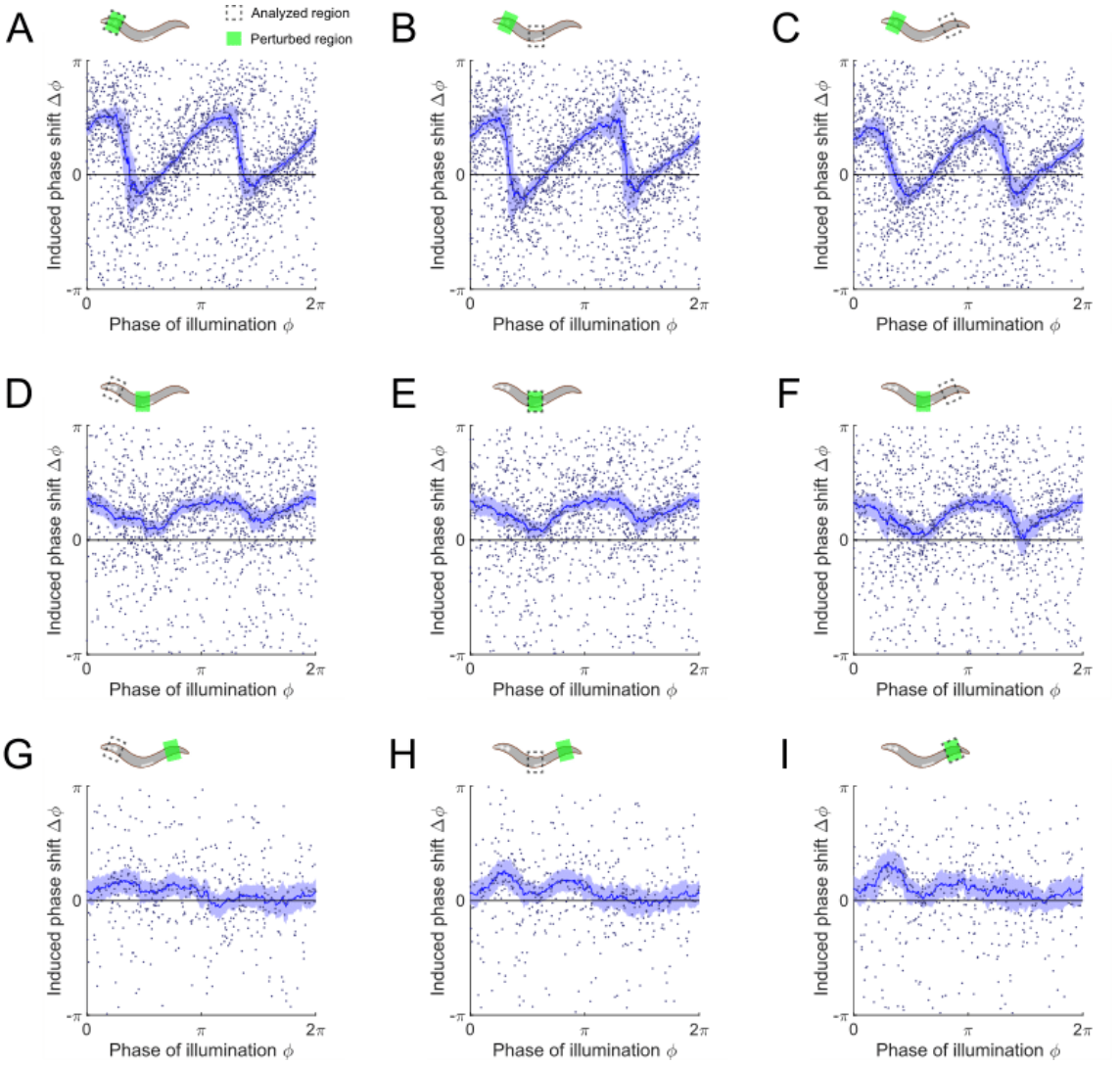

Figure S2.6. Phase response curves of *Pmyo-3::NpHR* worms induced by a 0.1 s optogenetic muscle inhibition, perturbed and measured at various body regions.

Anterior = 0.1-0.3; middle = 0.4-0.6; posterior = 0.6-0.8 along the worm body. (*Upper*) Schematics illustrating the selected spatial regions for optogenetic inhibition (Green shaded area) and phase response analysis (dashed box).

(A-C) PRCs induced by muscle inhibition of the anterior region (991 trials using 337 worms),

measured from (A) anterior, (B) middle, and (C) posterior regions, respectively.

(D-F) PRCs induced by muscle inhibition of the middle region (687 trials using 276 worms), measured from (D) anterior, (E) middle, and (F) posterior regions, respectively.

(G-I) PRCs induced by muscle inhibition of the posterior region (235 trials using 76 worms), measured from (G) anterior, (H) middle, and (I) posterior regions, respectively. For all the above plots, each point indicates a single illumination (0.1 s duration, 532 nm wavelength) of one worm. Experimental curves were obtained using a moving average along the *x*-axis with 0.16π in bin width. Filled area of each curve represents 95% confidence interval with respect to each bin of data points.

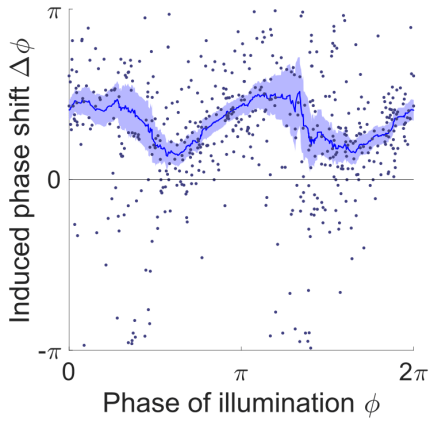

Figure S2.7. Phase response curve of transgenic worms that express NpHR in all cholinergic neurons.

Curve was obtained from 270 trials of transient inhibitions of cholinergic neurons in the head region using 135 worms. Each point represents a single illumination (0.055 s duration, 532 nm wavelength) of one worm. Filled area represents 95% confidence interval.

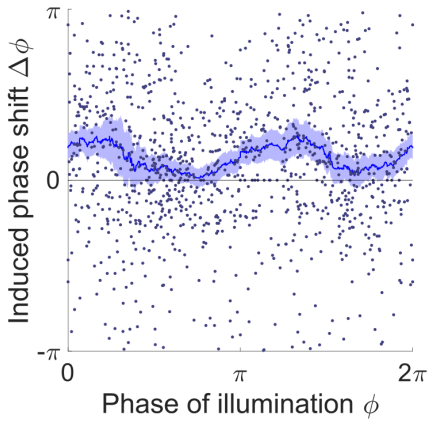

Figure S2.8. Phase response curve of transgenic worms that express Arch in the B-type motor neurons.

Curve was obtained from 551 trials of transient inhibitions of the B-type motor neurons in the head region using 160 worms. Each point represents a single illumination (0.055 s duration, 532 nm wavelength) of one worm. Filled area represents 95% confidence interval.

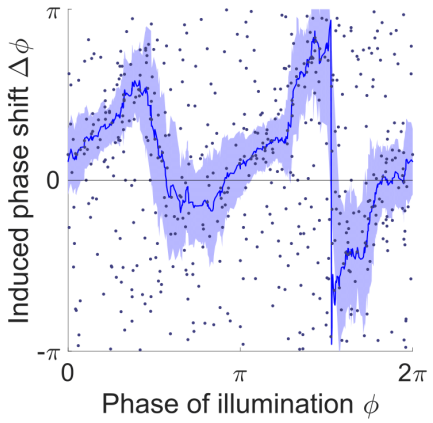

Figure S2.9. Phase response curve of transgenic worms that express NpHR in the body wall muscles but lack the GABA receptor for the D-type motor neurons.

Curve was obtained from 259 trials of transient inhibitions of head muscles using 192 worms. Each point represents a single illumination (0.1 s duration, 532 nm wavelength) of one worm. Filled area represents 95% confidence interval.

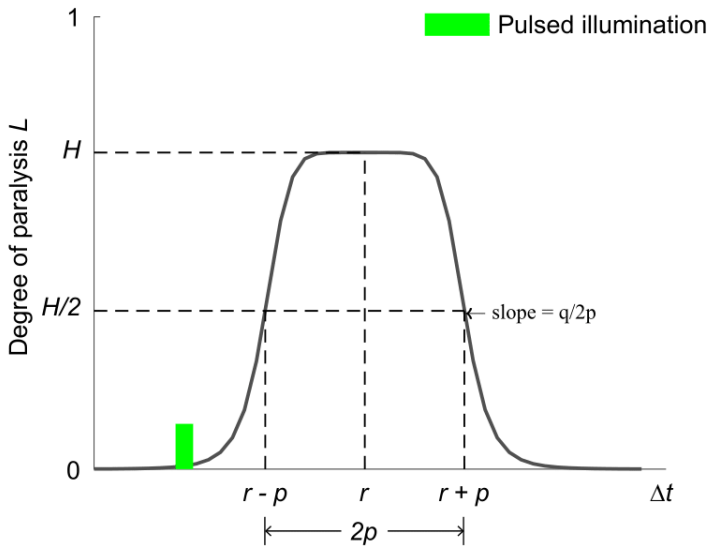

Figure S2.10. Bell-shaped function for modeling the optogenetic muscle inhibition (Eqn. S2.14).

The curve models the degree of paralysis due to the optogenetic muscle inhibition as a function of time. Referring to **Eqn. S2.14**, the fractional variable H describes the reduced proportion of muscle moment when a worm reaches maximal paralysis after illumination. r describes the time of maximal paralysis with respect to the occurrence of illumination. p determines the time interval during which the paralyzing degree exceeds $H/2$. q and p together determine the paralyzing rate.

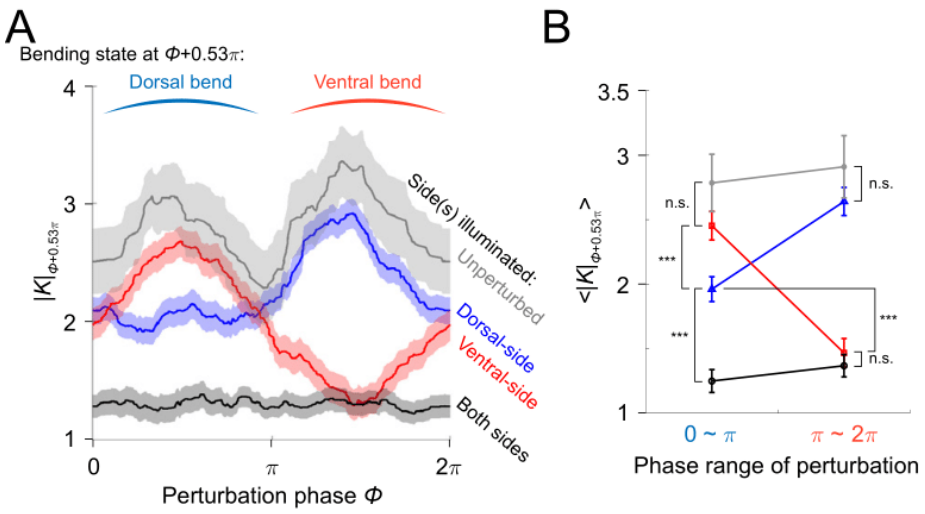

Figure S2.11. Paralyzing effect analysis of muscle inhibitions induced by illumination on different sides of the worm's head segment.

(A) Spectra of paralyzing effects across all phases of illuminations, represented by absolute curvature $|K|$ of the head region. $|K|$ shown on *y*-axis is the value obtained 0.53π later in phase (or 0.3 s in time with respect to locomotion period 1.13 s) after the illumination phase ϕ. The specific phase difference 0.53π (or 0.3 s time difference) was chosen for computing the paralyzing effects because a max paralysis is achieved at ~0.3 s after illumination as shown in **Fig. 3B**. Grey curve represents control ATR+ (no light) group (414 trials using 116 worms). Red curve represents ATR+ group with only ventral side being illuminated (373 trials using 176 worms). Blue curve represents ATR+ group with only dorsal side being illuminated (576 trials using 242 worms). Black curve represents ATR+ group with both sides being illuminated (991 trials using 337 worms). All curves were obtained using a moving average along the *x*-axis with 0.4π in bin width and filled areas represent 95% confidence interval.

(B) Average paralyzing effects during dorsal bend and ventral bend, represented by mean

absolute curvature $\langle|K|\rangle$ averaged across range $[0,\pi]$ (dorsal bend) and $[\pi,2\pi]$ (ventral bend), respectively. Colors indicate different conditions of experiment in the same way as in A. (***) indicates p<0.0005 using paired *t* test.

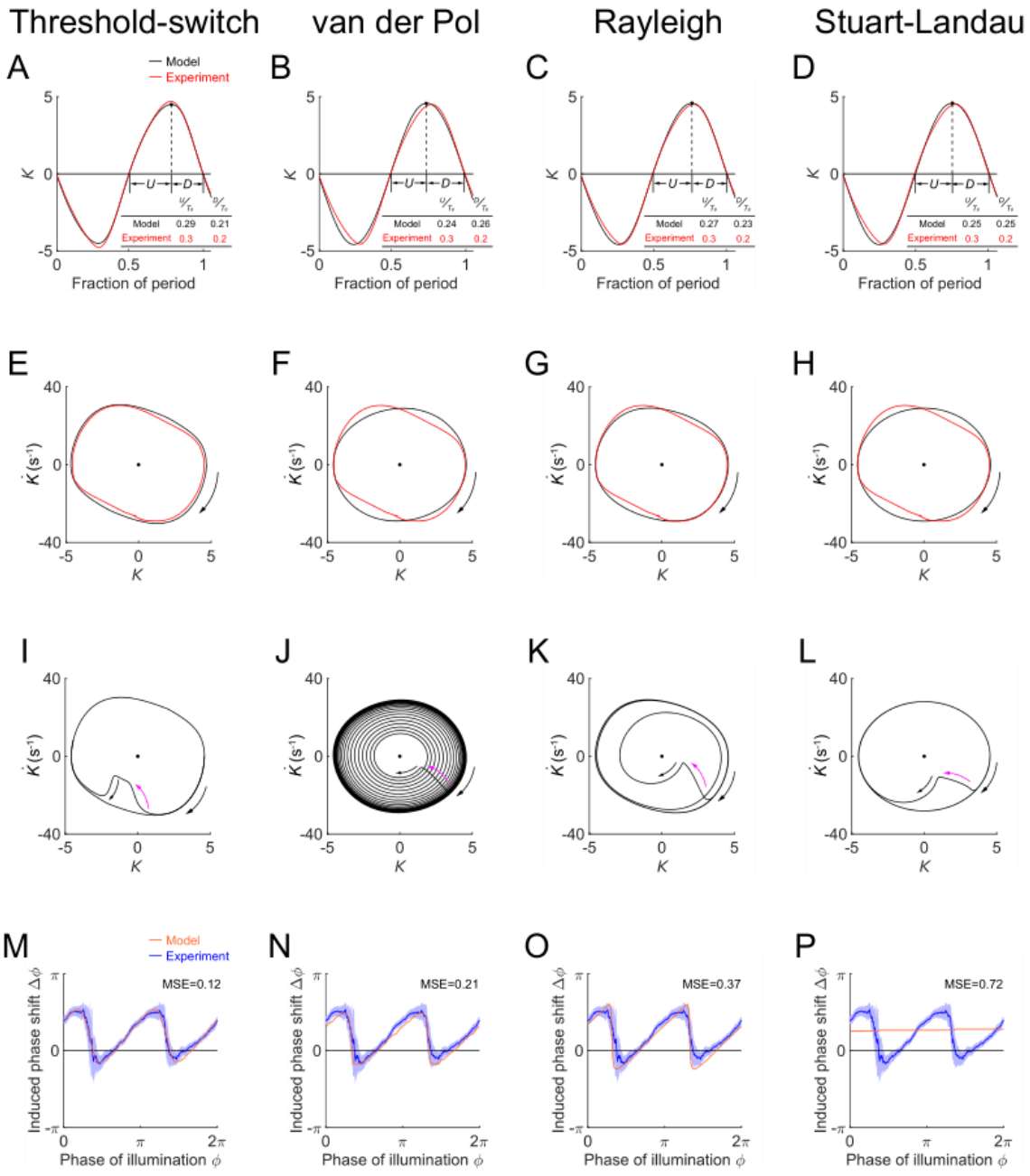

Figure S2.12. Performance of model oscillators: threshold-switch (column 1), van der Pol (column 2), Rayleigh (column 3), and Stuart-Landau (column 4).

(A-D) Time-varying curvatures of the worm's head region, measured from experiments (red, 5047

cycles using 116 worms) or produced by models (black). The four models match the experimental curvature with MSEs ≈ 0.18, 0.44, 0.18, and 0.39, respectively. (Inset table) U/T_0 (fraction of period of bending toward the ventral or dorsal directions) and D/T_0 (fraction of period of straightening toward a straight posture), for experiments or models, respectively.

(E-H) Phase portrait graphs of the free-running dynamics shown in A-D, measured from experiments (red) or produced by models (black). Arrow indicates clockwise movement.

(I-L) Phase plots showing each model perturbed (indicated by purple arrow) away from the stable limit cycle and then recovering toward the equilibrium.

(M-P) Phase response curves with respect to both-side muscle inhibition, measured from experiments (blue, 991 trials using 337 worms) or produced by models (orange). Four models match the experimental PRC with MSE ≈ 0.12, 0.21, 0.37, and 0.72, respectively.

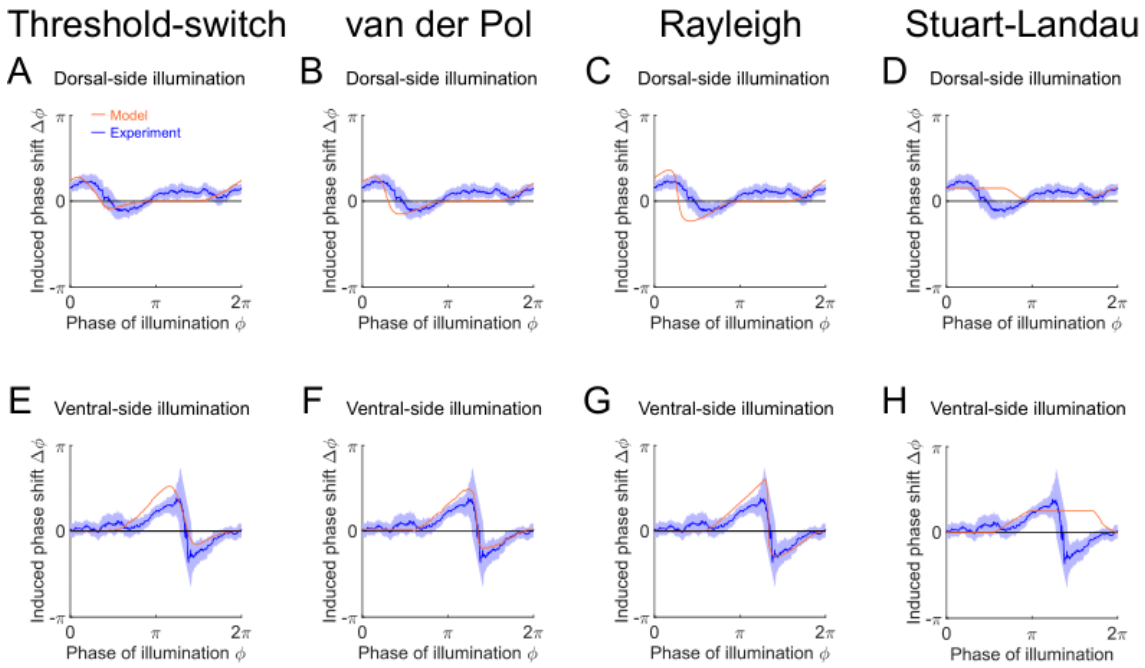

Figure S2.13. Phase response curves with respect to single-side muscle inhibition, simulated from model oscillators: threshold-switch (column 1), van der Pol (column 2), Rayleigh (column 3), and Stuart-Landau (column 4).

(A-D) PRCs with respect to dorsal-side muscle inhibition, measured from experiments (blue, 576 trials using 242 worms) or produced by models (orange).

(E-H) PRCs with respect to ventral-side muscle inhibition, measured from experiments (blue, 373 trials using 176 worms) or produced by models (orange). Filled areas of all experimental PRCs represent 95% confidence interval with respect to each bin of data points.

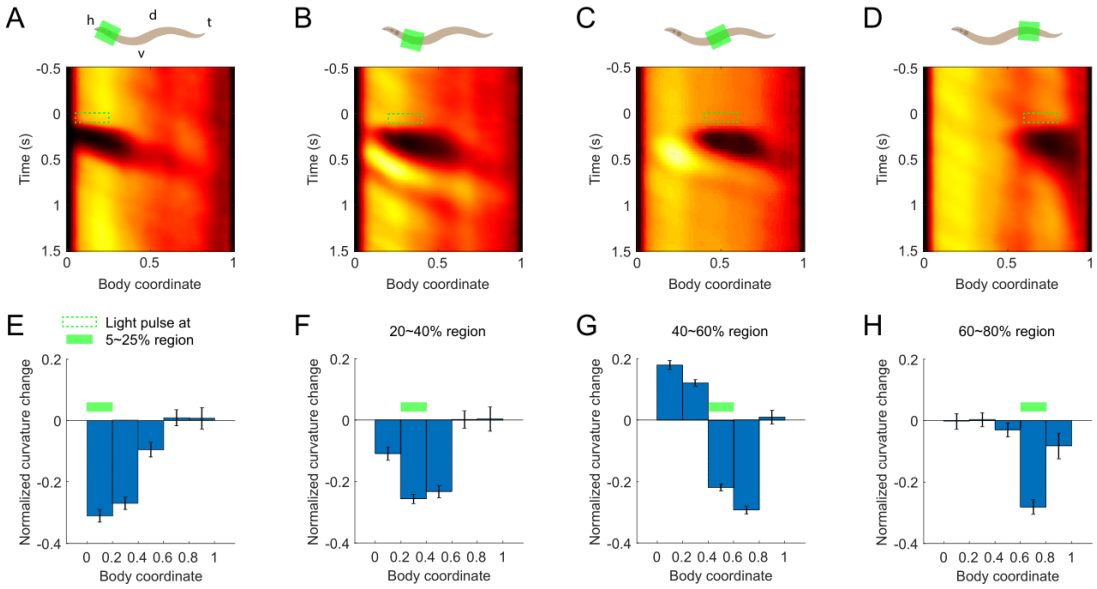

Figure S3.1. Curvature modulation in response to optogenetic muscle inhibition at various body regions.

(A-D) (*Upper*) Schematics denoting optogenetic muscle inhibition applied at head (A), neck (B), middle (C), posterior (D) regions of transgenic worms expressing Muscle::NpHR. h = head, t =

tail, d = dorsal, v = ventral. Regarding worm body regions, head = 0.05-0.25, neck = 0.2-0.4, middle = 0.4-0.6, posterior = 0.6-0.8 body coordinate. (*Lower*) Kymographs of mean absolute curvature around the 0.1 s inhibitions (green dashed box) in the indicated regions of worm body as shown by the corresponding schematics. 649 trials from 135 worms were used in (A); 466 trials from 75 worms were used in (B); 1160 trials from 206 worms were used in (C); 467 trials from 76 worms were used in (D).

(E-H) Undulatory amplitude change upon the transient optogenetic muscle inhibitions applied at indicated body regions, measured as mean ± SEM of the normalized curvature change of the first post-illumination curvature peak of various body regions from the head to the tail. Green bar indicates the 0.1 s laser illumination applied to the corresponding body region. Data for computing (E-H) were the same as used in (A-D), respectively.

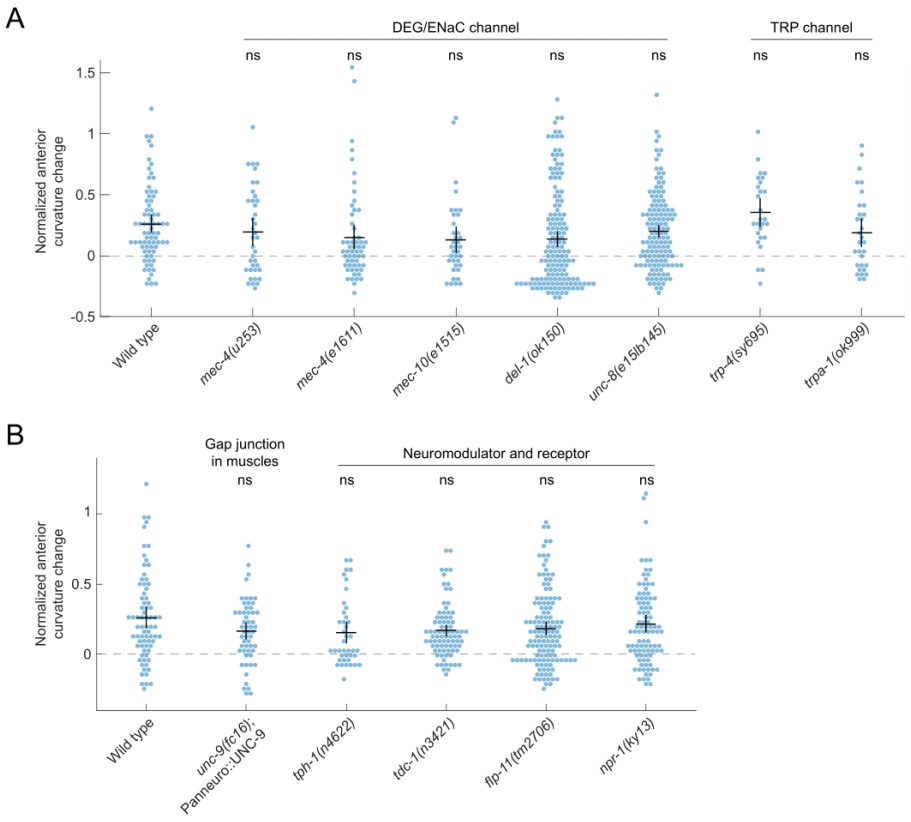

Figure S3.2. Several mechanoreceptors, neuromodulators and receptors, and muscles are not required for curvature compensatory response.

(A) Several mechanoreceptors do not contribute to curvature compensation. Data represent mean ± SEM of the normalized anterior curvature change in response to midbody constraint for DEG/ENaC channel mutants *mec-4(u253)*, *mec-4(e1611)*, *mec-10(e1515)*, *del-1(ok150)*, *unc-8(e15lb145)*, and TRP channel mutants *trp-4(sy695)*, *trpa-1(ok999)*. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each strain. ns: not significant when compared with wild type animals, Dunnett's multiple comparison tests.

(B) Signaling of several neuromodulators and gap junctions between muscle cells do not contribute to curvature compensation. Data represent mean ± SEM of the normalized anterior

curvature change in response to midbody constraint for *tph-1(n4622), tdc-1(n3421), npr-1(ky13)* mutants, and *unc-9(fc16)* mutants in which the UNC-9 innexin protein was rescued pan-neuronally. TPH-1 encodes tryptophan hydroxylase which is required for serotonin synthesis. TDC-1 encodes tyrosine decarboxylase which is required for synthesizing tyramine or octopamine. The *unc-9* rescued strain lacks gap junction only in muscle cells. Each data point is the mean of a 3 s period of constrained locomotion pooled across 10 or more animals for each strain. ns: not significant when compared with wild type animals, Dunnett's multiple tests.

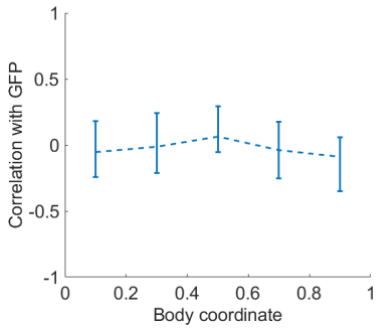

Figure S3.3. Average correlations of PDE::GFP signal with body curvature (N = 15).

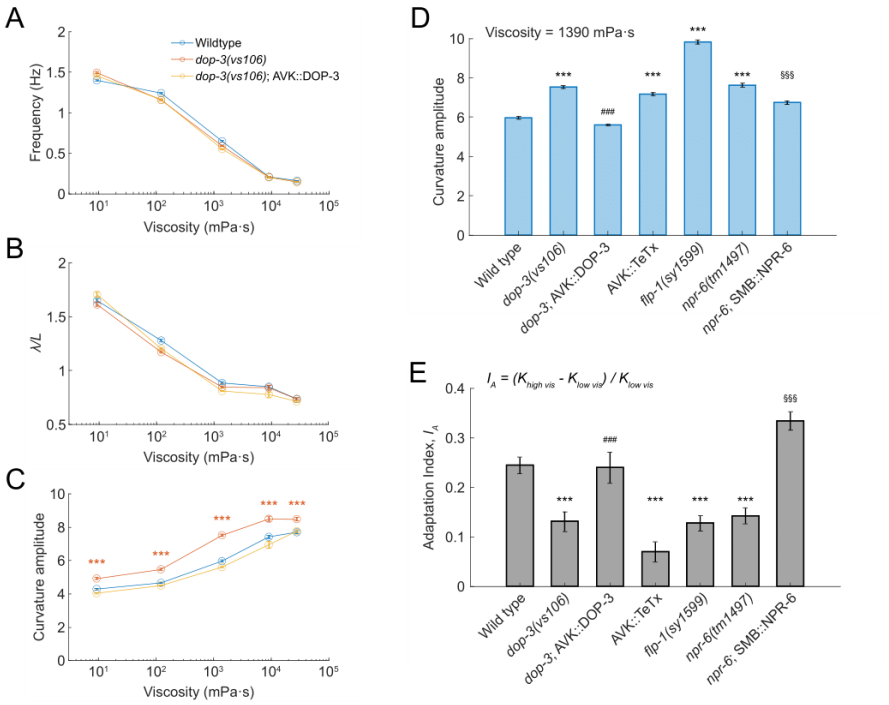

Figure S3.4. Curvature compensation mechanism is consistent with gait adaptation of bending amplitude in response to mechanical load.

(A-C) Locomotory parameters in different viscous solutions, measured from wild-type animals, *dop-3(vs106)* mutants, and *dop-3* mutants with *dop-3* function rescued in AVK neurons. (A) Mean undulatory frequency; (B) mean undulatory wavelength scaled by worm body length *L*; (C) mean curvature amplitude of undulation, **p<0.01, ***p<0.001, ns: not significant compared with wild type, Tukey-Kramer multiple comparison tests.

(D) Mean curvature amplitude of undulation in solutions of viscosity 1390 mPa·s, measured from the main strains used for studying curvature compensation. ***p<0.001 when compared with wild type, $^{###}$p<0.001 when compared with *dop-3* mutants, §§§p<0.001 when compared with *npr-6* mutants, Tukey-Kramer multiple comparison tests.

(E) An adaptation index was computed for each strain using the difference between the curvature

amplitudes in high (9079 mPa·s) and low (1390 mPa·s) viscous solutions divided by the curvature amplitude in the low viscous solution. ***p<0.001 when compared *dop-3* mutants with wild-type animals. Under each viscosity condition, 10 or more animals were tested for each strain.

APPENDIX B: SOFTWARE CODE

This appendix contains all the software code for data analyses described in Chapter 3. The original data and software files (along with compilation instructions) for Chapter 2 are available from an online repository (Ji et al., Dryad 2021: https://doi.org/10.5061/dryad.wwpzgmsk2).

The software for computing phase response curves (Chapter 2) was originally written by Dr. Anthony Fouad and revised by me. The Python scripts for analyzing worm posture during Ca^{2+} imaging (Chapter 3) were written by Dr. Andrew Ruba. The rest of software (Chapters 2 and 3) was entirely written by me.

[In order to minimize number of pages, materials are displayed in two columns with the font size modulated]

AnalyzeTool_Poolmean.m

```
% Worm shape analysis. This version is modified for
analyzing data obtained
% from worm locomotion on WormTunnel microfluidic chambers.
%
% First, periods during which a worm is undulating without
constraint will be
% collected and analyzed to generate a limit cycle for
normal undulatory
% dynamics (pool average).
%
%
% Next, periods during which a worm is moving under
constraint (usually at
% the middle of body) will be collected and analyzed to
generate phase
% dynamics during those periods
% Finally, combining the analyzed results from the above
two steps, a
% generalized compensatory factor kymogram will be
generated in a 2D
% heatmap form, which represents a function of time and
body coordinate.
% Also, information of the body portion that is being
constrained will be
% reflected on the kymogram.
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% START
MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
clc; close all; clear

global start_illum end_illum prefix pathname filename
global conc fps pix_per_mm wormthreshold isie decim filsize
spline_p initials

wormlabel  = 1;
pix_per_mm = 198.83;
prefix;
pathname;

curvlim = 0.2;
domovie = 1;
issavefiles = 1;
option1 = 'Yes (AVI only)';
option2 = 'Yes (MAT)';

button = length(questdlg('Load new
data?','',option1,option2,'No', option1));

if button == length(option2)
    disp('options2');
    [filename,pathname2] = uigetfile({'*.mat'});
    load([pathname2 filename]);
elseif button == length(option1)
    disp('option1');
    do_dialog = 1;
    %% Identify and analyze control periods
    if do_dialog
        try
            cd(pathname);
        catch
            pathname = pwd;
        end

        [filename,pathname]  = uigetfile('*.avi', 'Select
File');
%         MATfname = ['All-' 'imm Normal.mat'];
        MATfname = ['All-' filename(1:6) 'imm Normal.mat'];
        load(fullfile(pathname, MATfname));
    end
    %% Identify and analyze constrained periods
    if do_dialog
        vidObj = VideoReader(fullfile(pathname,filename));
        NumFrames = vidObj.NumFrames;
        FPS       = vidObj.FrameRate;
        NumFrames = vidObj.NumFrames;
        fps       = vidObj.FrameRate;
        if isempty(conc)
            conc = 0;
        end
        if isempty(wormlabel)
            wormlabel = 1;
        end
        if isempty(pix_per_mm)
            pix_per_mm = 1;
        end
        if isempty(wormthreshold)
            wormthreshold = 0.10;
        end
        if isempty(isie)
            isie = [1, NumFrames];
        end
```

```
        if isempty(decim)
            decim = 1;
        end
        if isempty(filsize)
            filsize = 0.2;
        end
        if isempty(start_illum)
            start_illum = 1;
        end
        if isempty(end_illum)
            end_illum = 1;
        end
        if isempty(spline_p)
            spline_p = 0.01;
        end
        if isempty(initials)
            initials = {'JHF'};
        end
        if isempty(initials)
            initials = {'JHF'};
        end
        fields={'conc','wormlabel', 'fps','pixels per
mm','Worm image threshold',...
            'istart/iend (use";"if multiple)', 'Decimation
(1=none))',...
            'Filter size / diameter',
'start_illum','end_illum', ...
            'spline fit parameter', 'Make movie?', 'Your
initials', 'Save files?'};
        if exist('isie', 'var')
            answer = inputdlg(fields, 'Cancel to clear
previous', 1, ...

{num2str(conc),num2str(wormlabel),num2str(fps),num2str(pix_
per_mm),num2str(wormthreshold),...
                mat2str(isie), num2str(decim),...

num2str(filsize),num2str(start_illum),num2str(end_illum), .
..
                num2str(spline_p), num2str(domovie),
initials{1}, num2str(issavefiles)});
        else
            answer = inputdlg(fields, '', 1);
        end

        if isempty(answer)
            pause;
        end

        conc = str2double(answer{1});
        wormlabel = str2double(answer{2});
        fps = str2double(answer{3});
        pix_per_mm = str2double(answer{4});
        wormthreshold = str2double(answer{5});
        isie = Str2Mat(answer{6});
        decim = str2double(answer{7});
        filsize = str2double(answer{8});
        start_illum = str2double(answer{9});
        end_illum = str2double(answer{10});
        spline_p = str2double(answer{11});
        domovie = str2double(answer{12});
        initials = answer(13);
        issavefiles = str2double(answer{14});
    end

    nperiods    = size(isie, 1);
    curv_const  = cell(nperiods, 1);
    angle_const = cell(nperiods, 1);
    len_const   = cell(nperiods, 1);
    rgn_const   = cell(nperiods, 1);
    fullnewdirname = cell(nperiods, 1);
    w_diam      = cell(nperiods, 1);
    %  Compute undulatory variables for constrained groups
    for kk = 1 : nperiods
        do_const = 1;
        thisperiod = isie(kk, :);
        options = {conc, wormlabel, fps, pix_per_mm,
wormthreshold,...
                   thisperiod, decim, filsize, start_illum,
end_illum,...
                   spline_p, domovie, initials, pathname,
filename, do_const, issavefiles};
        [curv_const{kk}, ~, angle_const{kk}, len_const{kk},
rgn_const{kk}, fullnewdirname{kk}, w_diam{kk}]...
            = WORMSHAPE_MAINCALCULATION(vidObj, options);
    end
    close all
end
%% Calculate the average normal phase plot from control
groups
[Kc_all, dKdtc_all, Kp_data, dKdtp_data, T0_avg]...
    = Generalized_cfactor_for_microfluidics(CURV_all,
curv_const, fps);
% Determining control variables for normalizing phase
states at each time
% point and body coordinate
numtrials = numel(Kp_data);
numsamplepts = 100;
numcurvpts   = 100;
a  = .15; c = a * T0_avg;
Zc = Kc_all + 1i*c*dKdtc_all;
Pc = unwrap(angle(Zc), [], 2);
[~, Sc] = meshgrid(1:numsamplepts, 1:numcurvpts);
% Generate interpolant (in a bulk manner)
FR = scatteredInterpolant(Pc(:), Sc(:), Zc(:), 'linear',
'nearest');
% Constrcting complex curvature dynamics for pulsed group
CR = cell(numtrials,1);
for i = 1 : numtrials
    Kp     = Kp_data{i};
    dKdtp  = dKdtp_data{i};
    Zp     = Kp + 1i*c*dKdtp;
    Pp_ori = angle(Zp); % do not use unwrap
    Ppld   = Pp_ori(:);
    Ppld(Ppld>0) = Ppld(Ppld>0) - 2*pi;
    Pp = reshape(Ppld, size(Pp_ori));
    [~, Sp] = meshgrid(1:size(Pp,2), 1:size(Pp,1));
    Rp = abs(Zp);
    Zc4pld = FR(Pp(:), Sp(:));
    Zc4p = reshape(Zc4pld, size(Pp));
    Rc4p = abs(Zc4p);
    CR{i}   = (Rp) ./ Rc4p;
end
%%  Plotting data
issave = 1;
path4save = pathname;
fname  = strrep(filename, '_bkgsubtracted.avi', '');
%% 3-D phase portrait plot of normal undulation
figure(1); clf
numsamplepts = 100;
numcurvpts   = 100;
curvrgn_analysis = 1 : numcurvpts;
TX_mesh = meshgrid(1 : numsamplepts, curvrgn_analysis);
hold on
plot3(Kc_all', dKdtc_all', TX_mesh)  % isosegmental line
plot3(Kc_all,  dKdtc_all,  TX_mesh') % isophasic line
hold off
view([30,30])
xl = xlim;
yl = ylim;
zlim([5 95])
xlabel('K')
ylabel('dKdt')
zlabel('Body coordinate')
set(gca, 'FontSize', 12, 'Position',
[0.13,0.11,0.775,0.815])

%% GUI with interactive response-plot updates for phase
portrait plots
s  = 30;
f  = figure(2); clf
ax = axes('Parent',f,'position',[0.2 0.25  0.65 0.65],
'PlotBoxAspectRatio', [1,0.81,0.75]);
faseplot2  = @(ax, s) phasePlot2(curvrgn_analysis, Kc_all',
dKdtc_all', s, ax, xl, yl);
faseplot2(ax, s);
b =
uicontrol('Parent',f,'Style','slider','Position',[81,34,419
,23],...
                'value',s, 'min',5, 'max',95);
bgcolor = f.Color;
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[45,37,30,20],...
                'String','Head','BackgroundColor',bgcolor);
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[505,37,30,20],...
                'String','Tail','BackgroundColor',bgcolor);
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[240,10,100,23],...
                'String','Body
coordinate','BackgroundColor',bgcolor);

b.Callback = @(es,ed) faseplot2(ax, es.Value);

%% Cfactor as a function of time and body coordinate
(averaged over trials)
% customCMap = hsvCustomCMap(CR{1});
for i = 1 : numtrials
    thisCR   = CR{i};
    thiscurv = curv_const{i};
    thisrgn  = rgn_const{i};
    t = (0 : size(thisCR, 2)-1)/fps;
    figure(2+2*i-1);clf;
    set(gcf, 'Position',[118,366,542,506])
    imagesc(curvrgn_analysis, t, thisCR')
    hold on
```

```
        B = bwboundaries(thisrgn, 'noholes');
        for jj = 1 : numel(B)
            B1 = B{jj};
            patch(B1(:,2), B1(:,1)/fps, 'red',
'FaceColor','none', 'EdgeColor','r',
'LineStyle',':','LineWidth',2)
        end
    hold off
    xlabel('Body coordinate (0=head)')
    ylabel('Time (sec)')
    set(gca,'FontSize',20)
    colormap('jet')
    cb = colorbar('AxisLocation','out');
    ylabel(cb, 'Cfactor')
    if issave
        saveas(gcf, [fullfile(path4save, fname) '_cfactor x
t' sprintf('%d',i) '.fig'])
        saveas(gcf, [fullfile(path4save, fname) '_cfactor x
t' sprintf('%d',i) '.png'])
    end
    figure(2+2*i);clf;
    set(gcf, 'Position',[118,366,500,500])
    imagesc(curvrgn_analysis, t, thiscurv)
    hold on
        B = bwboundaries(thisrgn, 'noholes');
        for jj = 1 : numel(B)
            B1 = B{jj};
            patch(B1(:,2), B1(:,1)/fps, 'red',
'FaceColor','none', 'EdgeColor','r',
'LineStyle',':','LineWidth',2)
        end
    hold off
    xlabel('Body coordinate (0=head)')
    ylabel('Time (sec)')
    set(gca,'FontSize',20)
    colormap(cmap_redblue(.7))
    cb = colorbar('AxisLocation','out');
    ylabel(cb, 'Curvature')
    if issave
        saveas(gcf, [fullfile(path4save, fname) '_curv x t'
sprintf('%d',i) '.fig'])
        saveas(gcf, [fullfile(path4save, fname) '_curv x t'
sprintf('%d',i) '.png'])
    end
end
if issave
        save([fullfile(path4save, fname) '_data.mat'],
'len_const', 'rgn_const', 'Kp_data', 'dKdtp_data','CR',
'w_diam', 'fps')
end
```

Anterior_K_analysis.m

```
N = numel(CURV_all);
t0 = 3;
T0 = t0 * fps;
curv_rgn = 10:30;
Kmax = [];
for i = 1 : N
    curv = CURV_all{i};
    v = mean(curv(:,curv_rgn),2);
    T = numel(v);
    num_subperiod = floor(T/T0);
    for j = 1 : num_subperiod
        win = (1 + (j-1)*T0):(j * T0);
        kmax = max(abs(v(win)));
        Kmax = cat(1, Kmax, kmax);
    end
end

K = mean(Kmax);
K_SEM = std(Kmax)/sqrt(N);
```

Background_subtraction.m

```
% p = {
%     '/Volumes/HF_BACKUP/Compensatory experiment-
Microfluidic/N2_70um'
%     };
% nformat = '*.avi';
% %% process in group
% for i = 1:numel(p)
%     listing = dir(fullfile(p{i}, nformat));
%     numvideo = numel(listing);
%     for j = 1 : numvideo
%         fprintf('Processing folder %d of %d, video %d
of %d\n', i, numel(p), j, numvideo)
%         BkgSubtraction_Output(listing(j).name, p{i});
%     end
% end

%% process individual
p = {
    'C:\Users\fffei\Desktop\data to be transferred to
Main drive\FQ2747_60um_3-27-2022';...
    };

for n = 1 : numel(p)
    fprintf('Exps %d out of %d\n', n, numel(p))
    Dir = dir(p{n});
    for ii = 1 : numel(Dir)
        fname = Dir(ii).name;
        if fname(1) == '.'
            continue
        end
        BkgSubtraction_Output(fname, p{n}, 'd');
    end
end
```

BkgSubtraction_Output.m

```
function BkgSubtraction_Output(filename, pathname,
fieldmode)
% Load video
newfilename =
[erase(filename,'.avi'),'_bkgsubtracted','.avi'];

vidObj = VideoReader(fullfile(pathname,filename));
vidWidth = vidObj.Width;
vidHeight = vidObj.Height;
numFrames = ceil(vidObj.Duration * vidObj.FrameRate);

k = 1;
% Generate the background
numSamples = 1800;
while k <= numSamples
    if ~hasFrame(vidObj)
        break
    end
    currentFrame = double(readFrame(vidObj));
    if k == 1
        background = currentFrame;
    else
        %         fprintf('%d\n',k)
        %         background = ((k-1)*background +
currentFrame)/k;
        if fieldmode == 'd' % if dark-field
            background = min(background, currentFrame);
        elseif fieldmode == 'b' % if bright-field
            background = max(background, currentFrame);
        end
    end
    k = k + 1;
end
if vidObj.VideoFormat(1) == 'R'
    background = uint8(mean(background, 3));
else
    background = uint8(background);
end
vidObj.CurrentTime = 0;
bkgname = fullfile(pathname,[erase(filename,'.avi')
'_background.bmp']);
imwrite(background, bkgname);
% imgbkg = imread(bkgname);
% figure(1); clf
% image(imgbkg); colormap('gray')
% set(gcf, 'Position', [1892,170,1327,1027])
% answer = length(questdlg('Need to modify background
image?', 'BKG modification', 'Yes', 'No', 'No'));
% if answer == 2
%     close;
% elseif answer == 3
%     hold on;
%     title('Indicate ROI to be modified');
%     [bw1, xi1, yi1] = roipoly;
%     patch(xi1, yi1, 'g', 'FaceColor',
'none','EdgeColor','g', 'LineStyle',':')
%     title('Indicate ROI as substitute');
%     [bw2, xi2, yi2] = roipoly;
%     patch(xi2, yi2, 'g', 'FaceColor',
'none','EdgeColor','r', 'LineStyle',':')
%     % replace the pixel colors of ROI1 with the average
color from ROI2
%     tmp = uint8(mean(imgbkg(bw2),'all'));
%     background(bw1) = tmp;
%     % rewrite the modified image and then save it
%     imwrite(background, bkgname);
%     close;
% end
% Background subtraction and create a new video
k = 1;
v = VideoWriter(fullfile(pathname,newfilename),'Grayscale
AVI');
v.FrameRate = vidObj.FrameRate;
open(v)
tic
while hasFrame(vidObj)
    currentFrame = readFrame(vidObj);
    if vidObj.VideoFormat(1) == 'R'
```

```
        currentFrame = uint8(mean(currentFrame, 3));
    end
    if fieldmode == 'd' % if dark-field
        newFrame = currentFrame - background;
    elseif fieldmode == 'b' % if bright-field
        newFrame = background - currentFrame;
    end
    writeVideo(v,newFrame);
    k = k + 1;
    if k == floor(0.33 * numFrames)
        tmp = toc;
        tmp = round(tmp/60);
        fprintf('33%% complete, %d minutes passed\n',tmp)
    elseif k == floor(0.67 * numFrames)
        tmp = toc;
        tmp = round(tmp/60);
        fprintf('66%% complete, %d minutes passed\n',tmp)
    end
end
tmp = toc;
tmp = round(tmp/60);
fprintf('100%% complete, %d minutes passed\n',tmp)
close(v);

end
```

Do_collect_strains_info_beeswarm.m

```
clear; close all; clc
Parentdir = 'F:\Compensatory experiments\Combined_data';
P_all = {
        'Combined_N2',     'N2';...
%          'Combined_N2_50mM DA', 'N2+DA bath';...
%          'Combined_CB1112', 'cat-2';...
%          'Combined_CB1112_DAADAB', 'cat-2+DA bath';...
%          'Combined_PVD-ChR2_light-off',  'dop-1';...
%          'Combined_LX636',  'dop-1';...
%          'Combined_LX702',  'dop-2';...
%          'Combined_LX703',  'dop-3';...
%          'Combined_LX703_NOADAB', 'dop-3+DA bath';...
%          'Combined_FG58',   'dop-4';...
%          'Combined_LX706',  'dop-1/2';...
%          'Combined_LX705',  'dop-1/3';...
%          'Combined_CB156',  'dop-2/3';...
%          'Combined_RB1657',  'dop-1/2/3';...
%          'Combined_RM2702', 'dat-1';...
%          'Combined_CB1112_NOADAB', 'cat-2(single DA)';...
%          'Combined_LX703_unc-47_rescue', 'dop-3;Punc-
47::dop-3(+)';...
%          'Combined_LX703_ser-2_rescue', 'dop-3;PVD::dop-
3(+)';...
%          'Combined_LX703_myo-3_rescue', 'dop-
3;Muscle::dop-3(+)'
%          'Combined_LX703_acr-2_rescue', 'dop-3;Pacr-
2::dop-3(+)';...
%          'Combined_LX703_acr-5_rescue', 'dop-3;Pacr-
5::dop-3(+)';...
%          'Combined_ZX2201', 'dop-3;AVK::dop-3(+)';...
%          'Combined_KP2018'  'egl-21';...
%          'Combined_MT1219'  'egl-3';...
%          'Combined_CB156',  'unc-25';...
%          'Combined_CB382',  'unc-49';...
%          'Combined_CB678',  'lon-2';...
%          'Combined_KHK641', 'trp-1/2';...
%          'Combined_MT6308', 'eat-4';...
%          'Combined_RB1657', 'hpo-30';...
%          'Combined_NC279',  'Mock-ablated';...
%          'Combined_VM6365', 'ADE+CEP-ablated'
%          'Combined_PDE_killed', 'PDE-ablated';...
%          'Combined_LX645', 'PDE-ablated in DA';...
%          'Combined_PDE-TeTx', 'Pdat-1::TeTx';...
%          'Combined_PDE-TeTx_DA', 'Pdat-1::TeTx in DA';...
%          'Combined_TV19861','dma-1;PVD::dma-1';...
%          'Combined_PVD-HisCl_soak30min_15mM',
'PVD::HisCl1(+) medium';...
%          'Combined_PVD-HisCl_soak30min_20mM',
'PVD::HisCl1(+) high';...
%          'Combined_ZX819_off', 'PVD::ChR2(+) light-';...
%          'Combined_ZX819_on', 'PVD::ChR2(+) light+';...
%          'Combined_JT734','goa-1';...
%          'Combined_KP1087','dgk-1';...
%          'Combined_TQ296', 'egl-10';...
%          'Combined_MT1093', 'gpb-2';...
%          'Combined_ZX819_off', 'egl-30';...
%          'Combined_ZX819_on','egl-8';...
%          'Combined_VM6365_2','egl-16';...
         'Combined_CB169', 'unc-31';...
         'Combined_NY16', 'flp-1(yn4)';...
         'Combined_FQ2747', 'flp-1(sy1599)';...
%           'Combined_AVK mock-ablation', 'Mock
ablation';...
%           'Combined_AVK-ablation', 'AVK-ablated';...
%           'Combined_FQ2747', 'AVK::TeTx';...
         'Combined_RB1657', 'npr-6';...
         'Combined_ZX2037', 'npr-6;SMB::npr-6(+)';...
         'Combined_ZX3058', 'SMB::ICE';...
%           'Combined_YX139',  'unc-9 (rescued in
neurons)';...
%           'Combined_MT14984', 'tph-1';...
%           'Combined_MT13113', 'tdc-1';...
%           'Combined_HBR507', 'flp-11';...
%           'Combined_CX4148', 'npr-1';...
%           'Combined_TU253',  'mec-4';...
%           'Combined_CB1611', 'mec-4(d)';...
%           'Combined_CB1515', 'mec-10';...
%           'Combined_CB1338', 'mec-3';...
%           'Combined_PVD-HisCl_soak30min_15mM',  'del-
1';...
%           'Combined_PVD-HisCl_soak10min_10mM',  'unc-
8';...
%           'Combined_TQ296',  'trp-4';...
%           'Combined_RB1052', 'trpa-1';...

        };
N_strains = size(P_all,1);
Genonames = cell(1, N_strains);
Y = cell(1, N_strains);
Ya = cell(1, N_strains);
Ym = cell(1, N_strains);
Yp = cell(1, N_strains);
F_bar = zeros(size(Y));
X_bar = 1 : N_strains;
F_err = zeros(size(Y));
H2one = zeros(size(Y));
Pvals2one = zeros(size(Y));
H2wt = zeros(size(Y));
Pvals2wt = zeros(size(Y));
for i = 1:N_strains
    fprintf('Strain %d out of %d', i, N_strains)
    Genonames{i} = P_all{i,2};
    pathname = fullfile(Parentdir, P_all{i,1});
    % all subfolders
    subfolderlist = dir(fullfile(pathname, '*_dir'));
    N_subfolder = size(subfolderlist, 1);
    for j = 1 : N_subfolder
        subfoldername = fullfile(subfolderlist(j).folder,
subfolderlist(j).name);
        ctrlfile = dir(fullfile(subfoldername,
'All*.mat'));
        ctrlfilename = ctrlfile.name;
        t1 = 3;
        [Q_anterior, Q_immobile, Q_posterior] =
PostProcessing_cfactor_statistic_auto(ctrlfilename,
subfoldername, t1);
        Ya{i} = cat(1, Ya{i}, Q_anterior-1);
        Ym{i} = cat(1, Ym{i}, Q_immobile-1);
        Yp{i} = cat(1, Yp{i}, Q_posterior-1);
        Q_anterior(isnan(Q_anterior)) = [];
        Y{i} = cat(1, Y{i}, Q_anterior-1);
    end
end
% %% Plot scatter plot to show the relationship between
anterior and immobile
% figure(1)
% for i = 1 : N_strains
%     subplot(5, 5, i)
%     ya = Ya{i};
%     ym = Ym{i};
%     todel = any(cat(2,isnan(ym),isnan(ya)), 2);
%     ym(todel) = [];
%     ya(todel) = [];
%     scatter(ym, ya, 36, 'k', 'filled')
%     p = polyfit(ym, ya, 1);
%     Yafit = polyval(p, ym);
%     Yares = ya - Yafit;
%     SSres = sum(Yares.^2);
%     SStot = (length(ya)-1) * var(ya);
%     rsqrt = sqrt(1 - SSres/SStot);
%     hold on
%     fplot(@(x) polyval(p,x), [min(ym) max(ym)], 'r')
%     hold off
%     xlabel('M')
%     ylabel('A')
%     title([Genonames{i} sprintf('  (R = %.4f)', rsqrt)])
%     xlim([.4 2])
%     ylim([.4 2])
% end

%% Plot bargraph for all strains with normalized amplitudes
comparing with 1
X1 = [];
Ymean = zeros(1, N_strains);
for i = 1 : N_strains
    Ymean(i) = mean(Y{i});
end
I = 1 : N_strains;
% [~, I] = sort(Ymean, 'descend');
```

```
Y_sort = Y(I');
Genonames_sort = Genonames(I');
for i = 1 : N_strains
    X1 = cat(1, X1, i*ones([numel(Y_sort{i}) 1]));
end
% figure(2)
% for i = 1:numel(X)
%     x = X{i};
%     y = Y{i};
%     F_bar(i) = mean(y);
%     F_err(i) = std(y)./sqrt(numel(x));
%     [H2one(i), Pvals2one(i)] = ttest(y, 1, 'Tail',
'right');
%     hold on
%     scatter(x, y, 10, 'b','filled')
%     txt = num2str(Pvals2one(i), '%.1e');
%     if H2one(i) == 0
%         text(X_bar(i), .25,txt, 'HorizontalAlignment',
'center', 'Color', 'r')
%     else
%         text(X_bar(i), .25,txt, 'HorizontalAlignment',
'center', 'Color', 'k')
%     end
%     hold off
% end
%
% figure(2); hold on
% bar(X_bar, F_bar, 'FaceColor', 'none', 'LineWidth', 2)
% errorbar(X_bar, F_bar, F_err, 'LineStyle', 'none',
'Color', 'k', 'LineWidth',1.5)
% hold off
%
% hold on
% line([0 N_strains+1],[1 1], 'Color', 'red', 'LineStyle',
'--', 'LineWidth', 1)
% hold off
% ylim([0 2.5])
% xticks(X_bar)
% xticklabels(Genonames)
% ylabel('Normalized amplitude (|K|)')
% title('Comparison with 1 (one-tail one-sample t test)')
% set(gcf, 'Position', [740,220,530,498])
% set(gca, 'FontSize', 15)
% set(gca, 'XTickLabelRotation', 45)
%% Plot bargraph for all strains with normalized amplitudes
comparing with wt
Y1 = [];
for i = 1:numel(Y_sort)
    Y1 = cat(1, Y1, Y_sort{i});
    [H2wt(i), Pvals2wt(i)] = ttest2(Y_sort{1}, Y_sort{i},
'Tail', 'both');
%     [H2wt(i), Pvals2wt(i)] = ttest2(y, Y{1}, 'Tail',
'both');
%     hold on
%     if i~= 1
%         txt = num2str(Pvals2wt(i), '%.1e');
%         if H2wt(i) == 1
%             text(X_bar(i), .25,txt,
'HorizontalAlignment', 'center', 'Color', 'r')
%         else
%             text(X_bar(i), .25,txt,
'HorizontalAlignment', 'center', 'Color', 'k')
%         end
%     end
%     hold off
end

figure(5);clf
beeswarm(X1,Y1,'sort_style','hex','dot_size',.5,'overlay_st
yle','ci','corral_style','omit');
% ylim([-0.5 1.5])
% xlim([0.3 3.7])
xticks(X_bar)
xticklabels(Genonames_sort)
ylabel(sprintf('Normalized anterior\n curvature change'))
% title('Comparison with wildtype (two-tail two-sample t
test)')
set(gcf,'Position', [69,291,1200,400])
set(gca,'FontName','Arial','FontSize',20)
set(gca, 'XTickLabelRotation', 0)
%% Fit linear model with continuous factors (constrained
curvature change) and categorical factors (strain types)
for i = 1 : N_strains
    Genotype = Genonames{i};
    ya = Ya{i};
    ym = Ym{i};
    todelete = any([isnan(ya), isnan(ym)],2);
    ya(todelete) = [];
    ym(todelete) = [];
    N_trials = numel(ya);
    for j = 1 : N_trials
        Trial_info = {Genotype, ym(j), ya(j)};
        if i==1 && j==1
            T_exp = table(categorical({Genotype}),
double(ym(j)), double(ya(j)), ...
                'VariableNames',{'Genotype', 'Curv_imm',
'Curv_ant'});
        else
            T_exp = [T_exp; Trial_info];
        end
    end
end

mdl = fitlm(T_exp, 'ResponseVar', 'Curv_ant',...
            'PredictorVars', {'Genotype', 'Curv_imm'},...
            'CategoricalVars', {'Genotype'})
% writetable(mdl.Coefficients,'F:\Compensatory
experiments\Collective Results\fitLM.xls','Sheet',1)
rownames = mdl.CoefficientNames';
% % %% Fit linear model with continuous factors
(constrained curvature change) and categorical factors
(strain types) with full indicator variables
% % Genotypes_all = T_exp{:,1};
% % temp_Genotypes= dummyvar(categorical(Genotypes_all));
% % N2      = temp_Genotypes(:,1);
% % dop_1   = temp_Genotypes(:,2);
% % dop_2   = temp_Genotypes(:,3);
% % dop_3   = temp_Genotypes(:,4);
% % dop_4   = temp_Genotypes(:,5);
% % dop_12  = temp_Genotypes(:,6);
% % dop_13  = temp_Genotypes(:,7);
% % dop_123 = temp_Genotypes(:,8);
% % dat_1   = temp_Genotypes(:, 9);
% % cat_2   = temp_Genotypes(:,10);
% % mec_4   = temp_Genotypes(:,11);
% % mec_4d  = temp_Genotypes(:,12);
% % mec_10  = temp_Genotypes(:,13);
% % mec_3   = temp_Genotypes(:,14);
% % del_1   = temp_Genotypes(:,15);
% % unc_31  = temp_Genotypes(:,16);
% % unc_8   = temp_Genotypes(:,17);
% % unc_25  = temp_Genotypes(:,18);
% % unc_49  = temp_Genotypes(:,19);
% % lon_2   = temp_Genotypes(:,20);
% % trp_4   = temp_Genotypes(:,21);
% % trpa_1  = temp_Genotypes(:,22);
% % trp_12  = temp_Genotypes(:,23);
% % Curv_imm = T_exp{:,2};
% % Curv_ant = T_exp{:,3};
% % T_exp2   = table(N2, dop_1, dop_2, dop_3, dop_4,
dop_12, dop_13, dop_123, ...
% %                  dat_1, cat_2, mec_4, mec_4d, mec_10,
mec_3, del_1, unc_31, ...
% %                  unc_8, unc_25, unc_49, lon_2, trp_4,
trpa_1, trp_12, Curv_imm, Curv_ant);
% % mdl2     = fitlm(T_exp2, 'Curv_ant ~ N2 + dop_1 + dop_2
+ dop_3 + dop_4 + dop_12 + dop_13 + dop_123 + dat_1 + cat_2
+ mec_4 + mec_4d + mec_10 + mec_3 + del_1 + unc_31 + unc_8
+ unc_25 + unc_49 + lon_2 + trp_4 + trpa_1 + trp_12 +
Curv_imm - 1')
%% One-way ANOVA
[p,tbl,stats] = anova1(Y1',X1');
multcompare(stats)
```

Gcamp_correlation_soma.m

```
clc; clear;
Parent = 'C:\Users\fffei\Dropbox\Paper\Compensatory reponse
mechanism\data optogenetics\GCaMP expts new\BZ555 p05
agarpad\Intermediate data\New folder';
Pr = dir(fullfile(Parent, '*w*'));
N = numel(Pr);
numcurvpts = 100;
numseg = 5;
fps = 10;
nn  = numcurvpts/numseg;
K   = [];
F   = [];
Tmin = 10; % 100 180
t0  = 5;
for i = 1 : N
    Prname = Pr(i).name;
    fullname = fullfile(Parent, Prname);
    if Prname(1) == '1'
        Pd = dir(fullfile(fullname, 'v_*'));
        Pv = dir(fullfile(fullname, 'd_*'));
    elseif Prname(1) == '2'
        Pd = dir(fullfile(fullname, 'd_*'));
        Pv = dir(fullfile(fullname, 'v_*'));
    end
    Pg = dir(fullfile(fullname, 'soma*'));
    Fd = double(imread(fullfile(Pd.folder, Pd.name))');
Fd([1:21 end-20:end], :) = [];
    Fv = double(imread(fullfile(Pv.folder, Pv.name))');
Fv([1:21 end-20:end], :) = [];
    Fc = Fv - Fd;
    Fg = double(imread(fullfile(Pg.folder, Pg.name)));
```

```
    Fc_resc = interp1(0 : size(Fc,1)-1, Fc, (0 :
99)./99.*(size(Fc,1)-1))';
    fg = mean(Fg, 2);
    f0 = prctile(fg,25);
    fg = (fg-f0)/f0;
    T  = size(fg, 1);
    if T <= Tmin
        continue
    end
    % down-sample the curvature kymograph in body
coordinate
    K_per_worm = zeros(T, numseg);
    for j = 1 : numseg
        curvrgn = 1 + nn*(j-1) : nn*j;
        fc = mean(Fc_resc(:, curvrgn),2);
        K_per_worm(:,j) = fc;
    end
    % down-sample the curvature and fluorescence signal in
time
    T0 = floor(T/t0);
    Kn_per_worm = zeros(T0, numseg);
    Fn_per_worm = zeros(T0, 1);
    for k = 1 : T0
        timerange = 1 + t0*(k-1) : t0*k;
        Kn_per_worm(k, :) = mean(K_per_worm(timerange, :),
1);
        Fn_per_worm(k)    = mean(fg(timerange));
    end
    K = cat(1, K, Kn_per_worm/30);
    F = cat(1, F, Fn_per_worm);
end
R_all = zeros(1, numseg);
RL_all = zeros(1, numseg);
RU_all = zeros(1, numseg);

figure(1); clf
for i = 1 : numseg
    subplot(1, numseg, i)
    plot(K(:,i), F, '.');
    [R, P, RL, RU] = corrcoef(K(:,i), F);
    xlabel(sprintf('K (%.1f portion)', (i-.5)/(numseg)));
    ylabel('\Delta F/F0');
    title(sprintf('R = %f\n P = %f', R(1,2), P(1,2)))
    R_all(i) = R(1,2);
    RL_all(i) = RL(1,2);
    RU_all(i) = RU(1,2);
%     ylim([47300 84000])
    set(gca, 'FontSize', 12)
end

X = ((1:numseg)-0.5)/numseg;
figure(2); clf
errorbar(X, R_all, RL_all, RU_all)
xlabel('Body coordinate')
ylabel('Correlation with GCaMP')
xlim([0 1])
ylim([-1 1])
set(gca, 'FontSize', 12)

%% Generate GCaMP vs curvature scatter plot for mid-body
KK= K(:,3);
% Concatenate the F results by maxima and minima
    F  = F(:);
    KK = KK(:);
    Fl = F;

% Sort data by phase at pulse
    [~,idx]    = sort(KK);
    K_sort     = KK(idx);
    F_sort     = Fl(idx);

% Plot results
opengl software;
    figure(3);
%     clf;
    hold on

%Moving average
    Npoints     = numel(F_sort);
    w_idx       = round(0.25 * Npoints);               %
Normal is 0.15 The width of the median bin in elements.
Also the N value for each bin. Note that this method could
be invalid if the phases are not sampled approximately
equally.
    AVG = movmean(F_sort,w_idx);
    SEM = movstd(F_sort,w_idx)/sqrt(w_idx);
    shadedErrorBar(K_sort,AVG,SEM,'r'); hold on;

% Individual points
%
plot(K_sort,F_sort,'.','MarkerSize',8,'Color',0.5*[0.5 0.5
1]);
    xlabel('Normalized mid-body curvature');
    ylabel('D F/F0')

    set(gcf,'Color','w','Position',[1192 217 700 570]);
    set(gca,'FontSize',20);
    ylim([0 .3])
    xlim([-5 5])

%% Generate GCaMP vs curvature scatter plot for mid-body
KK= K(:,3);
% Concatenate the F results by maxima and minima
F  = F(:);
KK = KK(:);
Fl = F;

% Sort data by phase at pulse
[~,idx]    = sort(KK);
K_sort     = KK(idx);
F_sort     = Fl(idx);

% Plot results
opengl software;
figure(4);
%     clf;
hold on

bin = -5:1:5;
[kk,ff] = aggregatehist(bin, K_sort, F_sort);
dev = cellfun(@std, ff)./sqrt(cellfun('size', ff,1));
avg = cellfun(@mean, ff);
kmid = (bin(1:end-1)+bin(2:end))./2;

plot(kmid,avg,'y')
hold on;
errorbar(kmid, avg, dev, 'linestyle', 'none', 'marker',
'o');
% set(gca, 'xtick', bin, 'xgrid', 'on');



% Individual points
%
plot(K_sort,F_sort,'.','MarkerSize',8,'Color',0.5*[0.5 0.5
1]);
xlabel('Normalized mid-body curvature');
ylabel('D F/F0')

set(gcf,'Color','w','Position',[1192 217 700 570]);
set(gca,'FontSize',20);
% ylim([0 .3])
xlim([-5 5])
```

Gcamp_time_dynamic.m

```
clc; clear;
Parent = 'C:\Users\fffei\Dropbox\Paper\Compensatory reponse
mechanism\data optogenetics\GCaMP expts new\Combined_SWF331
p05 agarpad\N folder';
Pr = dir(fullfile(Parent, '*w*'));
N = numel(Pr);
numcurvpts = 100;
numseg = 5;
fps = 10;
nn  = numcurvpts/numseg;
x   = (0:numcurvpts-1)/(numcurvpts-1);
K   = [];
F   = [];
Tmin = 100;
t0  = 5;
II = [56 58 61 63 1 4 6 7 19 21 27 28 36 49 53 56 58 61
63];
FFg = [];
FFc = [];
for i = [1] % 1
    Prname = Pr(i).name;
    fullname = fullfile(Parent, Prname);
    if Prname(1) == '1'
        Pd = dir(fullfile(fullname, 'v_*'));
        Pv = dir(fullfile(fullname, 'd_*'));
    elseif Prname(1) == '2'
        Pd = dir(fullfile(fullname, 'd_*'));
        Pv = dir(fullfile(fullname, 'v_*'));
    end
    Pg = dir(fullfile(fullname, 'soma*'));
    Fd = double(imread(fullfile(Pd.folder, Pd.name))');
Fd([1:21 end-20:end], :) = [];
    Fv = double(imread(fullfile(Pv.folder, Pv.name))');
Fv([1:21 end-20:end], :) = [];
    Fc = Fv - Fd;
    Fg = double(imread(fullfile(Pg.folder, Pg.name)));
    Fc_resc = interp1(0 : size(Fc,1)-1, Fc, (0 :
99)./99.*(size(Fc,1)-1))';
    fg = mean(Fg, 2);
    f0 = prctile(fg,25);
    fg = (fg-f0)/f0;
```

```
    T  = size(fg, 1);
    t  = (0:T-1)/fps;
    if T <= Tmin
        continue
    end
%     figure(1);clf
%     subplot(211)
%     plot(t, fg);
%     xlim([0 t(end)])
%     ylabel('D F/F0')
%     subplot(212)
%     imagesc(t,x, Fc_resc');
%     xlim([0 t(end)])
%     ylabel('Body coordinate')
%     xlabel('Time (sec)')
%     expo = 0.7;
%     colormap(cmap_redblue(expo));
    if any(II(:) == i)
        t0 = 9.9*fps;
        if i == 63
            fg      = fg(1:t0);
            Fc_resc = Fc_resc(1:t0,:);
        end
        FFg = cat(1, FFg, fg);
        FFc = cat(1, FFc, Fc_resc);
    end
%     i
%     T
end
T  = size(FFg, 1);
t  = (0:T-1)/fps;
%% Free

figure(1);clf
a = tiledlayout(2,1);
ax1 = nexttile;
plot(t, FFg);
xlim([0 t(end)])
ylabel('D F/F0')
set(gca, 'FontSize', 20)

ax2 = nexttile;
imagesc(t,x, flip(FFc',1));
set(gca, 'YDir', 'Normal')
expo = 0.7;
colormap(cmap_redblue(expo));
xlim([0 t(end)])
ylabel('Body coordinate')

xlabel(a,'Time (sec)', 'FontSize', 20)
set(gca, 'FontSize', 20)

xticklabels(ax1,{})
a.TileSpacing = 'compact';

%% Sinusoid
% t = t.*3/4;
FFc = FFc(:, 25:74)';
FFcs = interp1(0 : size(FFc,1)-1, FFc, (0 :
99)./99.*(size(FFc,1)-1))';
figure(2);clf
a = tiledlayout(2,1);
ax1 = nexttile;
plot(t, FFg);
ylim([-.2 1])
xlim([0 t(end)])
ylabel('D F/F0')
set(gca, 'FontSize', 20)

ax2 = nexttile;
imagesc(t,x, flip(FFcs',1));
set(gca, 'YDir', 'Normal')
expo = 0.7;
colormap(cmap_redblue(expo));
xlim([0 t(end)])
ylabel('Body coordinate')

xlabel(a,'Time (sec)', 'FontSize', 20)
set(gca, 'FontSize', 20)

xticklabels(ax1,{})
a.TileSpacing = 'compact';
```

Generalized_cfactor_for_microfluidics.m

```
function [Kc_all, dKdtc_all, Kp_data, dKdtp_data,
T0_avg]...
    = Generalized_cfactor_for_microfluidics(curv_ctrl,
curv_const, fps)
% GENERALIZED_CFACTOR defines and calculates a generalized
factor to
% quantify the compensation effects of anterior curvature
which is induced
% by (mechanically) perturbing the middle curvature during
the forward
% locomotion of a worm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%-09-24-
20-%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Modified the algorithm for computing generalized
compensatory factor of
%   the normal orbits over body coordinates.
%   Specific steps:
%   1. Define the time window(s) of normal undulations.
%   2. Calculate the average normal phase plot where phase
angles are
%   defined using peak-finding method
%   3. I divided the body into 5 different segments with
equal length and
%   calculate phase plot for each segments individually,
since curvature
%   dynamics varies across body coordinates.
%   Specifically, body is divided into segments 10%, 30%,
50%, 70%, and 90%
%   and phase plots elsewhere are computed through
interpolation or
%   extrapolation.

Nc = numel(curv_ctrl);
numcurvpts = size(curv_ctrl{1}, 2);
numsamplepts = 100;
curvrgn_analysis = 1 : numcurvpts;
curvrgn_sample = 10:5:90;
curvrgn_rd = 2;
numsegs = numel(curvrgn_sample);
Kc_avg_all = zeros(numsamplepts, numsegs);
dKdtc_avg_all = zeros(numsamplepts, numsegs);
T_all = [];
for ii = 1 : numsegs
    curvrgn = curvrgn_sample(ii);
    fprintf('Computing normal phase plot at region %.0f%%.
', curvrgn)
    Kc_data = {};
    dKdtc_data = {};
    Ntrials_c  = 0;
    curvrgn_win = curvrgn-curvrgn_rd : curvrgn+curvrgn_rd;
    for i = 1:Nc
        curvdatafiltered = curv_ctrl{i};
        v   = mean(curvdatafiltered(:,curvrgn_win),2);
        % Find all peaks
        [imax, imin] = C2_get_curvature_peaks(v,1);
        % Find the peaks in the period which is not
affected by illumination
        [imax, imin] = verify_extrema(v, imax, imin);
        imax = unique(imax);
        imin = unique(imin);
        imax(v(imax)<=0) = [];
        imin(v(imin)>=0) = [];
        T0 = mean([diff(imax); diff(imin)])/fps;
        zci = @(v) find(v(:).*circshift(v(:), [-1 0]) <=
0);
        zx = zci(v);
        num_max = numel(imax);
        for j = 1 : num_max - 1
            imaxs = imax(j);
            imaxe = imax(j+1);
            current_zx = zx(zx>imaxs & zx<imaxe);
            if numel(current_zx)~=2
                continue;
            end
            K = v(imaxs: imaxe);
            dKdt = gradient(K)*fps;
            dKdt(1) = 0;
            dKdt(end) = 0;
            Kc_data = [Kc_data; K];
            dKdtc_data = [dKdtc_data; dKdt];
            Ntrials_c = Ntrials_c + 1;
        end
        % record the period into matrix T
        current_seg = curvrgn_sample(ii);
        if current_seg>=20 && current_seg<=40
            T_all   = [T_all, T0];
        end
    end
    numhfsamplepts = numsamplepts/2;
    N_cycs = numel(Kc_data);
    Kc_resc_data = zeros(N_cycs, numsamplepts);
    dKdtc_resc_data = zeros(N_cycs, numsamplepts);
    for i = 1 : N_cycs
        v    = Kc_data{i};
        dvdt = dKdtc_data{i};
        [~, imin] = min(v);
        if length(imin)>1
            tmpdist2midpt = abs(imin - length(v)/2);
            [~,tmpI] = min(tmpdist2midpt);
            imin = imin(tmpI);
        end
```

```
        % rescale Ks and dKdts half cycle by half cycle
        vhf1 = v(1:imin); vhf2 = v(imin:end);
        qx1  = numel(vhf1) / (numel(vhf1)-1);
        qx2  = numel(vhf2) / (numel(vhf2)-1);
        vhf1_resc = interpl((0:numel(vhf1)-1), vhf1,
(numel(vhf1)-1)*(0:numhfsamplepts-1)/(numhfsamplepts-
1),'linear');
        vhf2_resc = interpl((0:numel(vhf2)-1), vhf2,
(numel(vhf2)-1)*(0:numhfsamplepts-1)/(numhfsamplepts-
1),'linear');

        dvdthf1 = dvdt(1:imin); dvdthf2 = dvdt(imin:end);
        qy1  = numel(dvdthf1) / (numel(dvdthf1)-1);
        qy2  = numel(dvdthf2) / (numel(dvdthf2)-1);
        dvdthf1_resc = interpl((0:numel(dvdthf1)-1),
dvdthf1, (numel(dvdthf1)-1)*(0:numhfsamplepts-
1)/(numhfsamplepts-1),'linear');
        dvdthf2_resc = interpl((0:numel(dvdthf2)-1),
dvdthf2, (numel(dvdthf2)-1)*(0:numhfsamplepts-
1)/(numhfsamplepts-1),'linear');

        % combine halfs to get full cycles
        v_rescaled = [vhf1_resc vhf2_resc];
        dvdt_rescaled = [dvdthf1_resc dvdthf2_resc];
        Kc_resc_data(i,:) = v_rescaled;
        dKdtc_resc_data(i,:) = dvdt_rescaled;
    end
    % Calculating averages for current segment
    Kc_resc_avg    = mean(Kc_resc_data,1);
    dKdtc_resc_avg = mean(dKdtc_resc_data,1);
    Kc_avg_all(:, ii) = Kc_resc_avg;
    dKdtc_avg_all(:, ii) = dKdtc_resc_avg;
    fprintf('\n'); % To go to a new line after reaching
100% progress
end
T0_avg = mean(T_all, 'omitnan');
% Adjust the averaged curvature cycle by fixing the minimum
point and
% scaling the near-end points so that it will equal to the
negative
% amplitude
for ii = 1 : numsegs
    v  = Kc_avg_all(:,ii);
    [~,pf] =  min(v); pf = pf(1);
    vf = abs(v(pf));
    v(1 : pf-1) = v(1 : pf-1) + (pf - (1 : pf-1)')/(pf - 1)
* (vf - v(1));
    v(pf : end) = v(pf : end) + ((pf : numel(v))' -
pf)/(numel(v) - pf) * (vf - v(end));
    Kc_avg_all(:,ii) = v;
end
% Using interpolation and extrapolation to predict the
phase plots on other
% segments of a worm
Kc_all    = interpl(curvrgn_sample,
Kc_avg_all',curvrgn_analysis, 'makima')';
dKdtc_all  = interpl(curvrgn_sample,
dKdtc_avg_all',curvrgn_analysis, 'makima')';
Kc_all    = Kc_all';
dKdtc_all = dKdtc_all';
%%
%%%%%%%%%%%%%%%%%%%%
if isempty(curv_const)
    Kp_data = [];
    dKdtp_data = [];
else
    % analysis of the contrained group
    Np = numel(curv_const);
    Ntrials_p = 0;
    Kp_data    = {};
    dKdtp_data = {};
    for i = 1:Np
        fprintf('Analyzing trial %d',i)
        curvdatafiltered = curv_const{i};

        Ntrials_p = Ntrials_p + 1;

        % Analyzing bulk curvature
        Kb    = curvdatafiltered(:,curvrgn_analysis);
        dKdtb = gradient(Kb')' * fps;

        Kp_data    = [Kp_data; Kb'];
        dKdtp_data = [dKdtp_data; dKdtb'];

        fprintf('\n')
    end
end
end

    function [imax,imin] = verify_extrema(v,imax,imin)

        % get the mean amplitudes
        vmax = mean(v(imax));
        vmin = mean(v(imin));

        if vmin > vmax
            itemp = imax;
            imax  = imin;
            imin  = itemp;
        end


    end
```

Generating_normal_atlas.m

```
% Worm shape analysis. This version is modified for
analyzing data obtained
% from worm locomotion on WormTunnel microfluidic chambers.
%
% First, periods during which a worm is undulating without
constraint will be
% collected and analyzed to generate a limit cycle for
normal undulatory
% dynamics.
%
% Next, periods during which a worm is moving under
constraint (usually at
% the middle of body) will be collected and analyzed to
generate phase
% dynamics during those periods
%
% Finally, combining the analyzed results from the above
two steps, a
% generalized compensatory factor kymogram will be
generated in a 2D
% heatmap form, which represents a
function
ion of time and body coordinate.
% Also, information of the body portion that is being
constrained will be
% reflected on the kymogram.
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%% START
MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
clc; close all; clear

global start_illum end_illum prefix pathname filename
global conc fps pix_per_mm wormthreshold isie decim filsize
spline_p initials


wormlabel  = 1;
pix_per_mm = 198.83;
prefix;
pathname;

curvlim = 0.2;
domovie = 0;
issavefiles = 0;
iscontinue = 1;

option1 = 'Yes (AVI only)';
option2 = 'Yes (MAT)';

fname  = questdlg('Pre or Post-const undulation?','','Pre-
','Post-','All-', 'All-');
button = length(questdlg('Load new
data?','',option1,option2,'No', option1));
CURV_all = {};
dCURV_all = {};
pathname_all = {};
if button == length(option2)
    disp('options2');
    [filename,pathname] = uigetfile({'*.mat'});
    matfname  = [fname filename(1:6)];
    load([pathname filename]);
elseif button == length(option1)
    disp('option1');
    do_dialog = 1;
    %% Identify and analyze control periods
    while iscontinue == 1

    if do_dialog
        try
            cd(pathname);
        catch
            pathname = pwd;
        end
        [filename,pathname]  = uigetfile('*.avi', 'Select
File');
        pathname_all = [pathname_all, pathname];
        matfname  = [fname filename(1:6)];
        vidObj = VideoReader(fullfile(pathname,filename));
        NumFrames = vidObj.NumFrames;
```

```
        fps       = vidObj.FrameRate;
        if isempty(conc)
            conc = 0;
        end
        if isempty(wormlabel)
            wormlabel = 1;
        end
        if isempty(pix_per_mm)
            pix_per_mm = 1;
        end
        if isempty(wormthreshold)
            wormthreshold = 0.1;
        end
        if isempty(isie)
            isie = [1, NumFrames];
        end
        if isempty(decim)
            decim = 1;
        end
        if isempty(filsize)
            filsize = 0.2;
        end
        if isempty(start_illum)
            start_illum = 1;
        end
        if isempty(end_illum)
            end_illum = 1;
        end
        if isempty(spline_p)
            spline_p = 0.01;
        end
        if isempty(initials)
            initials = {'JHF'};
        end

        fields={'conc','wormlabel', 'fps','pixels per
mm','Worm image threshold',...
            'istart/iend (use";"if multiple)', 'Decimation
(1=none))',...
            'Filter size / diameter',
'start_illum','end_illum', ...
            'spline fit parameter', 'Make movie?', 'Your
initials', 'Save files?'};
        if exist('isie', 'var')
            answer = inputdlg(fields, 'Cancel to clear
previous', 1, ...

{num2str(conc),num2str(wormlabel),num2str(fps),num2str(pix_
per_mm),num2str(wormthreshold),...
                mat2str(isie), num2str(decim),...

num2str(filsize),num2str(start_illum),num2str(end_illum), .
..
                num2str(spline_p), num2str(domovie),
initials{1}, num2str(issavefiles)});
        else
            answer = inputdlg(fields, '', 1);
        end

        if isempty(answer)
            pause;
        end

        conc = str2double(answer{1});
        wormlabel = str2double(answer{2});
        fps = str2double(answer{3});
        pix_per_mm = str2double(answer{4});
        wormthreshold = str2double(answer{5});
        isie = Str2Mat(answer{6});
        decim = str2double(answer{7});
        filsize = str2double(answer{8});
        start_illum = str2double(answer{9});
        end_illum = str2double(answer{10});
        spline_p = str2double(answer{11});
        domovie = str2double(answer{12});
        initials = answer(13);
        issavefiles = str2double(answer{14});
    end
    fileID = fopen(fullfile(pathname, 'timestemp for
ctrl.txt'),'a+');
    fprintf(fileID, [filename,': ', answer{6},'\n']);
    fclose(fileID);
    nperiods   = size(isie, 1);
    curv_ctrl  = cell(nperiods, 1);
    dcurvdt_ctrl = cell(nperiods, 1);
    angle_ctrl = cell(nperiods, 1);
    len_ctrl   = cell(nperiods, 1);
    %  Compute undulatory variables for control groups
    for kk = 1 : nperiods
        do_const = 0;
        thisperiod = isie(kk, :);
        options = {conc, wormlabel, fps, pix_per_mm,
wormthreshold,...
                   thisperiod, decim, filsize, start_illum,
end_illum,...
                   spline_p, domovie, initials, pathname,
filename, do_const, issavefiles};
        [curv_ctrl{kk},dcurvdt_ctrl{kk}, angle_ctrl{kk},
len_ctrl{kk}]...
            = WORMSHAPE_MAINCALCULATION(vidObj, options);
    end

    % flip some periods where head/tail misidentified
    for i = 1 : nperiods
        K = curv_ctrl{i};
        dKdt = dcurvdt_ctrl{i};
        ang  = angle_ctrl{i};
        % debug plot
        figure(10); clf
        imagesc(K)
        colormap(cmap_redblue(0.7))
        caxis([-25 25])
        colorbar
        hold on
        answer = length(questdlg('Need to flip some
period?', '','Yes', 'No', 'No'));
        if answer == 3
            title('Indicate the period that need to be
flipped')
            % flip curvature and dKdt
            [~, flpy1] = ginput(1);
            flpy1 = max([floor(flpy1) 1]);
            line([1 100], [flpy1
flpy1],'Color','white','LineStyle','--')
            [~, flpy2] = ginput(1);
            flpy2 = min([floor(flpy2) size(K,1)]);
            line([1 100], [flpy2
flpy2],'Color','white','LineStyle','--')
            K2flip = K(flpy1 : flpy2, :);
            dKdt2flip = dKdt(flpy1 : flpy2, :);
            ang2flip  = ang(flpy1 : flpy2, :);
            K(flpy1 : flpy2, :) = flip(K2flip,2);
            dKdt(flpy1 : flpy2, :) = flip(dKdt2flip,2);
            ang(flpy1 : flpy2, :) = flip(ang2flip,2);
        end
        % updata data
        curv_ctrl{i}  = K;
        dcurvdt_ctrl{i} = dKdt;
        angle_ctrl{i}  = ang;
        % show updated plot
        figure(10); clf
        imagesc(K)
        colormap(cmap_redblue(0.7))
        colorbar
        pause(1)
    end
     % end fliping loop
    CURV_all = [CURV_all; curv_ctrl];
    dCURV_all = [dCURV_all; dcurvdt_ctrl];
    Cbutton =
length(questdlg('Continue?','','Yes','No','Yes'));
    if Cbutton == 3
        iscontinue = 1;
    else
        iscontinue = 0;
    end
    close all
    end
end
%% Calculate the average normal phase plot from control
groups
fps = 30;
[Kc_all, dKdtc_all, Kp_data, dKdtp_data, T0_avg]...
    = Generalized_cfactor_for_microfluidics(CURV_all, [],
fps);


numsamplepts = 100;
numcurvpts   = 100;
a  = .15; c = a * T0_avg;
Zc = Kc_all + 1i*c*dKdtc_all;
Pc = unwrap(angle(Zc), [], 2);
[~, Sc] = meshgrid(1:numsamplepts, 1:numcurvpts);
% Generate interpolant (in a bulk manner)
FR = scatteredInterpolant(Pc(:), Sc(:), Zc(:), 'linear',
'nearest');

%%  Plotting data
curvrgn_analysis = 1 : numcurvpts;
% 3-D phase portrait plot of normal undulation
figure(1); clf
TX_mesh = meshgrid(1 : numsamplepts, curvrgn_analysis);
hold on
plot3(Kc_all', dKdtc_all', TX_mesh)  % isosegmental line
plot3(Kc_all,  dKdtc_all,  TX_mesh') % isophasic line
hold off
```

```
view([30,30])
xl = xlim;
yl = ylim;
zlim([5 95])
xlabel('K')
ylabel('dKdt')
zlabel('Body coordinate')
set(gca, 'FontSize', 12, 'Position',
[0.13,0.11,0.775,0.815])

% GUI with interactive response-plot updates for phase
portrait plots
s  = 30;
f  = figure(2); clf
ax = axes('Parent',f,'position',[0.2 0.25  0.65 0.65],
'PlotBoxAspectRatio', [1,0.81,0.75]);
faseplot2  = @(ax, s) phasePlot2(curvrgn_analysis, Kc_all',
dKdtc_all', s, ax, xl, yl);
faseplot2(ax, s);
b =
uicontrol('Parent',f,'Style','slider','Position',[81,34,419
,23],...
              'value',s, 'min',5, 'max',95);
bgcolor = f.Color;
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[45,37,30,20],...
                'String','Head','BackgroundColor',bgcolor);
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[505,37,30,20],...
                'String','Tail','BackgroundColor',bgcolor);
uicontrol('Parent',f,'Style','text','FontSize',12,'Position
',[240,10,100,23],...
                'String','Body
coordinate','BackgroundColor',bgcolor);

b.Callback = @(es,ed) faseplot2(ax, es.Value);
%% save the NormalUndulation.mat file to all visited
folders
for i = 1 : numel(pathname_all)
    fname4save = fullfile(pathname_all{i}, [matfname 'imm
Normal.mat']);
    save(fname4save, 'Kc_all', 'dKdtc_all', 'Zc', 'FR',
'CURV_all', 'fps', 'T0_avg')
end
```

PostProcessing_cfactor_beeswarm.m

```
clc; clear; close all
% get the normal undulation data
[ctrlfilename,pathname] = uigetfile({'*.mat'});
% locate the folder
dir_list = dir(fullfile(pathname, '*data.mat'));
nworms   = numel(dir_list);
t0   = 3; % sampling period
t_step = t0; % step of moving during sampling
v    = [];% preallocate for data pool after sampling with
period t0. dim(v) = N * i0 * 403
prog = 0;
issave = 1;
%% Control group
fprintf('------Progress: %3.0f%% \n',prog);
% load and resampling the data
load(fullfile(pathname, ctrlfilename))
nnormw = numel(CURV_all);
do_flip = length(questdlg('Need to do flipping for ctrl?',
'','Yes', 'No', 'No'));
for i = 1 : nnormw
    prog = 100*i/nnormw;
    if i > 1
        fprintf('\b\b\b\b%3.0f%%',prog);
    end

    i0 = floor(t0*fps);
    step = floor(t_step*fps);
    K    = CURV_all{i};
    dKdt = gradient(K')'*fps;
    s    = 1 : size(K, 2);
    if do_flip == 3
        %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
        % debug plot
        figure(10); clf
        imagesc(K)
        colormap(cmap_redblue(0.7))
        caxis([-25 25])
        colorbar
        set(gcf, 'Position', [581,42,584,1314])
        hold on
        answer = length(questdlg('Need to flip some
period?', '','Yes', 'No', 'No'));
        if answer == 3
            title('Indicate the period that need to be
flipped')
            % flip curvature and dKdt
            [~, flpy1] = ginput(1);
            flpy1 = max([floor(flpy1) 1]);
            line([1 100], [flpy1
flpy1],'Color','white','LineStyle','--')
            [~, flpy2] = ginput(1);
            flpy2 = min([floor(flpy2) size(K,1)]);
            line([1 100], [flpy2
flpy2],'Color','white','LineStyle','--')
            K2flip = K(flpy1 : flpy2, :);
            K_flipped = flip(K2flip,2);
            K(flpy1 : flpy2, :) = K_flipped;
        end
        dKdt = gradient(K')'*fps;
        %
    end
        % recalculate the generalized compensatory factor
and save it
        numsamplepts = 100;
        numcurvpts   = 100;
        a  = .15; c = a * T0_avg;
        Zc = Kc_all + 1i*c*dKdtc_all;
        Pc = unwrap(angle(Zc), [], 2);
        [~, Sc] = meshgrid(1:numsamplepts, 1:numcurvpts);
        % Generate interpolant (in a bulk manner)
        FR = scatteredInterpolant(Pc(:), Sc(:), Zc(:),
'linear', 'nearest');
        % Constrcting complex curvature dynamics for pulsed
group
        Zp     = K' + 1i*c*dKdt';
        Pp_ori = angle(Zp); % do not use unwrap
        Ppld   = Pp_ori(:);
        Ppld(Ppld>0) = Ppld(Ppld>0) - 2*pi;
        Pp = reshape(Ppld, size(Pp_ori));
        [~, Sp] = meshgrid(1:size(Pp,2), 1:size(Pp,1));
        Rp = abs(Zp);
        Zc4pld = FR(Pp(:), Sp(:));
        Zc4p = reshape(Zc4pld, size(Pp));
        Rc4p = abs(Zc4p);
        cfac = (Rp) ./ Rc4p;
        % updata data
        CURV_all{i}  = K;
        dCURV_all{i} = dKdt;
        % show updated plot
        if do_flip == 3
            figure(10); clf
            imagesc(K)
            colormap(cmap_redblue(0.7))
            colorbar
            pause(1)
        end
        %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
    % convert immobilization position from boolean exp to
coord exp
    u  = cfac';
    curv = K;
    n  = size(u, 1);
    iwin = 1 : i0;
    while iwin(end) <= n
        thisu = u(iwin, :);
        thisk = curv(iwin, :);
        thiscfac = thisu(:, 1:100);
        iwin = iwin + step;
        if max(thiscfac,[],'all') > 50
            continue
        end
        thisuk = cat(2, thisu, thisk);
        v   = cat(3, v, thisuk);
    end
end
fprintf('\n')
save(fullfile(pathname, ctrlfilename),
'CURV_all','dCURV_all','-append')
vc = permute(v, [3 1 2]);
%% Constrained group
v = [];
%%%
wconst = 60; % normally, the width of the channel is 60 um
%%%
prog = 0;
fprintf('------Progress: %3.0f%% \n',prog);
do_flip = length(questdlg('Need to do flipping for const?',
'','Yes', 'No', 'No'));
% load and resampling the data
load(fullfile(pathname, ctrlfilename))
for i = 1 : nworms
    prog = 100*i/nworms;
    if i > 1
        fprintf('\b\b\b\b%3.0f%%',prog);
    end
    thisworm  = dir_list(i);
    fname = fullfile(thisworm.folder, thisworm.name);
    load(fname)
```

```
    i0 = floor(t0*fps);
    step = floor(t_step*fps);
    ntrials = numel(CR);
    for j = 1:ntrials
        cfac = CR{j}';
        K    = Kp_data{j}';
        dKdt = dKdtp_data{j}';
        len  = len_const{j};
        rgc  = rgn_const{j};
        wdia = w_diam{j}*1000; % unit: um
        s =  1 : size(cfac, 2);
        t = (0 : size(cfac, 1)-1)'/fps;
        if do_flip == 3
            %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
            % debug plot
            figure(10); clf
            imagesc(K)
            colormap(cmap_redblue(0.7))
            caxis([-25 25])
            colorbar
            set(gcf, 'Position', [581,42,584,1314])
            hold on
            answer = length(questdlg('Need to flip some
period?', '','Yes', 'No', 'No'));
            if answer == 3
                title('Indicate the period that need to be
flipped')
                % flip curvature and dKdt
                [~, flpy1] = ginput(1);
                flpy1 = max([floor(flpy1) 1]);
                line([1 100], [flpy1
flpy1],'Color','white','LineStyle','--')
                [~, flpy2] = ginput(1);
                flpy2 = min([floor(flpy2) size(K,1)]);
                line([1 100], [flpy2
flpy2],'Color','white','LineStyle','--')
                K2flip = K(flpy1 : flpy2, :);
                K_flipped = flip(K2flip,2);
                K(flpy1 : flpy2, :) = K_flipped;
                dK2flip = dKdt(flpy1 : flpy2, :);
                dK_flipped = flip(dK2flip,2);
                dKdt(flpy1 : flpy2, :) = dK_flipped;
                r2flip = rgc(flpy1 : flpy2, :);
                r_flipped = flip(r2flip,2);
                rgc(flpy1 : flpy2, :) = r_flipped;
                %
                % recalculate the generalized compensatory
factor and save it
                numsamplepts = 100;
                numcurvpts   = 100;
                a  = .15; c = a * T0_avg;
                Zc = Kc_all + 1i*c*dKdtc_all;
                Pc = unwrap(angle(Zc), [], 2);
                [~, Sc] = meshgrid(1:numsamplepts,
1:numcurvpts);
                % Generate interpolant (in a bulk manner)
                FR = scatteredInterpolant(Pc(:), Sc(:),
Zc(:), 'linear', 'nearest');
                % Constrcting complex curvature dynamics
for pulsed group
                Zp     = K' + 1i*c*dKdt';
                Pp_ori = angle(Zp); % do not use unwrap
                Ppld   = Pp_ori(:);
                Ppld(Ppld>0) = Ppld(Ppld>0) - 2*pi;
                Pp = reshape(Ppld, size(Pp_ori));
                [~, Sp] = meshgrid(1:size(Pp,2),
1:size(Pp,1));
                Rp = abs(Zp);
                Zc4pld = FR(Pp(:), Sp(:));
                Zc4p = reshape(Zc4pld, size(Pp));
                Rc4p = abs(Zc4p);
                cr_new = (Rp) ./ Rc4p;
                % updata data
                CR{j} = cr_new;
                Kp_data{j} = K';
                dKdtp_data{j} = dKdt';
                rgn_const{j}  = rgc;
                % show updated plot
                figure(10); clf
                subplot(121)
                imagesc(K)
                colormap(cmap_redblue(0.7))
                colorbar
                subplot(122)
                imagesc(cr_new')
                colorbar
                pause(1)
            end
            %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
        end
        % convert immobilization position from boolean exp
to coord exp
        rgc(rgc == 0) = nan;
        imb = rgc .* repmat(s, [length(t), 1]);
        imbl = min(imb, [], 2, 'omitnan');
        imbr = max(imb, [], 2, 'omitnan');
        imbm = mean(imb, 2, 'omitnan');
        u  = cat(2, cfac, K, repmat([wdia, wconst],
[length(t) 1]), len, imbl, imbr, imbm);
        n  = size(u, 1);
        iwin = 1 : i0;
        while iwin(end) <= n
            thisu = u(iwin, :);
            thiscfac = thisu(:, 1:100);
            iwin = iwin + step;
            if max(thiscfac,[],'all') > 50
                continue
            end
            imbc = mean(thisu(:, 206));
            tmp = cat(2, thisu, repmat(imbc, [i0, 1]));
            v   = cat(3, v, tmp);
        end
    end
    save(fname,
'CR','Kp_data','dKdtp_data','rgn_const','len_const' ,'-
append')
end
fprintf('\n')
vp = permute(v, [3 1 2]);
%% Plotting results

% making new folder to save results
if issave == 1
    savefoldername = fullfile(pathname, 'Results');
    mkdir(savefoldername)
end

N   = size(vp, 1);
Loc_im = vp(:,1,207);
edges_locim = [1,25,40,60,100];
% figure(1);clf
% himb = histogram(Loc_im);
% title('distribution of trials to immob location')

Int_im = vp(:,1,201)./vp(:,1,202);
tightness = 0;
figure(2);clf
hint = histogram(Int_im);
title('distribution of trials to tightness of
immobilization')
set(gcf, 'Position', [1200,160,570,490])
if issave == 1
    saveas(gcf, fullfile(savefoldername, 'distrubution of
tightness.fig'));
    saveas(gcf, fullfile(savefoldername, 'distrubution of
tightness.png'));
end

Len = squeeze(mean(vp(:,:,203),2));
edges_len = [.8,.98, 1.08,1.3];
% figure(3);clf
% hlen = histogram(Len);
% title('distribution of trials to worm length')



% % cfactor's dependence on location of immobilization
(spectrum by tightness of immobilization)
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0.4 1.5];
% figure(4);clf
% % calculating the control group
% for i = 1 : length(edges_locim)-1 % first grouped by loc
of immo
%     thisedge = edges_locim(i:i+1);
%     idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
%     thisrgclr = vp(idx_rgc, :, 204:205);
%     thisrgclr_mean = squeeze(mean(thisrgclr,1));
%     L = mean(thisrgclr_mean(:,1));
%     R = mean(thisrgclr_mean(:,2));
%     ind_2nd   = Int_im;
%     edges_2nd = edges_len;
%     idx = idx_rgc;
%     thisintimb = mean(ind_2nd(idx));
%     thiscfac  = vp(idx, :,1:100);
%     thiscfac_mean = squeeze(mean(thiscfac,1));
%     h = {};
%     text_edge2 = {};
%     subplot(2,ceil((length(edges_locim)-1)/2),i)
%     %     imagesc(s2, t2, thiscfac_mean)
%     hold on
```

```
%     % plot the control group
%     tmph = plot(s, mean(squeeze(mean(vc(:,:,s),1)),1),
'LineWidth', 2, 'Color', 'k');
%     h = [h, tmph];
%     tmpt = sprintf('ctrl');
%     text_edge2 = [text_edge2, tmpt];
%     % plot the loc of immobiliaztion
%     line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     if ~isempty(thiscfac)
%         tmph = plot(s, mean(thiscfac_mean,1),
'LineWidth', 2);
%         h = [h, tmph];
%         tmpt = sprintf('%.1f(%.0d)', thisintimb,
sum(idx));
%         text_edge2 = [text_edge2, tmpt];
%     end
%     hold off
%     xlim([5 100])
%     xlabel('body coordinate')
%     ylabel('normalized amplitude (gfac)')
%     title(sprintf('immob @ %.0f to %.0f%%', L, R))
%     set(gca, 'FontSize', 15)
%     legend(h, text_edge2)
% end
% set(gcf, 'Position', [583,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'gfac_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'gfac_locim_tightness.png'));
% end

% % Cfac_const / Cfac_ctrl dependence on location of
immobilization (spectrum by tightness of immobilization)
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0.5 1.5];
% figure(5);clf
% cfac_ctrl = mean(squeeze(mean(vc(:,:,s),1)),1);
% % calculating the control group
% for i = 1 : length(edges_locim)-1 % first grouped by loc
of immo
%     thisedge = edges_locim(i:i+1);
%     idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
%     thisrgclr = vp(idx_rgc, :, 204:205);
%     thisrgclr_mean = squeeze(mean(thisrgclr,1));
%     L = mean(thisrgclr_mean(:,1));
%     R = mean(thisrgclr_mean(:,2));
%     ind_2nd   = Int_im;
%     edges_2nd = edges_len;
%     idx = idx_rgc;
%     thisintimb = mean(ind_2nd(idx));
%     thiscfac  = vp(idx, :,s);
%     thiscfac_mean = squeeze(mean(thiscfac,1));
%     cfac_const    = mean(thiscfac_mean,1);
%     h = {};
%     text_edge2 = {};
%     subplot(2,ceil((length(edges_locim)-1)/2),i)
%     %     imagesc(s2, t2, thiscfac_mean)
%     hold on
%     % plot the loc of immobiliaztion
%     line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     if ~isempty(thiscfac)
%         tmph = plot(s, cfac_const./ cfac_ctrl,
'LineWidth', 2);
%         h = [h, tmph];
%         tmpt = sprintf('%.1f(%.0d)', thisintimb,
sum(idx));
%         text_edge2 = [text_edge2, tmpt];
%     end
%     hold off
%     xlim([5 95])
%     ylim(intlim)
%     xlabel('body coordinate')
%     ylabel('ratio of amplitude (gfac)')
%     title(sprintf('immob @ %.0f to %.0f%%', L, R))
%     set(gca, 'FontSize', 15)
%     legend(h, text_edge2)
% end
% set(gcf, 'Position', [581,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'gfac_norm_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'gfac_norm_locim_tightness.png'));
% end

% % absK dependence on location of immobilization (spectrum
by tightness of immobilization)
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0 10];
% figure(6);clf
% k_ctrl = mean(squeeze(mean(abs(vc(:,:,s2)),1)),1);
% % calculating the control group
% for i = 1 : length(edges_locim)-1 % first grouped by loc
of immo
%     thisedge = edges_locim(i:i+1);
%     idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
%     thisrgclr = vp(idx_rgc, :, 204:205);
%     thisrgclr_mean = squeeze(mean(thisrgclr,1));
%     L = mean(thisrgclr_mean(:,1));
%     R = mean(thisrgclr_mean(:,2));
%     ind_2nd   = Int_im;
%     edges_2nd = edges_len;
%     idx = idx_rgc;
%     thisintimb = mean(ind_2nd(idx));
%     thisk  = abs(vp(idx, :,s2));
%     thisk_mean = squeeze(mean(thisk,1));
%     h = {};
%     text_edge2 = {};
%     subplot(2,ceil((length(edges_locim)-1)/2),i)
%     %     imagesc(s2, t2, thiscfac_mean)
%     hold on
%     % plot the control group
%     tmph = plot(s, k_ctrl, 'LineWidth', 2, 'Color', 'k');
%     h = [h, tmph];
%     tmpt = sprintf('ctrl');
%     text_edge2 = [text_edge2, tmpt];
%     % plot the loc of immobiliaztion
%     line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     if ~isempty(thisk)
%         tmph = plot(s, mean(thisk_mean,1), 'LineWidth',
2, 'Color', 'r');
%         h = [h, tmph];
%         tmpt = sprintf('%.1f(%.0d)', thisintimb,
sum(idx));
%         text_edge2 = [text_edge2, tmpt];
%     end
%     hold off
%     xlim([5 95])
%     xlabel('body coordinate')
%     ylabel('normalized amplitude (|K|)')
%     title(sprintf('immob @ %.0f to %.0f%%', L, R))
%     set(gca, 'FontSize', 15)
%     legend(h, text_edge2)
% end
% set(gcf, 'Position', [581,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'absK_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'absK_locim_tightness.png'));
% end

% % absK const/ctrl ratio dependence on location of
immobilization
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0.5 1.5];
% figure(7);clf
% k_ctrl = mean(squeeze(mean(abs(vc(:,:,s2)),1)),1);
% % calculating the control group
% for i = 1 : length(edges_locim)-1 % first grouped by loc
of immo
%     thisedge = edges_locim(i:i+1);
%     idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
%     thisrgclr = vp(idx_rgc, :, 204:205);
%     thisrgclr_mean = squeeze(mean(thisrgclr,1));
%     L = mean(thisrgclr_mean(:,1));
%     R = mean(thisrgclr_mean(:,2));
%     ind_2nd   = Int_im;
%     edges_2nd = edges_len;
%     idx = idx_rgc;
%     thisintimb = mean(ind_2nd(idx));
%     thisk  = abs(vp(idx, :,s2));
%     thisk_mean = squeeze(mean(thisk,1));
%     h = {};
%     text_edge2 = {};
%     subplot(2,ceil((length(edges_locim)-1)/2),i)
%     %     imagesc(s2, t2, thiscfac_mean)
%     hold on
%     % plot the loc of immobiliaztion
```

```
%     line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
%     if ~isempty(thisk)
%         tmph = plot(s, mean(thisk_mean,1)./k_ctrl,
'LineWidth', 2, 'Color', 'r');
%         h = [h, tmph];
%         tmpt = sprintf('%.1f(%.0d)', thisintimb,
sum(idx));
%         text_edge2 = [text_edge2, tmpt];
%     end
%     hold off
%     xlim([5 95])
%     ylim(intlim)
%     xlabel('body coordinate')
%     ylabel('ratio of amplitude (|K|)')
%     title(sprintf('immob @ %.0f to %.0f%%', L, R))
%     set(gca, 'FontSize', 15)
%     legend(h, text_edge2)
% end
% set(gcf, 'Position', [581,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'absKratio_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'absKratio_locim_tightness.png'));
% end


% absK const/ctrl ratio scatter plots and bar plots
s2 = 101:200;
s  = 1 : 100;
intlim = [0.5 1.5];
absk_ctrl = mean(squeeze(mean(abs(vc(:,:,s2)),1)),1);
% calculating the control group
thisedge = [35 65];
idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
num_trials = sum(idx_rgc);
thisconstlr_indmean = squeeze(mean(vp(idx_rgc, :,
204:205),2));
thisabsk_indmean  = squeeze(mean(abs(vp(idx_rgc, :,
s2)),2));
thisabsk_norm_indmean = thisabsk_indmean./repmat(absk_ctrl,
[num_trials,1]);
q_anterior  = zeros(num_trials,1);
q_immobile  = zeros(num_trials,1);
q_posterior = zeros(num_trials,1);
for i = 1 : num_trials
    immA = thisconstlr_indmean(i, 1);
    immP = thisconstlr_indmean(i, 2);
    absk_norm = thisabsk_norm_indmean(i, :);
    q_anterior(i) = mean(absk_norm(15:floor(immA)));
    q_immobile(i) =
mean(absk_norm(ceil(immA):floor(immP)));
    q_posterior(i) = mean(absk_norm(ceil(immP) : 85));
end
% Excluding the trials that were not successfully
immobilized
todelete = q_immobile>=1.2;
q_anterior(todelete) = [];
q_immobile(todelete) = [];
q_posterior(todelete)= [];
num_minitrials = numel(q_immobile);
%%%
num_trials = size(q_anterior,1);
% scatter plots
% anterior vs middle
figure(8); clf
scatter(q_immobile, q_anterior, 36, 'k', 'filled')
xlabel('Middle amplitude (|K|)')
ylabel('Anterior amplitude (|K|)')
set(gcf, 'Position', [1200,160,570,490])
if issave == 1
    saveas(gcf, fullfile(savefoldername, 'norm amplitude
(absk) scatter a vs m.fig'));
    saveas(gcf, fullfile(savefoldername, 'norm amplitude
(absk) scatter a vs m.png'));
end
% bar plots
x = [1*ones([num_trials 1]); 2*ones([num_trials 1]);
3*ones([num_trials 1])]; % anterior, middle, posterior
y = [q_anterior; q_immobile; q_posterior]; % anterior,
middle, posterior
figure(9); clf
beeswarm(x,y,'sort_style','hex','dot_size',.5,'overlay_styl
e','ci','corral_style','gutter');
xlim([0.3 3.7])
xticks([1 2 3])
xticklabels({'Anterior', 'Middle', 'Posterior'})
ylabel('Normalized bending amplitude')
set(gcf,'Position', [69,291,700,400])
set(gca,'FontName','Arial','FontSize',20)
if issave == 1
    saveas(gcf, fullfile(savefoldername, 'norm amplitude
(absk) bar.fig'));
    saveas(gcf, fullfile(savefoldername, 'norm amplitude
(absk) bar.png'));
end

% % Gfactor const/ctrl ratio scatter plots and bar plots
% s2 = 101:200;
% s1 = 1 : 100;
% intlim = [0.5 1.5];
% gfac_ctrl = mean(squeeze(mean(vc(:,:,s1),1)),1);
% % calculating the control group
% thisedge = [35 65];
% idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
% num_trials = sum(idx_rgc);
% thisconstlr_indmean = squeeze(mean(vp(idx_rgc, :,
204:205),2));
% thisgfac_indmean    = squeeze(mean(vp(idx_rgc, :,
s1),2));
% thisgfac_norm_indmean =
thisgfac_indmean./repmat(gfac_ctrl, [num_trials,1]);
% q_anterior  = zeros(num_trials,1);
% q_immobile  = zeros(num_trials,1);
% q_posterior = zeros(num_trials,1);
% for i = 1 : num_trials
%     immA = thisconstlr_indmean(i, 1);
%     immP = thisconstlr_indmean(i, 2);
%     gfac_norm = thisgfac_norm_indmean(i, :);
%     q_anterior(i) = mean(gfac_norm(12:floor(immA)));
%     q_immobile(i) =
mean(gfac_norm(ceil(immA):floor(immP)));
%     q_posterior(i) = mean(gfac_norm(ceil(immP) : 90));
% end
% % scatter plots
% % anterior vs middle
% figure(10); clf
% scatter(q_immobile, q_anterior, 36, 'k', 'filled')
% xlabel('Middle amplitude (gfac)')
% ylabel('Anterior amplitude (gfac)')
% set(gcf, 'Position', [1200,160,570,490])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername, 'norm gfac
amplitude (gfac) scatter a vs m.fig'));
%     saveas(gcf, fullfile(savefoldername, 'norm gfac
amplitude (gfac) scatter a vs m.png'));
% end

% % bar plots
% x = {1 + randn([num_trials 1])*.1, 2+randn([num_trials
1])*.1, 3+randn([num_trials 1])*.1}; % anterior, middle,
posterior
% y = {q_anterior, q_immobile, q_posterior}; % anterior,
middle, posterior
% F_bar = zeros(size(y));
% X_bar = [1 2 3];
% F_err = zeros(size(y));
% figure(11); clf
% for i = 1:numel(x)
%     X = x{i};
%     Y = y{i};
%     F_bar(i) = mean(Y);
%     F_err(i) = std(Y)./sqrt(numel(X));
%     hold on
%     scatter(X, Y,10, 'b','filled')
%     hold off
% end
% figure(11); hold on
% bar(X_bar, F_bar, 'FaceColor', 'none', 'LineWidth', 2)
% errorbar(X_bar, F_bar, F_err, 'LineStyle', 'none',
'Color', 'k', 'LineWidth',1.5)
% hold off
% ylim([0 3])
% xticks([1 2 3])
% xticklabels({'Anterior', 'Middle', 'Posterior'})
% ylabel('Normalized amplitude (gfac)')
% set(gcf, 'Position', [1200,160,570,490])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername, 'norm amplitude
(gfac) bar.fig'));
%     saveas(gcf, fullfile(savefoldername, 'norm amplitude
(gfac) bar.png'));
% end

% % absK const/ctrl ratio dependence on location of
immobilization
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0 4];
% figure(12);clf
% absk_ctrl = mean(squeeze(mean(abs(vc(:,:,s2)),1)),1);
```

```
% % calculating the control group
% thisedge = [40 60];
% idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
% thisrgclr = vp(idx_rgc, :, 204:205);
% thisrgclr_mean = squeeze(mean(thisrgclr,1));
% L = mean(thisrgclr_mean(:,1));
% R = mean(thisrgclr_mean(:,2));
% ind_2nd   = Int_im;
% idx = idx_rgc;
% thisintimb = mean(ind_2nd(idx));
% thisabsk = squeeze(mean(abs(vp(idx, :,s2)),2));
% num_trials = size(thisabsk,1);
% thisabsk_norm = thisabsk./repmat(absk_ctrl,
[num_trials,1]);
% % calculating the moving average and variance for norm
amplitude ratio
% thisabsk_mean = mean(thisabsk_norm,1);
% thisabsk_var  = std(thisabsk_norm,0,1)./sqrt(num_trials);
% ts = tinv([0.025  0.975],num_trials-1); % T-Score
% ts = mean(abs(ts));
% CI = ts.*thisabsk_var;
% hold on
% for j = 1 : num_trials
%     plot(thisabsk_norm(j, :), ':',
'LineWidth',.1,'Color', [.3 .3 .3])
% end
% shadedErrorBar(s, thisabsk_mean,CI,'b', .3); hold on;
% % plot the loc of immobiliaztion
% line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
% line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
% line([0 100], [1 1], 'Color', 'r')
% tmpt = sprintf('%.1f(%.0d)', thisintimb, sum(idx));
% hold off
% xlim([5 95])
% ylim(intlim)
% xlabel('body coordinate')
% ylabel('ratio of amplitude (|K|)')
% title(sprintf('immob @ %.0f to %.0f%%', L, R))
% legend(gca, tmpt)
% set(gca, 'FontSize', 15)
% set(gcf, 'Position', [581,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'abskratio_shaded_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'abskratio_shaded_locim_tightness.png'));
% end

% % gfac const/ctrl ratio dependence on location of
immobilization
% s2 = 101:200;
% s  = 1 : 100;
% intlim = [0 4];
% figure(13);clf
% gfac_ctrl = mean(squeeze(mean(vc(:,:,s),1)),1);
% % calculating the control group
% thisedge = [40 60];
% idx_rgc  = Loc_im>=thisedge(1) & Loc_im<=thisedge(2)&
Int_im >= tightness ;
% thisrgclr = vp(idx_rgc, :, 204:205);
% thisrgclr_mean = squeeze(mean(thisrgclr,1));
% L = mean(thisrgclr_mean(:,1));
% R = mean(thisrgclr_mean(:,2));
% ind_2nd   = Int_im;
% idx = idx_rgc;
% thisintimb = mean(ind_2nd(idx));
% thisgfac = squeeze(mean(vp(idx, :,s),2));
% num_trials = size(thisgfac,1);
% thisgfac_norm = thisgfac./repmat(gfac_ctrl,
[num_trials,1]);
% % calculating the moving average and variance for norm
amplitude ratio
% thisgfac_mean = mean(thisgfac_norm,1);
% thisgfac_var  = std(thisgfac_norm,0,1)./sqrt(num_trials);
% ts = tinv([0.025  0.975],num_trials-1); % T-Score
% ts = mean(abs(ts));
% CI = ts.*thisgfac_var;
% hold on
% for j = 1 : num_trials
%     plot(thisgfac_norm(j, :), ':',
'LineWidth',.1,'Color', [.3 .3 .3])
% end
% shadedErrorBar(s, thisgfac_mean,CI,'b', .3); hold on;
% % plot the loc of immobiliaztion
% line([L L],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
% line([R R],intlim,'Color','red','LineStyle','--',
'LineWidth',2)
% line([0 100], [1 1], 'Color', 'r')
% tmpt = sprintf('%.1f(%.0d)', thisintimb, sum(idx));
% hold off
% xlim([5 95])
% ylim(intlim)
% xlabel('body coordinate')
% ylabel('ratio of amplitude (gfac)')
% title(sprintf('immob @ %.0f to %.0f%%', L, R))
% legend(gca, tmpt)
% set(gca, 'FontSize', 15)
% set(gcf, 'Position', [581,41,849,740])
% if issave == 1
%     saveas(gcf, fullfile(savefoldername,
'gfacratio_shaded_locim_tightness.fig'));
%     saveas(gcf, fullfile(savefoldername,
'gfacratio_shaded_locim_tightness.png'));
% end
```

PostProcessing_cfactor_statistic_auto.m

```
function [Q_anterior, Q_immobile, Q_posterior] =
PostProcessing_cfactor_statistic_auto(ctrlfilename,
pathname, t1)
% locate the folder
dir_list = dir(fullfile(pathname, '*data*.mat'));
nworms   = numel(dir_list);
t0   = t1; % sampling period
t_step = t0; % step of moving during sampling
v    = [];% preallocate for data pool after sampling with
period t0. dim(v) = N * i0 * 403
prog = 0;
%% Control group
fprintf('------Progress: %3.0f%% \n',prog);
% load and resampling the data
load(fullfile(pathname, ctrlfilename))
nnormw = numel(CURV_all);
do_flip = 0;
for i = 1 : nnormw
    prog = 100*i/nnormw;
    if i > 1
        fprintf('\b\b\b\b%3.0f%%',prog);
    end

    i0 = floor(t0*fps);
    step = floor(t_step*fps);
    K    = CURV_all{i};
    dKdt = gradient(K')'*fps;
    s    = 1 : size(K, 2);
    if do_flip == 3
        %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
        % debug plot
        figure(10); clf
        imagesc(K)
        colormap(cmap_redblue(0.7))
        caxis([-25 25])
        colorbar
        set(gcf, 'Position', [581,42,584,1314])
        hold on
        answer = length(questdlg('Need to flip some
period?', '','Yes', 'No', 'No'));
        if answer == 3
            title('Indicate the period that need to be
flipped')
            % flip curvature and dKdt
            [~, flpy1] = ginput(1);
            flpy1 = max([floor(flpy1) 1]);
            line([1 100], [flpy1
flpy1],'Color','white','LineStyle','--')
            [~, flpy2] = ginput(1);
            flpy2 = min([floor(flpy2) size(K,1)]);
            line([1 100], [flpy2
flpy2],'Color','white','LineStyle','--')
            K2flip = K(flpy1 : flpy2, :);
            K_flipped = flip(K2flip,2);
            K(flpy1 : flpy2, :) = K_flipped;
        end
        dKdt = gradient(K')'*fps;
        %
    end
        % recalculate the generalized compensatory factor
and save it
        numsamplepts = 100;
        numcurvpts   = 100;
        a  = .15; c = a * T0_avg;
        Zc = Kc_all + 1i*c*dKdtc_all;
        Pc = unwrap(angle(Zc), [], 2);
        [~, Sc] = meshgrid(1:numsamplepts, 1:numcurvpts);
        % Generate interpolant (in a bulk manner)
        FR = scatteredInterpolant(Pc(:), Sc(:), Zc(:),
'linear', 'nearest');
        % Constrcting complex curvature dynamics for pulsed
group
        Zp     = K' + 1i*c*dKdt';
        Pp_ori = angle(Zp); % do not use unwrap
        Ppld   = Pp_ori(:);
```

```
        Ppld(Ppld>0) = Ppld(Ppld>0) - 2*pi;
        Pp = reshape(Ppld, size(Pp_ori));
        [~, Sp] = meshgrid(1:size(Pp,2), 1:size(Pp,1));
        Rp = abs(Zp);
        Zc4pld = FR(Pp(:), Sp(:));
        Zc4p = reshape(Zc4pld, size(Pp));
        Rc4p = abs(Zc4p);
        cfac = (Rp) ./ Rc4p;
        % updata data
        CURV_all{i}  = K;
        dCURV_all{i} = dKdt;
        % show updated plot
        if do_flip == 3
            figure(10); clf
            imagesc(K)
            colormap(cmap_redblue(0.7))
            colorbar
            pause(1)
        end
        %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
    % convert immobilization position from boolean exp to
coord exp
    u  = cfac';
    curv = K;
    n  = size(u, 1);
    iwin = 1 : i0;
    while iwin(end) <= n
        thisu = u(iwin, :);
        thisk = curv(iwin, :);
        thiscfac = thisu(:, 1:100);
        iwin = iwin + step;
        if max(thiscfac,[],'all') > 20
            continue
        end
        thisuk = cat(2, thisu, thisk);
        v    = cat(3, v, thisuk);
    end
end
fprintf('\n')
save(fullfile(pathname, ctrlfilename),
'CURV_all','dCURV_all','-append')
vc = permute(v, [3 1 2]);
%% Constrained group
v = [];
%%%
wconst = 60; % normally, the width of the channel is 60 um
%%%
prog = 0;
fprintf('------Progress: %3.0f%% \n',prog);
do_flip = 0;
% load and resampling the data
load(fullfile(pathname, ctrlfilename))
for i = 1 : nworms
    prog = 100*i/nworms;
    if i > 1
        fprintf('\b\b\b\b%3.0f%%',prog);
    end
    thisworm  = dir_list(i);
    fname = fullfile(thisworm.folder, thisworm.name);
    load(fname)
    i0 = floor(t0*fps);
    step = floor(t_step*fps);
    ntrials = numel(CR);
    for j = 1:ntrials
        cfac = CR{j}';
        K    = Kp_data{j}';
        dKdt = dKdtp_data{j}';
        len  = len_const{j};
        rgc  = rgn_const{j};
        wdia = w_diam{j}*1000; % unit: um
        s =  1 : size(cfac, 2);
        t = (0 : size(cfac, 1)-1)'/fps;
        if do_flip == 3
            %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
            % debug plot
            figure(10); clf
            imagesc(K)
            colormap(cmap_redblue(0.7))
            caxis([-25 25])
            colorbar
            set(gcf, 'Position', [581,42,584,1314])
            hold on
            answer = length(questdlg('Need to flip some
period?', '','Yes', 'No', 'No'));
            if answer == 3
                title('Indicate the period that need to be
flipped')
                % flip curvature and dKdt
                [~, flpy1] = ginput(1);
                flpy1 = max([floor(flpy1) 1]);
                line([1 100], [flpy1
flpy1],'Color','white','LineStyle','--')
                [~, flpy2] = ginput(1);
                flpy2 = min([floor(flpy2) size(K,1)]);
                line([1 100], [flpy2
flpy2],'Color','white','LineStyle','--')
                K2flip = K(flpy1 : flpy2, :);
                K_flipped = flip(K2flip,2);
                K(flpy1 : flpy2, :) = K_flipped;
                dK2flip = dKdt(flpy1 : flpy2, :);
                dK_flipped = flip(dK2flip,2);
                dKdt(flpy1 : flpy2, :) = dK_flipped;
                r2flip = rgc(flpy1 : flpy2, :);
                r_flipped = flip(r2flip,2);
                rgc(flpy1 : flpy2, :) = r_flipped;
                %
                % recalculate the generalized compensatory
factor and save it
                numsamplepts = 100;
                numcurvpts   = 100;
                a  = .15; c = a * T0_avg;
                Zc = Kc_all + 1i*c*dKdtc_all;
                Pc = unwrap(angle(Zc), [], 2);
                [~, Sc] = meshgrid(1:numsamplepts,
1:numcurvpts);
                % Generate interpolant (in a bulk manner)
                FR = scatteredInterpolant(Pc(:), Sc(:),
Zc(:), 'linear', 'nearest');
                % Constrcting complex curvature dynamics
for pulsed group
                Zp     = K' + 1i*c*dKdt';
                Pp_ori = angle(Zp); % do not use unwrap
                Ppld   = Pp_ori(:);
                Ppld(Ppld>0) = Ppld(Ppld>0) - 2*pi;
                Pp = reshape(Ppld, size(Pp_ori));
                [~, Sp] = meshgrid(1:size(Pp,2),
1:size(Pp,1));
                Rp = abs(Zp);
                Zc4pld = FR(Pp(:), Sp(:));
                Zc4p = reshape(Zc4pld, size(Pp));
                Rc4p = abs(Zc4p);
                cr_new = (Rp) ./ Rc4p;
                % updata data
                CR{j} = cr_new;
                Kp_data{j} = K';
                dKdtp_data{j} = dKdt';
                rgn_const{j}  = rgc;
                % show updated plot
                figure(10); clf
                subplot(121)
                imagesc(K)
                colormap(cmap_redblue(0.7))
                colorbar
                subplot(122)
                imagesc(cr_new')
                colorbar
                pause(1)
            end
            %%%%%%%%%%%%%%%%%%%%%%%% disable this once
corrected all data
        end
        % convert immobilization position from boolean exp
to coord exp
        rgc(rgc == 0) = nan;
        imb = rgc .* repmat(s, [length(t), 1]);
        imbl = min(imb, [], 2, 'omitnan');
        imbr = max(imb, [], 2, 'omitnan');
        imbm = mean(imb, 2, 'omitnan');
        u  = cat(2, cfac, K, repmat([wdia, wconst],
[length(t) 1]), len, imbl, imbr, imbm);
        n  = size(u, 1);
        iwin = 1 : i0;
        while iwin(end) <= n
            thisu = u(iwin, :);
            thiscfac = thisu(:, 1:100);
            iwin = iwin + step;
            if max(thiscfac,[],'all') > 50
                continue
            end
            imbaa = mean(thisu(:, 204), 'omitnan'); %
average anterior
            imbpa = mean(thisu(:, 205), 'omitnan'); %
average posterior
            if isnan(imbaa) || isnan(imbpa)
                continue
            end
            tmp = cat(2, thisu, repmat(imbaa, [i0, 1]),
repmat(imbpa, [i0, 1]));
            v   = cat(3, v, tmp);
        end
    end
```

```
    save(fname,
'CR','Kp_data','dKdtp_data','rgn_const','len_const' ,'-
append')
end
fprintf('\n')
vp = permute(v, [3 1 2]);
%% Computing results
N       = size(vp, 1);
Loc_ima = vp(:,1,207);
Loc_imp = vp(:,1,208);

Int_im = vp(:,1,201)./vp(:,1,202);
tightness = 0;

% absK const/ctrl ratio scatter plots and bar plots
s2 = 101:200;
s  = 1 : 100;
intlim = [0.5 1.5];
absk_ctrl = mean(squeeze(mean(abs(vc(:,:,s2)),1)),1);
% calculating the control group
thisedge = [35 65];
idx_rgc  = Loc_ima>=thisedge(1) & Loc_imp<=thisedge(2)&
Int_im >= tightness;
num_minitrials = sum(idx_rgc);
thisconstlr_indmean = squeeze(mean(vp(idx_rgc, :,
204:205),2, 'omitnan'));
thisabsk_indmean    = squeeze(mean(abs(vp(idx_rgc, :,
s2)),2, 'omitnan'));
thisabsk_norm_indmean = thisabsk_indmean./repmat(absk_ctrl,
[num_minitrials,1]);
T0 = t1; % counting 10 s as one trial
I0 = T0/t0;
q_anterior  = zeros(num_minitrials,1);
q_immobile  = zeros(num_minitrials,1);
q_posterior = zeros(num_minitrials,1);
for i = 1 : num_minitrials
    immA = thisconstlr_indmean(i, 1);
    immP = thisconstlr_indmean(i, 2);
    absk_norm = thisabsk_norm_indmean(i, :);
    q_anterior(i) = mean(absk_norm(15:thisedge(1)));
    q_immobile(i) =
mean(absk_norm(ceil(immA):floor(immP)));
    q_posterior(i) = mean(absk_norm(thisedge(2) : 85));
end
% % Excluding the trials that were not successfully
immobilized
% todelete = q_immobile>=1.0;
% q_anterior(todelete) = [];
% q_immobile(todelete) = [];
% q_posterior(todelete)= [];
% num_minitrials = numel(q_immobile);
% %%%
num_trials  = floor(num_minitrials/I0) + 1;
Q_anterior  = zeros(num_trials,1);
Q_immobile  = zeros(num_trials,1);
Q_posterior = zeros(num_trials,1);
for i = 1 : num_trials
    if i ~= num_trials
        tmprange = (1 + (i-1)*I0) : i*I0;
    else
        tmprange = (1 + (i-1)*I0) : num_minitrials;
    end
    Q_anterior(i) = mean(q_anterior(tmprange));
    Q_immobile(i) = mean(q_immobile(tmprange));
    Q_posterior(i) = mean(q_posterior(tmprange));
end
Q_anterior(Q_anterior>=2.5) = [];
```

Str2Mat.m

```
function D = Str2Mat(A)
%STR2MAT convert string to matrix
D = reshape(str2double(regexp(A,'\d*','match')),2,[])';

end
```

WORMSHAPE_MAINCALCULATION.m

```
function [curvdatafiltered, dKdt_data, angledatafiltered,
lendata, inconst, fullnewdirname, w_diam] =
WORMSHAPE_MAINCALCULATION(vidObj, options)
%WORMSHAPE_MAINCALCULATION Analyze the undulatory dynamics
of worms
conc          = options{1}; %
wormlabel     = options{2};
fps           = options{3};
pix_per_mm    = options{4}; %
wormthreshold = options{5};
thisperiod    = options{6};
decim         = options{7};
filsize       = options{8}; %
start_illum   = options{9}; %
end_illum     = options{10}; %
spline_p      = options{11};
domovie       = options{12};
initials      = options{13}; %
pathname      = options{14};
filename      = options{15};
do_const      = options{16};
issavefiles   = options{17};

expo = 0.7;
istart = thisperiod(1);
iend   = thisperiod(2);
skip = floor((iend-istart+1)/10);
resizefactor = 1;
invert_img   = 0;
decim_filter = ones(decim) / (decim^2);

if skip ==0
    skip = 1;
end
numframes  = iend - istart + 1;

numcurvpts = 100;

mov_size_multiplier = 1;
savefps = 30;
mov_quality = .9;
if issavefiles
    fullnewdirname =
fullfile(pathname,strrep(filename,'.avi',sprintf('_worm%d_%
d-%d',wormlabel,istart,iend)));
    mkdir(fullnewdirname);
end

if domovie   % MOV
    savefname=
strrep(filename,'.avi',sprintf('_%d-%d.mov',istart,iend));
    savepathfname = fullfile(fullnewdirname,savefname);
    mov_size_multiplier = 1;
    savefps = vidObj.FrameRate;
    mov_quality = 0.9;
end



%%%%%%%% PREVIEW IMAGES %%%%%%%%%%
%%% Mark the region of constraint
if do_const
    img = mean(read(vidObj,istart),3);
    img = imresize(img, resizefactor, 'bicubic');

    img = imfilter(img, decim_filter, 'same');
    img = img(1:decim:end,1:decim:end);

    figure(1);clf
    image(img);

    bkg = imread(strrep(filename, 'bkgsubtracted.avi',
'background.bmp'));
    figure(2); clf
    image(bkg); axis image
    [ysize, xsize] = size(bkg);
    hold on;
    title('background');
    text(10,20, 'select ROI: upper left then lower right',
'Color', 'white');
    [bkgx1, bkgy1] = ginput(1);
    bkgx1 = floor(bkgx1);
    bkgy1 = floor(bkgy1);
    bkgx1 = max([1,bkgx1]);
    bkgy1 = max([1,bkgy1]);
    bkgx1 = min([size(bkg,2),bkgx1]);
    bkgy1 = min([size(bkg,1),bkgy1]);
    plot([1 xsize], [bkgy1 bkgy1], '-r');
    plot([bkgx1 bkgx1], [1 ysize], '-r');
    [bkgx2, bkgy2] = ginput(1);
    bkgx2 = floor(bkgx2);
    bkgy2 = floor(bkgy2);

    % ADF EDIT: Make sure the crops are in-bounds
    bkgx2 = max([1,bkgx2]);
    bkgy2 = max([1,bkgy2]);
    bkgx2 = min([size(bkg,2),bkgx2]);
    bkgy2 = min([size(bkg,1),bkgy2]);

    plot([1 xsize], [bkgy2 bkgy2], '-r');
    plot([bkgx2 bkgx2], [1 ysize], '-r');

    xconst = [bkgx1, bkgx2, bkgx2, bkgx1]';
    yconst = [bkgy1, bkgy1, bkgy2, bkgy2]';
end
%%% Worm Analysis
j=0;
% manually remove bright none-worm objects
img = mean(read(vidObj,istart),3);
img = imresize(img, resizefactor, 'bicubic');
```

```
img = imfilter(img, decim_filter, 'same');
img = img(1:decim:end,1:decim:end);
lvl = min(min(img)) + wormthreshold* (-
min(min(img))+max(max(img)));
figure(1);clf;
imagesc(img> lvl); axis image;
figure(2);clf;
imagesc(img); colormap gray; axis image; hold on;
bw_remove = false(size(img));
iscontinue = 1;
while iscontinue == 1
    figure(2); hold on;
    title('Indicate none-worm ROI')
    answer = length(questdlg('Continue to remove none-
worm?', 'BKG modification', 'Continue', 'No', 'No'));
    if answer == 8
        iscontinue = 1;
        figure(2); hold on;
        [bw_nw, xi, yi] = roipoly;
        patch(xi, yi, 'g', 'FaceColor',
'none','EdgeColor','r', 'LineStyle',':');
        bw_remove = bw_remove | bw_nw;
    else
        iscontinue = 0;
        close all
    end
end
img(bw_remove) =  min(min(img));
% % manually select four pixel points from the background
to eliminate the
% % background noise
% figure(2); clf;
% imagesc(img); colormap gray; axis image; hold on;
% title('Pick four points as background pixels')
% [xs_bkg, ys_bkg] = ginput(4);
% plot(xs_bkg, ys_bkg, 'or')

% create sum image
for i=istart:skip:iend
    j = j+1;

    img = mean(read(vidObj,i),3);
    img = imresize(img, resizefactor, 'bicubic');

    img = imfilter(img, decim_filter, 'same');
    img = img(1:decim:end,1:decim:end);

    if invert_img
        img = 255-img;
    end
    if i == istart
        imgsum = single(img);
        [ysize, xsize] = size(img);
        imgmin = ones(size(img));
        imgdata = zeros(ysize, xsize,
length(istart:skip:iend));
    end
    figure(1);
    imagesc(img); colormap gray;hold on;
    axis image;    title(num2str(i));
    imgdata(:,:,j) = img;
    imgsum = imgsum + single(img);
end


figure(1);clf;
imagesc(imgsum); colormap jet; hold on;
title('sum image');

text(10,20, 'select ROI: upper left then lower right',
'Color', 'white');
[cropx1, cropy1] = ginput(1);
cropx1 = floor(cropx1);
cropy1 = floor(cropy1);

% ADF EDIT: Make sure the crops are in-bounds.
cropx1 = max([1,cropx1]);
cropy1 = max([1,cropy1]);
cropx1 = min([size(img,2),cropx1]);
cropy1 = min([size(img,1),cropy1]);

% Get the second corner of the ROI
plot([1 xsize], [cropy1 cropy1], '-r');
plot([cropx1 cropx1], [1 ysize], '-r');
[cropx2, cropy2 ] = ginput(1);
cropx2 = floor(cropx2);
cropy2 = floor(cropy2);

% ADF EDIT: Make sure the crops are in-bounds
cropx2 = max([1,cropx2]);
cropy2 = max([1,cropy2]);
cropx2 = min([size(img,2),cropx2]);
cropy2 = min([size(img,1),cropy2]);
plot([1 xsize], [cropy2 cropy2], '-r');
plot([cropx2 cropx2], [1 ysize], '-r');

%%%%%%%%%%%%%%%%%%%%%%%%%% MAIN
CALCULATIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% Parameter
Initiation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

showcalc = 0;
deinterlace = 1; interlaceframe = 1;
cropyes = 1;


curvdata = zeros(numframes,numcurvpts);
inconst  = zeros(numframes,numcurvpts);
areadata = zeros(numframes,1);
centroiddata =  zeros(numframes,2);

cv2i_data = zeros(numframes,numcurvpts+2,2);
angledata = zeros(numframes,numcurvpts+1);
path1_rescaled_data = zeros(numframes,numcurvpts,2);
path2_rescaled_data = zeros(numframes,numcurvpts,2);
corner_mean = zeros(numframes,1);
lendata = zeros(numframes, 1);

for j=1
    i = istart + (j - 1);
    img = mean(read(vidObj,i),3);%%%%%%%%%%%%%%%% changed
    img = imresize(img, resizefactor, 'bicubic');

    img = imfilter(img, decim_filter, 'same');
    img = img(1:decim:end,1:decim:end);
    if invert_img
        img = 255-img;
    end

    img2 = abs(single(img(:,:,1))- imgmin);
    img = abs(single(img(:,:,1)));

    if deinterlace
        img(3-interlaceframe:2:end) =
img(interlaceframe:2:end);
        img2(3-interlaceframe:2:end) =
img2(interlaceframe:2:end);
    end
    if cropyes
        imgcrop = img(cropy1:cropy2,cropx1:cropx2);%

        imgcrop2 = img2(cropy1:cropy2,cropx1:cropx2);%

    else
        imgcrop = img;
        imgcrop2 = img2;
    end
    imgcrop = imgcrop';
    imgcrop2 = imgcrop2';

    imgcrop3 = imgcrop - imgcrop2;
    imgcrop4 = imgcrop3 - min(min(imgcrop3));
    [a,c] = find (imgcrop4 > 40);
    contour = [a,c];
    figure(2); hold off;
    imagesc(imgcrop4, [5 250]); hold on;
    colormap gray
end

ddd1 = [];
vvv1 = [];

for j=1:numframes
    i = istart + (j - 1);

    if i>vidObj.NumberOfFrames; break; end
%
    img = mean(read(vidObj,i),3);
    img(bw_remove) =  min(min(img));
    img = imresize(img, resizefactor, 'bicubic');
    img = imfilter(img, decim_filter, 'same');
    img = img(1:decim:end,1:decim:end);
    if invert_img
        img = 255-img;
    end

    img = abs(single(img(:,:,1))- imgmin);
    img2 = abs(single(img(:,:,1)));

    if deinterlace
        img(3-interlaceframe:2:end) =
img(interlaceframe:2:end);
        img2(3-interlaceframe:2:end) =
img2(interlaceframe:2:end);
```

```
    end
    if cropyes
        imgcrop = img(cropy1:cropy2,cropx1:cropx2);
        imgcrop2 = img2(cropy1:cropy2,cropx1:cropx2);
    else
        imgcrop = img;
        imgcrop2 = img2;
    end
    imgcrop = imgcrop';
    imgcrop2 = imgcrop2';

    corner_mean(j) = mean(mean(imgcrop(1:20,1:20)));


    figure(1); hold off;
    imagesc(imgcrop, [5 250]); hold on;
    text(20,20,[num2str(1*double(i-istart)/fps, '%.2f') '
s'], 'Color', 'w');
    axis image;

    if j==1
        colormap jet;
        [ysize, xsize ] = size(imgcrop);
        text(10,10,'click on head', 'Color', 'white');
        [headx, heady] = ginput(1);
        hold on;
        plot(headx, heady, 'or');
        headx0 = headx; heady0 = heady;
        text(10,10,'click on head', 'Color', 'black');
        text(10,10,'click on tail', 'Color', 'white');
        [tailx, taily] = ginput(1);
        tailx0 = tailx; taily0 = taily;
        text(10,10,'click on tail', 'Color', 'black');
        lvl = min(min(imgcrop)) + wormthreshold* (-
min(min(imgcrop))+max(max(imgcrop)));

        figure(1);clf;
        imagesc(imgcrop> lvl); axis image; hold on;
        text(10,10,'zoom in, press any key', 'Color',
'white');
        zoom on; zoom off;
        text(10,10,'zoom in, press any key', 'Color',
'black');

        title('click two points separated by worm
diameter');
        tmp1 = ginput(1);
        plot(tmp1(1),tmp1(2), 'ow');
        tmp2 = ginput(1);
        plot(tmp2(1),tmp2(2), 'ow');
        pause(.5);
        worm_diam = norm(tmp1-tmp2);
        title(['worm diameter = ' num2str(worm_diam) '
pixels']);
        worm_area_est = 10*worm_diam^2;
        sizethresh = round(worm_area_est / 2);
        if mod(round(filsize*worm_diam),2)==1
            filradius = round(filsize*worm_diam/2);
        else
            filradius = round(filsize*worm_diam/2)+1;
        end

        fil = fspecial('disk', filradius);
        if domovie
            MakeQTMovie('start',savepathfname);
            MakeQTMovie('size',
mov_size_multiplier*[size(img,2) size(img,1)]);
            MakeQTMovie('quality', mov_quality);
            MakeQTMovie('framerate', round(savefps));
        end
        colormap gray;
        zoom out;
        set(gcf, 'Position',    [ 129   190   310   463]);


    end

    img2 = conv2(single(imgcrop), fil, 'same');

    lvl = min(min(img2))+wormthreshold* (-
min(min(img2))+max(max(img2)));

    bw =(img2> lvl);

    if showcalc
        figure(2);
        imshow(bw);
    end

    bw2 = bwareaopen(bw,  sizethresh);
    bw3 = imcomplement(bw2);
    bw4 = bwareaopen(bw3, sizethresh);
    bw5 = imcomplement(bw4);

    STATS = regionprops(logical(bw5),'Area', 'Centroid');

    if size(STATS,1) == 0
        disp('Error: no worm found');
        break;
    end

    areadata(j) = STATS.Area;
    centroiddata(j,:) = STATS.Centroid;
    B = bwboundaries(bw5, 'noholes'); %  trace boundary
clockwise

    B1 = B{1}; % boundary coordinates

    B1_size = size(B1,1);

    ksep = ceil(B1_size/20);

    B1_plus = circshift(B1,[ksep 0]);
    B1_minus = circshift(B1,[-ksep 0]);

    AA = B1 - B1_plus;  % AA and BB are vectors between a
point on boundary and neighbors +- ksep away
    BB = B1 - B1_minus;

    cAA = AA(:,1) + sqrt(-1)*AA(:,2);
    cBB = BB(:,1) + sqrt(-1)*BB(:,2);

    B1_angle = unwrap(angle(cBB ./ cAA));

    min1 = find(B1_angle == min(B1_angle),1); % find point
on boundary w/ minimum angle between AA, BB
    B1_angle2 = circshift(B1_angle, -min1);
    min2a = round(.25*B1_size)-
1+find(B1_angle2(round(.25*B1_size):round(0.75*B1_size))==m
in(B1_angle2(round(.25*B1_size):round(0.75*B1_size))),1);
% find minimum in other half
    min2 = 1+mod(min2a + min1-1, B1_size);

    tmp = circshift(B1, [1-min1 0]);
    end1 = 1+mod(min2 - min1-1, B1_size);

    path1 = tmp(1:end1,:);
    path2 = tmp(end:-1:end1,:);

    if norm(path1(1,:) - [heady headx]) > norm(path1(end,:)
- [heady headx]) % if min1 is at tail, reverse both paths
        tmp = path1;
        path1 = path2(end:-1:1,:);
        path2 = tmp(end:-1:1,:);
    end

    heady = path1(1,1);
    headx = path1(1,2);
    taily = path1(end,1);
    tailx = path1(end,2);

    path_length = numcurvpts;

    path1_rescaled = zeros(path_length,2);
    path2_rescaled = zeros(path_length,2);
    path1_rescaled2 = zeros(path_length,2);
    path2_rescaled2 = zeros(path_length,2);

    path1_rescaled(:,1) = interp1(0:size(path1,1)-1,
path1(:,1), (size(path1,1)-1)*(0:path_length-
1)/(path_length-1), 'linear');
    path1_rescaled(:,2) = interp1(0:size(path1,1)-1,
path1(:,2), (size(path1,1)-1)*(0:path_length-
1)/(path_length-1), 'linear');
    path2_rescaled(:,1) = interp1(0:size(path2,1)-1,
path2(:,1), (size(path2,1)-1)*(0:path_length-
1)/(path_length-1), 'linear');
    path2_rescaled(:,2) = interp1(0:size(path2,1)-1,
path2(:,2), (size(path2,1)-1)*(0:path_length-
1)/(path_length-1), 'linear');


    for kk=1:path_length
        tmp1 = repmat(path1_rescaled(kk,:),
[path_length,1]) - path2_rescaled;
        tmp2 = sqrt(tmp1(:,1).^2 + tmp1(:,2).^2);
        path2_rescaled2(kk,:) =
path2_rescaled(find(tmp2==min(tmp2),1),:);
    end

    for kk=1:path_length
        tmp1 = repmat(path2_rescaled(kk,:),
[path_length,1]) - path1_rescaled;
        tmp2 = sqrt(tmp1(:,1).^2 + tmp1(:,2).^2);
        path1_rescaled2(kk,:) =
path1_rescaled(find(tmp2==min(tmp2),1),:);
```

```
    end



    dorsalx = path1_rescaled2(:,1);
    dorsaly = path1_rescaled2(:,2);
    ventralx = path2_rescaled2(:,1);
    ventraly = path2_rescaled2(:,2);

    dorsal = [ventralx,ventraly];
    ventral = [dorsalx,dorsaly];

    dorsalline =  round(dorsal);
    ventralline = round(ventral);


    a2 =[];
    a3 =[];


    for i = 1:length(ventralline)
        a1 = find(ventralline(i,1) == contour(:,1) &
ventralline(i,2) == contour(:,2));
        a4 = find(dorsalline(i,1) == contour(:,1) &
dorsalline(i,2) == contour(:,2));
        if a1 > 0
            a1 = 1;
        else
            a1 = 0;
        end
        a2 = cat(2,a2,a1);

        if a4 > 0
            a4 = 1;
        else
            a4 = 0;
        end
        a3 = cat(2,a3,a4);

    end
    ddd = sum(a3);
    vvv = sum(a2);
    ddd1 = cat(1,ddd1,ddd);
    vvv1 = cat(1,vvv1,vvv);
    comb_cont = cat(1,ddd1',vvv1');

    weight_fn = ones(path_length,1);
    tmp=round(path_length*0.2);
    weight_fn(1:tmp)=(0:tmp-1)/tmp;
    weight_fn(end-tmp+1:end)=(tmp-1:-1:0)/tmp;
    weight_fn = [weight_fn weight_fn];

    midline   = 0.5*(path1_rescaled+path2_rescaled);
    midline2a = 0.5*(path1_rescaled+path2_rescaled2);
    midline2b = 0.5*(path1_rescaled2+path2_rescaled);
    midline_mixed = midline2a .* weight_fn + midline .* (1-
weight_fn);

    figure(1); axis image;
    plot(path1_rescaled(1,2),  path1_rescaled(1,1), 'or');
hold on;
    plot(path2_rescaled(end,2),path2_rescaled(end,1),
'og'); hold on;

    Line = midline_mixed;

    interpfactor = 10;
    Line2 = interpl(Line, (1:(1/interpfactor):100)); %
worm's center line in xy coordinates in imgcrop

    xy = circshift(Line2, [0 1])'; df = diff(xy,1,2);



    t = cumsum([0, sqrt([1 1]*(df.*df))]);
    cv = csaps(t,xy,spline_p);

    dorsal_xy = circshift(dorsalline, [0 1])'; df =
diff(dorsal_xy,1,2);
    tmpt2 = cumsum([0, sqrt([1 1]*(df.*df))]);
    dorsal_cv = csaps(tmpt2,dorsal_xy,spline_p);

    ventral_xy = circshift(ventralline, [0 1])'; df =
diff(ventral_xy,1,2);
    tmpt2 = cumsum([0, sqrt([1 1]*(df.*df))]);
    ventral_cv = csaps(tmpt2,ventral_xy,spline_p);


    figure(1); axis image;
    fnplt(cv, '-g'); hold on;
    plot(xy(1,51),xy(2,51),'or');hold on ;% centre point of
cv
    fnplt(dorsal_cv, '-r'); hold on;
    fnplt(ventral_cv, '-g'); hold on;

    drawnow;


    if domovie && j>1
        MakeQTMovie('addframe');
    end

    if j==1
        plot([Line(1,2) headx0],[Line(1,1) heady0], '-oc');
        plot([Line(end,2) tailx0],[Line(end,1) taily0], '-
oc');
        pause(1);
    end

    cv2 =  fnval(cv, t)';
    df2 = diff(cv2,1,1); df2p = df2';

    splen = cumsum([0, sqrt([1 1]*(df2p.*df2p))]);
    lendata(j) = splen(end)/pix_per_mm;
    % interpolate to equally spaced length units
    cv2i = interpl(splen+.00001*(0:length(splen)-1),cv2,
(0:(splen(end)-1)/(numcurvpts+1):(splen(end)-1)));
    if do_const
        xyi  = interpl(splen+.00001*(0:length(splen)-
1),cv2, (0:(splen(end)-1)/(numcurvpts-1):(splen(end)-1)));
        xcrop = xyi(:,2);
        ycrop = xyi(:,1);
        xincr = cropx1;
        yincr = cropy1;
        ximg = xcrop + xincr;
        yimg = ycrop + yincr;
        % identify the part that is in the constraint ROI
        in = inpolygon(ximg, yimg, xconst, yconst);
        inconst(j,:) = in;
    end
    % store cv2i data

    cv2i_data(j,:,:) = cv2i;
    path1_rescaled_data(j,:,:) = path1_rescaled;
    path2_rescaled_data(j,:,:) = path2_rescaled;
    df2 = diff(cv2i,1,1);
    atdf2 =  unwrap(atan2(-df2(:,2), df2(:,1)));
    curv = unwrap(diff(atdf2,1));
    curvdata(j,:) = curv' * pix_per_mm;
    % calculate the angle of attack during worm's
locomotion
    atdf2 = atan2(-df2(:,2), df2(:,1));
    theta   = (mean(max(atdf2)) + mean(min(atdf2)))/2;
    xcenter = cv2i(1,1);
    ycenter = cv2i(1,2);
    center  = repmat([xcenter ycenter], size(cv2i, 1), 1);
    Ro      = [cos(theta) -sin(theta); sin(theta)
cos(theta)];
    % do the rotation
    cv2io = (Ro*(cv2i' - center') + center')';
    df2o   = diff(cv2io,1,1);
    atdf2o = atan2(-df2o(:,2), df2o(:,1));
    angledata(j,:) = atdf2o';
end  % end main loop

% Post-processing raw curvature data
curvdata_median = medfilt2(curvdata, [5 5]);
tmp = reshape(curvdata_median, [numel(curvdata_median),1]);
curv05 = prctile(tmp, 5);
curv95 = prctile(tmp, 95);
curvdata(curvdata > curv95) = curvdata_median(curvdata >
curv95);
curvdata(curvdata < curv05) = curvdata_median(curvdata <
curv05);
timefilter = 5;
bodyfilter = 5;
curvfilter = fspecial('average',[timefilter,bodyfilter]);
curvdatafiltered = imfilter(curvdata, curvfilter,
'replicate');
dKdt_data        = gradient(curvdatafiltered')'*fps;
% Post-processing raw angle data
angledata_median = medfilt2(angledata, [5 5]);
tmp = reshape(angledata_median,
[numel(angledata_median),1]);
angle05 = prctile(tmp, 5);
angle95 = prctile(tmp, 95);
angledata(angledata > angle95) =
angledata_median(angledata > angle95);
angledata(angledata < angle05) = angledata_median(angledata
< angle05);
timefilter = 5;
bodyfilter = 5;
anglefilter = fspecial('average',[timefilter,bodyfilter]);
angledatafiltered = imfilter(angledata,  anglefilter ,
'replicate');
w_diam = worm_diam/pix_per_mm;
```

```
if issavefiles
    if domovie
        MakeQTMovie('finish');
    end
end

end
```

curv_vs_act.py

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import math
import os
import shutil
from scipy.ndimage import interpolation
from scipy.ndimage import median_filter
from scipy.ndimage import gaussian_filter
from scipy.interpolate import UnivariateSpline
from skimage.segmentation import active_contour
from scipy.stats import pearsonr
import pickle
import csv
from numpy.polynomial.polynomial import polyfit
from scipy import stats

## exec(open("./curv_vs_act.py").read())

input_folder = r"C:\Users\fffei\Dropbox\Paper\Compensatory
reponse mechanism\data optogenetics\GCaMP expts new\SWF331
p05 agarpad round5\Analyzables\w1\c2\r\edge_detection"

## write normalized images for viewing, use fold change for
gcamp, use real curvature (K*l)
with open(input_folder+'/'+'dorsal_kymograph.pkl','rb') as
f:
    dorsal_kymograph = pickle.load(f)
with open(input_folder+'/'+'ventral_kymograph.pkl','rb') as
f:
    ventral_kymograph = pickle.load(f)
with
open(input_folder+'/'+'dorsal_body_kymograph.pkl','rb') as
f:
    dorsal_body_kymograph = pickle.load(f)
with
open(input_folder+'/'+'ventral_body_kymograph.pkl','rb') as
f:
    ventral_body_kymograph = pickle.load(f)

## remove frames due to segmentation errors or whatever
# toRemove = 234
# dorsal_kymograph = dorsal_kymograph[:toRemove]
# ventral_kymograph = ventral_kymograph[:toRemove]
# dorsal_body_kymograph = dorsal_body_kymograph[:toRemove]
# ventral_body_kymograph =
ventral_body_kymograph[:toRemove]

# gcamp_bg = 2200
# dorsal_kymograph_pic = ((dorsal_kymograph-
np.min(dorsal_kymograph))/(np.max(dorsal_kymograph)-
np.min(dorsal_kymograph)))*255.0
# dorsal_kymograph_pic =
dorsal_kymograph_pic.astype('uint8')
# dorsal_kymograph = dorsal_kymograph - gcamp_bg
# temp =
np.histogram(dorsal_kymograph.flatten(),bins=np.arange(0,np
.max(dorsal_kymograph),10))
# temp = list(zip(temp[0],temp[1]))
# temp = list(sorted(temp))
# dorsal_kymograph = dorsal_kymograph / temp[-1][1]
# cv2.imwrite(input_folder+'/'+'dorsal_kymograph.tif',
dorsal_kymograph)
# ventral_kymograph_pic = ((ventral_kymograph-
np.min(ventral_kymograph))/(np.max(ventral_kymograph)-
np.min(ventral_kymograph)))*255.0
# ventral_kymograph_pic =
ventral_kymograph_pic.astype('uint8')
# ventral_kymograph = ventral_kymograph - gcamp_bg
# temp =
np.histogram(ventral_kymograph.flatten(),bins=np.arange(0,n
p.max(ventral_kymograph),10))
# temp = list(zip(temp[0],temp[1]))
# temp = list(sorted(temp))
# ventral_kymograph = ventral_kymograph / temp[-1][1]
# cv2.imwrite(input_folder+'/'+'ventral_kymograph.tif',
ventral_kymograph)
c_one = 9999999
c_two = 9999999
cutoff_one =
np.median(dorsal_body_kymograph.flatten()[dorsal_body_kymog
raph.flatten()!=0]) +
c_one*np.std(dorsal_body_kymograph.flatten()[dorsal_body_ky
mograph.flatten()!=0])
cutoff_two =
np.median(ventral_body_kymograph.flatten()[ventral_body_kym
ograph.flatten()!=0]) +
c_two*np.std(ventral_body_kymograph.flatten()[ventral_body_
kymograph.flatten()!=0])
dorsal_body_kymograph[dorsal_body_kymograph>cutoff_one] =
0.0
ventral_body_kymograph[ventral_body_kymograph>cutoff_two] =
0.0
body_min =
np.min([np.min(dorsal_body_kymograph),np.min(ventral_body_k
ymograph)])
body_max =
np.max([np.max(dorsal_body_kymograph),np.max(ventral_body_k
ymograph)])
dorsal_body_kymograph_pic = ((dorsal_body_kymograph -
body_min)/(body_max-body_min))*255.0
dorsal_body_kymograph_pic =
dorsal_body_kymograph_pic.astype('uint8')
ventral_body_kymograph_pic = ((ventral_body_kymograph -
body_min)/(body_max-body_min))*255.0
ventral_body_kymograph_pic =
ventral_body_kymograph_pic.astype('uint8')


## increase length of pic
# new_length = 1000
# dorsal_body_kymograph_pic =
np.array([dorsal_body_kymograph_pic[0] for i in
range(new_length)])
# ventral_body_kymograph_pic =
np.array([ventral_body_kymograph_pic[0] for i in
range(new_length)])

cv2.imwrite(input_folder+'/'+'dorsal_body_kymograph.tif',
dorsal_body_kymograph_pic)
cv2.imwrite(input_folder+'/'+'ventral_body_kymograph.tif',
ventral_body_kymograph_pic)

if False:
    # ## analysis
    dorsal_kymograph = dorsal_kymograph.astype('float')
    ventral_kymograph = ventral_kymograph.astype('float')
    dorsal_body_kymograph =
dorsal_body_kymograph.astype('float')
    ventral_body_kymograph =
ventral_body_kymograph.astype('float')

    ## adjust size so all are same size as
dorsal_body_kymograph
    ## dorsal_kymograph
    new_dorsal_kymograph = []
    for original_length, new_length in
zip(dorsal_kymograph.shape, dorsal_body_kymograph.shape):
        new_dorsal_kymograph.append(np.linspace(0,
original_length-1, new_length))
    coords = np.meshgrid(*new_dorsal_kymograph,
indexing='ij')
    dorsal_kymograph =
interpolation.map_coordinates(dorsal_kymograph, coords)
    new_dorsal_kymograph = None

    # ventral_kymograph
    new_ventral_kymograph = []
    for original_length, new_length in
zip(ventral_kymograph.shape, dorsal_body_kymograph.shape):
        new_ventral_kymograph.append(np.linspace(0,
original_length-1, new_length))
    coords = np.meshgrid(*new_ventral_kymograph,
indexing='ij')
    ventral_kymograph =
interpolation.map_coordinates(ventral_kymograph, coords)
    new_ventral_kymograph = None

    # ventral_body_kymograph
    new_ventral_body_kymograph = []
    for original_length, new_length in
zip(ventral_body_kymograph.shape,
dorsal_body_kymograph.shape):
        new_ventral_body_kymograph.append(np.linspace(0,
original_length-1, new_length))
    coords = np.meshgrid(*new_ventral_body_kymograph,
indexing='ij')
    ventral_body_kymograph =
interpolation.map_coordinates(ventral_body_kymograph,
coords)
    new_ventral_body_kymograph = None

    fig = plt.figure()
    ax = fig.add_subplot(111)
```

```
    x = dorsal_body_kymograph.flatten()-
ventral_body_kymograph.flatten()
    y = dorsal_kymograph.flatten()
    xy = list(sorted(list(zip(x,y))))
    x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
    y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
    b, m = polyfit(x, y, 1)
    r = pearsonr(x, y)
    ax.scatter(x,y,color='blue',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
    ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
    plt.legend(loc="upper right")
    ax.set_xlabel('Normalized dorsal curvature',
fontsize=14)
    ax.set_ylabel('Fold change dorsal activation',
fontsize=14)
    plt.savefig(input_folder+'/'+'dorsal.png')
    plt.show()
    plt.close()

    fig = plt.figure()
    ax = fig.add_subplot(111)
    x = dorsal_body_kymograph.flatten()-
ventral_body_kymograph.flatten()
    y = ventral_kymograph.flatten()
    xy = list(sorted(list(zip(x,y))))
    x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
    y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
    b, m = polyfit(x, y, 1)
    r = pearsonr(x, y)
    ax.scatter(x,y,color='red',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
    ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
    plt.legend(loc="upper right")
    ax.set_xlabel('Normalized ventral curvature',
fontsize=14)
    ax.set_ylabel('Fold change ventral activation',
fontsize=14)
    plt.savefig(input_folder+'/'+'ventral.png')
    plt.show()
    plt.close()

    fig = plt.figure()
    ax = fig.add_subplot(111)
    b, m = polyfit(dorsal_body_kymograph.flatten(),
ventral_body_kymograph.flatten(), 1)
    r = pearsonr(dorsal_body_kymograph.flatten(),
ventral_body_kymograph.flatten())

ax.scatter(dorsal_body_kymograph.flatten(),ventral_body_kym
ograph.flatten(),color='gray',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
    ax.plot(dorsal_body_kymograph.flatten(), b + m *
dorsal_body_kymograph.flatten(), '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
    plt.legend(loc="upper right")
    ax.set_xlabel('Normalized dorsal body curvature',
fontsize=14)
    ax.set_ylabel('Normalized ventral body curvature',
fontsize=14)
    plt.savefig(input_folder+'/'+'body.png')
    plt.show()
    plt.close()

    fig = plt.figure()
    ax = fig.add_subplot(111)
    b, m = polyfit(dorsal_kymograph.flatten(),
ventral_kymograph.flatten(), 1)
    r = pearsonr(dorsal_kymograph.flatten(),
ventral_kymograph.flatten())

ax.scatter(dorsal_kymograph.flatten(),ventral_kymograph.fla
tten(),color='gray',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
    ax.plot(dorsal_kymograph.flatten(), b + m *
dorsal_kymograph.flatten(), '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
    plt.legend(loc="upper right")
    # ax.set_xlim(-0.1,1.1)
    # ax.set_ylim(-0.1,1.1)
    ax.set_xlabel('Fold change dorsal muscle activation',
fontsize=14)
    ax.set_ylabel('Fold change ventral muscle activation',
fontsize=14)
    plt.savefig(input_folder+'/'+'muscle.png')
    plt.show()
    plt.close()

    ## get immobilization factor for body
    skipFactor = 10
    dbk = 0
    for i in range(skipFactor,len(dorsal_body_kymograph)):
        diff = dorsal_body_kymograph[i] -
dorsal_body_kymograph[i-skipFactor]
        dbk += np.sum(np.abs(diff))/len(diff)
    vbk = 0
    for i in range(1,len(ventral_body_kymograph)):
        diff = ventral_body_kymograph[i] -
ventral_body_kymograph[i-skipFactor]
        vbk += np.sum(np.abs(diff))/len(diff)

print('dorsal_body_kymograph',dbk/len(dorsal_body_kymograph
))

print('ventral_body_kymograph',vbk/len(ventral_body_kymogra
ph))

print('total_body_kymograph',((vbk/len(ventral_body_kymogra
ph))+(dbk/len(dorsal_body_kymograph)))/2)
    output =
[('dorsal_body_kymograph',dbk/len(dorsal_body_kymograph)),(
'ventral_body_kymograph',vbk/len(ventral_body_kymograph)),(
'total_body_kymograph',((vbk/len(ventral_body_kymograph))+(
dbk/len(dorsal_body_kymograph)))/2)]
    with open(input_folder+'/'+'immobility.csv', 'w') as
csv_file:
        writer = csv.writer(csv_file)
        for row in output:
            writer.writerow(row)

    ## get immobilization factor for muscles
    skipFactor = 10
    dk = 0
    for i in range(skipFactor,len(dorsal_kymograph)):
        diff = dorsal_kymograph[i] - dorsal_kymograph[i-
skipFactor]
        dk += np.sum(np.abs(diff))/len(diff)
    vk = 0
    for i in range(1,len(ventral_kymograph)):
        diff = ventral_kymograph[i] - ventral_kymograph[i-
skipFactor]
        vk += np.sum(np.abs(diff))/len(diff)
    print('dorsal_kymograph',dk/len(dorsal_kymograph))
    print('ventral_kymograph',vk/len(ventral_kymograph))

print('average_kymograph',((vk/len(ventral_kymograph))+(dk/
len(dorsal_body_kymograph)))/2)
    output =
[('dorsal_kymograph',dk/len(dorsal_kymograph)),('ventral_ky
mograph',vk/len(ventral_kymograph)),('average_kymograph',((
vk/len(ventral_kymograph))+(dk/len(dorsal_kymograph)))/2)]
    with open(input_folder+'/'+'immobility_muscles.csv',
'w') as csv_file:
        writer = csv.writer(csv_file)
        for row in output:
            writer.writerow(row)

# ## calculate derivative of body kymograph with respect to
time
# skipFactor = 10
# dbk_deriv = []
# for i in range(len(dorsal_body_kymograph)):
#     if i < skipFactor or i > len(dorsal_body_kymograph)-
skipFactor-1:
#
dbk_deriv.append(np.zeros(len(dorsal_body_kymograph[i])))
#     else:
#         dbk_deriv.append([])
#         for j in range(len(dorsal_body_kymograph[i])):
#             diff = dorsal_body_kymograph[i+skipFactor][j]
- dorsal_body_kymograph[i-skipFactor][j]
#             dbk_deriv[-1].append(diff)
#         dbk_deriv[-1] = np.array(dbk_deriv[-1])
# dbk_deriv = np.array(dbk_deriv)
# dbk_deriv_img = ((dbk_deriv/np.max(dbk_deriv))*255.0)
# cv2.imwrite(input_folder+'/'+'dbk_derivative.tif',
dbk_deriv_img.astype('uint8'))
#
# vbk_deriv = []
# for i in range(len(ventral_body_kymograph)):
#     if i < skipFactor or i > len(ventral_body_kymograph)-
skipFactor-1:
#
vbk_deriv.append(np.zeros(len(ventral_body_kymograph[i])))
#     else:
#         vbk_deriv.append([])
#         for j in range(len(ventral_body_kymograph[i])):
#             diff =
ventral_body_kymograph[i+skipFactor][j] -
ventral_body_kymograph[i-skipFactor][j]
```

```
#             vbk_deriv[-1].append(diff)
#         vbk_deriv[-1] = np.array(vbk_deriv[-1])
# vbk_deriv = np.array(vbk_deriv)
# vbk_deriv_img = ((vbk_deriv/np.max(vbk_deriv))*255.0)
# cv2.imwrite(input_folder+'/'+'vbk_derivative.tif',
vbk_deriv_img.astype('uint8'))
#
# ## calculate derivative of muscle kymograph with respect
to time
# skipFactor = 10
# dk_deriv = []
# for i in range(len(dorsal_kymograph)):
#     if i < skipFactor or i > len(dorsal_kymograph)-
skipFactor-1:
#
dk_deriv.append(np.zeros(len(dorsal_kymograph[i])))
#     else:
#         dk_deriv.append([])
#         for j in range(len(dorsal_kymograph[i])):
#             diff = dorsal_kymograph[i+skipFactor][j] -
dorsal_kymograph[i-skipFactor][j]
#             dk_deriv[-1].append(diff)
#         dk_deriv[-1] = np.array(dk_deriv[-1])
# dk_deriv = np.array(dk_deriv)
# dk_deriv_img = ((dk_deriv/np.max(dk_deriv))*255.0)
# cv2.imwrite(input_folder+'/'+'dk_derivative.tif',
dk_deriv_img.astype('uint8'))
#
# vk_deriv = []
# for i in range(len(ventral_kymograph)):
#     if i < skipFactor or i > len(ventral_kymograph)-
skipFactor-1:
#
vk_deriv.append(np.zeros(len(ventral_kymograph[i])))
#     else:
#         vk_deriv.append([])
#         for j in range(len(ventral_kymograph[i])):
#             diff = ventral_kymograph[i+skipFactor][j] -
ventral_kymograph[i-skipFactor][j]
#             vk_deriv[-1].append(diff)
#         vk_deriv[-1] = np.array(vk_deriv[-1])
# vk_deriv = np.array(vk_deriv)
# vk_deriv_img = ((vk_deriv/np.max(vk_deriv))*255.0)
# cv2.imwrite(input_folder+'/'+'vk_derivative.tif',
vk_deriv_img.astype('uint8'))
#
# fig = plt.figure()
# ax = fig.add_subplot(111)
# x = dorsal_body_kymograph.flatten()-
ventral_body_kymograph.flatten()
# y = dk_deriv.flatten()
# xy = list(sorted(list(zip(x,y))))
# x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
# y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
# b, m = polyfit(x, y, 1)
# r = pearsonr(x, y)
# ax.scatter(x,y,color='blue',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
# ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
# plt.legend(loc="upper right")
# ax.set_xlabel('Normalized dorsal curvature', fontsize=14)
# ax.set_ylabel('Change in fold change dorsal activation',
fontsize=14)
# plt.savefig(input_folder+'/'+'dorsal_deriv.png')
# plt.show()
# plt.close()
#
# fig = plt.figure()
# ax = fig.add_subplot(111)
# x = dorsal_body_kymograph.flatten()-
ventral_body_kymograph.flatten()
# y = vk_deriv.flatten()
# xy = list(sorted(list(zip(x,y))))
# x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
# y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
# b, m = polyfit(x, y, 1)
# r = pearsonr(x, y)
# ax.scatter(x,y,color='red',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
# ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
# plt.legend(loc="upper right")
# ax.set_xlabel('Normalized ventral curvature',
fontsize=14)
# ax.set_ylabel('Change in fold change ventral activation',
fontsize=14)
# plt.savefig(input_folder+'/'+'ventral_deriv.png')
# plt.show()
```

plt.close()
#
fig = plt.figure()
ax = fig.add_subplot(111)
x = dbk_deriv.flatten()
y = dk_deriv.flatten()
xy = list(sorted(list(zip(x,y))))
x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
b, m = polyfit(x, y, 1)
r = pearsonr(x, y)
ax.scatter(x,y,color='blue',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
# ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
# plt.legend(loc="upper right")
# ax.set_xlabel('Change in dorsal curvature', fontsize=14)
# ax.set_ylabel('Change in fold change dorsal activation',
fontsize=14)
# plt.savefig(input_folder+'/'+'dorsal_body_deriv.png')
# plt.show()
# plt.close()
#
fig = plt.figure()
ax = fig.add_subplot(111)
x = vbk_deriv.flatten()
y = vk_deriv.flatten()
xy = list(sorted(list(zip(x,y))))
x = np.array([xx for xx,yy in xy if xx > 0.02 or xx < -
0.02])
y = np.array([yy for xx,yy in xy if xx > 0.02 or xx < -
0.02])
b, m = polyfit(x, y, 1)
r = pearsonr(x, y)
ax.scatter(x,y,color='red',s=4, label='Pearsons R:
'+str('%.3f'%r[0]),alpha=0.05)
# ax.plot(x, b + m * x, '-',c='black',
label=str('%.3f'%m)+'x + '+str('%.3f'%b))
# plt.legend(loc="upper right")
# ax.set_xlabel('Change in ventral curvature', fontsize=14)
# ax.set_ylabel('Change in fold change ventral activation',
fontsize=14)
# plt.savefig(input_folder+'/'+'ventral_body_deriv.png')
# plt.show()
# plt.close()

worm_straightener.py

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import math
import os
import shutil
from scipy.ndimage import interpolation
from scipy.ndimage import median_filter
from scipy.interpolate import UnivariateSpline
from skimage.segmentation import active_contour
import pickle
import csv
from skimage import morphology, img_as_bool
import scipy.io

## exec(open("./worm_straightener.py").read())


red_input_folder =
r"C:\Users\fffei\Dropbox\Paper\Compensatory reponse
mechanism\data optogenetics\GCaMP expts new\SWF331 p05
agarpad round5\Analyzables\w11\c3\r"
green_input_folder =
r"C:\Users\fffei\Dropbox\Paper\Compensatory reponse
mechanism\data optogenetics\GCaMP expts new\SWF331 p05
agarpad round5\Analyzables\w11\c3\g"

try:
    shutil.rmtree(red_input_folder + '\\edge_detection')
except:
    pass

images = os.listdir(red_input_folder)
saveImages = True
draw = True
## make head = tail and tail = head corners
flipHeadTail = True
## make dorsal_contour = ventral_contour, vice versa
flipDorsalVentral = True
## reverse contours so they read from head to tail
reverseDorsalVentral = False
## flip direction of perpendicular angle during centerline
detection
flipAngle = True
```

```
## make sure curvature is correct
flipCurvature = False

os.mkdir(red_input_folder + '\\edge_detection')

def distance(p,q):
    return np.sqrt((p[0]-q[0])**2+(p[1]-q[1])**2)
def distance_pts(p,_contours):
    distance_lst = []
    for q in _contours[0]:
        distance_lst.append((distance(p[0],q[0]),q))
    distance_lst = list(sorted(distance_lst, key=lambda x:
x[0]))
    return distance_lst[0][1]

dorsal_kymograph = []
ventral_kymograph = []
dorsal_body_kymograph = []
ventral_body_kymograph = []
last_head_corner = np.array([None])
last_tail_corner = np.array([None])
last_dorsal_contour = np.array([None])
last_ventral_contour = np.array([None])
skipList = []
oneWormDone = False
midline_contours = []
center_of_masses = []

skip = -1

for image in images:
    # if image != '0092.tif':
    #     continue
    ## skip images if necessary and write previous image to
folder
    if image in skipList:

cv2.imwrite(red_input_folder+'/'+'edge_detection'+'/'+image
, img)
        continue

    skip+=1
    # if skip<443 or skip>495:
    #     continue

    ## read in image
    img = cv2.imread(red_input_folder+'/'+image,
cv2.IMREAD_UNCHANGED)
    img =
cv2.cvtColor(img,cv2.COLOR_GRAY2RGB).astype('uint8')
    print(image)

    ## filter to make smoother edges
    # dilation and erosion can be used to accentuate or
deaccentuate features
    ## kernel_one = 7, kernel_two = 3, 5 median blur, 0.1
threshold for 3.2 um images, use large median blur to round
edges
    kernel_one = np.ones((21,21), np.int16)
##    img = cv2.dilate(img, kernel_one, iterations=1)
##    img = cv2.erode(img, kernel_one, iterations=1)
    #plt.imshow(img),plt.show()
    ## second round of eroding so contour edges align with
pyo3 expression
    kernel_two = np.ones((5,5), np.int16)
    # img = cv2.erode(img, kernel_two, iterations=1)
    #plt.imshow(img),plt.show()
    ## clean all noise after dilatation and erosion
    medianBlur_one = 7
    img = cv2.medianBlur(img.astype('uint8'),
medianBlur_one)
    #plt.imshow(img),plt.show()
    ## threshold to make clean edges
    img = img.astype('uint8')
    # plt.imshow(img),plt.show()
    threshold_one = .1
    img[img < threshold_one] = 0
    img[img >= threshold_one] = 255
    img = img.astype('uint8')
    #plt.imshow(img),plt.show()

    #gray scale image
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    # Find Canny edges
    edged = cv2.Canny(gray, 0, 255)
    #plt.imshow(edged),plt.show()

    # Finding Contours
    # Use a copy of the image e.g. edged.copy()
    # since findContours alters the image
    contours, hierarchy = cv2.findContours(edged,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    if len(contours) != 1:
        print('more than one contour')

cv2.imwrite(red_input_folder+'/'+'edge_detection'+'/'+image
, img)
        skipList.append(image)
        midline_contours.append(np.array([]))
        center_of_masses.append([])
        continue

    ## corner selection using circle at each point on
contour
    ii = np.where(edged == 255)
    corners_params = [10,50]
    ii_r = corners_params[0]
    ii_sums = []
    for ii_y,ii_x in zip(ii[0],ii[1]):
        zero_img = np.zeros(edged.shape)
        cv2.circle(zero_img,(ii_x,ii_y),ii_r,1,-1)
        ii_circle = np.where(zero_img == 1)
        zero_img = zero_img*gray
        zero_img = np.sum(zero_img)
        ii_sums.append((zero_img,[ii_x,ii_y]))
    ii_sums = list(sorted(ii_sums))
    corners = [np.array([np.array(ii_sums[0][1])])]
    for zi,iixiiy in ii_sums:
        if distance(iixiiy,corners[0][0]) >
corners_params[1]:
            corners.append(np.array([np.array(iixiiy)]))
            break
    corners = np.array(corners)

    ## selection corner using cv2.goodfeaturestotrack
    # corners_params = [2,0.01,175,10]
    # corners =
cv2.goodFeaturesToTrack(gray,corners_params[0],corners_para
ms[1],corners_params[2],blockSize=corners_params[3])
    # corners = np.int0(corners)
    #

    ## split contour up into dorsal and ventral contours
    ## find contour position closest to corners
    if last_head_corner.any() == None:
        head = distance_pts(corners[0],contours)
        tail = distance_pts(corners[1],contours)
    else:
        if distance(last_head_corner[0],corners[0][0]) <
distance(last_head_corner[0],corners[1][0]):
            head = distance_pts(corners[0],contours)
            tail = distance_pts(corners[1],contours)
        else:
            head = distance_pts(corners[1],contours)
            tail = distance_pts(corners[0],contours)

    last_head_corner = head
    last_tail_corner = tail
    head_idx = None
    tail_idx = None

    ## break single contours into dorsal and ventral
contour
    for idx,var in enumerate(contours[0]):
        if var[0][0] == head[0][0] and var[0][1] ==
head[0][1]:
            head_idx = idx
        if var[0][0] == tail[0][0] and var[0][1] ==
tail[0][1]:
            tail_idx = idx
    if head_idx - tail_idx < 0:
        dorsal_contour =
np.array(list(reversed(list(contours[0][head_idx:tail_idx])
)))
        ventral_contour =
np.array(list(contours[0][tail_idx:]) +
list(contours[0][0:head_idx]))
    else:
        dorsal_contour =
np.array(list(contours[0][head_idx:]) +
list(contours[0][0:tail_idx]))
        ventral_contour =
np.array(list(reversed(list(contours[0][tail_idx:head_idx])
)))

    ## orient contours so they are consistently in the same
direction
    if last_dorsal_contour.any() != None:
        if
distance(last_dorsal_contour[0][0],dorsal_contour[0][0]) >
distance(last_dorsal_contour[-1][0],dorsal_contour[0][0]):
            dorsal_contour = dorsal_contour[::-1]
        if
distance(last_ventral_contour[0][0],ventral_contour[0][0])
```

```
> distance(last_ventral_contour[-
1][0],ventral_contour[0][0]):
            ventral_contour = ventral_contour[::-1]
    last_dorsal_contour = dorsal_contour
    last_ventral_contour = ventral_contour

    ## if necessary, flip head/tail and dorsal/ventral
    if flipHeadTail:
        h = head
        t = tail
        head = t
        tail = h
    if flipDorsalVentral:
        d = dorsal_contour
        v = ventral_contour
        dorsal_contour = v
        ventral_contour = d
    if reverseDorsalVentral:
        dorsal_contour = dorsal_contour[::-1]
        ventral_contour = ventral_contour[::-1]

    ## get body midline and calculate curvature
    ## make dorsal and ventral indieces same scale
    dorsal_contour_idx = list(range(len(dorsal_contour)))
    ventral_contour_idx = list(range(len(ventral_contour)))
    if len(dorsal_contour_idx) > len(ventral_contour_idx):
        new_dorsal_contour_idx = []
        for original_length, new_length in
zip(np.array(dorsal_contour_idx).shape,
np.array(ventral_contour_idx).shape):
            new_dorsal_contour_idx.append(np.linspace(0,
original_length-1, new_length))
        coords = np.meshgrid(*new_dorsal_contour_idx,
indexing='ij')
        dorsal_contour_idx =
interpolation.map_coordinates(dorsal_contour_idx, coords)
        new_dorsal_contour_idx = None
    else:
        new_ventral_contour_idx = []
        for original_length, new_length in
zip(np.array(ventral_contour_idx).shape,
np.array(dorsal_contour_idx).shape):
            new_ventral_contour_idx.append(np.linspace(0,
original_length-1, new_length))
        coords = np.meshgrid(*new_ventral_contour_idx,
indexing='ij')
        ventral_contour_idx =
interpolation.map_coordinates(ventral_contour_idx, coords)
        new_ventral_contour_idx = None

    ## calculate body midline contour using average half
distance of line scan across worm
    # skeleton =
morphology.medial_axis(img_as_bool(img[:,:,0]))
    skeleton = img.copy()
    skeleton[skeleton > 0] = 1
    skeleton = morphology.skeletonize(img[:,:,0]>0)
    # plt.imshow(skeleton),plt.show()
    skel_idx = np.argwhere(skeleton==True)
    midline_contour = []
    def dist(pt1,pt2):
        d = np.sqrt((pt2[1]-pt1[1])**2+(pt2[0]-pt1[0])**2)
        return d
    prev_pt = head[0]
    while len(skel_idx) != 0:
        temp = []
        for pt_idx in range(len(skel_idx)):
            pt = (skel_idx[pt_idx][1],skel_idx[pt_idx][0])
            pt_distance = dist(prev_pt,pt)
            temp.append((pt_distance,pt_idx))
        temp = list(sorted(temp))

midline_contour.append([[skel_idx[temp[0][1]][1],skel_idx[t
emp[0][1]][0]]])
        prev_pt =
[skel_idx[temp[0][1]][1],skel_idx[temp[0][1]][0]]
        skel_idx = np.delete(skel_idx,[temp[0][1]],axis=0)

    skip_one = 3
    # min_contour =
np.min((len(dorsal_contour),len(ventral_contour)))
    # for i in range(len(dorsal_contour_idx)):
    #     if i-skip_one >= 0 and i+skip_one < min_contour:
    #         d_i = dorsal_contour_idx[i-skip_one]
    #         d_i_1 = dorsal_contour_idx[i+skip_one]
    #         v_i = ventral_contour_idx[i-skip_one]
    #         v_i_1 = ventral_contour_idx[i+skip_one]
    #         d_x,d_y = dorsal_contour[d_i][0]
    #         d_x_1,d_y_1 = dorsal_contour[d_i_1][0]
    #         v_x,v_y = ventral_contour[v_i][0]
    #         v_x_1,v_y_1 = ventral_contour[v_i_1][0]
    #         ## calculate perpendicular angle
    #         if flipAngle:
    #             d_angle = math.atan2((d_y_1-d_y), (d_x_1-
d_x)) + math.radians(90)
    #             v_angle = math.atan2((v_y_1-v_y), (v_x_1-
v_x)) - math.radians(90)
    #         else:
    #             d_angle = math.atan2((d_y_1-d_y), (d_x_1-
d_x)) - math.radians(90)
    #             v_angle = math.atan2((v_y_1-v_y), (v_x_1-
v_x)) + math.radians(90)
    #         ## find halfway point for dorsal/ventral
contours
    #         d_point = dorsal_contour[d_i].astype('float')
    #         v_point =
ventral_contour[v_i].astype('float')
    #         onWorm = True
    #         d_point[0][0] =
d_point[0][0]+2.0*math.cos(d_angle)
    #         d_point[0][1] =
d_point[0][1]+2.0*math.sin(d_angle)
    #         v_point[0][0] =
v_point[0][0]+2.0*math.cos(v_angle)
    #         v_point[0][1] =
v_point[0][1]+2.0*math.sin(v_angle)
    #         dist = 0.25
    #         while onWorm:
    #             d_point[0][0] =
d_point[0][0]+dist*math.cos(d_angle)
    #             d_point[0][1] =
d_point[0][1]+dist*math.sin(d_angle)
    #             v_point[0][0] =
v_point[0][0]+dist*math.cos(v_angle)
    #             v_point[0][1] =
v_point[0][1]+dist*math.sin(v_angle)
    #             if
img[int(d_point[0][1])][int(d_point[0][0])].any() == 0:
    #                 d_x_new,d_y_new = d_point[0]
    #                 d_x_mid = int((d_x+d_x_new)/2)
    #                 d_y_mid = int((d_y+d_y_new)/2)
    #
midline_contour.append([[d_x_mid,d_y_mid]])
    #                 onWorm = False

    midline_contour = np.array(midline_contour)

    ## calculate normalized curvature
    Kl = []
    l = 0.0
    skip_two = 20
    ## calculate curvature
    for i in range(len(midline_contour)):
        if i > skip_two and i < len(midline_contour)-
skip_two:
            x0,y0 = midline_contour[i-skip_two][0]
            x1,y1 = midline_contour[i][0]
            x2,y2 = midline_contour[i+skip_two][0]
            ## previous way - might be wrong
            # dy0dx0 = (y1-y0)/((x1-x0))
            # dy1dx1 = (y2-y1)/((x2-x1))
            # d2y0dx02 = (dy1dx1-dy0dx0)/((x1-x0))
            # if dy0dx0 == 0 or dy1dx1 == 0 or
np.abs(dy0dx0) > 9999999 or np.abs(dy1dx1) > 99999999:
            #     K = 0.0
            # else:
            #     K =
(d2y0dx02)/(1+(dy0dx0)**2.0)**(3.0/2.0)
            ## different way -
https://math.stackexchange.com/questions/3528424/relation-
between-curvature-and-radius-of-curvature
            theta1 = np.arctan2(y1-y0,x1-x0)
            # if theta1 < 0:
            #     theta1 += 2*np.pi
            theta2 = np.arctan2(y2-y1,x2-x1)
            # if theta2 < 0:
            #     theta2 += 2*np.pi
            dtheta = theta2-theta1
            dtheta = (dtheta + np.pi) % (2*np.pi) - np.pi
            ds = np.sqrt((y1-y0)**2+(x1-
x0)**2)+np.sqrt((y2-y1)**2+(x2-x1)**2)
            K = dtheta/ds
            #
print(i,np.degrees(theta1),np.degrees(theta2),K)

#print(i,x0,y0,x1,y1,x2,y2,dy0dx0,dy1dx1,d2y0dx02,K)
        else:
            K = 0.0
        Kl.append(K)
    #calculate length of worm
    for i in range(len(midline_contour)):
        if i % skip_two == 0:
            try:
                x0,y0 = midline_contour[i][0]
                x1,y1 = midline_contour[i+skip_two][0]
                l += np.sqrt((y1-y0)**2+(x1-x0)**2)
```

```
            except:
                pass
    Kl = np.array(Kl)
    Kl = Kl*l
    # print(Kl)

    if len(dorsal_body_kymograph) > 0:
        new_Kl = []
        for original_length, new_length in zip(Kl.shape,
Kl_shape):
            new_Kl.append(np.linspace(0, original_length-1,
new_length))
        coords = np.meshgrid(*new_Kl, indexing='ij')
        Kl = interpolation.map_coordinates(Kl, coords)
        new_Kl = None
    else:
        Kl_shape = Kl.shape

    d_b_k = []
    v_b_k = []
    for i in Kl:
        if i < 0:
            d_b_k.append(np.abs(i))
        else:
            d_b_k.append(0.0)
    for i in Kl:
        if i > 0:
            v_b_k.append(np.abs(i))
        else:
            v_b_k.append(0.0)
    if not flipCurvature:
        dorsal_body_kymograph.append(d_b_k)
        ventral_body_kymograph.append(v_b_k)
    else:
        dorsal_body_kymograph.append(v_b_k)
        ventral_body_kymograph.append(d_b_k)

    if draw:
        # # Draw all contours
        # -1 signifies drawing all contours
        ## all colors are (B, G, R)
        #cv2.drawContours(img, contours, -1, (255, 0, 0),
1)
        cv2.drawContours(img, dorsal_contour, -1, (255, 0,
0), 3)
        cv2.drawContours(img, ventral_contour, -1, (0, 0,
255), 3)
        cv2.drawContours(img, midline_contour, -1, (128, 0,
128), 1)
        #cv2.drawContours(img,rev_midline_contour, -1, (0,
0, 0), 1)

        ## add corners to output
        #for i in corners:
        for i in (head):
            x,y = i.ravel()
            cv2.circle(img,(x,y),3,(0,255,0),-1)
        for i in (tail):
            x,y = i.ravel()
            cv2.circle(img,(x,y),3,(0,100,0),-1)

    if saveImages:

cv2.imwrite(red_input_folder+'/'+'edge_detection'+'/'+image
, img)
        #plt.imshow(img),plt.show()

    ## pull contour data from gcamp/green channel
    green_img = cv2.imread(green_input_folder+'/'+image,
cv2.IMREAD_UNCHANGED)
    roi_size = 3
    d_k = []
    for d_c in dorsal_contour:
        x,y = d_c[0]
        roi = green_img[y-roi_size:y+roi_size+1, x-
roi_size:x+roi_size+1]
        d_k.append(np.mean(roi))
    ## resize line
    d_k = np.array(d_k)
    if len(dorsal_kymograph) > 0:
        new_d_k = []
        for original_length, new_length in zip(d_k.shape,
dorsal_shape):
            new_d_k.append(np.linspace(0, original_length-
1, new_length))
        coords = np.meshgrid(*new_d_k, indexing='ij')
        d_k = interpolation.map_coordinates(d_k, coords)
        dorsal_kymograph.append(d_k)
        new_d_k = None
    else:
        dorsal_shape = d_k.shape
        dorsal_kymograph.append(d_k)

    v_k = []
    for v_c in ventral_contour:
        x,y = v_c[0]
        roi = green_img[y-roi_size:y+roi_size+1, x-
roi_size:x+roi_size+1]
        v_k.append(np.mean(roi))
    ## resize line
    v_k = np.array(v_k)
    if len(ventral_kymograph) > 0:
        new_v_k = []
        for original_length, new_length in zip(v_k.shape,
ventral_shape):
            new_v_k.append(np.linspace(0, original_length-
1, new_length))
        coords = np.meshgrid(*new_v_k, indexing='ij')
        v_k = interpolation.map_coordinates(v_k, coords)
        ventral_kymograph.append(v_k)
        new_v_k = None
    else:
        ventral_shape = v_k.shape
        ventral_kymograph.append(v_k)

    ## worm straightening
    skip_three = 5
    radius = 25
    straight_worm = None
    prev_distance = 0
    prev_x,prev_y = midline_contour[0][0]
    for i in range(len(midline_contour)):
        if i > skip_three and i < len(midline_contour)-
skip_three:
            x0,y0 = midline_contour[i-skip_three][0]
            x1,y1 = midline_contour[i][0]
            x2,y2 = midline_contour[i+skip_three][0]
            next_distance = np.sqrt((y1-y0)**2+(x1-x0)**2)
            #
print(i,'/',len(midline_contour),prev_distance+next_distanc
e,np.sqrt((x1-prev_x)**2+(y1-prev_y)**2))
            if True: #np.sqrt((x1-prev_x)**2+(y1-
prev_y)**2) > next_distance + prev_distance:
                # print('calculating')
                ## get corners of oblique box
                prev_distance = np.sqrt((y2-y1)**2+(x2-
x1)**2)
                prev_x,prev_y = (x1,y1)
                d = np.sqrt(np.sqrt((y2-y0)**2+(x2-
x0)**2)**2 + radius**2)
                theta_pos = math.atan2((y2-y0), (x2-x0)) +
math.radians(90)
                theta_neg = math.atan2((y2-y0), (x2-x0)) -
math.radians(90)
                c1 =
(x1+radius*math.cos(theta_pos),y1+radius*math.sin(theta_pos
))
                # c2 =
(x2+radius*math.cos(theta_pos),y2+radius*math.sin(theta_pos
))
                c3 =
(x1+radius*math.cos(theta_neg),y1+radius*math.sin(theta_neg
))
                # c4 =
(x2+radius*math.cos(theta_neg),y2+radius*math.sin(theta_neg
))
                #
cv2.circle(img,(int(c1[0]),int(c1[1])),5,(255,0,0),-1)
                #
cv2.circle(img,(int(c2[0]),int(c2[1])),5,(0,255,0),-1)
                #
cv2.circle(img,(int(c3[0]),int(c3[1])),5,(0,0,255),-1)
                #
cv2.circle(img,(int(c4[0]),int(c4[1])),5,(255,255,255),-1)
                # cv2.circle(img,(x2,y2),5,(100,100,100),-
1)
                # cv2.circle(img,(x1,y1),5,(50,50,50),-1)
                # cv2.circle(img,(x0,y0),5,(0,0,0),-1)
                #
cv2.imwrite(red_input_folder+'/'+'edge_detection'+'/'+image
, img)
                row_theta = 0 #math.atan2((c2[1]-c1[1]),
(c2[0]-c1[0]))
                row_dist = 0 #np.sqrt((y2-y0)**2+(x2-
x0)**2)
                col_theta = math.atan2((c3[1]-c1[1]),
(c3[0]-c1[0]))
                col_dist = 2*radius
                temp = []
                for j in np.arange(0,col_dist,1):
                    ref_x = c1[0]+j*math.cos(col_theta)
                    ref_y = c1[1]+j*math.sin(col_theta)
                    ref_x = int(np.round(ref_x))
                    ref_y = int(np.round(ref_y))
                    try:
```

```
temp.append([green_img[ref_y][ref_x]])
                    except:
                        temp.append([0])
                    # for k in np.arange(0,row_dist,1):
                    #     ix = ref_x+k*math.cos(row_theta)
                    #     iy = ref_y+k*math.sin(row_theta)
                    #     ix = int(np.round(ix))
                    #     iy = int(np.round(iy))
                    #     temp[-
1].append(green_img[iy][ix])
                if straight_worm == None:
                    straight_worm = temp
                else:
                    for row_i in range(len(temp)):
                        straight_worm[row_i] =
straight_worm[row_i]+temp[row_i]
    try:
        os.mkdir(red_input_folder +
'\\edge_detection\\straight_worm')
    except:
        pass
    ## resize
    straight_worm = np.array(straight_worm)
    if oneWormDone:
        new_straight_worm = []
        for original_length, new_length in
zip(straight_worm.shape, straight_worm_shape):
            new_straight_worm.append(np.linspace(0,
original_length-1, new_length))
        coords = np.meshgrid(*new_straight_worm,
indexing='ij')
        straight_worm =
interpolation.map_coordinates(straight_worm, coords)
        new_straight_worm = None
    else:
        straight_worm_shape = straight_worm.shape
        oneWormDone = True


cv2.imwrite(red_input_folder+'/'+'edge_detection'+'/'+'stra
ight_worm'+'/'+image, straight_worm.astype('uint16'))

    midline_contour = np.array([i[0] for i in
midline_contour])
    midline_contours.append(midline_contour)

center_of_masses.append([np.mean(midline_contour.T[0]),np.m
ean(midline_contour.T[1])])

##    if image == '0001.tif':
##        print('head:',head)
##        print('dorsal_contour[0]:',dorsal_contour[0])
##        break


dorsal_kymograph = np.array(dorsal_kymograph)
ventral_kymograph = np.array(ventral_kymograph)
dorsal_body_kymograph = np.array(dorsal_body_kymograph)
ventral_body_kymograph = np.array(ventral_body_kymograph)

scipy.io.savemat(red_input_folder+'/'+'edge_detection'+'/'+
'data.mat', mdict={'midline_contours':
midline_contours,'center_of_masses':center_of_masses})

with open(red_input_folder+'/'+'edge_detection'+'/'+
'dorsal_kymograph.pkl', 'wb') as f:
    pickle.dump(dorsal_kymograph, f)
with open(red_input_folder+'/'+'edge_detection'+'/'+
'ventral_kymograph.pkl', 'wb') as f:
    pickle.dump(ventral_kymograph, f)
with open(red_input_folder+'/'+'edge_detection'+'/'+
'dorsal_body_kymograph.pkl', 'wb') as f:
    pickle.dump(dorsal_body_kymograph, f)
with open(red_input_folder+'/'+'edge_detection'+'/'+
'ventral_body_kymograph.pkl', 'wb') as f:
    pickle.dump(ventral_body_kymograph, f)

variables = {
    'saveImages':saveImages,
    'draw':draw,
    'flipHeadTail':flipHeadTail,
    'flipDorsalVentral':flipDorsalVentral,
    'reverseDorsalVentral':reverseDorsalVentral,
    'flipAngle':flipAngle,
    'skipList':skipList,
    'flipCurvature':flipCurvature,
    'kernel_one':len(kernel_one),
    'kernel_two':len(kernel_two),
    'medianBlur_one':medianBlur_one,
    'threshold_one':threshold_one,
    'corners_params':corners_params,
    'skip_one':skip_one,
    'skip_two':skip_two,
    'roi_size':roi_size,
}
with open(red_input_folder+'/'+'edge_detection'+'/'+
'variables.csv', 'w') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in variables.items():
        writer.writerow([key, str(value)])
```

absK_spatio.m

```
function absK_min = absK_spatio( p_pulse,
curvrgn_perturb,curvrgn_analyze, TRange,paradur,
phaserange,curvphiwindow, outpath, issave)
%ABSK_SPATIO makes a 2-D plot of abs(K) as a function of s
and t near
% the time of stimulus. The result is an averaged result
over all trials in
% the same experiment group

[~, strainname] = fileparts(p_pulse);
ddp     = dir(p_pulse);
ddp     = verify_dirlist(ddp,0,'.mat');
Np      = numel(ddp);
numsamplepts = 100;
t = TRange(1) : (TRange(2)-TRange(1))/(numsamplepts-1) :
TRange(2);
x = (0 : numsamplepts-1)./(numsamplepts-1);
k = 0;
for i = 1 : Np
    fprintf('Analyzing trial %d',i)

load(fullfile(p_pulse,ddp(i).name),'curvdatafiltered','fps'
,'istart','start_illum','end_illum');
    curv = curvdatafiltered;
    is = start_illum - istart;
% % check each illumination
% figure(10); clf
% imagesc(x,t, curv)
% colormap(cmap_redblue(0.7))
% colorbar
% xlabel('Body coordinate (Head = 0)')
% ylabel('Time (s)')
%
% paralrgn = [0.6 0; 0.6 paradur; 0.8 paradur; 0.8 0];
% p = patch(paralrgn(:,1), paralrgn(:,2), 'green');
% p.FaceColor = 'none';
% p.EdgeColor = 'green';

    if is + round(TRange(1)*fps) <= 0 || is +
round(TRange(2)*fps) >= size(curv, 1)
        fprintf(' - SKIPPED - Out of time range\n');
        continue;
    end
    icyc = (is + round(TRange(1)*fps)) : (is +
round(TRange(2)*fps));
    if phaserange(2) - phaserange(1)~=1
        % Get the charateristic curvature to define the
phase of undulation
            v =
mean(curvdatafiltered(:,curvrgn_perturb),2);

        % Find all peaks
            [imax, imin] = C2_get_curvature_peaks(v,1);

        % Find the peaks immediately before and after the
start point
            [imax, imin] = verify_extrema(v, imax, imin);
            imaxbef =
imax(find(imax<is+round(curvphiwindow(2)*fps),2,'last'));
            imaxaft =
imax(find(imax>is+round(curvphiwindow(3)*fps),2,'first'));
            iminbef =
imin(find(imin<is+round(curvphiwindow(2)*fps),2,'last'));
            iminaft =
imin(find(imin>is+round(curvphiwindow(3)*fps),2,'first'));

        % Exclude this trial if not enough peaks were found
            if numel(imaxbef) < 2 || numel(imaxaft) < 2 ||
numel(iminaft) < 2 || numel(iminbef) < 2
                fprintf(' - SKIPPED - Not enough peaks\n');
                continue;
            end
        % Calculate the period of undulation
            T0 = mean([diff(imaxaft), diff(imaxbef),
diff(iminaft), diff(iminbef)]);

        % Calculate phi, the phase of the stimulus
            phiu = mod(is - imaxbef(2),T0)/T0;
            phil = mod(is - iminbef(2),T0)/T0 - 0.5;
            if phil < 0
                phil = phil + 1;
            end
%             dphi = abs((phiu - phil)/phil);
```

```
%             if dphi >0.3
%                 fprintf(' - SKIPPED - Can determine the
stimulus phase\n');
%                 continue;
%             end
            phi = mean([phiu phil]);
            if phi > phaserange(2) || phi < phaserange(1)
                fprintf(' - SKIPPED - Out of phaserange of
interest\n');
                continue
            end
%             if phaserange(2) <= 0.5 && imaxbef(2)<=(is -
T0*phaserange(2))
%                 fprintf(' - SKIPPED - phase doesnt
match\n');
%                 continue
%             end
%             if phaserange(1) >= 0.5 && iminbef(2)<=(is -
T0*(phaserange(2)-0.5))
%                 fprintf(' - SKIPPED - phase doesnt
match\n');
%                 continue
%             end
%             Taft = diff(imaxaft);
%             Tbef = diff(imaxbef);
%             dT   = abs(Taft - Tbef)/Tbef;
%         % Exclude this trial if the frequency changes by
a lot
%             if abs(dT) > 0.3 || (T0/fps)>1.75
%                 fprintf(' - SKIPPED - Tratio = %0.2f - T
= %0.2f\n',dT,T0/fps);
%                 continue;
%             end
    end
    fprintf('\n');
    vpatch    = curv(icyc, :);
    K_rescaled = interpl(0:size(vpatch,1)-1, vpatch,
(size(vpatch,1)-1).*(0:numsamplepts-1)/(numsamplepts-
1),'linear');
    k = k+1;
    if k == 1
        absKp_resc_avg = abs(K_rescaled);
    else
        absKp_resc_avg = 1/k*sum(cat(3, (k-
1)*absKp_resc_avg, abs(K_rescaled)),3);
    end
end
% % Find the location of the max compensation
% itmp = (t>=0)&(t<=1);
% Q = absKp_resc_avg(itmp, :);
% figure
% [~, idxmax] = max(Q(:));
% [iTmax, iSmax] = ind2sub(size(Q),idxmax);
% Tmax = t(iTmax)+1; Smax = x(iSmax);
% imagesc(x,t(itmp), Q)
% hold on
% quiver(0.4, 0.1, Smax-0.4, Tmax-0.1,'LineWidth',2);
% colorbar;
% colormap jet
% hold off
% saveas(gcf, fullfile(outpath,['ZabsK_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.fig']))
% saveas(gcf, fullfile(outpath,['ZabsK_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.png']))
% % Visualization
% scrsz = get(groot, 'ScreenSize');
% figure('Position',[1 scrsz(4)*0/8 scrsz(3)/4
scrsz(4)*7/8]);
figure
imagesc(x,t, absKp_resc_avg)
colorbar;
xlabel('Body coordinate (Head = 0)')
ylabel('Time (s)')
title(['|K| ' sprintf('(%.2f',phaserange(1)*2) '\pi-'
sprintf('%.2f', phaserange(2)*2) '\pi)' sprintf('\n %d
trials', k)])
set(gca, 'FontSize', 15)
expo = 0.7;
colormap('jet');
caxis([0.5 5])
paralrgn = [curvrgn_perturb(1)/100 0;
curvrgn_perturb(1)/100 paradur; curvrgn_perturb(2)/100
paradur; curvrgn_perturb(2)/100 0];
% paralrgn = [0.05 0; 0.05 0.1; 0.25 0.1; 0.25 0];
p = patch(paralrgn(:,1), paralrgn(:,2), 'green');
p.FaceColor = 'none';
p.EdgeColor = 'g';
if issave
    saveas(gcf, fullfile(outpath,['absK_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.fig']))
    saveas(gcf, fullfile(outpath,['absK_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.png']))
end
% % Make a contour plot
% figure('Position',[1 scrsz(4)*0/8 scrsz(3)/4
scrsz(4)*7/8]);
% contourf(x,t, absKp_resc_avg, 6)
% xlabel('Body coordinate (Head = 0)')
% ylabel('Time (s)')
% title(['|K| ' sprintf('(%.2f',phaserange(1)*2) '\pi-'
sprintf('%.2f', phaserange(2)*2) '\pi)' sprintf('\n %d
trials', k)])
% p = patch(paralrgn(:,1), paralrgn(:,2), 'green');
% p.FaceColor = 'none';
% p.EdgeColor = 'k';
% ax = gca;
% ax.YDir = 'reverse';
% saveas(gcf, fullfile(outpath,['absK contour_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.fig']))
% saveas(gcf, fullfile(outpath,['absK contour_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.png']))

% figure('Position',[1 scrsz(4)*0/8 scrsz(3)*5/8
scrsz(4)*2/8])
figure
hold on
paralrgn = [0 4; 0 0; paradur 0; paradur 4];
p = patch(paralrgn(:,1), paralrgn(:,2), 'green');
p.EdgeColor = 'none';
absK_anterior =
mean(absKp_resc_avg(:,curvrgn_analyze(1):curvrgn_analyze(en
d)),2);
% absK_anterior = absK_anterior./max(absK_anterior);
plot(t, absK_anterior, 'LineWidth',2)
hold off
set(gca, 'FontSize', 15)
xlabel('Time (s)')
ylabel('|K|')
if issave
    saveas(gcf, fullfile(outpath,['absK1D_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.fig']))
    saveas(gcf, fullfile(outpath,['absK1D_' strainname
sprintf('_%.0f-%.0f', phaserange(1)*100, phaserange(2)*100)
'.png']))
end
fps = numsamplepts/(TRange(2)-TRange(1));
i_p5 = (round(- TRange(1)*fps)) : (round((-
TRange(1)+0.5)*fps));
% t_p5 = t(i_p5);
v_p5 = absK_anterior(i_p5);
absK_min = min(v_p5);
end

function [imax,imin] = verify_extrema(v,imax,imin)

% get the mean amplitudes
 vmax = mean(v(imax));
 vmin = mean(v(imin));

 if vmin > vmax
     itemp = imax;
     imax  = imin;
     imin  = itemp;
 end


end
```

Anterior_Local_Relationship.m

```
clc; close all; clear
p_pulse = {
            'D:\Dropbox\Paper\Compensatory reponse
mechanism\data optogenetics\Combined_148_p1_17pct_P4-6';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po31';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po45';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po75';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po111';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po31_polarize1';...
            'E:\compensatory experiments\Optogenetics\2021-
04-18_YX148_p1_bothside_P4-6_po57_polarize2';...
            'E:\compensatory experiments\Optogenetics\2021-
04-24_YX148_p1_bothside_P4-6_po57_polarize3';...
            'E:\compensatory experiments\Optogenetics\2021-
04-24_YX148_p1_bothside_P4-6_po57_polarize4';...
```

```
          'E:\compensatory experiments\Optogenetics\2021-
04-24_YX148_p1_bothside_P4-6_po57_polarize5';...
          };
outpath = 'D:\Dropbox\Paper\Compensatory reponse
mechanism\Opto dosage Results';
curvrgn_anterior = 16:30; % Anterior 15~27% Local 40~60%
curvrgn_local    = 40:60;
curvrgn_perturb  = 40:60;
curvphiwindow_anterior = [-0.8 -0.0 0.2 1.5]; % Anterior [-
0.8 -0.0 0.2 1.5]
curvphiwindow_local = [-0.8 -0.0 0.1 1.5]; % Local [-0.8 -
0.0 0.1 1.5]
curvphiwindow_perturb = [-0.8 -0.0 0.2 1.5]; % Excitation
[-0.8 -0.0 0.2 1.5] Inhibition [-0.8 -0.0 0.5 1.5]
peaklevel = 1; % upto four
plotflag = 0;
do_comparison = 1;
do_save = 1;
dur_pulse = 0.1;

p = p_pulse{6};
% Produce individual points for anterior response in terms
of phase
[phi_anterior, F_anterior, AVG_anterior, CI95_anterior]
=...
                        Compensatory_Response( p,
curvrgn_perturb, curvrgn_anterior,peaklevel,...

curvphiwindow_perturb,curvphiwindow_anterior,...
                        outpath, plotflag, do_comparison,
do_save);
drawnow
[phi_local, F_local, AVG_local, CI95_local] =...

Compensatory_Response_Inhibition( p, curvrgn_perturb,
curvrgn_local,peaklevel,...

curvphiwindow_perturb,curvphiwindow_local,...
                        outpath, plotflag, do_comparison,
do_save, dur_pulse);


% Divide phase reange (0 ~ 2pi) into 6 bins
num_bins  = 12;
phasestep = 2*pi/num_bins;
Fa_avg    = zeros(num_bins, 1);
Fp_avg    = zeros(num_bins, 1);
Fa_sem    = zeros(num_bins, 1);
Fp_sem    = zeros(num_bins, 1);
for i = 1 : num_bins
    phaserange = phasestep.*(i-1 : i);
    phaserange(1)
    idx1 = (phi_anterior >= phaserange(1) & phi_anterior <
phaserange(end));
    Fa_avg(i) = mean(F_anterior(idx1));
    Fa_sem(i) = std(F_anterior(idx1)) /
sqrt(numel(F_anterior(idx1)));

    idx2 = (phi_local >= phaserange(1) & phi_local <=
phaserange(end));
    Fp_avg(i) = mean(F_local(idx2));
    Fp_sem(i) = std(F_local(idx2)) /
sqrt(numel(F_local(idx2)));
    numel(F_local(idx2))
end
figure
plot(Fp_avg, Fa_avg, '*')
% errorbar(Fp_avg, Fa_avg, Fa_sem, Fa_sem, Fp_sem, Fp_sem,
'o')

figure
todelete = phi_anterior<0 | phi_anterior>2*pi;
phi_anterior(todelete) = [];
AVG_anterior(todelete) = [];
todelete = phi_local<0 | phi_local>2*pi;
phi_local(todelete) = [];
AVG_local(todelete) = [];

tmp = min([numel(AVG_local) numel(AVG_anterior)]);
step = 8;

plot(AVG_local(1:step:tmp), AVG_anterior(1:step:tmp), '-*')
```

Calculations_All_spatioPhase_Response.m

```
% Calculations_All_Phase_Response.m

    clear; clc; close all

% Use PHASE_sort_kymogram_usingManThresh.m to automatically
exclude most bad trials.

% Paths to analyze , labels for each file (comment line to
exclude)
    p           = {

'/Users/hongfei/Dropbox/Paper/Compensatory reponse
mechanism/data
optogenetics/Combined_148_p1_17pct_all','Combined_148_p1_17
pct_all';...

'/Users/hongfei/Dropbox/Paper/Compensatory reponse
mechanism/data optogenetics/Combined_148_p1_17pct_P2-
4','Combined_148_p1_17pct_P2-4';...

'/Users/hongfei/Dropbox/Paper/Compensatory reponse
mechanism/data optogenetics/Combined_148_p1_17pct_P4-
6','Combined_148_p1_17pct_P4-6';...

'/Users/hongfei/Dropbox/Paper/Compensatory reponse
mechanism/data optogenetics/Combined_148_p1_17pct_P6-
8','Combined_148_p1_17pct_P6-8';...


                };

    outpath        = '/Users/hongfei/Dropbox/Paper/motor
circuit/Revision for Elife resubmission/Figures/PRC over
body coordinates induced by perturbations at various
regions';
% Parameters
    toffsetCTRL = 0;
%     curvrgn     = 15 : 35; % the default region
%     aWin = 1*[-0.8 -0.1 0.5 1.5];% Range for analysis
around the START and END of
%                               % illumination (units =
seconds):
%                               %
[befwindow,befbuffer,aftbuffer,aftwindow]
%                               % Usual window is:
%                               % [-0.8 -0.1 0.5 1.5]
    plotdebugflag=0;
    do_analysis_absK = 0;
    curvphiwindow_analyze = [-0.8 -.1 .5 1.5];
    curvphiwindow_perturb = [-0.8 -.1 .5 1.5];
    % Generate
    rgnstart = 10; rgnend = 90;
    Step = 5;
    numrgns = (rgnend - rgnstart)/Step;
    numsamplepts = 100;
    phi_eq = (0:numsamplepts-1)./(numsamplepts-1).*(2*pi);
    x      = (0:numrgns-1)./(numrgns-1);
    curvrgn_perturb = 15:35; % Default 15:35
    for n = 1%:size(p,1)
        PRC_2D     = zeros(numsamplepts, numrgns);
        PRC_2D_sm  = zeros(numsamplepts, numrgns);
        for i = 1 : numrgns
            curvrgn       = (rgnstart+(i-1)*Step) :
(rgnstart+i*Step);
            [phi, F_AVG]  =
Syncronizing_Response(p{n,1},curvrgn_perturb, curvrgn,
curvphiwindow_analyze,curvphiwindow_perturb,outpath,plotdeb
ugflag);
            idx    = (phi>=0 & phi<=2*pi);
            phi2   = phi(idx);
            F_AVG2 = F_AVG(idx);
            [phi3, ia, ic] = unique(phi2);
            F_AVG3 = F_AVG2(ia);
            F_AVG_resc = interp1(phi3, F_AVG3, phi_eq,
'linear');
            %         figure
            %         plot(phi_eq, F_AVG_resc)
            %         xlim([0 2*pi])
            %         ylim([-pi pi])
            PRC_2D(:,i)     = F_AVG_resc;
            PRC_2D_sm(:,i)  = smooth(phi_eq,
F_AVG_resc,0.08, 'rloess');
        end

        % smooth the PRC_2D
        for i = 1:numrgns
            PRC_2D_sm(:,i) = smooth(phi_eq,
PRC_2D(:,i),0.08, 'rloess');
        end


        [~, sname] = fileparts(p{n,1});


        % Plotting


        % normal 2D heat map plot
        figure
        imagesc(phi_eq', x', PRC_2D_sm')
        expo = 0.7;
        map = colormap(cmap_redblue(expo));
```

```
        [cmin, cmax] = caxis;
        caxis([-cmax cmax])
        set(gca,'xaxisLocation','top')
        xlabel('\phi')
        xticks([0 pi 2*pi])
        xticklabels({'0', '\pi', '2\pi'})
        ylabel('Body coordinate')
        yticks([0 0.5 1])
        yticklabels({'0', '0.5', '1'})
        set(gca,'FontSize',14)
        c = colorbar('Ticks', [-2.5,-1.5, 0,1.5, 2.5],...
            'TickLabels', {'-0.8\pi','Advance', '0',
'Delay', '0.8\pi'}, 'Location', 'southoutside',...
            'Direction','reverse');
        c.Label.String = 'Phase shift (\Delta\phi)';
        pbaspect([1 1.1 1])
        set(gcf,'Color','w','Position',[1 217 550 572]);
        fout = fullfile(outpath,
sprintf('%sPhaseResponseCurve_fullbody.svg', p{n,2}));
        saveas(gcf, fout)
        saveas(gcf,strrep(fout,'.svg','.fig'));


%         % Plotting the PRC as a 3D surf
%         [X, Y] = meshgrid(x, phi_eq);
%         figure
%         surf(X,Y,PRC_2D_sm)
%         ylim([0 2*pi])
%         ylabel('\phi')
%         yticks([0 pi 2*pi])
%         yticklabels({'0', '\pi', '2\pi'})
%         xlabel('Body coordinate')
%         xticks([0 0.5 1])
%         xticklabels({'0', '0.5', '1'})
%         zlabel('Phase difference')
%         set(gcf,'Color','w','Position',[700 217 550
572]);
%         fout = fullfile(outpath,
sprintf('%sPhaseResponseCurve_3Dplot.fig', p{n,2}));
%         saveas(gcf, fout)
    end
```

Compensatory_Response.m

```
function [phi_sort, F_sort, CIRC_AVG, CIRC_CI95] =
Compensatory_Response(p_pulse, curvrgn_perturb,
curvrgn_analyze, peaklevel,...

curvphiwindow_perturb,curvphiwindow_analyze, outpath,...
                                plotflag, do_comparison,
do_save, type_perturb)
%COMPENSATORY_RESPONSE Calculate the amplitude of the peak
immediately
%after the pulse, and calculate the difference between
amplitudes of this
%peak and the peak just before the pulse and show this
difference as a
%function of perturbation phase
%
%Outputs:
%           p_pulse: directory of data being evaluated
%           curvrgn1: region being illuminated
%           curvrgn2: region being analyzed
%           phaserange: range of phase of interest
%           curvphiwindow: a window to exclude the effect
of pulse on
%                           curvature dynamics
%           outpath: directory for outcomes

[~, strainname] = fileparts(p_pulse);
ddp  = dir(p_pulse);
ddp  = verify_dirlist(ddp,0,'.mat');
Np   = numel(ddp);
% Cycle through each file, get the peak before and after
the pulse
for i = 1:Np
    % Display progress
    fprintf('Analyzing trial %d',i)
    % Load the file's curvature data

load(fullfile(p_pulse,ddp(i).name),'curvdatafiltered','fps'
,'istart','start_illum');
    %-----------------------------------------------------
----------%
    if rand>0.5
%         fprintf('\tflipped');
%         curvdatafiltered = -curvdatafiltered;
    end
    % Extract the curvature vector
    v1       = mean(curvdatafiltered(:,curvrgn_perturb),2);
    t        = (0:length(v1)-1) ./ fps;
    % Get the start point
    is = start_illum - istart;
    % Find all peaks
    [imax,imin]     = C2_get_curvature_peaks(v1,1);
    % Find the peaks immediately before and after the start
point
    [imax,imin]= verify_extrema(v1,imax,imin);
    imaxbef1 =
imax(find(imax<is+round(curvphiwindow_perturb(2)*fps),2,'la
st'));
    imaxaft1 =
imax(find(imax>is+round(curvphiwindow_perturb(3)*fps),2,'fi
rst'));
    iminbef1 =
imin(find(imin<is+round(curvphiwindow_perturb(2)*fps),2,'la
st'));
    iminaft1 =
imin(find(imin>is+round(curvphiwindow_perturb(3)*fps),2,'fi
rst'));
    % Exclude this trial if not enough peaks were found
    if numel(imaxbef1) < 2 || numel(imaxaft1) < 2 ||
numel(iminaft1) < 2 || numel(iminbef1) < 2
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        F_aft1(i) = nan;
        F_aft2(i) = nan;
        F_aft3(i) = nan;
        F_aft4(i) = nan;
        Fl_aft1(i) = nan;
        Fl_aft2(i) = nan;
        Fl_aft3(i) = nan;
        Fl_aft4(i) = nan;
        Fu_aft1(i) = nan;
        Fu_aft2(i) = nan;
        Fu_aft3(i) = nan;
        Fu_aft4(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        ts2(i) = nan;
        te2(i) = nan;
        fprintf(' - SKIPPED - Not enough peaks\n');
        continue;
    end
    % Calculate the period of undulation
    T0 = mean([diff(imaxaft1), diff(imaxbef1),
diff(iminaft1), diff(iminbef1)]);
    % Calculate the change in period after - should exclude
those with high
    % changes or frequency changes a lot
    dT = abs((diff(imaxaft1) -
diff(imaxbef1))/diff(imaxbef1));

    % Exclude this trial if the frequency changes by a lot
    if dT > 0.2 || (T0/fps) >1.75
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        F_aft1(i) = nan;
        F_aft2(i) = nan;
        F_aft3(i) = nan;
        F_aft4(i) = nan;
        Fl_aft1(i) = nan;
        Fl_aft2(i) = nan;
        Fl_aft3(i) = nan;
        Fl_aft4(i) = nan;
        Fu_aft1(i) = nan;
        Fu_aft2(i) = nan;
        Fu_aft3(i) = nan;
        Fu_aft4(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        ts2(i) = nan;
        te2(i) = nan;
        fprintf(' - SKIPPED - Tratio = %0.2f - T
= %0.2f\n',dT,T0/fps);
        continue;
    end
    % Calculate the phase at which the illum occurs, by
maximum and minimum
%     phiu(i) = mod(is - imaxbef(2),T0)/T0;
%     phil(i) = mod(is - iminbef(2),T0)/T0 - 0.5;
%     if phil(i) < 0
%         phil(i) = phil(i) + 1;
%     end

    v2       = mean(curvdatafiltered(:,curvrgn_analyze),2);
    % Find all peaks
    [imax,imin]     = C2_get_curvature_peaks(v2,1);
    % Find the peaks immediately before and after the start
point
    [imax,imin]= verify_extrema(v2,imax,imin);
    imaxbef2 =
imax(find(imax<is+round(curvphiwindow_analyze(2)*fps),2,'la
st'));
```

```
    imaxaft2 =
imax(find(imax>is+round(curvphiwindow_analyze(3)*fps),2,'fi
rst'));
    iminbef2 =
imin(find(imin<is+round(curvphiwindow_analyze(2)*fps),2,'la
st'));
    iminaft2 =
imin(find(imin>is+round(curvphiwindow_analyze(3)*fps),2,'fi
rst'));

    if numel(imaxbef2) < 2 || numel(imaxaft2) < 2 ||
numel(iminaft2) < 2 || numel(iminbef2) < 2
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        F_aft1(i) = nan;
        F_aft2(i) = nan;
        F_aft3(i) = nan;
        F_aft4(i) = nan;
        Fl_aft1(i) = nan;
        Fl_aft2(i) = nan;
        Fl_aft3(i) = nan;
        Fl_aft4(i) = nan;
        Fu_aft1(i) = nan;
        Fu_aft2(i) = nan;
        Fu_aft3(i) = nan;
        Fu_aft4(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        ts2(i) = nan;
        te2(i) = nan;
        fprintf(' - SKIPPED - Not enough peaks\n');
        continue;
    end

    % Calculate the phase at which the illum occurs, by
maximum and minimum
    phiu(i) = mod(is - imaxbef2(2),T0)/T0;
    phil(i) = mod(is - iminbef2(2),T0)/T0 - 0.5;
    if phil(i) < 0
        phil(i) = phil(i) + 1;
    end

    if do_comparison
        Levels_bef     = [imaxbef2(1); iminbef2(1);
imaxbef2(2); iminbef2(2) ];
        Levels_bef_sort = sort(Levels_bef);
        Levels_aft     = [imaxaft2(1); iminaft2(1);
imaxaft2(2); iminaft2(2) ];
        Levels_aft_sort = sort(Levels_aft);
        Amp_aft1(i) = abs(v2( Levels_aft_sort(1) ));
        Amp_aft2(i) = abs(v2( Levels_aft_sort(2) ));
        Amp_aft3(i) = abs(v2( Levels_aft_sort(3) ));
        Amp_aft4(i) = abs(v2( Levels_aft_sort(4) ));
        % Calculate the change in amplitude, F, by the
maxima
        if type_perturb == 's'
            Amp_befu(i) = abs(v2(imaxbef2(2)));
            Fu_aft1(i)       = (Amp_aft1(i) - Amp_befu(i))/
(Amp_aft1(i) + Amp_befu(i)) * 2;
            Fu_aft2(i)       = (Amp_aft2(i) - Amp_befu(i))/
(Amp_aft2(i) + Amp_befu(i)) * 2;
            Fu_aft3(i)       = (Amp_aft3(i) - Amp_befu(i))/
(Amp_aft3(i) + Amp_befu(i)) * 2;
            Fu_aft4(i)       = (Amp_aft4(i) - Amp_befu(i))/
(Amp_aft4(i) + Amp_befu(i)) * 2;
            % Calculate the change in amplitude, F, by the
minima
            Amp_befl(i) = abs(v2(iminbef2(2)));
            Fl_aft1(i)       = (Amp_aft1(i) - Amp_befl(i))/
(Amp_aft1(i) + Amp_befl(i)) * 2;
            Fl_aft2(i)       = (Amp_aft2(i) - Amp_befl(i))/
(Amp_aft2(i) + Amp_befl(i)) * 2;
            Fl_aft3(i)       = (Amp_aft3(i) - Amp_befl(i))/
(Amp_aft3(i) + Amp_befl(i)) * 2;
            Fl_aft4(i)       = (Amp_aft4(i) - Amp_befl(i))/
(Amp_aft4(i) + Amp_befl(i)) * 2;
        elseif type_perturb == 'i'
            % Calculate the change in amplitude, F, by the
maxima
            Amp_befu(i) = abs(v2(imaxbef2(2)));
            Fu_aft1(i)       = (Amp_aft1(i) - Amp_befu(i))/
Amp_befu(i);
            Fu_aft2(i)       = (Amp_aft2(i) - Amp_befu(i))/
Amp_befu(i);
            Fu_aft3(i)       = (Amp_aft3(i) - Amp_befu(i))/
Amp_befu(i);
            Fu_aft4(i)       = (Amp_aft4(i) - Amp_befu(i))/
Amp_befu(i);
            % Calculate the change in amplitude, F, by the
minima
            Amp_befl(i) = abs(v2(iminbef2(2)));
            Fl_aft1(i)       = (Amp_aft1(i) - Amp_befl(i))/
Amp_befl(i);
            Fl_aft2(i)       = (Amp_aft2(i) - Amp_befl(i))/
Amp_befl(i);
            Fl_aft3(i)       = (Amp_aft3(i) - Amp_befl(i))/
Amp_befl(i);
            Fl_aft4(i)       = (Amp_aft4(i) - Amp_befl(i))/
Amp_befl(i);
        end
    end

    is2   = Levels_bef_sort(1)-is; % start point of the
segment to plot
    ie2   = Levels_aft_sort(2)-is;

    ts2(i)   = is2/fps;
    te2(i)   = ie2/fps;

    if plotflag

        % Crop the curvature curve to only the relevant
times
        is2   = Levels_bef_sort(1)-2; % start point of the
segment to plot
        ie2   = Levels_aft_sort(4)+2;

        ts2(i)   = is2/fps;
        te2(i)   = ie2/fps;

        % Generate plot
        tshift = t - t(is);

        figure(1); clf;
        subplot(2,1,1); hold on;
        patch('Faces', 1:4, 'Vertices', [0 min(v2); 0.1
min(v2); 0.1 max(v2); 0 max(v2)], 'FaceColor', 'green',
'EdgeColor', 'none');
%         line([tshift(is)  tshift(is)] , [min(v2)
max(v2)],'Color','g','LineWidth',3)
        plot(tshift,v1,'-k','LineWidth',2);

plot(tshift(imaxbef1),v1(imaxbef1),'+r','MarkerSize',16,'Li
neWidth',2)

plot(tshift(imaxaft1),v1(imaxaft1),'+r','MarkerSize',16,'Li
neWidth',2)

plot(tshift(iminbef1),v1(iminbef1),'+b','MarkerSize',16,'Li
neWidth',2)

plot(tshift(iminaft1),v1(iminaft1),'+b','MarkerSize',16,'Li
neWidth',2)
        title(sprintf('Mid-body phi = %f',phiu(i)));
        hold off;
        xlim([tshift(is2),tshift(ie2)])
        ylim([-10 10])

        subplot(2,1,2); hold on;
        patch('Faces', 1:4, 'Vertices', [0 min(v2); 0.1
min(v2); 0.1 max(v2); 0 max(v2)], 'FaceColor', 'green',
'EdgeColor', 'none');
%         line([tshift(is)  tshift(is)] , [min(v2)
max(v2)],'Color','g','LineWidth',3)
        plot(tshift,v2,'-k','LineWidth',2);

plot(tshift(imaxbef2),v2(imaxbef2),'+r','MarkerSize',16,'Li
neWidth',2)

plot(tshift(imaxaft2),v2(imaxaft2),'+r','MarkerSize',16,'Li
neWidth',2)

plot(tshift(iminbef2),v2(iminbef2),'+b','MarkerSize',16,'Li
neWidth',2)

plot(tshift(iminaft2),v2(iminaft2),'+b','MarkerSize',16,'Li
neWidth',2)
        title(sprintf('Anterior phi = %f',phiu(i)));
        hold off;
        xlim([tshift(is2),tshift(ie2)])
        ylim([-10 10])
        drawnow;
        %pause(1)
    end



    % Calculate the change in amplitude, F, by the maxima
    Amp_befu(i) = abs(v2(imaxbef2(2)));
    Levels_aft     = [imaxaft2(1); iminaft2(1);
imaxaft2(2); iminaft2(2) ];
    Levels_aft_sort = sort(Levels_aft);
    Amp_aft(i) = abs(v2( Levels_aft_sort(peaklevel) ));
    if type_perturb == 's'
```

```
        Fu(i)       = (Amp_aft(i) - Amp_befu(i))/
(Amp_aft(i) + Amp_befu(i))*2;
    elseif type_perturb == 'i'
        Fu(i)       = (Amp_aft(i) - Amp_befu(i))/
Amp_befu(i);
    end
    % Calculate the change in amplitude, F, by the minima
    Amp_befl(i) = abs(v2(iminbef2(2)));
    Amp_aft(i) = abs(v2(min([imaxaft2(1) iminaft2(1)])));
    if type_perturb == 's'
        Fl(i)       = (Amp_aft(i) - Amp_befl(i))/
(Amp_aft(i) + Amp_befl(i))*2;
    elseif type_perturb == 'i'
        Fl(i)       = (Amp_aft(i) - Amp_befl(i))/
Amp_befl(i);
    end
    fprintf('\n');



end
%% Generate compensatory response curves

% Concatenate the F results by maxima and minima
    phi = cat(1,phiu(:),phil(:))*2*pi;
    F   = cat(1,Fu(:),Fl(:));

    F = F(:);
    phi = phi(:);
    todelete = phi<0 | phi>2*pi |isnan(F) | abs(F)>=5;
    phi1 = phi;
    phi1(todelete) = [];
    Fl = F;
    Fl(todelete) = [];
    phi1=reshape(phi1,1,[]);
    Fl=reshape(Fl,1,[]);

% Sort data by phase at pulse
    [~,idx]     = sort(phi1);
    phi_sort    = phi1(idx);
    F_sort      = Fl(idx);

% Pad the end with beginning data to enable circular
averaging
    phi_pad1     = phi_sort(phi_sort>3*pi/2);
    F_pad1       = F_sort(phi_sort>3*pi/2);
    phi_pad2     = phi_sort(phi_sort<pi/2);
    F_pad2       = F_sort(phi_sort<pi/2);
    phi_sort    = cat(2,phi_pad1-
2*pi,phi_sort,phi_pad2+2*pi);
    F_sort      = cat(2,F_pad1,F_sort,F_pad2);
% Plot results
opengl software;
    figure;
%     clf;
    hold on

%Moving average
    Npoints     = numel(F_sort);
    w_idx       = round(0.15 * Npoints);                % 
Normal is 0.15 The width of the median bin in elements.
Also the N value for each bin. Note that this method could
be invalid if the phases are not sampled approximately
equally.
    CIRC_AVG  = movmean(F_sort,w_idx);
    CIRC_CI95 = movstd(F_sort,w_idx)./sqrt(w_idx);
    shadedErrorBar(phi_sort,CIRC_AVG,CIRC_CI95,'b'); hold
on;

% Individual points

plot(phi_sort,F_sort,'.','MarkerSize',8,'Color',0.5*[0.5
0.5 1]);
    xlabel('Phase of illumination \phi');
    ylabel([sprintf('Normalized bending\n curvature
change,') ' \DeltaK/K'])

    set(gcf,'Color','w','Position',[1192 217 700 570]);
    set(gca,'FontSize',12)

    set(gca,'FontSize',28,'xtick',[0 pi
2*pi],'xticklabel',{'0' ,'\pi' , '2\pi'});
    ylim([-1 1]);
    xlim([0,2*pi]);

% Plot zero line
    line([0 2*pi],[0 0],'Color','k','LineWidth',1);


    hold off;
    if do_save
        saveas(gcf, fullfile(outpath,['CRC_' strainname
'_level' num2str(peaklevel) '_analyze'
num2str(curvrgn_analyze(1)) '-'
num2str(curvrgn_analyze(end)) '.fig']))
        saveas(gcf, fullfile(outpath,['CRC_' strainname
'_level' num2str(peaklevel) '_analyze'
num2str(curvrgn_analyze(1)) '-'
num2str(curvrgn_analyze(end)) '.png']))
    end
    fprintf('trials = %d\n',floor(numel(Fl)/2));


% Plot curvature amplitude before and during inhibition
if do_comparison
    phi = cat(1,phiu(:),phil(:))*2*pi;
    F_aft1 = cat(1,Fu_aft1(:),Fl_aft1(:)); F_aft1 =
F_aft1(:);
    F_aft2 = cat(1,Fu_aft2(:),Fl_aft2(:)); F_aft2 =
F_aft2(:);
    F_aft3 = cat(1,Fu_aft3(:),Fl_aft3(:)); F_aft3 =
F_aft3(:);
    F_aft4 = cat(1,Fu_aft4(:),Fl_aft4(:)); F_aft4 =
F_aft4(:);

    todelete1 = isnan(F_aft1) | abs(F_aft1)>5 | phi > 2*pi
| phi < 0;
    todelete2 = isnan(F_aft2) | abs(F_aft2)>5 | phi > 2*pi
| phi < 0;
    todelete3 = isnan(F_aft3) | abs(F_aft3)>5 | phi > 2*pi
| phi < 0;
    todelete4 = isnan(F_aft4) | abs(F_aft4)>5 | phi > 2*pi
| phi < 0;

    F_aft1(todelete1) = [];
    F_aft2(todelete2) = [];
    F_aft3(todelete3) = [];
    F_aft4(todelete4) = [];

    F_aft1 = reshape(F_aft1,1,[]);
    F_aft2 = reshape(F_aft2,1,[]);
    F_aft3 = reshape(F_aft3,1,[]);
    F_aft4 = reshape(F_aft4,1,[]);

    F_aft1_AVG = mean(F_aft1);
    F_aft1_SEM = std(F_aft1) / sqrt(numel(F_aft1));
    F_aft2_AVG = mean(F_aft2);
    F_aft2_SEM = std(F_aft2) / sqrt(numel(F_aft2));
    F_aft3_AVG = mean(F_aft3);
    F_aft3_SEM = std(F_aft3) / sqrt(numel(F_aft3));
    F_aft4_AVG = mean(F_aft4);
    F_aft4_SEM = std(F_aft4) / sqrt(numel(F_aft4));
    figure
    bar([F_aft1_AVG; F_aft2_AVG; F_aft3_AVG; F_aft4_AVG]);
    hold on
    errorbar([F_aft1_AVG; F_aft2_AVG; F_aft3_AVG;
F_aft4_AVG],[F_aft1_SEM; F_aft2_SEM; F_aft3_SEM;
F_aft4_SEM],'.' );
    hold off

    ylabel([sprintf('Normalized bending\n curvature
change,') ' \DeltaK/K'])
    xticks([1 2 3 4])
    xticklabels({'1','2','3','4'})
    set(gca,'FontSize',20)
    numaft1 = numel(F_aft1)
    numaft2 = numel(F_aft2)
    numaft3 = numel(F_aft3)
    numaft4 = numel(F_aft4)
    if do_save
        saveas(gcf, fullfile(outpath,['CRbars_' strainname
'_analyze' num2str(curvrgn_analyze(1)) '-'
num2str(curvrgn_analyze(end)) '.fig']))
        saveas(gcf, fullfile(outpath,['CRbars_' strainname
'_analyze' num2str(curvrgn_analyze(1)) '-'
num2str(curvrgn_analyze(end)) '.png']))
    end
end

% % estimate the size of quality time window
todelete = isnan(ts2);
ts2(todelete) = [];
te2(todelete) = [];

ts_AVG = mean(ts2)
te_AVG = mean(te2)
ts_SEM = std(ts2)
te_SEM = std(te2)
end

    function [imax,imin] = verify_extrema(v,imax,imin)

        % get the mean amplitudes
        vmax = mean(v(imax));
        vmin = mean(v(imin));
```

```
        if vmin > vmax
            itemp = imax;
            imax  = imin;
            imin  = itemp;
        end


    end
```

Do_absK_spatio.m

```
clc; clear;
p_pulse = {
%             'F:\Compensatory
experiments\Optogenetics\2021-08-25_SWF325_p1_P4-6_17pct'
%             'F:\phase\Combined_148_p1_17pct';
%             'F:\phase\Combined_148_p1_17pct_P2-4';
%             'F:\phase\Combined_148_p1_17pct_P4-6';
%             'F:\phase\Combined_148_p1_17pct_P6-8';
           'F:\phase\Combined_148_p1_17pct_ctrl';
           };
outpath = '/Users/hongfei/Dropbox/Paper/Compensatory
reponse mechanism/Results';
issave = 0;
TRange = [-0.5 1.5];
curvrgn_analyze = [60 80]; % 10:30
curvrgn_perturb = [60 80];
aWin = 1*[-0.8 -0.1 0.2 1.5]; % 0 0.2
numdiv = 1;
Tmax = zeros(numdiv, 1);
Smax = zeros(numdiv, 1);
for n = 1:numel(p_pulse)
    paradur = .1;
    for i = 1:numdiv
        phaserange = [(i-1)/numdiv i/numdiv];
        absK_min = ...
            absK_spatio( p_pulse{n}, curvrgn_perturb,
curvrgn_analyze,...
            TRange, paradur ,phaserange,aWin, outpath,
issave);
    end
end
```

Do_compensatory_response.m

```
clc;  clear
p_pulse = {
%             'F:\phase\Combined_148_p1_17pct_P4-6';
%             'F:\phase\Combined_ZM5398_dor_p1_17pct_P40-
60';
%             'F:\Compensatory
experiments\Optogenetics\2021-08-13_YX287_p1_bothside_P4-
6_20pct';
%             'F:\Compensatory
experiments\Optogenetics\2021-08-13_YX288_p1_bothside_P4-
6_20pct';
%             'F:\Compensatory
experiments\Optogenetics\2021-08-11_YX289_p1_dorsal_P4-
6_20pct';
%             'F:\Compensatory
experiments\Optogenetics\2021-08-18_YX290_p1_dorsal_P4-
6_20pct';
%             'F:\Compensatory
experiments\Optogenetics\2021-04-18_YX148_p1_bothside_P4-
6_po111';
%             'F:\Compensatory
experiments\Optogenetics\2021-08-25_SWF325_p1_P4-6_17pct';
%             'F:\phase\Combined_148_p1_17pct';
%             'F:\phase\Combined_148_p1_17pct_P2-4';
%             'F:\phase\Combined_148_p1_17pct_P4-6';
           'F:\phase\Combined_148_p1_17pct_P6-8';
%             'F:\phase\Combined_148_p1_17pct_ctrl';
           };
outpath = '/Users/hongfei/Dropbox/Paper/Compensatory
reponse mechanism/Results';
curvrgn_analyze = 40:60; % Anterior 15~27% Local 40~60%
curvrgn_perturb = 60:80; %
curvphiwindow_analyze = [-0.8 -0.1 0.1 1.5]; % Excitation
[-0.8 -0.0 0.1 1.5] Inhibition [-0.8 -0.0 0.3 1.5]
curvphiwindow_perturb = [-0.8 -0.0 0.5 1.5]; % Excitation
[-0.8 -0.0 0.1 1.5] Inhibition [-0.8 -0.0 0.5 1.5]
peaklevel = 1; % up to four
plotflag = 0;
do_comparison = 1;
do_save = 0;
type_perturb = 'i';
F_avg = zeros(numel(p_pulse,1));
F_sem = zeros(numel(p_pulse,1));
X1 = [];
Y1 = [];
for i = 1 : numel(p_pulse)
    if i == 2
        curvphiwindow_analyze = [-0.8 -0.0 0.1 1.5];
        type_perturb = 'i';
    end
    [~, F_sort] =  Compensatory_Response( p_pulse{i},
curvrgn_perturb, curvrgn_analyze,peaklevel,...

curvphiwindow_perturb,curvphiwindow_analyze,...
                                          outpath,
plotflag, do_comparison, do_save, type_perturb);
    X1 = cat(1, X1, i*ones([numel(F_sort) 1]));
    Y1 = cat(1, Y1, F_sort');
    F_avg(i) = mean(F_sort);
    F_sem(i) = std(F_sort)/sqrt(numel(F_sort));
end
% %%
% figure(1);clf
% %
beeswarm(X1,Y1,'sort_style','hex','dot_size',.5,'overlay_st
yle','ci','corral_style','gutter');
% bar([1 2], F_avg)
% hold on
% errorbar([1 2], F_avg, F_sem, '.')
% hold off
% ylabel([sprintf('Normalized bending\n curvature change,')
' \DeltaK/K'])
% xticks([1 2])
% xticklabels({'1','2'})
% set(gca,'FontSize',20)
```

Do_compensatory_response_wholebody.m

```
clc; close all; clear
p_pulse = {
           'F:\phase\Combined_148_p1_17pct';
           'F:\phase\Combined_148_p1_17pct_P2-4';
           'F:\phase\Combined_148_p1_17pct_P4-6';
           'F:\phase\Combined_148_p1_17pct_P6-8';
           'F:\phase\Combined_148_p1_17pct_ctrl';
           };
outpath = '/Users/hongfei/Dropbox/Paper/Compensatory
reponse mechanism/Results';
CURVRGN_analyze = {5:25; 20:40; 40:60; 60:80; 80:100}; %
Anterior 15~27% Local 40~60%
CURVRGN_perturb = {5:25; 20:40; 40:60; 60:80; 20:40};  %
curvphiwindow_analyze = [-0.8 -0.0 0.1 1.5]; % Excitation
[-0.8 -0.0 0.1 1.5] Inhibition [-0.8 -0.0 0.3 1.5]
curvphiwindow_perturb = [-0.8 -0.0 0.5 1.5]; % Excitation
[-0.8 -0.0 0.1 1.5] Inhibition [-0.8 -0.0 0.5 1.5]
peaklevel = 1; % up to four
plotflag = 0;
do_comparison = 1;
do_save = 0;
type_perturb = 'i';
F_avg = zeros(numel(CURVRGN_perturb),
numel(CURVRGN_analyze));
F_sem = zeros(numel(CURVRGN_perturb),
numel(CURVRGN_analyze));
for i = 1 : numel(CURVRGN_perturb)
    curvrgn_perturb = CURVRGN_perturb{i};
    for j = 1 : numel(CURVRGN_analyze)
        curvrgn_analyze = CURVRGN_analyze{j};
        [~, F_sort] =  Compensatory_Response( p_pulse{i},
curvrgn_perturb, curvrgn_analyze,peaklevel,...
            curvphiwindow_perturb,curvphiwindow_analyze,...
            outpath, plotflag, do_comparison, do_save,
type_perturb);
        F_avg(i,j) = mean(F_sort);
        F_sem(i,j) = std(F_sort)/sqrt(numel(F_sort));
    end
end
% %%
```

Do_K_spatio.m

```
clc; clear; close all
p_pulse = {'E:\phase\Combined_148_p1_17pct_P4-6';...
           'E:\phase\Combined_148_p1_17pct_ctrl';...

'C:\Users\fffei\Dropbox\PRC\Combined_148_0p100_ctrl';...
           'E:\phase\2019-11-07_ZM5398_dor_p5_17_p40-
60';...
           'E:\phase\2019-11-06_ZM5398_dor_p1_17_p40-
60';...
           'E:\phase\Combined_ZM5398_dor_p5_17pct_P40-60'
           };
outpath = 'C:\Users\fffei\Dropbox\Paper\motor
circuit\Compensatory reponse mechanism\Preliminary
Results';
TRange = [-1 2];
curvrgn_perturb = 40:60;
curvrgn_analyze = 10:30;
curvphiwindow_perturb = [-0.8 -0.1 0.5 1.5];
curvphiwindow_analyze = [-0.8 -0.1 0.25 1.5];
numdiv = 1;
for n = 6%numel(p_pulse)
    paradur = 0.5;
    for i = 1:numdiv
        phaserange = [(i-1)/numdiv i/numdiv];
```

```
        K_spatio( p_pulse{n}, curvrgn_perturb,
curvrgn_analyze,...
                  TRange,paradur,
phaserange,curvphiwindow_perturb,curvphiwindow_analyze,
outpath)
    end
end
```

Do_paralysis_response_analysis.m

```
% clc; close all; clear
p_pulse = {

'F:\phase\Combined_148_p1_dor_30pct_p5_25_ALL';...

'F:\phase\Combined_148_p1_ven_30pct_p5_25_ALL';...
           'F:\phase\Combined_148_p1_35pct';...
           'F:\phase\Combined_148_p1_35pct_ctrl';...
           };
outpath = 'C:\Users\fffei\Dropbox\Paper\motor
circuit\Compensatory reponse mechanism\Preliminary Results
2';
Tpara = 0.3;
curvrgn_analyze = 15:35; % 15:35
curvrgn_perturb = 15:35;
curvphiwindow_analyze = [-0.8 -.1 .5 1.5];
curvphiwindow_perturb = [-0.8 -.1 .5 1.5];
plotflag = 0;
color = '-k';
for n = 4
   [phi_sort, P_sort] =
Paralysis_Response_absK( p_pulse{n}, curvrgn_perturb,
curvrgn_analyze,...

curvphiwindow_perturb,curvphiwindow_analyze, Tpara,...
                                        outpath,
plotflag, color);

end
%
figure(1)
Dt  = 0.3;
T0  = 1.14;
A = 4.2;
width = 0.4*2*pi;
funD = @(x) A*abs(cos(x+ 2*pi*Dt/T0));
funI = @(t) (integral(funD, t-width/2, t+width/2))/width;
hold on
fplot(funI, [0 2*pi])
hold off

xlabel('\phi','FontSize',20);
ylabel('|K| (0.3 after illum)','FontSize',12);

set(gca,'FontSize',12)
set(gca,'FontSize',28,'xtick',[0 pi
2*pi],'xticklabel',{'0' ,'\pi' , '2\pi'});
xlim([0,2*pi]);

% hold on
% Npoints     = numel(P_sort);
% w_idx       = round(0.2 * Npoints);  % Normal is 0.15 The
width of the median bin in elements. Also the N value for
each bin. Note that this method could be invalid if the
phases are not sampled approximately equally.
% AVG = movmean(P_sort,w_idx);
% SEM = movstd(P_sort,w_idx)./sqrt(w_idx);
% shadedErrorBar(phi_sort,smooth(AVG-1),smooth(SEM), '-c',
0.5);
% hold off
```

Do_state_analysis.m

```
clc; clear;close all
p_pulse = '/Users/hongfei/Dropbox/Paper/Compensatory
reponse mechanism/data
optogenetics/Combined_148_p1_17pct_ctrl';
curvrgn_anterior = 15:35;
curvrgn_middle   = 40:60;
curvphiwindow = [-4.5 -0.0 0.3 1.5];
numdiv = 1;
doplotdebug = 0;
load('TmaxSmax.mat')
for i = 1 : numdiv
    phaserange = [(i-1)/numdiv i/numdiv];
    if i == 1
        ctrdata = {};
    else
        ctrdata = {rescaled_data{9}; rescaled_data{10};
rescaled_data{11}; rescaled_data{12}; raw_data{3};
raw_data{4}; rescaled_data{13}};
    end
    [rescaled_data, raw_data] =
State_analysis_2019a(p_pulse, curvrgn_anterior,
curvrgn_middle,...
        phaserange, curvphiwindow, doplotdebug, ctrdata);
end
```

Paralysis_Response_absK.m

```
function [phi_sort, P_sort] =
Paralysis_Response_absK(p_pulse, curvrgn_perturb,
curvrgn_analyze,...

curvphiwindow_perturb,curvphiwindow_analyze,Tpara,
outpath,...
                                    plotflag, color)
%SYNCRONIZING_RESPONSE Calculate
%
%Outputs:
%           p_pulse: directory of data being evaluated
%           curvrgn1: region being illuminated
%           curvrgn2: region being analyzed
%           phaserange: range of phase of interest
%           curvphiwindow: a window to exclude the effect
of pulse on
%                           curvature dynamics
%           outpath: directory for outcomes

[~, strainname] = fileparts(p_pulse);
ddp  = dir(p_pulse);
ddp  = verify_dirlist(ddp,0,'.mat');
Np   = numel(ddp);
numsamplepts = 100;
Period = [];
% Cycle through each file, get the peak before and after
the pulse
for i = 1:Np
    % Display progress
    fprintf('Analyzing trial %d',i)
    % Load the file's curvature data

load(fullfile(p_pulse,ddp(i).name),'curvdatafiltered','fps'
,'istart','start_illum');
    %-------------------------------------------------------
----------%

    % Extract the curvature vector
    v1       = mean(curvdatafiltered(:,curvrgn_perturb),2);
    t       = (0:length(v1)-1) ./ fps;
    % Get the start point
    is = start_illum - istart;

    % Find all peaks
    [imax,imin]    = C2_get_curvature_peaks(v1,1);
    % Find the peaks immediately before and after the start
point
    [imax,imin]= verify_extrema(v1,imax,imin);
    imaxbef =
imax(find(imax<is+round(curvphiwindow_perturb(2)*fps),2,'la
st'));
    imaxaft =
imax(find(imax>is+round(curvphiwindow_perturb(3)*fps),2,'fi
rst'));
    iminbef =
imin(find(imin<is+round(curvphiwindow_perturb(2)*fps),2,'la
st'));
    iminaft =
imin(find(imin>is+round(curvphiwindow_perturb(3)*fps),2,'fi
rst'));
    % Exclude this trial if not enough peaks were found
    if numel(imaxbef) < 2 || numel(imaxaft) < 2 ||
numel(iminaft) < 2 || numel(iminbef) < 2
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        fprintf(' - SKIPPED - Not enough peaks\n');
        continue;
    end
        % Calculate the period of undulation
    T0 = mean([diff(imaxaft), diff(imaxbef), diff(iminaft),
diff(iminbef)]);
    % Calculate the change in period after - should exclude
those with high
    % changes or frequency changes a lot
    dT = abs((diff(imaxaft) -
diff(imaxbef))/diff(imaxbef));

    % Exclude this trial if the frequency changes by a lot
    if dT > 0.2 || (T0/fps) >1.75
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        fprintf(' - SKIPPED - Tratio = %0.2f - T
= %0.2f\n',dT,T0/fps);
```

```
        continue;
    end

    % Calculate the phase at which the illum occurs, by
maximum and minimum
    phiu(i) = mod(is - imaxbef(2),T0)/T0;
    phil(i) = mod(is - iminbef(2),T0)/T0 - 0.5;
    if phil(i) < 0
        phil(i) = phil(i) + 1;
    end

    v2       = mean(curvdatafiltered(:,curvrgn_analyze),2);
    % Find all peaks
    [imax,imin]     = C2_get_curvature_peaks(v2,1);
    % Find the peaks immediately before and after the start
point
    [imax,imin]= verify_extrema(v2,imax,imin);
    imaxbef =
imax(find(imax<is+round(curvphiwindow_analyze(2)*fps),2,'la
st'));
    imaxaft =
imax(find(imax>is+round(curvphiwindow_analyze(3)*fps),2,'fi
rst'));
    iminbef =
imin(find(imin<is+round(curvphiwindow_analyze(2)*fps),2,'la
st'));
    iminaft =
imin(find(imin>is+round(curvphiwindow_analyze(3)*fps),2,'fi
rst'));

    if numel(imaxbef) < 2 || numel(imaxaft) < 2 ||
numel(iminaft) < 2 || numel(iminbef) < 2
        F(i)  = nan;
        Fu(i) = nan;
        Fl(i) = nan;
        phiu(i)= nan;
        phil(i) = nan;
        fprintf(' - SKIPPED - Not enough peaks\n');
        continue;
    end

    if plotflag

        % Crop the curvature curve to only the relevant
times
        is2   = min([min(imaxbef) min(iminbef)])-2; % start
point of the segment to plot
        ie2   = max([max(imaxaft) max(iminaft)])+2;

        % Generate plot
        figure(1); clf;
        plot(t,v2,'-k','LineWidth',2); hold on;
%
plot(t(imaxbef),v2(imaxbef),'+r','MarkerSize',16,'LineWidth
',2)

plot(t(imaxaft(1)),v2(imaxaft(1)),'+r','MarkerSize',16,'Lin
eWidth',2)
%
plot(t(iminbef),v2(iminbef),'+b','MarkerSize',16,'LineWidth
',2)

plot(t(iminaft(1)),v2(iminaft(1)),'+b','MarkerSize',16,'Lin
eWidth',2)
        line([t(is)  t(is)] , [min(v2)
max(v2)],'Color','g','LineWidth',3)
        title(sprintf('phi = %f',phiu(i)));
        hold off;
        xlim([t(is2),t(ie2)])
        drawnow;
        pause(2)
    end
    Kp3 = v1(is + round(Tpara*fps));

    F(i) = Kp3;
%     F(i) = abs(Kp3);

    fprintf('\n');

    Period = [Period; T0];

end
%% Generate compensatory response curves

% Concatenate the F results by maxima and minima
%     phi = cat(1,phiu(:),phil(:))*2*pi;
    phi = ((phiu(:)+phil(:))/2)*2*pi;
    F = F(:);
    phi = phi(:);
    todelete = phi<0 | phi>2*pi |isnan(F);
    phi(todelete) = [];
    F(todelete) = [];
    phi=reshape(phi,1,[]);
    F=reshape(F,1,[]);

% Sort data by phase at pulse
    [~,idx]     = sort(phi);
    phi_sort    = phi(idx);
    P_sort      = F(idx);

% Pad the end with beginning data to enable circular
averaging
    phi_pad1     = phi_sort(phi_sort>3*pi/2);
    F_pad1       = P_sort(phi_sort>3*pi/2);
    phi_pad2     = phi_sort(phi_sort<pi/2);
    F_pad2       = P_sort(phi_sort<pi/2);
    phi_sort    = cat(2,phi_pad1-
2*pi,phi_sort,phi_pad2+2*pi);
    P_sort      = cat(2,F_pad1,P_sort,F_pad2);
% Plot results
    opengl software;
    figure(1);
%     clf;
    hold on

%Moving average
    Npoints     = numel(P_sort);
    w_idx       = round(0.25 * Npoints);  % Normal is 0.15
The width of the median bin in elements. Also the N value
for each bin. Note that this method could be invalid if the
phases are not sampled approximately equally.
    AVG = movmean(P_sort,w_idx);
    SEM = movstd(P_sort,w_idx)./sqrt(w_idx);
    AVG = smooth(AVG);
    SEM = smooth(SEM);
    shadedErrorBar(phi_sort,AVG,SEM, color, 0.5);
    hold on;

% Individual points
%
plot(phi_sort,P_sort,'.','MarkerSize',8,'Color',0.5*[0.5
0.5 1]);
    xlabel('\phi','FontSize',20);
    ylabel('|K| (0.3 after illum)','FontSize',12);

    set(gcf,'Color','w','Position',[1192 217 608 572]);
    set(gca,'FontSize',12)
    set(gca,'FontSize',28,'xtick',[0 pi
2*pi],'xticklabel',{'0' ,'\pi' , '2\pi'});
    xlim([0,2*pi]);

% Plot zero line


    hold off;

%     saveas(gcf, fullfile(outpath,['ParaC_' strainname
'.fig']))
%     saveas(gcf, fullfile(outpath,['ParaC_' strainname
'.png']))
    fprintf('trials = %d\n',numel(F)/2);

    Period_avg = mean(Period)/fps;
    Period_sem = std(Period)/sqrt(numel(Period))/fps;
    fprintf('T0 = %d,\n SEM = %d\n',Period_avg,
Period_sem);
end

    function [imax,imin] = verify_extrema(v,imax,imin)

        % get the mean amplitudes
        vmax = mean(v(imax));
        vmin = mean(v(imin));

        if vmin > vmax
            itemp = imax;
            imax  = imin;
            imin  = itemp;
        end


    end
```

shadedErrorGaussian.m

```
function H = shadedErrorGaussian(x, mu, sg )
%SHADEDERRORGAUSSIAN
hold on
for i = 1: numel(x)
    MU = mu{i};
    if numel(MU) == 1
        scatter(x(i), MU, 'filled',
'MarkerEdgeColor','none',...
            'MarkerFaceColor','k');
    elseif numel(MU) == 2
        scatter([x(i) x(i)], MU, 'filled',
'MarkerEdgeColor','none',...
```

```
            'MarkerFaceColor','k');
    end
end
hold off
end
```

Analysis_Tool.m

```
clc; clear; close all
subfoldername = 'LX703_30pct';
path    = fullfile('D:\Dropbox\Paper\Compensatory reponse
mechanism\data_gait_adaptation',subfoldername);
savefilename = fullfile('D:\Dropbox\Paper\Compensatory
reponse
mechanism\data_gait_adaptation\intermediate',[subfoldername
'.mat']);
list = dir(fullfile(path, '*.mat'));
spline_p = 0.01;
numcurvpts = 100;
curvrgn  = 10:30;
n_trials = numel(list);
Total_time = 0;
expo = .7;
f_data  = [];
l_data  = [];
th_data = [];
curv_data = [];
do_wavelength = 1;
do_saveparameters = 1;
for i = 1 : numel(list)
    fprintf('%i out of %i\n', i, numel(list))
    f  = list(i).name;
    load(fullfile(path,f))
    xy = cat(3, clinex, cliney);
    iT = size(xy,1);
    iS = size(xy,2);

    % do wavelength analysis
    if do_wavelength

        indicator = curvdatafiltered;

        c2n = bsxfun(@gt, indicator, mean(indicator,
'omitnan'));
        maskhead = 0.15;
        maskneck = 0.17;
        masktail = 0.30;
        minimum_fraction_for_fit = 0.5;
        c3 = edge(single(c2n),'sobel', 0);

        [c4, numlab] = bwlabel(c3);

        numcycles2 = 0;


        okdatatmp = zeros(numlab, 1);

        normrthresh = 220;

        %   draw fit limits
        figure(9);clf; imagesc(c2n);
colormap(cmap_redblue(expo));
        %
        hold on;
        %     subplot(133); hold on;
        curvsigndatatmp = [];
        for n=1:numlab
            c5 = (c4 == n);
%                 figure(23);
%                 imagesc(c5); hold on;
            [y, x] = find(c5);

            % determine if this is + or - transition
            yshift = 3;
            yshifted = ceil(1+0.5*(1+sign(y-yshift)) .* (y-
yshift-1));
            curvshift = zeros(size(x));
            for jj=1:length(x)
                curvshift(jj) = indicator(yshifted(jj),
x(jj));
            end
            %
            %         disp(n);
            %         disp('mean(c2shift)');
            %         disp(mean(curvshift));
            %
            % exclude points near head and tail for fitting
            tmp = x;
            x=x(logical((tmp>=maskhead * numcurvpts) .*
(tmp<=(1-masktail) * numcurvpts)));
            y=y(logical((tmp>=maskhead * numcurvpts) .*
(tmp<=(1-masktail) * numcurvpts)));
            if max(x) - min(x) >=  (1-maskhead-
masktail)*minimum_fraction_for_fit*numcurvpts
                [p,S] = polyfit(x,y,1);
                if S.normr < normrthresh % if the curve is
close to a straight line
                    if mean(curvshift) > 0
                        plotcol = '-g';
                    else
                        plotcol = '--g';
                    end
                    figure(9)
                    plot(polyval(p,1:numcurvpts), plotcol);
hold on;
                    numcycles2 = numcycles2 + 1;
                    slopedatatmp(n) = p(1);
                    timedatatmp(n) = p(2);
                    okdatatmp(n) = 1;
                    negshift = (mean(curvshift) > 0);
                    curvsigndatatmp(n) = negshift;
                    xpos = 5;
                    ypos = p(2)-1;
                    if p(2)<1
                        xpos = numcurvpts/4;
                        ypos = 5;
                    end
                    if negshift
                        xpos = 45;
                        ypos = p(2) + p(1)*xpos-1;
                    end
                    text(xpos,ypos,num2str([numcycles2
p(1)]), 'Color', 'white'); hold on;
                end
            end
        end
        if isempty(curvsigndatatmp)
            continue;
        end
        figure(9);hold on;
        plot( [0.5+maskhead*numcurvpts
0.5+maskhead*numcurvpts],[1 iT], ':k');
        plot( [0.5+ (1-masktail)*numcurvpts 0.5+ (1-
masktail)*numcurvpts],[1 iT], ':k');
        xlabel(sprintf('%d out of %d', i, n_trials))

        title(strcat(num2str(sum(curvsigndatatmp)),'
positive, ', num2str(numcycles2-sum(curvsigndatatmp)),...
            ' negative.  Click on bad fits, press
return'));
        badfits = ginput;

        epsilon = 4;  % how close to fit you need to click
        for j=1:size(badfits,1)
            for n=1:numlab
                if okdatatmp(n)
                    if abs(timedatatmp(n) +
slopedatatmp(n)*badfits(j,1) - badfits(j,2))<epsilon
                        %
disp(strcat('matches #', num2str(n)));
                        okdatatmp(n) = 0;
                    end
                end
            end
        end

        numcycles2 = 0;
        c4b = c4;
        for n=1:numlab
            if okdatatmp(n)
                numcycles2 = numcycles2+1;
                slopedata(numcycles2) = slopedatatmp(n);
                timedata(numcycles2) = timedatatmp(n);

curvsigndata(numcycles2)=curvsigndatatmp(n);
                c4b(c4b==n) = numcycles2;
            else
                c4b(c4b==n) = 0;
            end
        end

        figure(9);clf;
        imagesc(c2n); hold on;

        plot( [0.5+maskhead*numcurvpts
0.5+maskhead*numcurvpts],[1 iT], ':w');
        plot( [0.5+ (1-masktail)*numcurvpts 0.5+ (1-
masktail)*numcurvpts],[1 iT], ':w');
        xlabel(sprintf('%d out of %d', i, n_trials))

        for n=1:numcycles2
            if curvsigndata(n)
                plotcol = '-g';
            else
                plotcol = '--g';
```

```
            end

            plot(polyval([slopedata(n)
timedata(n)],1:numcurvpts), plotcol); hold on;
            %               text(5,p(2)-1,num2str([n
numcycles2 S.normr]), 'Color', 'white'); hold on;
            %               if p(2)<1
            %                   text(5,2,num2str([n
numcycles2 S.normr]), 'Color', 'white'); hold on;
            %               end
        end

        if mean(curvsigndata) ~= 0.5
            msgbox('Warning: unequal number of positive and
negative fits','','error')
        end

        title('Press return to continue');
        pause;
        figure(9)


        title('click two points separated in time by N
cycles');
        t1 = ginput(1);
        if numel(t1) <2
            continue
        end
        plot( [1 numcurvpts],[t1(2) t1(2)], '-w');

        t2 = ginput(1);
        if numel(t2) <2
            continue
        end
        plot( [1 numcurvpts],[t2(2) t2(2)], '-w');
        %   draw fit limits

        plot( [0.5+maskhead*numcurvpts
0.5+maskhead*numcurvpts],[1 iT], ':k');
        plot( [0.5+ (1-masktail)*numcurvpts 0.5+ (1-
masktail)*numcurvpts],[1 iT], ':k');

        idx = sort([t1(2) t2(2)]);
        v_front_all = (c2n( :,floor(maskhead *
numcurvpts)));
        dv_front = edge(single(v_front_all),'sobel', 0);
        idx_edge = find(dv_front == 1);
        [~, loc1] = min(abs((idx(1) - idx_edge)));
        idx(1) = idx_edge(loc1);
        [~, loc2] = min(abs((idx(2) - idx_edge)));
        idx(2) = idx_edge(loc2);


        v_front = c2n( idx(1)+1: idx(2)-1,floor(maskhead *
numcurvpts));
        v3_front = edge(single(v_front),'sobel', 0);
        num_halfcycles = sum(v3_front)+1;
        answer = inputdlg(sprintf('Enter number of cycles
(suggestion: %.1f)', num_halfcycles/2));

        if isempty(answer{1})
            num_cycles = num_halfcycles/2;
        else
            num_cycles = str2double(answer{1});
        end

        period = abs(t1(2) - t2(2)) / num_cycles;

        title(strcat(num2str(num_cycles),' cycles, ',
num2str(numcycles2), ' fits'), 'Interpreter', 'None');


        for n=1:numcycles2
            if curvsigndata(n)
                plotcol = '-g';
            else
                plotcol = '--g';
            end
            plot(polyval([slopedata(n)
timedata(n)],(1:numcurvpts)), plotcol); hold on;

        end
    end

    df = diff(xy,1,2);
    lendata = zeros(size(xy,1),1);
    angles_per_trial = [];
    curvs_per_trial  = [];
    for j = idx(1) : idx(2)
        if isnan(mean(curvdatafiltered(j,:)))
            continue
        end
        df2d = squeeze(df(j,:,:))';
        xy2d = squeeze(xy(j,:,:))';

        s  = cumsum([0, sqrt([1 1]*(df2d.*df2d))]);
        cv = csaps(s,xy2d,spline_p);
        cv2 =  fnval(cv, s)';
        df2 = diff(cv2,1,1); df2p = df2';
        splen = cumsum([0, sqrt([1 1]*(df2p.*df2p))]);
        lendata(j) = splen(end);
        cv2i = interp1(splen+.00001*(0:length(splen)-
1),cv2,...
            (0:(splen(end)-1)/(numcurvpts+1):(splen(end)-
1)));
        df2   = diff(cv2i,1,1);
        atdf2 = atan2(-df2(10:90,2), df2(10:90,1));

        theta   = (mean(max(atdf2)) + mean(min(atdf2)))/2;
        xcenter = cv2i(1,1);
        ycenter = cv2i(1,2);
        center  = repmat([xcenter ycenter], size(cv2i, 1),
1);
        Ro      = [cos(theta) -sin(theta); sin(theta)
cos(theta)];
        % do the rotation
        cv2io = (Ro*(cv2i' - center') + center')';
        df2o   = diff(cv2io,1,1);
        atdf2o = atan2(-df2o(10:90,2), df2o(10:90,1));
%         angle_data(j,:) = atdf2o';
        max_angle = abs(mean(max(atdf2o)));
        % compute average of attack angle over body
coordinate and over periods
        % of cycle
        if max_angle < pi/2 * 0.95 && max_angle > 0
            angles_per_trial   = [angles_per_trial;
max_angle];
        end

    end

    v = mean(curvdatafiltered(idx(1) : idx(2), curvrgn),2);
%     [imax, imin] = C2_get_curvature_peaks(v,1);
%     [imax, imin]  = verify_extrema(v,imax,imin);
%     imax(imax == iT | imax == 1) = [];
%     imin(imin == iT | imin == 1) = [];
%     imax(v(imax)<=0) = [];
%     imin(v(imin)>=0) = [];
    ddx = floor(abs(idx(1) - idx(2))./num_cycles);
    for jj = 1 : num_cycles
        idxs = (1+(jj-1)*ddx):(jj*ddx);
        curvs_per_trial = [curvs_per_trial;
max(abs(v(idxs)))];
    end
    % number of fits
    fprintf('number of fits = %d \n', numcycles2)
    % worm length
    wormlength = mean(lendata);
    %         fprintf('wormlength (spline) (pix) = %f \n',
wormlength)

    % undulation period
    %         fprintf('mean period (frames) = %f \n',
period)
    period_s = period/fps;
    fprintf('mean period (sec) = %f \n',period_s)
    % undulation frequency
    %         frequency = 1/period;
    %         fprintf('mean frequency (frames-1) = %f \n',
frequency)
    frequency_hz = 1/period_s;
    fprintf('mean frequency (sec-1) = %f \n', frequency_hz)

    % wavevelocity
    wavevelocity = mean(1./abs(slopedata),'omitnan') *
wormlength/numcurvpts * fps;
    %         fprintf('mean wave velocity (pix/sec) = %f
\n', wavevelocity)
    %
    %         stdwavevelocity = std(1./abs(slopedata))*
wormlength/numcurvpts * fps;
    %         fprintf('std wave velocity (pix/sec) = %f
\n', stdwavevelocity)

    % wavelength
    wavelength = wavevelocity * period_s;
    %         fprintf('wavelength (pix) = %f \n',
wavelength)

    wavelength_norm = wavelength / wormlength;
    fprintf('wavelength (norm) = %f \n', wavelength_norm)
    %         wavelengthdata = 1./abs(slopedata) *
wormlength/numcurvpts * period;

    % angle of attack
    halfperiod = floor(period/2);
```

```
    N_halfcycles =
floor(length(angles_per_trial)/halfperiod) + 1;
    for ii = 1 : N_halfcycles
        if ii<N_halfcycles
            range_period = (1 + (ii-1)*halfperiod) :
(halfperiod*ii);
        else
            range_period = (1 + (ii-1)*halfperiod) :
length(angles_per_trial);
        end
        angles_per_period(ii) =
mean(max(angles_per_trial(range_period)));
    end
    angle_attack = mean(angles_per_trial, 'omitnan');
    angle_attack_degree = angle_attack/pi*180;
    fprintf('attack angle (degree) = %f \n',
angle_attack_degree)

    % peak curvature
    curvature = mean(curvs_per_trial, 'omitnan');
    fprintf('curvature (norm) = %f \n', curvature)

    f_data = [f_data; frequency_hz];
    l_data = [l_data; wavelength_norm];
    th_data = [th_data; angle_attack_degree];
    curv_data = [curv_data; curvature];

end

f_mean_all  = mean(f_data);
f_std_all   = std(f_data);
f_SEM_all   = f_std_all/sqrt(n_trials);

l_data(l_data>2.5) = [];
l_mean_all  = mean(l_data);
l_std_all   = std(l_data);
l_SEM_all   = l_std_all/sqrt(n_trials);

th_mean_all = mean(th_data, 'omitnan');
th_std_all  = std(th_data, 'omitnan');
th_SEM_all  = th_std_all/sqrt(n_trials);

curv_mean_all = mean(curv_data, 'omitnan');
curv_std_all  = std(curv_data, 'omitnan');
curv_SEM_all  = curv_std_all/sqrt(n_trials);

if do_saveparameters
    save(savefilename,
'f_data','l_data','th_data','curv_data',...
        'f_mean_all','f_std_all','f_SEM_all',...
        'l_mean_all','l_std_all','l_SEM_all',...
        'th_mean_all','th_std_all','th_SEM_all',...
        'curv_mean_all','curv_std_all','curv_SEM_all')
end

function [imax,imin] = verify_extrema(v,imax,imin)

% get the mean amplitudes
vmax = mean(v(imax));
vmin = mean(v(imin));

if vmin > vmax
    itemp = imax;
    imax  = imin;
    imin  = itemp;
end


end
```

mutants.m

```
% close all;
clear; clc
Vis     = [1390 9079]'; % Viscosity  (mPa¬∑s)
pct     = {'30','40'};
N       = numel(Vis);

f_N2     = zeros(N,1);
f_N2_SEM = zeros(N,1);
f_dop3    = zeros(N,1);
f_dop3_SEM= zeros(N,1);
f_avkdop3    = zeros(N,1);
f_avkdop3_SEM= zeros(N,1);
f_avkTeTx    = zeros(N,1);
f_avkTeTx_SEM= zeros(N,1);
f_flp1    = zeros(N,1);
f_flp1_SEM= zeros(N,1);
f_npr6    = zeros(N,1);
f_npr6_SEM= zeros(N,1);
f_smbnpr6    = zeros(N,1);
f_smbnpr6_SEM= zeros(N,1);

l_N2     = zeros(N,1);
l_N2_SEM = zeros(N,1);
l_dop3    = zeros(N,1);
l_dop3_SEM= zeros(N,1);
l_avkdop3    = zeros(N,1);
l_avkdop3_SEM= zeros(N,1);
l_avkTeTx    = zeros(N,1);
l_avkTeTx_SEM= zeros(N,1);
l_flp1    = zeros(N,1);
l_flp1_SEM= zeros(N,1);
l_npr6    = zeros(N,1);
l_npr6_SEM= zeros(N,1);
l_smbnpr6    = zeros(N,1);
l_smbnpr6_SEM= zeros(N,1);

k_N2     = zeros(N,1);
k_N2_SEM = zeros(N,1);
k_dop3    = zeros(N,1);
k_dop3_SEM= zeros(N,1);
k_avkdop3    = zeros(N,1);
k_avkdop3_SEM= zeros(N,1);
k_avkTeTx    = zeros(N,1);
k_avkTeTx_SEM= zeros(N,1);
k_flp1    = zeros(N,1);
k_flp1_SEM= zeros(N,1);
k_npr6    = zeros(N,1);
k_npr6_SEM= zeros(N,1);
k_smbnpr6    = zeros(N,1);
k_smbnpr6_SEM= zeros(N,1);
% K_1mPas     = [8.5478; 8.7481; 8.7701; 13.1554; 12.4032;
9.1767; 9.2779];
% K_SEM_1mPas = [0.1743; 0.1936; 0.1969; 0.3115;  0.3022;
0.2447; 0.1639];
th_N2     = zeros(N,1);
th_N2_SEM = zeros(N,1);
th_dop3    = zeros(N,1);
th_dop3_SEM= zeros(N,1);
th_avkdop3    = zeros(N,1);
th_avkdop3_SEM= zeros(N,1);
th_avkTeTx    = zeros(N,1);
th_avkTeTx_SEM= zeros(N,1);
th_flp1    = zeros(N,1);
th_flp1_SEM= zeros(N,1);
th_npr6    = zeros(N,1);
th_npr6_SEM= zeros(N,1);
th_smbnpr6    = zeros(N,1);
th_smbnpr6_SEM= zeros(N,1);

k2_N2      = zeros(N,1); rk_30N2      = zeros(1,1);
rk2_30N2      = zeros(1,1);
k2_dop3    = zeros(N,1); rk_30dop3      = zeros(1,1);
rk2_30dop3      = zeros(1,1);
k2_avkdop3 = zeros(N,1); rk_30avkdop3      = zeros(1,1);
rk2_30avkdop3      = zeros(1,1);
k2_avkTeTx = zeros(N,1); rk_30avkTeTx      = zeros(1,1);
rk2_30avkTeTx      = zeros(1,1);
k2_flp1    = zeros(N,1); rk_30flp1      = zeros(1,1);
rk2_30flp1      = zeros(1,1);
k2_npr6    = zeros(N,1); rk_30npr6      = zeros(1,1);
rk2_30npr6      = zeros(1,1);
k2_smbnpr6 = zeros(N,1); rk_30smbnpr6      = zeros(1,1);
rk2_30smbnpr6      = zeros(1,1);



pname = 'D:\Dropbox\Paper\Compensatory reponse
mechanism\data_gait_adaptation\intermediate';
Name_n2 = 'N2';
Name_dop3 = 'LX703';
Name_avkdop3 = 'ZX2201';
Name_avkTeTx = 'FQ2747';
Name_flp1 = 'PS8997';
Name_npr6 = 'ZX2037U';
Name_smbnpr6 = 'ZX2037R';
Genotypes = {'N2','dop-3','AVK:dop-3(+)','AVK::TeTx','flp-
1','npr-6','SMB:npr-6(+)'};
for i = 1 : N % N2
    filename_n2 = [Name_n2 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_n2))
    f_N2(i) = mean(f_data);
    f_N2_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_N2(i) = mean(l_data);
    l_N2_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_N2(i)   = mean(curv_data,'omitnan');
    k2_N2(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30N2  = mean(1./curv_data,'omitnan');
        rk2_30N2  = mean(1./curv_data.^2,'omitnan');
    end
    k_N2_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_N2(i) = mean(th_data, 'omitnan');
```

```
    th_N2_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % dop-3
    filename_dop3 = [Name_dop3 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_dop3))
    f_dop3(i) = mean(f_data);
    f_dop3_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_dop3(i) = mean(l_data);
    l_dop3_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_dop3(i)    = mean(curv_data,'omitnan');
    k2_dop3(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30dop3  = mean(1./curv_data,'omitnan');
        rk2_30dop3  = mean(1./curv_data.^2,'omitnan');
    end
    k_dop3_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_dop3(i) = mean(th_data,'omitnan');
    th_dop3_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % avk::dop-3
    filename_avkdop3 = [Name_avkdop3 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_avkdop3))
    f_avkdop3(i) = mean(f_data);
    f_avkdop3_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_avkdop3(i) = mean(l_data);
    l_avkdop3_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_avkdop3(i)    = mean(curv_data,'omitnan');
    k2_avkdop3(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30avkdop3  = mean(1./curv_data,'omitnan');
        rk2_30avkdop3  = mean(1./curv_data.^2,'omitnan');
    end
    k_avkdop3_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_avkdop3(i) = mean(th_data,'omitnan');
    th_avkdop3_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % avk::TeTx
    filename_avkTeTx = [Name_avkTeTx '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_avkTeTx))
    f_avkTeTx(i) = mean(f_data);
    f_avkTeTx_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_avkTeTx(i) = mean(l_data);
    l_avkTeTx_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_avkTeTx(i)    = mean(curv_data,'omitnan');
    k2_avkTeTx(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30avkTeTx  = mean(1./curv_data,'omitnan');
        rk2_30avkTeTx  = mean(1./curv_data.^2,'omitnan');
    end
    k_avkTeTx_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_avkTeTx(i) = mean(th_data,'omitnan');
    th_avkTeTx_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % flp-1
    filename_flp1 = [Name_flp1 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_flp1))
    f_flp1(i) = mean(f_data);
    f_flp1_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_flp1(i) = mean(l_data);
    l_flp1_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_flp1(i)    = mean(curv_data,'omitnan');
    k2_flp1(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30flp1  = mean(1./curv_data,'omitnan');
        rk2_30flp1  = mean(1./curv_data.^2,'omitnan');
    end
    k_flp1_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_flp1(i) = mean(th_data,'omitnan');
    th_flp1_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % npr-6
    filename_npr6 = [Name_npr6 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_npr6))
    f_npr6(i) = mean(f_data);
    f_npr6_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_npr6(i) = mean(l_data);
    l_npr6_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_npr6(i)    = mean(curv_data,'omitnan');
    k2_npr6(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30npr6  = mean(1./curv_data,'omitnan');
        rk2_30npr6  = mean(1./curv_data.^2,'omitnan');
    end
```

```
    k_npr6_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_npr6(i) = mean(th_data,'omitnan');
    th_npr6_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N % smb::npr-6
    filename_smbnpr6 = [Name_smbnpr6 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_smbnpr6))
    f_smbnpr6(i) = mean(f_data);
    f_smbnpr6_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_smbnpr6(i) = mean(l_data);
    l_smbnpr6_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_smbnpr6(i)    = mean(curv_data,'omitnan');
    k2_smbnpr6(i)  = mean(curv_data.^2,'omitnan');
    if i==1
        rk_30smbnpr6  = mean(1./curv_data,'omitnan');
        rk2_30smbnpr6  = mean(1./curv_data.^2,'omitnan');
    end
    k_smbnpr6_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_smbnpr6(i) = mean(th_data,'omitnan');
    th_smbnpr6_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
figure(1);clf
F = cat(1, f_N2', f_dop3', f_avkdop3', f_avkTeTx', f_flp1',
f_npr6', f_smbnpr6');
F_SEM = cat(1, f_N2_SEM', f_dop3_SEM', f_avkdop3_SEM',
f_avkTeTx_SEM',...
    f_flp1_SEM', f_npr6_SEM', f_smbnpr6_SEM');
b = bar(F, 'grouped');
hold on
[ngroups, nbars] = size(F);
% Get the x coordinate of the bars
x = nan(nbars, ngroups);
for i = 1:nbars
    x(i,:) = b(i).XEndPoints;
end
% Plot the errorbars
errorbar(x',F,F_SEM,'k','linestyle','none');
hold off
ylabel('f (Hz)')
set(gca, 'XTickLabel',Genotypes)
legend({'1300', '10000'},'Location','northeast')


figure(2);clf
L = cat(1, l_N2', l_dop3', l_avkdop3', l_avkTeTx', l_flp1',
l_npr6', l_smbnpr6');
L_SEM = cat(1, l_N2_SEM', l_dop3_SEM', l_avkdop3_SEM',
l_avkTeTx_SEM',...
    l_flp1_SEM', l_npr6_SEM', l_smbnpr6_SEM');
b = bar(L, 'grouped');
hold on
[ngroups, nbars] = size(L);
% Get the x coordinate of the bars
x = nan(nbars, ngroups);
for i = 1:nbars
    x(i,:) = b(i).XEndPoints;
end
% Plot the errorbars
errorbar(x',L,L_SEM,'k','linestyle','none');
hold off
ylabel('\lambda/L')
set(gca, 'XTickLabel',Genotypes)
legend({'1300', '10000'},'Location','northeast')


figure(3);clf
K = cat(1, k_N2', k_dop3', k_avkdop3', k_avkTeTx', k_flp1',
k_npr6', k_smbnpr6');
% K = cat(2, K_1mPas, K);
K_SEM = cat(1, k_N2_SEM', k_dop3_SEM', k_avkdop3_SEM',
k_avkTeTx_SEM',...
    k_flp1_SEM', k_npr6_SEM', k_smbnpr6_SEM');
% K_SEM = cat(2, K_SEM_1mPas, K_SEM);
V = K(:,1);
V_SEM = K_SEM(:,1);
b = bar(V, 'grouped');
hold on
[ngroups, nbars] = size(V);
% Get the x coordinate of the bars
x = nan(nbars, ngroups);
for i = 1:nbars
    x(i,:) = b(i).XEndPoints;
end
% Plot the errorbars
errorbar(x',V,V_SEM,'k','linestyle','none');
hold off
ylabel('K')
set(gca, 'XTickLabel',Genotypes)
% legend({'1300', '10000'},'Location','northeast')
```

```
figure(4);clf
T = cat(1, th_N2', th_dop3', th_avkdop3', th_avkTeTx',
th_flp1', th_npr6', th_smbnpr6');
T_SEM = cat(1, th_N2_SEM', th_dop3_SEM', th_avkdop3_SEM',
th_avkTeTx_SEM',...
    th_flp1_SEM', th_npr6_SEM', th_smbnpr6_SEM');
b = bar(T, 'grouped');
hold on
[ngroups, nbars] = size(T);
% Get the x coordinate of the bars
x = nan(nbars, ngroups);
for i = 1:nbars
    x(i,:) = b(i).XEndPoints;
end
% Plot the errorbars
errorbar(x',T,T_SEM,'k','linestyle','none');
hold off
ylabel('Angle of attack, (deg)')
set(gca, 'XTickLabel',Genotypes)
legend({'1300', '10000'},'Location','northeast')

figure(5); clf % relative change of curvature
DK = (K(:,2) - K(:,1))./K(:,1);
K2_30 = cat(1, k2_N2(1)', k2_dop3(1)', k2_avkdop3(1)',
k2_avkTeTx(1)',...
    k2_flp1(1)', k2_npr6(1)', k2_smbnpr6(1)');
K2_40 = cat(1, k2_N2(2)', k2_dop3(2)', k2_avkdop3(2)',
k2_avkTeTx(2)',...
    k2_flp1(2)', k2_npr6(2)', k2_smbnpr6(2)');
rK2_30 = cat(1, rk2_30N2', rk2_30dop3', rk2_30avkdop3',
rk2_30avkTeTx',...
    rk2_30flp1', rk2_30npr6', rk2_30smbnpr6');
rK_30 = cat(1, rk_30N2', rk_30dop3', rk_30avkdop3',
rk_30avkTeTx',...
    rk_30flp1', rk_30npr6', rk_30smbnpr6');
DK_SEM = ((K(:,1) - K(:,2)).^2./(K(:,2).^2)...
    .*((K_SEM(:,1).^2+K_SEM(:,2).^2)./(K(:,1) - K(:,2)).^2
+ K_SEM(:,1).^2./K(:,2).^2)).^.5;
% DK_SEM = ((K2_40 + K2_30 - 2*K(:,1).*K(:,2)).*rK2_30 -...
%     (K(:,1) - K(:,2)).^2.*rK_30.^2).^.5;
b = bar(DK);
hold on
% Plot the errorbars
errorbar(1:numel(Genotypes),DK,DK_SEM,'k','linestyle','none
');
hold off
ylabel('Adaptation Index, \Delta K/K_{low vis}')
set(gca, 'XTickLabel',Genotypes)
%%
figure(12); clf
subplot(311)
load('N2_40pct.mat')
h1 = histogram(l_data,'Normalization',
'pdf','BinWidth',0.05);
xlim([0 3])
subplot(312)
load('LX703_40pct.mat')
h2 = histogram(l_data,'Normalization',
'pdf','BinWidth',0.05);
xlim([0 3])
subplot(313)
load('ZX2201_40pct.mat')
h3 = histogram(l_data,'Normalization',
'pdf','BinWidth',0.05);
xlim([0 3])
%

%% Gaussian mixture model
figure(13); clf
load('N2_45pct.mat')
[l,vi] = ksdensity(l_data, 'Kernel','epanechnikov');
plot(vi,l); xlim([0 3])
hold on
load('LX703_45pct.mat')
[l,vi] = ksdensity(l_data, 'Kernel','epanechnikov');
plot(vi,l);
load('ZX2201_45pct.mat')
[l,vi] = ksdensity(l_data, 'Kernel','epanechnikov');
plot(vi,l);
hold off
legend({'N2','dop-3','AVK::dop-3'},'Location','southeast')
```

N2_dop3_rescue.m

```
% close all;
clear; clc
Vis     = [9.2 121 1390 9079 27900]'; % Viscosity  (mPa¬∑s)
pct     = {'05','15','30','40','45'};
N       = numel(Vis);

f_N2     = zeros(N,1);
f_N2_SEM = zeros(N,1);
f_dop    = zeros(N,1);
f_dop_SEM= zeros(N,1);
f_avk    = zeros(N,1);
f_avk_SEM= zeros(N,1);

l_N2     = zeros(N,1);
l_N2_SEM = zeros(N,1);
l_dop    = zeros(N,1);
l_dop_SEM= zeros(N,1);
l_avk    = zeros(N,1);
l_avk_SEM= zeros(N,1);

k_N2     = zeros(N,1);
k_N2_SEM = zeros(N,1);
k_dop    = zeros(N,1);
k_dop_SEM= zeros(N,1);
k_avk    = zeros(N,1);
k_avk_SEM= zeros(N,1);

th_N2     = zeros(N,1);
th_N2_SEM = zeros(N,1);
th_dop    = zeros(N,1);
th_dop_SEM= zeros(N,1);
th_avk    = zeros(N,1);
th_avk_SEM= zeros(N,1);

pname = 'C:\Users\fffei\Dropbox\Paper\Compensatory reponse
mechanism\data_gait_adaptation\intermediate';
Name_n2 = 'N2';
Name_dop = 'LX703';
Name_avk = 'ZX2201';
for i = 1 : N
    filename_n2 = [Name_n2 '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_n2))
    f_N2(i) = mean(f_data);
    f_N2_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_N2(i) = mean(l_data);
    l_N2_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_N2(i)   = mean(curv_data,'omitnan');
    k_N2_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_N2(i) = mean(th_data, 'omitnan');
    th_N2_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N
    filename_dop = [Name_dop '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_dop))
    f_dop(i) = mean(f_data);
    f_dop_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_dop(i) = mean(l_data);
    l_dop_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_dop(i)    = mean(curv_data,'omitnan');
    k_dop_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_dop(i) = mean(th_data,'omitnan');
    th_dop_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
for i = 1 : N
    filename_avk = [Name_avk '_' pct{i} 'pct.mat'];
    load(fullfile(pname, filename_avk))
    f_avk(i) = mean(f_data);
    f_avk_SEM(i) = std(f_data)/sqrt(numel(f_data));
    l_avk(i) = mean(l_data);
    l_avk_SEM(i) = std(l_data)/sqrt(numel(l_data));
    k_avk(i)    = mean(curv_data,'omitnan');
    k_avk_SEM(i) =
std(curv_data,'omitnan')/sqrt(numel(curv_data));
    th_avk(i) = mean(th_data,'omitnan');
    th_avk_SEM(i) =
std(th_data,'omitnan')/sqrt(numel(th_data));
end
figure(1);clf
errorbar(Vis, f_N2, f_N2_SEM, '-o','MarkerSize',10)
hold on
errorbar(Vis, f_dop, f_dop_SEM, '-o','MarkerSize',10)
hold off
hold on
errorbar(Vis, f_avk, f_avk_SEM, '-o','MarkerSize',10)
hold off
ylim([0 2])
xlim([3 10^5])
xlabel('Viscosity (mPa¬∑s)')
ylabel('f (Hz)')
set(gca, 'XScale', 'log', 'XTick',[1 10 100 1000 10000
100000],'FontSize', 18, 'Box', 'off')
legend({'N2','dop-3','AVK::dop-3'},'Location','northeast')


figure(2);clf
errorbar(Vis, l_N2, l_N2_SEM, '-o','MarkerSize',10)
hold on
errorbar(Vis, l_dop, l_dop_SEM, '-o','MarkerSize',10)
```

```
hold off
hold on
errorbar(Vis, l_avk, l_avk_SEM, '-o','MarkerSize',10)
hold off
ylim([0.5 2])
xlim([3 10^5])
xlabel('Viscosity (mPa¬∑s)')
ylabel('\lambda/L')
set(gca, 'XScale', 'log', 'XTick',[1 10 100 1000 10000
100000], 'FontSize', 18, 'Box', 'off')
legend({'N2','dop-3','AVK::dop-3'},'Location','northeast')

figure(3);clf
errorbar(Vis, k_N2, k_N2_SEM, '-o','MarkerSize',10)
hold on
errorbar(Vis, k_dop, k_dop_SEM, '-o','MarkerSize',10)
hold off
hold on
errorbar(Vis, k_avk, k_avk_SEM, '-o','MarkerSize',10)
hold off
ylim([0 10])
xlim([3 10^5])
xlabel('Viscosity (mPa¬∑s)')
ylabel('K')
set(gca, 'XScale', 'log', 'XTick',[1 10 100 1000 10000
100000], 'FontSize', 18, 'Box', 'off')
%%
legend({'N2','dop-3','AVK::dop-3'},'Location','southeast')

figure(4);clf
errorbar(Vis, th_N2, th_N2_SEM, '-o','MarkerSize',10)
hold on
errorbar(Vis, th_dop, th_dop_SEM, '-o','MarkerSize',10)
hold off
hold on
errorbar(Vis, th_avk, th_avk_SEM, '-o','MarkerSize',10)
hold off
ylim([0 60])
xlim([3 10^5])
xlabel('Viscosity (mPa¬∑s)')
ylabel('Angle of attack, (deg)')
set(gca, 'XScale', 'log', 'XTick',[1 10 100 1000 10000
100000], 'FontSize', 18, 'Box', 'off')
legend({'N2','dop-3','AVK::dop-3'},'Location','southeast')

%% Gaussian mixture model
load('N2_45pct.mat')
k_data_N2 = curv_data;
load('ZX2201_45pct.mat')
k_data_dop3 = curv_data;
[h,p,ci,stats] = ttest2(k_data_N2,k_data_dop3)
```

BIBLIOGRAPHY

Acebrón, J.A., Bonilla, L.L., Vicente, C.J.P., Ritort, F., and Spigler, R. (2005). The Kuramoto model: A simple paradigm for synchronization phenomena. Reviews of Modern Physics *77*, 137.

Ache, J.M., and Matheson, T. (2013). Passive joint forces are tuned to limb use in insects and drive movements without motor activity. Current Biology *23*, 1418–1426.

Akitake, B., Ren, Q., Boiko, N., Ni, J., Sokabe, T., Stockand, J.D., Eaton, B.A., and Montell, C. (2015). Coordination and fine motor control depend on Drosophila TRPγ. Nature Communications *6*, 1–13.

Alexander, R.M. (2013). Principles of animal locomotion (Princeton University Press).

Alvarez, F.J., Jonas, P.C., Sapir, T., Hartley, R., Berrocal, M.C., Geiman, E.J., Todd, A.J., and Goulding, M. (2005). Postnatal phenotype and localization of spinal cord V1 derived interneurons. Journal of Comparative Neurology *493*, 177–192.

Andersson, O., Forssberg, H., Grillner, S., and Wallen, P. (1981). Peripheral feedback mechanisms acting on the central pattern generators for locomotion in fish and cat. Canadian Journal of Physiology and Pharmacology *59*, 713–726.

Bamber, B.A., Beg, A.A., Twyman, R.E., and Jorgensen, E.M. (1999). The Caenorhabditis elegans unc-49 locus encodes multiple subunits of a heteromultimeric GABA receptor. Journal of Neuroscience *19*, 5348–5359.

Bargmann, C.I. (1998). Neurobiology of the Caenorhabditis elegans genome. Science *282*, 2028–2033.

Bargmann, C.I. (2012). Beyond the connectome: how neuromodulators shape neural circuits.

Bioessays *34*, 458–465.

Bässler, U. (1977). Sensory control of leg movement in the stick insect Carausius morosus. Biological Cybernetics *25*, 61–72.

Berri, S., Boyle, J.H., Tassieri, M., Hope, I.A., and Cohen, N. (2009). Forward locomotion of the nematode C. elegans is achieved through modulation of a single gait. HFSP Journal *3*, 186–193.

Bidaye, S.S., Bockemühl, T., and Büschges, A. (2018). Six-legged walking in insects: how CPGs, peripheral feedback, and descending signals generate coordinated and adaptive motor rhythms. Journal of Neurophysiology *119*, 459–475.

Blight, A.R. (1977). The Muscular Control of Vertebrate Swimming Movements. Biological Reviews *52*, 181–218.

Borgmann, A., Hooper, S.L., and Büschges, A. (2009). Sensory feedback induced by front-leg stepping entrains the activity of central pattern generators in caudal segments of the stick insect walking system. Journal of Neuroscience *29*, 2972–2983.

Bourane, S., Duan, B., Koch, S.C., Dalet, A., Britz, O., Garcia-Campmany, L., Kim, E., Cheng, L., Ghosh, A., Ma, Q., et al. (2015a). Gate control of mechanical itch by a subpopulation of spinal cord interneurons. Science *350*, 550–554.

Bourane, S., Grossmann, K.S., Britz, O., Dalet, A., Del Barrio, M.G., Stam, F.J., Garcia-Campmany, L., Koch, S., and Goulding, M. (2015b). Identification of a Spinal Circuit for Light Touch and Fine Motor Control. Cell *160*, 503–515.

Bowman, W.C. (2006). Neuromuscular block. British Journal of Pharmacology *147*, S277–S286.

Bowtell, G., and Williams, T. (1991). Anguilliform body dynamics: modelling the interaction

between muscle activation and body curvature. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences *334*, 385–390.

Boyle, J.H., Berri, S., and Cohen, N. (2012). Gait modulation in C. elegans: an integrated neuromechanical model. Frontiers in Computational Neuroscience *6*, 10.

Brenner, S. (1974). The genetics of Caenorhabditis elegans. Genetics *77*, 71–94.

Brodfuehrer, P.D., and Friesen, W.O. (1986). From stimulation to undulation: a neuronal pathway for the control of swimming in the leech. Science *234*, 1002–1004.

Brown, T.G. (1911). The intrinsic factors in the act of progression in the mammal. Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character *84*, 308–319.

Bryden, J., and Cohen, N. (2008). Neural control of Caenorhabditis elegans forward locomotion: the role of sensory feedback. Biological Cybernetics *98*, 339–351.

Buchanan, J.T., and Grillner, S. (1987). Newly identified' glutamate interneurons' and their role in locomotion in the lamprey spinal cord. Science *236*, 312–314.

Bui, T.V., Akay, T., Loubani, O., Hnasko, T.S., Jessell, T.M., and Brownstone, R.M. (2013). Circuits for Grasping: Spinal dI3 Interneurons Mediate Cutaneous Control of Motor Behavior. Neuron *78*, 191–204.

Büschges, A., and Mantziaris, C. (2021). Proprioception: Blurring the boundaries of central and peripheral control. Current Biology *31*, R444–R445.

Butler, V.J., Branicky, R., Yemini, E., Liewald, J.F., Gottschalk, A., Kerr, R.A., Chklovskii, D.B., and Schafer, W.R. (2015). A consistent muscle activation strategy underlies crawling and swimming in Caenorhabditis elegans. Journal of the Royal Society Interface *12*, 20140963.

Calhoun, A.J., Tong, A., Pokala, N., Fitzpatrick, J.A., Sharpee, T.O., and Chalasani, S.H. (2015). Neural mechanisms for evaluating environmental variability in Caenorhabditis elegans. Neuron *86*, 428–441.

Cang, J., and Friesen, W.O. (2000). Sensory modification of leech swimming: rhythmic activity of ventral stretch receptors can change intersegmental phase relationships. Journal of Neuroscience *20*, 7822–7829.

Cang, J., Yu, X., and Friesen, O.W. (2001). Sensory modification of leech swimming: interactions between ventral stretch receptors and swim-related neurons. Journal of Comparative Physiology A *187*, 569–579.

Cermak, N., Stephanie, K.Y., Clark, R., Huang, Y.-C., Baskoylu, S.N., and Flavell, S.W. (2020). Whole-organism behavioral profiling reveals a role for dopamine in state-dependent motor program coupling in C. elegans. Elife *9*, e57093.

Chalfie, M., Sulston, J.E., White, J.G., Southgate, E., Thomson, J.N., and Brenner, S. (1985). The neural circuit for touch sensitivity in Caenorhabditis elegans. Journal of Neuroscience *5*, 956–964.

Chalfie, M., White, J., and Wood, W.B. (1988). The nematode Caenorhabditis elegans (New York: Cold Spring Harbor Laboratory Press).

Chase, D.L., Pepper, J.S., and Koelle, M.R. (2004). Mechanism of extrasynaptic dopamine signaling in Caenorhabditis elegans. Nature Neuroscience *7*, 1096–1103.

Chen, J., Friesen, W.O., and Iwasaki, T. (2011). Mechanisms underlying rhythmic locomotion: body–fluid interaction in undulatory swimming. Journal of Experimental Biology *214*, 561–574.

Chronis, N., Zimmer, M., and Bargmann, C.I. (2007). Microfluidics for in vivo imaging of neuronal and behavioral activity in Caenorhabditis elegans. Nature Methods *4*, 727–731.

Churgin, M.A., McCloskey, R.J., Peters, E., and Fang-Yen, C. (2017). Antagonistic serotonergic and octopaminergic neural circuits mediate food-dependent locomotory behavior in Caenorhabditis elegans. Journal of Neuroscience *37*, 7811–7823.

Cohen, A.H., and Wallén, P. (1980). The neuronal correlate of locomotion in fish. Experimental Brain Research *41*, 11–18.

Cohen, N., and Denham, J.E. (2019). Whole animal modeling: piecing together nematode locomotion. Current Opinion in Systems Biology *13*, 150–160.

Cook, S.J., Jarrell, T.A., Brittin, C.A., Wang, Y., Bloniarz, A.E., Yakovlev, M.A., Nguyen, K.C., Tang, L.T.-H., Bayer, E.A., and Duerr, J.S. (2019). Whole-animal connectomes of both Caenorhabditis elegans sexes. Nature *571*, 63–71.

Croll, N.A. (1970). The behaviour of nematodes: their activity, senses and responses. The Behaviour of Nematodes: Their Activity, Senses and Responses.

Cruse, H. (1976). The control of body position in the stick insect (Carausius morosus), when walking over uneven surfaces. Biological Cybernetics *24*, 25–33.

Delcomyn, F. (1980). Neural basis of rhythmic behavior in animals. Science *210*, 492–498.

Deng, L., Denham, J., Arya, C., Yuval, O., Cohen, N., and Haspel, G. (2020). Inhibition underlies fast undulatory locomotion in C. elegans. BioRxiv.

Denham, J.E., Ranner, T., and Cohen, N. (2018). Signatures of proprioceptive control in Caenorhabditis elegans locomotion. Philosophical Transactions of the Royal Society B: Biological Sciences *373*, 20180208.

Dickinson, M.H., Farley, C.T., Full, R.J., Koehl, M.A.R., Kram, R., and Lehman, S. (2000). How

animals move: an integrative view. Science *288*, 100–106.

Dietz, V. (2002). Proprioception and locomotor disorders. Nature Reviews Neuroscience *3*, 781–790.

Dietz, V., Quintern, J., and Sillem, M. (1987). Stumbling reactions in man: significance of proprioceptive and pre-programmed mechanisms. The Journal of Physiology *386*, 149–163.

Dong, X., Kheiri, S., Lu, Y., Xu, Z., Zhen, M., and Liu, X. (2021). Toward a living soft microrobot through optogenetic locomotion control of Caenorhabditis elegans. Science Robotics *6*, eabe3950.

Donnelly, J.L., Clark, C.M., Leifer, A.M., Pirri, J.K., Haburcak, M., Francis, M.M., Samuel, A.D., and Alkema, M.J. (2013). Monoaminergic orchestration of motor programs in a complex C. elegans behavior. PLoS Biology *11*, e1001529.

Driscoll, M., and Kaplan, J. (1997). Mechanotransduction.

Eastwood, A.L., Sanzeni, A., Petzold, B.C., Park, S.-J., Vergassola, M., Pruitt, B.L., and Goodman, M.B. (2015). Tissue mechanics govern the rapidly adapting and symmetrical response to touch. Proceedings of the National Academy of Sciences *112*, E6955–E6963.

Ekeberg, Ö., and Grillner, S. (1999). Simulations of neuromuscular control in lamprey swimming. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences *354*, 895–902.

Fang-Yen, C., Wyart, M., Xie, J., Kawai, R., Kodger, T., Chen, S., Wen, Q., and Samuel, A.D. (2010). Biomechanical analysis of gait adaptation in the nematode Caenorhabditis elegans. Proceedings of the National Academy of Sciences *107*, 20323–20328.

Fisher, N.I., Lewis, T., and Embleton, B.J. (1993). Statistical analysis of spherical data (Cambridge university press).

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. Biophysical Journal *1*, 445.

Flavell, S.W., Pokala, N., Macosko, E.Z., Albrecht, D.R., Larsch, J., and Bargmann, C.I. (2013). Serotonin and the neuropeptide PDF initiate and extend opposing behavioral states in C. elegans. Cell *154*, 1023–1035.

Forssberg, H. (1979). Stumbling corrective reaction: a phase-dependent compensatory reaction during locomotion. J Neurophysiol *42*, 936–953.

Forssberg, H., Grillner, S., and Rossignol, S. (1975). Phase dependent reflex reversal during walking in chronic spinal cats. Brain Research *85*, 103–107.

Forssberg, H., Grillner, S., and Rossignol, S. (1977). Phasic gain control of reflexes from the dorsum of the paw during spinal locomotion. Brain Research *132*, 121–139.

Fouad, A.D., Teng, S., Mark, J.R., Liu, A., Ji, H., Cornblath, E., and Fang-Yen, C. (2017). Distributed rhythm generators underlie. Caenorhabditis Elegans.

Fouad, A.D., Teng, S., Mark, J.R., Liu, A., Alvarez-Illera, P., Ji, H., Du, A., Bhirgoo, P.D., Cornblath, E., and Guan, S.A. (2018). Distributed rhythm generators underlie Caenorhabditis elegans forward locomotion. Elife *7*, e29913.

Fouad, A.D., Liu, A., Du, A., Bhirgoo, P.D., and Fang-Yen, C. (2021). Thermal laser ablation with tunable lesion size reveals multiple origins of seizure-like convulsions in Caenorhabditis elegans. Scientific Reports *11*, 1–9.

Friesen, W.O. (2009). Central Pattern Generators: Sensory Feedback. Encyclopedia of Neuroscience.

Gao, S., Guan, S.A., Fouad, A.D., Meng, J., Kawano, T., Huang, Y.-C., Li, Y., Alcaire, S., Hung, W., and Lu, Y. (2018). Excitatory motor neurons are local oscillators for backward locomotion. Elife *7*, e29915.

Gjorgjieva, J., Biron, D., and Haspel, G. (2014). Neurobiology of Caenorhabditis elegans locomotion: where do we stand? Bioscience *64*, 476–486.

Goulding, M. (2009). Circuits controlling vertebrate locomotion: moving in a new direction. Nat Rev Neurosci *10*, 507–518.

Graham, D. (1985). Pattern and control of walking in insects. In Advances in Insect Physiology, (Elsevier), pp. 31–140.

Gray, J. (1933). Studies in animal locomotion: I. The movement of fish with special reference to the eel. Journal of Experimental Biology *10*, 88–104.

Gray, J., and Lissmann, H.W. (1964). The locomotion of nematodes. Journal of Experimental Biology *41*, 135–154.

Gray, J.M., Hill, J.J., and Bargmann, C.I. (2005). A circuit for navigation in Caenorhabditis elegans. Proceedings of the National Academy of Sciences *102*, 3184–3191.

Grillner, S. (2003). The motor infrastructure: from ion channels to neuronal networks. Nature Reviews Neuroscience *4*, 573–586.

Grillner, S. (2006). Biological pattern generation: the cellular and computational logic of networks in motion. Neuron *52*, 751–766.

Grillner, S. (2021). The execution of movement: A spinal affair. Journal of Neurophysiology *125*, 693–698.

Grillner, S., and El Manira, A. (2020). Current principles of motor control, with special reference to vertebrate locomotion. Physiological Reviews *100*, 271–320.

Grillner, S., and Jessell, T.M. (2009). Measured motion: searching for simplicity in spinal locomotor networks. Current Opinion in Neurobiology *19*, 572–586.

Grillner, S., and Wallen, P. (2002). Cellular bases of a vertebrate locomotor system–steering, intersegmental and segmental co-ordination and sensory control. Brain Research Reviews *40*, 92–106.

Grillner, S., McClellan, A., and Perret, C. (1981). Entrainment of the spinal pattern generators for swimming by mechano-sensitive elements in the lamprey spinal cord in vitro. Brain Research *217*, 380–386.

Grillner, S., Hellgren, J., Menard, A., Saitoh, K., and Wikström, M.A. (2005). Mechanisms for selection of basic motor programs–roles for the striatum and pallidum. Trends in Neurosciences *28*, 364–370.

Guo, Z.V., and Mahadevan, L. (2008). Limbless undulatory propulsion on land. Proceedings of the National Academy of Sciences *105*, 3179–3184.

Gutkin, B.S., Ermentrout, G.B., and Reyes, A.D. (2005). Phase-response curves give the responses of neurons to transient inputs. Journal of Neurophysiology *94*, 1623–1635.

Han, B., Dong, Y., Zhang, L., Liu, Y., Rabinowitch, I., and Bai, J. (2017). Dopamine signaling tunes spatial pattern selectivity in C. elegans. Elife *6*, e22896.

Hansen, P.D., Woollacott, M.H., and Debu, B. (1988). Postural responses to changing task conditions. Experimental Brain Research *73*, 627–636.

Haspel, G., O'Donovan, M.J., and Hart, A.C. (2010). Motoneurons dedicated to either forward or backward locomotion in the nematode Caenorhabditis elegans. Journal of Neuroscience *30*, 11151–11156.

Haspel, G., Deng, L., Harreguy, M.B., and Tanvir, Z. (2020). Elegantly. In The Neural Control of Movement, (Elsevier), pp. 3–29.

Hilde, K.L., Levine, A.J., Hinckley, C.A., Hayashi, M., Montgomery, J.M., Gullo, M., Driscoll, S.P., Grosschedl, R., Kohwi, Y., Kohwi-Shigematsu, T., et al. (2016). Satb2 Is Required for the Development of a Spinal Exteroceptive Microcircuit that Modulates Limb Position. Neuron *91*, 763–776.

Hills, T., Brockie, P.J., and Maricq, A.V. (2004). Dopamine and glutamate control area-restricted search behavior in Caenorhabditis elegans. Journal of Neuroscience *24*, 1217–1225.

Hobert, O. (2003). Behavioral plasticity in C. elegans: paradigms, circuits, genes. Journal of Neurobiology *54*, 203–223.

Hu, Z., Pym, E.C., Babu, K., Murray, A.B.V., and Kaplan, J.M. (2011). A neuropeptide-mediated stretch response links muscle contraction to changes in neurotransmitter release. Neuron *71*, 92–102.

Hums, I., Riedl, J., Mende, F., Kato, S., Kaplan, H.S., Latham, R., Sonntag, M., Traunmüller, L., and Zimmer, M. (2016). Regulation of two motor patterns enables the gradual adjustment of locomotion strategy in Caenorhabditis elegans. Elife *5*, e14116.

Iwasaki, T., Chen, J., and Friesen, W.O. (2014). Biological clockwork underlying adaptive

rhythmic movements. Proceedings of the National Academy of Sciences *111*, 978–983.

Izhikevich, E.M. (2007). Dynamical systems in neuroscience (MIT press).

Izquierdo, E.J., and Beer, R.D. (2018). From head to tail: a neuromechanical model of forward locomotion in Caenorhabditis elegans. Philosophical Transactions of the Royal Society B: Biological Sciences *373*, 20170374.

Ji, H., Fouad, A.D., Teng, S., Liu, A., Alvarez-Illera, P., Yao, B., Li, Z., and Fang-Yen, C. (2021a). Phase response analyses support a relaxation oscillator model of locomotor rhythm generation in Caenorhabditis elegans. Elife *10*, e69905.

Ji, H., Fouad, A.D., Teng, S., Liu, A., Alvarez-Illera, P., Yao, B., Li, Z., and Fang-Yen, C. (2021b). Phase response analyses support a relaxation oscillator model of locomotor rhythm generation in Caenorhabditis elegans. ELife *10*, e69905.

Ji, H., Fouad, A., Teng, S., Liu, A., Alvarez-Illera, P., Yao, B., Li, Z., and Fang-Yen, C. (2021c). Phase response analyses support a relaxation oscillator model of locomotor rhythm generation in Caenorhabditis elegans (Dryad).

Johnson, C.L., Lewis, T.J., and Guy, R. (2021). Neuromechanical Mechanisms of Gait Adaptation in C. elegans: Relative Roles of Neural and Mechanical Coupling. SIAM Journal on Applied Dynamical Systems *20*, 1022–1052.

Kaplan, H.S., Thula, O.S., Khoss, N., and Zimmer, M. (2020). Nested neuronal dynamics orchestrate a behavioral hierarchy across timescales. Neuron *105*, 562–576.

Karbowski, J., Schindelman, G., Cronin, C.J., Seah, A., and Sternberg, P.W. (2008). Systems level circuit model of C. elegans undulatory locomotion: mathematical modeling and molecular genetics. Journal of Computational Neuroscience *24*, 253–276.

Kato, S., Kaplan, H.S., Schrödel, T., Skora, S., Lindsay, T.H., Yemini, E., Lockery, S., and Zimmer, M. (2015). Global Brain Dynamics Embed the Motor Command Sequence of Caenorhabditis elegans. Cell *163*, 656–669.

Katz, D.F., Blake, J.R., and Paveri-Fontana, S.L. (1975). On the movement of slender bodies near plane boundaries at low Reynolds number. Journal of Fluid Mechanics *72*, 529–540.

Kawano, T., Po, M.D., Gao, S., Leung, G., Ryu, W.S., and Zhen, M. (2011). An imbalancing act: gap junctions reduce the backward motor circuit activity to bias C. elegans for forward locomotion. Neuron *72*, 572–586.

Kiehn, O. (1998). Neuronal mechanisms for generating locomotor activity (New York Academy of Sciences).

Kiehn, O. (2011). Development and functional organization of spinal locomotor circuits. Current Opinion in Neurobiology *21*, 100–109.

Kiehn, O. (2016). Decoding the organization of spinal circuits that control locomotion. Nature Reviews Neuroscience *17*, 224–238.

Kindt, K.S., Quast, K.B., Giles, A.C., De, S., Hendrey, D., Nicastro, I., Rankin, C.H., and Schafer, W.R. (2007). Dopamine mediates context-dependent modulation of sensory plasticity in C. elegans. Neuron *55*, 662–676.

Koch, S.C., Del Barrio, M.G., Dalet, A., Gatto, G., Günther, T., Zhang, J., Seidler, B., Saur, D., Schüle, R., and Goulding, M. (2017). RORβ Spinal Interneurons Gate Sensory Transmission during Locomotion to Secure a Fluid Walking Gait. Neuron *96*, 1419-1431.e5.

Kratsios, P., Stolfi, A., Levine, M., and Hobert, O. (2012). Coordinated regulation of cholinergic motor neuron traits through a conserved terminal selector gene. Nature Neuroscience *15*, 205.

Kristan, W.B., and Calabrese, R.L. (1976). Rhythmic swimming activity in neurones of the isolated nerve cord of the leech. Journal of Experimental Biology *65*, 643–668.

Kunert, J.M., Proctor, J.L., Brunton, S.L., and Kutz, J.N. (2017). Spatiotemporal feedback and network structure drive and encode Caenorhabditis elegans locomotion. PLoS Computational Biology *13*, e1005303.

Leifer, A.M., Fang-Yen, C., Gershow, M., Alkema, M.J., and Samuel, A.D. (2011). Optogenetic manipulation of neural activity in freely moving Caenorhabditis elegans. Nature Methods *8*, 147–152.

Li, W., Feng, Z., Sternberg, P.W., and Xu, X.S. (2006). A C. elegans stretch receptor neuron revealed by a mechanosensitive TRP channel homologue. Nature *440*, 684–687.

Liu, Q., Chen, B., Gaier, E., Joshi, J., and Wang, Z.-W. (2006). Low conductance gap junctions mediate specific electrical coupling in body-wall muscle cells of Caenorhabditis elegans. Journal of Biological Chemistry *281*, 7881–7889.

Lockery, S.R., Lawton, K.J., Doll, J.C., Faumont, S., Coulthard, S.M., Thiele, T.R., Chronis, N., McCormick, K.E., Goodman, M.B., and Pruitt, B.L. (2008). Artificial dirt: microfluidic substrates for nematode neurobiology and behavior. Journal of Neurophysiology *99*, 3136–3143.

LONG, J.H., Jr. (1998). Muscles, Elastic Energy, and the Dynamics of Body Stiffness in Swimming Eels1. American Zoologist *38*, 771–792.

López-Cruz, A., Sordillo, A., Pokala, N., Liu, Q., McGrath, P.T., and Bargmann, C.I. (2019). Parallel multimodal circuits control an innate foraging behavior. Neuron *102*, 407–419.

Marder, E., and Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. Current Biology *11*, R986–R996.

Marder, E., and Calabrese, R.L. (1996). Principles of rhythmic motor pattern generation. Physiological Reviews *76*, 687–717.

Mayer, W.P., and Akay, T. (2018). Stumbling corrective reaction elicited by mechanical and electrical stimulation of the saphenous nerve in walking mice. Journal of Experimental Biology *221*, jeb178095.

McIntire, S.L., Jorgensen, E., Kaplan, J., and Horvitz, H.R. (1993). The GABAergic nervous system of Caenorhabditis elegans. Nature *364*, 337–341.

McVea, D.A., and Pearson, K.G. (2007). Long-lasting, context-dependent modification of stepping in the cat after repeated stumbling-corrective responses. J Neurophysiol *97*, 659–669.

Mendes, C.S., Bartos, I., Akay, T., Márka, S., and Mann, R.S. (2013). Quantification of gait parameters in freely walking wild type and sensory deprived Drosophila melanogaster. Elife *2*, e00231.

Milligan, B., Curtin, N., and Bone, Q. (1997). Contractile properties of obliquely striated muscle from the mantle of squid (Alloteuthis subulata) and cuttlefish (Sepia officinalis). The Journal of Experimental Biology *200*, 2425–2436.

Morris, C., and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. Biophysical Journal *35*, 193–213.

Mullins, O.J., Hackett, J.T., Buchanan, J.T., and Friesen, W.O. (2011). Neuronal control of swimming behavior: comparison of vertebrate and invertebrate model systems. Progress in Neurobiology *93*, 244–269.

Nadim, F., Zhao, S., and Bose, A. (2012). A PRC description of how inhibitory feedback promotes oscillation stability. In Phase Response Curves in Neuroscience, (Springer), pp. 399–417.

Nagumo, J., Arimoto, S., and Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. Proceedings of the IRE *50*, 2061–2070.

Nave, C.R. (1995). Relaxation oscillator concept.

Nelson, L.S., Rosoff, M.L., and Li, C. (1998). Disruption of a neuropeptide gene, flp-1, causes multiple behavioral defects in Caenorhabditis elegans. Science *281*, 1686–1690.

Niebur, E., and Erdös, P. (1991). Theory of the locomotion of nematodes: dynamics of undulatory progression on a surface. Biophysical Journal *60*, 1132–1146.

O'Hagan, R., Chalfie, M., and Goodman, M.B. (2005). The MEC-4 DEG/ENaC channel of Caenorhabditis elegans touch receptor neurons transduces mechanical signals. Nature Neuroscience *8*, 43–50.

Olivares, E., Izquierdo, E.J., and Beer, R.D. (2021). A neuromechanical model of multiple network rhythmic pattern generators for forward locomotion in C. elegans. Frontiers in Computational Neuroscience *15*, 10.

Oranth, A., Schultheis, C., Tolstenkov, O., Erbguth, K., Nagpal, J., Hain, D., Brauner, M., Wabnig, S., Costa, W.S., and McWhirter, R.D. (2018). Food sensation modulates locomotion by dopamine and neuropeptide signaling in a distributed neuronal network. Neuron *100*, 1414–1428.

Pearce, R.A., and Friesen, W.O. (1984). Intersegmental coordination of leech swimming: comparison of in situ and isolated nerve cord activity with body wall movement. Brain Research *299*, 363–366.

Pearson, K. (2000). Motor systems. Current Opinion in Neurobiology *10*, 649–654.

Pearson, K.G. (2004). Generating the walking gait: role of sensory feedback. In Progress in Brain

Research, (Elsevier), pp. 123–129.

Picton, L.D., Bertuzzi, M., Pallucchi, I., Fontanel, P., Dahlberg, E., Björnfors, E.R., Iacoviello, F., Shearing, P.R., and El Manira, A. (2021). A spinal organ of proprioception for integrated motor action feedback. Neuron *109*, 1188–1201.

Pierce-Shimomura, J.T., Chen, B.L., Mun, J.J., Ho, R., Sarkis, R., and McIntire, S.L. (2008). Genetic analysis of crawling and swimming locomotory patterns in C. elegans. Proceedings of the National Academy of Sciences *105*, 20982–20987.

Pietras, B., and Daffertshofer, A. (2019). Network dynamics of coupled oscillators and phase reduction techniques. Physics Reports.

Potocanac, Z., Pijnappels, M., Verschueren, S., van Dieën, J., and Duysens, J. (2016). Two-stage muscle activity responses in decisions about leg movement adjustments during trip recovery. J Neurophysiol *115*, 143–156.

Prochazka, A., Sontag, K.-H., and Wand, P. (1978). Motor reactions to perturbations of gait: proprioceptive and somesthetic involvement. Neuroscience Letters *7*, 35–39.

Ranner, T. (2020). A stable finite element method for low inertia undulatory locomotion in three dimensions. Applied Numerical Mathematics *156*, 422–445.

Rayleigh, J.W.S.B. (1896). The theory of sound (Macmillan).

Roberts, T.J., and Azizi, E. (2011). Flexible mechanisms: the diverse roles of biological springs in vertebrate movement. Journal of Experimental Biology *214*, 353–361.

Roberts, A., Li, W.-C., and Soffe, S.R. (2010). How neurons generate behaviour in a hatchling amphibian tadpole: an outline. Frontiers in Behavioral Neuroscience *4*, 16.

Roseberry, T.K., Lee, A.M., Lalive, A.L., Wilbrecht, L., Bonci, A., and Kreitzer, A.C. (2016). Cell-type-specific control of brainstem locomotor circuits by basal ganglia. Cell *164*, 526–537.

Rosenblum, M. (2018). Inferring the phase response curve from observation of a continuously perturbed oscillator. Scientific Reports *8*, 1–10.

Rossignol, S. (2006). Dubuc R, Gossard JP. Dynamic Sensorimotor Interactions in Locomotion. Physiol Rev *86*, 89–154.

Rugiero, F., Drew, L.J., and Wood, J.N. (2010). Kinetic properties of mechanically activated currents in spinal sensory neurons. The Journal of Physiology *588*, 301–314.

Sauvage, P. (2007). Etude de la locomotion chez C. elegans et perturbations mecaniques du mouvement. PhD Thesis. PhD dissertation, Laboratoire Matiere et Systemes Compiexes (Paris, France ….

Sawin, E.R., Ranganathan, R., and Horvitz, H.R. (2000). C. elegans locomotory rate is modulated by the environment through a dopaminergic pathway and by experience through a serotonergic pathway. Neuron *26*, 619–631.

Schultheiss, N.W., Prinz, A.A., and Butera, R.J. (2011). Phase response curves in neuroscience: theory, experiment, and analysis (Springer Science & Business Media).

Schwarz, J., and Bringmann, H. (2017). Analysis of the NK2 homeobox gene ceh-24 reveals sublateral motor neuron control of left-right turning during sleep. ELife *6*, e24846.

Shen, Y., Wen, Q., Liu, H., Zhong, C., Qin, Y., Harris, G., Kawano, T., Wu, M., Xu, T., and Samuel, A.D. (2016). An extrasynaptic GABAergic signal modulates a pattern of forward movement in Caenorhabditis elegans. Elife *5*, e14197.

Smeal, R.M., Ermentrout, G.B., and White, J.A. (2010). Phase-response curves and synchronized neural networks. Philosophical Transactions of the Royal Society B: Biological Sciences *365*, 2407–2422.

Smith, J.L., Chung, S.H., and Zernicke, R.F. (1993). Gait-related motor patterns and hindlimb kinetics for the cat trot and gallop. Experimental Brain Research *94*, 308–322.

Stiernagle, T. (2006). Maintenance of C. elegans (February 11, 2006), WormBook, ed. The C. elegans Research Community, WormBook, doi/10.1895/wormbook. 1.101. 1.

Sulston, J., and Hodgkin, J. (1988). The nematode Caenorhabditis elegans.

Sulston, J., Dew, M., and Brenner, S. (1975). Dopaminergic neurons in the nematode Caenorhabditis elegans. Journal of Comparative Neurology *163*, 215–226.

Susoy, V., Hung, W., Witvliet, D., Whitener, J.E., Wu, M., Park, C.F., Graham, B.J., Zhen, M., Venkatachalam, V., and Samuel, A.D. (2021). Natural sensory context drives diverse brain-wide activity during C. elegans mating. Cell *184*, 5122–5137.

Svoboda, K., and Li, N. (2018). Neural mechanisms of movement planning: motor cortex and beyond. Current Opinion in Neurobiology *49*, 33–41.

Tao, L., Porto, D., Li, Z., Fechner, S., Lee, S.A., Goodman, M.B., Xu, X.S., Lu, H., and Shen, K. (2019). Parallel Processing of Two Mechanosensory Modalities by a Single Neuron in C. elegans. Developmental Cell *51*, 617–631.

Tytell, E.D., Carr, J.A., Danos, N., Wagenbach, C., Sullivan, C.M., Kiemel, T., Cowan, N.J., and Ankarali, M.M. (2018). Body stiffness and damping depend sensitively on the timing of muscle activation in lampreys. Integrative and Comparative Biology *58*, 860–873.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2007). Scatter search and local NLP solvers: A multistart framework for global optimization. INFORMS Journal on Computing *19*, 328–340.

Van der Pol, B. (1926). LXXXVIII. On “relaxation-oscillations.” The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science *2*, 978–992.

VAN DER POL, B. (1940). Biological rhythms considered as relaxation oscillations. Acta Medica Scandinavica *103*, 76–88.

Vidal-Gadea, A., Topper, S., Young, L., Crisp, A., Kressin, L., Elbel, E., Maples, T., Brauner, M., Erbguth, K., and Axelrod, A. (2011). Caenorhabditis elegans selects distinct crawling and swimming gaits via dopamine and serotonin. Proceedings of the National Academy of Sciences *108*, 17504–17509.

Von Stetina, S.E., Treinin, M., and Miller, D.M. (2006). The motor circuit. Int. Rev. Neurobiol *69*, 125–167.

Wallace, H.R. (1968). The dynamics of nematode movement. Annual Review of Phytopathology *6*, 91–114.

Wen, Q., Po, M.D., Hulme, E., Chen, S., Liu, X., Kwok, S.W., Gershow, M., Leifer, A.M., Butler, V., and Fang-Yen, C. (2012). Proprioceptive coupling within motor neurons drives C. elegans forward locomotion. Neuron *76*, 750–761.

Wendler, G. (1968). Ein Analogmodell der Beinbewegungen eines laufenden Insekts. Kybernetik *18*, 68–74.

White, J.G., Southgate, E., Thomson, J.N., and Brenner, S. (1986). The structure of the nervous system of the nematode Caenorhabditis elegans. Philos Trans R Soc Lond B Biol Sci *314*, 1–340.

Wilson, D.M. (1961). The central nervous control of flight in a locust. Journal of Experimental Biology *38*, 471–490.

WILSON, D.M., and WEIS-FOGH, T. (1962). Patterned activity of coordinated motor units, studied in flying locusts. Journal of Experimental Biology *39*, 643–667.

Windhorst, U. (2007). Muscle proprioceptive feedback and spinal networks. Brain Research Bulletin *73*, 155–202.

Winfree, A.T. (2001). The geometry of biological time (Springer Science & Business Media).

Wisleder, D., Zernicke, R.F., and Smith, J.L. (1990). Speed-related changes in hindlimb intersegmental dynamics during the swing phase of cat locomotion. Experimental Brain Research *79*, 651–660.

Wolf, H., and Pearson, K.G. (1988). Proprioceptive input patterns elevator activity in the locust flight system. Journal of Neurophysiology *59*, 1831–1853.

Woo, S.-H., Lukacs, V., De Nooij, J.C., Zaytseva, D., Criddle, C.R., Francisco, A., Jessell, T.M., Wilkinson, K.A., and Patapoutian, A. (2015). Piezo2 is the principal mechanotransduction channel for proprioception. Nature Neuroscience *18*, 1756–1762.

Xu, T., Huo, J., Shao, S., Po, M., Kawano, T., Lu, Y., Wu, M., Zhen, M., and Wen, Q. (2018). Descending pathway facilitates undulatory wave propagation in Caenorhabditis elegans through gap junctions. Proceedings of the National Academy of Sciences *115*, E4493–E4502.

Yeon, J., Kim, J., Kim, D.-Y., Kim, H., Kim, J., Du, E.J., Kang, K., Lim, H.-H., Moon, D., and Kim, K. (2018). A sensory-motor neuron type mediates proprioceptive coordination of steering in C. elegans via two TRPC channels. PLoS Biology *16*, e2004929.

Yu, X., and Friesen, W.O. (2004). Entrainment of leech swimming activity by the ventral stretch receptor. Journal of Comparative Physiology A *190*, 939–949.

Yu, X., Nguyen, B., and Friesen, W.O. (1999). Sensory feedback can coordinate the swimming activity of the leech. Journal of Neuroscience *19*, 4634–4643.

Zagoraiou, L., Akay, T., Martin, J.F., Brownstone, R.M., Jessell, T.M., and Miles, G.B. (2009). A Cluster of Cholinergic Premotor Interneurons Modulates Mouse Locomotor Activity. Neuron *64*, 645–662.

Zhang, Y., Narayan, S., Geiman, E., Lanuza, G.M., Velasquez, T., Shanks, B., Akay, T., Dyck, J., Pearson, K., Gosgnach, S., et al. (2008). V3 Spinal Neurons Establish a Robust and Balanced Locomotor Rhythm during Walking. Neuron *60*, 84–96.

Zhen, M., and Samuel, A.D. (2015). C. elegans locomotion: small circuits, complex functions. Current Opinion in Neurobiology *33*, 117–126.